**A Novel Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multitasking Processes in Scientific Service Facilities**

Nikolaos Rakovitis, Nan Zhang, Jie Li[[1]](#footnote-1)

Centre for Process Integration, Department of Chemical Engineering and Analytical Science,

The University of Manchester, Manchester, M13 9PL, United Kingdom

# Abstract

Scientific service facilities examine a number of samples from different customers for several physical and chemical properties using processing units with large capacities. A processing unit can process a great number of samples simultaneously. The process in such scientific service facility can be treated as a multi-tasking multipurpose batch process. Despite the great interest in developing models for scheduling of process industry during the past three decades, scheduling of multi-tasking multipurpose batch processes in a scientific service facility has not been considered adequately. In this work, we develop three novel mathematical models using the well-established unit-specific event-based modelling approach. The computational results demonstrate that the proposed mathematical models are able to reduce the number of event points required, which leads to a significant reduction in the model size and computational time. One of the proposed models in which the timing variables are defined based on processing units is the most efficient in most cases especially when minimization of makespan is used as the objective, where at least one order of magnitude less computational time than all other models is required to generate the optimum solution compared to other existing models.

**Keywords**: Scheduling, multi-tasking, batch process, mixed-integer linear programming, unit-specific event-based approach

# 1 Introduction

Process industries always seek ways to maximize their productivity, minimize their operating cost, and achieve efficient inventory management to survive in a highly competitive market. Scheduling is one of the important managerial tools for such industries to better utilize materials and machines and as a result to increase their profit. However, most of the existing industries use heuristic-based or spreadsheet-based methods which are only limited to generate a feasible solution for simple processes. Therefore, both academic and industrial research focuses on methods that are able to generate optimal schedules in reasonable computational time. Mathematical programming especially mixed-integer programming approach has gained much attention since it can often generate optimal schedules not only for simple processes but also in complicated ones.

 Batch processes are widely used in process industries such as chemicals, pharmaceuticals, food industry, scientific service facilities and iron and steel industry because of their flexibility to produce high valued products, especially if small production of each product is required. Furthermore, they are ideal in cases of seasonal orders by different customers. The batch process is usually classified into single or multi-stage multiproduct batch process and multipurpose batch process (Kopanos and Puigjaner, 2019). In these processes, usually at most one task is allowed to be processed in a processing unit at a time. Scheduling of these processes has received considerable attention in the past three decades (Floudas and Lin, 2004; Méndez *et al.*, 2006; Li *et al.*, 2010; Maravelias, 2012; Harjunkoski *et al.*, 2014). Discrete- and continuous-time modelling approaches have been proposed to develop a great number of mathematical models based on state-task network (Kondili *et al.*, 1993) and resource-task network (Pantelides, 1994). The discrete-time modelling approach divides the scheduling horizon into time intervals of known length where the start and end time of an activity must be exactly at the time interval points. As a result, a great number of time intervals are often required, which significantly increases the model size. The continuous-time modelling approach can be further divided into process-slot (Sundaramoorthy and Karimi, 2005), global event-based (Maravelias and Grossmann, 2003), unit-specific event-based (Ierapetritou and Floudas, 1998; Shaik and Floudas, 2009; Li and Floudas, 2010; Tang et al., 2012; Li et al., 2016), unit-slot (Sursarla et al., 2010; Li and Karimi, 2011) and sequence-based (Méndez and Cerdá, 2000; Hui *et al*., 2000; Méndez and Cerdá, 2003) modelling approaches. The continuous-time modelling approach divides the scheduling horizon into time intervals of unknown length, leading to less time points, batches, slots or event points required compared to the discrete-time modelling approach. The advantages of the unit-specific event-based modelling approach have been well established in the literature (Shaik et al, 2006; Shaik and Floudas, 2009; Li and Floudas, 2010), often requiring less number of event points. The details about these modelling approaches and mathematical models can be found in Floudas and Lin (2004), Méndez *et al.* (2006) and Harjunkoski *et al*. (2014).

 In a scientific service facility, a number of samples from different customers are examined for a number of chemical or/and physical properties. In order to examine such properties, a scientific service facility uses a number of machines with each containing a number of slots. Since these machines contain many slots, it is possible to have samples from different customers that are processed at a time simultaneously in a machine. In other words, multiple tasks can be processed in a machine at a time in such scientific service facilities, which is different from the discussed single-tasking batch processes with at most one task being processed in a unit at a time. Each customer requires a different number of physical and chemical properties to be examined. Therefore, each sample group is examined in different processing units. In other words, different samples can follow different processing paths. The processes in scientific service facilities are considered as multi-tasking multipurpose batch process (Lagzi *et al.*, 2017a). A typical scientific service facility receives around 3000-5000 samples from 40-60 different customers every day (Lagzi *et al.*, 2017a).

 Most mathematical models that have been developed for the batch process with at most one task being processed in a processing unit at a time cannot be directly applied to the multi-tasking multipurpose batch process in scientific service facilities. Few efforts have focused on optimal scheduling of such multi-tasking batch process in scientific service facilities. Patil *et al.* (2015) developed a discrete-time model for scheduling of multi-tasking batch processes in scientific service facilities. Lagzi *et al.* (2017a) used process-slot continuous-time modelling approach for the same scheduling problem. Lagzi *et al.* (2017b) developed a discrete-time formulation using non-uniform time grid based on the work of xx and compared the performance with that of the discrete-time (Patil et al., 2015) and process-slot continuous-time (Lagzi et al., 2017a) formulations. By solving a number of examples, it was concluded that the discrete-time formulations using uniform and non-uniform time grids requires less computational time than process slot-based alternative, especially for large-scale problems. However, those two discrete-time formulations are possible to lead to suboptimum solutions in some cases, especially if a coarse discretization is used since a unit can only start examining a property exactly at time interval points. The non-uniform discrete-time model of Lagzi *et al.* (2017b) was extended to consider allocation of personnel to active machines (Santos *et al*., 2018) and two conflicting objectives (Lee *et al*., 2019).

 In this work, we use the unit-specific event-based modelling approach whose advantages have been well established in the literature (Shaik and Floudas, 2009; Li and Floudas, 2010) to develop efficient models for scheduling of multi-tasking batch processes in a scientific service facility. In this unit-specific event-based modelling approach, timing variables could be defined either based on tasks similar to the definition of Shaik and Floudas (2009) or based on units (Ierapetritou and Floudas 1998). In order to examine the capabilities of both timing variable representations, we develop three different unit-specific event-based mathematical models. While in the first two models we define a number of timing variables based on tasks in the process, in the third model we introduce a number of timing variables based on processing units in the process. The main difference between the first two models are the tightening constraints. The first two models could be considered as an extension of the model of Shaik and Floudas (2009) for allowing multiple tasks to take place in a unit simultaneously. The third model is completely different from all existing models. A number of examples are solved to illustrate the capability of the proposed three formulations and compared with the existing mathematical models in the literature. The computational results demonstrate that the proposed mathematical models are able to reduce the number of event points required, which leads to a much smaller model size compared to the existing models in the literature (Patil *et al.* 2015; Lagzi *et al*. 2017a; Lagzi *et al*. 2017b). The third model with the timing variables defined based on processing units is the most superior since it generates the optimum solution in significantly less computational time, especially when minimization of makespan is used as the objective, where at least one order of magnitude less computational time than all other models is required to generate the optimum solution compared to other existing models.

# 2 Problem Statement

Figure 1 illustrates a general multi-tasking batch process in a scientific service facility. The scientific service facility receives *O* (*o* = 1, 2, 3, ..., *O* orders/sample groups from different customers that are required to be examined for a total of *P* (*p* = 1, 2, 3, …, *P*) properties using totally *J* (*j* = 1, 2, 3, …, *J*) machines (or processing units). Each order/sample group contains a number of samples. We assume that all samples in an order/sample group are examined for the same number of properties without loss of generality. This is because if an order/sample group contains samples that are examined for different properties, this order/sample group will be divided into different orders/sample groups. Each order/sample group has to be examined for a number of properties based on the customer request. The property examination sequence for an order/sample group is known a priori. However, each order/sample group could have different property examination sequence and thus follow a different processing path. A machine (or unit) can only examine one property. Each machine (or unit) is allowed to examine a number of samples from different orders/sample groups at the same time depending on its capacity. The examination time of a machine (or unit) *j* is known and denoted as *αj* It only depends on the property that is required to be examined, not the batch size. If a machine (or unit) starts examining some samples, then a new sample can be processed only after the completion of all current samples. In other words, a machine (or unit) cannot be interrupted during the examination. The examination of an order/sample group for a property in a machine (or unit) is considered as a *task*. The examination of different properties for the same or different orders/sample groups is treated as different tasks. The examination of different orders/sample groups on the same machine is also treated as different tasks. There are total *I* (*i* = 1, 2, …, *I*) tasks and each machine can process **I***j* tasks.

 An order/sample group has three statuses depending on if its properties are examined. While an order/sample group that is received without any properties examined is called “raw material”, an order/sample group with some properties examined is called “intermediate state”. An order/sample group that has been completely examined is called “final product”. There are total *S* (*s* = 1, 2, 3, …, *S*) states. In Figure 1, states “S1, S2, ..., SO–1, SO” denote “raw material states”, “SO+1, SO+2, …, S2O–1, S2, S2O+1,…, SPO” are “intermediate states” and “SPO+1, SPO+2, …, S-1, S” are “final products”. The “raw material” is included in a set **S***R*, the “intermediate state” is denoted as **S***IN*, and the “final product” is denoted as **S***F*. Each task can “consume” or “produce” at most a state. Tasks that produce a state *s* are denoted as and tasks that consume a state *s* are denoted as . The portion of a state *s* that is used for task *i* is denoted as *ρi*,*s*. If a task *i* consumes a state *s*, then *ρi*,*s* = –1. If a task *i* produces a state *s*, then *ρi*,*s* = 1.

 Each “intermediate state” has its own dedicated storage. There are several intermediate storage policies for each intermediate state including unlimited intermediate storage policy (UIS), finite intermediate storage policy (FIS) and no intermediate storage policy (NIS). There are also several possible wait policies for an intermediate state in a processing unit after processing including unlimited wait policy (UW), limited wait policy (LW) and zero wait policy (ZW). In a scientific service facility, a sample is allowed to stay in the processing time without any restriction after examination. Thus, UW policy is applied. We also assume unlimited intermediate storage policy for samples. With all of these, the scheduling problem can be stated as follows,

Given:

1. *O* orders/sample groups, the number of samples in each order/sample group, properties and their examination sequence for each order/sample group;
2. *J* machines (or processing units), minimum and maximum capacities, suitable properties and tasks, processing times;
3. The scheduling horizon *H*.

Determine:

1. Optimal processing schedule involving task allocations, start and end timings, sequences and batch sizes;
2. Inventory profiles.

Operating rules:

1. More than one tasks are allowed to be processed in a processing unit simultaneously;
2. Each machine (or unit) can examine only one property;
3. Batch splitting and mixing is allowed for each order/sample group.

Assuming:

1. All parameters are deterministic;
2. The processing time of a machine (or unit) *j* is fixed (denoted as *αj*). It only depends on the property that is required to be examined, not the batch size;
3. Unlimited feed materials are available;
4. Unlimited storage policy for all states;
5. Unlimited resources where required are available;
6. Unlimited wait policy for intermediate states.

 The objective of the given problem is to maximize the number of samples examined, during a specified scheduling horizon (maximization of productivity) or to minimize the time required to examine all properties of a specific number of samples (minimization of makespan).

**3 Mathematical Formulation**

As discussed before, the capabilities of the unit-specific event-based modelling approach have been well established in the literature (Shaik and Floudas, 2009; Li and Floudas, 2010), which is used to develop three mathematical models for the given problem due to different ways for the definition of timing variables. Next, we present these three models in detail.

## Models M1a and M1b

In the models **M1a and M1b**, the timing variables are defined based on tasks in the process, which is similar to the definition of most existing unit-specific event-based models (Shaik and Floudas, 2009; Li and Floudas, 2010). We also follow the approach of Shaik and Floudas (2009) that uses a parameter to regulate the maximum allowable number of event points that a task is allowed to span over. It should be noted that the examination of the same order/sample group in different machines (or units) for the same properties is treated as *different tasks* in order to define timing variables based on tasks.

### *Allocation constraints*

We define three-index binary variables *wi*,*n*,*n*′ to denote if a task *i* is active from event point *n* to event point *n*′ (*n* ≤ *n*′), which is similar to those in the model of Shaik and Floudas (2009). If a task is allowed to span over multiple event points, it should start or end at only one event point. Constraint (1) guarantees that a task *i* have no more than one start or end in different event points.

 ∀*j*, *i* ∊ **I***j*, *n*, Δ*n* > 0 (1)

Note that constraint (1) is valid for every task that can be processed in a unit *j*. Therefore, it allows multiple tasks to take place in the same unit simultaneously. This constraint (1) is different from those of Shaik and Floudas (2009).

 We define new 0-1 continuous variables *yj*,*n*,*n*′ to denote if a unit *j* is active from event point *n* to *n*′. Since multiple tasks are allowed to take place in the same unit simultaneously, constraints (2) and (3) are introduced to establish the relationship between *wi*,*n*,*n*′ and *yj*,*n*,*n*′. More specifically, constraint (2) states that when a task is active from event points *n* to *n*′ (i.e., *wi*,*n*,*n*′ = 1), a unit *j* that is able to process that task *i* must be also active (*yj*,*n*,*n*′ = 1). Furthermore, if none of the tasks that can be processed in a unit *j* is active, this unit *j* is forced to be inactive (*yj*,*n*,*n*′ = 0) as indicated in constraint (3).

 ∀*j*, *i* ∊ **I***j*, *n*, *n ≤ n*′ *≤ n+*Δ*n* (2)

 ∀*j*, *n*, *n ≤ n*′ *≤ n+*Δ*n* (3)

It should be noted that constraints (2) and (3) enforce *yj*,*n*,*n*′ can take value 0 or 1 and therfore they are defined as 0-1 continuous variables.

### *Capacity constraints*

As previously discussed, multiple sample groups can be examined in a processing unit simultaneously. We define *bi*,*n*,*n*′ to denote the batch size of a sample group processed by a task *i*. The summation of all samples that are examined in the same unit *j* should be within its minimum unit capacity () and maximum unit capacity (). Therefore, constraint (4) is introduced to avoid a capacity violation.

 ∀*j*, *n*, *n* ≤ *n*′≤ *n+*Δ*n* (4)

### *Material balance constraints*

The amount of a material state *s* stored at the beginning of an event point *n* should be equal to its storage amount at the beginning of the previous event point (*n* – 1) plus the amount of the state produced at event point (*n* – 1) (i.e., *ρi*,*s* > 0), minus the amount of the state consumed at event point (i.e., *ρi*,*s* < 0).

 ∀*s*, *n* > 1 (5)

Notice that constraint (5) does not include the amount of a material state *s* stored at the beginning of the first event point. The amount of a material state *s* stored at the beginning of the first event point should be equal to the initial amount of state *s* minus the amount of the state consumed by tasks that start to process state *s* at the beginning of the first event point.

 ∀*s*, *n =* 1 (6)

The material balance constraints are similar to those of Shaik and Floudas (2009).

### *Duration constraints*

The processing duration of task *i* is computed using constraint (7) if it is not allowed to span over multiple event points (Δ*n* = 0).

 ∀*i*, *n*,Δ*n* = 0 (7)

If a task *i* is allowed to span over multiple event points (i.e, Δ*n* > 0), then constraints (8) and (9) are applied.

 ∀*i*, *n*,Δ*n* > 0 (8)

 ∀*i*, *n*, *n* ≤ *n*′≤ *n+*Δ*n*,Δ*n* > 0 (9)

#### *Same task in the same unit*

A task *i* at event point (*n* + 1) must always start after it completes at the previous event point *n* as specified by constraint (10).

 ∀*i, n < N* (10)

If a task is allowed to span over multiple event points, then the start time of task *i* at event point (*n* + 1) should be equal to the finish time of the same task at the previous event point *n* if task *i* is active at event point *n* but it continues being active at the next event point, as indicated in constraint (11).



 ∀*i*, *n*,Δ*n* > 0 (11)

#### *Different tasks in the same unit*

A task *i* at an event point (*n* + 1) must always start after any other task *i*′ that can be processed at the same unit as this task completes at event point *n*.

 ∀*j*, *i*∊ **I***j*, *i*′∊ **I***j*, *i* ≠ *i*′, *n* < *N* (12)

#### *Different tasks in different units*

Constraint (13) is introduced to define the sequence between tasks in different units that produce and consume the same state *s*. A consumption task *i* at event point (*n* + 1) must start after a production task *i*′ related to the same state completes at event point *n*, if the producing task finishes processing materials at event point *n*.

 ∀*s* ∊ **S***IN*, *j* ≠ *j*′, *i* ∊ , *i*′∊ , *i ≠ i*′, *n* < *N* (13)

### *Tightening constraints*

Shaik and Floudas (2009) introduced a number of tightening constraints in order to tight the relaxation of their MILP formulation. However, these constraints are proposed with the assumption that at most one task is allowed to be processed in a unit at a time. Therefore, these constraints cannot be used in this multi-tasking scheduling problem. In this work, we present two different tightening constraints. In the computational results, we will compare the performance of these two different tightening constraints. The first one is the modification of the tightening constraints from Shaik and Floudas (2009) as indicated in constraint (14).

 ∀ *j* (14)

 In order to develop the second tightening constraints, we introduce new variables and to denote the start and end time of a unit *j* at event point *n*. According to constraint (15), the finish time of a unit *j* at an event point *n* must be after the start time of this unit at the same event point plus the maximum processing time of the tasks that are available to be processed in the unit. Furthermore, according to constraint (16), the start time of unit *j* at event point (*n* + 1) must always be after the finish time at the previous event point *n*.

 ∀ *j*, *n*, *n* ≤ *n*′≤ *n+*Δ*n* (15)

 ∀ *j*, *n* < *N* (16)

### *Variable bounds*

All timing variables should take values less than the scheduling horizon, as indicated in constraints (17)-(20).

 ∀ *i*, *n* (17)

 ∀ *i*, *n* (18)

 ∀ *j*, *n* (19)

 ∀ *j*, *n* (20)

The number of samples examined must always be less than the maximum capacity. Therefore, constraint (21) defines the upper limit for these variables.

 ∀ *i*, *n*, *n* ≤ *n*′≤ *n+*Δ*n*  (21)

### *Objective function*

Two objective functions are considered: maximization of productivity and minimization of makespan.

#### *Maximization of productivity*

Usually, it is better for a scientific service facility to complete the examination of all samples received during the specified scheduling horizon. However, this may not be achieved. Therefore, it is necessary to maximize the total number of samples that can be examined within the scheduling horizon, as indicated in constraint (22).

 (22)

where *ps* is a weighted value, which takes the value of 1 for “intermediate products” and 5 for “final products”.

***Minimization of makespan***

Another objective is to minimize the time required to complete the examination of all samples received from customers. We define a variable *MS* to denote that the minimum time needed to examine all properties for all samples, which should exceed the finish time of all tasks at the last event point in the process.

 ∀*i*, *n* = *N* (23a)

 ∀*j* (23b)

 ∀*i*, *n* = *N* (23c)

To achieve minimization of makespan, one additional constraint should be considered to ensure that all samples are examined.

 ∀ *s* ∊ **S***P*, *n* = *N* (24)

where *Ds* is the total amount of samples that have to be examined. Note that at the last event point all samples have to be examined for all properties. Therefore, we only consider “production states” in (24). Furthermore, the total number of samples received from customers should be equal to the number of samples required to be examined at the last event point.

 We complete the model **M1a** which comprises eqs. 1-14, 17-18 and 21-22 for maximization of productivity and eqs. 1-13, 21, 23a, 23b and 24 for minimization of makespan. This model **M1a** uses the first tightening constraints (i.e., constraints 14 and 23b). Another model **M1b** using the second tightening constraints (i.e., constraints 15-16) are completed which comprises eqs. 1-13 and 15-22 for maximization of productivity and eqs. 1-13, 15-16, 21, 23a, 23c and 24 for minimization of makespan. It should be noted that although different tasks in the same unit may not start at the same time from the schedule generated using the models **M1a** and **M1b**, it is easy to revise the schedule to make sure that different tasks in the same unit start at the same time without any effect on the objective function.

## Model M2

In this model, the timing variables are defined based on units. Since multiple tasks are allowed to take place at the same units simultaneously, we want to know their active status of each unit at a time. Thus, we define binary variables *wj*,*n*,*n*′ to denote if a unit *j* is active from an event point *n* to another event point *n*′ (*n*′ > *n*).

### *Allocation constraints*

Although a unit *j* is allowed to be active over multiple event points, it can start or end only at one event point, as indicated in constraint (25).

 ∀*j*, *n*,Δ*n* > 0 (25)

Note that constraint (25) is valid for each unit *j* without involving any task. Thus, it does not restrict the number of tasks that are allowed to be processed in a unit *j*.

### *Capacity constraints*

We define continuous variables *bi*,*j*,*n*,*n*′ to denote the batch size that is processed by a task *i* in a unit *j* from event point *n*to event point *n*′. Recall that multiple tasks are allowed to be processed in a unit *j* simultaneously. The total batch size processed in a unit *j* should be within the minimum () and maximum () capacities of this unit *j* at a time, as indicated by constraint (26).

 ∀*j*, *n*, *n* ≤ *n*′≤ *n+*Δ*n* (26)

### *Material balance constraints*

The amount of a material state *s* stored at the beginning of event point *n* should be equal to the amount of the state *s* stored at the beginning of event point (*n* – 1), plus the amount of this state produced by tasks at the end of event point (*n* – 1) (i.e., *ρi*,*s* > 0), minus the amount of state *s* consumed by tasks at the beginning of event point *n* (i.e., *ρi*,*s* < 0).

 ∀*s*, *n* > 1 (27)

The amount of a material state stored at the beginning of the first event point should be equal to the initial amount of state minus the amount of the state consumed by tasks that start to process state at the beginning of the first event point.

 ∀*s*, *n =* 1 (28)

### *Duration constraints*

The processing duration of a unit *j* is defined by constraint (29). The constraint assumes constant processing time (*αj*) for all tasks, which is unit dependent only. This is true for property examination in a scientific service facility. Constraint (29) indicates that the end time of a unit *j* at event point *n*′ must be greater than the start time of this unit *j* at event point *n* plus the constant processing time if unit *j* is active from event point *n* to event point *n*′.

 ∀*j*, *n*, *n* ≤ *n*′≤ *n+*Δ*n* (29)

### *Sequencing constraints*

To sequence different tasks in different units, we define continuous variable *Ts*,*n*to denote the time that state *s* is available at event point *n*. The end time of tasks that produce a state *s* from event point *n*′ to event point *n* must be before the time that state *s* is available at event point *n* as denoted by constraint (30). We assume that “Raw material states” are available at the beginning of the scheduling horizon, while “final product states” are not “consumed” by any task. Therefore, they are not considered in constraint (30).

 ∀ *s* ∊ **S***IN* , *j*, *n*, (30)

Furthermore, the start time of tasks that consume state *s* from event point (*n* + 1) to event point *n*′, must be after the time that state *s* is available to be consumed at event point *n* as specified by constraint (31).

 ∀ *s* ∊ **S***IN* , *j*, *n* < *N*, (31)

Similar to constraint (30), constraint (31) is also only valid for “intermediate material states”.

A unit *j* at event point (*n* + 1) must always start after any other process in unit *j* ends at event point *n*. This is because the property examination for a new order/sample groups should wait until the current examination completes.

 ∀ *j*, *n* < *N* (32)

The start time that state *s* is available to be consumed at event point *n* must always be before the time that is available to be consumed at the next event point (*n* + 1), as denoted by constraint (33).

 ∀ *s* ∊ **S***IN*, *n* < *N* (33)

### *Variable bounds*

 ∀ *j*, *n* (34)

 ∀ *j*, *n* (35)

 ∀ *s*, *n* (36)

 ∀ *j*, *i*∈**I***j*, *n* ≤ *n*′ ≤ *n*+Δ*n* (37)

### *Objective function*

Similar to the models **M1a** and **M1b**, two objective functions are considered for the model **M2** including maximization of profit and minimization of makespan.

#### *Maximization of productivity*

Given a specific scheduling horizon, objective function (38) is used to maximize the total number of samples that can be examined at the end of the scheduling horizon.

 (38)

#### *Minimization of makespan*

For minimization of makespan, the objective function (39a) is introduced.

 ∀ *j*, *n = N* (39a)

 ∀ *j*, *n = N* (39b)

Similar to models **M1a** and **M1b**, one more constraint should be considered to ensure that all properties are examined in all samples in the case of minimization of makespan.

 ∀ *s* ∊ **S***P*, *n* = *N* (40)

 We complete the model **M2** which comprises eqs. 25-38 for maximization of productivity and eqs. 25-33, 37, 39-40 for minimization of makespan.

# 4 Computational studies

We solve 32 examples to illustrate the capabilities of the proposed models. Example 1 is the illustrative example from Lagzi *et al.* (2017a) in which two groups of samples are examined for four properties in the facility having six machines. The necessary data are given in Tables 1-2. Examples 2-20 are generated randomly following discrete uniform distribution. Examples 2-6 have ten groups of samples with each containing from 50 to 80 samples. These sample groups have to be examined for 1 to 4 properties. Examples 7-8 have five groups of samples with 1-4 properties to be examined. One sample has to be examined for more than once for the same property since a property needs to be examined in two or more different conditions such as varying temperature or pressure. This could often happen in a scientific service facility, as illustrated in Lagzi *et al.* (2017a). Examples 9-19 involve 2-10 groups of samples with each containing from 50 to 80 samples having 3-8 properties to be examined. The necessary data for Examples 2-20 can be found in the **Supplementary Material**. A scheduling horizon of 480 min (i.e., 8 hours) is considered for Examples 2-19. Example 20 has 100 groups of samples with each containing 200 to 300 samples having a total 25049 samples to be examined. There are 25 properties that are required to be examined with each group having 8-9 properties. The scheduling horizon is 40 hours.

 Examples 21-32 have the same number of processing units and properties to those of Lagzi *et al.* (2017b). The processing time, and maximum capacity of each processing unit in Examples 21-32 are also exactly same as those of Lagzi et al. (2017b). Other data are generated randomly following the discrete uniform distribution since they are not provided in Lagzi et al. (2017b). While 5 sample groups are considered for Examples 21-25, 10 sample groups are involved in Examples 26-30. Each sample group in Examples 21-30 contains from 50 to 80 samples. Example 31 considers 100 sample groups with each group containing 200-300 samples. A total of 25245 samples have to be examined. Example 32 contains 100 sample groups with 250-350 samples for each group. For Example 32, a total of 30067 samples have to be examined. Two scheduling horizons including *H* = 480 min and *H* = 1440 min are investigated for Examples 21-30, whilst a scheduling horizon of 40 hours (*H* = 2400 min) was examined for Examples 31-32. All sample groups are able to be processed using 11 predefined processing paths with each having 1-10 machines as shown in Table 3 and Figure 2. It should be noted that Examples 20-32 represent large-scale actual scientific service facilities (Lagzi *et al.*, 2017b). The necessary data for Examples 21-32 can be found in the **Supplementary Material**. All examples vary with the number of sample groups, properties, machines, and scheduling horizon. All examples are solved in CPLEX 12/GAMS 24.6.1 on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7. We set the maximum CPU time for all examples as 1 hour.

 We also compare the performance of the proposed three models **M1a, M1b** and **M2** with those discrete-time models of Patil *et al.* (2015) and Lagzi *et al*. (2017b) and the process-slot continuous-time model of Lagzi *et al.* (2017a). While the discrete-time model of Patil *et al*. (2015) uses a uniform time grid for all units in which the scheduling horizon is divided into time intervals of equal length, the model of Lagzi *et al*. (2017b) uses non-uniform time grid in which the scheduling horizon is divided into time intervals of varying length. The length of each time interval in Patil *et al*. (2015) is equal to the greatest common factor of all tasks since all tasks have to start and end exactly at the time points. In the model of Lagzi *et al*. (2017b) the maximum time interval length was set to 60 minutes. For units with processing time less than 60 minutes the length of each time interval was equal to the processing time. On the other hand, for units with larger processing times, the time interval is set to the maximum length (i.e., 60 minutes). It should be mentioned that the models from Patil *et al.* (2015) and Lagzi *et al.* (2017a-b) did not consider makespan minimization. These models are extended for minimization of makespan in this paper which are presented in **Appendices A and B**.

***Example 1***

This example involves two groups of samples (group 1 and group 2). There are in total 4 properties (P1-P4) using in 6 units (J1-J6). The property examination sequences for two groups of samples are P1-P3-P4 for group 1 and P1-P2-P3-P4 for group 2. Property 1 is examined in unit J1. Property 2 is examined in units J2 and J3. Property 3 is examined in units J4 and J5. Property 4 is examined in unit J6. The examination of each property from a sample group is denoted as a *task*. For instance, we use task I1 to denote the examination of P1 for the sample group 1 in unit J1 and use task I2 to denote the examination of P1 for the sample group 2 in unit J1. There are in total 10 tasks (I1-I10). We use state S1 to denote the initial status of the sample group 1. We use states S1-S9 to denote the status of the two sample groups. The state-task network for this example is illustrated in Figure 3. The computational results are given in Table 4. From Table 4, it can be seen that all six models solve Example 1 in very small CPU time (< 1 s) for both objective functions. However, the proposed models **M1a, M1B** and **M2** lead to smaller model size than the existing models and hence they can be potentially superior, especially in the case of minimization of makespan, where they also lead to a tighter MILP relaxation. The optimal solutions are generated with Δ*n* = 0 from the proposed models **M1a, M1b** and **M2.** The optimal schedule generated using the mathematical model **M2** with maximization of productivity is depicted in Figure 4. From the optimal schedule (Figure 4) it can be observed that two tasks are processed in the same unit simultaneously. For instance, unit J1 examines the property P1 of 70 samples from the sample group 1 and 70 samples from the sample group 2 simultaneously during 0 to 50 minutes.

 Another remarkable finding is that the discrete-time model of Patil *et al*. (2015) leads to a much tighter MILP relaxation than all other models when the objective is to maximize productivity. More specifically, it is interesting that the solution from the relaxed MILP is exactly identical to the optimal solution. Even though the discrete-time model of Patil *et al.* (2015) leads to large model size, it requires similar computational time than the rest. On the contrary, for minimization of makespan the discrete-time formulation of Patil *et al*. (2015) leads to a much worse relaxation than other models. By comparing the uniform discrete-time model of Patil *et al.* (2015) with that non-uniform discrete-time model of Lagzi *et al.* (2017b), it can be concluded that the use of a non-uniform discretization can lead to smaller model size. More specifically, if maximization of productivity is used as objective, the model of Lagzi *et al.* (2017b) requires approximately half discrete variables than the model of Patil *et al.* (2015) (715 vs 1394). Therefore, Lagzi *et al.* (2017b) formulation can potentially be more efficient in terms of computational time than the discrete-time model of Patil *et al.* (2015). However, since a coarser discretization is used, it can lead to suboptimum solutions. For instance, if minimization of makespan is used as objective, the model of Lagzi *et al.* (2017b) leads to 9.9% worse solution than other models (555 min vs 500 min).

 From Table 4 it can also be seen that all continuous-time formulations (i.e., Lagzi et al. 2017a, **M1a**, **M1b** and **M2**) require the same number of event points or slots for both objective functions. Although the models **M1a** and **M1b** lead to the same MILP relaxation, the model **M1b** has more continuous variables and constraints than **M1a** due to the introduction of additional variables and with additional related constraints (e.g., constraints 15 and 16). As a result, the model **M1a** leads to slightly smaller CPU time than the model **M1b**.

### *Other examples*

The computational results for Examples 2-32 with the objective of maximization of productivity are given in Tables 5-8, whilst the results for Examples 2-19, 21-23 and 27 with the objective of minimization of makespan are given in Tables 9-11. The column “event points” in Tables 4-11 presents the number of event points required for **M1a**, **M1b**, and **M2**, the number of slots required for the model of Lagzi et al. (2017a) and the number of time intervals required for the models of Patil et al. (2015) and Lagzi et al. (2017b).

### *Maximization of productivity*

Table 5 presents the computational results for Examples 2-8. From Table 5, it can be concluded that the model **M2** is the most superior since it requires less computational time than the models of Patil *et al.* (2015), Lagzi *et al.* (2017a), **M1a** and **M1b**. The main reason is that the model **M2** leads to a much smaller model size especially less number of discrete variables required. For example, it can generate the optimum solution for Example 2 by using 89.1% less constraints than the model of Lagzi *et al.* (2017a) (i.e., 703 vs. 6424), 85.7% less constraints than the model of Patil *et al.* (2015) (i.e., 703 vs. 4907), 73.8% less constraints than the model **M1a** (i.e., 703 vs. 2682) and 74.2% less constraints than the model **M1b** (i.e., 703 vs. 2730). Furthermore, the model **M2** requires 82.1% less discrete variables than the model of Lagzi *et al.* (2017a) (i.e., 260 vs. 1456), 94.1% less discrete variables than the model of Patil et al. (2015) (i.e., 260 vs. 4412) and 43.5% less discrete variables than the models **M1a** and **M1b** (260 vs. 460). Even though the model **M2** also leads to a smaller model size than the model of Lagzi *et al.* (2017b), (51.9% less discrete variables, 62.8% less continuous variables and 12.8% less constraints), it requires more computational time. This is mainly because the model of Lagzi *et al.* (2017b) leads to a much tighter MILP relaxation. However, in most cases, the model of Lagzi *et al.* (2017b) is only limited to generate a suboptimum solution in contrast to the proposed model **M2** which generates the optimum solution for all these examples.

 The uniform discrete-time formulation of Patil *et al.* (2015) performs better than the models **M1a**, **M1b** and the process-slot continuous-time model of Lagzi *et al.* (2017a). This is mainly due to the fact that the solution from the relaxed MILP of Patil et al. (2015) is exactly the same as the optimum solution for all these examples. However, the exceptionally high model size makes the model inferior to the model **M2**. By comparing the models **M1a, M1b** and **M2** and the process-slot model of Lagzi *et al.* (2017a), it can be concluded that proposed models **M1a**, **M1b** and **M2** generate optimum solutions in much less computational time, due to the fact that they are much tighter and hence lead to smaller model size. More importantly, the process-slot model of Lagzi *et al.* (2017a) requires more slots than the models **M1a**, **M1b** and **M2** in some examples. This is because all tasks in the process have to start or end at the same slot points. It can also be observed that the model **M2** requires more event points than models **M1a** and **M1b** for Example 5. The main possible reason is due to the constraints (30)-(31), which impose all states that can be processed in a unit *j* to be available after the unit finishes tasks or before the unit begins to process tasks once the unit is active regardless which task is processed in a unit *j*. Consequently, more event points than models **M1a** and **M1b** are required to generate the optimal solution for this Example 5. It is interesting that even though the model **M2** requires one more event point than these models, it still leads to a much smaller model size. As a result, it generates the optimum solution in significantly less amount of CPU time than **M1a** (0.328 s vs. 20.64 s) and **M1b** (0.328 s vs. 19.33 s).

 By examining the models **M1a** and **M1b**, both of them lead to the same MILP relaxation for all these examples. The model **M1a** leads to smaller number of continuous variables and constraints but the same number of discrete variables compared to the model **M1b**. For instance, for Example 4 the model **M1a** require 1086 continuous variables and 4667 constraints to generate the optimal solution, while the model **M1b** require 1170 and 4739 respectively. However, the model **M1a** requires slightly more computational time than the model **M1b** in some cases. For both models **M1a** and **M1b** the computational time required is within the same order of magnitude. For instance, the model **M1a** needs 84.69 s to generate the optimum solution for Example 4, whereas the model **M1b** requires 80.11 s.

 Table 6 lists the computational results for Examples 9-20. As it is demonstrated, the model **M2** is the most superior among all six formulations for most examples (Examples 9-17). The optimal schedule for Example 17 is illustrated in Figure 5. From Figure 5, it can be again confirmed that samples from different customers can be processed simultaneously in the same unit. However, the model **M2** generate the optimum solution, but it fails to converge within 1 hour for more complex examples (e.g., Examples 18-19), whereas the uniform discrete-time formulation of Patil *et al.* (2015) generates the optimum solution for Examples 18-19 within 1 hour. Consequently, it seems that the tighter relaxation of the discrete-time model of Patil *et al*. (2015) makes it more efficient for more complex problems when productivity is maximized. The process-slot model of Lagzi *et al.* (2017a) and the models **M1a** and **M1b** require more computational time to generate the optimum solution than the discrete-time model of Patil *et al.* (2015) for most examples even though they lead to smaller model size. Among all models, the one of Lagzi *et al*. (2017a) seems to have the worse MILP relaxation and hence it performs worse than the other models for most examples. The model of Lagzi *et al.* (2017b) requires significantly less computational time than all other models for all examples especially for large and complex examples (Examples 18-20) with less than one second required to generate a solution for Examples 18-19 and less than one minute for Example 20. However, the solution quality is much worse compared to other models. Similar observations can be made for the models **M1a** and **M1b** as those from previous Examples 2-8. For instance, the proposed model **M2** is able to generate the solution of 69664 cu for Example 20, while the model of Lagzi *et al.* (2017b) generates a significantly worse solution of 53100 cu.

 Tables 7-8 present the computational results for Examples 21-32. From Tables 7-8, it seems that the discrete-time model of Patil *et al.* (2015) as well as the models **M1a**, **M1b** and **M2** are more efficient compared to the process-slot model of Lagzi *et al.* (2017a). For instance, in Example 27b, the formulation of Lagzi *et al.* (2017a) could generate the optimal solution but could not converge after 1 hour, while all other formulations are able to generate the optimal solution in less than 1 minute. The developed model **M2** is more superior compared to the discrete-time model of Patil *et al.* (2015). For example, the model **M2** requires 99.0% less computational time than the discrete-time model of Patil *et al.* (2015) (0.188 s vs. 18.69 s) to generate the optimal solution for Example 27b. The model of Lagzi *et al.* (2017b) is also able to generate the optimum solution for Examples 21-30. Furthermore, the tight relaxation of the Lagzi *et al.* (2017b) model makes it as efficient as the mathematical models **M1a**, **M1b** and **M2** for this set of examples. However, the model of Lagzi *et al.* (2017b) fails to generate the optimal solution for large-scale examples. For instance, for Example 31 the model of Lagzi *et al.* (2017b) generates a solution of 34805 cu, while the proposed model **M2** generates a better solution of 36965 cu. Similarly, for Example 32 the proposed model **M2** is able to generate a significantly better solution of 49991 cu than the model of Lagzi *et al.* (2017b) which generates a solution of 39827 cu. On the other hand, both mathematical models **M1a** and **M1b** are more efficient than the discrete-time model of Patil *et al.* (2015). More specifically, both **M1a** and **M1b** models require at least one order of magnitude less computational time than the model of Patil *et al.* (2015) to generate the optimal solution for all these examples. This is because both models lead to significantly smaller model size. For instance, both models **M1a** and **M1b** require98.6% less discrete variables (966 vs 71220) to generate the optimal solution for Example 21a.

### *Minimization of makespan*

Table 9 gives the computational results for Examples 2-8. From Table 9, it seems that the process-slot model of Lagzi *et al.* (2017a) is not suitable for the problem of makespan minimization since it is unable to generate the optimum solution within 1 hour for most examples. The model of Lagzi *et al.* (2017b) is also not suitable since it fails to generate the optimum solution for all these examples. The discrete-time formulation of Patil *et al.* (2015) and the model **M2** are able to generate the optimum solution within 1 hour. Between these two models, the model **M2** is more efficient since it generates the optimum solution in at least two orders of magnitude less computational time. This is due to the fact that the model **M2** leads to both much smaller model size and a tighter MILP relaxation. Similarly, the models **M1a** and **M1b** require less computational time than the discrete-time formulation of Patil *et al.* (2015), due to their much smaller model size and tighter MILP relaxation. However, in some cases such as Examples 3 and 6, the proposed models fail to converge after 1 hour, while the discrete-time of Patil *et al.* (2015), requires significantly less computational time to converge (93.5 s and 17.3 s respectively).

 Table 10 presents the computational results for Examples 9-19. From these examples, it can be concluded that the model **M2** is superior compared to the other five models, even though it requires more number of event points compared to models **M1a** and **M1b** in some cases. It can also be observed that the model **M2** is able to generate solutions for more complex examples, (Examples 18-19). More specifically, the model **M2** can generate the optimum solution for Example 18 within a minute (i.e., 28.5 s). However, the models of Patil *et al.* (2015) and Lagzi *et al.* (2017a) are unable to generate a feasible solution and the models **M1a** and **M1b** fail to generate the optimum solution after 1 hour. The superiority of the model **M2** lays to the fact that it not only leads to smaller model size but also leads to tighter MILP relaxation for makespan minimization problems. From Table 10, it seems that both the models **M1a** and **M1b** perform better than the models Patil et al. (2015), Lagzi et al. (2017a) and Lagzi et al. (2017b). The main reason that the models **M1a** and **M1b** aremore efficient is that they both lead to smaller model size and tighter MILP relaxation. For instance, in Example 15, the model **M1a** requires one order of magnitude less computational time than the discrete-time formulation of Patil *et al.* (2015) (1.451 s vs 13.74 s), two orders of magnitude less computational time than the process-slot model of Lagzi *et al.* (2017a) (1.451 s vs 119.7 s) and one order of magnitude less computational time than the model of Lagzi et al. (2017b) (1.451 s vs 13.7 s). However, both **M1a** and **M1b** fail to generate optimum solutions in more complex problems (Examples 17-19). By comparing the models **M1a** and **M1b**, both tightening constraints lead to the same MILP relaxation, indicating that they are very similar. Furthermore, the model **M1a** leads to slightly smaller model size. Despite that, for both models the computational time required is within the same order of magnitude.

 Table 11 presents the computational results for Examples 21-23 and 27, which are highly complex. The computational results for Examples 20, 24-26 and 28-32 are not presented because none of the models could generate optimal solutions or even feasible solutions for these six examples within the predefined CPU time (i.e., 1 hr). From Table 11, it seems that the model **M2** is able to generate the best solution within 1 hour. The discrete-time models of Patil *et al.* (2015) and Lagzi *et al.* (2017b) are unable to generate a feasible solution within 1 hour. Therefore, it can be concluded the model **M2** is superior compared to the other four models. The models **M1a** and **M1b** are only able to generate feasible solutions for Examples 21-23. However, they both fail to converge after 1 hour. Similar observations can be done regarding the model size of **M1a** and **M1b**.

# 5 Conclusions

In this paper, we developed three novel MILP mathematical formulations using the well-established unit-specific event-based modelling approach for scheduling of multi-tasking multipurpose batch processes in a scientific service facility. Multiple tasks were allowed to be processed simultaneously in the same units. While the timing variables were defined based on tasks of the process in the first two models (**M1a and M1b**), they were introduced based on processing units of the process in the third model (**M2**). Two different tightening constraints were proposed in the models **M1a** and **M1b** to improve their MILP relaxation. The computational results demonstrate that the model **M2** is the most efficient for most examples since it generates the optimum solution in significantly less amount of computational time than all other models. The proposed tightening constraints for the models **M1a** and **M1b** resulted in the same MILP relaxation for all examples. Although the model **M1a** has less number of constraints and continuous variables than the model **M1b**, it seems that their performance is almost the same. The future work is to employ rolling-horizon decomposition approach to solve all examples especially those large-scale complex problems that the best model **M2** fail to solve. Even though this work is focused on scientific service facilities, it can be also implemented in any multipurpose batch process industry which allows multiple tasks to take place simultaneously in a processing unit.

**Acknowledgements**

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

# Nomenclature

### *Sets*

: tasks

: units that can process task

: tasks that consume state

: tasks that produce state

: units

: event points

: processes

: units that are able to process process

: states

: raw material states

: intermediate states

: product states

### Indicies

*i*: tasks

*j*: units

*s*: states

*n*: event points

### *Parameters*

: processing time of task

: processing time of unit *j*

: maximum amount of materials that can be processed at task

: minimum capacity of unit

: maximum capacity of unit

: total amount of samples that have to be examined

: scheduling horizon

: proportion of state that is consumed/produced from task

: initial amount of available state

: maximum number of event points that a task is allowed to span

: large positive number

: maximum amount of materials that can be processed at task

: minimum capacity of unit

: maximum capacity of unit

: total amount of samples that have to be examined

: scheduling horizon

: proportion of state that is consumed/produced from task

: initial amount of available state

: maximum number of event points that a task is allowed to span

: a large positive number

### *Binary variables*

: 1 if task is active from event point to event point

: 1 if unit is active from event point to event point

### *Integer variables*

: amount of materials that are processed in task from event point to event point

: amount of materials that are processed in task which takes place at unit from event point to event point

### *Continuous variables*

: makespan

: amount of state that has to be stored at event point

: start time of task at event point

: end time of task at event point

: start time of unit at event point

: end time of unit at event point

: 1 if unit is active from event point to event point

: total profit

## Appendix A Discrete-time mathematical model proposed by Patil et al. (2015)

### *Sets*

: tasks

: tasks that belong in process *p*

: tasks that consume state

: tasks that produce state

: units

: processes

: units that are able to process process

: states

: time slots

### *Parameters*

: minimum capacity of unit

: maximum capacity of unit

: number of resources available for property

: initial amount of available state

: Duration of examination of property

: proportion of state that is consumed/produced from task at time slot

### *Binary variables*

: binary variable which take the value 1 if unit is active at time slot

### *Integer variables*

: amount of materials that are processed in task which belongs to process at time slot

### *Continuous variables*

: amount of state that has to be stored at time slot

 ∀*s*, *t* >2 (A1)

 ∀*s*, *t* = 2 (A2)

 ∀*p*, *j*∊ **P***j*, *t* (A3)

 ∀*p*, *t* (A4)

*Maximization of productivity*

 (A5)

*Minimization of makespan*

 ∀*p*, *j*∈**P***j*, *t* (A6)

 ∀*s*, *i*, *p*, *t* = *T* (A7)

While the model of Patil et al. (2015) for maximization of productivity includes constraints A1-A5, the model of Patil et al. (2015) for makespan minimization consists of A1-A4 and A6-A7.

## Appendix B Continuous-time mathematical model proposed by Lagzi et al. (2017a)

### *Sets*

: tasks

: units that can process task

: tasks that consume state

: tasks that produce state

: units

: event points

 processes

: units that are able to process process

: states

### *Parameters*

: minimum capacity of unit

: maximum capacity of unit

: scheduling horizon

: earliest available time that task is available

: earliest available time that unit is available

: initial amount of materials in task

: processing time of unit

### *Binary variables*

: binary variable which take the value one if task is assigned to unit to start being processed at event point

### *Integer variables*

: amount of materials from task that begins processing in unit at time point

: amount of materials from task that completes its processing at unit at event point

: amount of materials from task that continues its processing at unit at time point

### *Continuous variables*

: length of time slot

: location of event point

: amount of time remaining to complete processing materials from task that continue to be processed at unit at event point

: amount of materials from task that have visited process and are waiting to visit process at event point

: 0-1 continuous variable which take the value one if a subset of materials from task completed their processing at unit at event point

: 0-1 continuous variable which take the value one if a subset of materials from task continues to be processed at unit at event point

: 0-1 continuous variable which take the value one if unit starts processing materials at event point

 (B1)

 ∀ *n* >1 (B2)

 ∀*j*, *i*∊ **I***j*, *n* (B3)

 ∀*j*, *i*∊ **I***j*, *n* (B4)

 ∀*j*, *n* (B5)

 ∀*j*, *i*∊ **I***j*, *n* > 1 (B6)

 ∀*j*, *i*∊ **I***j*, *n* (B7)

 ∀*j*, *i*∊ **I***j*, *n* (B8)

 ∀*j*, *n* (B9)

 ∀*j*, *i*∊ **I***j*, *i ≠ I0*, *n* (B10)

 ∀*j*, *i*∊ **I***j*, *n* (B11)

 ∀*j*, *n* (B12)

 ∀*j*, *i*∊ **I***j*, *n* (B13)

 ∀*j*, *n* (B14)

 ∀*j*, *i*∊ **I***j*, *n* (B15)

 ∀*j*, *n*  (B16)

 ∀ *i*, *p*, *k* = 1 (B17)

 ∀*j*, *i*∊ **I***j*, *n* (B18)

 ∀ *i*, *k*, *n* = 1 (B19)

 ∀ *i*, *k*, *n* > 1 (B20)

 ∀*j*, *i*∊ **I***j*, *n* (B21)

 ∀*j*, *i*∊ **I***j*, *n* < *N* (B22)

*Maximization of productivity*

 (B23)

*Minimization of makespan*

 ∀ *n* = *N* (B24)

 ∀ *s* ∊ **S***P* (B25)

While the model of Lagzi et al. (2017a) for maximization of productivity includes constraints B1-B23, the model of Lagzi et al. (2017a) for makespan minimization consists of B1-B22 and B24-B25.

# References

Floudas, C. A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers and Chemical engineering*.* 28(11). 2109-2129. <https://doi.org/10.1016/j.compchemeng.2004.05.002>.

Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial application of production scheduling models and solution methods. Computers and Chemical Engineering*.* 62(5). 161-193. <https://doi.org/10.1016/j.compchemeng.2013.12.001>.

Hui C, Gupta A, van der Meulen H A J. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers & Chemical Engineering, 2000, 24(12): 2705 – 2717 [https://doi.org/10.1016/S0098-1354(00)00623-2](https://doi.org/10.1016/S0098-1354%2800%2900623-2)

Ierapetritou, M., G. Floudas, C., 1998. A. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial & Engineering Chemistry*.* 37(11). 4341-4359. https://doi.org/10.1021/ie970927g.

Kondili, E., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations-I MILP formulation. Computers & Chemical Engineering*.* 17(2). 211-227. [https://doi.org/10.1016/0098-1354(93)80015-F](https://doi.org/10.1016/0098-1354%2893%2980015-F).

Kopanos G., Puigjaner L., 2019. Solving Large-Scale Production Scheduling and Planning in the Process industries, Springer, https://doi.org/10.1007/978-3-030-01183-3

Lagzi, S., Fukasawa, R., Ricardez-Sandoval, L., 2017a. A multitasking continuous time formulation for short-term scheduling of operations in multipurpose plants. Computers & Chemical Engineering*.*97. 135-146. <https://doi.org/10.1016/j.compchemeng.2016.11.012>.

Lagzi, S., Lee, D. Y., Fukasawa, R., Ricardez-Sandoval, L., 2017b. A computational study of continuous and discrete time formulations for a class of short-term scheduling problems for multipurpose plants. Industrial and Engineering Chemistry Research. 56(31). 8940-8953. https://doi.org/ 10.1021/acs.iecr.7b01718.

Lee, S. Y., Fukasawa, R., Ricardez-Sandoval, L., 2019. Bi-objective short-term scheduling in a rolling horizon framework: a priori approaches with alternative operational objectives. Computers and Operations research. 111.141-154 <https://doi.org/10.1016/j.cor.2019.06.006>

Li J., Karimi I. A., 2011. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. Industrial & Engineering Chemistry Research, 50(15). 9156-9174 <https://doi.org/10.1021/ie102321b>

Li J., Xiao X., Floudas C. A., 2016. Integrated gasoline blending and order delivery operations: part I. short-term scheduling and global optimization for single and multi-period operations. AIChE Journal, 62(6). 2043-2070 <https://doi.org/10.1002/aic.15168>

Li, J., Floudas, C.A., 2010. Optimal Event Point Determination for Short-Term Scheduling of Multipurpose Batch Plants via Unit-Specific Event-Based Continuous-Time Approaches, Industrial & Engineering Chemistry Reshearch. 49(16). 7446-7469. <https://doi.org/10.1021/ie901842k>.

Li, J., Sursarla, N., Karimi, I. A., Shaik, M. A., Floudas, C. A., 2010. An Analysis of Some Unit-Specific Event-Based Models for the Short-Term Scheduling of Noncontinuous Processes. Industrial and Engineering Chemistry Research*.* 49(2). 633-647. https://doi.org/ 10.1021/ie801879n.

Maravelias C. T., Grossmann I. E., 2003. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research, 42(13). 3056-3074 https://doi.org/10.1021/ie020923y

Maravelias, C. T., 2012. General framework and modeling approach classification for chemical production scheduling. AIChE journal*.* 58(6). 1812-1828. <https://doi.org/10.1002/aic.13801>.

Méndez C. A., Cerdá J. 2000. Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. Computers & Chemical Engineering, 24(2-7). 369-376 [https://doi.org/10.1016/S0098-1354(00)00482-8](https://doi.org/10.1016/S0098-1354%2800%2900482-8)

Méndez C. A., Cerdá J. 2003. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. Optimization and Engineering, 4(1-2). 7-22 [https://doi.org/10.1023/A:1021856229236](https://doi.org/10.1023/A%3A1021856229236)

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjukoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering*.*30(6-7). 913-946.<https://doi.org/10.1016/j.compchemeng.2006.02.008>.

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

Patil, B. P., Fukasawa, R., Ricardez-Sandoval, L. A., 2015. Scheduling of operations in a large-scale Scientific services facility via multicommodity flow and an optimization-based algorithm. Industrial and Engineering Chemistry Research*.* 54(5). 1628-1639. <https://doi.org/10.1021/ie503660f>.

Santos F. Fukasawa, R., Ricardez-Sandoval, L., 2018. An integrated personnel allocation and machine scheduling problem for industrial size multipurpose plants. IFAC-PapersOnLine. 51(18). 156-161 <https://doi.org/10.1016/j.ifacol.2018.09.292>

Shaik M. A., Janak S. L., Floudas C. A., 2006. Continuous-time models for short-term scheduling of multipurpose batch plants: a comparative study. Industrial & Engineering Chemistry Research, 45(18). 6190-6209 <https://doi.org/10.1021/ie0601403>

Shaik, M. A., Floudas, C. A., 2008. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. Computers & Chemical Engineering. 32(1-2). 260-274. <https://doi.org/10.1016/j.compchemeng.2007.05.007>.

Shaik, M. A., Floudas, C. A., 2009. Novel Unified Modeling Approach for Short-Term Scheduling. Industrial & Engineering Chemistry Research*.* 48(6). 2947-2964. [https://doi.org/10.1021/ie8010726](https://doi.org/10.1016/j.compchemeng.2007.05.007).

Sundaramoorthy, A., Karimi, I. A., 2005. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. Chemical engineering science*.* 60(10). 2679-2702. <https://doi.org/10.1016/j.ces.2004.12.023>.

Sursarla N., Li J., Karimi I., 2010. A novel approach to scheduling multipurpose batch plants using unit-slots. AIChE Journal, 56(7). 1859-1879 <https://doi.org/10.1002/aic.12120>

Tang. Q. H., Li. J., Floudas C. A., Deng M. X., Yan Y. B., Xi Z. H., Chen. P. H., Kong J. Y., 2012. Optimization framework for process scheduling of operation-dependent automobile assembly lines. Optimization Letters, 6(4). 797-824 https://doi.org/10.1007/s11590-011-0303-5

1. To whom correspondence should be addressed. jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622 [↑](#footnote-ref-1)