Scheduling Multistage Batch Plants with Sequence-Dependent Changeovers

Pedro M. Castro and Augusto Q. Novais
Dept. de Modelação e Simulação de Processos, Instituto Nacional de Engenharia, Tecnologia e Inovação, 1649-038 Lisboa, Portugal

DOI 10.1002/aic.11799
Published online June 29, 2009 in Wiley InterScience (www.interscience.wiley.com).

This article deals with the optimal short-term scheduling of multistage batch plants with parallel units and sequence-dependent changeovers, together with the optimal selection of the number and size of batches to be produced. A new resource-task network-based, multiple time-grid continuous-time formulation is proposed that explicitly considers a virtual, shared, intermediate storage unit per stage to keep track of material transfer between processing units belonging to consecutive stages of production. Adequate material transfer is implicitly ensured through mass balances and timing constraints relating the times of event points of dissimilar grids. The new formulation is compared with a conceptually different approach from another research group. The results for several example problems show that the new formulation is tighter, typically requiring fewer event points to find the global optimal solution, and is significantly more efficient computationally. The results also show that for single batch problems other approaches are preferable. © 2009 American Institute of Chemical Engineers *AIChE J,* 55: 2122–2137, 2009

Keywords: optimization, mathematical modeling, mixed integer programming, continuous-time, unit-specific

Introduction

Scheduling in the process industry has received considerable attention during the last 15 years and extensive reviews can be found in the literature.[1–3] Because of the great variety of aspects that need to be taken into consideration, many approaches have been proposed to handle specific problem types. However, the main developments have resulted from more general approaches that can effectively handle a large number of problem features. Important landmarks are the unified frameworks for process representation, i.e., state-task networks[4] (STN) and resource-task networks (RTN).[5] Scheduling models are first characterized by the representation of time, be it discrete or continuous. Discrete-time models[4,5] divide the time horizon into a fixed number of equal length intervals and can sometimes be a good option. However, most of the times, the characteristics of the problem compels us to employ a continuous-time model, which can be either time grid[6–12] or sequence based.[13–15]

Nowadays, it is clear that single time-grid, continuous-time models based on unified frameworks[8,9,11] are the most general for batch plants, since they can consider resource constraints other than equipment, together with various storage policies and multiple batches of a product where batch mixing/splitting is allowed. The one by Castro et al.[9] can even handle continuous processing tasks. However, it is also true that they are not usually the most efficient for some classes of problems. Recent detailed computational studies have shown that multiple time-grid (also known as unit-specific) formulations can be orders of magnitude faster in problems arising from multipurpose batch plants[12,16] and also from multistage multiproduct batch plants,[17,18] both under an unlimited intermediate storage policy. These studies have also shown that, even for the same class of problems, there is no single best approach and that the most effective strategy seems to be the use of a set of competitive models in parallel.

Correspondence concerning this article should be addressed to P. M. Castro at pedro.castro@ineti.pt

Traditional approaches for multistage multiproduct batch plants[13–15,17–19] consider that the number and size of batches is known a priori.[20] This information may come from a procedure that works with the customer orders like the first step of the approach proposed by Gupta and Karimi[21] for the scheduling of a two-stage batch plant with intermediate storage. Different batches of the same product are treated as distinct entities that maintain their identity throughout the processing stages similarly to batches belonging to different products. Hence, they can be viewed as multiproduct single batch approaches. Decoupling batching from scheduling decisions may not be an easy task in processes with parallel units of dissimilar capacities.[22] Furthermore, it often leads to suboptimal solutions since one cannot take advantage of batch mixing/splitting to capitalize on the different capacities from one stage to the next.

Maravelias and coworkers[20,22] have shown that considering batching and scheduling simultaneously yields better solutions than those obtained by existing two-stage methods. Their formulations are sequence based and can be viewed as explicit batching approaches, since there are binary variables that identify whether a particular batch of a certain order has been selected. They were first proposed[20] with global precedence 5-index sequencing variables in problems involving fixed processing times and were later[22] extended to cases with sequence-dependent changeover costs, through the use of immediate precedence sequencing variables, involving task durations that are batch size dependent.

A conceptually different type of explicit batching approach has been proposed by Castro et al.[23] for single-stage plants with fixed and equal batch sizes for dissimilar batches of the same product. Aggregated processing and changeover tasks were considered, each being characterized by an integer variable that gives the number of batches of a product to produce in a particular unit. It has been shown to be significantly better than its implicit batching counterpart that considers a single batch per task, mostly due to the need of fewer event points to find the global optimal solution.

The number of event points required to find the global optimal solution is indeed a very important performance indicator for time grid/slot-based continuous-time formulations. Studies[12,16,17] have shown that unit-specific approaches need fewer event points than single time-grid ones, which in practice is translated into the ability to tackle larger problems. Finding the adequate number of event points is in fact a challenge on its own, which is typically addressed by an iterative procedure where a single increase normally leads to an increase in computational effort of one order of magnitude. Thus, evaluating the performance for an excessively high number of event points[18] may be misleading.

Unit-specific, single batch, multistage formulations like the ones of Castro and Grossmann[17] and Liu and Karimi[18] are minimum event-point approaches[24] because (i) each task lasts a single time interval and (ii) there is no implicit relation between time points belonging to units of consecutive stages when modeling material transfer. Although 4-index binary variables are used in Ref. 18, since each unit is allocated to a particular stage, it is sufficient to use 3-index variables to identify the execution of a product in a unit, at a certain event point. The two models[17,18] differ mainly in the location of event points, which can be either at the start[17] or at the end of the time slot,[18] with the latter being a better option when release dates are not an issue. In problems involving multiple batches of a product, one needs to ensure not only that the timing between consecutive stages is adequate but also that there is enough intermediate in storage (and/or just finishing production) to be consumed by the subsequent task to take advantage of batch mixing and splitting. Both the multipurpose formulation of Shaik and Floudas (SF)[12] and the multistage one of Castro and Novais[24] rely on relationships among time points of dissimilar grids to efficiently model the mass balances. These have the effect of raising the number of event points needed to find the optimal solution, less so for the latter approach, which was found to be a better performer. Nevertheless, in single batch problems, the formulation of Castro and Grossmann[17] was the best of the three.

Another relevant and more recent work was undertaken by Erdirik-Dogan and Grossmann,[25] who have proposed a slot-based formulation for multiproduct batch plants with sequence-dependent changeovers. The work has been motivated by a real world application where the main difficulty is the handling of two-stage production with intermediate storage. The model is classified as being multistage but the network structure is more complex than the one assumed in multistage models. More specifically, the equipment units can be allocated to tasks belonging to both stages, a feature normally associated to multipurpose plants. The proposed model is quite complex and, for a certain time period, features 4-index binary variables to determine if a slot of a certain unit is completed before a slot of another unit starts. Big-*M* constraints are then used to relate the various slots starting points. To overcome the fact that the model becomes computationally expensive to solve, mainly because the required number of time slots is not known a priori, an efficient bilevel decomposition approach is proposed.

Although models with 3-index binary variables can address multistage problems with sequence-dependent changeovers simply by changing one set of constraints[12] (when compared with the case of no changeovers), it is also true that they increase both in size due to the addition of one more time index, and in complexity, due to the addition of a big-*M* term. One valid alternative[19] is to increase the number of binary variables through the definition of combined processing and changeover tasks featuring an additional product index. This approach, which has been shown to be competitive, has the advantage of leading to tighter timing constraints since the required changeovers are explicitly modeled.

This article proposes a new model based on 4-index binary variables for the short-term scheduling of multistage batch plants with multiple product batches and sequence-dependent changeovers. It uses insights from our previous work on single batch problems[19] and on multiple batch problems without changeovers.[24] When compared with the former, this conceptually new approach uses the same set of binary variables but their domain is significantly larger, which will have a detrimental effect on computational performance. The new formulation is thoroughly compared with those of Castro et al.[19] and SF[12] for the single batch case, through the solution of seven example problems taken from the literature, and for three different objective functions: total cost, total earliness, and makespan minimization. Three benchmark example problems are proposed to evaluate the

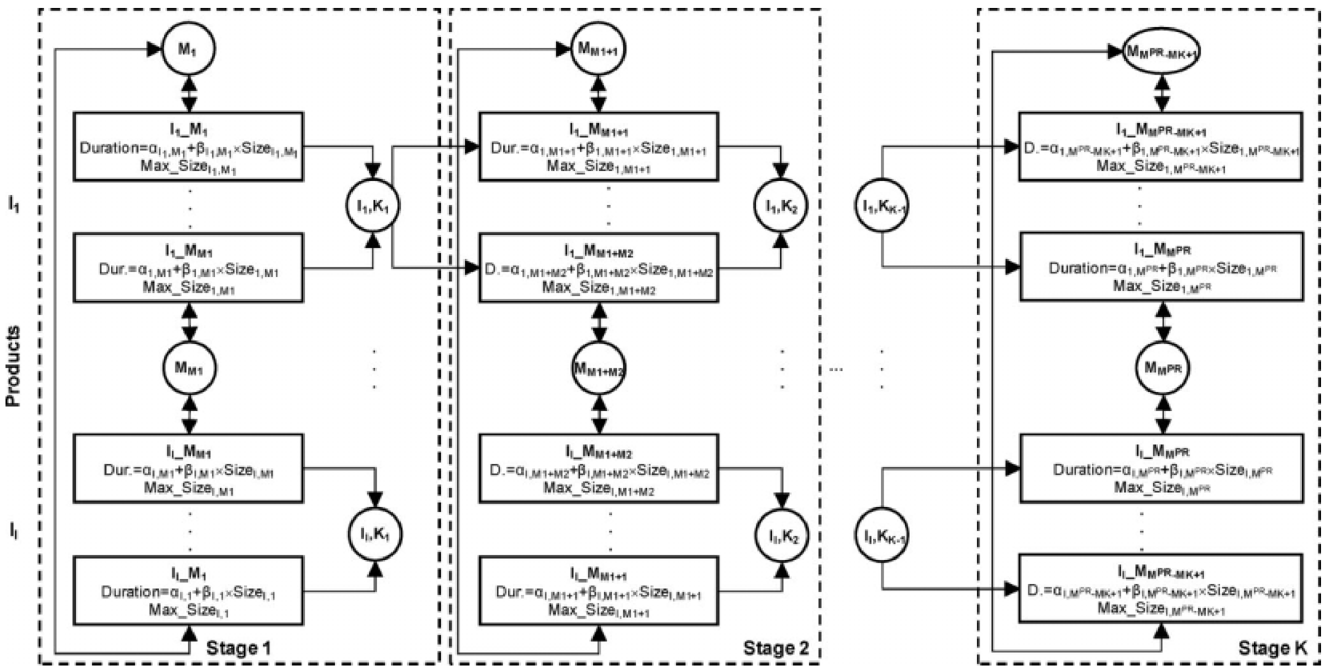

Figure 1. Schematic of a multistage multiproduct batch plant.

performance of the multiple batch models. These are solved for a couple of scenarios and objectives of revenue maximization and makespan minimization.

Problem Definition

Given a multistage, multiproduct plant with $k \in K$ processing stages, $i \in I$ products, and $m \in M^{\mathrm{PR}}$ equipment units, the goal is to determine the number and size of batches for each product, the assignment of batches to units and the sequence of batches in each unit so as to meet a certain objective. Processing times are specified by a constant $\alpha_{i,m}$ plus a term $\beta_{i,m}$ that is proportional to the batch size $\mathrm{Size}_{i,m}$, which cannot exceed a certain maximum value, $\mathrm{Max\_Size}_{i,m}$, with parameter $\mathrm{cl}_{i,i',m}$ holding the sequence-dependent changeover times.

A particular equipment unit can handle all products belonging to set I_m and is allocated to a single stage, with set M_k including those belonging to Stage k. Unlimited, product dedicated intermediate storage, and unlimited wait (UIS/UW) policies[26] are assumed, while transfer times between units are negligible. In general, multiple batches of product i can be produced and a different number of batches may be involved in dissimilar stages of production, i.e., batches can be split or mixed. As a special case, problems with fixed batch sizes (and fixed processing times) involving a single batch per product in every stage will be considered. In such problems, the products release r_i and due dates d_i are assumed to be known and are enforced as hard constraints.

Four alternative objective functions are tested. When maximizing revenue, the goal is to produce the most valuable and less time-consuming products, while meeting their minimum demands Δ_i. The selling prices v_i are given and the time horizon H is fixed. For single batch problems, common objectives also involving a fixed time horizon are as follows: total cost minimization, aiming to find the lowest possible product-unit assignment cost (given by $c_{i,m}$), and total earliness minimization that targets to end production of every product as close as possible to its due date. Finally, in makespan minimization, the total production time for a given product demand Δ_i is minimized.

RTN process representation

The process representation of a generic multistage multiproduct plant is given in Figure 1 in the form of a RTN. The plant resources (circles) consist of all equipment units and material states. Products are numbered from 1 to I, with I_{I} representing the last element of the set. The processing tasks, depicted as rectangles, identify the making of a product in a unit. These range from M_1 to $M_{M^{\mathrm{PR}}}$. Resources and tasks are represented within the confinement of the corresponding stage. As an example, in Stage 2 (K_2), product I_1 can be processed in parallel processing units ranging from M_{M_1+1} to $M_{M_1+M_2}$, with M_1 giving the number of units belonging to Stage 1. The corresponding tasks consume the material resource (I_1,K_1) and produce (I_1,K_2).

Review of Formulation of Shaik and Floudas

The scheduling formulation of SF[12] can handle multistage multiproduct problems as a special case of the generic multipurpose plant problem type. Basically, it is an updated version of the earlier work of Ierapetritou and Floudas[6] that now relies on the RTN instead of the STN process representation. We have recently[24] adapted the nomenclature of model SF to the multistage case and thoroughly compared the formulation to two other conceptually different multiple time-grid approaches, for problems without sequence-dependent changeovers. Because in the presence of sequence-dependent changeovers only one set of constraints needs to be changed,[12] in the interest of space only the model's most important features will be highlighted.

The SF formulation uses 3-index binary variables $N_{i,m,t}$ to identify the execution of product i in unit m at event point t, with the amount produced being given by the continuous extent variables $\xi_{i,m,t}$. Excess resource variables $R_{m,t}$ keep track of resource availability and are defined as binary, although they might also be of the continuous type. Variables $S_{i,m,t}$ are used to account for the material state of product i in storage unit m at event point t. Since we will be defining one storage unit per intermediate stage, the index m ($m \in M^{\mathrm{ST}} \wedge m \notin M^{\mathrm{PR}}$) is directly linked to index k and so the tasks and resources used are those represented in Figure 1. In terms of time representation, and even though the SF model is commonly known as unit specific, it should be more appropriately classified[24] as task specific as it does not rely on explicit time grids for the equipment units. Timing variables $T^{\mathrm{s}}_{i,m,t}$ keep track of the starting time of product i in unit m at event point t. Appropriate sequencing constraints relate the absolute times of consecutive event points involving (i) the same product in the same unit; (ii) different products in the same unit; and (iii) the same product in different units belonging to consecutive stages of production. In the presence of sequence-dependent changeovers, the second set of constraints needs to be disaggregated over another time index and big-M terms employed, as seen in Eq. 1, where the big-M parameter is the time horizon, H.

$$T^{\mathrm{s}}_{i,m,t} \geq T^{\mathrm{s}}_{i',m,t'} + \alpha_{i',m}N_{i',m,t'} + \beta_{i',m}\xi_{i',m,t'} + \mathrm{cl}_{i',i,m}N_{i,m,t} - H(1 - N_{i',m,t'}) - H\left(\sum_{\substack{t''\in T\\ t'<t''<t}}\sum_{i''\in I_{m,t''}} N_{i'',m,t''}\right)$$
$$\forall m \in M^{\mathrm{PR}}, t, t' \in T, t > t', i \in I_{m,t}, i' \in I_{m,t'}, i \neq i' \quad (1)$$

When compared with the constraint proposed by Liu and Karimi,[18] the additional time index (t') is required due to one important conceptual difference. Although Liu and Karimi's[18] model can enforce, without loss of generality, that two consecutive tasks have no empty slots between them, in the case of SF, intermediate unused slots are typically required. Thus, Eq. 1 ensures that if product i is processed in unit m at slot t and product i' in the same unit at point $t' = t - 2$, the starting time of i is greater than the ending time of i' plus the appropriate changeover time between i' and i if there is no product being processed at $t'' = t - 1$.

Reformulation of Model by Castro et al. (CGN)

Castro et al.[19] proposed two continuous-time formulations that are able to address the multistage multiproduct scheduling problem, wherever a single batch of every product is involved. Formulation CT4I is now revised and reformulated to use one less event point[24] and to reduce the integrality gap. In the following, its most relevant features and modifications will be highlighted.

Combined processing and changeover tasks

The first aspect is that formulation CGN uses combined processing and changeover tasks. Two product indices (i, i') are used to characterize such tasks with the first indicating the product being produced and the second the one that follows. Its duration will then comprise both the processing time of i and the required changeover between i and i', but not the processing time of i'. Figure 2 illustrates this aspect, where it can be seen that the output material state, e.g., (I_1,K_1), is produced at the end of the processing part of the task, whereas the equipment resource where the task is executed is made available only after its changeover part is completed. In reality, the different material states are not considered explicitly and this is why dotted circles are used to highlight such resources. Instead, timing variables $\mathrm{TD}_{i,k}$ are used to determine the transfer time of product i in Stage k. They must be greater than the product ending time in Stage k (Eq. 2) and lower than its starting time in Stage k + 1 (Eq. 3), where the 4-index binary variables $\overline{N}_{i,i',m,t}$ assign the execution of a combined task to a unit m and also to a time point t. Note that these are big-M constraints.

$$\mathrm{TD}_{i,k} \geq T_{t,m} + \sum_{\substack{i'\in I_m\\ i\in I_{i',m,t}}} \alpha_{i,m}\overline{N}_{i,i',m,t} - H\left(1 - \sum_{\substack{t'\in T\\ t'\geq t}}\sum_{\substack{i'\in I_m\\ i\in I_{i',m,t'}}} \overline{N}_{i,i',m,t'}\right)$$
$$\forall k \in K, m \in M_k, i \in I_m, t \in T, k \neq |K| \quad (2)$$

$$\mathrm{TD}_{i,k-1} \leq T_{t,m} + H\left(1 - \sum_{\substack{t'\in T\\ t'\leq t}}\sum_{\substack{i'\in I_m\\ i\in I_{i',m,t'}}} \overline{N}_{i,i',m,t'}\right)$$
$$\forall k \in K, m \in M_k, i \in I_m, t \in T, k \neq 1 \quad (3)$$

Location of event points

Formulation CGN is a unit-specific, event-based approach where one time grid is explicitly defined for each processing unit (Figure 3), with the timing variables $T_{t,m}$ holding the absolute time of event point t of the time grid linked to unit m. When compared with CT4I,[19] it features one less event point, the ending point of the last time interval, to better illustrate the fact that relevant event points are only those where tasks are started. The exact location of event points varies from unit to unit.

Figure 4 gives an overview on how the formulation works for a simple example comprising 4 products, 3 processing units, and 2 stages. One important property is that each combined task lasts a single time interval, that is, the required number of event points on each machine is equal to the number of products assigned to that machine. In this case, there is a single unit (M_3) allocated to the second stage, so four event points ($|T| = |I|$) are enough to guarantee global optimal solutions, as it was also noted by Liu and Karimi.[26] Although each task lasts one time interval, it does not necessarily mean that when starting at t they end exactly at event point t + 1. Note that task (I_1,I_2) in M_1 ends well before the second event point, whereas task (I_1,I_4) ends exactly at the second event point of the time grid linked to M_3. In general, the difference in time between two consecutive event points must be greater than the duration of the combined task being executed. This is ensured by Eq. 4 or by Eq. 5 whenever minimizing makespan. Note that MS is the variable defining

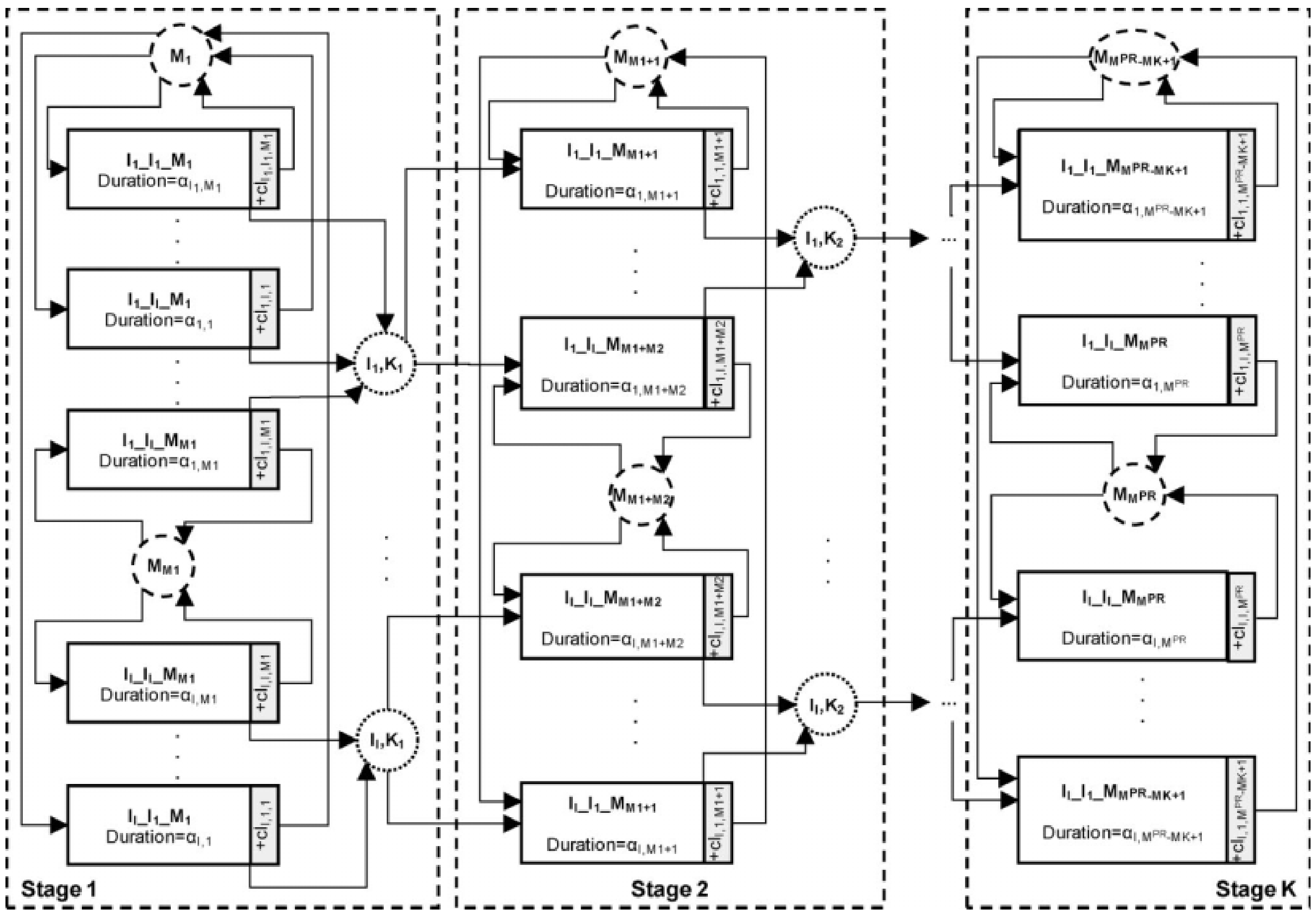

Figure 2. RTN representation of the process illustrating important features of model CGN.

the makespan, which replaces the time horizon (H) on the second term on the LHS when compared with Eq. 4. Nevertheless, recall that material transfer can occur immediately after the end of the processing part of the task (Eq. 2), as it is illustrated for I_1 (from M_1 to M_3) and I_4 (from M_2 to M_3).

$$T_{t+1,m}|_{t\neq|T|} + H|_{t=|T|} - T_{t,m} \geq \sum_{i'\in I_m}\sum_{i\in I_{i',m,t}} [\overline{N}_{i,i',m,t} \times (\alpha_{i,m} + \mathrm{cl}_{i,i',m})] \; \forall m \in M^{\mathrm{PR}}, t \in T \quad (4)$$

$$T_{t+1,m}|_{t\neq|T|} + MS|_{t=|T|} - T_{t,m} \geq \sum_{i'\in I_m}\sum_{i\in I_{i',m,t}} [\overline{N}_{i,i',m,t} \times (\alpha_{i,m} + \mathrm{cl}_{i,i',m})] \; \forall m \in M^{\mathrm{PR}}, t \in T \quad (5)$$

It is also important to emphasize that there is no direct relation between event points belonging to time grids of dissimilar units. Although highly unlikely due to timing constraints, it is perfectly possible for a particular product to start at the last event point in Stage k and at the first event point in Stage $k + 1$. To reduce solution degeneracy, tasks will be assigned in sequence from the first to the last event point. Although there are other possibilities for placing nonzero time slots,[26] placing them at the beginning of the grid is highly advantageous for defining the objective of total earliness minimization, see Eq. 6. By doing this, we know a priori that the absolute times of all unused time slots belonging to last stage units will converge to their upper bounds, H.

$$\min \sum_{i\in I} d_i - \sum_{t\in T}\sum_{m\in M_{|K|}} \left(T_{t,m} + \sum_{i'\in I_m}\sum_{i\in I_{i',m,t}} \alpha_{i,m}\overline{N}_{i,i',m,t} \right) - H[|I| - |M_{|K|}| \cdot |T|] \quad (6)$$

This task placing strategy is also conducive to reduce the domain of the binary variables (see Ref. 19 for a detailed explanation). The lowest time at which product i can start to be processed in unit m, $\mathrm{lb}_{i,m}$ is determined by Eq. 7. This is then used to define the lower bound on the absolute time of event point t of unit m, Eq. 8, which can be viewed as release date constraints for units belonging to the first stage. An upper bound, $\mathrm{ub}_{i,i',m}$ can also be calculated, Eq. 9, and enforced, Eq. 10. The latter is only active when there is a task starting at t in unit m; otherwise it is relaxed to its global upper bound, Eq. 11 (see also Figure 3). Equation 12

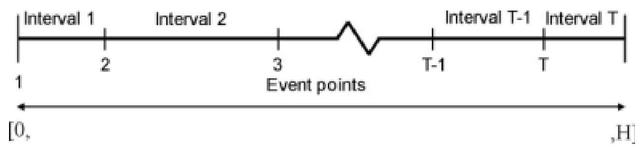

Figure 3. Underlying time grid for each equipment unit.

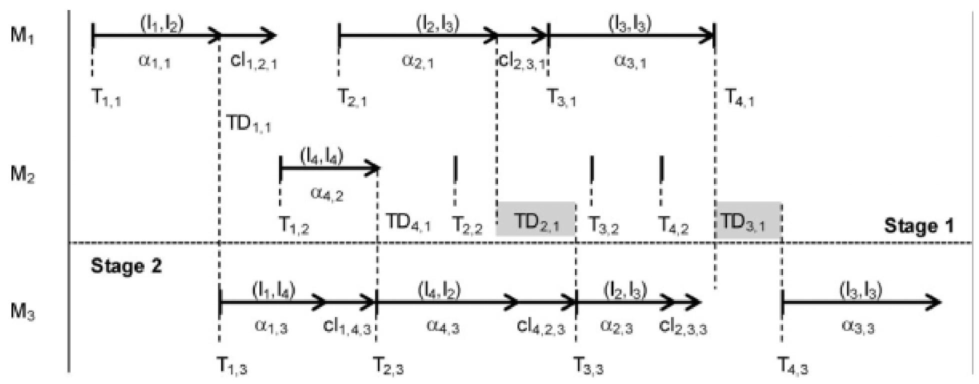

Figure 4. Illustration of formulation CGN ($|I| = 4$; $|M_{PR}| = 3$; $|K| = 2$; $|T| = 4$) with orders being assigned from the first to the last event point.

defines set $I_{i',m}$, which contains all products that can precede i' in unit m. Equation 13 defines the earliest time at which event point t of unit m can become active, $\text{lbm}_{m,t}(\text{cl}_{i,m}^{\min} = \min_{i' \neq i} \text{cl}_{i,i',m})$. Finally, all the information is combined into the definition of set $I_{i',m,t}$ (see Eq. 14) to generate the domain of binary variables $\overline{N}_{i,i',m,t}$. Note that the last time interval can only feature tasks with the same product index simply because there are no more slots to process any more tasks. In fact, the last task to be executed on a particular unit will feature a single product index, regardless of being executed on the first or last time slot, as is illustrated in Figure 4.

$$\text{lb}_{i,m} = r_i + \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k' \not> k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} \alpha_{i,m'} \ \forall i \in I, m \in M_i \tag{7}$$

$$T_{t,m} \geq \sum_{i' \in I_m} \sum_{i \in I_{i',m,t}} \overline{N}_{i,i',m,t} \text{lb}_{i,m} \ \forall m \in M^{\text{PR}}, t \in T \tag{8}$$

$$\text{ub}_{i,i',m} = \min\left(d_i - \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k' > k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} \alpha_{i,m'} - \alpha_{i,m}, d_{i'} - \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k' > k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} \alpha_{i',m'} - \alpha_{i,m} - \text{cl}_{i,i',m} - \alpha_{i',m}|_{i \neq i'} \right) \ \forall i, i' \in I, m \in M_i \cap M_{i'} \tag{9}$$

$$T_{t,m} \leq \sum_{i' \in I_m} \sum_{i \in I_{i',m,t}} \overline{N}_{i,i',m,t} \text{ub}_{i,i',m} + H\left(1 - \sum_{i' \in I_m} \sum_{i \in I_{i',m,t}} \overline{N}_{i,i',m,t}\right) \ \forall m \in M^{\text{PR}}, t \in T \tag{10}$$

$$T_{t,m} \leq H = \max_{i \in I_m} d_i \ \forall m \in M^{\text{PR}}, t \in T \tag{11}$$

$$I_{i',m} = \{ i \in I_m : \text{ub}_{i,i',m} \geq \text{lb}_{i,m} \} \ \forall i' \in I, m \in M_{i'} \tag{12}$$

$$\text{lbm}_{m,t} = \min_{i \in I_m} \text{lb}_{i,m} + \min_{i \in I_m} (\alpha_{i,m} + \text{cl}_{i,m}^{\min}) \cdot (t-1) \ \forall m \in M^{\text{PR}}, t \in T \tag{13}$$

$$I_{i',m,t} = \{ i \in I_{i',m} : \text{ub}_{i,i',m} \geq \text{lbm}_{m,t} \wedge (t \neq |T| \vee i = i') \} \ \forall i' \in I, m \in M_{i'}, t \in T \tag{14}$$

Equipment cleaning states

Before a combined task can be executed in an equipment unit, the unit must be at the corresponding cleaning state. Cleaning states result from the disaggregation of equipment unit resources, which do not need to be used (unlike in the original formulation[19] or in the SF model). This is the reason why unit resources in Figure 2, which are typical resources in RTN-based formulations, are circled by dash rather than solid lines. Excess resource variables $C_{i,m,t}$ are used to account for cleaning states, with $C_{i,m}^0$ defining the initial one. For unit M_1, the several possible cleaning states are illustrated in Figure 5. Generally speaking, tasks (i,i',m) will consume state (i,m) at the start and produce state (i',m) at the end so that a task (i',i'',m) can immediately follow. The other novelty in CGN is that tasks with a single product index will not be regenerating the state they are consuming (e.g., $I_1\_I_1\_M_1$), simply because there is no need while single batch problems are being considered (Eq. 15). Note that in the third term on the RHS of the excess resource balances for cleaning states, the summation is for $i \neq i'$ (Eq. 16). This modeling option, which is known to be a powerful constraint in RTN-based formulations, enables the setting up to zero of all excess state variables (see Eq. 17), and will force without using additional constraints such as in Castro et al.[19] and Liu and Karimi[26]: (i) tasks to be allocated from the first to the last time interval; (ii) the last active task to have a single product index. A particular unit will normally start at a certain state unless there are no tasks allocated to it (Eq. 18).

$$\sum_{t \in T} \sum_{m \in M_k} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} = 1 \ \forall i \in I, k \in K \tag{15}$$

$$C_{i,m,t} = C_{i,m}^0|_{t=1} + C_{i,m,t-1}|_{t \neq 1} + \sum_{\substack{i' \in I_{i,m,t-1} \\ i' \neq i}} \overline{N}_{i',i,m,t-1} - \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} \ \forall i \in I, m \in M_i, t \in T \tag{16}$$

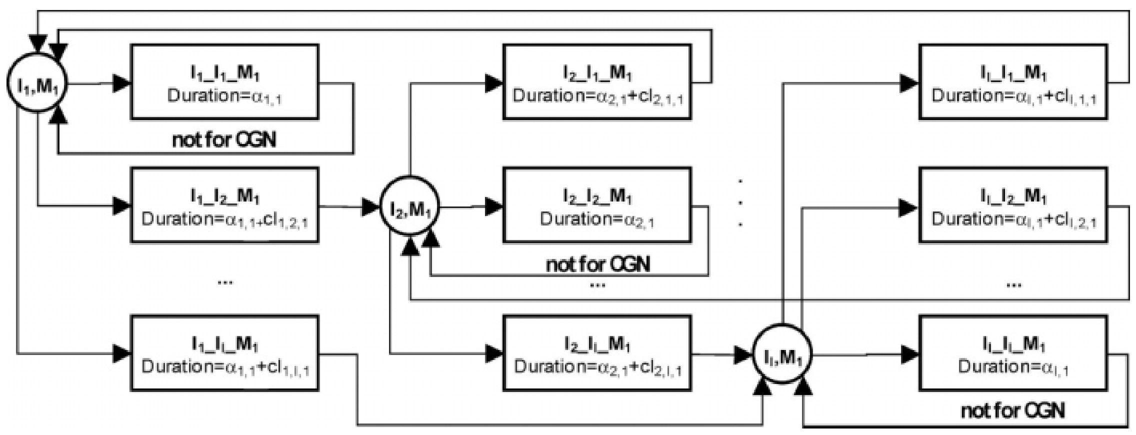
I1,M1
I1_I1_M1 Duration=α1,1
not for CGN
I1_I2_M1 Duration=α1,1+cl1,2,1
I1_II_M1 Duration=α1,1+cl1,I,1
I2,M1
I2_I1_M1 Duration=α2,1+cl2,1,1
I2_I2_M1 Duration=α2,1
not for CGN
I2_II_M1 Duration=α2,1+cl2,I,1
II,M1
II_I1_M1 Duration=αI,1+clI,1,1
II_I2_M1 Duration=αI,1+clI,2,1
II_II_M1 Duration=αI,1
not for CGN

Figure 5. Part of the RTN representation for unit M_1 showing all possible cleaning states.

$$C_{i,m,t} = 0 \; \forall i \in I, m \in M_i, t \in T \qquad (17)$$

$$\sum_{i \in I_m} C^0_{i,m} \leq 1 \; \forall m \in M^{\mathrm{PR}} \qquad (18)$$

Other constraints

For completeness, the remaining sets of constraints for formulation CGN are given next. Total cost minimization is achieved by Eq. 19, while for makespan minimization, Eq. 20 is used instead. The remaining four sets (Eqs. 21–24) are not mandatory but were found[19] to improve the model performance, where Eqs. 23 and 24 are to be used solely for the objective of makespan minimization.

$$\min \sum_{t \in T} \sum_{m \in M} \sum_{i' \in I_m} \sum_{i \in I_{i',m,t}} \overline{N}_{i,i',m,t} c_{i,m} \qquad (19)$$

$$\min MS \qquad (20)$$

$$\mathrm{TD}_{i,k} \geq r_i + \sum_{t \in T} \sum_{\substack{k' \in K \\ k' \leq k}} \sum_{\substack{m \in M_{k'} \\ m \in M_i}} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \alpha_{i,m} \overline{N}_{i,i',m,t} \quad \forall i \in I, k \in K, k \neq |K| \qquad (21)$$

$$\mathrm{TD}_{i,k} \leq d_i - \sum_{t \in T} \sum_{\substack{k' \in K \\ k' > k}} \sum_{\substack{m \in M_{k'} \\ m \in M_i}} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \alpha_{i,m} \overline{N}_{i,i',m,t} \quad \forall i \in I, k \in K, k \neq |K| \qquad (22)$$

$$MS \geq T_{t,m} + \sum_{i' \in I_m} \sum_{\substack{i \in I_{i',m,t} \\ k \neq |K|}} \left[\overline{N}_{i,i',m,t} \cdot \left(\alpha_{i,m} + \sum_{\substack{k' \in K \\ k' > k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} \alpha_{i,m'} \right) \right] + \sum_{\substack{t' \in T \\ t' \geq t}} \sum_{i' \in I_m} \sum_{\substack{i \in I_{i',m,t'} \\ k = |K|}} [\overline{N}_{i,i',m,t'} \cdot (\alpha_{i,m} + \mathrm{cl}_{i,i',m})] \quad \forall k \in K, m \in M_k, t \in T \qquad (23)$$

$$MS \geq \mathrm{TD}_{i,k} + \sum_{\substack{k' \in K \\ k' > k}} \sum_{\substack{m \in M_{k'} \\ m \in M_i}} \sum_{t \in T} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \alpha_{i,m} \overline{N}_{i,i',m,t} \quad \forall i \in I, k \in K, k \neq |K| \qquad (24)$$

New Multiple Time Grid Approach (CN)

The new unit-specific continuous-time approach shares some features with CGN and with the model recently proposed by Castro and Novais.[24] Like in CGN, 4-index binary variables $\overline{N}_{i,i',m,t}$ are used to account for the combined processing and changeover tasks, but now their domain ($I_{i',m,t}$) is larger simply because it can no longer be assumed that all time slots, up to the last active task on a particular unit, will be occupied (compare Eq. 25 with Eq. 14). This stems from the need to have mass balances on the product material states to handle multiple batch problems, which forces the establishment of a relationship among event points of dissimilar time grids, along the one proposed by Castro and Novais.[24] As a consequence, the number of event points needed to find the global optimal solution in single batch problems is normally larger for CN and also for SF when compared with CGN.[24]

$$I_{i',m,t} = \{ i \in I_m : t \neq |T| \vee i = i' \} \forall i' \in I, m \in M_{i'}, t \in T \qquad (25)$$

Material and equipment states

In the new formulation, CN, like in SF (and contrary to CGN), material states are handled explicitly through the use of excess variables $S_{i,m,t}$. It is assumed that states resulting from a particular stage exist in a virtual shared intermediate storage unit $m \in M^{\mathrm{ST}}$. In other words, there is a 1:1 correspondence between stages and storage units. This is illustrated in Figure 6, where storage units are numbered in sequence to the processing units, $M = \mathrm{M}^{\mathrm{PR}} \cup \mathrm{M}^{\mathrm{ST}}$. As an example, state ($I_1$,$K_1$) exists in the first storage unit $M_{\mathrm{M}^{\mathrm{PR}}+1}$. Storage units are highlighted by dotted circles to emphasize that they are not treated in the same way as processing units. The availability of the latter has to be accounted because they can be consumed and produced over time. Processing units are highlighted by dash circles to emphasize that they are disaggregated into their relevant cleaning states, as

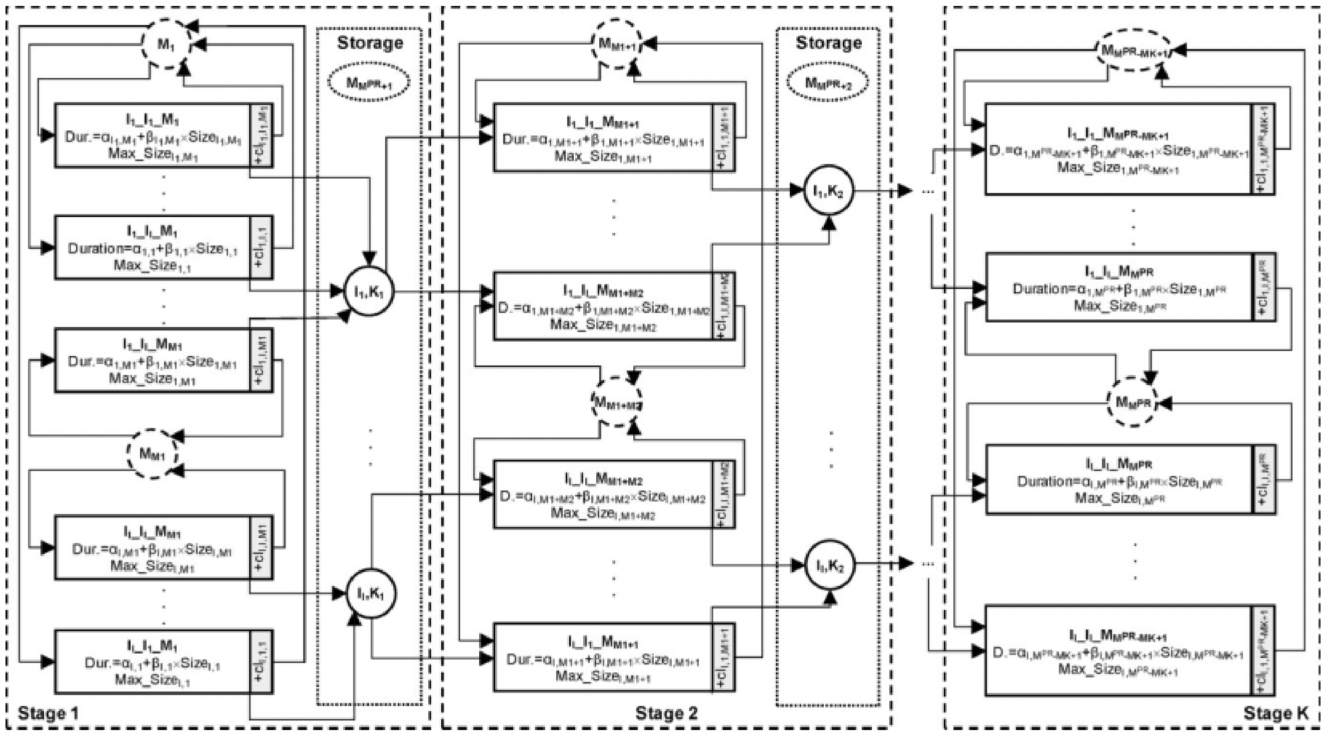
Figure 6. RTN representation of the process illustrating important features of model CN.

illustrated in Figure 5. Now, contrary to CGN, consecutive batches of a product may need to be processed in a particular unit, so combined tasks with a single product index must regenerate the state they consume. As a result, the constraints involving the excess variables $C_{i,m,t}$ and $C^0_{i,m}$ are slightly modified. In general, model CN uses Eqs. 26–28 rather than Eqs. 16–18, even though for single batch problems Eq. 16 is preferred over Eq. 26.

$$C_{i,m,t} = C^0_{i,m}|_{t=1} + C_{i,m,t-1}|_{t\neq 1} + \sum_{i'\in I_{i,m,t-1}} \overline{N}_{i',i,m,t-1} - \sum_{\substack{i'\in I_m \\ i\in I_{i',m,t}}} \overline{N}_{i,i',m,t} \ \forall i \in I, m \in M_i, t \in T \quad (26)$$

$$C_{i,m,t} \leq 1 \ \forall i \in I, m \in M_i, t \in T \quad (27)$$

$$\sum_{i\in I_m} C^0_{i,m} = 1 \ \forall m \in M^{\mathrm{PR}} \quad (28)$$

The excess resource balance constraints for material states are given in Eq. 29. It can be seen that the availability of material state m of product i at event point t increases by the amount generated by tasks executed in processing units m' belonging to the same stage ($m' \in M^{\mathrm{to}}_m$) and decreases by those executed in units m'' belonging to the next stage ($m'' \in M^{\mathrm{fr}}_m$), given by variable $\xi_{i,m'',t}$. The starting event point for such tasks is that of the constraint domain, unlike in SF, where the relevant event point for tasks consuming the material state under consideration is $t-1$.[12,24] It is assumed that the amount processed by the task is equal to the amount of material consumed, whereas the yield associated to the amount produced can be accounted through parameter $\overline{v}_{i,m}$.

$$S_{i,m,t} = S_{i,m,t-1} + \sum_{\substack{m'\in M^{\mathrm{to}}_m \\ i\in I_{m'}}} \overline{v}_{i,m'} \xi_{i,m',t} - \sum_{\substack{m''\in M^{\mathrm{fr}}_m \\ i\in I_{m''}}} \xi_{i,m'',t} \ \forall i \in I, m \in M^{\mathrm{ST}}, t \in T \quad (29)$$

Location of event points

The CN formulation is a unit-specific approach that uses one time grid for each equipment unit, like the one shown in Figure 3, not only for processing units, as in CGN, but also for the intermediate storage units. Timing variables $T_{t,m}$ hold the absolute time of event point t in unit m. Figure 7 gives an overview on how the formulation works for a simple two-stage problem and shows the relationship between the time grids of the intermediate storage unit and the processing units located before and after it ($|I| = 6$, $|M^{\mathrm{PR}}| = 6$, $|M^{\mathrm{ST}}| = 1$, $|K| = 2$, $|T| = 2$). Each combined task requires a single event point, with its starting time defining its exact location. Identically to CGN, two consecutive event points are spaced apart by an amount of time at least equal to the duration of the task taking place. Equations 30–31 differ from Eqs. 4 and 5 simply in the last term on the RHS, which accounts for batch size-dependent extents. The novelty results from the direct relation that is assumed between event point t from a particular storage unit, and event point t from processing units sending to and receiving material from it.

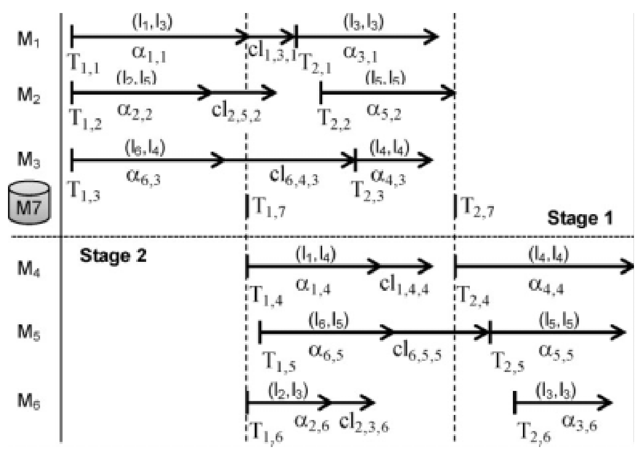

Figure 7. Overview of formulation CN for a simple two-stage problem.

$$T_{t+1,m}|_{t\neq|T|} + H|_{t=|T|} - T_{t,m} \geq \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} [\overline{N}_{i,i',m,t} \cdot (\alpha_{i,m} + \mathrm{cl}_{i,i',m})] + \sum_{i\in I_m} \beta_{i,m}\xi_{i,m,t} \ \forall m \in M^{\mathrm{PR}}, t \in T \tag{30}$$

$$T_{t+1,m}|_{t\neq|T|} + MS|_{t=|T|} - T_{t,m} \geq \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} [\overline{N}_{i,i',m,t} \times (\alpha_{i,m} + \mathrm{cl}_{i,i',m})] + \sum_{i\in I_m} \beta_{i,m}\xi_{i,m,t} \ \forall m \in M^{\mathrm{PR}}, t \in T \tag{31}$$

It is clear that material transfer must be allowed to occur immediately after the end of the processing part of the task. The task finishing last of all those starting at event point t in same-stage units, will set the time of event point t belonging to the grid of the corresponding storage unit. In Figure 7, it is the ending time of task (I_1,I_3), starting at $t = 1$ in M_1, that is equal to $T_{1,7}$ (absolute time of the first event point of storage unit M_7). On the other hand, the second event point is set by task (I_5,I_5) in M_2. Naturally, CN allows for task (I_2,I_3) in M_6 to start immediately after the end of task (I_2,I_5) in M_2. Likewise, for task (I_6,I_5) in M_5. However, such option would lead to an increase in the number of event points required and tasks (I_1,I_3) in M_1 and (I_6,I_4) in M_3 would have to start no earlier than the second event point, even though they could start at exactly the same absolute time. The general constraint is given in Eq. 32 and does not involve big-M terms.

$$T_{t,m'} \geq T_{t,m} + \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} \alpha_{i,m}\overline{N}_{i,i',m,t} + \sum_{i\in I_m} \beta_{i,m}\xi_{i,m,t} \ \forall m' \in M^{\mathrm{ST}}, m \in M^{\mathrm{PR}}, m \in M^{\mathrm{to}}_{m'}, t \in T \tag{32}$$

It is not only the highest ending processing time of units sending material to storage that is contributing to set the event point location of the storage unit but also the lowest starting time of units receiving material from storage. This is ensured by a simple set of constraints, also involving only three indices, see Eq. 33. In terms of the first event point, note in Figure 7 that $T_{1,7} = T_{1,4} = T_{1,6}$, while task ($I_6$,$I_5$) starts in M_5 at a later time.

$$T_{t,m'} \leq T_{t,m} \ \forall m' \in M^{\mathrm{ST}}, m \in M^{\mathrm{PR}}, m \in M^{\mathrm{fr}}_{m'}, t \in T \tag{33}$$

It is important to clarify at this point that unit M_7 in Figure 7, is virtual, like generally all predefined intermediate storage units. Recall that unlimited, dedicated intermediate storage is assumed, which can be translated into the assumption of as many intermediate tanks as the number of products, all with unlimited capacity. Thus, we are not concerned with the number of products that are stored simultaneously in M_7, which is translated in the observation that I_2 and I_6 are stored together before being transferred to Stage 2 (see Figure 7). Neither are we with the time at which storage tasks need to be performed.

The other set of constraints (Eq. 34) involving the timing variables is not mandatory, but makes the model more efficient for makespan minimization. It is significantly more complex than Eq. 23 because we need to account for the contribution of the batch size-dependent term on the task duration. Note that we cannot simply add the minimum values of $\beta_{i,m}$ of units belonging to subsequent stages, because the total amount processed may be split among existing parallel units.

$$\begin{aligned} MS \geq{}& T_{t,m} + \sum_{i'\in I_m} \sum_{\substack{i\in I_{i',m,t} \\ k\neq|K|}} [\overline{N}_{i,i',m,t} \cdot (\alpha_{i,m} + \sum_{\substack{k'\in K \\ k'>k}} \min_{\substack{m'\in M_{k'} \\ m'\in M_i}} \alpha_{i,m'})] \\ &+ \sum_{\substack{t'\in T \\ t'\geq t}} \sum_{i'\in I_m} \sum_{\substack{i\in I_{i',m,t'} \\ k=|K|}} [\overline{N}_{i,i',m,t'} \cdot (\alpha_{i,m} + \mathrm{cl}_{i,i',m})] \\ &+ \sum_{i\in I_m} \left[\xi_{i,m,t} \left(\beta_{i,m} + \overline{v}_{i,m} \sum_{\substack{k'\in K \\ k'>k}} \left[\left(\prod_{\substack{k''\in K \\ k<k''<k'}} \min_{\substack{m'\in M_{k''} \\ m'\in M_i}} \overline{v}_{i,m'} \right) \times \min_{\substack{m'\in M_{k'} \\ m'\in M_i}} \beta_{i,m'}/|M_{k'}| \right] \right) \right] \ \forall k \in K, m \in M_k, t \in T \end{aligned} \tag{34}$$

Other constraints

The CN model is completed with the constraints next described. For the objectives of total cost, makespan, or total earliness minimization, Eqs. 19, 20, and 6, respectively, can be used. For the multiple product batches problems, the objective of revenue maximization is accounted for by Eq. 35. Equation 36 ensures that the continuous extent variables are zero whenever the binary extent variables are zero and that the total amount processed does not exceed $V^{\max}_m$, the maximum capacity of the processing unit. Finally, Eq. 37 guarantees that all product demands are met.

$$\max \sum_{m\in M_{|K|}} \sum_{i\in I_m} \sum_{t\in T} v_i \cdot \overline{v}_{i,m} \cdot \xi_{i,m,t} \tag{35}$$

$$\xi_{i,m,t} \leq V^{\max}_m \sum_{\substack{i'\in I_m \\ i\in I_{i',m,t}}} \overline{N}_{i,i',m,t} \ \forall m \in M^{\mathrm{PR}}, i \in I_m, t \in T \tag{36}$$

$$\sum_{\substack{m\in M_{|K|} \\ i\in I_m}} \sum_{t\in T} \overline{v}_{i,m} \cdot \xi_{i,m,t} \geq \Delta_i \ \forall i \in I \tag{37}$$

Table 1. Single Batch Problems: Overview of Computational Performance (CPU) for Total Cost Minimization

Problem/Model	Optimum	CGN	CN	SF
P7C ($\|I\| = 8$, $\|M^{PR}\| = 6$, $\|K\| = 2$)	18	**1.44**	3.36	57.9
P8C ($\|I\| = 8$, $\|M^{PR}\| = 6$, $\|K\| = 2$)	1,087	**1.17**	4.76	497
P9C ($\|I\| = 8$, $\|M^{PR}\| = 6$, $\|K\| = 3$)	82	4,868	**1,149**	60,000[*]
P10C ($\|I\| = 8$, $\|M^{PR}\| = 6$, $\|K\| = 3$)	647	**17.5**	1,621[†]	7,172[‡]
P11C ($\|I\| = 12$, $\|M^{PR}\| = 6$, $\|K\| = 2$)	71	**39.1**	6,745	15,322
P12C ($\|I\| = 8$, $\|M^{PR}\| = 8$, $\|K\| = 4$)	125	60,000[§]	1,495[¶]	**5,183**[**]
P13C ($\|I\| = 15$, $\|M^{PR}\| = 4$, $\|K\| = 2$)	96	1,939	**631**	22,151

BPS, best possible solution at the time of termination; MRL, maximum resource limit exceeded; NS, no solution found; SO, suboptimal solution returned; OM, out of memory.
[*]MRL, BPS = 81.
[†]SO = 649.
[‡]SO = 649.
[§]MRL, NS, BPS = 122.3.
[¶]SO = 129.
[**]SO = 126.

Computational Studies

The performance of the three alternative formulations is now evaluated. It is important to note, at this point, that this study also included another formulation based on 3-index binary variables, which used the same material transfer concept of CN, but not an explicit modeling of equipment cleaning states like SF. However, the results showed this other formulation to be inferior in performance to CN and so it was not included in this article.

Most MILP models were implemented and solved in GAMS 22.5, using the CPLEX 10.2 solver on a Pentium-4 3.4 GHz processor, with 2 GB of RAM and running Windows XP Professional. The instances from PMB3, on the other hand, were solved in GAMS 22.8 (CPLEX 11.1) on an Intel Core2 Duo T9300 processor running at 2.5 GHz, with 4 GB of RAM and running Windows Vista Enterprise. All problems were solved to optimality (1E-6 relative tolerance) unless stated otherwise.

Single batch problems

The first set of problems includes single batch problems and is taken from the literature.[19] The seven example problems (P7–P13) have been thoroughly analyzed by multiple time-grid and sequencing-based continuous-time formulations, discrete-time, and constraint programming models. Their complexity ranges from 8 products in 6 units and 2 stages, to 15 products in 4 units and 2 stages, and further to 8 products in 8 units in 4 stages. Because the optimal solutions are known for the minimization of three alternative objective functions (*C*: total cost; *M*: makespan; *E*: total earliness), such problems provide a good testing environment. For comparative purposes, we have kept the original numbering and nomenclature.

Table 1 lists the results for total cost minimization, where the best formulation for a particular example problem is identified in bold. It can be seen that the CGN formulation was the best performer in four of the seven problems and the best one overall. It could find the optimal solution for all problems except P12C for which not even a feasible solution could be found after 60,000 CPUs. Note that this is the only 4-stage problem, so the failure is probably caused by the larger number of timing constraints that are required to determine the products transfer times (Eqs. 2 and 3). These results are consistent with those reported by Castro et al.[19] for formulation CT4I. The conceptually new model CN was better than CGN for P9C, P12C, and P13C, but was unable to find the optimal solution both for P10C and P12C. This is caused by the use of an insufficient number of event points. The model by SF[12] was the worst overall performer but it did manage to find the solution 126, the best of the group, for P12C.

We know from Castro and Novais[24] that in terms of the number of event points to find the global optimal solution, we have by definition $|T|_{CGN} \le |T|_{CN}$ and most of the times $|T|_{CN} \le |T|_{SF}$. The absolute difference is typically of one or two event points but, in the case of SF, it also depends on the number of stages. Thus, it was unexpected to find that, while CGN solved P10C with 5 event points, CN returned a suboptimal solution (649) from $|T| = 5$ up to 9 event points (for the upper bound, up to a maximum resource limit of 8000 CPUs), whereas SF returned 652 for $|T| = 6$ and 649 for $|T| = 7$. However, when the optimal solution in Figure 8 is considered, it is realized that a significant larger number of event points will be required by CN. This is because there are products allocated in Stage k to an event point of a lower index than the one where they are allocated in Stage $k - 1$, a situation that is not possible in model CN. This occurs for all except Product 3.

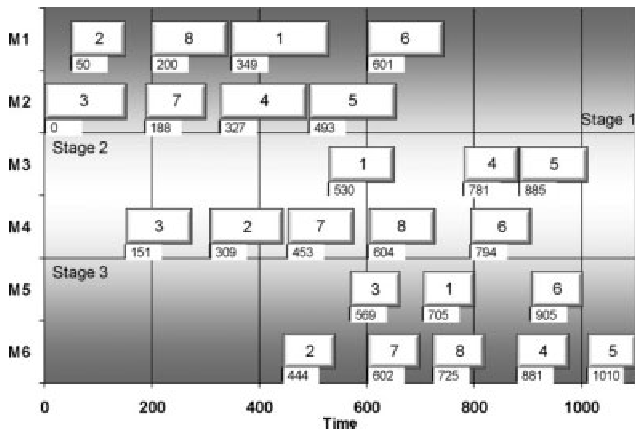

Figure 8. Optimal solution for P10C showing event point location for model CGN.

Table 2. Single Batch Problems: Overview of Computational Performance (CPU) for Makespan Minimization

Problem/Model	Optimum	CGN	CN	SF
P7M	542	**18.5**	60,000*	3,042
P8M	584	**7.02**	3,001	228
P9M	915	**232**	351	2,490
P10M	914	**25.7**	389	975
P11M	233	**11,023**	60,000†	60,000‡
P12M	265	**8,649**	4,380§	22,845
P13M	273	**41,966**	46,098¶	60,000**

*MRL, BPS = 503.8.
†MRL, SO = 237, BPS = 226.5.
‡MRL, SO = 248, BPS = 181.
§SO = 268.
¶SO = 282.
**MRL, SO = 305, BPS = 225.4.

The results for makespan minimization, given in Table 2, are somewhat different. There is now a single best performer, CGN, which is able to find the global optimal solutions for all problems and also prove optimality within the maximum resource limit. In fact, it is now significantly better than its predecessor, CT4I,[19] which could not find optimal solutions for P11M and P13M within 60,000 CPUs (CPLEX 9.1 on the same hardware). This performance can be explained by the substantial reduction observed in the integrality gap (Table 3), which is even more pronounced for earliness minimization. This is mostly due to the elimination of the output link from single product tasks and the resulting model simplifications, and not due to the use of time grids with one less event point. The values in Table 3 were calculated using Eq. 38, where RMIP stands for the optimal solution of the relaxed linear programming problem (LP). In terms of the comparison between CN and SF, one realizes that SF is faster in the less complex problems P7M-P8M, by at least one order of magnitude, and can find the optimal solution for P12M unlike CN, which is a better option for the rest.

$$\frac{\mathrm{RMIP}^{\mathrm{CGN}} - \mathrm{RMIP}^{\mathrm{CT4I}}}{\mathrm{MILP} - \mathrm{RMIP}^{\mathrm{CT4I}}} \times 100 \qquad (38)$$

It is also interesting to compare the size of the mathematical problems. Recall that CGN and CN use the same set of 4-index binary variables, so it is expected that the total number of binary variables is about the same, whenever the two employ an equal number of event points. This can be seen in Table 4 for P11M and P13M, where the slightly higher number for CN in the former, derives from a larger domain. On the other hand, SF uses 3-index binary variables, so it is no surprise that the total number of binary variables is significantly lower than for its counterparts. The drawback is that it requires a substantial larger number of constraints, mostly from the need to relate starting times of different tasks[12]/products[24] in the same unit. Note that a total of 25,136 constraints are required for the 15-product problem (P13M), a value that exceeds those for CGN and CN by a factor of 20. The new approach, CN, relates the time grids of dissimilar units directly, so it is no surprise that it is the formulation requiring the fewest constraints. Nevertheless, it does need more variables than CGN to ensure a proper material transfer between consecutive stages of production ($S_{i,m,t}$ together with the timing variables of storage units $T_{t,m}$, $m \in M^{\mathrm{ST}}$, are used instead of $\mathrm{TD}_{i,k}$).

Table 3. Reduction in Integrality Gap (%) from Formulation CT4I[19] to CGN

Problem/Objective	Cost	Makespan	Earliness
P7	–	19	100
P8	50	0	100
P9	0	57	98
P10	30	47	83
P11	0	44	87
P12	1.7	34	100
P13	–	54	78

Table 5 lists the results for total earliness minimization. It is clear that this is the easiest objective function to tackle with CGN, since all problems are solved in less than 6 min. When compared with formulation CT4I,[19] we achieve roughly a one order of magnitude decrease in computational effort, which can be explained by a major reduction in integrality gap, of more than 90% on average, see Table 3. As a result, the solution obtained for P13E and $|T| = 8$ could be confirmed for $|T| = 9$, still in a reasonable time (463 CPUs), virtually confirming that it is the global optimal solution. The performance of CN is also very good and it is indeed the fastest for P9E, P10E, and P13E. However, there is a serious limitation in the use of Eq. 6 as the objective function. This is because the new formulation cannot ensure that there are no idle slots between intervals occupied by processing tasks and so it may not be possible to make the starting times of idle slots belonging to last stage units, equal to the time horizon (see Ref. 24 for further details). As a consequence, suboptimal solutions (e.g., P8E, P11E, and P12E) cannot be improved by further increasing the number of event points. Model CN can be made completely general for earliness minimization by using similar constraints to those employed by SF,[24] but such big-M constraints severely compromise model performance as it can be seen for the latter.

Table 4. Number of Event Points and Model Entities Used by the Different Formulations for Makespan Minimization

Entity	Event Points (|T|)			Discrete Variables			Single Variables			Constraints		
Problem/Model	CGN	CN	SF	CGN	CN	SF	CGN	CN	SF	CGN	CN	SF
P7M	3	4	4	719	1148	165	934	1444	340	400	362	1442
P8M	3	4	4	589	932	165	804	1228	340	412	362	1484
P9M	4	5	6	994	1462	220	1265	1859	502	606	500	2464
P10M	4	5	6	906	1226	220	1177	1623	502	606	500	2516
P11M	5	5	5	3386	3448	318	3861	3976	668	906	600	5845
P12M	4	5	7	1453	1976	312	1830	2536	738	880	700	3673
P13M	8	8	9	6360	6360	516	6948	7061	1133	1167	794	25136

Table 5. Single Batch Problems: Overview of Computational Performance (CPU) for Total Earliness Minimization

Problem/Model	Optimum	CGN	CN	SF
P7E	88	**0.22**	1.22	449
P8E	90	**0.17**	26.8*	1,291
P9E	217	5.72	**1.72**	269
P10E	99	1.72	**0.66**	189
P11E	209	**29.6**	237[†]	60,000[‡]
P12E	150	**344**	160[§]	16,764[¶]
P13E	571	234	**186**	60,000**

*SO = 93.
[†]SO = 211.
[‡]MRL, SO = 275, BPS = 0.
[§¶]SO = 151.
**MRL, SO = 730, BPS = 0.

Since the suboptimal solutions returned by CN are within 5% of the global optimal ones, we preferred to keep it fast and less general.

Before closing this single batch section, it is important to stress out that there are other types of approaches that can be significantly more efficient than the tested unit-specific approaches. In particular, a general precedence, sequencing variable-based continuous-time model,[14] was able to solve all total cost minimization problems to global optimality except P10C, for which it ran out of memory after 3.3 h, in a mere 6 s.[19] It is however not as good as CGN for makespan and total earliness minimization. For makespan minimization, constraint programming remains the best approach, which was able to solve all problems in roughly 1 h or less.[19]

Multiple batch problems

So far, the discussion has been centered on CGN mainly for two reasons. First, because the improvements have been very significant; second, because the other two models were, with a few exceptions, simply not as good. For multiple batch problems, CGN and sequence-based models are no longer applicable and we are left with the new approach, CN, and SF[12] model.

Multistage scheduling problems featuring multiple product batches, tasks with duration dependent on the amount of material being handled and sequence-dependent changeovers, cannot be found in the literature to the best of our knowledge. Two of the three example problems considered in this section were generated using a three-stage, single product problem as a basis (example 1 of Refs. 12 and 16). Problems PMB1 and PMB2 have 2 and 3 products, respectively, and their processing data can be found in Figure 9. The product values are $v_{I_1} = 5$, $v_{I_2} = 6$, and $v_{I_3} = 4$ \$/ton, whereas the sequence-dependent changeovers are given in Table 6. Problem PMB3 involves 5 products, 4 units, and 2 stages with mostly randomly generated data (Tables 7 and 8). The

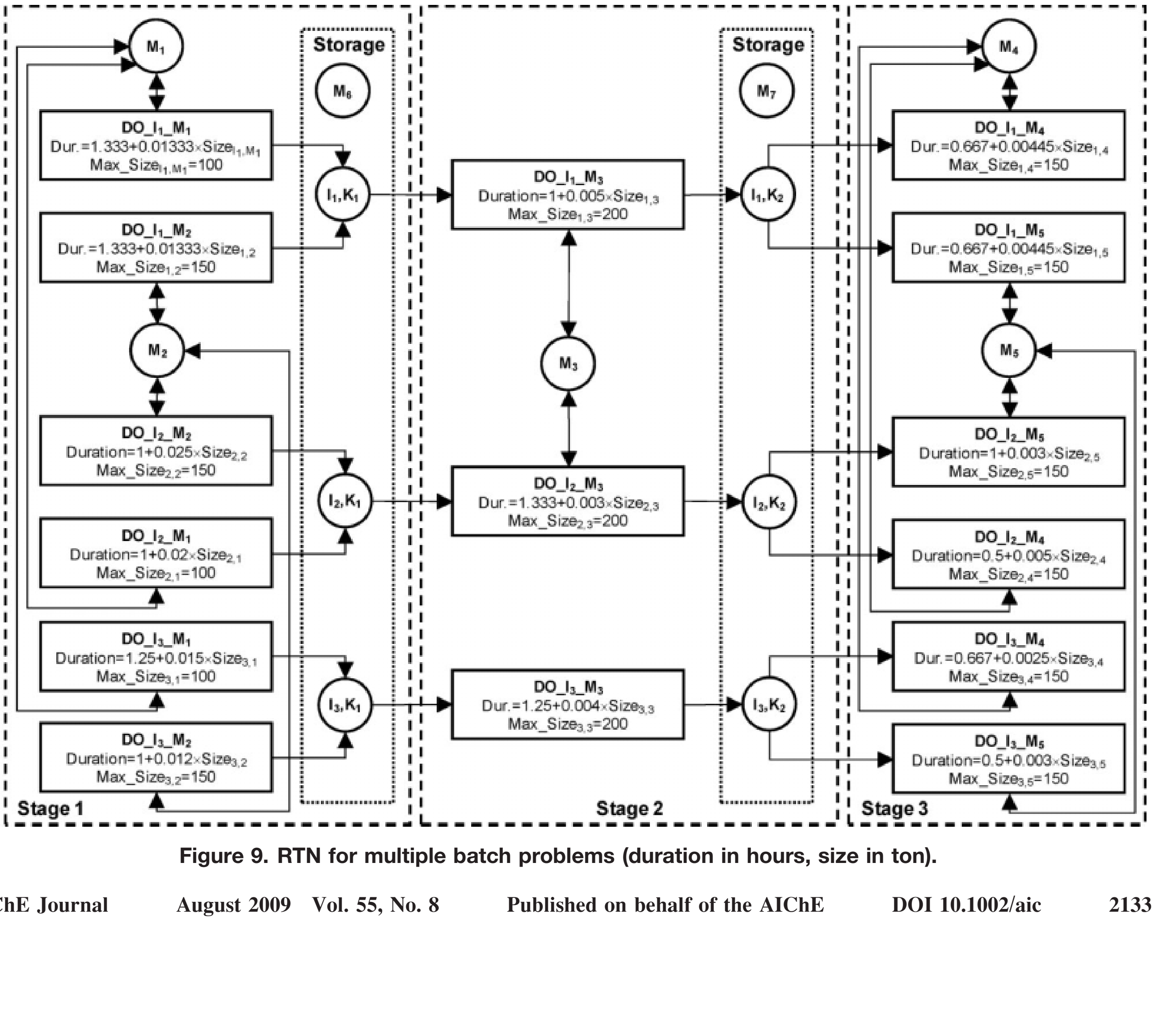
Products
I1
I2
I3
Stage 1
Stage 2
Stage 3
Storage
Storage
M1
M2
M3
M4
M5
M6
M7
I1,K1
I2,K1
I3,K1
I1,K2
I2,K2
I3,K2
DO_I1_M1 Dur.=1.333+0.01333×Size1,M1 Max_Size1,M1=100
DO_I1_M2 Dur.=1.333+0.01333×Size1,2 Max_Size1,2=150
DO_I2_M2 Duration=1+0.025×Size2,2 Max_Size2,2=150
DO_I2_M1 Duration=1+0.02×Size2,1 Max_Size2,1=100
DO_I3_M1 Duration=1.25+0.015×Size3,1 Max_Size3,1=100
DO_I3_M2 Duration=1+0.012×Size3,2 Max_Size3,2=150
DO_I1_M3 Duration=1+0.005×Size1,3 Max_Size1,3=200
DO_I2_M3 Dur.=1.333+0.003×Size2,3 Max_Size2,3=200
DO_I3_M3 Dur.=1.25+0.004×Size3,3 Max_Size3,3=200
DO_I1_M4 Dur.=0.667+0.00445×Size1,4 Max_Size1,4=150
DO_I1_M5 Dur.=0.667+0.00445×Size1,5 Max_Size1,5=150
DO_I2_M5 Duration=1+0.003×Size2,5 Max_Size2,5=150
DO_I2_M4 Duration=0.5+0.005×Size2,4 Max_Size2,4=150
DO_I3_M4 Dur.=0.667+0.0025×Size3,4 Max_Size3,4=150
DO_I3_M5 Duration=0.5+0.003×Size3,5 Max_Size3,5=150

Figure 9. RTN for multiple batch problems (duration in hours, size in ton).

Table 6. Sequence-Dependent Changeovers (h) for Problems PMB1-2

	M_1/M_2			M_3			M_4/M_5		
Unit Product	I_1	I_2	I_3	I_1	I_2	I_3	I_1	I_2	I_3
I_1	–	0.25	0.15	–	0.167	0.15	–	0.167	0.15
I_2	0.333	–	0.25	0.25	–	0.167	0.25	–	0.167
I_3	0.333	0.15	–	0.25	0.15	–	0.25	0.167	–

Table 7. Problem Data for PMB3

	M_1		M_2		M_3		M_4	
	α (h)	β (h/ton)	α (h)	β (h/ton)	α (h)	β (h/ton)	α (h)	β (h/ton)
I_1	0.8	0.016	0.8	0.019	0.653	0.015	0.653	0.013
I_2	0.832	0.017	0.832	0.02	1.468	0.039	1.468	0.029
I_3	1.31	0.026	1.31	0.034	0.74	0.017	0.74	0.015
I_4	1.426	0.029	1.426	0.038	1.269	0.033	1.269	0.025
I_5	0.674	0.013	0.674	0.015	1.134	0.029	1.134	0.023
Max_Size (ton)	100		150		200		150	

Table 8. Sequence-Dependent Changeovers (h) for PMB3

	M_1/M_4				
Unit Product	I_1	I_2	I_3	I_4	I_5
I_1	–	0.246	0.299	0.163	0.159
I_2	0.126	–	0.156	0.270	0.121
I_3	0.068	0.071	–	0.226	0.097
I_4	0.254	0.14	0.214	–	0.101
I_5	0.049	0.26	0.015	0.248	–

values are $v_{I_1} = 4.4$, $v_{I_2} = 6.9$, $v_{I_3} = 6.2$, $v_{I_4} = 8.1$, $v_{I_5} = 5.4$ \$/ton.

Two objective functions are considered, revenue maximization and makespan minimization. For the former objective, PMB1-2 are solved for two alternative scenarios in terms of minimum product demand (Δ_i), with higher values requiring a longer time horizon (H), the production of more batches and the use of a larger number of event points.

Table 9 lists the results for revenue maximization. Each problem has been solved for consecutive values of $|T|$ as part of the standard search procedure to find the global optimal

Table 9. Computational Results for Multiple Batch Problems for Revenue Maximization

Problem	Model	$\|T\|$	Discrete Variables	Single Variables	Constraints	RMIP (\$)	MIP (\$)	CPU
PMB1 ($H = 12$; $\Delta_i = 200$)	CN	3	50	154	113	3,400	3,200	0.28
	SF	5	55	136	161	3,400	3,200	0.17
	CN	4	70	205	148	4145.78	3382.32	1.69
	SF	6	70	175	228	4,600	3382.32	2.39
	CN	5	90	256	183	4501.94	3382.32	21.2
	SF	7	85	214	305	5066.53	3382.32	38.2
PMB1 ($H = 16$; $\Delta_i = 350$)	CN	5	90	256	183	5523.57	5002.63	0.84
	SF	7	85	214	305	5,650	5002.63	3.19
	CN	6	110	307	218	6031.45	5105.74	9.2
	SF	8	100	253	392	6648.77	5105.74	157
	CN	7	130	358	253	6299.07	5105.74	194
	SF	9	115	292	489	6783.86	5105.74	16,364
PMB2 ($H = 10$; $\Delta_i = 100$)	CN	3	105	250	150	3115.44	2462.87	1.77
	SF	5	70	191	271	3,300	2462.87	0.98
	CN	4	150	338	197	3553.04	2462.87	33.1
	SF	6	90	247	414	4221.61	2462.87	52.1
PMB2 ($H = 16$; $\Delta_i = 300$)	CN	6	240	514	291	5884.18	4985.29	85.2
	SF	8	130	359	790	6,300	4985.29	407
	CN	7	285	602	338	6183.66	4985.29	1,858
	SF	9	150	415	1023	6738.65	4985.29	45,318
PMB3 ($H = 24$; $\Delta_i = 100, 300, 150, 100, 100$)	CN	4	320	541	222	6,815	6,680	3.76
	SF	5	100	286	765	6,815	6702.7	6.69
	CN	5	420	691	275	8491.3	8,020	46.1
	SF	6	124	355	1154	8,840	8,076	439
	CN	6	520	841	328	9,006	8378.7	1,846
	SF	7	148	424	1623	9922.6	8346.5	31,008*
	CN	7	620	991	381	9307.9	8463.4	16,358[†]

*OM, BPS = 9511.
[†]OM, BPS = 8692.

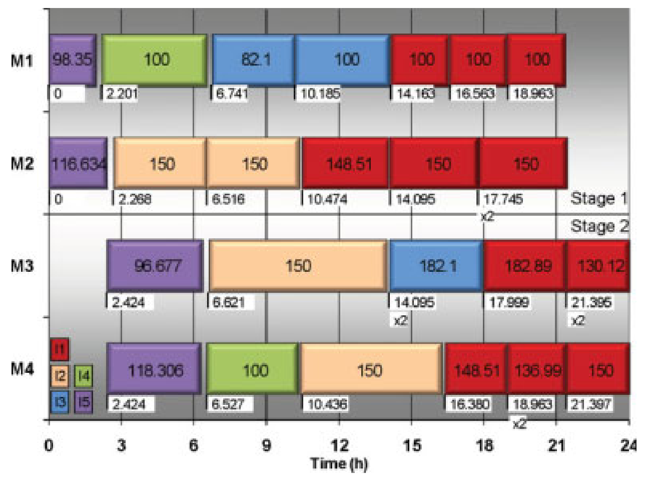

Figure 10. Best found solution for PMB3 for revenue maximization.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

solution. For problems PMB1-2, we found a relation between the number of event points required to find a particular solution by the new approach, CN, and that of SF,[12] $|T|_{SF} = |T|_{CN} + |K| - 1$; so the results are grouped according to this formula to facilitate the comparison. SF was able to find slightly better solutions for PMB3 ($|K| = 2$), for $|T|_{SF} =$ 5, 6 (than CN for $|T|_{CN} = 4, 5$). However, because of the more pronounced increase in computational effort, the results could not be confirmed for $|T|_{SF} = 7$, with the solver running out of memory. The same thing happened for $|T|_{CN} = 7$ but a significantly better solution was returned. It has \$8463.3 revenue and the corresponding schedule is given in Figure 10 together with the location and exact timing of the event points. Nevertheless, it remains to be confirmed if this is in fact the optimal solution to this problem. Overall, CN is tighter than SF, with the increase in integrality gap along the number of event points, being less pronounced. The number of binary variables is more or less the same for the two-product example (PMB1) but becomes significantly lower for SF as the number of products increases.

For makespan minimization, the differences in performance for PMB1-2 become more evident. From the results in Table 10, one can see that CN is two orders of magnitude faster in PMB1, which is not a complete surprise since for $|T| = 10$ the integrality gap is three times smaller than the one resulting from SF for $|T| = 12$. As a consequence, the new formulation could confirm that 29.859 is in fact the global optimal solution, unlike SF. The differences in integrality gap and performance between CN and SF are also important but less significant for PMB2, which is probably caused by the lower value of $|T|$ that is required. Note that PMB2 despite being a 3-product problem has a lower total demand ($900 = 300 \times 3$) than PMB1 ($2000 = 1000 \times 2$). For this problem, the worse performance of SF has a greater impact, since the optimal solution (given in Figure 11) cannot be found up to 16 h of computational time. Opposite results have been found for PMB3, for which SF proves optimality in one-third of the time required for CN using the exact same number of event points, i.e., 5.

Remarks

Time grid-based models, like the ones described in this article, have the drawback of requiring an iterative procedure to find the (global) optimal solution. The mathematical problems typically have to be solved for a few consecutive values of $|T|$ (number of event points of the grid) until no improvement is observed in the objective function or they become too difficult to solve. In this respect, sequence-based models are a better option since they have to be solved only once. However, those relying on general precedence models are not general in the presence of sequence-dependent changeovers because the optimal solution may be cutoff from the feasible space.[19,23] The same applies to models based on constraint programming.

As seen in the previous section, a single increase in $|T|$ has a large impact on computational effort. The practical upper bound on the number of event points naturally depends on the characteristics of the problem. For a given set of products, units, and stages, the simplest case is obtained for fixed assignments product/unit, where we only need to be concerned with product sequencing. Then, one has problems involving a single batch of every product, and the most complex scenario results from multiple product batches. Thus, the upper bound on $|T|$ will typically decrease

Table 10. Computational Results for Multiple Batch Problems for Makespan Minimization

Problem	Model	$\|T\|$	Discrete Variables	Single Variables	Constraints	RMIP (h)	MIP (h)	CPU
PMB1 ($\Delta_i = 1000$)	CN	10	190	511	407	29.04	30.366	3.39
	SF	12	160	409	839	26.241	30.366	203
	CN	11	210	562	447	27.336	29.859	70.6
	SF	13	175	448	976	26.223	29.859	14,730*
	CN	12	230	613	487	26.846	29.859	7,180
PMB2 ($\Delta_i = 300$)	CN	6	240	514	320	11.917	15.104	207
	SF	8	130	359	789	10.781	15.104	1,785
	CN	7	285	602	372	11.571	15.104	2,314
	SF	9	150	415	1,022	10.781	15.314	60,000†
PMB3 ($\Delta_i = 100, 300, 150, 100, 100$)	CN	3	220	391	180	15.137	17.547	0.46
	SF	4	76	217	455	11.958	17.261	0.56
	CN	4	320	541	237	13.474	16.068	88.4
	SF	5	100	286	764	11.956	15.955	978
	CN	5	420	691	294	12.910	15.955	3,031

*Solver terminated with an error, BPS = 26.76.
†MRL, BPS = 12.28.

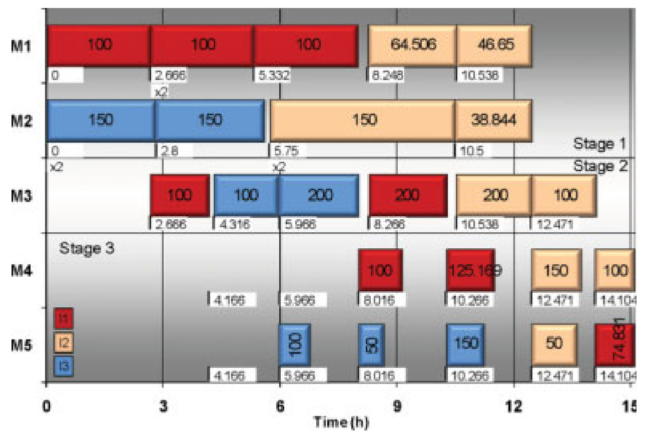

Figure 11. Optimal schedule for PMB2 for makespan minimization.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

as the complexity increases. In practice, current software and hardware capabilities will prevent us from going beyond 20 event points, sometimes not even close to this value. Naturally, this means that the capabilities of available continuous-time models are still far from those required by most medium or large-scale industrial problems. This when used as standalone approaches, for the solution of the full problem, the alternative is to use them as the basis of efficient decomposition methods, which has already been done by Janak et al.,[27] for a conceptually similar model to SF.

Finally, it is straightforward to adapt the proposed models to the use of a unit-dependent $|T|$ value. This has obvious advantages for model CGN, which requires a number of event points per unit equal to the number of assigned products, in sequencing problems. The need for the iterative procedure is thus avoided. For the other models it may not be such an easy task. In problems involving assignments, this situation may also be desirable wherever the number of processing units available in each stage is significantly unbalanced (e.g., 1 unit in Stage 1, and 5 units in Stage 2). Nevertheless, this is beyond the scope of this work because there is no systematic procedure to select the proper number of event points for each unit under any circumstance. In particular, the objective function also plays an important role, as we have seen in this work.

Conclusions

This article looked into the optimal short-term scheduling of multistage, multiproduct batch plants with sequence-dependent changeovers in cases involving multiple product batches. A new multiple time-grid continuous-time formulation has been proposed that relies on shared intermediate storage units, one virtual unit per stage, to ensure that material transfer between consecutive stages is done correctly, both in terms of time and quantity. Relevant constraints relate the absolute time of event points belonging to dissimilar time grids. This novel approach is different from the one in the well-known, unit-specific, multipurpose formulation of Floudas and coworkers,[6,10,12] where timing constraints relate the starting times of different tasks. In the new approach, 4-index binary variables are used to identify the execution of a combined processing and changeover task in a particular unit and on a certain event point. As a consequence, simple, aggregated tight constraints are obtained, whereas in the 3-index binary variable model of SF,[12] a large number of constraints need to be used, some of the big-M type. In other words, with the new model, the complexity is changed from the constraint to the binary variables level. The results from a set of 10 example problems, for alternative objective functions and different production scenarios, have shown that the new approach is significantly more efficient overall.

In problems involving a single batch per product, other approaches can be used as well. In this article, the closely related 4-index binary variable, multiple time-grid formulation of Castro et al.[19] has been reformulated to make it more efficient. One important adjustment concerns the use of one less event point per time grid. However, the major impact originates from the definition of combined processing and changeover tasks. Although combined tasks with different product indices remain unchanged, those with the same product index no longer regenerate the corresponding equipment state to prevent other tasks from being executed in that same equipment resource afterward. As it is possible to allocate tasks to time intervals in sequence, from first to last, all excess resource variables linked to equipment cleaning states can consequently be set to zero without compromising generality. The results have shown a major reduction in integrality gap, up to 100% for total earliness minimization, and a significant reduction in computational effort. The results have also shown that such approach is on average much better than the two multiple batch formulations considered.

Literature Cited

1. Kallrath J. Planning and scheduling in the process industry. *OR Spectrum.* 2002;24:219–250.
2. Floudas CA, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng.* 2004;28:2109–2129.
3. Méndez CA, Cerdá J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput Chem Eng.* 2006;30:913.
4. Kondili E, Pantelides CC, Sargent R. A general algorithm for short-term scheduling of batch operations. I. MILP formulation. *Comput Chem Eng.* 1993;17:211.
5. Pantelides CC. Unified frameworks for the optimal process planning and scheduling. In *Proceedings of the Second Conference on Foundations of Computer Aided Operations.* New York: Cache Publications, 1994:253.
6. Ierapetritou MG, Floudas CA. Effective continuous-time formulation for short-term scheduling. I. Multipurpose batch plants. *Ind Eng Chem Res.* 1998;37:4341–4359.
7. Giannelos NF, Georgiadis MC. A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. *Ind Eng Chem Res.* 2002;41:2178–2184.
8. Maravelias CT, Grossmann IE. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Ind Eng Chem Res.* 2003;42:3056–3074.
9. Castro PM, Barbosa-Póvoa AP, Matos HA, Novais AQ. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Ind Eng Chem Res.* 2004;43:105–118.
10. Janak SL, Lin X, Floudas CA. Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies. *Ind Eng Chem Res.* 2004;43:2516–2533.
11. Sundaramoorthy A, Karimi IA. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chem Eng Sci.* 2005;60:2679–2702.

12. Shaik MA, Floudas CA. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. *Comput Chem Eng.* 2008;32:260–274.
13. Méndez CA, Henning GP, Cerdá J. An MILP continuous-time approach to short-term scheduling of resource constrained multi-stage flowshop batch facilities. *Comput Chem Eng.* 2001;25:701–711.
14. Harjunkoski I, Grossmann I. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comput Chem Eng.* 2002;26:1533–1552.
15. Gupta S, Karimi IA. An improved MILP formulation for scheduling multiproduct. *Multistage Batch Plants Ind Eng Chem Res.* 2003; 42:2365–2380.
16. Shaik M, Janak S, Floudas C. Continuous-time models for short-term scheduling of multipurpose batch plants: a comparative study. *Ind Eng Chem Res.* 2006;45:6190.
17. Castro PM, Grossmann IE. New continuous-time MILP model for the short-term scheduling of multistage batch plants. *Ind Eng Chem Res.* 2005;44:9175–9190.
18. Liu Y, Karimi I. A scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. *Chem Eng Sci.* 2007;62:1549–1566.
19. Castro PM, Grossmann IE, Novais AQ. Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers. *Ind Eng Chem Res.* 2006;45: 6210–6226.
20. Prasad P, Maravelias CT. Batch selection, assignment and sequencing in multi-stage, multi-product processes. *Comput Chem Eng.* 2008;32:1114–1127.
21. Gupta S, Karimi IA. Scheduling a two-stage multiproduct process with limited product shelf life in intermediate storage. *Ind Eng Chem Res.* 2003;42:490–508.
22. Sundaramoorthy A, Maravelias CT. Simultaneous batching and scheduling in multistage multiproduct processes. *Ind Eng Chem Res.* 2008;47:1546–1555.
23. Castro PM, Erdirik-Dogan M, Grossmann IE. Simultaneous batching and scheduling of single stage batch plants with parallel units. *AIChE J.* 2008;54:183–193.
24. Castro PM, Novais AQ. Short-term scheduling of multistage batch plants with unlimited intermediate storage. *Ind Eng Chem Res.* 2008;47:6126–6139.
25. Erdirik-Dogan M, Grossmann IE. Slot-based formulation for the short-term scheduling of multistage batch plants with sequence-dependent changeovers. *Ind Eng Chem Res.* 2008;47:1159–1183.
26. Liu Y, Karimi IA. Novel continuous-time formulations for scheduling multistage batch plants with identical parallel units. *Comput Chem Eng.* 2007;31:1671–1693.
27. Janak SL, Floudas CA, Kallrath J, Vormbrock N. Production scheduling of a large-scale industrial batch plant. I. Short-term and medium-term scheduling. *Ind Eng Chem Res.* 2006;45:8234–8252.

Manuscript received Apr. 8, 2008, revision received Sept. 23, 2008, and final revision received Dec. 4, 2008.