A New Continuous-time MILP Model for the Short-Term Scheduling of Multi Stage Batch Plants

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

[†]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 a new multiple time grid continuous time MILP model for the short-term scheduling of multistage, multiproduct plants. It can handle both release and due dates and different objective functions efficiently, such as the minimization of total cost, total earliness or makespan. This formulation is compared to other existing mixed-integer linear programming approaches and to a constraint programming model. The results show that the proposed formulation is much more efficient than its uniform time grid counterpart and is comparable to a continuous-time formulation that uses global precedence sequencing variables. Discrete-time formulations are preferred for larger scheduling problems where a reasonable number of time points are enough to consider the exact problem data. The results also show that the constraint programming model is the best approach for makespan minimization.

[*] To whom correspondence should be addressed. Tel.: +351-217162712. Fax: +351-217167016. E-mail: pedro.castro@ineti.pt

1. Introduction

The research area of batch processing scheduling has received considerable attention in the past two decades. Traditionally, scheduling approaches have been classified according to different criteria[1]. The type of network structure under consideration is one possibility and can range from the simplest single machine problem, where the goal is to find a feasible or optimal order sequence, to more complex multipurpose plants, where each product may go through a different series of processing steps. In between, we have multistage, multiproduct plants, which will be the subject of this paper, where all products follow the same sequence of operations, one per stage. Another classification criterion is the type of method used. Well known examples are mathematical programming, generalized disjunctive programming (GDP) (Raman and Grossmann[2]), which can be reformulated as mathematical programming models, and constraint programming (CP) (Hentenryck[3]). The type of time representation is yet another possibility, which can be either treated implicitly through the model constraints (Mendez and Cerdá[4], Harjunkoski and Grossmann[5], Gupta and Karimi[6]) or explicitly through the timing variables. In the later case, the time horizon of interest is divided into several time intervals, which may have fixed or variable duration. The first option gives rise to discrete time formulations (Kondili et al.[7], Pantelides[8]), while the second option to continuous-time formulations. Continuous-time grids may employ a uniform time grid (Maravelias and Grossmann[9], and Castro et al.[10]) or multiple time grids, one time grid per plant equipment (Giannelos and Giorgiadis[11], Janak et al.[12]). Since constraint programming models can only handle integer processing data, they can be viewed as discrete-time models.

The size and complexity of mathematical programming models is greatly dependent on how time is treated. Discrete-time formulations use a much larger number of time intervals and sometimes may need to consider an approximated version of the problem to maintain problem tractability. Despite their large size, the resulting MILPs are very tight and usually require few nodes to find the optimal solution. Continuous-time formulations are more general but are also much more complex meaning that the size of the problems that can be solved to optimality is usually smaller. A recent review of discrete and

continuous-time approaches can be found in (Floudas and Lin[13]). For the specific case of single stage multiproduct plants, Castro and Grossmann[14] also compared discrete and continuous-time MILP formulations with the results generally favouring a discrete-time Resource Task Network (Pantelides[8]) formulation over its continuous-time, CP and hybrid MILP/CP counterparts. Concerning the continuous-time formulations, their proposed multiple time grid formulation was also shown to be better than the equivalent uniform time grid formulation and the MILP model of Jain and Grossmann[15], which uses assignment and sequencing variables instead of an explicit time grid, for the objectives of total cost minimization and total earliness minimization.

This paper extends the work of Castro and Grossmann[14] for single stage multiproduct plants to the more complex multistage case in which release and due dates are given and changeover terms are neglected. As will be shown, the model must account not only for the adequate timing of events occurring on a particular machine, but also for the interaction of machines processing the same order in consecutive stages. This poses considerable difficulties when considering multiple time grids since the machines are no longer independent. New sets of constraints are proposed to extend the multiple time grid continuous-time formulation to the multiple stage case. The new formulation is compared to RTN-based discrete and continuous-time formulations (uniform time grids), and to the continuous-time and constraint programming models of Harjunkoski and Grossmann[5]. The comparison is performed for three widely used objective functions: minimization of total cost, total earliness and makespan. For the latter we propose an efficient algorithm to be used by the discrete-time formulation, based on the work of Maravelias and Grossmann[16] that can find the optimal solution in a relatively low number of iterations.

The rest of the paper is structured as follows. Section 2 describes the scheduling problem under consideration. Section 3 presents existing Resource Task Network formulations that can be used to solve the problem at hand. Concerning the discrete-time formulation a new algorithm is proposed to tackle the objective of makespan minimization. The new multiple-time grid continuous-time formulation is given in section 4. Section 5 describes two other existing approaches that can be used, while section 6 features the computational studies. Finally, the conclusions are left for section 7.

2. Problem definition

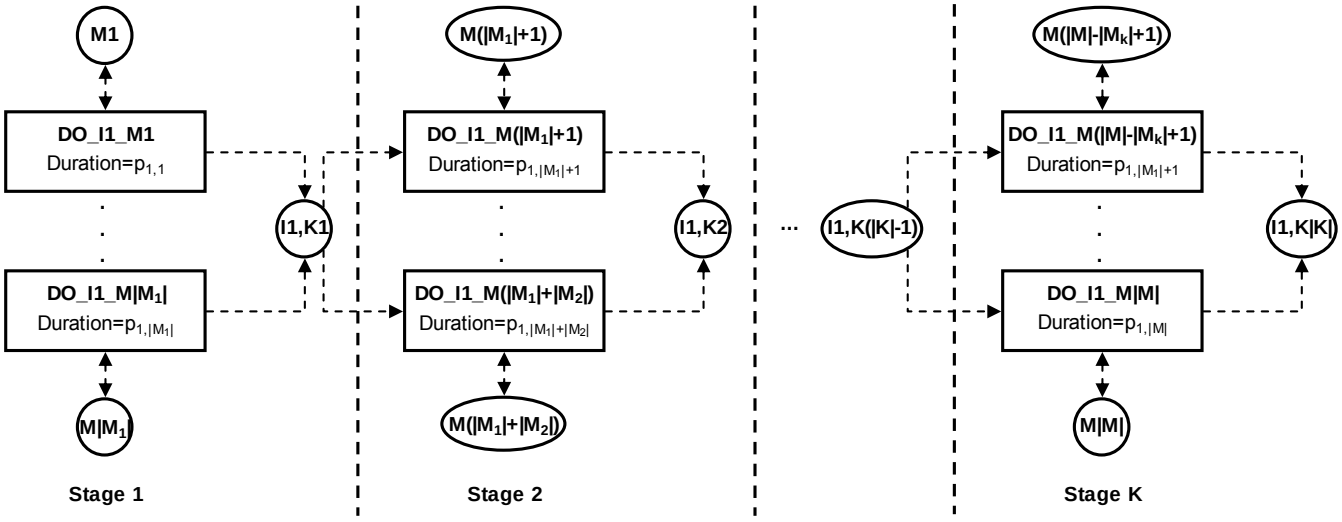

Figure 1. RTN process representation for order I1, featuring a total of |M| machines and |K| stages

In this paper, the short-term scheduling problem of multistage multiproduct batch plants is considered. A set I of product orders must go through a sequence of processing stages, belonging to set K, to reach the condition of final products. It is assumed that all orders go through all stages and that there is a unique sequence of stages for all orders. A total of M units are available, where a given unit can process all orders belonging to set I_m. Each unit belongs to a single stage, with set M_k including all units belonging to stage k. The processing times, $p_{i,m}$, release, r_i, and due dates, d_i, are assumed to be known, the latter being enforced as hard constraints. It is assumed that sequence dependent changeovers can be neglected. The RTN representation of the multistage, multiproduct plant is given in Figure 1 for a given order, e.g. I1. The processing task, designated DO, consumes the material corresponding to the order being processed, which must be at an appropriate state. The state of material is directed associated to the stage where it is produced, i.e. stage k consumes material at state k-1 and produces material at state k. Some of the existing mathematical formulations consider the material states explicitly as model variables, while others consider them implicitly, in some of the model constraints. The former rely on uniform time grids, while the latter rely on multiple time grids or no explicit time grid at all. Finally, we assume unlimited intermediate storage (UIS).

3. General RTN formulations for multipurpose plants adapted to the multistage case

General Resource Task Network formulations can be used to tackle the specific type of problem considered in this paper. These include the discrete-time formulation of Pantelides[8] and the continuous-time formulation of Castro et al.[10]. This section presents the two formulations, although different indices are used on the model variables and constraints (when compared to the original works) to take advantage of the multistage plant structure. The discrete and continuous-time formulations use a single time grid to keep track of the events taking place and similar sets of variables and constraints, but, as will be seen later on, give rise to Mixed Integer Linear Programs of different size and complexity.

3.1. Discrete-time formulation (F1)

The discrete-time formulation uses a uniform time grid, where all |T|-1 time intervals have the same duration, δ (see Figure 2). Usually, δ is chosen as the greatest common factor of the processing times, $p_{i,m}$, unless this implies an exceedingly large number of intervals. In such case, to maintain problem tractability, a higher value of δ is used and the problem data is rounded to a multiple of δ. This means that the processing times $p_{i,m}$ are converted to $\tau_{i,m}$, the duration of order i in unit m in number of time intervals, using the following formula: $\tau_{i,m} = \text{CEIL}(p_{i,m} / \delta)$, where CEIL is a function that rounds the argument to the next integer. Whenever the exact problem data is not considered, no guarantee exists that the optimal solution from the discrete-time formulation is the true optimal solution. Usually, lower rounding errors will lead to better approximations. Rounding up the processing times ensures that the solutions from the cost minimization and makespan minimization problems are upper bounds on the true optimal solution and the same is generally true for the earliness minimization problem. The approximated release dates are calculated through $\bar{r}_i = \text{CEIL}(r_i / \delta)\delta$, with the minimum value representing the absolute time of the first time point. The same can be done to determine the approximated due dates, $\bar{d}_i$, with the maximum value representing the absolute time of the last time point (see Figure 2).

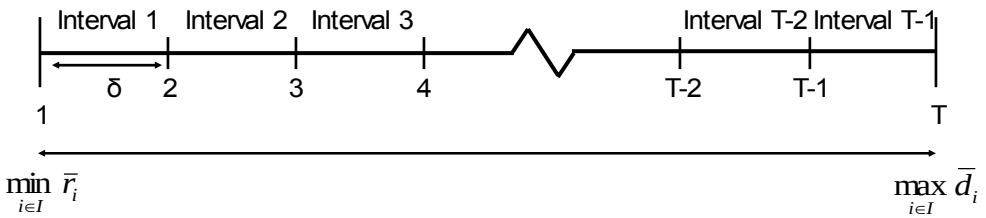

Figure 2. Uniform time grid for discrete-time formulation

The discrete-time formulation can be written in a very compact form. It uses three sets of variables and constraints for the general case plus the objective function. The binary variables $N_{i,m,t}$, denote the execution of task *i* on unit *m* at time *t*, whereas the excess resource variables denote the availability of a resource at a given time point. To facilitate the understanding of the model, we define one set of variables for each type of resource: $R_{m,t}$ for the equipment resources (the units, *m*) and $S_{i,k,t}$ for the material resources (state of material resulting from order *i* produced at stage *k*). These are defined as nonnegative continuous variables, although the excess resource balances (eqs 1 and 2) ensure that they can only take 0, 1 values. It is important to mention that not all binary variables $N_{i,m,t}$ need to be considered. Since we are dealing with multistage problems and the orders have release and due dates, appropriate sets $I_{t,m}$ and $M_{i,t}$ can be defined to include the orders that can start at point *t* on machine *m*, and the machines that can start processing order *i* at time *t*, respectively. This preprocessing step significantly reduces the number of binary variables of the model.

The excess resource balances are typical multiperiod balances, where the availability of a resource at time point *t* is equal to that at the previous time point, minus one if there is a task that consumes the resource (starting at *t*, second term on the right hand side), plus one if there is a task ending at *t* (starting at $t-\tau_{i,m}$, third term on the right hand side). Note in eq 2 that the state of material from order *i* at stage *k* is consumed whenever order *i* is processed on a unit belonging to stage $k+1$ ($m \in M_{k+1}$), and produced whenever it is processed on a unit belonging to stage *k*, see also Figure 1. Eq 3 ensures that all orders are processed on all stages and eq 4 avoids forbidden paths, i.e. orders processed on unit *m* cannot proceed to unit *m'* if that path is forbidden $(m,m') \in FP$.

$$R_{m,t} = (1\big|_{t=1} + R_{m,t-1}\big|_{t\neq 1}) - \sum_{i\in I_{t,m}} N_{i,m,t} + \sum_{i\in I_{t-p_{i,m},m}} N_{i,m,t-\tau_{i,m}} \quad \forall m \in M, t \in T \tag{1}$$

$$S_{i,k,t} = S_{i,k,t-1} - \sum_{\substack{m\in M_{k+1}\\ m\in M_{i,t}}} N_{i,m,t} + \sum_{\substack{m\in M_{k}\\ m\in M_{i,t-\tau_{i,m}}}} N_{i,m,t-\tau_{i,m}} \quad \forall i \in I, k \in K, t \in T \tag{2}$$

$$\sum_{t\in T_{i,m}} \sum_{\substack{m\in M_i\\ m\in M_k}} N_{i,m,t} = 1 \; \forall i \in I, k \in K \tag{3}$$

$$\sum_{t\in T_{i,m}} N_{i,m,t} + \sum_{t\in T_{i,m'}} N_{i,m',t} \leq 1 \; \forall i \in I, m \in M_i, m' \in M_i, (m,m') \in FP \tag{4}$$

Three different objective functions are considered in this paper. The discrete-time formulation can handle total cost minimization (eq 5) and total earliness minimization (eq 6) in a direct way, whereas a different approach must be used for makespan minimization (described in section 3.1.1). In eq 6, the second term on the right hand side represents the time corresponding to t=1 multiplied by the total number of orders, while the third term represents the distance (in real time) from the ending point of all orders in the last stage, to the beginning of the time horizon.

$$\min \sum_{t\in T_{i,m}} \sum_{m\in M_i} \sum_{i\in I} N_{i,m,t} c_{i,m} \tag{5}$$

$$\min \mathrm{Z} = \sum_{i\in I} \bar{d}_i - |I| \cdot \min_{i'\in I} \bar{r}_{i'} - \delta \cdot \sum_{t\in T_{i,m}} \sum_{\substack{m\in M_i\\ m\in M_k}} \sum_{\substack{k\in K\\ k=|K|}} \sum_{i\in I} \left[N_{i,m,t} (t + \tau_{i,m} - 1) \right] \tag{6}$$

3.1.1. Proposed algorithm for makespan minimization

In the discrete RTN-based formulation, the cardinality of set T (number of time points) is fixed. Since the duration of each time interval is also fixed, the time corresponding to the last time point is known a priori so it is not clear how to minimize the makespan. One possible solution is to define a new task that consumes the final material states of all orders and then minimize the starting point of this new task. Unfortunately, the resulting formulation behaves poorly and a different approach nust be used.

Maravelias and Grossmann[16] have recently proposed an algorithm for minimizing makespan with State Task Network discrete-time formulations. Starting with a relatively small number of time points, the discrete-time STN model is solved iteratively (the number of time points is increased by one

between successive iterations) until a feasible solution is found. Although, this feasible solution may not be the optimal solution for the objective function used (the authors used production maximization), it is the optimal solution in terms of makespan, which is equal to the time corresponding to the last time point of set T.

The algorithm used in this work is basically the same of Maravelias and Grossmann[16]. There are however two differences due to the type of problem being considered, which is not a production maximization problem. The first concerns the objective function used. Since the discrete-time formulation has a better performance for total earliness minimization than total cost minimization (see section 6.2), the former objective was preferred. The second concerns the estimation of the lower bound for the cardinality of set T, i.e. the value of |T| to use in the first iteration: $|T|^0$. This was calculated as follows. First, we calculate the minimum number of time intervals required to complete order *i*, eft_i, by adding up the existing number of time intervals until the release of the order (first term on the right-hand side), to the lowest processing times of the order over all stages, eq 7. The maximum value among the earliest finishing time can set $|T|^0$, but usually this value is set following the calculation of the predicted completion time in all stages. These values are calculated trough eq 8, where the numerator is the sum over all orders of the minimum processing time in stage *k* and the denominator is the number of machines in that stage. Like with eft_i, mct_k is expressed in number of time intervals. The limiting stage, KL, is assumed to be the stage with longest maximum completion time, eq 9. Finally, $|T|^0$ is determined by adding the maximum completion time, to the time that it takes to complete processing the fastest order on the other stages, $k \notin KL$. Since all these durations are in time interval units, we need to add 1 to get the total number of time points in the grid (see Figure 2 and eq 10).

$$eft_i = (\bar{r}_i - \min_{i' \in I} \bar{r}_{i'}) / \delta + \sum_{k \in K} \min_{\substack{m \in M_i \\ m \in M_k}} \tau_{i,m} \tag{7}$$

$$mct_k = \mathrm{CEIL}((\sum_{i \in I} \min_{\substack{m \in M_i \\ m \in M_k}} \tau_{i,m}) / |M_k|) \tag{8}$$

$$KL = \{k \mid mct_k = \max_{k' \in K} mct_{k'}\} \tag{9}$$

$$|T|^0 = \max(\max_{i \in I} eft_i, \max_{k \in KL} mct_k + \sum_{\substack{k \in K \\ k \notin KL}} \min_{\substack{i \in I \\ m \in M_i \\ m \in M_k}} \tau_{i,m}) + 1 \quad (10)$$

The calculated values of $|T|^0$ are usually close enough to the value that ensures feasibility, meaning that a relatively small number of iterations are required to find the optimal solution for makespan minimization when compared to the total number of time points used, see section 6.3. However, special problem features like forbidden paths tend to widen the gap between the lower bound and the optimal value of |T|. Despite the large number of iterations required to find the optimal solution, this is a valid approach since the infeasible problems are solved very fast, especially when the number of time points is distant from the optimal value, and so is the first feasible problem, meaning that the total computational time is still reasonably low, even for large MILPs. Overall, both the total computational effort and number of iterations usually increase with a decrease in the interval length, δ. The proposed algorithm is concluded by adding a new set of constraints to the discrete-time formulation, to ensure that all orders end in the available time horizon. Eq 11 states that the final state of material must equal 1 at the last event point, for all orders.

$$S_{i,|K|,|T|} = 1 \; \forall i \in I \quad (11)$$

3.2. Continuous-time formulation (F2)

Continuous-time formulations require considerably fewer time points than their discrete-time counterparts to find the global optimal solution to a particular problem. However, their performance is much more dependent on the cardinality of set T meaning that this value must be selected carefully to avoid generating intractable mathematical problems. An increase of |T| by one often has a large effect on computational effort and because of this we need to solve a few MILPs, until the global optimal solution is found. Like the discrete-time formulation, the continuous-time formulation uses a uniform time grid, but now the duration of each time interval is not fixed (see Figure 3).

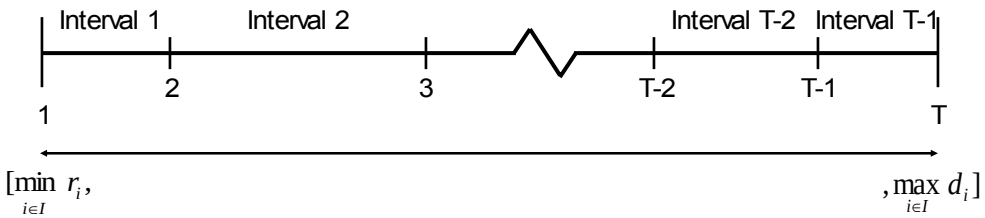

Figure 3. Uniform time grid for continuous-time formulation

The uniform-time grid continuous-time formulation[10] uses sets of variables that are similar to those used by the discrete-time formulation and the proposed multiple time grid formulation (see section 4). The main differences are the following. First, it is required to identify both the starting and ending point of a task, so the binary variables feature two time indices instead of one: $\overline{N}_{i,m,t,t'}$, where *t* identifies the starting time point and *t'* the ending time point. In order to decrease the complexity of the model it is assumed that the each task can only span a limited number of time intervals, i.e. $t' \leq t + \Delta t$, where Δt is a parameter specified by the user, which if too small, may exclude the global optimal solution, similarly to what happens with |T|. Second, since the time grid is common for all machines, the timing variables only feature the time index: T_t. It is important to note that some binary variables can be eliminated from the formulation by noting that at least one time interval is required per stage, meaning that orders processed in units belonging to stage 2 can only begin at time point 2 and so on. Likewise, in the other end of the time grid, orders processed in units belonging to stage |K|-1 can start at most at |T|-2 since at least one time interval must be left to process the order in the last stage and no order can start at the last time point. This information is used to define sets $I_{m,t,t'}$ and $M_{t,t'}$, which include all orders that can be processed on unit *m* starting at *t* and ending at *t'* and all units that can be active between points *t* and *t'*, respectively.

The model constraints are very similar to those used for the multiple time grid continuous-time formulation. Eqs 12 to 20 form the main block of the uniform time grid continuous-time formulation. For the minimization of total cost it is enough to add the objective function (eq 26), while for the minimization of total earliness, eqs 21 to 23, together with the objective function (eq 27) are required. Finally, for makespan minimization we need a new variable: MS (makespan), defined by eq 24. The

model is completed with eq 25 and the objective function (eq 28). Since the multiple time grid constraints will be explained thoroughly in the next section, here we only focus on the most significant differences. Besides the changes resulting from the use of a distinct set of binary variables, which causes some of the terms to use one more summation, or the constraints to be defined over one more time index, there are two important differences.

The use of a single time grid and the definition of material resources, like with the discrete-time formulation, means that eq 13 is sufficient to ensure that any given order can only be processed at stage *k* after stage *k*-1 has been completed. The uniform time grid formulation, however, does require the use of a new set of variables, DD_i, and a new set of constraints (eqs 21), to determine the delivery date of order *i*, whenever the objective is the minimization of total earliness. Eq 21 states that the delivery date of order *i* must not be greater than the absolute time of point *t* plus the order's processing time in the last stage if *i* starts to be processed in that stage, at time point *t*. It acts as an upper bound and since the solver will try to maximize all DD_i (see eq 27), no lower bounds are required. However, good lower bounds were found to improve the performance of the formulation and these are given in eqs 22 and 23. The former states that the delivery date of order *i* must not be lower than the ending time of the order in the final stage if it starts at or after time point *t*, while the latter states that the delivery date must not be lower than the absolute time of point *t* if the order ends at or before *t*.

$$R_{m,t} = (1\big|_{t=1} + R_{m,t-1}\big|_{t\neq 1}) - \sum_{i\in I_{m,t,t'}} \sum_{\substack{t'\in T \\ t<t'\leq t+\Delta t}} \overline{N}_{i,m,t,t'} + \sum_{i\in I_{m,t',t}} \sum_{\substack{t'\in T \\ t-\Delta t\leq t'<t}} \overline{N}_{i,m,t',t} \;\; \forall m\in M, t\in T \tag{12}$$

$$S_{i,k,t} = S_{i,k,t-1} - \sum_{\substack{m\in M_i \\ m\in M_{k+1} \\ m\in M_{t,t'}}} \sum_{\substack{t'\in T \\ t<t'\leq t+\Delta t}} \overline{N}_{i,m,t,t'} + \sum_{\substack{m\in M_i \\ m\in M_k \\ m\in M_{t',t}}} \sum_{\substack{t'\in T \\ t-\Delta t\leq t'<t}} \overline{N}_{i,m,t',t} \;\; \forall i\in I, k\in K, t\in T \tag{13}$$

$$\sum_{\substack{m\in M_i \\ m\in M_k \\ m\in M_{t,t'}}} \sum_{t\in T} \sum_{\substack{t'\in T \\ t<t'\leq t+\Delta t}} \overline{N}_{i,m,t,t'} = 1 \;\forall i\in I, k\in K \tag{14}$$

$$T_{t'} - T_t \geq \sum_{i\in I_{m,t,t'}} \overline{N}_{i,m,t,t'} p_{i,m} \;\; \forall m\in M_{t,t'}, t, t'\in T, t<t'\leq t+\Delta t \tag{15}$$

$$T_t \geq \sum_{i\in I_{m,t,t'}} \sum_{\substack{t'\in T \\ t<t'\leq t+\Delta t}} \overline{N}_{i,m,t,t'} (r_i + \sum_{\substack{k\in K \\ k\in K_m}} \sum_{\substack{k'\in K \\ k'<k}} \min_{\substack{m'\in M_{k'} \\ m'\in M_i}} p_{i,m'}) \;\forall m\in M, t\in T, t\neq |T| \tag{16}$$

$$T_t \le \sum_{i\in I_{m,t,t'}} \sum_{\substack{t'\in T \\ t<t'\le t+\Delta t}} \overline{N}_{i,m,t,t'}(d_i - \sum_{\substack{k\in K \\ k\in K_m}} \sum_{\substack{k'\in K \\ k'\ge k}} \min_{\substack{m'\in M_{k'} \\ m'\in M_i}} p_{i,m'}) + H(1-\sum_{i\in I_{m,t,t'}} \sum_{\substack{t'\in T \\ t<t'\le t+\Delta t}} \overline{N}_{i,m,t,t'}) \; \forall m\in M, t\in T, t\ne |T| \quad (17)$$

$$\sum_{t\in T_{i,m',t'}} \sum_{t'\in T} \overline{N}_{i,m,t,t'} + \sum_{t\in T_{i,m,t'}} \sum_{t'\in T} \overline{N}_{i,m',t,t'} \le 1 \; \forall i\in I, m\in M_i, m'\in M_i, (m,m')\in FP \quad (18)$$

$$T_t \ge \max_{i\in I} r_i \; \forall t\in T \quad (19)$$

$$T_t \le H = \max_{i\in I} d_i \; \forall t\in T \quad (20)$$

$$DD_i \le T_t + \sum_{\substack{m\in M_i \\ m\in M_k \\ m\in M_{t,t'}}} \sum_{k=|K|} \sum_{\substack{t'\in T \\ t<t'\le t+\Delta t}} \overline{N}_{i,m,t,t'} p_{i,m} + d_i(1-\sum_{\substack{m\in M_i \\ m\in M_k \\ m\in M_{t',t''}}} \sum_{k=|K|} \sum_{\substack{t'\in T \\ t<t'\le t+\Delta t}} \overline{N}_{i,m,t,t'}) \; \forall i\in I, t\in T, t\ne |T| \quad (21)$$

$$DD_i \ge T_t + \sum_{\substack{m\in M_i \\ m\in M_k \\ m\in M_{t,t'}}} \sum_{k=|K|} \sum_{\substack{t'\in T \\ t<t'\le t+\Delta t}} \overline{N}_{i,m,t,t'} p_{i,m} - H(1-\sum_{\substack{m\in M_i \\ m\in M_k \\ m\in M_{t',t''}}} \sum_{k=|K|} \sum_{\substack{t''\in T \\ t'<t''\le t'+\Delta t}} \sum_{\substack{t'\in T \\ t'\ge t}} \overline{N}_{i,m,t',t''}) \; \forall i\in I, t\in T, t\ne |T| \quad (22)$$

$$DD_i \ge T_t + d_i(1-\sum_{\substack{m\in M_i \\ m\in M_k \\ m\in M_{t'',t'}}} \sum_{k=|K|} \sum_{\substack{t''\in T \\ t'-\Delta t\le t''<t'}} \sum_{\substack{t'\in T \\ t'\le t}} \overline{N}_{i,m,t'',t'}) \; \forall i\in I, t\in T, t\ne 1 \quad (23)$$

$$MS = \sum_{t=|T|} T_t \quad (24)$$

$$T_t \le MS - \sum_{i\in I_{m,t,t'}} \sum_{\substack{t'\in T \\ t<t'\le t+\Delta t}} \overline{N}_{i,m,t,t'}(\sum_{\substack{k\in K \\ k\in K_m}} \sum_{\substack{k'\in K \\ k'\ge k}} \min_{\substack{m'\in M_{k'} \\ m'\in M_i}} p_{i,m'}) \; \forall m\in M, t\in T, t\ne |T| \quad (25)$$

$$\min \sum_{t\in T_{i,m,t'}} \sum_{t'\in T} \sum_{m\in M_i} \sum_{i\in I} \overline{N}_{i,m,t,t'} c_{i,m} \quad (26)$$

$$\min \sum_{i\in I} (d_i - DD_i) \quad (27)$$

$$\min MS \quad (28)$$

4. New multiple time grid, continuous time formulation (F3)

The use of multiple time grids instead of a single time grid allows us to consider fewer time points, and has been shown to be a very good option for single stage problems where the parallel units are totally independent[14]. For multiple stages, the modelling task is not so easy because only units belonging to the same stage do not interact. Equipment units belonging to consecutive stages that process the same order, have the constraint that the machine belonging to stage k+1 can only start after

the one belonging to stage k has ended. The difficulty arises from the fact that unlike models that use sequencing variables (see Harjunkoski and Grossmann[5]) the starting/ending time of an order on a given stage is not known explicitly as a model variable. In the multiple time grid formulation, that information comes from both the binary variable that identifies the execution of order i on machine m at time point t, $N_{i,m,t}$, and the time corresponding to that event point, $T_{t,m}$. Thus, to model the transfer of material between consecutive stages, we need to consider for each order all possible combinations between not only pairs of units but also pairs of time points, which may lead to a very large number of constraints. The number of constraints can be reduced if a new set of variables is added to the formulation, $TD_{i,k}$, which represent the transfer time of order i on stage k. This approach led to much better computational performances and is the only one described next.

Figure 4 gives an overview of how the formulation works. For simplicity, it considers only one unit per stage, three orders, three stages and four time points (the number of required time points on each machine is equal to the number of orders assigned to that machine plus one). Note that each order only lasts one time interval, but this does not necessarily mean that if the order starts at time point t that it ends at point t+1. While this is true for orders I2 and I3 in the first stage (unit M1), order I1 ends at time $T_{1,1}+p_{1,1}$, which is located before $T_{2,1}$. In the first stage, the time corresponding to the start point of the task must not be lower 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 not be lower than the ending time of the order at stage k, and must not be greater than the starting time of the order at stage k+1. In Figure 4, the transfer time of order I1 in stage 2 ($TD_{1,2}$) can be any value between $T_{1,2}+p_{1,2}$ and $T_{2,3}$ and this is shown as a grey filled rectangle. In contrast, the transfer time of order I2 in stage 2 ($TD_{2,2}$) is exactly equal to both the ending time of the order in that stage, $T_{2,2}+p_{2,2}$, and to the starting time of the order in the following stage, $T_{1,3}$. Notice also that order sequencing can vary from one stage to the other as seen for unit M2, which features I1-I2-I3, and unit M3, which has an I2-I1-I3 sequence.

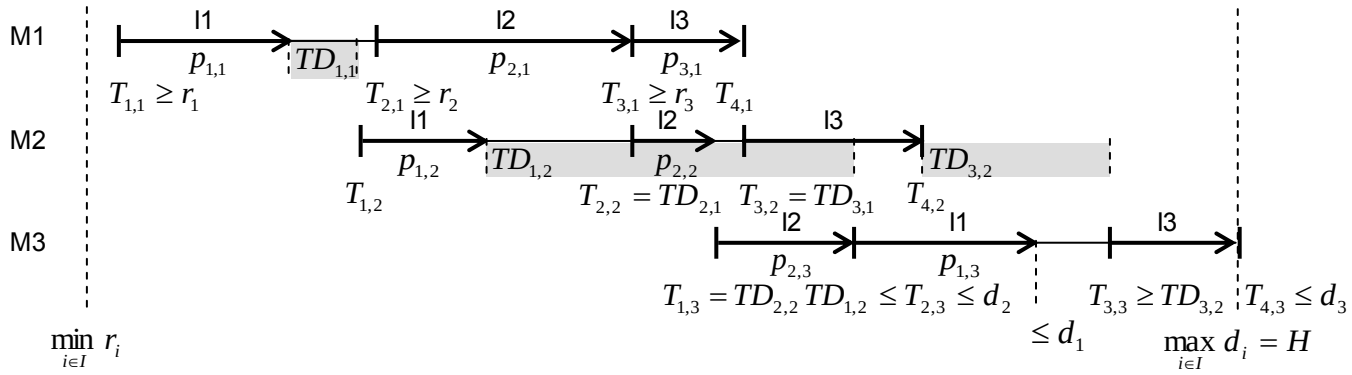

Figure 4. Possible solution from multiple time grid continuous-time formulation (|I|=3, |M|=3, |T|=4, |K|=3)

It is worthwhile mentioning that the proposed multiple time grid continuous-time formulation has one important conceptual difference from the one of Janak et al.[12]. In particular, there is no relation between time points belonging to different time grids, so the problem can be solved to global optimality by using the minimum number of time points possible. In contrast, the formulation of Janak et al.[12] requires at least one more time point per stage, due to their sequencing constraints relating different tasks in different units and also the material balances. For instance, the schedule shown in Figure 5 requires: 2 time points for each unit by the proposed formulation, 4 time points per unit by that of Janak et al.[12] (represented as n1-n4) and 7 time points total by the uniform-time grid formulation presented in section 3.2. Of course we cannot extrapolate these observations to actual computational performance because in each case the nature of the model is different.

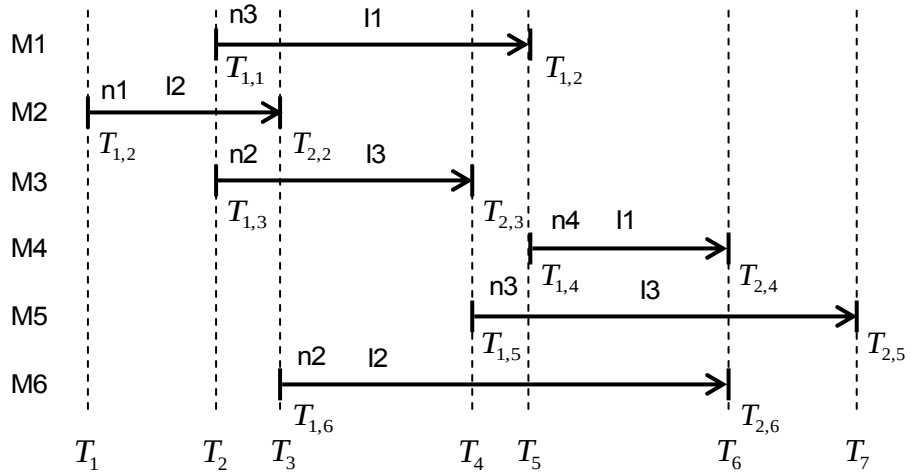

Figure 5. Number of event points required by different approaches (|I|=3, |M|=6, |K|=2)

The constraints of the multiple time grid continuous-time formulation are given next, together with a detailed explanation.

$$R_{m,t} = (1\big|_{t=1} + R_{m,t-1}\big|_{t\neq 1}) - \sum_{i\in I_m} N_{i,m,t} + \sum_{i\in I_m} N_{i,m,t-1} \;\forall m \in M, t \in T \tag{29}$$

$$\sum_{m\in M_i \wedge m\in M_k} \sum_{t\in T} N_{i,m,t} = 1 \;\forall i \in I, k \in K \tag{30}$$

Eq 29 is a multiperiod balance on the unit availability, where $R_{m,t}$=1 if the machine is not used at time interval *t*. Any task starting on unit *m* at time point *t* decreases the resource availability when compared to the previous time point, while any task starting on machine *m* at time point *t*-1 increases the resource availability at time point *t*. Due to the initial resource availability of one (first term on the right hand side), only one order can be executed at a given time point. Eq 30 ensures that all orders are processed exactly once on each stage.

$$T_{t+1,m} - T_{t,m} \geq \sum_{i\in I_m} N_{i,m,t} p_{i,m} \;\forall m \in M, t \in T, t \neq |T| \tag{31}$$

Eq 31 states that if order *i* is processed on unit *m* at time *t*, then the difference in time between time points *t*+1 and *t* on unit *m* must not be lower than the processing time of the task, $p_{i,m}$. Note that if no order is processed at time point *t*, the constraint is relaxed to $T_{t+1,m}$-$T_{t,m}$≥0.

$$T_{t,m} \geq \sum_{i\in I_m} N_{i,m,t} (r_i + \sum_{\substack{k'\in K \\ k'<k}} \min_{m'\in M_{k'} \atop m'\in M_i} p_{i,m'}) \;\forall k \in K, m \in M_k, t \in T, t \neq |T| \tag{32}$$

$$T_{t,m} \leq \sum_{i\in I_m} N_{i,m,t} (d_i - p_{i,m} - \sum_{\substack{k'\in K \\ k'>k}} \min_{m'\in M_{k'} \atop m'\in M_i} p_{i,m'}) + H(1 - \sum_{i\in I_m} N_{i,m,t}) \;\forall k \in K, m \in M_k, t \in T, t \neq |T| \tag{33}$$

Eq 32 is the release date constraint. It ensures that if a given order *i* is executed on unit *m* at time *t*, then the time corresponding to time point *t* on unit *m*, $T_{t,m}$, must not be lower than the order release date, if the unit belongs to the first stage (M_1). For units belonging to subsequent stages, e.g. stage k, appropriate lower bounds are given by adding the minimum processing times on each of the previous stages k'|k'<k. Eq 33 is the equivalent due date constraint. It ensures that if order *i* starts on unit *m* at time point *t*, then the corresponding time $T_{t,m}$ does not exceed the difference between the order's due date (d_i) and its processing time on that unit ($p_{i,m}$), if the machine belongs to the last stage ($M_{|K|}$). For

units belonging to previous stages, appropriate upper bounds are given by subtracting the minimum processing times on each of the subsequent stages. Note that if no order starts at that time point, the constraint is relaxed to its upper bound, i.e. eq 43.

Up to this point the constraints proposed have little changes when compared to the single stage multiple time grid continuous-time formulation given in Castro and Grossmann[14]. The new constraints for the multiple stage case are given in eqs 34 to 40.

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

$$TD_{i,k} \ge T_{t,m} + N_{i,m,t}\, p_{i,m} - H(1-\sum_{\substack{t'\in T\\ t'\ge t\\ t'\ne |T|}} N_{i,m,t'}) \;\forall k \in K, m \in M_k, i \in I_m, t \in T, k \ne |K|, t \ne |T| \tag{35}$$

Eq 34 states that the transfer time of order *i* in stage *k*-1 cannot be higher than the time corresponding to time point *t*, if order *i* starts to be processed on unit *m* (belonging to stage *k*) at that time point or at a previous one. Eq 35 ensures that the transfer time of order *i* on stage *k* is not lower than the ending time of order *i* on that stage. The constraint is active only if order *i* starts on unit *m* at time point *t* or at a subsequent time point.

$$TD_{i,k} \ge r_i + \sum_{\substack{t\in T\\ t\ne |T|}} \sum_{\substack{m\in M_{k'}\\ m\in M_i}} \sum_{\substack{k'\in K\\ k'\le k}} N_{i,m,t}\, p_{i,m} \;\forall i \in I, k \in K, k \ne |K| \tag{36}$$

$$TD_{i,k} \le d_i - \sum_{\substack{t\in T\\ t\ne |T|}} \sum_{\substack{m\in M_{k'}\\ m\in M_i}} \sum_{\substack{k'\in K\\ k'>k}} N_{i,m,t}\, p_{i,m} \;\forall i \in I, k \in K, k \ne |K| \tag{37}$$

$$TD_{i,k} \ge r_i + \sum_{\substack{t\in T\\ t\ne |T|}} \sum_{\substack{m\in M_k\\ m\in M_i}} N_{i,m,t}\, p_{i,m} \;\forall i \in I, k \in K, k = 1 \tag{38}$$

$$TD_{i,k} \le d_i - \sum_{\substack{t\in T\\ t\ne |T|}} \sum_{\substack{m\in M_{k+1}\\ m\in M_i}} N_{i,m,t}\, p_{i,m} \;\forall i \in I, k \in K, k = |K| - 1 \tag{39}$$

$$TD_{i,k} \ge TD_{i,k-1} - \sum_{\substack{t\in T\\ t\ne |T|}} \sum_{\substack{m\in M_k\\ m\in M_i}} N_{i,m,t}\, p_{i,m} \;\forall i \in I, k \in K, k \ne 1, k \ne |K| \tag{40}$$

Eqs 36 and 37 are not required, but usually improve the performance of the formulation both for total cost minimization (eq 47) and makespan minimization (eq 28). They act as lower and upper bounds on

the transfer times and are conceptually equivalent to eqs 32 and 33. For the objective of total earliness minimization (eq 48) it was found that replacing eqs 36 and 37 by eqs 38-40 improved the performance. Note that eqs 36/37 simplify to eqs 38/39 when applied to the first stage and the stage before the last, respectively. When the plant consists of only two stages, eq 40 does not apply, otherwise it relates the transfer times of consecutive stages.

$$\sum_{\substack{t\in T\\ t\neq|T|}}(N_{i,m,t}+N_{i,m',t})\leq 1\ \forall i\in I, m\in M_i, m'\in M_i, (m,m')\in FP \tag{41}$$

$$T_{t,m}\geq \min_{i\in I} r_i\ \forall m\in M, t\in T \tag{42}$$

$$T_{t,m}\leq H=\max_{i\in I} d_i\ \forall m\in M, t\in T \tag{43}$$

Eq 41 avoids forbidden paths, while eqs 42 and 43 define the lower and upper levels of the timing variables $T_{t,m}$. Note that these are usually weaker than constraints 32 and 33.

$$MS\geq \sum_{\substack{t\in T\\ t=|T|}} T_{t,m}\ \forall m\in M_{|K|} \tag{44}$$

$$T_{t,m}\leq MS-\sum_{i\in I_m} N_{i,m,t}\,(p_{i,m}+\sum_{\substack{k'\in K\\ k'>k}}\min_{\substack{m'\in M_{k'}\\ m'\in M_i}} p_{i,m'})\ \forall k\in K, m\in M_k, t\in T, t\neq|T| \tag{45}$$

$$TD_{i,k}\leq MS-\sum_{\substack{t\in T\\ t\neq|T|}}\sum_{m\in M_{k'}}\sum_{\substack{k'\in K\\ k'>k}} N_{i,m,t}\,p_{i,m}\ \forall i\in I, k\in K, k\neq|K| \tag{46}$$

For the objective of makespan minimization a new variable is required. Eq 44 defines variable *MS* (makespan) as the maximum ending time (absolute time of last time point) of the time grids corresponding to machines belonging to the last stage. The performance of the mathematical formulation can be greatly improved with the addition of two more sets of constraints, eqs 45 and 46, which significantly lower the integrality gap. Although these are quite similar to eqs 33 and 37, they are tighter constraints whenever the makespan is lower than the order's due date. Note that in eq 45, the makespan is a valid upper bound on the timing variables even when no order starts at time point t, whereas in eq 33 the big-M term is required to account for that case.

$$\min \sum_{t\in T}\sum_{m\in M_i}\sum_{i\in I} N_{i,m,t}c_{i,m} \tag{47}$$

$$\min \sum_{i \in I} d_i - \sum_{\substack{t \in T \\ t \neq |T|}} \sum_{m \in M_{|K|}} (T_{t,m} + \sum_{i \in I_m} N_{i,m,t} p_{i,m}) - H[|I| - |M_{|K|}|(|T| - 1)] \quad (48)$$

The objective of total cost minimization (eq 47) and makespan minimization (eq 28) are easy to define. In contrast, the objective of total earliness minimization (eq 48) is difficult to understand unless we note that since every task lasts only one time interval, we know a priori how many time points (of machines belonging to the last stage) will end being equal to their upper bound, H. The last term on the right hand side of eq 48, subtracts the starting times, which are accounted for in variables $T_{t,m}$, of such time points.

In summary, the multiple time grid continuous-time formulation has constraints 29-35 and 41-43 as its main block, and a few objective-specific sets of constraints. These are eqs 36-37 and 47 for total cost minimization, eqs 38-40 and 48 for total earliness minimization and eqs 28, 36-37, 44-46 and 49 for makespan minimization.

5. Other approaches

Harjunkoski and Grossmann[5] presented MILP, CP and hybrid MILP/CP models to address the multistage batch scheduling problem for total cost minimization. In this section, we highlight their main features and present the changes required to efficiently address the other two objectives considered in this paper.

5.1. MILP model with global precedence sequencing variables (F4)

The MILP model of Harjunkoski and Grossmann[5] uses binary variables $y_{i,m}$ to assign order i to machine m, and binary sequencing variables $x_{i,i',k}$ that equal one when order i precedes (global precedence and not just immediate precedence) order i' in stage k. In addition, the model also uses operational variables z_m to identify the use of unit m. The objective function used was total cost minimization plus a term that accounted for the initialization cost of units. In this paper, we are only concerned with common scheduling objective functions so the MILP was simplified by removing variables z_m from the formulation. As a consequence, constraint 33 from their original work was

dropped. The other objective functions under study, total earliness minimization and makespan minimization, can be implemented as follows, where the nomenclature of their original paper is used except for the new makespan variable.

$$\min \sum_{i \in I} (T_i^d - \sum_{\substack{k \in K \\ k=|K|}} c_{ik}) \tag{49}$$

$$MS \geq \sum_{\substack{k \in K \\ k=|K|}} c_{ik} \ \forall i \in I \tag{50}$$

$$\begin{aligned} &\sum_{i \in I} y_{im} (T_{im}^p + T_m^s) \leq MS - \min_{i \in I} \{ \sum_{k' \in K, k' > k} \min_{m' \in M(k'), (i,m') \notin \hat{B}} (T_{im'}^p + T_{m'}^s) \} \\ &- \min_{i \in I} \{ T_i^r + \sum_{k' \in K, k' < k} \min_{m' \in M(k'), (i,m') \notin \hat{B}} (T_{im'}^p + T_{m'}^s) \} \ \forall k \in K, \forall m \in M(k) \end{aligned} \tag{51}$$

Eq 49 defines the objective of total earliness as the sum over all orders of the difference between the due date and the completion time of the order in the last stage. The makespan must not be lower than the completion time of order *i* in stage |K|, eq 50. For makespan minimization, it is also convenient to replace constraint 41 of their original work with eq 51, since a better upper bound on the time horizon is the makespan instead of the maximum due date.

5.2. Constraint programming model (F5)

The CP model of Harjunkoski and Grossmann[5], F5, is based on the OPL Studio modelling language[17] and it takes advantage of special constructs designed for solving scheduling problems. The two main components of scheduling models in OPL Studio are activities and resources. For the scheduling problem at hand, the activities correspond to the processing tasks (order *i*, stage *k*), while each available equipment unit is defined both as a unary resource and as an alternative resource from the full set of machines. It is important to mention that the variables linked to a particular activity (*start*, *duration* and *end*) are of type integer. Besides these special variables, the CP model uses binary variables $y_{i,m}$ to assign order *i* to unit *m*, like the MILP model F4. In CP models, unlike MILPs, max functions can be used to define the model constraints. Thus, if one uses Harjunkoski and Grossmann[5] nomenclature, the

objective of total earliness and makespan minimization can simply be written by eqs 52 and 53, respectively.

$$\min \sum_{i \in I} (T_i^d - Stage[i,|K|].end) \tag{52}$$

$$\min Z = \max_{i \in I} (Stage[i,|K|].end) \tag{53}$$

5.3. Hybrid MILP/CP model

Harjunkoski and Grossmann[5] also considered a hybrid MILP/CP model, where the scheduling problem is decomposed in two parts. The first concerns the assignment problem and is formulated as a MILP, a simplified version of the full MILP, one without the constraints involving the sequencing variables. Once the assignments are known, a CP optimization problem is solved to minimize the weighted sum of the total violation of the due dates and the number of late orders. If there are no late orders, the assignments and sequences are valid and the optimal solution has been found. Otherwise, integer cuts are added to the MILP to avoid generating the assignments already tested. In multistage problems, unlike in single stage problems (Jain and Grossmann[15]), valid cuts are rather weak and a large number of iterations can be expected before the optimal solution is found. An alternative is to use stronger heuristic cuts that my cut-off the true optimal solution. By doing this, the authors were able to devise an efficient hybrid algorithm, which for virtually all problems gave the correct optimal solution. However, since their method was devised for a case where the objective function depends solely on the assignment variables and not on the sequencing variables (it is expected that the CP subproblems become much more difficult to solve if the sequencing variables influence the objective function), the hybrid model of Harjunkoski and Grossmann[5] is not considered in this paper.

6. Computational results

In this section, the performance of the five different approaches is illustrated trough the solution of thirty 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 minimization, E for total earliness

and M for makespan. Problems P1-P8 correspond to the multistage example problems 2.1-5.2 of Harjunkoski and Grossmann (2002). The data for problems P9-P10 are given in Table 1 and Table 2.

The four MILP models were solved to optimality (1E-6 relative tolerance), unless otherwise stated, by the commercial solver GAMS/CPLEX 9.0, whereas the CP model was solved in ILOG's OPL Studio 3.7.1. The computer used was a Pentium-4 2.8 GHz processor running Windows XP Professional. The computational effort required to solve the several test problems for the three different objective functions is given in Table 3 through Table 5, while more detailed computational statistics are left for Table 6 through Table 17.

6.1. Minimization of total cost

As can be seen from Table 3 the discrete-time formulation (F1) is a reasonable performer overall and the best one for P10C. Its performance is greatly influenced by the number of time points that are required to represent the exact problem data (i.e. for δ=1). Problems P1C-P8C require a very large number of time points, while the number for P9C/P10C is more reasonable (see Table 6 to Table 9). Fewer time points means fewer variables and constraints so even if the number of orders, units and stages increases, it might be easier to solve the mathematical problem (e.g. P10C is a more complex problem than P5C but it is solved much faster since it requires 283 time points while P5C needs 1001, see Table 7 and Table 9). Despite the very large number of time points required for most of the problems, all but P7C-P8C (δ=5 in order to maintain tractability since no solution was found for δ=2 after 10,000 CPUs) could be solved to optimality, with the exact problem data, in less than half an hour. This is only possible because of the following. First, the resulting MILPs have very low, if not zero, integrality gaps. Furthermore, it is very easy for CPLEX to find the optimal solution, which is often the first feasible solution and is usually found on the root node of the branch and bound tree. Second, most of the tasks can only start at a subset of the time points ($T_{i,m}$), which depending on the order's release and due dates and on the number of stages, can result in significantly fewer binary variables $N_{i,m,t}$ than

those predicted by multiplying |I|×|M|×|T| (e.g. problem P5C requires 40341 binary variables as opposed to 10×6×2001=120060).

The constraint programming model (F5) is related to the discrete-time model in the sense that they both consider integer data only. However, contrary to the discrete-time formulation, CP does not divide the time horizon into several time intervals, meaning that the number of variables and constraints is independent on the relation between the processing times and the time span. As a consequence, the computational effort depends solely on the complexity of the problem and is clear from Table 3 that as the number of orders, units and stages increases so does the computational effort, with CP being unable to find the optimal solution for problem P9C and proving optimality for P10C. In this respect, notice that problems P7C and P8C use more variables and constraints than P9C, but take much less time to solve (see Table 8 and Table 9). Overall, F5 is only the best performer for P1C, but has the advantage of always finding relatively good solutions very fast. More computational time means better solutions although it has been found that the additional computational effort for further improvements in the objective function usually increases rapidly (e.g. the best solution for P9C, 94, was found after 936 CPUs and no further improvements were observed until the termination time of 20000 CPUs, with the solution being still relatively distant from the optimum of 88).

Concerning the continuous-time formulations, it is clear from Table 3 that solving multistage problems with a uniform time grid formulation (F2) is the worst option of the five considered in this paper. Solving a particular problem implies specifying both the total number of time points, |T|, and the number of time points allowed between the beginning and end of a particular task, Δt. Adequate values for both of these entities are not easy to predict and this is a very important issue since they greatly influence the quality of the solution returned and the computational effort. There is, however, an easy way to predict the minimum number of time points required to find a feasible solution to the problem. Assuming the same number of units per stage (a feature shared by all test problems), the following formula can be used: CEIL(|I|×|K|/|M|)+|K|. This formula predicts for problems P1C-P4C and P9C, 5 and 7 time points, respectively. Interestingly, the predicted minimum value is sufficient to find the

optimal solution for problems P3C-P4C (with $\Delta t=1$, see Table 6), whereas problems P1C-P2C require 6 and 7 time points (with $\Delta t=2$). A good solution (cost=91) can be found for problem P9C for $|T|=7$ and $\Delta t=1$ in just 28 CPUs, but no further improvements were observed for $|T|=8$ and $\Delta t=2$ after 20,000 CPUs. Clearly, the uniform time grid continuous-time formulation is only competitive for smaller problems with lowly constrained schedules (this means that the duration of the tasks can be extended past their processing time in order for them to end at the next limiting time point and hence reduce the total number of time points, further details can be found in Castro et al.[10]). As schedules become more constrained and the values of both $|T|$ and Δt required to find the optimal solution increase, so do the number of variables and constraints, the integrality gap and the computational effort, which when added to the uncertainty of finding adequate values for those two entities, means that not even a feasible solution may be found in a reasonable amount of time (e.g. problems P5C-P8C, P10C).

The proposed multiple time grid continuous-time formulation (F3) is much more efficient than its uniform time grid counterpart (F2). Recall, that using one time grid per machine ensures that all processing tasks only last one time interval (see section 4), which is the same as saying that Δt can be set to one without risk of compromising the optimal solution. However, we still have to search for the optimal solution over $|T|$. The predicted minimum number of time points to find a feasible solution to the problem is calculated slightly different from the formula used for the uniform time grid since now all processing tasks can start at the first time point and not just the ones belonging to the first stage. Accordingly, all tasks can end at the last time point and not just the ones belonging to the last stage. The formula to use is the following: $\mathrm{CEIL}(\max_{k\in K}(|I|/|M_k|))+1$. Usually, the value of $|T|$ that ensures optimality is very close to the predicted minimum number. However, special plant characteristics, like forbidden paths may increase the gap, e.g. for problems P7C and P8C the predicted minimum number is equal to 7, while 13 time points are required to find the global optimal solution. Overall, the proposed formulation has very good overall performance and is the best by far for problem P7C. However, it was unable to find the optimal solution for problem P9C (cost of 89 vs. 88) in 20,000 CPUs, even though the

integrality gap was reduced to zero (when comparing to the optimal solution) earlier in the search. A different search strategy other than default may reduce dramatically the computational time, but this is beyond the scope of this paper.

Of the three continuous-time formulations tested, the MILP model with sequencing variables of Harjunkoski & Grossmann[5], F4, has the best overall performance. When compared to the proposed formulation, F3, it has the advantage of not being dependent on a priori model parameters (i.e. specifying |T|) that may compromise optimality. Furthermore, it requires fewer variables and constraints to model a particular problem. However, even though it was the best performer for P2C-P6C (see Table 3) and P8C-P9C, much better performances (by over one order of magnitude) were achieved by the proposed formulation for P7C and by the discrete-time formulation for P10C.

The optimal solution for problem P10C, with cost=154, is given in Figure 6. The machines have plenty of waiting periods but this was expected since the times at which the orders begin to be processed do not influence the objective function except for the fact that order's release and due dates must be respected. Of the total 40 assignments only 5 are not low cost assignments. Although this solution features the same number of orders per machine, this objective function typically leads to schedules with more orders being assigned to the most favourable machines (e.g. P3C and P5C-P6C) if of course enough time is available to process them.

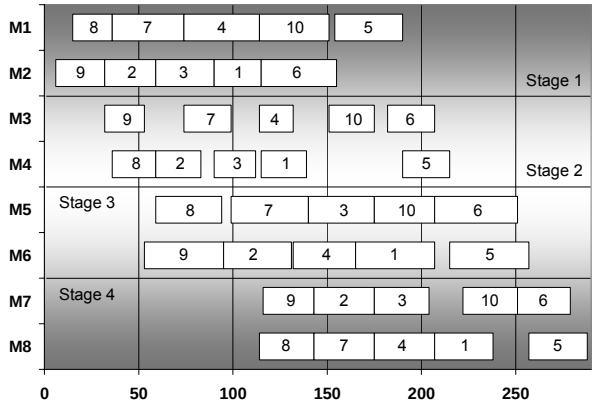

Figure 6. Optimal solution for problem P10C

6.2. Minimization of total earliness

While the objective of total cost minimization is only dependent on the assignments of orders to units, the objective of total earliness minimization also depends on the starting time of the orders on the last stage. This means that more variables are contributing for the objective function and that the degree of degeneracy is lower, which typically leads to a decrease in the computational effort. However, other factors may lead to an increase in the computational effort, e.g. higher integrality gap, which may be more important than the lower degree of degeneracy. Generally, it was found that the objective of total earliness minimization is more difficult to handle (see Table 10 to Table 13). The exception is the discrete-time formulation, and this is the reason why this objective is preferred in the algorithm for makespan minimization (see section 3.1.1).

The results in Table 4 show that the discrete-time formulation is the best approach for total earliness minimization. It was possible to use the real data for 8 of the 10 test problems and hence find the global optimal solutions even when using a very large number of time points (e.g. problem P6E requires 2001 time points). The two exceptions were problems P7E and P8E, mainly because of the large integrality gap (see Table 12) due to the several forbidden path constraints. Nevertheless, we can still solve P7E rather fast with $\delta=2$ (whereas no solution was found for P7C after 10,000 CPUs) while P8E was solved with $\delta=5$ (no solution was found after 20,000 CPUs for $\delta=2$). If the problem data allows for the use of a few hundred time intervals, it is expected that a further increase in complexity of the scheduling problem can still be handled by the discrete-time formulation.

The constraint programming formulation (F5) has also very good performance for the P1E-P8E. For all of these problems the optimal solution could be found in less than 30 CPUs. However, for P7E-P8E optimality could not be proved up to 20,000 CPUs and 55,000 CPUs, which does not favour the performance of CP. Furthermore, the best solution found for P9E in 50,000 CPUs is still very distant from the optimal solution (547 vs. 228), while for P10E, the best solution up to 45,000 CPUs is also non-optimal (189 vs. 184). Overall, the objective of total earliness minimization is the most difficult to

tackle by the constraint programming model. This is not surprising since the objective function involves many variables (see Hooker[18]).

The general continuous-time formulation (F2), like for total cost minimization, can only solve the four simplest instances and P9E, but now it does it with considerable more effort. This can be partially explained by an increase in the number of constraints (constraints 21-23 need to be added to the formulation, see section 3.2), but more importantly by an increase in the number of time points and the value of Δt that are required to find the global optimal solution. Notice for instance that while $|T|=5$ and $\Delta t=1$ are enough for P4C, $|T|=9$ and $\Delta t=2$ are not enough for P4E. As a consequence, the global optimal solution could only be found for the two simplest instances P1E/P2E and for the latter, at 20,000 CPUs the solver was still far from proving optimality. The best solutions found for P3E and P4E, 137, are very close to the global optimal solutions of 135, whereas due to the use of the minimum number of time points that ensures feasibility, the solution found for P9E (649) is very distant from the optimal one, 228.

While the number of time points increases for the uniform time grid formulation (F2), it usually remains the same or even decreases for the multiple time grid formulation (F3), when switching from total cost minimization to total earliness. Since in F3, the number of time points required is equal to the maximum number of orders assigned to a unit plus one, this tells us that there usually is at least one unit with more orders assigned to for total cost minimization, which is natural since the unit in question will typically be a low cost one. On the other hand, for total earliness minimization, time also matters, so there will be a more balanced distribution of orders through the available machines. The results show that the minimum number of time points that ensures feasibility (see formula in section 6.1) usually also ensures optimality (e.g. P1E-P6E, P10E). The results in Table 4 also show that the proposed formulation is very efficient for total earliness minimization, and that it is the best performer for problems P1E/P3E and P9E (2 stage problems). However, it is clear that its performance degrades as the number of stages increases with no solution being found for P10E (4 stage problem) up to 20,000 CPUs, even though 6 time points are enough to find a global optimal solution (see Figure 7). This can be

explained by noting that for each additional stage, more transfer times need to be determined by the model (variables $TD_{i,k}$) and that these are related to the times of the several time points by very complex constraints (eqs 34-35, 38-40). Of the three continuous-time formulations, F3 usually has the lowest integrality gap.

The continuous-time formulation with sequencing variables (F4) performs again very well and is the best for problems P5E-P8E. Now, a substantially decrease in performance seems to occur with an increase in the number of orders since the optimal solution for P9E (15-order problem) could not be found up to 18,000 CPUs (CPLEX ran out of memory with best solution found, 244, still distant from the optimum of 228). This can be explained by noting that F4 uses sequencing variables ($x_{i,i',k}$) with two indices for orders, so the number of sequencing variables per stage increases significantly making the problem much more difficult to solve even if the total number of variables and constraints remains basically the same when compared to P7E-P8E (see Table 12 and Table 13). It is also very difficult to prove optimality for P10E, with the solution for the relaxed MILP being still at 172 (optimum is 184) after 20,000 CPUs. Interestingly, the optimal solution is found in few minutes and the integrality gap remains constant after the hour.

The schedule shown in Figure 7 is an optimal solution for problem P10E (earliness=184). It can be seen that are several waiting periods, some lasting quite a while, particularly in machines belonging to the intermediate stages. Although some can be explained by the machine's need to wait for the material coming from the previous stage, it is clear that there is a lot of free time, meaning that this problem is highly degenerate but not as much as its total cost counterpart. Of the ten orders that make the problem, only three (I1, I5 and I6) are exactly delivered at their due dates. It is also interesting to note that the order completion sequence (I3-I4-I2-I7-I9-I8-I1-I10-I6-I5) is the same for the last 5 orders as that given by the earliest due date (EDD) heuristic, although the same is not true for the first 5 orders (I7-I9-I4-I2-I3).

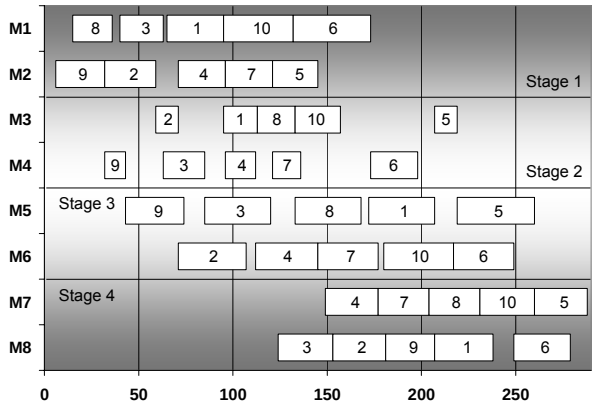

Figure 7. Optimal solution for problem P10E

6.3. Minimization of makespan

The objective of makespan minimization is the hardest of the three studied in this paper for the discrete-time formulation (F1) and for the continuous-time formulation with sequencing variables (F4), but is also the easiest one for the constraint programming model, F5 (see Table 5 and Table 14 to Table 17). In terms of solution degeneracy, this objective lies between that of total cost minimization and total earliness minimization since the assignments and starting times of all processing orders belonging to the limiting machine(s) and/or paths (transfer of material between consecutive stages) will contribute to the makespan.

The fact that the hardest objective for the mathematical programming approaches is the easiest objective for constraint programming is in agreement with observation by Hooker[18], who has highlighted their complementary strengths. To explain the good performance of CP for makespan minimization there is need to recall that the optimization process is performed by gradually tightening a bound on the objective function value. That is, the objective is treated as an additional constraint in the search. Unlike for the other two objectives, updating the objective function in makespan minimization leads to the tightening of the time horizon, which has a direct influence on the domain reduction of the model variables. Since this is one of the methods (together with constraint propagation) used by CP solvers to accelerate the search, the reason for the good performance is now clear. Generally, CP is the

best approach for makespan minimization although it is outperformed by the proposed formulation for problems P6M-P8M and by the discrete-time formulation for P10M.

Despite the need of a special algorithm to tackle this objective with a discrete-time formulation, we could still solve most of the problems (P1M-P4M, P9M-P10M) for the exact problem data (δ=1) and for five of them (the exception is P9M, see Table 17), this was done in less than one hour of computational time (note however that the reported CPU times do not include the generation time, which for the largest problems, in terms of variables and constraints, may be equal to one or two minutes per iteration). Like for the other two objectives, P7M and P8M are only tractable for larger interval lengths, which now are higher than before (δ=10). Nevertheless, the optimal solutions are close enough to the global optimal solutions (1470 vs. 1456 and 1660 vs. 1655, see Table 16); for P5M and P6M, δ=2, the difference was even smaller (1442 vs. 1435 and 1398 vs. 1394, see Table 15). With the exception of P7M and P8M, which have several forbidden paths, and P2M, the predicted lower bound for the cardinality of $|T|$ is within 10% of the required value that ensures feasibility in terms of earliness minimization and hence optimality in terms of makespan minimization. This ensures that the required number of iterations remains reasonable, thus guarantying the efficiency of the proposed approach. More specifically, it is the best approach for P10M, despite requiring 21 iterations to find the optimal solution.

The uniform time grid, continuous-time formulation (F2), has again the worst performance of the five. However, it could still find the global optimal solutions for problems P1M-P4M, even though optimality could not be proved for P4M, and a reasonable solution for P9M (makespan=271 vs. 235). The values of $|T|$ and Δt that ensure global optimality were found to be greater or equal to those required for total cost minimization and less or equal to those required for earliness minimization. Again this was expected, since tasks performed on non-limiting machines (at a given point in time) can have their duration extended in order to decrease the total number of time points required, while those that belong to the limiting path cannot, since they directly affect the makespan.

The proposed formulation (F3) requires the exact same number of time points to find the global optimal solution in makespan minimization as in total earliness minimization. Despite not being able to prove optimality for P9M and finding the optimal makespan for P10M (259 vs. 252) it has a very good overall performance being second only to the constraint programming model. The continuous-time formulation with sequencing variables (F4) could not prove optimality for P7M up to 20,000 CPUs and returned suboptimal solutions for problems P8M-P10M, the last one being terminated because the solver ran out of memory.

The optimal solution for P9M (the hardest problem), with a makespan of 235, is given in Figure 8. It is a very constrained schedule in terms of equipment utilization since there are only two waiting periods (1 time unit in M1 between I14 and I9, and 4 time units in M6 between I11 and I9). The makespan is set by M5 (235) although M4 and M6 end only a bit earlier, at 233. The first three orders to be processed in stage 1 (I10, I11 and I14) are those with the lowest release dates, which is consistent with the fact that the equipments need to start processing as soon as possible in order to minimize the makespan. After the transfer of I10 from stage 1 to stage 2 (M3 to M5), M5 becomes the limiting equipment even though it receives orders I15 and I12 just before the equipment becomes available (95 vs. 98 and 120 vs. 121, respectively). Note also that all orders are delivered well before their due dates.

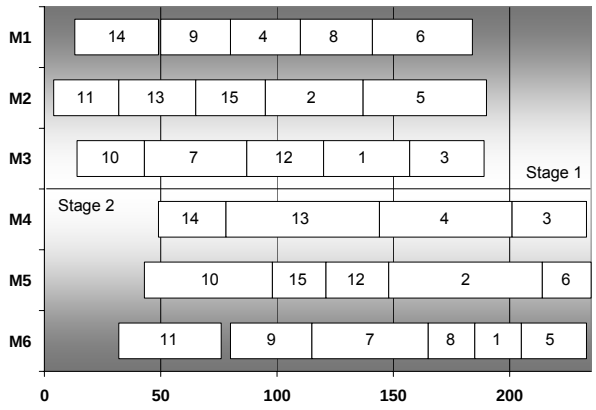

Figure 8. Optimal solution for problem P9M

7. Conclusions

This paper has presented a new continuous-time formulation for the short-term scheduling of multistage multiproduct batch plants where product orders are subject to both release and due dates. It is an extension of the formulation developed for the single stage case[14] since the formulation also relies on the use of multiple time grids, one for each equipment resource. While for the single stage case each machine is independent from the other, for multiple stages this is not true since any order can only start to be processed on a given stage after its processing has been completed in the previous stage. To model this plant characteristic, a new set of variables (the transfer times of the orders on the several stages) and new sets of constraints were developed. The proposed formulation was shown to be very efficient for the three different objective functions considered in this paper, minimization of total cost, total earliness and makespan.

The other goal of the paper has been to provide a critical review of other approaches that can also solve this specific type of scheduling problem. These included a RTN-based discrete-time formulation[8], a RTN-based continuous-time formulation[10], a continuous-time model with global precedence sequencing variables[5] and a constraint programming model[5]. A total of 30 examples were solved and the results allowed us to arrive at the following conclusions.

The most important conclusion is that there is not an ideal approach for all types of problems and objective functions. Besides the uniform-time grid continuous-time formulation, which is clearly inferior, all other four approaches have its strengths and drawbacks. Like for the single stage case[14], the discrete-time formulation was shown to have a very good performance, particularly for total earliness minimization, despite generating very large MILPs when considering the exact problem data. The number of time points used to model the scheduling problem plays a role at least as important as its complexity in terms of number of orders, machines and stages. For makespan minimization with a discrete-time formulation, an indirect approach must be used. This paper presents an efficient algorithm that accomplishes this goal. It is based on previous work[16] but uses a different method to estimate the lower bound on the makespan and a different objective function (earliness minimization) in the search

for the optimal makespan. For all test problems not featuring forbidden paths, the predicted lower bound was within 10% of the optimal makespan.

Smaller to medium sized problems requiring a very large number of time points by the discrete-time formulation, are more efficiently solved by continuous-time formulations. The performance of the proposed multiple-time grid continuous-time formulation clearly shows that is better to use several time grids instead of a uniform time grid. It should be noted, however, that in these problems the interaction between the different units was limited to the material transfer between different stages. Other features such as common resources and/or shared intermediate storage are probably better handled by uniform time grid approaches. The proposed formulation outperformed the continuous-time formulation with sequencing variables for makespan minimization, and performed similarly for the other two objectives. In terms of problem characteristics, the results suggest that the former should be preferred as the number of orders increases and the number of stages decreases. Finally, the constraint programming approach has the disadvantage of considering integer data only, like the discrete-time formulation, and also shows a rapidly decrease in performance with an increase in problem complexity. However, it was found to be the best approach for the objective of makespan minimization.

Acknowledgments

The authors gratefully acknowledge financial support from Fundação Calouste Gulbenkian and from the Centre for Advanced Process Decision-making at Carnegie Mellon.

Nomenclature

Sets/Indices

FP=forbidden paths, pairs of machines (*m*,*m'*) where orders processed in *m* cannot proceed to *m'*
I/*i*, *i'*= process orders
I_m= orders to be processed on machine *m*
$I_{m,t,t'}$=orders than can be processed on machine *m* starting at time point *t* and ending at *t'*
$I_{t,m}$= orders that can start at time point *t* on machine *m*
K/*k*=process stages
KL=predicted limiting stage for makespan minimization using discrete-time formulation
K_m=stages including machine *m*

M/m= process equipments (machines)
M_i=machines that can process order i
$M_{i,t}$=machines that can start processing order i at time point t
M_k=machines belonging to stage k
$M_{t,t'}$=machines that can be active between time points t and t'
$T/t, t', t''$=Points of the time grid
$T_{i,m}$=time points where order i can start to be processed on machine m
$T_{i,m,t'}$=time points where order i can start to be processed on machine m in order to end at time point t'

Parameters
$c_{i,m}$=cost of processing order i on machine m
d_i=due date of order i
$\bar{d}_i$ =normalised due date of order i (for discrete-time formulation)
eft_i=earliest ending time of order i (time intervals)
H=time horizon
mct_k=predicted completion time in stage k (time intervals)
$p_{i,m}$=processing time of order i on machine m
r_i=release date of order i
$\bar{r}_i$ =normalised release date of order i (for discrete-time formulation)
δ=duration of each time interval on the discrete-time grid
Δt=number of event points allowed between the beginning and end of a processing task
$\tau_{i,m}$=processing time of order i on machine m as an integer multiple of δ

Variables
DD_i=delivery date of order i
MS=makespan
$N_{i,m,t}$=binary variable that assigns the start of order i on machine m to time point t
$\overline{N}_{i,m,t,t'}$=binary variable that assigns the end of order i, processed on machine m, which began at t, to event 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_t=absolute time of event point t
$T_{t,m}$=absolute time of event point t on machine m

References

(1) Pinto, J.; Grossmann, I.E. Assignment and sequencing models for the scheduling of chemical processes. *Annals of Operations Research*. **1998**, *81*, 433-466.

(2) Raman, R.; Grossmann, I.E. Modeling and computational techniques for logic based integer programming. *Comp. Chem. Eng.* **1994**, *18*, 563.

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

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

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

(7) 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.

(8) 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.

(9) 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.

(10) 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.

(11) 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.

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

(14) Castro, P.; Grossmann, I.E. An Efficient MILP Model for the Short-term Scheduling of Single Stage Batch Plants. Submitted to Comp. Chem. Eng.

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

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

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

(18) Hooker, J.N. (2002). Logic, optimization and constraint programming. *INFORMS Journal on Computing.* **2002**, *14*, 295-321

List of Tables

Table 1. Data for problem P9

	Dates		$p_{i,m}/c_{i,m}$					
Order	r_i	d_i	*M1*	*M2*	*M3*	*M4*	*M5*	*M6*
I1	87	377	37/4	30/5	37/4	70/3	63/4	20/8
I2	55	396	51/2	42/3	40/3	70/3	66/3	57/4
I3	112	263	48/2	51/2	32/4	32/7	25/8	63/4
I4	28	375	30/5	57/1	56/1	57/4	58/4	63/4
I5	79	281	32/4	53/2	42/3	59/4	38/6	28/7
I6	24	256	43/3	50/2	58/1	68/3	21/8	33/7
I7	15	400	48/2	38/4	44/3	65/4	70/3	50/5
I8	108	285	31/5	44/3	52/2	56/4	53/5	20/8
I9	50	300	30/5	43/3	58/1	46/5	38/6	35/6
I10	14	246	33/4	51/2	29/5	61/4	55/4	54/5
I11	4	384	56/1	28/5	40/3	46/5	67/3	44/6
I12	20	271	45/3	47/3	33/4	60/4	27/7	43/6
I13	21	384	58/1	33/4	31/5	66/3	61/4	56/4
I14	13	283	36/4	42/3	34/4	29/7	34/7	45/5
I15	58	399	46/3	30/5	52/2	62/4	23/8	44/6

Table 2. Data for problem P10

	Dates		$p_{i,m}/c_{i,m}$							
Order	r_i	d_i	*M1*	*M2*	*M3*	*M4*	*M5*	*M6*	*M7*	*M8*
I1	65	238	30/3	25/4	18/5	24/2	35/6	42/4	29/5	31/4
I2	22	227	26/4	27/4	12/7	24/2	31/8	36/6	32/4	28/5
I3	40	230	23/5	31/3	10/8	22/3	35/6	40/5	29/5	29/5
I4	71	223	40/2	25/4	18/5	16/5	45/3	33/7	28/5	32/4
I5	13	288	36/2	24/4	12/7	25/2	41/4	42/4	28/5	31/4
I6	59	279	41/2	40/2	25/2	25/2	44/3	32/7	28/5	30/5
I7	26	208	38/2	25/4	25/2	15/6	41/4	32/7	27/5	32/4
I8	15	235	21/5	21/5	20/4	23/3	35/6	44/3	27/5	29/5
I9	6	209	30/3	26/4	21/4	11/8	31/8	42/4	27/6	26/6
I10	21	265	37/2	30/3	24/2	20/4	32/7	37/6	29/5	30/5

Table 3. Overview of computational performance (CPU s) for total cost minimization

Type of Model	Discrete-time MILP	Continuous-time MILP			CP
Problem/Model	F1	F2	F3	F4	F5
P1C (6 orders, 4 machines, 2 stages)	4.62	3.18	0.09	0.08	0.06
P2C (6 orders, 4 machines, 2 stages)	10.7	0.12	0.06	0.05	0.11
P3C (8 orders, 6 machines, 2 stages)	6.82	0.07	0.19	0.07	31.3
P4C (8 orders, 6 machines, 2 stages)	3.62	0.04	0.12	0.04	61.5
P5C (10 orders, 6 machines, 3 stages)	1429	-	2.58	0.07	165
P6C (10 orders, 6 machines, 3 stages)	111	-	7.93	0.11	95.9
P7C (12 orders, 8 machines, 3 stages)	1814$^{\blacklozenge}$	-	104	3803	349
P8C (12 orders, 8 machines, 3 stages)	2151$^{\blacklozenge}$	-	12.2	7.13	224
P9C (15 orders, 6 machines, 2 stages)	1667	20000†,*	20000†,*	21.8	20000†,*
P10C (10 orders, 8 machines, 4 stages)	123	-	2262	4070	60000†

†Maximum resource limit
*Suboptimal solution returned
$^{\blacklozenge}$Approximated solution

Table 4. Overview of computational performance (CPU s) for total earliness minimization

Type of Model	Discrete-time MILP	Continuous-time MILP			CP
Problem/Model	F1	F2	F3	F4	F5
P1E (6 orders, 4 machines, 2 stages)	1.19	2488	0.1	0.74	4.71
P2E (6 orders, 4 machines, 2 stages)	1.09	20000†	0.11	22.4	7.08
P3E (8 orders, 6 machines, 2 stages)	4.61	20000†,*	4.23	4.38	8.37
P4E (8 orders, 6 machines, 2 stages)	3.32	20000†,*	4.39	3.46	8.89
P5E (10 orders, 6 machines, 3 stages)	749	-	29.9	0.27	6.94
P6E (10 orders, 6 machines, 3 stages)	15.8	-	56.5	3.83	8.59
P7E (12 orders, 8 machines, 3 stages)	78.6$^{\blacklozenge}$	-	24.4	1.29	20000†
P8E (12 orders, 8 machines, 3 stages)	9.86$^{\blacklozenge}$	-	17.5	1.68	55000†
P9E (15 orders, 6 machines, 2 stages)	25.6	20000†,*	7.52	18000†,*	50000†,*
P10E (10 orders, 8 machines, 4 stages)	234	-	20000†,‡	20000†	45000†,*

†Maximum resource limit
‡No solution found
*Suboptimal solution returned
$^{\blacklozenge}$Approximated solution

Table 5. Overview of computational performance (CPU s) for makespan minimization

Type of Model	Discrete-time MILP	Continuous-time MILP			CP
Problem/Model	F1	F2	F3	F4	F5
P1M (6 orders, 4 machines, 2 stages)	12.7	46.7	0.19	0.17	0.02
P2M (6 orders, 4 machines, 2 stages)	82.3	32.5	0.14	0.22	0.04
P3M (8 orders, 6 machines, 2 stages)	165	6214	0.82	0.51	0.39
P4M (8 orders, 6 machines, 2 stages)	80.9	20000†	0.56	0.62	0.31
P5M (10 orders, 6 machines, 3 stages)	414♦	-	14.1	1417	10.3
P6M (10 orders, 6 machines, 3 stages)	1265♦	-	1.96	5.22	8.57
P7M (12 orders, 8 machines, 3 stages)	53.1♦	-	22.0	20000†	39.5
P8M (12 orders, 8 machines, 3 stages)	1474♦	-	14.6	20000†,*	61.6
P9M (15 orders, 6 machines, 2 stages)	46000	20000†,*	20000†	20000†,*	9430
P10M (10 orders, 8 machines, 4 stages)	1482	-	20000†,*	8742◊,*	16679

†Maximum resource limit
*Suboptimal solution returned
♦Approximated solution
◊Solver ran out of memory

Table 6. Computational statistics for problems P3C-P4C

Problem	P3C					P4C				
Model	F1	F2	F3	F4	F5	F1	F2	F3	F4	F5
|T|	1501	5	4			1501	5	4		
Discrete variables	27383	156	165	103		22483	156	165	103	
Single variables	60406	272	198	120	96	55506	272	198	120	96
Constraints	33054	208	267	384	215	33054	208	267	384	215
RMIP	16	16	16	16		1063	1063	1063	1063	
Obj	16	16	16	16	16	1063	1063	1063	1063	1063
CPU	6.82	0.07	0.19	0.07	31.3	3.62	0.04	0.12	0.04	61.5
Nodes/Choice points	0	0	23	0	212386	0	0	8	0	369192

Table 7. Computational statistics for problems P5C-P6C

Problem	P5C				P6C			
Model	F1	F3	F4	F5	F1	F3	F4	F5
|T|	2001	7			2001	7		
Discrete variables	40341	390	193		30428	390	193	
Single variables	112378	453	224	150	102465	453	224	150
Constraints	72077	699	601	352	72077	699	601	352
RMIP	89	89	89		886	874	881	
Obj	89	89	89	89	886	886	886	886
CPU	1429	2.58	0.07	165	111	7.93	0.11	95.9
Nodes/Choice points	0	339	0	327618	0	2139	0	205004

Table 8. Computational statistics for problems P7C-P8C

Problem	P7C				P8C			
Model	F1	F3	F4	F5	F1	F3	F4	F5
|T|	461	13			391	13		
Discrete variables	11373	1232	292		13252	1232	292	
Single variables	31658	1361	329	204	30457	1361	329	204
Constraints	20367	1939	1163	552	17287	1939	1163	552
RMIP	60	60	60		659.5	664	664	
Obj	61[†]	61	61	61	664[†]	664	664	664
CPU	1814	104	3803	349	2151	12.2	7.13	224
Nodes/Choice points	0	7529	3106811	1058338	0	1166	3290	594546

[†]Optimal solution of approximated problem (δ=5)

Table 9. Computational statistics for problems P9C-P10C

Problem	P9C					P10C			
Model	F1	F2	F3	F4	F5	F1	F3	F4	F5
|T|	397	8	7			283	6		
Discrete variables	18141	990	582	300		8034	448	260	
Single variables	32442	1287	640	331	180	21619	527	301	200
Constraints	14323	469	751	1342	375	13625	869	839	530
RMIP	88	87.03	87.09	86.74		153.5	146.56	143.87	
Obj	88	91	89	88	94	154	154	154	154
CPU	1667	20000[†]	20000[‡]	21.8	20000[*]	123	2262	4070	60000[♦]
Nodes/Choice points	0	1.56E6	5.56E6	10000	3266306	0	134363	2408475	22421889

[†]Resource limit exceeded (best possible solution= 88, total tree size= 476 MB)
[‡]Resource limit exceeded (best possible solution= 88, total tree size= 72 MB)
[*]Resource limit exceeded (solution found at 936 CPUs)
[♦]Resource limit exceeded (solution found at 12500 CPUs)

Table 10. Computational statistics for problems P3E-P4E

Problem	P3E					P4E				
Model	F1	F2	F3	F4	F5	F1	F2	F3	F4	F5
|T|	1501	9	4			1501	9	4		
Discrete variables	27383	641	165	103		22483	641	165	103	
Single variables	60406	857	198	120	96	55506	857	198	120	96
Constraints	33054	596	267	384	215	33054	596	267	384	215
RMIP	135	0	0	0		135	0	0	0	
Obj	135	137	135	135	135	135	137	135	135	135
CPU	4.61	15708	4.23	4.38	8.37	3.32	20000[†]	4.39	3.46	8.89
Nodes/Choice points	0	2767883	5404	9879	6622	0	5.51E6	4254	6848	5844

[†]Resource limit exceeded (best possible solution= 49.5, total tree size= 254 MB)

Table 11. Computational statistics for problems P5E-P6E

Problem	P5E				P6E			
Model	F1	F3	F4	F5	F1	F3	F4	F5
|T|	2001	6			2001	6		
Discrete variables	40341	326	193		30428	326	193	
Single variables	112378	383	224	150	102465	383	224	150
Constraints	72077	587	601	352	72077	587	601	352
RMIP	35	0	0		69	0	0	
Obj	35	35	35	35	69	69	69	69
CPU	749	29.9	0.27	6.94	15.8	56.5	3.83	8.59
Nodes/Choice points	0	3567	60	9728	0	10681	4441	8799

Table 12. Computational statistics for problems P7E-P8E

Problem	P7E				P8E			
Model	F1	F3	F4	F5	F1	F3	F4	F5
|T|	1151	13			391	13		
Discrete variables	28270	1232	292		13252	1232	292	
Single variables	78915	1361	329	204	30457	1361	329	204
Constraints	50727	1927	1163	552	17287	1939	1163	552
RMIP	100	0	0		140	260	0	
Obj	700[†]	700	700	700	1380[*]	1380	1380	1380
CPU	78.6	24.4	1.29	20000[‡]	9.86	17.5	1.68	55000[♦]
Nodes/Choice points	0	573	375	6134	0	310	458	4128

[†]Optimal solution of approximated problem (δ=2)
[‡]Resource limit exceeded (solution found at 27.6 CPUs)
[*]Optimal solution of approximated problem (δ=5)
[♦]Resource limit exceeded (solution found at 21.0 CPUs)

Table 13. Computational statistics for problems P9E-P10E

Problem	P9E					P10E			
Model	F1	F2	F3	F4	F5	F1	F3	F4	F5
|T|	397	7	7			283	6		
Discrete variables	18141	450	582	300		8034	448	260	
Single variables	32442	725	640	331	180	21619	527	301	200
Constraints	14323	655	751	1342	375	13625	869	839	530
RMIP	228	0	64.23	0		184	154.17	0	
Obj	228	649	228	244	547	184	-	184	189
CPU	25.6	20000[†]	7.52	18000[‡]	50000[*]	234	20000[♦]	20000[◊]	45000[°]
Nodes/Choice points	0	3.91E5	1719	6.72E6	305141795	0	2.34E5	1.28E7	16113186

[†]Resource limit exceeded (best possible solution= 87.25, total tree size= 123 MB)
[‡]Solver ran out of memory (best possible solution= 50.8, total tree size= 1.1 GB)
[*]Resource limit exceeded (solution found at 34000 CPUs)
[♦]No integer solution found (best possible solution= 184)
[◊]Resource limit exceeded (best possible solution= 172, total tree size= 2.6 MB)
[°]Resource limit exceeded (solution found at 7144 CPUs)

Table 14. Computational statistics for problems P3M-P4M

Problem	P3M					P4M				
Model	F1	F2	F3	F4	F5	F1	F2	F3	F4	F5
|T|	794	7	4			794	7	4		
Discrete variables	19296	453	165	103		14396	453	165	103	
Single variables	36765	615	198	120	96	31865	615	198	120	96
Constraints	17500	348	265	391	215	17500	348	265	391	215
RMIP	-	320.02	687.88	732.82		-	331.13	688.62	732.82	
Obj	793	793	793	793	793	793	793	793	793	793
CPU	165	6214	0.82	0.51	0.39	80.9	20000[†]	0.56	0.62	0.31
Iterations/Nodes/ Choice points	66	3098702	644	417	1570	66	7.14E6	237	581	1522

[†]Resource limit exceeded (best possible solution= 773, total tree size= 158 MB)

Table 15. Computational statistics for problems P5M-P6M

Problem	P5M				P6M			
Model	F1	F3	F4	F5	F1	F3	F4	F5
|T|	722	6			700	6		
Discrete variables	16915	326	193		11557	326	193	
Single variables	42908	383	224	150	36758	383	225	150
Constraints	26033	648	610	352	25241	648	610	352
RMIP	-	1426.4	1324.0		-	1355	1234.4	
Obj	1442[†]	1435	1435	1435	1398[†]	1394	1394	1394
CPU	414	14.1	1417	10.3	1265	1.96	5.22	8.57
Iterations/Nodes/ Choice points	11	1937		29461	35	156	7146	23619

[†]Optimal solution of approximated problem (δ=2)

Table 16. Computational statistics for problems P7M-P8M

Problem	P7M				P8M			
Model	F1	F3	F4	F5	F1	F3	F4	F5
|T|	148	13			162	13		
Discrete variables	4322	1232	292		6210	1232	292	
Single variables	10835	1361	329	204	13339	1361	329	204
Constraints	6595	2060	1174	552	7211	2060	1174	552
RMIP	-	1456	1455		-	1615	1545	
Obj	1470[†]	1456	1456	1456	1660[†]	1655	1685	1655
CPU	53.1	22.0	20000[‡]	39.5	1474	14.6	20000[*]	61.6
Iterations/Nodes/ Choice points	50	1224	1.67E7	115964	66	284	1.19E7	188230

[†]Optimal solution of approximated problem (δ=10)
[‡]Resource limit exceeded (best possible solution= 1455, total tree size= 3.4 MB)
[*]Resource limit exceeded (best possible solution= 1545, total tree size= 19 MB)

Table 17. Computational statistics for problems P9M-P10M

Problem	P9M					P10M			
Model	F1	F2	F3	F4	F5	F1	F3	F4	F5
|T|	232	8	7			247	6		
Discrete variables	9899	990	582	300		7426	448	260	
Single variables	18252	1287	640	331	180	19283	527	301	200
Constraints	8383	511	804	1356	375	11897	940	848	530
RMIP	-	174.37	203.66	218.7	-	-	248.10	184	
Obj	235	271	235	236	235	252	259	254	252
CPU	46000	20000[†]	20000[‡]	20000[*]	9430	1482	20000[♦]	8742[◊]	16679
Nodes/Choice points	22	2.09E6	1.48E6	6.26E6	24127287	21	8.94E5	4.93E6	24557874

[†]Resource limit exceeded (best possible solution= 205.2, total tree size= 949 MB)
[‡]Resource limit exceeded (best possible solution= 229.3, total tree size= 368 MB)
[*]Resource limit exceeded (best possible solution= 224.2, total tree size= 1.1 GB)
[♦]Resource limit exceeded (best possible solution= 252, total tree size= 225 MB)
[◊]Solver ran out of memory (best possible solution= 243, total tree size= 594 MB)