Two New Continuous-time Models for the Scheduling of Multistage Batch Plants with Sequence Dependent Changeovers

Pedro M. Castro,[*,†] *Ignacio E. Grossmann*[‡] *and Augusto Q. Novais*[†]

[†]Departamento de Modelação e Simulação de Processos, INETI, 1649-038 Lisboa, Portugal

[‡]Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

This paper presents two new multiple-time grid, continuous-time mixed integer linear program (MILP) models for the short-term scheduling of multistage, multiproduct plants featuring equipment units with sequence dependent changeovers. Their main difference results from the explicit consideration of changeover tasks as model variables rather than implicitly through model constraints. The former approach is more versatile in terms of type of objective function that can be efficiently handled (minimization of total cost, total earliness and makespan) and, despite generating larger mathematical problems, it is also a better performer in single stage problems. The latter is better suited for multistage problems, where the former approach has some difficulties even in finding feasible solutions, particularly as the number of stages increases. The performance of both formulations is compared to other mixed integer linear program and constraint programming models. The results show that multiple time grid models are better suited for single stage problems or when minimizing total earliness, that the constraint programming model is the best approach for

[*] To whom correspondence should be addressed. Tel.: +351-210924109. Fax: +351-217167016. E-mail: pedro.castro@ineti.pt

makespan minimization and that the continuous-time model with global precedence variables is the best overall performer.

1. Introduction

Scheduling problems can be tackled by a variety of optimization approaches as well as other solution methods[1]. For instance, mathematical programming (MP) models, usually leading to mixed integer linear programming (MILP) problems, have received considerable attention in the literature. The focus has ranged essentially from specific to general types of network configurations, from pure batch to pure continuous type of processes, from short-term to periodic modes of operation and from discrete to continuous representations of time. While some are more robust than others, small changes in the characteristics of the problem can make some MP models highly inefficient, even non-applicable. Constraint programming (CP)[2], originally developed to solve feasibility problems, has also been extended to solve optimization problems, particularly scheduling problems. CP and MP approaches have complementary strengths[3] and some researchers[4-8] have already taken full advantage of this, by developing hybrid methods that are considerable more efficient than the standalone approaches.

Most of the recent MP scheduling models are based on a continuous-time representation[5,9-16]. Those employing one or more time grids[9-12,14,16] focus on general multipurpose plants and on the development of increasingly efficient models. An important recent advance was the introduction by Sundaramoorthy and Karimi[16] of a formulation without big-M constraints that proved more efficient than other competing methods. Discrete-time formulations for scheduling of multipurpose plants go further back in time starting with the seminal paper of Kondili et al.[17] that also had the merit of introducing the state-task network (STN) process representation, and which was soon followed by the resource-task network (RTN) based model of Pantelides[18]. Discrete and continuous-time approaches have complementary strengths and a mixed-time representation model has recently been presented by Maravelias[19] for the simultaneously optimization of scheduling and supply chain management problems.

While recent reviews[1,20] have appeared in the literature that discuss the relative merits of the various MP and CP approaches, they rely on performance data that often involve different problems

and distinct hardware and software tools. This paper avoids such limitations by following a more hands-on approach, where as much as six alternative models are tested on the solution of a large set of example problems concerning the short-term scheduling of single/multistage, multiproduct batch plants, featuring sequence dependent changeovers. Also analysed is the influence of the objective function on model performance.

This paper can be viewed as the third part of an extensive comparative study, where the previous two have focused on single stage[21] and multistage[22] problems featuring either sequence independent changeovers (setup times) or none at all. However, this paper goes beyond a mere comparison between different methods, since two of them are new. Both are multiple time grid continuous-time models that are extensions to the one presented by Castro and Grossmann[22] with respect to the handling of sequence dependent changeovers. They are conceptually different on how changeover tasks are handled: either explicitly, which is a more general approach in terms of variety of objective functions that can be handled efficiently, or implicitly in the model constraints, which has the advantage of generating smaller sized models. Another important novel aspect is the combination of processing and changeover tasks into a single set of tasks, which contributes to both reduction in the number of model variables and solution degeneracy, and this is done for one of the new continuous-time models, as well as for the discrete-time model. The other approaches involved in the comparison include a continuous-time model with global precedence sequencing variables[5] (SV), a CP model based on OPL Studio modelling language[23] and a hybrid MILP/CP model (single stage only). Not included is the single grid, continuous-time model of Castro et al.[14], simply because it was shown to be a poor performer in the previous studies[21-22] and due to the fact that the number of event points required to find the optimal solution would increase even more with the consideration of changeover tasks, which would inevitably lead to even larger MILPs. Other uniform time grid continuous-time formulations[11,16] are expected to have similar drawbacks and thus are also not considered.

The rest of the paper is structured as follows. Section 2 defines the scheduling problem under consideration. Section 3 gives a thorough description of the problem highlighting some of the conceptually different approaches that can be used to model it. Concerning the handling of

changeovers, two alternative, continuous-time modelling options will be identified leading to the development of two new MILP models that are presented in sections 4 and 5. The other featured approaches are described in section 6, while section 7 presents the detailed computational studies. The strengths and limitations of each approach are summarized in section 8, while the conclusions are left for section 9.

2. Problem definition

In this paper, the short-term scheduling problem of multistage, multiproduct batch plants is considered. Given are: a set of product orders $i \in I$ that must follow a sequence of processing stages, $k \in K$, to reach the condition of final products; a set of available equipment units $m \in M$, each belonging to a single stage, with set M_k including those units belonging to stage k. Given also are the duration of the processing tasks, $p_{i,m}$, and those of the cleaning tasks, $cl_{i,i',m}$; the release, r_i and due dates, d_i, all being enforced as hard constraints. Additional data consisting of the processing cost of order i in machine m, $c_{i,m}$, is required whenever the objective is the minimization of the total cost.

It is assumed that all orders go through all stages, that there is a unique sequence of stages for all orders, and that unlimited intermediate storage (UIS) is available between stages. Non-zero processing times are required to allow for an order to be processed on a given machine. Hence, the set of orders that can be processed in machine m is defined by $I_m = \{i \in I : p_{i,m} > 0\} \; \forall m \in M$. It is also assumed that any given order is only executed once over the time horizon. This makes it possible to define $cl_{i,i,m}$=0.

3. Conceptual representation: process vs. model entities

Processes involving sequence dependent changeovers are generally more difficult to model than those involving sequence independent setup times or no setup times at all. Some models, like those using global precedence sequencing variables (SV) (see section 6.1) or based on constraint programming (CP) (see section 6.3) hardly require any changes. In contrast, time-grid based approaches need to be significantly altered, both discrete- (DT) and continuous-time approaches, the latter irrespective of being single or multiple time grid models. Naturally, model adaptability to a different type of problem is directly related to the types of variables and constraints that are used,

which in turn are derived from some conceptualization. When developing a model, the modeler must make a few fundamental decisions that will have a major impact on its structure and performance. For this specific type of scheduling problem there are two main decisions. The first concerns the way in which the processing and cleaning tasks are modeled. The second concerns the treatment of time.

The processing and cleaning tasks can be handled either explicitly, as model variables, or embedded in some of the model constraints. Time grid-based approaches consider the processing tasks explicitly and binary variables are used to identify their starting time point on the grid. Concerning changeover tasks, discrete-time[17-18] and continuous-time RTN-based[24] models define them explicitly whereas continuous-time STN-based[9-12] as well as constraint programming models[2] consider them implicitly. Continuous-time models based on sequencing variables[5,13] consider all types of tasks implicitly.

The two new multiple time grid continuous-time models that are given in the next couple of sections handle changeover tasks in an opposite way. The first, and more general one (in terms of variety of objective functions that it can handle), uses a set of binary variables $\overline{N}_{i,i',m,t}$ that identify the execution of order *i* in unit *m* starting at time point *t*, together with the changeover task to allow for order *i*' to immediate follow on the same equipment. This grouping of processing and cleaning tasks into a single set of variables, instead of considering them separately, is a novel idea that improves the performance of the model due to the use of fewer variables and reduces the degeneracy since the cleaning task is performed immediately after the processing task has ended. Note also that it is possible to have *i*=*i*' in cases where *i* will be the last order to be processed on the machine under consideration (the duration of the combined task is equal to that of just the processing task since $cl_{i,i,m}$=0), although it is not mandatory that all the machines end with such tasks since many will be non-limiting machines that can have $cl_{i,i',m}\neq 0$ without compromising the optimal solution.

3.1. Resource task network process representation

The use of combined processing and changeover tasks (named *DO* in the illustrations) gives rise to the Resource Task Network[2] (RTN) process representations given in Figure 1 and Figure 2, which

are the basis of both the multiple time grid continuous-time formulation (see section 4) and the discrete-time representation (see section 6.2). It is important to emphasize that the tasks time index (t) is not important for the representation, meaning that the RTNs are valid despite the fact that index t has a slightly different meaning in each formulation, as it will be seen later on. Thus, three indices remain (two order indices and one machine index). The execution of a task involves consumption/production of several resources that, although treated in exactly the same way by the mathematical formulations, are split into three different types to facilitate model understanding: a) the equipment resources (the elements of set M); b) the material states, which are directly associated to the order i under consideration and to the stage k where the material is produced; c) the cleaning states, which are linked to the order (i) that equipment unit m is ready to handle. Due to large number of resources involved and for the sake of clarity, we have divided the overall RTN into two superstructures. While Figure 1 focuses on the first two resource types and on the material states modifications due to the processing part of the task, Figure 2 focuses on cleaning states changes caused by the changeover part of the task. The former RTN applies for a given order (i.e. I1), while the latter for a given machine (i.e. M1).

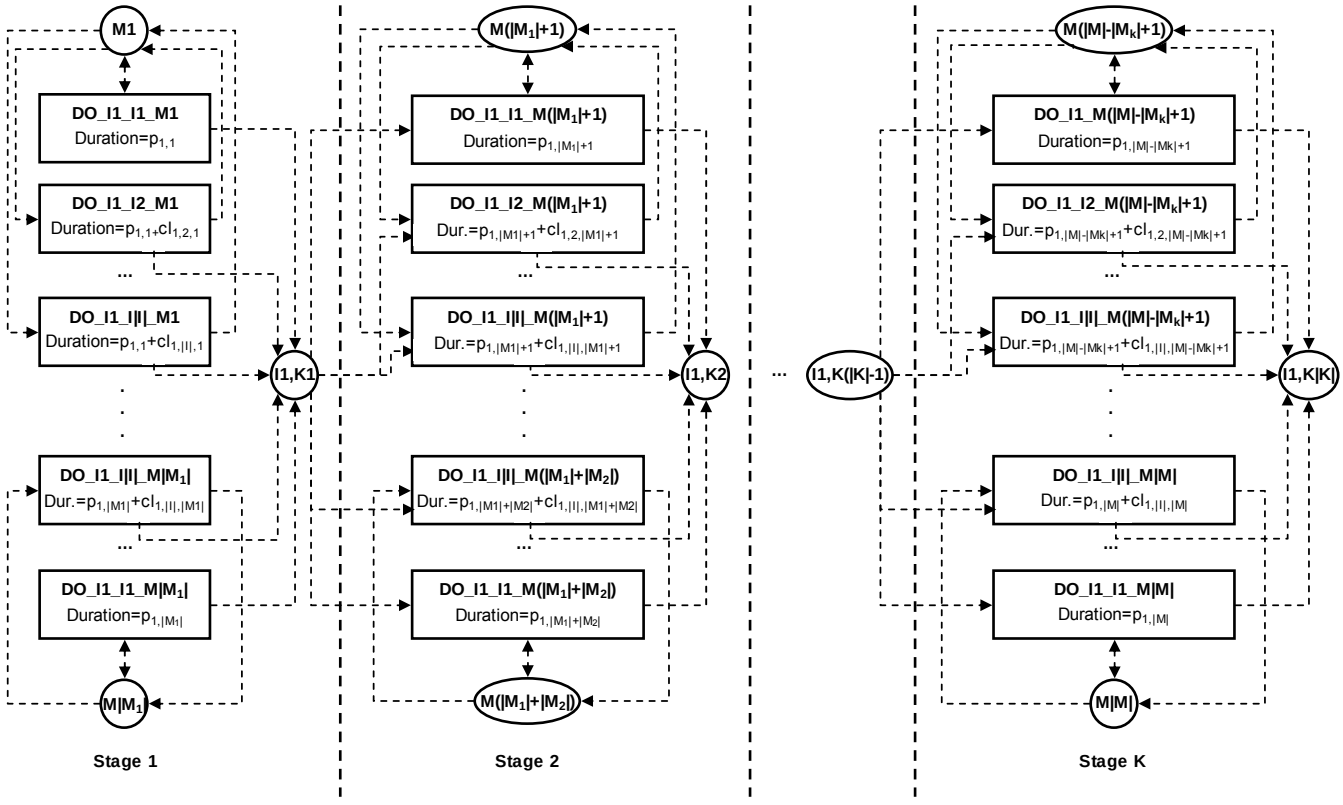

Figure 1. RTN process representation for order I1, featuring a total of |M| machines and |K| stages (changes on the units cleaning states omitted for simplification).

In Figure 1, the equipment units and material states (circles) are represented within the boundaries of the corresponding stage (vertical dash lines). For example, stage 2 (K2) includes all machines belonging to set M_2, whose elements are those of set M with numbers (if equipment numbering is based on an ordered distribution of units among stages) ranging from $|M_1|+1$ to $|M_1|+|M_2|$. A particular combined task, represented as a rectangle (e.g. DO_I1_I2_M($|M_1|+1$), consumes both the equipment unit (e.g. M($|M_1|+1$)) where it is processed and the corresponding material state, which is associated to the stage prior to the one the task belongs to (e.g. I1,K1). Both resources are consumed when the task begins. The same task also produces two resources, but now these events can occur at different points in time. While the same equipment unit is produced when the task ends (e.g. M($|M_1|+1$)), thus regenerating the equipment resource (the associated dashed lines have arrows in both ends), the material state (e.g. I1,K2) can be produced earlier (whenever $cl_{i,i',m}\neq 0$), exactly $p_{i,m}$ time units after the task has started. This is the reason why the origin of the arrow that denotes the production of the material resource is further to the left. It is also worth noting that the explicit consideration of material states as model variables is only possible when using a single time grid, like for the discrete-time formulation (see section 6.2). The multiple time grid continuous-time formulation (see section 4) uses different sets of variables and constraints.

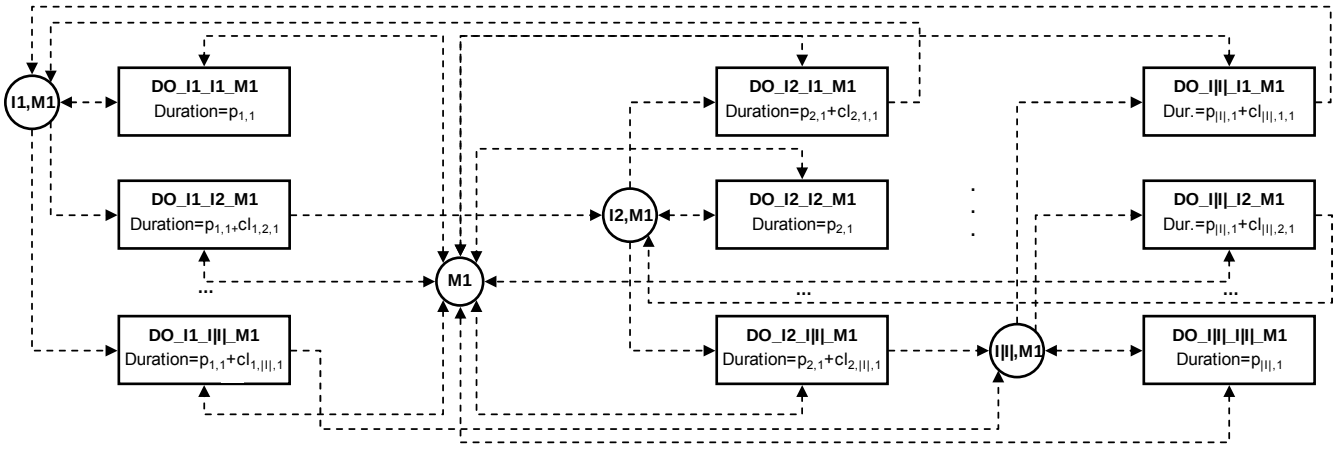

Figure 2. RTN process representation for unit M1, showing all possible cleaning states (changes on the orders material states omitted for simplification).

For the combined task to be executed in an equipment unit, it must be in an appropriate cleaning state. For unit M1, the several possibilities are illustrated in Figure 2. Tasks with the same order index (i=i') consume and produce the same cleaning state (besides the equipment resource), respectively at the start and end of the task. However, tasks involving different orders are more

frequent and involve different cleaning states. For example, task DO_I2_I1_M1 consumes resources M1 and I2,M1 at its start and produces resources M1 and I1,M1 at its end. The initial cleaning state of every equipment will be a variable of the model in order for the most convenient ones to be selected.

The second novel multiple time grid continuous-time formulation (see section 5) handles changeover times implicitly so that only the processing tasks are considered. As a consequence, cleaning states are not required and the binary variables that identify the execution of the task involve only one order index $N_{i,m,t}$. The RTN representation of the process featuring those tasks is a simplified version of the one shown in Figure 1 and is exactly the same as the one used for handling the same type of problem without sequence dependent changeovers. Such superstructure can be found in Castro and Grossmann[22].

3.2. Handling of time

The several formulations considered in this paper treat time differently. The two new multiple time grid continuous-time formulations, as well as the RTN-based discrete-time formulation, divide the time horizon into a fixed number, |T|-1, of time intervals. The number of tasks that can fit into the time horizon is greatly dependent on the number of time points in time-grid based continuous-time formulations (also sometimes called event points), and less so in discrete-time formulations. Fewer time points are also used by the former type of approach (a dozen is a practical upper bound), while the latter usually rely on tens or even hundreds of them. Another major difference is that while the discrete-time formulation features equal length (δ) intervals, meaning that the time corresponding to each time point is known *a priori*, continuous-time formulations treat those times as model variables.

The new multiple time grid continuous-time formulations use, as the name suggests, several time grids to locate the tasks. More specifically, |M| unit specific time grids are employed. It is assumed that all time grids feature the same number of event points although it is straightforward to adapt them to a different number per grid. The rationale behind this option is that the use of a unit dependent value for |T| increases the number of *a priori* decisions to make that can affect the quality of the final solution, and also because that option has been found to be an efficient one in single

stage[21] and multistage[22] multiproduct scheduling problems similar to those considered here. Nevertheless, in some problems, it may pay off to develop a set of rules leading to the specification of a different number of time points for the several time grids but this is beyond the scope of this paper.

The selection of the cardinality of set *T* involves the following trade-off: too few points makes it impossible to find the global optimum, and too many makes the problem intractable. Although only the final iteration will be reported for each example problem, a few MILPs usually need to be solved in sequence, by using single increments in |T|, until no further improvements in the objective function are observed. Due to the property of every order lasting exactly one time interval, a useful lower bound on the optimal value of |T| is the following: $\max_{k \in K}(\lceil |I|/|M_k| \rceil) + 1$. The continuous-time grid associated to each equipment unit is given in Figure 3, where the minimum release date is the lower bound on the absolute time of the first time point and the maximum due date as the upper bound on the absolute time of the last time point.

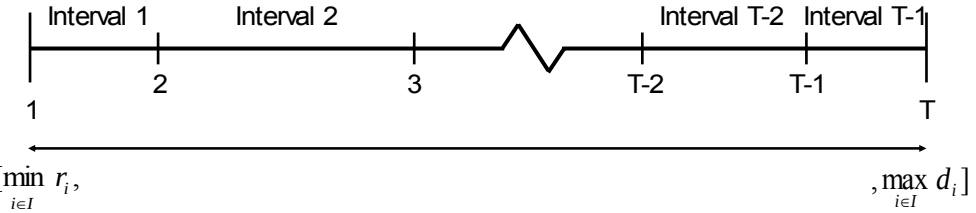

Figure 3. Continuous-time grid employed (one for each equipment unit).

In the discrete-time formulation all the events report to a single time grid and the intervals are often called global time intervals[1]. The length of each interval, δ, is often chosen as the greatest common factor between times $p_{i,m}$ and $p_{i,m}+cl_{i,i',m}$ since, as explained in section 3.1, there will be events occurring at these relative (to the start of the task) times. Whenever the greatest common factor leads to too many time intervals, meaning problem intractability, a higher value of δ is used and all the data is rounded up to its next integer multiple. The partial and total duration of the combined tasks are converted from actual time units to a time interval basis by using: $\tau_{i,m} = \lceil p_{i,m}/\delta \rceil$, $\bar{\tau}_{i,i',m} = \lceil (p_{i,m} + cl_{i,i',m})/\delta \rceil$. As for the release and due dates, they remain on the real time scale: $\bar{r}_i = \lceil r_i/\delta \rceil \cdot \delta$ and $\bar{d}_i = \lceil d_i/\delta \rceil \cdot \delta$. Note that rounding the problem data implies the consideration of an approximated version of the problem that may or may not lead to the true

optimal solution. Furthermore, to ensure feasibility, it would have been more appropriate to use $\overline{d}_i = \lfloor d_i / \delta \rfloor \cdot \delta$. The uniform discrete-time grid is given in Figure 4.

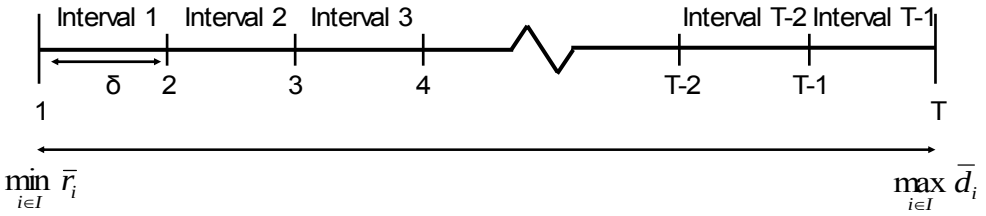

Figure 4. Uniform discrete-time grid.

The other featured continuous-time formulation does not rely on explicit time grids. Instead of allocating tasks to different time intervals, a totally different concept is exploited that relies on sequencing variables to ensure that every machine only handles one order at a time. As will be seen in section 6.1, the model variables and constraints feature no time index, giving it an important advantage when compared to the other continuous-time and discrete-time formulations: no decisions that may eventually compromise its solution need to be taken before solving the problem. That is, the model needs only to be solved once and the resulting solution will always[†] be an exact and global optimal solution (if solved to a zero optimality gap). The same can be said for the CP formulation, although this is more accurately classified as a discrete-time model since all the activities (i.e. tasks) must have integer durations.

4. New general multiple-time-grid continuous formulation, featuring four-index variables (CT4I)

While the two new multiple-time grid continuous-time formulations are novel in the sense that they can handle sequence dependent changeovers, they share many characteristics with their single stage[21] and multistage[22] predecessors. Thus, instead of entering a detailed explanation of every aspect of the formulations and drawing comparisons with other multiple time grid formulations[9-10,12,25], which are given in those recent papers, we will focus mostly on the aspects relating to the novel feature.

[†] An exception may occur as discussed in section 7.1.2.

The formulation uses the already mentioned 4-index binary variables $\overline{N}_{i,i',m,t}$ to assign the execution of the combined processing and changeover task, to a particular machine and also to a certain time point. The other set of binary[‡] variables used are the excess resource variables $R_{m,t}$ that identify equipment availability (=1) at a given event point. The remaining variables, all nonnegative, are the timing variables $T_{t,m}$ and $TD_{i,k}$, which represent the absolute time of time points t and m and the transfer time of order i in stage k, respectively, and also the new variables $C_{i,m,t}$ and $C^0_{i,m}$. While $C_{i,m,t}$ are also excess resource variables that, when equal to 1, indicate that equipment m is ready to handle order i at time point t, the latter arise from the need to define an initial state.

Figure 5 gives an overview of how the formulation works. For simplicity, it considers only one equipment unit per stage, three orders and three stages. One important property of the mathematical formulation is that each combined task only lasts for a single time interval, meaning that the required number of time points on each machine is equal to the number of orders assigned to that machine plus 1, which for this illustrative example implies 4 time points. Although each task lasts one time interval, it does not necessarily mean that when starting at t, it ends exactly at time point t+1. For instance, the execution of order I1 followed by cleaning to order I2, i.e. combined task (I1,I2), which starts at the first event point, ends well before the time of the second event point both in M1 (first stage) and M2 (second stage). Note again, as already emphasized when describing Figure 1, that the beginning of (I2,I3) in stage 2 occurs exactly after the end, in stage 1, of the processing part of the combined task (at $T_{2,2}=T_{2,1}+p_{2,1}$) but before its full completion ($T_{2,2}\leq T_{2,1}+p_{2,1}+cl_{2,3,1}$). The changeover part of the task only affects events occurring on the same machine, meaning that the appropriate cleaning time must be considered between the processing of different orders. In the first stage, the time corresponding to the start point of the task must be greater than the release date of the order, r_i. Accordingly, in the last stage, orders must be concluded before their due dates, d_i. The transfer time of materials from stage k to stage k+1 must be greater than the ending time of the order's processing part of the task at stage k and must be lower than the

[‡] They can also be defined as continuous variables since the model constraints ensure that $R_{m,t}=\{0,1\}$. However, based on experience, it is better to define them as binary variables.

order starting time at stage k+1. In Figure 5, the transfer time of order I1 in stage 1 ($TD_{1,1}$) can be any value $\in[T_{1,1}+p_{1,1},T_{1,2}]$ (this time span is represented as a gray-filled rectangle), whereas that of order 2 in stage 1 ($TD_{2,1}$) is equal to both its lower and upper bounds: $T_{2,1}+p_{2,1}=T_{2,2}$. As it was mentioned in section 3, it is not required for the last task to be processed to feature equal order indices although this will be true if such task is executed in the last time interval (in Figure 5 all equipment units end with (I3,I3)). The other features that are worth mentioning are that the time of the first and last time points, respectively in the first and last stage, do not need to match neither the origin (minimum release date) nor the time horizon (maximum due date), and that order sequencing can vary from one stage to the other, as seen for unit M2, which has I1-I2-I3, and unit M3, which has an I2-I1-I3 sequence.

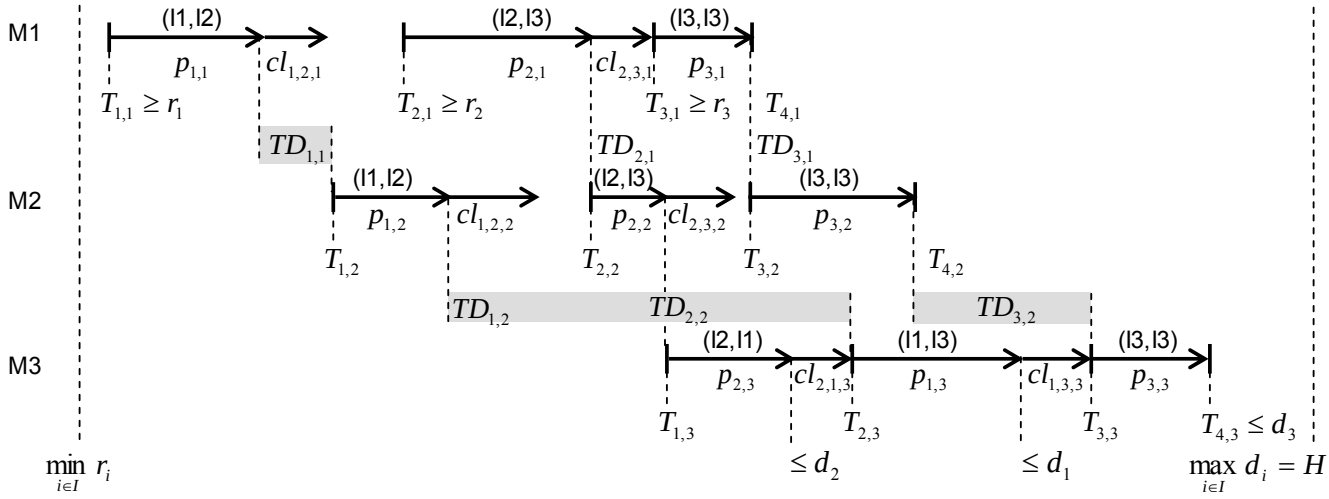

Figure 5. Possible solution from multiple-time grid, continuous-time formulation F1 (|I|=3, |M|=3, |K|=3, and |T|=4).

There are other characteristics of CT4I that are not apparent from Figure 5. More specifically, the number of time intervals can exceed the optimal number of tasks executed on a given machine or, in other words, there can be time points where no tasks are started. As a consequence, those time intervals can have a duration ranging from zero to the full (if no orders are assigned to the machine at hand) time span, and can be located anywhere, e.g. at the first, second or last time intervals. We took full advantage of this property when developing the objective function for total earliness minimization[20], which basically forces all dummy time intervals to have zero duration and be the last ones of the corresponding time grid. For other objective functions, the assignment of orders to

time points from first to last, which is illustrated in Figure 6, is enforced by an appropriate set of constraints (see eq 9).

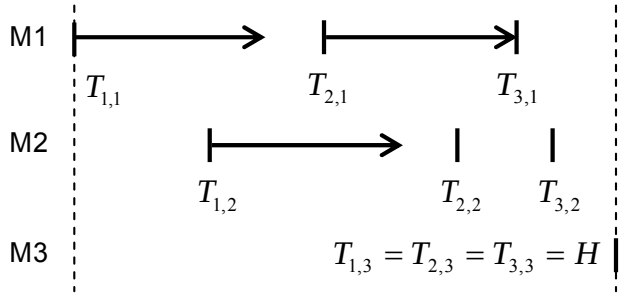

Figure 6. Possible solution from multiple-time grid, continuous-time formulation F1 (|I|=3, |M|=3, |K|=1 and |T|=3). Orders are assigned from the first to last time points.

The constraints that compose the multiple-time grid formulation are given next, but first we will define some additional sets and parameters that allow to reduce the number of variables and hence make the formulation more efficient. We start by determining the lowest time, $lb_{i,m}$, at which order i can start to be processed in unit m. If the machine belongs to the first stage, then the lower bound is the release date of the order, otherwise we have to add up to this value the minimum processing time in previous stages (eq 1).

$$lb_{i,m} = r_i + \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k'<k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} p_{i,m'} \quad \forall i \in I, m \in M_i \tag{1}$$

An upper bound (eq 2) can also be defined for order i but now this will depend on the order that is processed next.

$$ub_{i,i',m} = \min\left(d_i - \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k'>k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} p_{i,m'} - p_{i,m}, d_{i'} - \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k'>k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_{i'}}} p_{i',m'} - p_{i,m} - cl_{i,i',m} - p_{i',m}\Big|_{i \neq i'}\right)$$
$$\forall i, i' \in I, m \in M_i \cap M_{i'} \tag{2}$$

Thus, the new parameter, $ub_{i,i',m}$, features three time indices and its value can be set by either order i or i'. More specifically, if order i is the limiting one, the highest time at which the task can start is determined after subtracting from its due date, the minimum processing time on all subsequent stages plus the order processing time in unit m. If, instead, order i' is the limiting one, a similar approach is followed in terms of i'. Nevertheless, additional terms are required to include $p_{i,m}$ and also $cl_{i,i',m}$. Note also that $p_{i',m}$ is only considered whenever $i \neq i'$.

Eq 3 defines set $I_{i',m}$, which contains all orders that can precede order i' in machine m.

$$I_{i',m} = \{i \in I_m : ub_{i,i',m} \geq lb_{i,m}\} \; \forall i' \in I, m \in M_{i'} \quad (3)$$

Eq 4 defines the earliest time at which unit *m* can become active, $lbm_{m,t}$ and is determined for all time points.

$$lbm_{m,t} = \min_{i \in I_m} lb_{i,m} + \min_{i \in I_m}(p_{i,m} + cl_{i,m}^{\min}) \cdot (t-1) \; \forall m \in M, t \in T, t \neq |T| \quad (4)$$

For *t*=1, it is equal to the minimum possible starting time of all orders, while for t>1 we need to take into account the duration of the shortest combined task (note that $cl_{i,m}^{\min} = \min_{i' \neq i} cl_{i,i',m}$) as many times as the number of existing time intervals up to time point *t*. We could even be more thorough and replace the second term of eq 4 by the sum of the two smallest $p_{i,m} + cl_{i,m}^{\min}$ terms for *t*=3, the three smallest terms for *t*=4 and so on.

Finally, all this information can be combined into the definition of set $I_{i',m,t}$ (see eq 5), generating the domain of the binary variables $\overline{N}_{i,i',m,t}$. Note that in the last time interval (tasks starting at t=|T|-1) only tasks with the same order index can be executed since there are not enough time intervals to process any more tasks.

$$I_{i',m,t} = \{i \in I_{i',m} : ub_{i,i',m} \geq lbm_{m,t} \wedge (t \neq |T|-1 \vee i = i')\} \; \forall i' \in I, m \in M_{i'}, t \in T, t \neq |T| \quad (5)$$

4.1. Excess resource balance constraints

The excess resource balances are typical multiperiod material balance expressions, in which the excess amount at point *t* is equal to the excess amount at point *t*-1 adjusted by the amounts produced/consumed by all tasks starting or ending at *t*. For the equipment resources it can be said that the unit is not being used at time point *t*, i.e. is available in excess ($R_{m,t}$=1), if no order starts to be processed in it at *t*, otherwise there is no excess resource ($R_{m,t}$=0). The constraints are as follows.

$$R_{m,t} = 1 - \sum_{i' \in I_m} \sum_{i \in I_{i',m,t}} \overline{N}_{i,i',m,t} \; \forall m \in M, t \in T \quad (6)$$

The constraints related to the cleaning states are slightly more complex simply because the execution of a given task usually involves production and consumption of different states. Furthermore, the initial resource availability for all equipment units is no longer 1, and are in fact

actual model variables, which only appear in constraints belonging to the first time point (first term on the right hand side of eq 7).

$$C_{i,m,t} = C^0_{i,m}\Big|_{t=1} + C_{i,m,t-1}\Big|_{t\neq 1} + \sum_{i'\in I_{i,m,t-1}} \overline{N}_{i',i,m,t-1} - \sum_{\substack{i'\in I_m \\ i\in I_{i',m,t}}} \overline{N}_{i,i',m,t} \;\forall i\in I, m\in M_i, t\in T \tag{7}$$

Eq 8 ensures that there is but one initial equipment state for each machine.

$$\sum_{i\in I_m} C^0_{i,m} = 1 \;\forall m\in M \tag{8}$$

To reduce solution degeneracy and to improve the performance of the model, we enforce tasks to be allocated to time points with as low an index as possible. This is the same as saying that equipment availability increases from start to finish.

$$R_{m,t} \geq R_{m,t-1} \;\forall m\in M, t\in T, t\neq 1 \tag{9}$$

4.2. Timing constraints

The difference between the absolute times of any two time points must be greater than the duration of the combined task.

$$T_{t+1,m} - T_{t,m} \geq \sum_{i'\in I_m}\sum_{i\in I_{i',m,t}} [\overline{N}_{i,i',m,t}\cdot(p_{i,m} + cl_{i,i',m})] \;\forall m\in M, t\in T, t\neq |T| \tag{10}$$

Eq 11 ensures that the absolute time of time point *t* in unit *m* is greater than its predetermined lower bound (see eq 1). Note that for machines belonging to the first stage we get release date constraints. The global lower bound, as already shown in Figure 3, is the minimum release date (eq 12).

$$T_{t,m} \geq \sum_{i'\in I_m}\sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t} lb_{i,m} \;\forall m\in M, t\in T, t\neq |T| \tag{11}$$

$$T_{t,m} \geq \min_{i\in I_m} r_i \;\forall m\in M, t\in T \tag{12}$$

The next constraint is the equivalent upper bound constraint, where $ub_{i,i',m}$ is calculated through eq 2. This is a big-M constraint, meaning that is only active when there is a task starting at *t*, in unit *m*, otherwise it is relaxed to its global upper bound, eq 14.

$$T_{t,m} \leq \sum_{i'\in I_m}\sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t} ub_{i,i',m} + H\Big(1 - \sum_{i'\in I_m}\sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t}\Big) \;\forall m\in M, t\in T, t\neq |T| \tag{13}$$

$$T_{t,m} \le H = \max_{i \in I_m} d_i \ \forall m \in M, t \in T \tag{14}$$

For multistage problems, we need to relate the absolute times of consecutive stages by means of the transfer time variables $TD_{i,k}$. While eq 15a ensures that the order transfer time in stage k-1 is earlier than its starting time in stage k, eq 16a states that its transfer time in stage k must be greater than the order completion time (just the processing part of the task) in that stage (processed in machine $m \in M_k$). Both are big-M constraints that only become active if the task starts at event point t belonging to time grid m. Eqs 15a-16a can be replaced by constraints 15-16, which usually lead to a better performance. These include more binary variables inside the big-M term, which is possible since order i can only be processed once on each machine (see eq 21). Thus, eq 15 includes all tasks (from order i) that start at or before t making them tighter when solving the relaxed model, and hopefully getting partition of the tasks over fewer time intervals, which facilitates branching.

$$TD_{i,k-1} \le T_{t,m} + H(1 - \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t}) \ \forall k \in K, m \in M_k, i \in I_m, t \in T, k \ne 1, t \ne |T| \tag{15a}$$

$$TD_{i,k} \ge T_{t,m} + \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} p_{i,m} - H(1 - \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t}) \ \forall k \in K, m \in M_k, i \in I_m, t \in T, k \ne |K|, t \ne |T| \tag{16a}$$

$$TD_{i,k-1} \le T_{t,m} + H(1 - \sum_{\substack{t' \in T \\ t' \le t}} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t'}}} \overline{N}_{i,i',m,t'}) \ \forall k \in K, m \in M_k, i \in I_m, t \in T, k \ne 1, t \ne |T| \tag{15}$$

$$TD_{i,k} \ge T_{t,m} + \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} p_{i,m} - H(1 - \sum_{\substack{t' \in T \\ t' \ge t \\ t' \ne |T|}} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t'}}} \overline{N}_{i,i',m,t'}) \ \forall k \in K, m \in M_k, i \in I_m, t \in T, k \ne |K|, t \ne |T| \tag{16}$$

Whenever the objective is makespan minimization, a new variable is required (MS) that must be greater than the ending time of all tasks. Eq 17 is a constraint that ensures this goal by relating the variable to the starting time of all time points. It can be described as follows. When applied to stage k, unit m, and time point t, the second term on the right-hand side is only active when $k \ne |K|$ and represents the processing time of the order starting at time point t in unit m plus its minimum processing time in the following stages. It is equivalent to the term used for multistage plants without sequence dependent changeovers[22]. On the other hand, the third term on the RHS is only active when dealing with the last stage, and represents the duration of all combined tasks starting in unit m at or after time point t. Its origin results from performance tests performed while solving

single stage plants and it is a new term since the objective of makespan minimization was not considered in Castro and Grossmann[21].

$$MS \geq T_{t,m} + \sum_{i' \in I_m} \sum_{\substack{i \in I_{i',m,t} \\ k \neq |K|}} [\overline{N}_{i,i',m,t} \cdot (p_{i,m} + \sum_{\substack{k' \in K \\ k' > k}} \min_{m' \in M_{i}} p_{i,m'})] + \sum_{\substack{t' \in T \\ t' \geq t \\ t' \neq |T|}} \sum_{i' \in I_m} \sum_{\substack{i \in I_{i',m,t'} \\ k = |K|}} [\overline{N}_{i,i',m,t'} \cdot (p_{i,m} + cl_{i,i',m})] \; \forall k \in K, m \in M_k, t \in T, t \neq |T| \quad (17)$$

The makespan variable can also be related to the transfer time variables by a similar constraint. Although not strictly necessary, eq 18 improves the performance of the formulation.

$$MS \geq TD_{i,k} + \sum_{\substack{k' \in K \\ k' > k}} \sum_{m \in M_{k'}} \sum_{\substack{t \in T \\ t \neq |T|}} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} p_{i,m} \; \forall i \in I, k \in K, k \neq |K| \quad (18)$$

The final sets of timing constraints are also both efficient and optional. They act as lower and upper bounds on the transfer times and are conceptually equivalent to eqs 11 and 13, although now, since the constraints are per order and not per unit, the processing times are multiplied by the appropriate binary variables instead of considering the minimum possible values (which are implicit in parameters $lb_{i,m}$ and $ub_{i,i',m}$, see eqs 1-2). Note also that no big-M terms are required for the upper bound constraints.

$$TD_{i,k} \geq r_i + \sum_{\substack{t \in T \\ t \neq |T|}} \sum_{\substack{k' \in K \\ k' \leq k}} \sum_{m \in M_{k'}} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} p_{i,m} \; \forall i \in I, k \in K, k \neq |K| \quad (19)$$

$$TD_{i,k} \leq d_i - \sum_{\substack{t \in T \\ t \neq |T|}} \sum_{\substack{k' \in K \\ k' > k}} \sum_{m \in M_{k'}} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} p_{i,m} \; \forall i \in I, k \in K, k \neq |K| \quad (20)$$

4.3. Operational constraints

The single set of operational constraints states that all orders must be processed exactly once in every stage.

$$\sum_{\substack{t \in T \\ t \neq |T|}} \sum_{m \in M_k} \sum_{\substack{i' \in I_m \\ i \in I_{i',m,t}}} \overline{N}_{i,i',m,t} = 1 \; \forall i \in I, k \in K \quad (21)$$

4.4. Objective functions

The mathematical formulation can handle the three alternative objective functions considered in this paper. These are total cost minimization, eq 22, total earliness minimization, eq 23, and makespan minimization, eq 24.

$$\min \sum_{\substack{t\in T\\ t\neq|T|}} \sum_{m\in M} \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t} c_{i,m} \tag{22}$$

$$\min \sum_{i\in I} d_i - \sum_{\substack{t\in T\\ t\neq|T|}} \sum_{m\in M_{|K|}} (T_{t,m} + \sum_{i'\in I_m} \sum_{i\in I_{i',m,t}} \overline{N}_{i,i',m,t} p_{i,m}) - H[|I| - |M_{|K|}| \cdot (|T| - 1)] \tag{23}$$

$$\min MS \tag{24}$$

In summary, the formulation features constraints 6-16 and 19-21 as its building block. The objective of total cost minimization also requires eq 22, that of total earliness minimization, eq 23, while makespan minimization also uses eqs 17, 18 and 24.

5. New multiple-time-grid continuous formulation, featuring three-index variables (CT3I)

The second new continuous-time formulation uses binary variables with just three indices, $N_{i,m,t}$, and as a consequence gives rise to much smaller mathematical formulations than CT4I. The most significant conceptual difference comes from the fact that it does not need to consider explicit cleaning states for the equipment units since some of the timing constraints make sure that the appropriate cleaning time is taken into consideration. Because of this, it is closer to the multiple time grid formulation of Castro and Grossmann[22] for multistage plants without sequence dependent changeovers. However, as will be described next, the strategy used for implicitly handling changeovers is completely opposite to the one used for minimizing total earliness meaning that the previously developed form of this objective function (as given in eq 23) is incompatible with the new formulation.

Figure 7 illustrates how CT3I works with a simple single stage example since the differences from CT4I occur within the equipment units (the transfer of material between stages is similar). Before going into the details, two new sets of parameters need to be defined. First, the maximum changeover time from order *i* in unit *m* is calculated through eq 25. Then, eq 26, determines the difference between the maximum and actual changeover times from order *i* to *i*' in unit *m*. With a

view to account for the actual process and changeover time an indirect procedure is used. Assigning order *i* to unit *m* at time point *t* makes the length of interval *t* (if not the last) to equal at least $p_{i,m}+cl_{i,m}^{\max}$, which corresponds to consider the worst case scenario in terms of changeovers. To get the true changeover time to order *i*' we must subtract $cl_{i,i',m}^{\Delta}$, if task *i*' is to be performed at *t*+1, as can be seen between orders I1 and I4 in M1 and also I5 and I2 in M2. It is worth noting that the difference between the absolute times of two consecutive time points may even be greater if, for example, the release date of the following product is located further ahead in time (see I2 in Figure 7). This technique makes it more advantageous for two consecutive orders to be executed in consecutive time intervals. Furthermore, all possible feasible solutions are covered since the maximum changeover term is not considered for tasks starting at the last time interval since in such case only the processing time is used. This can be seen for orders I4, I2 and I3. As a consequence, orders will typically be assigned from the last but one, to the first time points, in an opposite manner to F1.

$$cl_{i,m}^{\max} = \max_{i' \in I_m} cl_{i,i',m} \;\forall m \in M, i \in I_m \tag{25}$$

$$cl_{i,i',m}^{\Delta} = cl_{i,m}^{\max} - cl_{i,i',m} \;\forall m \in M, i, i' \in I_m \tag{26}$$

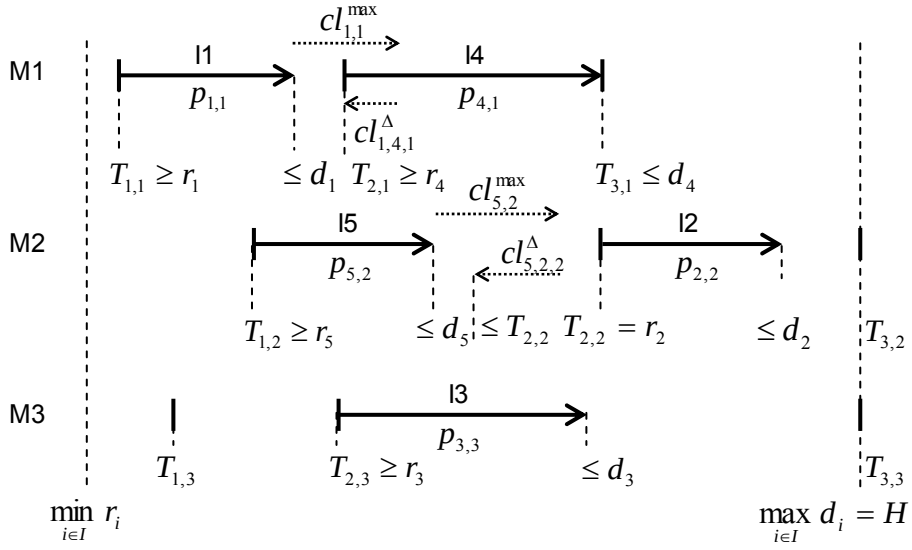

Figure 7. Possible solution from multiple-time grid, continuous-time formulation F2 (|I|=5, |M|=3, |K|=1 and |T|=3). Orders are assigned from the last but one to the first time points.

The constraints of the three-index binary variables, continuous time formulation are given next. When compared to Castro and Grossmann,[22] only those that were modified, to account for sequence dependent changeovers, are explained, together with new sets of constraints.

$$R_{m,t} = 1 - \sum_{\substack{i \in I_m \\ t \neq |T|}} N_{i,m,t} \quad \forall m \in M, t \in T \tag{27}$$

$$R_{m,t} \leq R_{m,t-1} \quad \forall m \in M, t \in T, t \neq 1, t \neq |T| \tag{28}$$

Equation 28 is analogous to eq 9 and leads to improved performance. The idea is once again to reduce the number of degenerate solutions, now by forcing all orders to be assigned from the last to the first time interval. As a consequence, unit availability will decrease from start to finish with the exception of the last time point, which corresponds to the end of the time horizon, where all equipment units become available.

$$T_{t+1,m} - T_{t,m} \geq \sum_{i' \in I_m} N_{i',m,t} p_{i',m} + (N_{i,m,t} cl_{i,m}^{\max})\Big|_{t \neq |T|-1} - \sum_{\substack{i' \in I_m \\ t \neq |T|-1}} N_{i',m,t+1} cl_{i,i',m}^{\Delta} \quad \forall m \in M, i \in I_m, t \in T, t \neq |T| \tag{29}$$

Equation 29 is the root of CT3I and has already been described while explaining Figure 7. The handling of sequence dependent changeovers makes order aggregation no longer possible so the domain of eq 29 now features three indices: *i*, *m* and *t*. The first term on the RHS can be replaced by $N_{i,m,t} \cdot p_{i,m}$, which has the advantage of making the constraint easier to understand but has the disadvantage of making it less tight and hence less efficient.

$$T_{t,m} \geq \sum_{i \in I_m} N_{i,m,t} lb_{i,m} \quad \forall m \in M, t \in T, t \neq |T| \tag{30}$$

$$hb_{i,m} = d_i - p_{i,m} - \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k' > k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} p_{i,m'} \quad \forall i \in I, m \in M_i \tag{31}$$

$$T_{t,m} \leq \sum_{i \in I_m} N_{i,m,t} hb_{i,m} + H(1 - \sum_{i \in I_m} N_{i,m,t}) \quad \forall m \in M, t \in T, t \neq |T| \tag{32}$$

Parameter $hb_{i,m}$ represents the highest time at which order *i* can start to be processed on unit *m* and is used in eq 32 to define upper bounds for the absolute time of the several time points. Note that eq 32 translates into the due date constraint, whenever $m \in M_{|K|}$.

$$TD_{i,k-1} \leq T_{t,m} + H(1 - \sum_{\substack{t' \in T \\ t' \leq t}} N_{i,m,t'}) \quad \forall k \in K, m \in M_k, i \in I_m, t \in T, k \neq 1, t \neq |T| \tag{33}$$

$$TD_{i,k} \geq T_{t,m} + N_{i,m,t}\, p_{i,m} - H(1 - \sum_{\substack{t' \in T \\ t' \geq t \\ t' \neq |T|}} N_{i,m,t'}) \; \forall k \in K, m \in M_k, i \in I_m, t \in T, k \neq |K|, t \neq |T| \tag{34}$$

$$cl_{i,m}^{\min} = \min_{\substack{i' \in I_m \\ i' \neq i}} cl_{i,i',m} \; \forall m \in M, i \in I_m \tag{35}$$

$$\begin{aligned} MS \geq\; & T_{t,m} + \sum_{\substack{i \in I_m \\ k \neq |K}} [N_{i,m,t} \cdot (p_{i,m} + \sum_{\substack{k' \in K \\ k' > k}} \min_{m' \in M_{i}} p_{i,m'})] + \\ & \sum_{\substack{t' \in T \\ t' \geq t \\ t' \neq |T|}} \sum_{\substack{i \in I_m \\ k = |K|}} [N_{i,m,t'} \cdot (p_{i,m} + cl_{i,m}^{\min}\big|_{t \neq |T|-1})] \; \forall k \in K, m \in M_k, t \in T, t \neq |T| \end{aligned} \tag{36}$$

$$MS \geq TD_{i,k} + \sum_{\substack{k' \in K \\ k' > k}} \sum_{m \in M_{i}} \sum_{\substack{t \in T \\ t \neq |T|}} N_{i,m,t}\, p_{i,m} \; \forall i \in I, k \in K, k \neq |K| \tag{37}$$

$$TD_{i,k} \geq r_i + \sum_{\substack{t \in T \\ t \neq |T|}} \sum_{k' \in K} \sum_{\substack{m \in M_{k'} \\ k' \leq k \\ m \in M_i}} N_{i,m,t}\, p_{i,m} \; \forall i \in I, k \in K, k \neq |K| \tag{38}$$

$$TD_{i,k} \leq d_i - \sum_{\substack{t \in T \\ t \neq |T|}} \sum_{k' \in K} \sum_{\substack{m \in M_{k'} \\ k' > k \\ m \in M_i}} N_{i,m,t}\, p_{i,m} \; \forall i \in I, k \in K, k \neq |K| \tag{39}$$

$$\sum_{\substack{t \in T \\ t \neq |T|}} \sum_{\substack{m \in M_i \\ m \in M_k}} N_{i,m,t} = 1 \; \forall i \in I, k \in K \tag{40}$$

Equation 36 is equivalent to eq 17, but while we can use the actual changeover time in the last term on the RHS of the latter, for the former, since changeovers are modelled implicitly, we are limited to use the minimum cleaning time, as calculated by eq 35.

$$\min \sum_{\substack{t \in T \\ t \neq |T|}} \sum_{m \in M} \sum_{i \in I_m} N_{i,m,t} c_{i,m} \tag{41}$$

The mathematical formulation can handle as the objective function, the minimization of either the total cost or the makespan, eqs 41 and 24, respectively. Its core consists of constraints 12, 14, 27-30, 32-34 and 38-40, plus eq 41 for total cost, and eqs 24, 36 and 37, for makespan.

6. Other approaches

The two new continuous-time formulations will be compared to other four conceptually different approaches. These include a continuous-time formulation based on global precedence variables (SV) instead of relying on an explicit time grid; a uniform time grid, discrete-time formulation (DT), a constraint programming (CP) model and, for total cost minimization of single stage plants, a hybrid

MILP/CP approach. In this section, we highlight their main features and present the changes required to efficiently tackle the problems under consideration.

6.1. Continuous-time formulation with global precedence sequencing variables (SV)

The other featured continuous-time formulation also gives rise to an MILP and is essentially the one of Harjunkoski and Grossmann[5], without the operational design variables. Binary variables $y_{i,m}$ are employed to assign order i to unit m and binary sequencing variables $x_{i,i',k}$ are used to identify the global precedence of order i over i' in stage k. Based on these, the ending times $T^f_{i,k}$ of any two orders can be related through the big-M constraints given in eqs 42-43. These now feature an extra term (the third on the RHS) to account for the sequence dependent changeover times, which the model by Harjunkoski and Grossmann[5] did not consider. Other improvements concerning the efficient handling of other objective functions that are relevant to this work can be found in Castro and Grossmann[22].

$$T^f_{i',k} \geq T^f_{i,k} + p_{i',m} + cl_{i,i',m} \cdot x_{i,i',k} - H(3 - x_{i,i',k} - y_{i,m} - y_{i',m}) \forall i,i' \in I, k \in K, m \in M_i \cap M_{i'} \cap M_k, i' > i \quad (42)$$

$$T^f_{i,k} \geq T^f_{i',k} + p_{i,m} + cl_{i',i,m} \cdot (1 - x_{i,i',k}) - H(2 - x_{i,i',k} - y_{i,m} - y_{i',m}) \forall i,i' \in I, k \in K, m \in M_i \cap M_{i'} \cap M_k, i' > i \quad (43)$$

6.2. RTN-based discrete-time formulation (DT)

The discrete-time formulation used is based on the original work of Pantelides[18] despite the fact that the different types of resources are not aggregated. It relies on the Resource Task Network process representation like CT4I and CT3I and thus it has a few similar features with its continuous-time counterparts. More so with CT4I since it also considers combined tasks and the same type of binary variables $\overline{N}_{i,i',m,t}$. The discrete-time grid, however, relies on a single and uniform time grid, making it easier to consider alternative material states (through variables $S_{i,k,t}$) and also avoiding the use of timing variables.

Due to the fact that the time corresponding to each time point is known *a priori*, we can determine the lowest and highest points at which each task can start. Despite referring to time points instead of actual time values, eqs 44 and 45 are similar in concept to eqs 1 and 2. In eqs 44 note that $\min_{i' \in I} \overline{r}_{i'} / \delta$

represents the time of the first time point in terms of number of time intervals (see also Figure 4). Since the third term on the RHS calculates the minimum possible duration in previous stages (also in number of time intervals), the first and third term in eq 44 in fact determine the number of time intervals between the lowest possible starting point of the task and the first time point. If there are no intervals in between, then the task can start at the first time point, and this is the reason why we add a 1 (second term in the RHS). The domain of the model binary variables is then calculated through eq 46.

$$\overline{lb}_{i,m} = (\overline{r}_i - \min_{i' \in I} \overline{r}_{i'}) / \delta + 1 + \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k' < k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} \tau_{i,m'} \quad \forall i \in I, m \in M_i \tag{44}$$

$$\overline{ub}_{i,i',m} = \min(\overline{d}_i / \delta - \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k' > k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_i}} \tau_{i,m'} - \tau_{i,m}, \overline{d}_{i'} / \delta - \sum_{k \in K_m} \sum_{\substack{k' \in K \\ k' > k}} \min_{\substack{m' \in M_{k'} \\ m' \in M_{i'}}} \tau_{i',m'} - \overline{\tau}_{i,i',m} - \tau_{i',m}\big|_{i \neq i'}) \\ - \min_{i' \in I} \overline{r}_{i'} / \delta + 1 \; \forall i, i' \in I, m \in M_i \cap M_{i'} \tag{45}$$

$$\overline{I}_{i',m,t} = \{i \in I_m : \overline{lb}_{i,m} \leq t \leq \overline{ub}_{i,i',m}\} \; \forall m \in M, i' \in I_m, t \in T \tag{46}$$

The discrete-time formulation can be written in a very compact form. It uses 5 sets of variables and constraints (eq 47-50 and eq 8) plus the objective function: eq 51 for total cost minimization and eq 52 for total earliness minimization. The model constraints are given below and with the exception of the excess resource balances for the cleaning states (variables $C_{i,m,t}$), which are new, are similar to those of Castro and Grossmann[22]. The reader is directed to this reference for a detailed explanation of the model constraints and also for the technique used for makespan minimization, which involves solving the problem several times, for different cardinalities of T, while minimizing total earliness.

$$R_{m,t} = (1\big|_{t=1} + R_{m,t-1}\big|_{t \neq 1}) - \sum_{i' \in I_m} \sum_{i \in \overline{I}_{i',m,t}} \overline{N}_{i,i',m,t} + \sum_{i' \in I_m} \sum_{i \in \overline{I}_{i',m,t-\overline{\tau}_{i,i',m}}} \overline{N}_{i,i',m,t-\overline{\tau}_{i,i',m}} \quad \forall m \in M, t \in T \tag{47}$$

$$S_{i,k,t} = S_{i,k,t-1} - \sum_{\substack{m \in M_{k+1} \\ m \in M_i}} \sum_{\substack{i' \in I \\ i \in \overline{I}_{i',m,t}}} \overline{N}_{i,i',m,t} + \sum_{\substack{m \in M_k \\ m \in M_i}} \sum_{\substack{i' \in I \\ i \in \overline{I}_{i',m,t-\tau_{i,m}}}} \overline{N}_{i,i',m,t-\tau_{i,m}} \quad \forall i \in I, k \in K, t \in T \tag{48}$$

$$C_{i,m,t} = (C^0_{i,m}\big|_{t=1} + C_{i,m,t-1}\big|_{t \neq 1}) - \sum_{m \in M_i} \sum_{\substack{i' \in I_m \\ i \in \overline{I}_{i',m,t}}} \overline{N}_{i,i',m,t} + \sum_{m \in M_i} \sum_{i' \in \overline{I}_{i,m,t-\overline{\tau}_{i',j,m}}} \overline{N}_{i',i,m,t-\overline{\tau}_{i',j,m}} \quad \forall i \in I, m \in M_i, t \in T \tag{49}$$

$$\sum_{\substack{m \in M_i \\ m \in M_k}} \sum_{t \in T} \sum_{\substack{i' \in I_m \\ i \in \overline{I}_{i',m,t}}} \overline{N}_{i,i',m,t} = 1 \; \forall i \in I, k \in K \tag{50}$$

$$\min \sum_{t \in T} \sum_{m \in M} \sum_{i' \in I_m} \sum_{i \in \overline{I}_{i',m,t}} \overline{N}_{i,i',m,t} c_{i,m} \tag{51}$$

$$\min Z = \sum_{i \in I} \bar{d}_i - |I| \cdot \min_{i' \in I} \bar{r}_{i'} - \delta \cdot \sum_{t \in T} \sum_{m \in M_{|K|}} \sum_{i' \in I_m} \sum_{i \in \bar{I}_{i',m,t}} \left[\bar{N}_{i,i',m,t} (t + \tau_{i,m} - 1) \right] \quad (52)$$

6.3. Constraint programming formulation (CP)

The constraint programming (CP) formulation used is basically the one presented by Harjunkoski and Grossmann[5], which is based in ILOG's OPL Studio modelling language[23]. However, the issue of sequence and machine dependent changeovers is difficult to implement in OPL Studio code and could only be handled after advice from ILOG support staff. For this reason we find it relevant to include the full model. The reader is also directed to Maravelias and Grossmann[4] for a brief description of OPL Studio global constraints and special constructs specifically developed for scheduling applications.

The two main components of scheduling models in OPL Studio are activities and resources. The activities correspond to the processing tasks and are referred to a given order *i* (belonging to the enumerated type Orders) and to a given stage *k* (belonging to Stages, ranging from 1 to Nstages). Since an appropriate changeover time must pass between consecutive activities, we define a transition type that will access the transition matrix given the appropriate element, in this case *i*, see eq 53. The transition matrix is given by parameter *chgover*[Units,Orders,Orders] (equivalent to $cl_{i,i',m}$) and is associated to the appropriate equipment unit *m* belonging to the range type Units. The units are defined as unary resources since they can only be used by one activity at a time (eq 54) and are also the elements of the group of Machines, the alternative resources (eq 55). Note that in eq 54, the transition matrix is referenced with only one (*m*) of its three indices. Also, declaring that the several units are alternative from an activity standpoint, is absolutely vital to ensure an efficient CP model. Further relevant declarations involve the binary assignment variable *y*[*i*,*m*] and the boundaries of the time horizon, which are related to the minimum release and the maximum due dates (eqs 57-58).

Activity DO[*i* in Orders, *k* in Stages] transitionType *i*; (53)

UnaryResource unit[*m* in Units] (*chgover*[*m*]); (54)

AlternativeResources Machines(unit); (55)

var int *y*[Orders,Units] in 0..1; (56)

scheduleOrigin=min(i in Orders) $r[i]$; (57)

scheduleHorizon = max(i in Orders) $d[i]$; (58)

The model constraints are given next. Eq 59 states that order i can only start to be processed on the first stage after its release date. The execution of order i in the last stage must also end before its due date, eq 60. Each activity needs to be performed in one equipment belonging to the group of alternative resources, eq 61. The duration of activities belonging to stage k is then, in effect, bounded by the minimum and maximum processing times of the order in that stage (eq 62). Eq 63, states that if unit m is selected to process order i in stage k, then the duration of the activity must equal the matching processing time. Also, the corresponding assignment variable must equal 1. If unit m does not belong to stage k, or cannot process order i, then it cannot be selected to perform the activity, eq 64. Finally, eq 65 states that order i can only be processed in stage k after going through the previous stage.

DO[i,1].start≥r[i] $\forall i \in I$ (59)

DO[i,Nstages].end≤d[i] $\forall i \in I$ (60)

DO[i,k] requires Machines $\forall i \in I$, $\forall k \in K$ (61)

$$\min_{\substack{m \in M_k \\ m \in M_i}} p[i,m] \le \text{DO}[i,k].\text{duration} \le \max_{\substack{m \in M_k \\ m \in M_i}} p[i,m] \; \forall i \in I, k \in K \quad (62)$$

$$\begin{aligned} &\text{activityHasSelectedResource}(\text{DO}[i,k], \text{Machines}, \text{unit}[m]) \Rightarrow \\ &\text{DO}[i,k].\text{duration} = p[i,m] \;\&\; y[i,m] = 1 \; \forall i \in I, \forall k \in K, \forall m \in M_i \cap M_k \end{aligned} \quad (63)$$

$$\text{not activityHasSelectedResource}(\text{DO}[i,k], \text{Machines}, \text{unit}[m]) \; \forall i \in I, \forall k \in K, \forall m \notin M_i \vee m \notin M_k \quad (64)$$

DO[i,k] precedes DO[i,k+1] $\forall i \in I$, $\forall k \in K, k \neq |K|$ (65)

Three alternative objective functions, eqs 66-68, for total cost, total earliness and makespan minimization, are respectively given by:

$$\min \sum_{i \in I} \sum_{m \in M_i} y[i,m] \cdot c[i,m] \quad (66)$$

$$\min \sum_{i \in I} (d[i] - DO[i, \text{Nstages}].\text{end}) \quad (67)$$

$$\min Z = \max_{i \in I} DO[i, \text{Nstages}].\text{end} \quad (68)$$

6.4. Hybrid formulation (MILP/CP)

The hybrid model of Jain and Grossmann[6] together with the knapsack constraints of Maravelias and Grossmann[7] to improve the integer cuts, is also an efficient option for single stage problems where the objective is total cost minimization. It uses a simplified version of model SV, one where only the assignment variables are considered, to determine optimal assignments of orders to machines. Since no sequencing variables are used, some assignments may be infeasible, something that is checked through the solution of a CP feasibility problem for each equipment unit. For each infeasible unit, integer cuts are added to avoid getting the same assignments on the next solution of the MILP. Several iterations are usually required until all machines are proved feasible meaning that the optimal solution has been found. Although the same decomposition strategy can be used with other objective functions for single stage problems, the method is likely to worsen as the CP is required to solve an optimization rather than a feasibility problem. For multistage problems and for total cost minimization, Harjunkoski and Grossmann[5] also tried a hybrid MILP/CP method and found that, unlike in single stage problems, valid cuts are rather weak and that a large number of iterations can be expected before the optimal solution is found. Although the authors devised stronger heuristic cuts, they sometimes cut off the true optimal solution. In view of the above, the use of the hybrid MILP/CP approach was not considered in cases other than total cost minimization for single stage problems.

7. Computational Results

In this section, the performance of the six different approaches is illustrated through the solution of 39 example problems. These are identified by a number and two additional characters, where the last identifies the objective function being considered, e.g., C for total cost, E for total earliness, and M for makespan minimization. Most of the data has been taken from the example problems given in Castro and Grossmann[21-22], although the changeover times where generated randomly (up to a maximum of 60% of the units average processing time). Since these changeover times take up a lot of space in tables, we have opted not include the data in the paper and give it instead as supporting information (most challenging examples only: P5-P6, P11-P13).

For solving the MILPs resulting from the continuous and discrete-time models (CT4I, CT3I, SV, DT) we have used commercial solver GAMS/CPLEX 9.1, with a relative tolerance of 1E-6, and all problems were solved to optimality, unless otherwise stated. The constraint programming (CP) and hybrid (MILP/CP) models where implemented and solved in ILOG's OPL Studio 3.7.1. Concerning hardware, a computer consisting of a Pentium-4 3.4 GHz processor with 1 GB of RAM and running Windows XP Professional was used.

The results have been grouped by type of problem under consideration, single or multistage, and then by objective function. An overview of the computational effort is given in Tables 1,3-5,7-8, while more detailed computational statistics for some of the problems are left for Tables 2 and 6. The discussion of the results is given in sections 7.1 and 7.2, by type of problem.

7.1. Single Stage Problems

The single stage problems under consideration range from 12 orders in 3 equipment units to 20 orders in 5 units. Although a greater number of problems could be considered, the large amount of computational resources used suggests this set corresponds to a representative set.

7.1.1. Total cost minimization

As can be seen from Table 1, the two new multiple time grid continuous formulations are the best performers for total cost minimization by a significant margin in relation to all but the hybrid MILP/CP model. We were surprised by the fact that CT4I was more efficient than CT3I, particularly in P5C. In Table 2, one can see that CT4I exceeds the number of binary variables employed in CT3I by a factor of 10, has a slightly lower but similar integrality gap and is solved faster by almost two orders of magnitude. Model SV required even fewer binary variables and although it can always find the global optimal solution, it failed to prove optimality in three cases (P3C, P5C and P6C), either because the maximum resource limit was achieved, or because the solver ran out of memory. The CP model exhibited a better performance than SV but failed to find the optimal solution for P5C. At the bottom of the list comes the discrete-time formulation (DT), which, due to the large number of time points that are required to handle the exact problem data, could only solve approximate versions of the problems. For instance, P2C needs 191 time points for δ=2, which results in a good

approximation of the problem data. For δ=5 the combined processing times are somewhat overestimated but despite this, the optimal solution can still be found in some cases (P3C, P5C).

7.1.2. Total earliness minimization

For total earliness minimization, CT4I continues to be the best performer even though it fails to find the optimal solution for P6E, see Table 3. For this problem we have to rely on DT, which is also a very good performer. In particular, all example problems except P5E (δ=5) could be solved with the exact problem data, which led to a maximum of 471 time points in P1E and a very large problem size with 381 time points and a total of 343143 binary variables, 383279 single variables and 40031 constraints, for P6E. The other two formulations that can handle this objective function, have a significant decrease in performance when compared to total cost minimization. SV generates search trees that explode in size rather rapidly and hence lead to the solver running out of memory (P3E-P6E). CP performs slightly better since, while also failing to find the optimal solutions for P3E, P5E and P6E, finds better solutions for the former examples. Furthermore, it solves P1E-P2E significantly faster. Problem P3E is the most interesting problem of the lot, since the CP formulation terminated with an optimal solution (561) that is in fact suboptimal. This fact allowed us to identify the most significant limitation of the CP formulation, which is also a limitation of model SV.

Before going into the detailed explanation let us provide some relevant problem data. The optimal solution of P3E features an optimal sequence I3-I14-I11-I10 in M1. The processing times are given by $p_{3,1}$=113, $p_{14,1}$=23, $p_{11,1}$=83, $p_{10,1}$=73, the corresponding changeover times by $cl_{3,14,1}$=2, $cl_{14,11,1}$=8, $cl_{11,10,1}$=2, the release dates by r_3=40, r_{10}=10, r_{11}=50, r_{14}=60 and the due dates by d_3=310, d_{14}=200, d_{11}=300, d_{10}=370. Also required is the data element $cl_{3,11,1}$=39, and the orders optimal delivery dates: 175, 200, 295 and 370, respectively. The optimal schedule for unit M1 is given in Figure 8 together with the best solution that can be obtained by models CP and SV. The difference between the two schedules is minimal, in the optimal solution (shown above) the starting time of order I3 is delayed up to an absolute time of 62, allowing it to end 2 time units later, where 2 is the exact difference between the optimal total earliness values (559 vs. 561). So why cannot order I3 start earlier in the solution of models F3 and F5? The reason lies in other model constraints that relate it

to its global successors and prevent this from happening. Based on SV (for CP the explanation is the same although such constraint is implicit), eq 42, when applied to the orders and equipment unit in question, explicitly states that $T^f_{11,1} - T^f_{3,1} \geq 122$, meaning that the difference between the ending times of orders I11 and I3 must be greater than 122 (the processing time of order I11 plus the changeover time from I3 to I11). Thus, we cannot take full advantage of the fact that there is one order, I14, that can fit between I3 and I11. Although it is unlikely that such combination of processing data occurs in a real industrial environment, this example clearly highlights one of the strengths of time grid-based models when compared to approaches based on explicit or implicit sequencing of tasks.

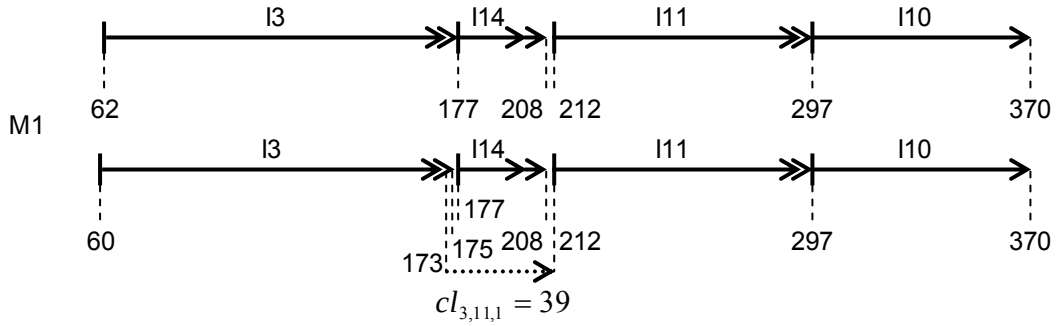

Figure 8. Part of the optimal schedule for example P3E. Optimal solution (above) and suboptimal solution (below) from continuous-time MILP with global precedence sequencing variables and CP models.

7.1.3. Makespan Minimization

The objective of makespan minimization is the most difficult for the multiple time grid continuous formulations. However, CT4I and CT3I can always find good solutions to the problem, see Table 4. Due to its significantly larger size, the MILPs resulting from CT4I tend to originate faster growing branch and bound search trees meaning that the solver runs out of memory faster for the two most difficult problems (P5M-P6M). Nevertheless, and despite the larger size, CT4I seems again to be slightly better despite CT3I being almost three orders of magnitude faster for P4M. The other continuous-time formulation, SV, has a similar performance when compared to the multiple time grid formulations and also suffers from the same problem of generating fast-growing trees.

The CP model is the best overall performer for makespan minimization since it is the fastest for P1M-P3M and can also find the best solution for P5M, for which the optimal solution is still

unknown. Note, however, that its performance for P6M is rather weak, since the best solution found after more than 15 h of computational time, 237, is still far from the best known solution of 164. It is also clear from Table 4, that, as the number of orders and equipment units increases, so does the computational effort.

The DT model has the merit of finding the best solution for P6M and doing so while solving an approximated version of the problem (δ=2). It is fair to say, based on the values of δ that can be used, that the performance of DT for makespan minimization lies between that observed for total earliness and total cost minimization. The fact that it is better than for total cost minimization is somewhat surprising since makespan minimization usually involves several iterations[22,26] before the optimal solution is found. In this respect it is worthwhile mentioning that the observed DT peak performance for P4M is simply because the predicted minimum number of time intervals ensures feasibility and hence only one iteration is required in the search for the optimal solution. This unusual behavior (for example P1M requires a total of 53 iterations) is simply because the bottleneck for P4M lies with the release date of a particular order. In other words, the optimal makespan is equal to the earliest possible ending time of that order.

7.2. Multistage Problems

The seven multistage problems under consideration range from 8 orders in 6 units and 2 stages (16 batches) to 15 orders in 4 units and 2 stages (30 batches) and to 8 orders in 8 units and 4 stages (32 batches).

7.2.1. Total Cost Minimization

The results given in Table 5 show that the CP and SV models are the best performers for total cost minimization. While CP was always able to prove optimality, SV failed to do so for P9C (the solver ran out of memory after more than 12000 CPUs) but managed to solve all other problems in less than 7 CPUs. The continuous-time formulations were able to find the global optimal solutions for all problems except P12C. For that problem, both CT4I and CT3I could not find even a feasible solution up to the maximum resource limit of 48000 and 60000 CPUs, respectively. This behavior was not totally unexpected since the multiple time grid model from which they originate exhibited

the same difficulties[22] when solving a problem also involving 4 stages. Finally, the discrete-time formulation is by far the worst performer and the gap is more significant than for single stage problems. Only relatively coarse (δ=5) time grids could be considered and even that value generated very difficult mathematical problems for P9C (no feasible solution) and P13C (unable to prove optimality after 100,000 CPUs). It is worth noting that we know for sure that P9C is feasible for (δ=5) since we were able to find a feasible solution for P9E for the same δ value (see Table 7).

7.2.2. Total Earliness Minimization

Like for single stage problems, the new CT4I formulation is the best performer for total earliness minimization (see Table 7). All problems can be solved to optimality in less than one hour and it was able to find the best solution for P13E. However, since optimality was proved for 9 time points only, we do not know for sure if this is in fact the global optimal solution. The other approaches were unable to confirm this finding: both SV and CP can only get to inferior solutions and DT can only solve an approximated version of the problem. We can assume that the best solution found for P11E is the global optimal solution since CT4I obtained the same result both for 6 (values reported in Table 7) and 7 (optimality proved in 5900 CPUs) time points. Concerning the discrete-time formulation, this objective enables us to use finer time grids (δ=2 for P7E and P8E) than for the two others but it is the worst formulation of the group.

7.2.3. Makespan Minimization

The results of Table 8 show that the CP model emerges as the best approach for makespan minimization since all problems can be solved to optimality in approximately one hour. With the exception of P13M, the SV model is also successful at finding the optimal solutions but fails to prove optimality for P8M, P11M and P13M. Concerning the novel multiple time grid formulations, CT3I is at least as competitive as SV and better than CT4I, which returned slightly worse solutions for P11M and P13M.

8. Overview of Main Features of Alternative Formulations

In order to mark the end of an extensive comparison between several conceptually different approaches, of which this paper is the third part, we find it convenient to summarize their main characteristics and suggest a ranking. Table 9 provides the most relevant conclusions.

8.1. Time Grid-based Continuous-Time Formulations

The research[21,22] has shown us that the use of a single time grid to solve single/multistage problems, not involving shared resources such as utilities or manpower, is clearly a bad option. The results were based on the RTN formulation of Castro et al[14], which involves the need to specify both the number of time points and the number of time intervals that any task can span. Although recent developments[16] have brought a formulation that does not need the latter specification to achieve a good performance in multipurpose problems, we believe that the main drawback of such single time grid formulations lies with the large value of $|T|$ that is required to find global optimal solutions to the problem. This problem can become more severe when sequence dependent changeover tasks are involved, since the end of the processing task and its subsequent cleaning task will generally occur at different time points. Naturally, the use of a single time grid is more advantageous to model the transfer of material between consecutive stages, so it can still be useful in small problems involving shared resources and not featuring sequence dependent changeovers.

In view of the above, the development of multiple time grid formulations was the next logical step. Their ability to fit any task, be it just processing or combined processing and changeover, into a single time interval, was a significant achievement, since it allowed to consider the minimum possible number of time points per grid, i.e. one time interval per order allocated to the equipment unit in question. Using a smaller number of time points in any given time grid seems to be the key issue since the developed multiple time grid formulation actually requires more time points in total ($|T|\times|M|$), than those used by single time grid formulations. The use of multiple time grids also allowed for a very efficient way of dealing with the objective of total earliness minimization, mainly because it is the only continuous-time mathematical programming approach, to the best of our knowledge, for which the solution of the relaxed problem (LP) can differ from zero, corresponding

to the lowest integrality gaps. Despite the fact that it requires substantial changes to deal with sequence dependent changeovers, it was found to be the best performer in single stage problems. In multistage problems, it becomes more difficult to model the transfer of material between consecutive stages and this becomes evident by its increasingly poor performance as the number of stages increases. Other important characteristics are its overall consistency irrespective of the objective function under consideration and solution dependency on the selected number of time points, which inevitably leads to an iterative procedure in the search for the optimal solution.

8.2. Continuous-Time Formulation based on Global Precedence Sequencing Variables

Not having to rely on explicit time grid(s) has the obvious advantage of needing to solve every problem only once in the search for the global optimal solution. The use of global precedence sequencing variables also leads to mathematical problems involving fewer binary variables. Furthermore, the model is basically the same irrespective of sequence dependent changeovers being involved or not. Its main advantage is the ability to find very good solutions, very fast, which when allied to sufficient computational effort, translates into the formulation being able to often find the global solutions for these test problems, even though it is sometimes impossible, due to rapidly growing tree sizes, to prove optimality. As a result, it has to be considered the best approach for multistage problems, even though, as it was shown, it has an important limitation since in exceptional cases involving sequence dependent changeovers, it may cut-off the optimal solution from the feasible space.

8.3. Discrete-time formulation

The discrete-time formulation is not a clear-cut case. For problems without sequence dependent changeovers it should always be considered as an option. The fact that it relies on a single time grid, which facilitates the modelling of shared resources, together with its characteristically low integrality gap, when compared to the continuous-time formulation, makes it very efficient even for very large problem sizes, particularly for total earliness minimization in single stage problems. Naturally, there is always the issue of discretizing the time horizon, which is often viewed as a disadvantage since it usually means considering approximate versions of the problem by rounding

its data. However, it can also be regarded as an advantage, since it gives an obvious way of relaxing the problem (by increasing the interval length and hence decreasing the number of time intervals) and still come up with very good solutions if feasibility is not compromised by the rounding errors. For this reason, the computational performance is less dependent than its continuous-time counterparts on issues like number of orders, units and stages. The addition of sequence dependent changeovers leads to a significant increase in the number of binary variables to be considered and consequently in the model size, causing a significantly decrease in performance, more so in multistage problems. Another disadvantage of the discrete-time formulation is that it needs an iterative procedure for makespan minimization that may involve several iterations and a large amount of computational resources before the optimal solution is found.

8.4. Formulations relying on Constraint Programming

Constraint programming formulations are a good option since OPL Studio global constraints and special constructs make the model quite competitive. When compared to the alternative approaches it is significantly better when sequence dependent changeovers are involved. Concerning the different objective functions, it is clear that the performance of the CP model improves as the number of variables involved in the objective function decreases, excelling for makespan minimization, which is in agreement with observations by Hooker[3]. Similarly to the single time grid approaches, handling of shared resources should not be a problem. Its main disadvantage lies in the fact that the first feasible solutions are usually not as good as those of the mathematical programming approaches, and that the optimal solution is usually found much later in the search, which implies that considerable computational time may be required to find a good suboptimal solution (note that search strategies other than the default for the MILP and CP solvers may eventually lead to different conclusions). Also, it only handles integer data, a limitation that although easily overcome by a change of basis to consider real numbers, leads to a significant decrease in performance, in what is a very similar effect to that observed in the discrete-time formulation when the number of time intervals is increased.

Finally, the hybrid MILP/CP model is a very good alternative particularly for single stage problems where the objective function depends solely on the assignment variables and not on the

sequencing variables, e.g. total cost minimization. The problem is divided into two parts, the first (master problem) finds the optimal assignments of orders to equipment units, which may or not be feasible in one or more units. Feasibility is then checked by solving CP feasibility problems for every equipment unit. Several iterations are usually required before the first feasible assignments, which are also optimal, are found. This is a clear disadvantage since for large problems, the simplified MILP can still be very difficult to solve, which means going through just a small number of iterations in considerable computational time and yet the possibility of ending up with no feasible solution at all. For other problem types, hybrid MILP/CP models have not proved as powerful. Overall, the development of successful hybrid methods is directly linked to the finding of a tight MILP master problem, the relaxed scheduling problem, which necessarily includes valid and efficient integer cuts resulting from the solution of the CP subproblem (either a feasibility or an optimization problem) in previous iterations (see Hooker[27] and Maravelias and Grossmann[4]).

9. Conclusions

This paper has presented two new continuous-time formulations for the short-term scheduling of single/multistage, multiproduct batch plants, where equipment units are subject to sequence dependent changeovers and product orders to both release and due dates. The formulations rely on the use of multiple time grids, one per equipment resource and are extensions of previous work[21,22]. Their main difference lies on the consideration of changeover tasks. While one formulation uses binary variables linked to such tasks, giving rise to 4-index binary variables, the other maintains the 3-index binary variables of the previous model[22] and changes one set of constraints to make it possible to handle sequence dependent changeovers. The form of such constraints forces tasks to be assigned to time points in a decreasing order, which is contrary to the technique developed[21] for total earliness minimization since it relies on tasks being assigned to time points in an increasing order, i.e. in the opposite way. Both formulations were shown to be very efficient in single stage problems with the most surprising result coming from the fact that the 4-index binaries formulation was found to be slightly better than its 3-index binaries counterpart, despite featuring a number of binary variables that can be up to one order of magnitude larger. This behaviour results from the use of tighter timing constraints (measured by a lower integrality gap) by the former, where all tasks that

can be executed in a given unit and time interval are aggregated into the same set of constraints, whereas the 3-index binaries formulation has the corresponding constraint disaggregated due to the need to consider one task defining order index in the constraint domain. As the number of stages increases, the performance of the developed multiple time grid formulations decrease steadily and feasibility may even be compromised.

The other goal of the paper has been to provide a critical review of other approaches that are suitable for this specific type of scheduling problem. These included an RTN-based discrete-time formulation[18], a continuous-time model with global precedence sequencing variables[5], a constraint programming model[5] and a hybrid MILP/CP model[6] (this last one just for single stage and total cost minimization problems). A total of 39 examples were solved and the results, together with those of the two previous works[21-22], allowed us to identify the main features, strengths and weaknesses of each approach, which were thereafter summarized in a comprehensive table.

Finally, it is reasonable to present our views on what we believe to be the best model, the continuous-time formulation with global precedence sequencing variables. Even though other approaches may perform significantly better in some problems, in particularly the multiple time grid formulation when minimizing total earliness and the constraint programming model when minimizing makespan, there are a few arguments that give an edge to that formulation, the most important being its ability to always find very good solutions with modest computational effort. This will be a critical point when considering real-world applications consisting of hundreds of batches, dozens of pieces of equipment and long scheduling periods, where to guarantee optimality is no longer the major issue. Preliminary work has shown that decomposition techniques based on that same model can be applied for the efficient solution of industrial based problems and this will be the subject of our future work on this subject.

Nomenclature

Sets/Indices
I/i, i'= process orders
$I_{i',m}$=orders that can precede order *i*' in unit *m*
$I_{i',m,t}$, $\bar{I}_{i',m,t}$ =orders that can be followed by order *i*' in unit *m* starting at time point *t*
I_m= orders to be processed in unit *m*
K/k=process stages

K_m=stage where unit m belongs
M/m= process equipment units
M_i=machines that can process order i
M_k=machines belonging to stage k
$T/t, t', t''$=Points of the time grid

Parameters

$c_{i,m}$=processing cost of order i in machine m
$cl_{i,m}^{\min}$ =minimum changeover time from order i in unit m
$cl_{i,m}^{\max}$ =maximum changeover time from order i in unit m
$cl_{i,i',m}^{\Delta}$ =difference between maximum and actual changeover from order i to i' in unit m
$cl_{i,i',m}$=duration of changeover task from order i to i' in unit m
d_i=due date of order i
$\bar{d}_i$ =normalised due date of order i
H=time horizon
$hb_{i,m}$= highest time at which order i can start to be processed in unit m
$lb_{i,m}$=lowest time at which order i can start to be processed in unit m
$\bar{lb}_{i,m}$ = lowest possible starting point of order i in unit m
$lbm_{m,t}$=lowest time at which unit m at time point t can become active
$p_{i,m}$=processing time of order i in machine m
r_i=release date of order i
$\bar{r}_i$ =normalised release date of order i
$ub_{i,i',m}$=highest time at which order i (followed by order i') can start to be processed in unit m
$\bar{ub}_{i,i',m}$ =highest starting point of order i (followed by order i') in unit m
δ=duration of each time interval in the discrete-time grid
$\tau_{i,m}$=processing time of order i on machine m as an integer multiple of δ
$\bar{\tau}_{i,i',m}$ = duration of combined processing and cleaning task of order i to i' in unit m as an integer multiple of δ

Variables

$C_{i,m,t}$=excess amount of equipment state associated to order i of unit m at time point t
$C_{i,m}^{0}$ =initial amount of equipment state associated to order i of unit m
MS=makespan
$N_{i,m,t}$=binary variable that assigns the start of order i in unit m to time point t
$\bar{N}_{i,i',m,t}$ =binary variable that assigns the start of order i (followed by i') in unit m at time point t
$R_{m,t}$=excess amount of machine m at time point t
$S_{i,k,t}$=excess amount of material resulting for order i produced at stage k at time point t
$T_{i,k}^{f}$ =ending time of order i in stage k
$T_{t,m}$=absolute time of event point t in unit m
$TD_{i,k}$=transfer time of order i in stage k
$x_{i,i',k}$=binary global precedence sequencing variable of order i over i' in stage k
$y_{i,m}$=binary assignment variable of order i to unit m

List of Tables

Table 1. Single stage problems: Overview of computational performance (CPU s) for total cost minimization.

problem/model	optimum	CT4I	CT3I	SV	DT	CP	MILP/CP
P1C (|I|=12, |M|=3)	101	10.0	11.5	118	713[d]	0.75	13.4
P2C (|I|=12, |M|=3)	87	5.42	2.73	33.1	1250[e]	0.19	1.11
P3C (|I|=15, |M|=5)	121	23.2	28.8	20000[a]	1395[e]	133	37.4
P4C (|I|=15, |M|=5)	106	3.80	1.88	510	2602[f]	15.1	6.81
P5C (|I|=20, |M|=5)	163	66.9	4996	12217[b]	551[g]	57000[i]	6935
P6C (|I|=20, |M|=5)	146	27.9	39.9	15844[c]	2084[h]	7620	83.3

AS=approximate solution of the problem, δ value within brackets where 1 corresponds to considering the exact problem data. FTP=fewer time points were used than those required to find the optimal solution, |T| value within brackets. BPS=best possible solution at the time of termination. MRL=maximum resource limit exceeded. NS=no solution found. OM=solver ran out of memory. SO= suboptimal solution returned.

[a]MRL, BPS=119.09.[b]OM, BPS=155.42. [c]OM, BPS=143.14. [d]AS (δ=5), SO=105. [e]AS(δ=2). [f]AS(δ=2), SO=107. [g]AS(δ=5). [h]AS(δ=5), SO=147.[i]MRL, SO=166.

Table 2. Computational statistics for problem P5C

model	CT4I	CT3I	SV	DT	CP	MILP/CP
|T|	6	6	-	87	-	-
discrete variables	6924	530	290	45360	-	-
single variables	7655	561	311	54596	120	-
constraints	731	621	1946	9161	180	-
RMIP	157.59	155.78	152.45	160.36	-	-
Obj	163	163	163	163[b]	166	163
CPU	66.9	4996	12217[a]	551	57000[c]	6935
nodes/choice points/major iterations	2141	1.69E6	2.34E6	178	1.84E8	211

[a]OM, BPS=155.42. [b]AS(δ=5). [c]MRL.

Table 3. Single stage problems: Overview of computational performance (CPU s) for total earliness minimization.

problem/model	optimum	CT4I	SV	DT	CP
P1E (|I|=12, |M|=3)	690	4.28	427	451	2.36
P2E (|I|=12, |M|=3)	146	4.11	653	190	92.1
P3E (|I|=15, |M|=5)	559	17.9	6842[b]	4506	11146[g]
P4E (|I|=15, |M|=5)	54	4612	9059[c]	169	3058
P5E (|I|=20, |M|=5)	1187	208	9821[d]	3614[f]	83000[h]
P6E (|I|=20, |M|=5)	150	62.3[a]	6259[e]	522	142000[i]

[a]SO=164, FTP(|T|=5), OM for |T|=6 with worse solution. [b]OM, SO=667, BPS=0. [c]OM, BPS=0. [d]OM, SO=1458, BPS=0. [e]OM, SO=190, BPS=0. [f]AS(δ=5), SO=1230. [g]SO=561, although solver solved to optimality (special case). [h]MRL, SO=1214. [i]MRL, SO=767.

Table 4. Single stage problems: Overview of computational performance (CPU s) for makespan minimization.

problem/model	optimum	CT4I	CT3I	SV	DT	CP
P1M (|I|=12, |M|=3)	409	15.5	193	17.5	8073[g]	0.98
P2M (|I|=12, |M|=3)	171	3.55	11.7	14.9	29509	0.84
P3M (|I|=15, |M|=5)	291	201	8373	17255	980[h]	122
P4M (|I|=15, |M|=5)	147	1435	1.76	2.42	61.5	702
P5M (|I|=20, |M|=5)	337?	8782[a]	36692[c]	8011[e]	8311[i]	54000[k]
P6M (|I|=20, |M|=5)	164?	6290[b]	20000[d]	16062[f]	13891[j]	55000[l]

[a]OM, SO=338, BPS=330. [b]OM, SO=168, BPS=158. [c]OM, SO=347, BPS=319. [d]MRL, SO=166, BPS=150. [e]OM, SO=347, BPS=290. [f]OM, SO=167, BPS=147. [g]AS(δ=2), SO=410. [h]AS(δ=5), SO=295. [i]AS(δ=5), SO=340. [j]AS(δ=2). [k]MRL, best solution=337. [l]MRL, SO=237.

Table 5. Multistage problems: Overview of computational performance (CPU s) for total cost minimization.

problem/model	optimum	CT4I	CT3I	SV	DT	CP
P7C (|I|=8, |M|=6, |K|=2)	18	3.76	0.64	0.5	440[d]	17.2
P8C (|I|=8, |M|=6, |K|=2)	1087	0.91	1.95	2.3	262[d]	159
P9C (|I|=8, |M|=6, |K|=3)	82	1481	1773	12014[c]	55500[e]	49.7
P10C (|I|=8, |M|=6, |K|=3)	647	3.97	1.62	0.38	13754[f]	70.2
P11C (|I|=12, |M|=6, |K|=2)	71	112	5.88	6.20	1181[d]	11623
P12C (|I|=8, |M|=8, |K|=4)	125	48000[a]	60000[b]	4.81	304[g]	1075
P13C(|I|=15, |M|=4, |K|=2)	96	481	85.6	2.88	100000[h]	924

[a]MRL, NS, BPS=121.4. [b]MRL, NS, BPS=118.3. [c]OM, BPS=79.2. [d]AS(δ=5). [e]MRL, NS(δ=5). [f]AS(δ=5), SO=649. [g]AS(δ=5), SO=132. [h]MRL, AS(δ=5), SO=101, BPS=95.53.

Table 6. Computational statistics for problem P12C

problem/model	CT4I	CT3I	SV	DT	CP
|T|	6	6	-	56	-
discrete variables	1796	368	176	5648	-
single variables	2317	441	209	11537	160
constraints	1153	1041	545	5865	360
RMIP	111.14	111	111	123.86	-
Obj	-	-	125	132[c]	125
CPU	48000[a]	60000[b]	4.81	304	1075
nodes/choice points	389500	6.34E6	8866	362	4351343

[a]MRL, NS, BPS=121.4. [b]MRL, NS, BPS=118.3. [c]AS(δ=5), SO=132.

Table 7. Multistage problems: Overview of computational performance (CPU s) for total earliness minimization.

problem/model	optimum	CT4I	SV	DT	CP
P7E (|I|=8, |M|=6, |K|=2)	88	1.56	2.76	16672[c]	5.58
P8E (|I|=8, |M|=6, |K|=2)	90	1.38	48.2	1339[c]	8.41
P9E (|I|=8, |M|=6, |K|=3)	217	19.7	15.3	1258[d]	127
P10E (|I|=8, |M|=6, |K|=3)	99	3.09	278	187[e]	8.66
P11E (|I|=12, |M|=6, |K|=2)	209	216	60000[a]	520[f]	60000[i]
P12E (|I|=8, |M|=8, |K|=4)	150	143	315	163[g]	2963
P13E (|I|=15, |M|=4, |K|=2)	571?	3448	2500[b]	60000[h]	55000[j]

[a]MRL, BPS=196. [b]OM, SO=600, BPS=157. [c]AS(δ=2). [d]AS(δ=5), SO=230. [e]AS(δ=5), SO=105. [f]AS(δ=5). [g]AS(δ=5, obj=148). [h]MRL, AS(δ=5), SO=950, BPS=597.6. [i]MRL, SO=243. [j]MRL, SO=848.

Table 8. Multistage problems: Overview of computational performance (CPU s) for makespan minimization.

problem/model	optimum	CT4I	CT3I	SV	DT	CP
P7M (|I|=8, |M|=6, |K|=2)	542	60.2	33.9	21.8	156[h]	2.06
P8M (|I|=8, |M|=6, |K|=2)	584	40.9	8.89	55000[e]	228[i]	0.27
P9M (|I|=8, |M|=6, |K|=3)	915	778	8.83	71.5	150000[j]	60.1
P10M (|I|=8, |M|=6, |K|=3)	914	87.3	18.0	1439	66118[k]	2.00
P11M (|I|=12, |M|=6, |K|=2)	233	60000[a]	14082	23408[f]	15430[l]	1013
P12M (|I|=8, |M|=8, |K|=4)	265	36541	54440	496	11315[m]	732
P13M (|I|=15, |M|=4, |K|=2)	273	60000[c]	60000[d]	8714[g]	60000[n]	3721

[a]MRL, SO=234, BPS=231. [c]MRL, SO=313, BPS=270.7. [d]MRL, SO=299, BPS=263. [c]OM, BPS=79.2. [e]MRL, BPS=571. [f]OM, BPS=220.1. [g]OM, SO=294, BPS=247.7. [h]AS(δ=5), SO=550. [i]AS(δ=2), SO=586. [j]MRL, NS(δ=10). [k]AS(δ=10), SO=930. [l]AS(δ=5), SO=240. [m]AS(δ=5), SO=280. [n]MRL, NS(δ=5).

Table 9. Overview of the main characteristics of the tested approaches for the short-term scheduling of single/multistage multiproduct batch plants

Type of model	Continuous-time			Discrete-time	CP	Hybrid MILP/CP
Feature/Based on	Single time grid	Multiple time grids	Global precedence sequencing variables	Single, uniform time grid	Global constraints and special constructs	Sequence of assignment and feasibility problems
Sequencing of tasks	Implicit through assignment of tasks to ordered time points		Explicit through model variables	Implicit through assignment of tasks to ordered time points	Implicit through activities starting and ending times	
Modeling of material transfer between stages	Explicit through excess resource variables	----------Implicit in model constraints----------		Explicit through excess resource variables	Implicit through global constraint *precedes*	N.A.
Modeling of changeovers	Explicit through processing and cleaning extent variables	Either explicit through binary variables or implicitly in model constraints	Implicitly in model constraints	Explicit through combined processing and cleaning extent variables	Implicitly through transition matrices associated to activities	
Objective functions handled	----------Minimization of total cost, total earliness and makespan----------					Total cost minimization
A priori decisions that can affect final solution	Number of time points; number of intervals any task can span	Number of time points of each time grid	None	Duration of uniform time intervals	None	None
Single stage performance	Very poor	Very good	Good	Good	Good	Very good
Multistage performance	Poor	Good	Very good	Fair	Good	N.A.
Strengths	Handling of shared resources	Minimization of total earliness; overall consistency for other objectives	Ability to find very good solutions very fast; generates relatively small problem sizes	Ability to solve approximated versions of a problem; minimization of total earliness; handling of shared resources	Minimization of makespan; handling of shared resources	Minimization of total cost in single stage problems
Limitations	Solution is highly dependent on a priori decisions; only useful for very simple problems	Solution dependency on number of time points; inadequacy to handle shared resources; can fail to find feasible solutions in problems involving several stages	Minor chance of leading to suboptimal solutions when dealing with sequence dependent changeovers; global optimality can be hard to prove; shared resources can be difficult to handle	Solution dependence on the chosen interval length; can lead to prohibitively large problem sizes when sequence dependent changeovers are involved; requires iterative procedure for makespan minimization where optimal solution is the first feasible one	Minor chance of leading to suboptimal solutions when dealing with sequence dependent changeovers; can stuck in poor solutions since optimization goes from bad to good solutions; handles integer data only	Only efficient for total cost minimization in single stage problems; can fail to find feasible solutions whenever simplified assignment problem is difficult to solve; first feasible solution is optimal; handles integer data only

References

(1) Méndez, C.A.; Cerdá, J.; Grossmann, I.E.; Harjunkoski, I.; Fahl, M. State-of-the-art Review of Optimization Methods for Short-Term Scheduling of Batch Processes. Comp. Chem. Eng. **2006**. In press.

(2) Hentenryck, P.V. Constraint satisfaction in logic programming; MIT Press: Cambridge, MA, 1989.

(3) Hooker, J.N. Logic, optimization and constraint programming. *INFORMS J. Comput.* **2002**, *14*, 295.

(4) Maravelias, C.T.; Grossmann, I.E. A Hybrid MILP/CP Decomposition Approach for the Continuous-time Scheduling of Multipurpose Batch Plants. Comp. Chem. Eng. **2004**, *28*, 1921.

(5) Harjunkoski, I.; Grossmann, I.E. Decomposition Techniques for Multistage Scheduling Problems using Mixed-integer and Constraint Programming Methods. *Comp. Chem. Eng.* **2002**, *26*, 1533.

(6) Jain, V.; Grossmann, I.E. Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing.* **2001**, *13*, 258.

(7) Maravelias, C. A Decomposition Framework for the Scheduling of Single and Multistage Processes. *Comp. Chem. Eng.* **2006**, *30*, 407.

(8) Roe, B.; Papageorgiou, L.G:, Shah, N. A hybrid MILP/CP algorithm for multipurpose batch process scheduling. *Comp. Chem. Eng.* **2005**, *29*, 1277.

(9) Giannelos, N.F.; Georgiadis, M.C. A Simple Continuous-Time Formulation for Short-Term Scheduling of Multipurpose Batch Processes. *Ind. Eng. Chem. Res.* **2002**, *41*, 2178.

(10) Giannelos, N.F.; Georgiadis, M.C. A Novel Event-Driven Formulation for Short-Term Scheduling of Multipurpose Continuous Processes. *Ind. Eng. Chem. Res.* **2002**, *41*, 2431.

(11) Maravelias, C.T.; Grossmann, I.E. New General Continuous-Time State-Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.* **2003**, *42*, 3056.

(12) Janak, S.L.; Lin, X.; Floudas, C.A. 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.

(13) Méndez, C.A.; Cerdá J. An efficient MILP continuous-time formulation for short-term scheduling of multiproduct continuous facilities. *Comp. Chem. Eng*. **2002**, *26*, 687.

(14) Castro, P.M.; Barbosa-Póvoa, A.P.; Matos, H.A.; Novais, A.Q. Simple Continuous-time Formulation for Short-Term Scheduling of Batch and Continuous Processes. *Ind. Eng. Chem. Res.* **2004**, *43*, 105.

(15) Gupta, S.; Karimi, I.A. An Improved MILP Formulation for Scheduling Multiproduct Multistage Batch Plants. *Ind. Eng. Chem. Res.* **2003**, *42*, 2365.

(16) Sundaramoorthy, A.; Karimi, I.A. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chem. Eng. Sci.* **2005**, *60*, 2679.

(17) Kondili, E.; Pantelides, C.C.; Sargent, R. A General Algorithm for Short-Term Scheduling of Batch Operations I. MILP Formulation. *Comp. Chem. Eng.* **1993**, *17*, 211.

(18) Pantelides, C.C. Unified Frameworks for the Optimal Process Planning and Scheduling. In *Proceedings of the Second Conference on Foundations of Computer Aided Operations*; Cache Publications: New York, 1994; pp 253.

(19) Maravelias, C.T. Mixed-Time Representation for State-Task Network Models. *Ind. Eng. Chem. Res.* **2005**, *44*, 9129.

(20) Floudas, C.A:; Lin, X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Comp. Chem. Eng.* **2004**, *28*, 2109.

(21) Castro, P.M.; Grossmann, I.E. An Efficient MILP Model for the Short-Term Scheduling of Single Stage Batch Plants. Comp. Chem. Eng. **2006**. In press.

(22) Castro, P.M.; Grossmann, I.E. New Continuous-Time MILP Model for the Short-Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res. **2005**, *44*, 9175.

(23) Hentenryck, P.V. *The OPL Optimization Programming Language*. MIT Press: Cambridge, MA, 1999.

(24) Castro, P. M.; Barbosa-Póvoa, A.P.; Novais, A.Q. A Divide and Conquer Strategy for the Scheduling of Process Plants Subject to Changeovers Using Continuous-Time Formulations. Ind. Eng. Chem. Res. **2004**, *43*, 7939.

(25) Pinto, J.; Grossmann, I. A Continuous Time Mixed Integer Linear Programming Model for the Short-Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res. **1995**, *34*, 3037.

(26) Maravelias, C.; Grossmann, I.E. Minimization of the Makespan with a Discrete-Time State-Task Network Formulation. Ind. Eng. Chem. Res. **2003**, *42*, 6252.

(27) Hooker, J. N. A Hybrid Method for Planning and Scheduling. In Lecture Notes in Computer Science, Vol. 3258. Editor: M. Wallace, Springer, 2004, pp 305-316.

List of captions for figures

Figure 1. RTN process representation for order I1, featuring a total of |M| machines and |K| stages (changes on the units cleaning states omitted for simplification).

Figure 2. RTN process representation for unit M1, showing all possible cleaning states (changes on the orders material states omitted for simplification).

Figure 3. Continuous-time grid employed (one for each equipment unit).

Figure 4. Uniform discrete-time grid.

Figure 5. Possible solution from multiple-time grid, continuous-time formulation F1 (|I|=3, |M|=3, |K|=3, and |T|=4).

Figure 6. Possible solution from multiple-time grid, continuous-time formulation F1 (|I|=3, |M|=3, |K|=1 and |T|=3). Orders are assigned from the first to last time points.

Figure 7. Possible solution from multiple-time grid, continuous-time formulation F2 (|I|=5, |M|=3, |K|=1 and |T|=3). Orders are assigned from the last but one to the first time points.

Figure 8. Part of the optimal schedule for example P3E. Optimal solution (above) and suboptimal solution (below) from continuous-time MILP with global precedence sequencing variables and CP models.