Front. Chem. Sci. Eng. 2019, 13(4): 784–802
https://doi.org/10.1007/s11705-019-1858-4

RESEARCH ARTICLE

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

Nikolaos Rakovitis[1], Nan Zhang[1], Jie Li (✉)[1], Liping Zhang[2]

1 Centre for Process Integration, School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, UK
2 Department of Industrial Engineering, School of Machinery and Automation, Wuhan University of Science and Technology, Wuhan 430081, China

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

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

1 Introduction

Nowadays, it is more important than ever for multipurpose batch process industry to maximize their productivity by simultaneously minimize their costs, fuel and raw material consumption and ecological footprint to be able to survive in a highly competitive market. Developing optimal schedules is one of the main tools that multipurpose batch process industry can utilize to optimize their processes. Although heuristics-based and spreadsheet-based methods are often used to generate schedules, they are restricted to simple batch processes and often produce suboptimal schedules. Mathematical programming especially mixed-integer programming approaches have been received much attention in the past three decades because they can be used for more complicated batch processes and often provide optimal schedules. Before developing mathematical models, it is crucial to well represent the multipurpose batch process. Two representations have been proposed including state-task network and resource-task network representations. The state-task network representation (STN) is proposed by Kondili et al. [1] in which all materials in the process are represented by states and processing operations in units are treated as tasks. While states are represented with circles (state nodes), tasks are depicted with rectangles (task nodes). The connections between states and task nodes are depicted with arrows. No resources such as processing units, storage tanks, utilities and manpower are demonstrated in the STN representation. Therefore, the resource-task network representation (RTN) is proposed by Pantelides [2] in which resources used by tasks are explicitly included. Based on the STN and RTN representations, several modelling approaches have been proposed for optimal scheduling of multipurpose batch processes resulting in a great number of mathematical models in the last three decades [3–7]. These modelling approaches include discrete-time [1,8,9], and continuous-time modelling approaches. The continuous-time modelling approaches include slot-based [10–12], global event-based [13–15], unit-specific event-based [16–19] and sequence-based modelling approaches [20–22]. The slot-based modelling approaches can be further classified into process-slot [11,12] and unit-slot [12,23] modelling approaches.

In the discrete-time modelling approach, the scheduling horizon is divided into time intervals of uniform or nonuniform lengths, where the start and end times of each

Received December 1, 2018; accepted May 9, 2019

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

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

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

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

2 Problem description

A general multipurpose batch process facility including J ($j = 1, 2,\ldots, J$) processing units such as reactors, separators and heaters. The STN representation of a multipurpose batch process facility is presented in Fig. 1.

These units are used to produce P ($p = 1, 2,\ldots, P$) final products using F ($f = 1, 2,\ldots, F$) feeds. I ($i = 1, 2,\ldots, I$) tasks will be processed in the processing units. Each processing unit can process $\mathbf{I}_j$ tasks. At each time, at most one task can be processed in a processing unit. Besides final products, intermediate states are also produced. There are total S ($s = 1, 2,\ldots, S$) states including feeds, intermediate states, and final products. The feeds are denoted as $\mathbf{S}^{R}$, the intermediate states are denoted as $\mathbf{S}^{IN}$, and the final products are included in the set of $\mathbf{S}^{FP}$. The proportion of each state s produced or consumed by a task i in a unit j is denoted by $\rho_{i,j,s}$. While positive values of $\rho_{i,j,s}$ denote production of state s during the processing of task i in unit j, negative values of $\rho_{i,j,s}$ denote consumption of state s during the processing of task i in unit j. After production, each batch is allowed to be mixed with other batches or split into several batches for further processing. Some intermediate states are also allowed to be recycled back if necessary. Each intermediate state has its dedicated storage. If the storage capacity for an intermediate state is unlimited, then it is called unlimited intermediate storage (UIS) policy. If the storage capacity is limited or finite, then it is called finite intermediate storage (FIS) policy. If there is no intermediate storage, then it is called no intermediate storage (NIS) policy. In this paper, we assume UIS for all states including intermediate states, feeds and final products. After production in a processing unit, an intermediate state may or may not be allowed to remain in this processing unit. If an intermediate state has to be transferred immediately to storage or other processing units after production, it is called zero wait (ZW) policy. If an intermediate state is allowed to remain in a processing unit with unlimited time, then it is called unlimited wait (UW) policy. If an intermediate state is allowed to be held in a processing unit with a certain time, then it is called limited wait (LW) policy. In this paper, we also assume UW policy for all intermediate states. By introducing this, the scheduling problem can be stated as follows, Given: (1) STN representation of a multipurpose batch facility; (2) J units, unit capacities, suitable tasks and their processing times; (3) S states, the portion of states produced or consumed from a task in a processing unit; (4) Product prices; (5) Scheduling horizon. Determine: (1) Optimal production schedule involving task allocations, start and end timings, sequences and batch sizes; (2) Inventory profiles. Operating rules: (1) At most one task can be processed in a processing unit at any time; (2) Batch mixing and splitting is allowed. Assumptions: (1) All parameters are deterministic; (2) The processing time of a task in a processing unit depends on a fixed processing time (denoted as α_{ij} plus a variable process time based on the batch sizes, which is denoted as $(\beta_{ij} \cdot b_{ij})$; (3) Unlimited feed materials are available; (4) Unlimited storage policy for all states; (5) Unlimited resources where required are available; (6) Unlimited wait policy for intermediate states.

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

3 Motivating example

Let consider an example whose STN representation is depicted in Fig. 2. In this example, a raw material S1 is converted into a final product S3 through two tasks (i.e., I1 and I2) in two processing units (J1 and J2). The scheduling horizon is 9 h. The objective is to maximize the productivity of product S3. All relevant data for this example are given in Table 1. We use the mathematical model of Shaik and Floudas [25] to solve this example. We obtain the optimal solution of 500.00 cu using 2 events. The optimal schedule is illustrated in Fig. 3. As seen from Fig. 3, the task I1, processed in unit J1, produces 100.00 cu of S2 at event point N1, which is further processed at task

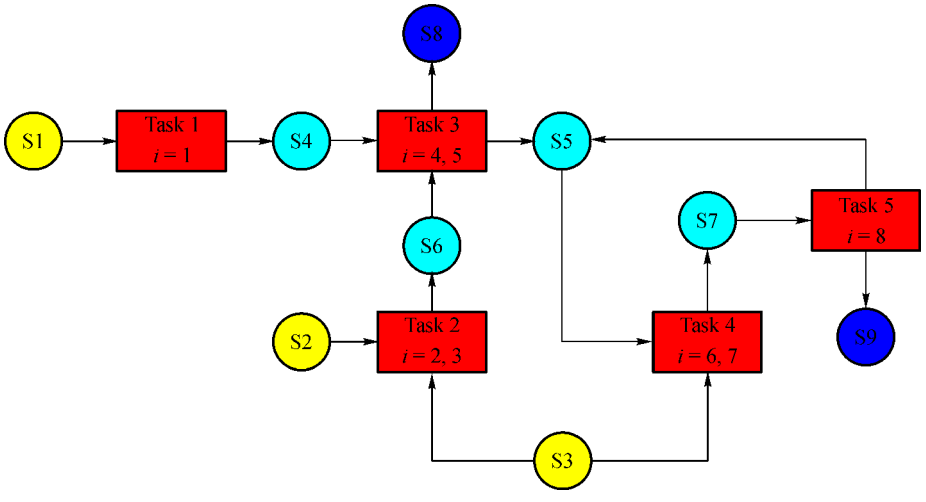

Fig. 1 STN representation of a multipurpose batch process facility (Example 2).

I2 in unit J2 to produce final product S3 with 100.00 cu at event point N2. This is because the task I2 in unit J2 is a consuming task of S2 and the task I1 in unit J1 is a production task for S2. Therefore, the task I2 must always start at event point N2 since the production task I1 take place at event point N1 based on the model of Shaik and Floudas [25]. However, we can use one event point for the optimal schedule through analysis. Figure 4 illustrates the optimal schedule with only one event point. From Fig. 4, it can be observed that the consuming task I2 takes place at the same event point as the production task I1, but not in real time. In real time, task I2 still takes place after I1 is completed. By doing this, we can reduce one event point required for generating the optimal solution. As discussed previously, the model size and computational performance largely depend on the number of event points required. This motivates us to develop new mathematical formulations for scheduling of multipurpose batch facilities by allowing consuming and production tasks related to the same states take place at the same event points to reduce the number of event points required.

4 Definition of recycling tasks

Despite the fact that allowing all related production and consumption tasks take place at the same event points can potentially reduce the number of event points and increase computational efficiency, we could obtain suboptimal solutions in some cases by allowing all production and consumption tasks related to the same states to take place at the same event points. Consider the following example which is depicted in Fig. 5 and Table 2. If production and consumption tasks related to the same states are not allowed to take place at the same event points, the optimal productivity of 1656 cu is generated with four event points from the model of Shaik and Floudas [25]. However, if all production and consumption tasks related to the same states are allowed to take place at the same event points, then the suboptimum productivity of 1511 cu is generated. This occurs from tasks I3, I4 and I5. Note that tasks I4 and I5 produce two states S2 and S4 and task I3 consumes state S2 and produces state S3. If these tasks (i.e., tasks I3, I4, and I5) are allowed to take place at the same event points, it is not possible for these tasks to take place at the same time in real time, which leads to suboptimum solutions.

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

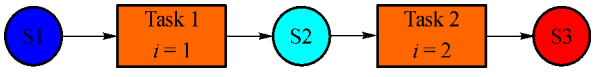

Fig. 2 STN representation of the motivating example.

Table 1 Data for motivating example

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

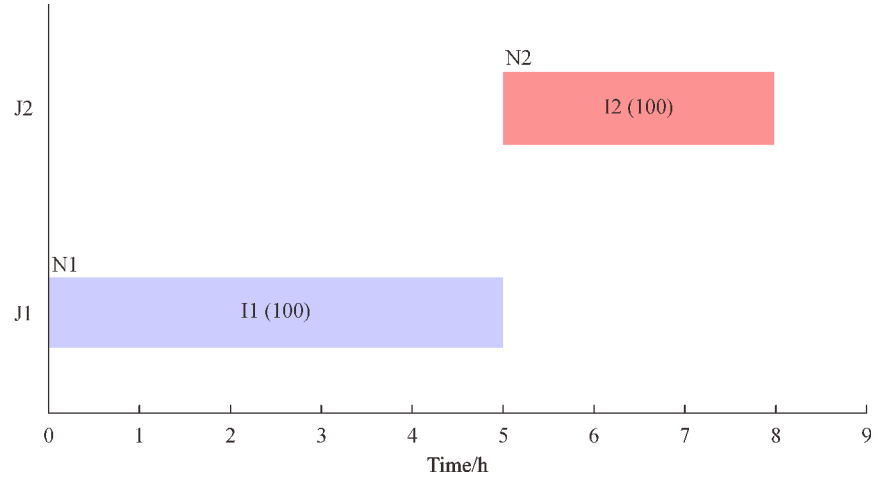

Fig. 3 Optimal schedule for the motivating example using two event points from the model of Shaik and Floudas [25].

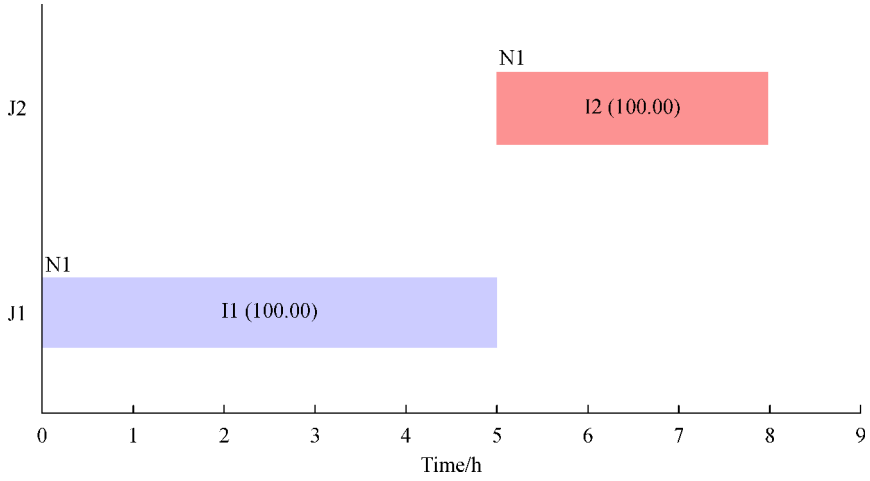

Fig. 4 Optimal schedule for the motivating example using one event point.

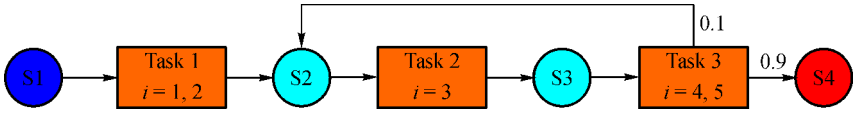

Fig. 5 STN representation of motivating example 2.

Table 2 Results for motivating example 2[a)]

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

a) SF: the model of Shaik and Floudas [25]; T-S: the revised model of Shaik and Floudas [25] allowing all production and consumption tasks related to the same states take place at the same event points; CPU: Central processing unit; RMILP: relaxed mixed-integer linear program.

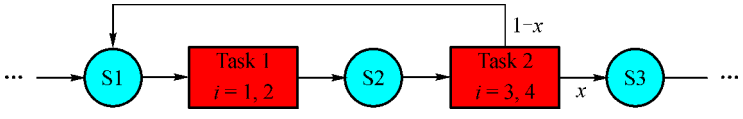

Fig. 6 Illustration of recycling tasks where all tasks are recycling tasks.

5 Mathematical formulation

5.1 Time representation

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

5.2 Model M1

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

5.2.1 Allocation constraints

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

$$\sum_{i \in I_j} \sum_{n-\Delta n \leqslant n' \leqslant n} \sum_{n \leqslant n'' \leqslant n'+\Delta n} w_{i,j,n',n''} \leqslant 1 \quad \forall j,n. \quad (1)$$

5.2.2 Capacity constraints

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

$$B_{i,j}^{\min} w_{i,j,n,n'} \leqslant b_{i,j,n,n'} \leqslant B_{i,j}^{\max} w_{i,j,n,n'}$$

$$\forall j,i \in \mathbf{I}_j, n \leqslant n' \leqslant n + \Delta n. \tag{2}$$

5.2.3 Material balance

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

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

$$+ \sum_j \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^C)} \rho_{i,j,s} \sum_{n \leqslant n' \leqslant n+\Delta n} b_{i,j,n,n'} \quad \forall s,n = 1, \tag{3}$$

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

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

$$+ \sum_j \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^C)} \rho_{i,j,s} \sum_{n \leqslant n' \leqslant n+\Delta n} b_{i,j,n,n'} \quad \forall s,n > 1. \tag{4}$$

5.2.4 Processing duration constraints

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

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

$$\forall j,i \in \mathbf{I}_j, n \leqslant n' \leqslant n + \Delta n. \tag{5}$$

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

5.2.5 Sequencing constraints

Same or different tasks in the same unit: An event point n on a processing unit j must always start after its previous event point on the same unit finishes.

$$T_{j,n+1}^{\mathrm{s}} \geqslant T_{j,n}^{\mathrm{f}} \quad \forall j, n < N. \tag{6}$$

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

$$T_{s,n} \leqslant T_{s,n+1} \quad \forall s \in \mathbf{S}^{\mathrm{IN}}, n. \tag{7}$$

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

$$T_{s,n} \geqslant T_{j,n}^{\mathrm{f}} - M \left(1 - \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P)} \sum_{n-\Delta n \leqslant n' \leqslant n} w_{i,j,n',n} \right)$$

$$\forall s \in \mathbf{S}^{\mathrm{IN}}, j, \sum_{i \in \mathbf{I}_j} \rho_{s,i} > 0, n. \tag{8}$$

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

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

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

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

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

$$\forall s \in \mathbf{S}^{\mathrm{IN}}, j, \sum_{j'} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}^P_S \cap \mathbf{I}^R)} \rho_{s,i'} > 0, n. \quad (10)$$

5.2.6 Objectives

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

$$z = \sum_s p_s \sum_j \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_S)} \sum_n \sum_{n \leqslant n' \leqslant n+\Delta n} \rho_{i,j,s} \cdot b_{i,j,n,n'}. \quad (11)$$

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

$$MS \geqslant T^{\mathrm{f}}_{j,n} \quad \forall j,\ n = N. \quad (12)$$

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

$$ST_{s,n} + \sum_j \sum_{i \in (I^P_S \cap I^R)} \rho_{i,j,s} \sum_{n-\Delta n \leqslant n' \leqslant n} b_{i,j,n',n} \geqslant D_s$$

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

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

5.3 Model M2

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

5.3.1 Allocation constraints

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

$$\sum_{i \in \mathbf{I}_j} \sum_{n-\Delta n \leqslant n' \leqslant n} \sum_{n \leqslant n'' \leqslant n'+\Delta n} w_{i,n',n''} \leqslant 1 \quad \forall j, n. \quad (14)$$

5.3.2 Capacity constraints

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

$$B^{\min}_i w_{i,n,n'} \leqslant b_{i,n,n'} \leqslant B^{\max}_i w_{i,n,n'}$$

$$\forall j, i \in \mathbf{I}_j, n \leqslant n' \leqslant n + \Delta n. \quad (15)$$

5.3.3 Material balance constraints

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

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

$$+ \sum_{i \in \mathbf{I}^C_S} \rho_{i,s} \sum_{n \leqslant n' \leqslant n+\Delta n} b_{i,n,n'} \quad \forall s, n = 1, \quad (16)$$

$$ST_{s,n} = ST_{s,n-1} + \sum_{i \in \mathbf{I}^P_S, i \notin \mathbf{I}^{\mathrm{R}}} \rho_{i,s} \sum_{n-\Delta n \leqslant n' \leqslant n} b_{i,n',n}$$

$$+ \sum_{i \in (\mathbf{I}^P_S \cap \mathbf{I}^R)} \rho_{i,s} \sum_{n-1-\Delta n \leqslant n' \leqslant n-1} b_{i,n',n-1}$$

$$+ \sum_{i \in \mathbf{I}^C_S} \rho_{i,s} \sum_{n \leqslant n' \leqslant n+\Delta n} b_{i,n,n'} \quad \forall s, n > 1. \quad (17)$$

5.3.4 Processing duration constraints

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

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

$$\forall\ i \in \mathbf{I}_j, n \leqslant n' \leqslant n + \Delta n. \tag{18}$$

5.3.5 Sequencing constraints

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

$$T^{\mathrm{s}}_{i,n+1} \geqslant T^{\mathrm{f}}_{i,n} \quad \forall i \in \mathbf{I}_j, n. \tag{19}$$

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

$$T^{\mathrm{s}}_{i,n+1} \geqslant T^{\mathrm{f}}_{i',n} \quad \forall j, i \in \mathbf{I}_j, i' \in \mathbf{I}_j, i \neq i', n < N. \tag{20}$$

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

$$T^{\mathrm{s}}_{i,n} \geqslant T^{\mathrm{f}}_{i',n} - M\left(1 - \sum_{n-\Delta n \leqslant n' \leqslant n} w_{i',n',n}\right)$$

$$\forall j \neq j',\ i \in (\mathbf{I}^C_S \cap \mathbf{I}_j),\ i' \in (\mathbf{I}^P_S \cap \mathbf{I}_{j'}),\ i' \notin \mathbf{I}^R, n, \tag{21}$$

$$T^{\mathrm{s}}_{i,n+1} \geqslant T^{\mathrm{f}}_{i',n} - M\left(1 - \sum_{n-\Delta n \leqslant n' \leqslant n} w_{i',n',n}\right)$$

$$\forall j \neq j',\ i \in (\mathbf{I}^C_S \cap \mathbf{I}_j),\ i' \in (\mathbf{I}^P_S \cap \mathbf{I}_{j'} \cap \mathbf{I}^R), n < N. \tag{22}$$

5.3.6 Tightening constraint

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

$$\sum_{i \in \mathbf{I}_j} \sum_{n} \sum_{n \leqslant n' \leqslant n+\Delta n} (\alpha_i \cdot w_{i,n,n'} + \beta_i \cdot b_{i,n,n'}) \leqslant H \quad \forall j. \tag{23}$$

5.3.7 Objectives

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

$$z = \sum_{s} p_s \sum_{i \in \mathbf{I}^P_S} \sum_{n} \sum_{n \leqslant n' \leqslant n+\Delta n} \rho_{i,s} b_{i,n,n'}. \tag{24}$$

The second objective is to minimize makespan.

$$MS \geqslant T^{\mathrm{f}}_{i,n} \quad \forall i,\ n = N. \tag{25}$$

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

$$ST_{s,n} + \sum_{i \in (I^P_S \cap I^R)} \rho_{i,s} \sum_{n-\Delta n \leqslant n' \leqslant n} b_{i,n',n} \geqslant D_s$$

$$\forall s \in \mathbf{S}^p, n = N. \tag{26}$$

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

6 Computational studies

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

The computational results from **M1** and **M2** are presented in Tables 15–20. While Tables 15–19 present the results from **M1** and **M2** with maximization of productivity as the objective, Table 20 presents the results from both models with minimization of makespan as the objective. From Tables 15–20, it can be observed that both **M1** and **M2** models are able to generate optimum solutions using less number of event points, which leads to smaller model sizes and less computational time. For instance, both **M1** and **M2** models require two event points less than the model of Shaik and Floudas [25] to generate the optimal solutions in all instances in Example 1 (see Table 15). More

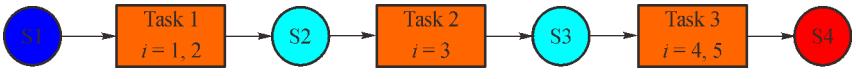

Fig. 7 STN representation of Example 1.

Table 3 Data for Example 1

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

Table 4 Data for Example 2

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

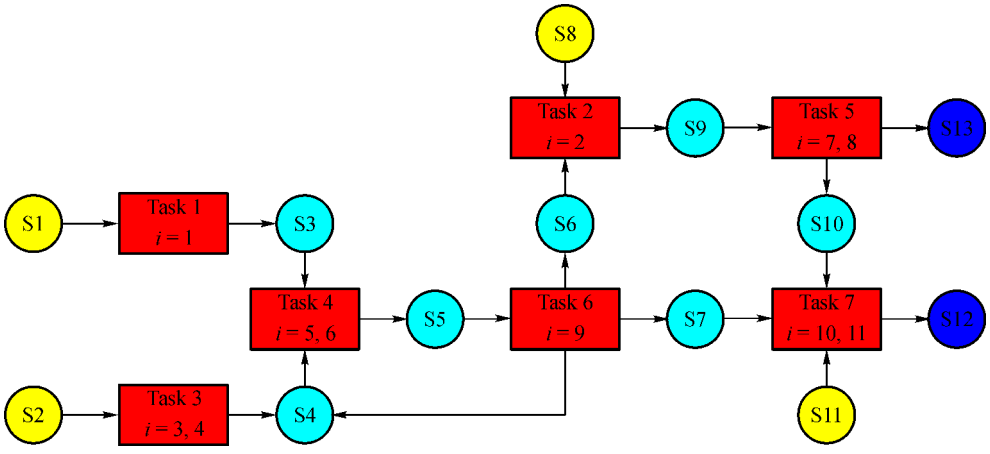

Fig. 8 STN representation of Example 3.

Table 5 Data for Example 3

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

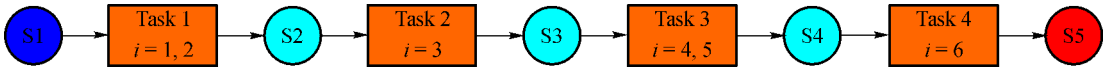

Fig. 9 STN representation of Example 4.

Table 6 Data for Example 4

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

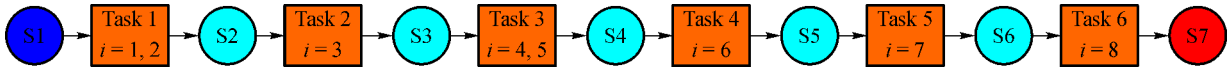

Fig. 10 STN representation of Example 5.

Table 7 Data for Example 5

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

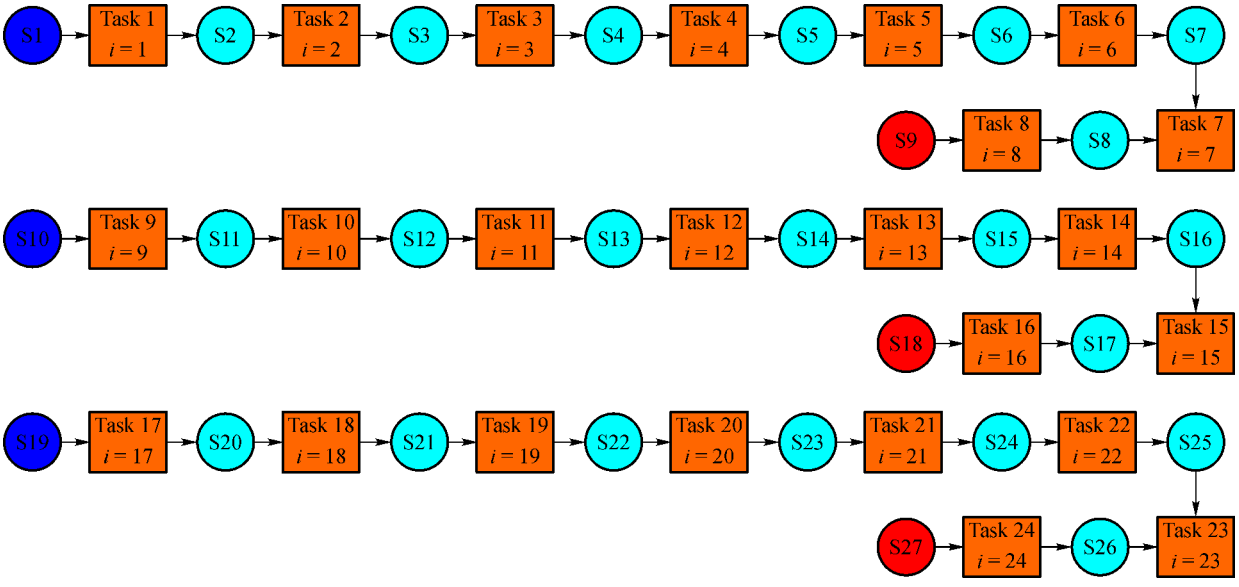

Fig. 11 STN representation of Example 6.

Table 8 Data for Example 6

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

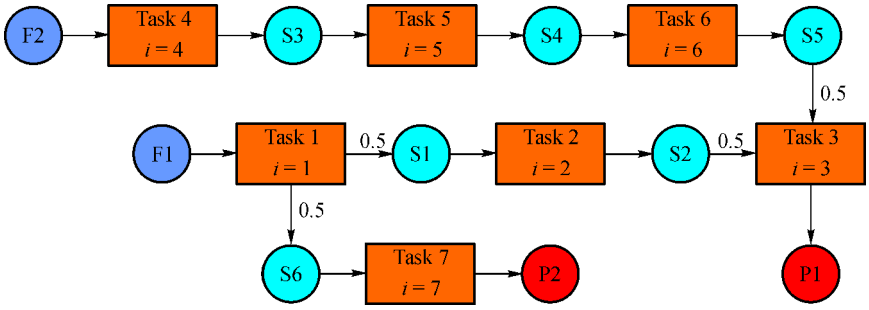

Fig. 12 STN representation of Example 7.

Table 9 Data for Example 7

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

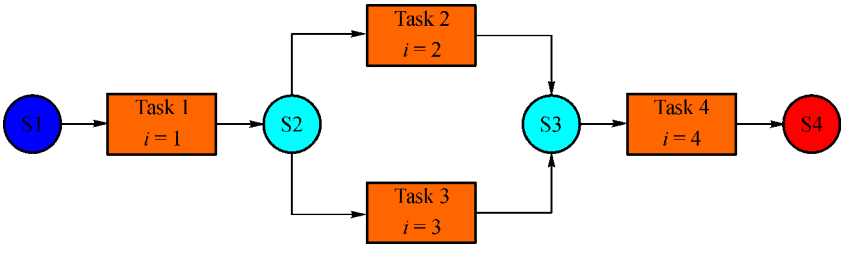

Fig. 13 STN representation of Example 8.

Table 10 Data for Example 8

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

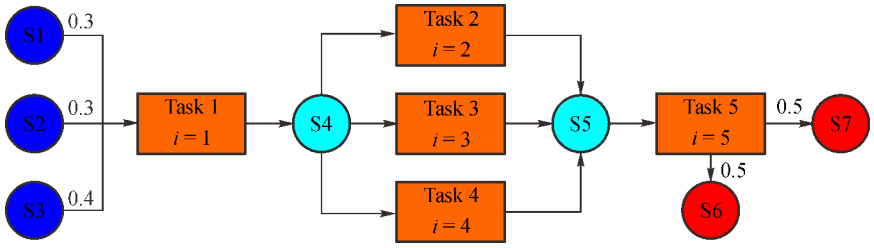

Fig. 14 STN representation of Example 9.

Table 11 Data for Example 9

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

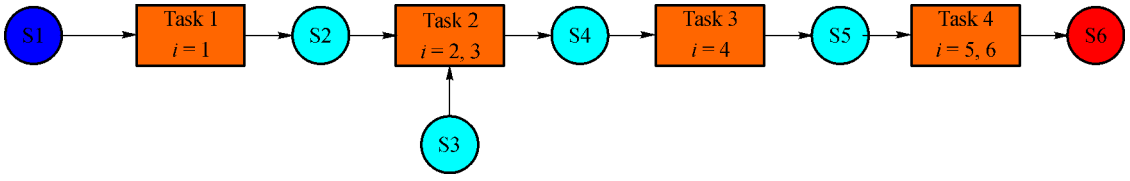

Fig. 15 STN representation of Example 10.

Table 12 Data for Example 10

Task	Processing unit	α_i	β_i	$B_i^{\min}$	$B_i^{\max}$
1	1	17.333	0.866	0	20
2	2	2.667	0.133	0	20
3	3	2.667	0.133	0	20
4	4	4.000	0.200	0	20
5	5	5.333	0.266	0	20
6	6	5.333	0.266	0	20

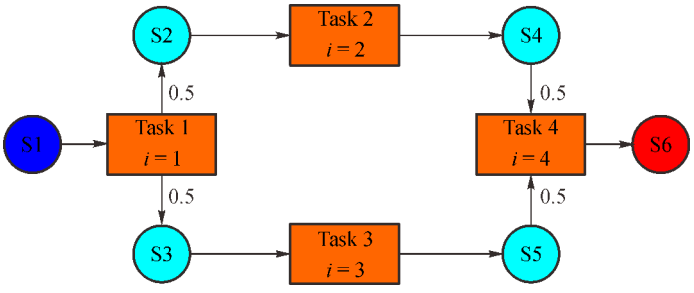

Fig. 16 STN representation of Examples 11 and 12.

Table 13 Data for Example 11

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

Table 14 Data for Example 12

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

Table 15 Computational results for Example 1 using maximization of productivity as objective[a)]

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

a) $\Delta n = 0$ for all cases

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

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

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

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

Table 18 Computational results for Examples 4–7 using maximization of productivity as objective[a)]

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

a) $\Delta n = 0$ for all cases

Table 19 Computational results for Examples 8–12 using maximization of productivity as objective[a)]

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

a) $\Delta n = 0$ for all cases

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

Example	Model	Event points	CPU time /s	RMILP /h	MILP /h	Discrete variables	Continuous variables	Constraints
1a (H = 50 h)	**SF**	14	11.45	24.24	27.88	70	268	394
$D_{s4} = 2000$	**M2**	12	5.30	25.36	27.88	60	230	342
	M1	12	6.96	24.24	27.88	60	254	383
1b (H = 100 h)	**SF**	23	7.50	48.47	52.07	115	439	646
$D_{s4} = 4000$	**M2**	21	5.70	50.06	52.07	105	401	594
	M1	21	4.81	48.47	52.07	105	443	671
2a(H = 50 h)	**SF**	9	96.00	10.78	19.34	72	301	515
$D_{s8} = 200$	**M2**	9	28.78	10.78	19.34	72	301	515
$D_{s9} = 200$	**M1**	9	46.58	18.68	19.34	72	265	425
2b(H = 100 h)	**SF**	19	3600[a)]	45.57	46.31	152	631	1105
$D_{s8} = 500$	**M2**	19	3600[b)]	45.57	46.31	152	631	1107
$D_{s9} = 400$	**M1**	19	3600[c)]	45.57	46.31	152	555	905
3a (H = 50 h)	**SF**	7	0.187	11.07	13.37	77	327	572
$D_{s12} = 100$	**M2**	7	0.250	11.07	13.37	77	327	572
$D_{s13} = 200$	**M1**	7	0.374	11.25	13.37	77	306	501
3b (H = 50 h)	**SF**	10	0.515	12.50	17.03	110	465	827
$D_{s12} = 250$	**M2**	10	0.374	12.76	17.03	110	465	827
$D_{s13} = 250$	**M1**	10	0.359	14.27	17.03	110	435	723

a) Relative gap 1.10%; b) Relative gap 1.43%; c) Relative gap 1.58%. Note $\Delta n = 0$ for all cases

specifically, in Example 1d, the model of Shaik and Floudas [25] requires 9 event points, whereas both models **M1** and **M2** require 7 event points, resulting in a reduction in binary variables by 20% (45 vs. 35). The optimal schedule for Example 1d using the model **M1** is illustrated in Fig. 17. From Fig. 17, it can be observed that the related production and consumption tasks take place at the same event points because all production tasks in this example are treated as non-recycling tasks. For instance, task I1 processed in unit J1 is a non-recycling task, which takes place at event point

N1. Its related consumption task is task I3 processed in unit J3 since this task I3 consumes state S2 produced from task I1, which also takes place at the same event point N1.

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

From Tables 16–17, we can observe that the proposed formulations **M1** and **M2** require the same number of event points with the model of Shaik and Floudas [25] for Examples 2–3. For these examples, **M1** and **M2** do not reduce the computational time too much because of the same number of event points required. It should be noted though that when tasks have to span over multiple event points, then model **M1** can significantly reduce the computational time. This main reason may come from the Constraint (5) which is tighter than those in Shaik and

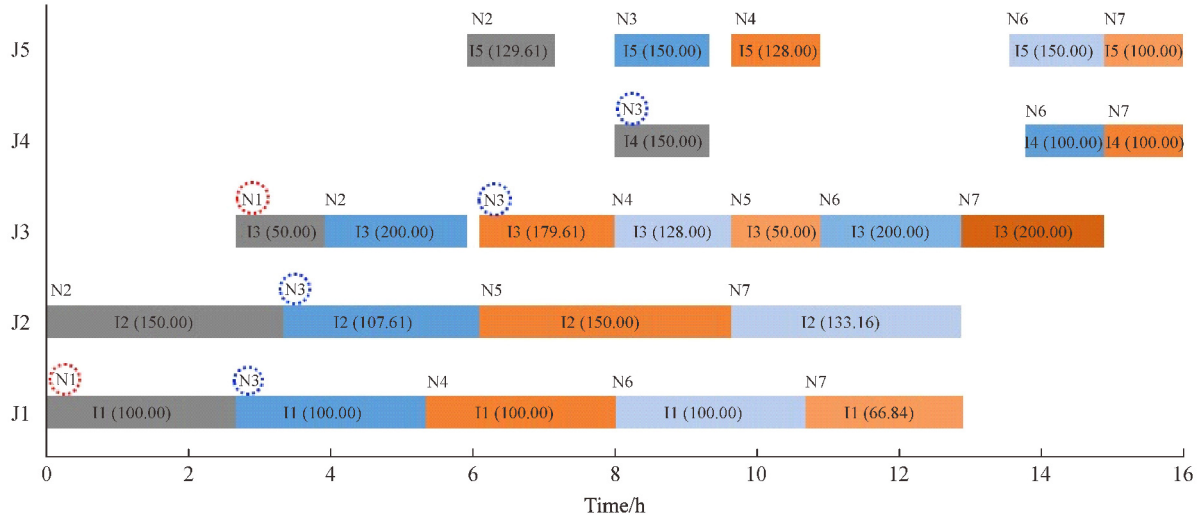

Fig. 17 Optimal schedule of Example 1d using the model **M1**, with maximization of productivity as the objective.

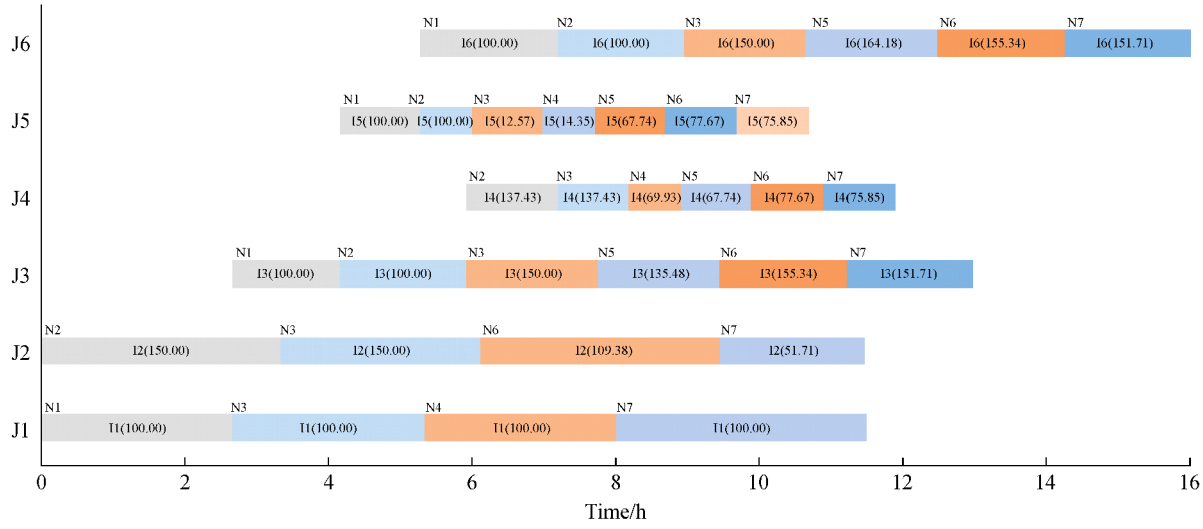

Fig. 18 Optimal schedule for Example 4 using the model **M1** with maximization of productivity as the objective.

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

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

Floudas [25] when a task has to span over multiple event points. For instance, **M1** requires 49% less computational time than the model of Shaik and Floudas [25] (5.41 s vs. 2.78 s) to solve Example 2b and 72% less computational time (3159 s vs. 892 s) for Example 3b. On the other hand, even though **M2** require slightly less computational time than the model of Shaik and Floudas [25] for both Example 2b (5.41 s vs. 5.10 s) and 3b (3159 s vs. 3141 s), it requires 46% more computational time than the model **M1** (2.78 s vs. 5.10 s) to solve Example 2b and 72% more computational time (892 s vs. 3141 s) to solve Example 3b. Therefore, it can be concluded that the proposed model **M1** is the most efficient.

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

7 Conclusions

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

magnitude in most cases. In the future, we will extend the proposed models **M1** and **M2** to solve other more complex intermediate storage policies such as FIS and NIS.

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

Nomenclature

Task-specific event-based model

Indices

i,i': tasks

j,j': units

n,n', n'': event points

s: states

Sets

I: tasks

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

$\mathbf{I}_S^{\mathrm{C}}$: tasks that consume state s

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

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

J: units

N: event points

S: states

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

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

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

Parameters

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

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

D_S: demand of state s

H: scheduling horizon

p_s: price of state s

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

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

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

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

Binary Variables

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

Continuous Variables

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

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

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

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

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

Unit-specific event-based model

Indices

i,i': tasks

j,j': units

n,n', n'': event points

s: states

Sets

I: tasks

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

$\mathbf{I}_S^{\mathrm{C}}$: tasks that consume state s

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

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

J: units

N: event points

S: states

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

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

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

Parameters

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

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

D_S: demand of state s

H: scheduling horizon

p_s: price of state s

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

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

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

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

Binary variables

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

Continuous variables

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

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

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

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

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

References

1. Kondili E, Pantelides C C, Sargent R W H. A general algorithm for short-term scheduling of batch operations-I MILP formulation. Computers & Chemical Engineering, 1993, 17(2): 211–227
2. Pantelides C. Unified frameworks for optimal process planning and scheduling. Proceedings of the Second Conference on Foundations of Computer Aided Operations, 1994, 253–274
3. Floudas C A, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. Computers & Chemical Engineering, 2004, 28(11): 2109–2129
4. Méndez C A, Cerdá J, Grossmann I E, Harjukoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers & Chemical Engineering, 2006, 30(6-7): 913–946
5. Li J, Susarla N, Karimi I A, Shaik M A, Floudas C A. An analysis of some unit-specific event-based models for the short-term scheduling of noncontinuous processes. Industrial & Engineering Chemistry Research, 2010, 49(2): 633–647
6. Maravelias C T. General framework and modeling approach classification for chemical production scheduling. AIChE Journal. American Institute of Chemical Engineers, 2012, 58(6): 1812–1828
7. Harjunkoski I, Maravelias C, Bongers P, Castro P, Engell S, Grossmann I, Hooker J, Méndez C, Sand G, Wassick J. Scope for industrial application of production scheduling models and solution methods. Computers & Chemical Engineering, 2014, 62(5): 161–193
8. Velez S, Maravelias C T. Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling. Computers & Chemical Engineering, 2013, 53(11): 70–85
9. Lee H, Maravelias C T. Discrete-time mixed integer programming models for short-term scheduling in multipurpose environments. Computers & Chemical Engineering, 2017, 107: 171–183
10. Pinto J M, Grossmann I E. A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research, 1995, 34(9): 3037–3051
11. Sundaramoorthy A, Karimi I A. A simpler better slot-based continuous-time formulation for short-term scheduling in multi-purpose batch plants. Chemical Engineering Science, 2005, 60(10): 2679–2702
12. Susarla N, Li J, Karimi I. A novel approach to scheduling multipurpose batch plants using unit-slots. AIChE Journal. American Institute of Chemical Engineers, 2010, 56(7): 1859–1879
13. Zhang X, Sargent R W H. The optimal operation of mixed production facilities-A general formulation and some approaches for the solution. Computers & Chemical Engineering, 1996, 20(6-7): 897–904
14. Castro P, Barbosa-Póvoa A P F D, Matos H. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research, 2001, 40(9): 2059–2068
15. Maravelias C T, Grossmann I E. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research, 2003, 42(13): 3056–3074
16. Ierapetritou M G, Floudas C A. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial & Engineering Chemistry, 1998, 37(11): 4341–4359
17. Li J, Floudas C. Optimal event point determination for short-term scheduling of multipurpose batch plants via unit-specific event-based continuous-time approaches. Industrial & Engineering Chemistry Research, 2010, 49(16): 7446–7469
18. Tang Q H, Li J, Floudas C A, Deng M X, Yan Y B, Xi Z H, Chen P H, Kong J Y. Optimization framework for process scheduling of operation-dependent automobile assembly lines. Optimization Letters, 2012, 6(4): 797–824
19. Li J, Xiao X, Floudas C A. Integrated gasoline blending and order delivery operations: Part I. Short-term scheduling and global optimization for single and multi-period operations. AIChE Journal. American Institute of Chemical Engineers, 2016, 62(6): 2043–2070
20. Méndez C A, Cerdá J. Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. Computers & Chemical Engineering, 2000, 24(2-7): 369–376
21. Hui C, Gupta A, van der Meulen H A J. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers & Chemical Engineering, 2000, 24(12): 2705–2717
22. Méndez C A, Cerdá J. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. Optimization and Engineering, 2003, 4(1-2): 7–22
23. Li J, Karimi I A. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. Industrial & Engineering Chemistry Research, 2011, 50(15): 9156–9174
24. Shaik M A, Janak S L, Floudas C A. Continuous-time models for short-term scheduling of multipurpose batch plants: A comparative study. Industrial & Engineering Chemistry Research, 2006, 45(18): 6190–6209
25. Shaik M, Floudas C. Novel unified modeling approach for short-term scheduling. Industrial & Engineering Chemistry Research, 2009, 48(6): 2947–2964
26. Shaik M, Vooradi R. Short-term scheduling of batch plants: Reformulation for handling material transfer at the same event. Industrial & Engineering Chemistry, 2017, 56(39): 11175–11185