An Efficient MILP Model for the Short-Term Scheduling of Single 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 multiple time grid continuous time MILP model for the short-term scheduling of single stage, multiproduct batch plants where the objective is the minimization of total cost or total earliness. It can handle both release and due dates and it can determine the products delivery dates explicitly if these need to be considered in the objective function. This formulation is compared to other mixed-integer linear programming approaches that have appeared in the literature, to a constraint programming model, and to a hybrid mixed integer linear/constraint programming algorithm. The results show that the proposed formulation is significantly more efficient than the MILP and CP models and comparable to the hybrid model when the objective is the minimization of total cost. For one large instance, both methods exceeded the time limit but the hybrid method failed to find a feasible solution. The results also show that a discrete-time formulation performs very efficiently even when a large number of time intervals are used.

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

1. Introduction

Scheduling is concerned with allocation of resources over time so as to execute the processing tasks required to manufacture a given set of products (Pinedo, 2001). Depending on the amount of resources and time that one has available, finding a feasible or an optimal schedule with respect to a certain objective, can be trivial or very complex. The simplest scheduling problem is the single machine sequencing problem. If more than one equipment or machine exists, the solution involves not only order sequencing of tasks or orders, but also their assignment to a particular machine. The machines can be arranged in parallel, in series or on a more complex structure where a given machine can be used for tasks belonging to different stages of the process. On multistage problems we need to account for the flow of material between stages to ensure that a given order only starts being processed on a stage after it has gone through the previous one, which adds more complexity to the problem. In the limit, orders do not have an identity throughout the process and the materials that make the orders must be considered instead. In that case, order sequencing becomes undefined and different types of variables must be used to model this more general problem.

General mathematical formulations for scheduling are either based on the State Task Network (Kondili et al., 1993) or Resource Task Network (Pantelides, 1994) process representations. STN or RTN-based formulations can be applied to any process plant featuring operations spanning from batch to continuous, consisting of fixed or variable duration tasks, with the major difference being that the STN treats equipment resources implicitly, while the RTN treats them explicitly. Earlier formulations use a discrete representation of time and may involve several thousands of binary variables in order to handle a sufficiently fine discretization that closely matches the exact problem data. As a consequence, when solving large problems, problem data is usually rounded to maintain problem tractability. Overall, discrete-time mixed integer linear programming formulations are usually very tight and perform well for a variety of objective functions, even makespan minimization (Maravelias & Grossmann, 2003b). Continuous-time formulations began to appear in the last decade, and have received much attention recently. Recent examples of STN-based formulations are the work of Giannelos & Georgiadis (2002a, 2002b), Maravelias & Grossmann (2003a) and Janak et al. (2004), while for the RTN we have the work of Castro et al. (2004a, 2005). In continuous-time formulations one needs to specify the number of points that compose the time grid(s), depending on whether the formulation uses a single time grid (Maravelias & Grossmann, 2003a, Castro et al., 2004a, 2005) or one for each equipment resource (Giannelos & Georgiadis, 2002a, 2002b, Janak et al., 2004). Both the computational effort and quality of the solution depend greatly on the number of time points selected, so one must solve a few problems before the optimal solution is found. Overall, MILP continuous-time formulations consider the exact problem data but tend to have larger integrality gaps, meaning that they can only be used to solve small problems. They are also less flexible in terms of handling different objective functions and general, efficient constraints for modeling both release and due dates have still not appeared in the literature.

Single or multistage, multiproduct plants have special characteristics that allow for a different type of mathematical programming approach, where some of the model variables are more closely related to the real world decisions (e.g. assign order *i* to machine *m*, make order *i* before *i*'). Examples of sequential MILP short-term scheduling models can be found in Pinto & Grossmann (1995), Mendez & Cerda (2000, 2002), Jain & Grossmann (2001), Harjunkoski & Grossmann (2002). When compared to the general continuous-time formulations, the more recent sequential models do not need to specify the number of event points, meaning that they only need to be solved once. Another important advantage is that heuristic rules can be used to pre-order items in order to decrease the complexity of the problems and allow for bigger problems to be solved.

Constraint programming (CP) is another technique that can be used for solving some classes of scheduling problems (Baptiste, Le Pape, & Nuijten, 2001). CP is particularly effective for solving feasibility problems and seems to be better suited than traditional MILP approaches in special types of discrete optimization problems where finding a feasible solution is difficult. The lack of an obvious relaxation, however, makes CP worse for loosely constrained problems, where the focus is on finding the optimal solution among many feasible ones and proving optimality. Overall, CP and MILP have complementary strengths that can be combined into hybrid algorithms, yielding considerable computational improvements when compared to the standalone approaches. Examples of these are the work of Jain & Grossmann (2001) for single-stage, Harjunkoski & Grossmann (2002) for multistage multiproduct plants, and Maravelias & Grossmann (2004a, 2004b) for multipurpose plants.

This paper presents a new RTN-based continuous-time MILP model for minimizing cost in the short-term scheduling of single stage multiproduct plants with parallel units of machines. It is based on the general formulation of Castro et al. (2004a), but has one important difference: a different time grid is used for each machine of the process instead of a single time grid for all events taking place. New constraints are presented that allow the consideration of both release and due dates in a general way. We also add new variables to the model that represent the delivery dates of the several product orders. In this way, more complex objective functions, such as earliness minimization can also be considered. This however, gives rise to loose MILP models with which only medium-sized problems can be solved to optimality. In order to address larger problems effectively, we propose an approximation algorithm that first uses a discrete-time formulation to determine the assignments of orders to machines, and then solves the earliness minimization single machine problem for all machines of the process. The new formulation is shown to perform much better than other continuous-time MILP formulations and standalone CP models, and is comparable to the hybrid MILP/CP algorithm of Maravelias & Grossmann (2004b) on a set of example problems where the objective is the minimization of total cost. The strengths and limitations of the six approaches under consideration will be emphasized.

The rest of the paper is structured as follows: Section 2 gives the problem definition. Section 3 presents the new multiple time grid continuous-time formulation as well as the simplified version (for the single stage parallel machine problem) of the general discrete and uniform time grid continuous-time formulations. The main characteristics of other approaches that have been used to solve this specific type of problem are given in section 4, while the results for the two sets of well known problems is left for section 5. Finally, the conclusions are given in section 6.

2. Problem definition

In this paper, the short-term scheduling problem of single stage multiproduct batch plants with parallel units is considered. A set I of product orders is to be processed on a set M of dissimilar parallel machines, where any given machine m can process all orders belonging to set I_m. The processing time of order i on machine m is assumed to be known ($p_{i,m}$), as well as its release r_i and due dates d_i, which are treated as hard constraints. It is also assumed that if setup times exist, they are not sequence dependent, so that they can be incorporated on the processing time. Two alternative objectives are considered: i) minimization of the total cost, where the processing cost of order i on unit m is given by $c_{i,m}$; ii) minimization of total earliness.

3. Mathematical formulations

In this section, three Mixed-Integer Linear Programming models based on the Resource Task Network are presented. Although they use similar sets of variables and constraints, each treats time differently, giving rise to problems of different size and complexity. The type of time grid used by each formulation is shortly described before the model entities are presented.

3.1. Uniform-time grid formulations

In uniform time grid formulations, all events taking place report to a single time grid. The time horizon of interest H, given by difference between the highest due date and the lowest release date ($\max_{i\in I} d_i - \min_{i\in I} r_i$), is divided into |T|-1 time slots. If a discrete representation of time is used, all intervals will have the same duration (δ), with H being a multiple of δ. All processing times must also be multiples of δ, which means that a very small value of δ, and consequently a large number of time intervals, may be required to achieve good approximation to the exact problem data. If one finds that the resulting mathematical problem becomes too difficult to solve, one can use a higher value of δ and round the duration of the tasks to the next integer multiple of δ ($\tau_{i,m}$). The drawback is that suboptimal or infeasible solutions may result since the problem being considered is an approximation of the real one. The discrete-time grid is shown in Figure 1.

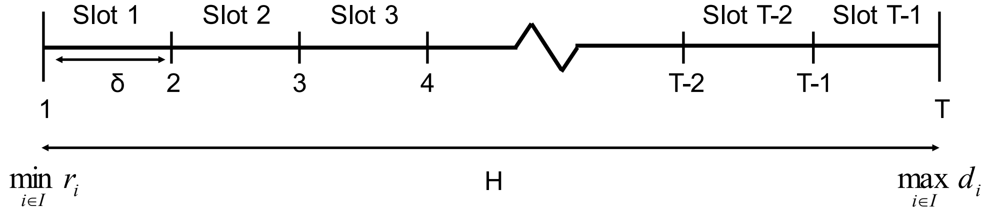

Figure 1. Uniform time grid for discrete-time formulation

The number of time intervals can be reduced if a continuous representation of time is used instead. In such a case, the time points (the elements of set T) usually have at least one task starting or ending[†] so are often called event points. The absolute time of all time points that compose the time grid, and hence the duration of all time intervals, is known only after solving the model. The lower bound on the absolute time of the first event point is equal to the minimum release date and the upper bound on the absolute time of the last event point is equal to the maximum due date. The continuous-time grid is shown in Figure 2.

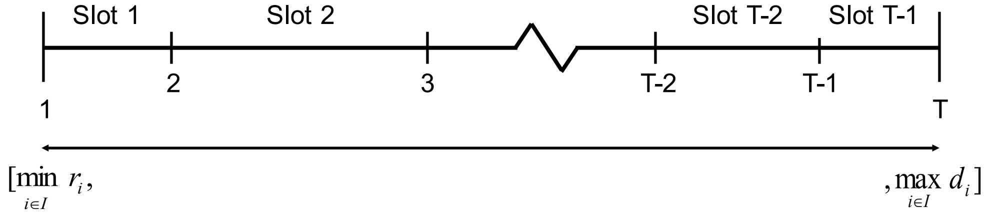

Figure 2. Uniform time grid for continuous-time formulation

3.1.1. Discrete-time formulation (F1)

The discrete-time formulation is very simple since it uses only two sets of variables and constraints plus the objective function. The processing of order i on machine m starting at time point t is identified through the binary variable $N_{i,m,t}$, while the availability of a given machine at the same time point is

[†] When more time points are used than those required to find the optimal solution, there can be time points where no task is starting or ending. However, these are easily identified since they will have the same absolute time as other event points.

given by the excess resource variable $R_{m,t}$ (equal to one if the machine is available, zero otherwise). It is worth mentioning that the excess resource balance (eq 1) ensures that the excess resource variables $R_{m,t}$ can only take 0 or 1 values, so they can be defined as continuous variables instead of binary variables. The large number of time intervals usually required in discrete-time formulations makes this option computationally more effective. Note however, that when few intervals are used, e.g. in continuous-time formulations, the effect can be the opposite so the two alternatives should be tried.

Due to the release and due dates, each order can only start on a subset of the total number of time points on the grid, |T|. Furthermore, since the approximated processing times ($\tau_{i,m}$) on the various machines can be different, the number of possible starting points will also be machine dependent. The set of orders that can start at time point t on machine m is specified in $I_{t,m}$ and its consideration significantly reduces the number of binary variables in the model. The other required subsets are given in the Nomenclature section.

The first constraint is the excess resource balance, which is a typical multiperiod balance, where the availability of machine m at time point t is equal to that at the previous time point, minus one if there is a task starting at t, plus one if there is a task ending at t (starting at t-$\tau_{i,m}$). Notice that 1 represents the initial resource availability, only used at the first time point. The second constraint is simpler, stating that all orders must be processed exactly once.

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

$$\sum_{t\in T_{i,m}} \sum_{m\in M_i} N_{i,m,t} = 1 \ \forall i \in I \tag{2}$$

As mentioned in section 2, two different objective functions will be considered. The first, minimization of total cost, is given in eq 3,

$$\min \sum_{t\in T_{i,m}} \sum_{m\in M_i} \sum_{i\in I} N_{i,m,t} c_{i,m} \tag{3}$$

The second, minimization of total earliness, is more difficult to define. In eq 4, the second term represents the time corresponding to the first time point, while the third term represents the ending times of all processing tasks (number of time intervals spanned by all tasks multiplied by the duration of each time interval).

$$\min \mathrm{Z} = \sum_{i \in I} d_i - \min_{i' \in I} r_{i'} - \sum_{t \in T_{i,m}} \sum_{m \in M_i} \sum_{i \in I} N_{i,m,t} (t + \tau_{i,m} - 1)\delta \tag{4}$$

3.1.2. Continuous-time formulation (F2)

The continuous-time formulation presented below is a simplified version of the general formulation presented in Castro et al. (2004a) with the addition of release and due date constraints. Due to the special characteristics of the problem being considered, a processing task is now referred by two indexes (order *i*, plus *m* for the machine where it is executed) and we only need to be concerned with the equipment resources, |M|.

The continuous time formulation uses more sets of variables and constraints than its discrete-time counterpart due to the fact that the time corresponding to each event point (T_t) is unknown. The other important difference is that when considering batch tasks, we do not know a priori how many time intervals a particular task will span. Thus, both the starting and ending event point of the task must be considered, which means that the binary extent variables will have two time indexes instead of one: $\overline{N}_{i,m,t,t'}$. Regarding this set of variables, it will be assumed that each task can only span a limited number of time intervals: $t' \leq t + \Delta t$. Although Δt may act as a hidden constraint, the use of a value smaller than the maximum one (|T|-1) substantially improves the performance of the formulation (further details can be found in Castro et al., 2005).

The first two constraints (eq 5 and 6) are equivalent to eq 1 and 2, but now one more summation is required to find the exact event point where the task ends (if started at *t*) or starts (if it ends at *t*), see eq 5. Notice also that all orders can start at any point but the last. This is because we do not know for sure which variables can be eliminated from the formulation without compromising the optimal solution, even so it is expected that orders with earlier due dates will end at lower event points.

$$R_{m,t} = (1|_{t=1} + R_{m,t-1}|_{t\neq 1}) - \sum_{i \in I_m} \sum_{\substack{t' \in T \\ t < t' \leq t+\Delta t}} \overline{N}_{i,m,t,t'} + \sum_{i \in I_m} \sum_{\substack{t' \in T \\ t-\Delta t \leq t' < t}} \overline{N}_{i,m,t',t} \quad \forall m \in M, t \in T \tag{5}$$

$$\sum_{m \in M_i} \sum_{t \in T} \sum_{\substack{t' \in T \\ t < t' \le t + \Delta t}} \overline{N}_{i,m,t,t'} = 1 \; \forall i \in I \tag{6}$$

The next set of constraints relates the time of two event points to the processing time of the task occurring between those two event points. Although it can either be written per order or per machine, the latter option is preferred (eq 7) since it leads to a better performance (see Castro et al., 2004a). The model would be complete if it were not for the release and due date constraints. Eq 8 states that if order *i* starts at time point *t*, the absolute time of event point *t* cannot be lower than its release date. Eq 9 is the equivalent constraint for due dates, and is formulated as a big-M constraint (active if there is an order being processed on machine *m* that actually ends at *t*, otherwise the constraint is relaxed, see eq 10). Eq 11 fixes the time of the last event point to the maximum due date, which although not necessary, also improves the performance of the model.

$$T_{t'} - T_t \ge \sum_{i \in I_m} \overline{N}_{i,m,t,t'} p_{i,m} \; \forall m \in M, t, t' \in T, t < t' \le t + \Delta t \tag{7}$$

$$T_t \ge \sum_{i \in I_m} \sum_{\substack{t' \in T \\ t < t' \le t + \Delta t}} \overline{N}_{i,m,t,t'} r_i \; \forall m \in M, t \in T, t \ne |T| \tag{8}$$

$$T_t \le H(1 - \sum_{i \in I_m} \sum_{\substack{t' \in T \\ t - \Delta t \le t' < t}} \overline{N}_{i,m,t',t}) + \sum_{i \in I_m} \sum_{\substack{t' \in T \\ t - \Delta t \le t' < t}} \overline{N}_{i,m,t',t} d_i \; \forall m \in M, t \in T, t \ne 1 \tag{9}$$

$$T_t \le H \; \forall t \in T \tag{10}$$

$$T_{|T|} = H = \max_{i \in I} d_i \tag{11}$$

If one intends to minimize the total cost, the model ends with eq 14. Otherwise, if one wants to minimize the total earliness (eq 15), one additional set of positive continuous variables DD_i, the delivery date of order *i*, and two more sets of big-M constraints, must be defined. Note that in eq 12, the delivery date of order *i* must not be lower than the absolute time of event point *t* if it ends at or after *t*. Similarly, in eq 13, the delivery date must not be higher than T_t if the order ends at or before *t*.

$$DD_i \ge T_t - H(1 - \sum_{m \in M_i} \sum_{\substack{t'' \in T \\ t' - \Delta t \le t'' < t'}} \sum_{\substack{t' \in T \\ t' \ge t}} \overline{N}_{i,m,t'',t'}) \; \forall i \in I, t \in T, t \ne 1 \tag{12}$$

$$DD_i \leq T_t - d_i \Big(1 - \sum_{m \in M_i} \sum_{\substack{t'' \in T \\ t' - \Delta t \leq t'' < t'}} \sum_{\substack{t' \in T \\ t' \leq t}} \overline{N}_{i,m,t'',t'}\Big) \; \forall i \in I, t \in T, t \neq 1 \quad (13)$$

$$\min \sum_{\substack{t' \in T \\ t < t' \leq t + \Delta t}} \sum_{t \in T} \sum_{m \in M_i} \sum_{i \in I} \overline{N}_{i,m,t,t'} c_{i,m} \quad (14)$$

$$\min \sum_{i \in I} (d_i - DD_i) \quad (15)$$

3.2. Multiple time grid, continuous time formulation (F3)

The continuous-time formulation presented below uses multiple time grids, i.e. each machine uses a time grid similar to the one given in Figure 2. As a consequence, events occurring on machine *m* will report only to time grid *m*, and all |M| time grids are completely independent, meaning that no relation is assumed between event points of different time grids. Nevertheless, the time grids have two things in common: i) same number of time points; ii) same lower and upper bounds on the first and last time points, respectively. A representation of the multiple time grid formulation, which highlights the independence of the several time grids, is shown in Figure 3.

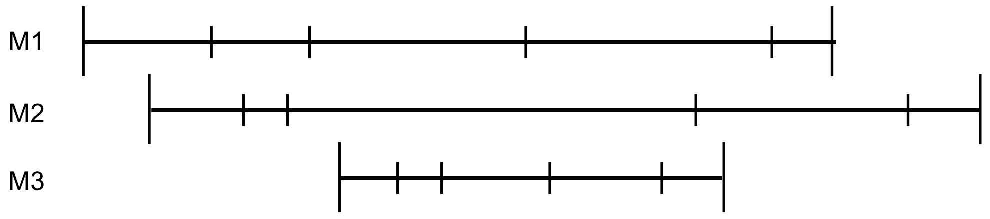

Figure 3. Possible solution of non-uniform time grid continuous-time formulation (|I|=15, |M|=3, |T|=6)

The advantage of considering |M| independent time grids is that we can assume, without loss of generality, that all processing tasks last exactly one time interval. This means that if a task starts at event point *t*, it will end at *t*+1 so only one time index needs to be considered in the binary extent variables, $N_{i,m,t}$, just like in the discrete time formulation. The other difference in terms of model variables, when compared to the uniform time grid continuous-time formulation, is that |M|×|T| event points must be assigned a time, so the continuous variables that represent the absolute time of a given event point have two indices: $T_{t,m}$. Variables $R_{m,t}$ have the same meaning as in the two previous models, but based on computational experience it is better to consider them as binary variables.

The model constraints shown next are very similar to the constraints of the uniform-time grid formulation, so there is no need of explaining them again. There are however, two exceptions. The first, is that we also consider time matching constraints (eq 19) to enforce the difference between the times of two consecutive event points to be equal to the processing time of the order being executed, if there is one (if not, the constrained is relaxed to $T_{t+1,m}$-$T_{t,m}$≤H). These constraints, although not necessary, lead to better computational performances whenever its inclusion does not increase the number of time points required to find the global optimum solution (note that without the time matching constraint each task may last more than its processing time, and thus incorporate an eventual idle time of the machine where it is being processed, which is an effective way of decreasing the required number of time points). Note also, that if one wants to remove solutions that feature waiting periods between orders, then the first term on the right-hand side of eq 19 can be removed, which is equivalent to turning equation 18 into an equality. Although this typically leads to an improved computational performance, there is the risk of excluding the optimal solution from the formulation. The second exception, is that it is not convenient to fix the time of the last time point of all grids to the maximum due date, since we do not know where that order is going to be processed (see eq 22). The objective for minimizing earliness is written in exactly the same way as for model F2 (see eq 15).

$$R_{m,t} = (1\big|_{t=1} + R_{m,t-1}\big|_{t\neq 1}) - \sum_{i\in I_m} N_{i,m,t} + \sum_{i\in I_m} N_{i,m,t-1} \;\forall m \in M, t \in T \tag{16}$$

$$\sum_{m\in M_i} \sum_{t\in T} N_{i,m,t} = 1 \;\forall i \in I \tag{17}$$

$$T_{t+1,m} - T_{t,m} \geq \sum_{i\in I_m} N_{i,m,t} p_{i,m} \;\forall m \in M, t \in T, t \neq |T| \tag{18}$$

$$T_{t+1,m} - T_{t,m} \leq H(1 - \sum_{i\in I_m} N_{i,m,t}) + \sum_{i\in I_m} N_{i,m,t} p_{i,m} \;\forall m \in M, t \in T, t \neq |T| \tag{19}$$

$$T_{t,m} \geq \sum_{i\in I_m} N_{i,m,t} r_i \;\forall m \in M, t \in T, t \neq |T| \tag{20}$$

$$T_{t,m} \leq H(1 - \sum_{i\in I_m} N_{i,m,t-1}) + \sum_{i\in I_m} N_{i,m,t-1} d_i \;\forall m \in M, t \in T, t \neq 1 \tag{21}$$

$$T_{t,m} \leq H = \max_{i \in I} d_i \quad \forall m \in M, t \in T \tag{22}$$

$$DD_i \geq T_{t,m} - H(1 - \sum_{m \in M_i} \sum_{\substack{t' \in T \\ t' \geq t}} N_{i,m,t'-1}) \quad \forall i \in I, m \in M_i, t \in T, t \neq 1 \tag{23}$$

$$DD_i \leq T_{t,m} - d_i(1 - \sum_{m \in M_i} \sum_{\substack{t' \in T \\ t' \leq t}} N_{i,m,t'-1}) \quad \forall i \in I, m \in M_i, t \in T, t \neq 1 \tag{24}$$

$$\min \sum_{t \in T} \sum_{m \in M_i} \sum_{i \in I} N_{i,m,t} c_{i,m} \tag{25}$$

4. Other approaches

The single stage parallel scheduling problem can also be solved by other approaches. The three alternative models tested are essentially those described in Jain & Grossmann (2001). Thus, instead of showing the detailed models we will focus on the main characteristics of each method and highlight its differences and similarities to the formulations shown in section 3. The changes made to the original continuous-time and hybrid models will also be mentioned.

4.1. MILP model with sequencing variables (F4)

Not all continuous-time formulations need to consider one or more time grids explicitly. If one considers binary assignment $x_{i,m}$ and sequencing variables $y_{i,i'}$, no time indexes are required to generate a MILP that can solve the problem at hand. In this way, only one problem needs to be solved to find the optimal solution, instead of a few problems (in the search for the adequate number of event points, |T|, see sections 3.1.2 and 3.2). Other advantages include the possibility to enforce or forbid certain product sequences and the way sequence-dependent due dates are considered. Enforcing and/or forbidding certain product sequences means considering fewer binary sequencing variables, while sequence dependent due dates are treated with exactly the same constraints (one more term is added to the timing constraints relating the starting time of orders *i* and *i'*). These two problem characteristics, which are very common in reality, can be difficult to implement on the mathematical formulations presented in section 3, and usually involve adding more variables and constraints, thus increasing its complexity.

When testing the performance of the MILP model of Jain & Grossmann (2001), it was found that for the objective of minimizing cost, removing the logical cuts (constraints 22 and 23 of their original work), which involve a large number of constrains (constraint 23 uses 4 indexes: *i*,*i*',*m*,*m*'), generally yields better computational performance. For the objective of minimizing earliness, the results were not conclusive so we opted to maintain those constraints.

4.2. Constraint programming model (F5)

The same scheduling problem can also be modeled using constraint programming (CP). CP models, in contrast to MILP models, are highly dependent on the CP package used to model the problem. In this paper we use ILOG's OPL modeling language (van Hentenryck, 1999), which has a set of constructs especially designed for scheduling problems. The basic OPL modeling framework is similar to the Resource Task Network in how it looks at the problem: a set of activities (tasks) that need to be performed using a certain set of resources, where the equipment resources (the machines) are defined as *unary resources*. CP models can be viewed as discrete-time models with intervals of one time unit length, since all variables of a given activity (*start*, *duration* and *end*, with *start*+*duration*=*end*) are integer variables.

4.3. Hybrid MILP/CP model (F6)

The strengths of both models can be combined by using a simplified version of the MILP for the assignments, and then solve |M| single machine problems with CP to sequence the orders that were assigned to a particular machine. When minimizing the total cost (defined in the MILP by $\Sigma x_{i,m}c_{i,m}$), this decomposition strategy has the advantage of always leading to the global optimum solution since the objective function only depends on the assignment variables and the simplified MILP is a less constrained model than the full MILP. The only drawback is that the assignments may be infeasible on one or more machines, but that can be overcome simply be adding integer cuts to the MILP and iterating until all CP sequencing problems are feasible. While the same decomposition strategy can be used when minimizing earliness, or generally when considering an objective that is a function of the sequencing variables, the performance of this decomposition is likely to worsen as then the CP has to solve an optimization problem rather than a feasibility problem.

Following the work of Jain & Grossmann (2001), Bockmayr & Pisaruk (2003) generalized the integer cuts and used them in a branch and cut framework, while Sadykov & Wolsey (2005) proposed a tighter formulation for the MILP and explored several integrated schemes. Also, Maravelias & Grossmann (2004b) have proposed a pre-processing algorithm that generates knapsack constraints or cover cuts for certain subsets of orders that can be added to the cut pool of the MILP a priori. The pre-processing algorithm was shown to reduce the computational effort by one order of magnitude in the set of instances studied by Jain & Grossmann (2001). This work uses the knapsack constraints proposed by Maravelias & Grossmann (2004b) for the single stage plant.

5. Computational results

In this section, the performance of the mathematical formulations is illustrated through the solution of several example problems. Two sets of case studies will be considered. The first set concerns total cost minimization and all 6 approaches presented in sections 3 and 4 will be tested. The second set of case studies involves the minimization of total earliness, and all except the hybrid MILP/CP approach (for the reasons explained in section 4.3) will be tested. All MILP models, where solved to optimality (1E-6 relative tolerance), unless otherwise stated, on a Pentium-4 2.8GHz machine, running the commercial solver GAMS/CPLEX 9.0. The CP and hybrid MILP/CP models where implemented and solved in ILOG's OPL studio 3.7, on the same machine.

5.1. Problem set 1: minimize total cost

The first six problems to be considered correspond to the single stage example problems 3.1-5.2 of Harjunkoski and Grossmann (2002), with the size of the problems spanning from 12 orders on three machines to 20 orders on five machines. It will be seen that half of the models tested can solve all of these 6 instances rather fast, so in order to find the best approach, four other, five machine problems, are solved. The data for these problems are given in Table 1, where problem S1G comprises the first 25 orders and problem S1J the full set of orders. Further and less constrained versions of these problems were also solved: in problem S1H the processing times of S1G were increased 5% and rounded to the closest integer value, while in S1I the processing times of S1J were decreased by 20% and rounded. The results obtained are summarized in Table 2 and discussed in the next couple of sections.

Table 1. Data for problems S1G and S1H

	Dates (day)		$p_{i,m}$ (day)/$c_{i,m}$				
Order	r_i	d_i	*M1*	*M2*	*M3*	*M4*	*M5*
I1	66	110	30/5	53/2	39/4	36/4	56/1
I2	40	188	30/5	50/2	50/2	57/1	43/3
I3	65	163	35/4	45/3	42/3	45/3	41/3
I4	28	137	46/3	40/3	38/4	41/3	31/5
I5	36	221	46/3	34/4	50/2	55/1	57/1
I6	10	126	47/3	49/2	31/5	29/5	54/2
I7	56	286	38/4	55/1	33/4	34/4	51/2
I8	50	237	29/5	57/1	42/3	42/3	49/2
I9	20	254	48/2	54/2	50/2	38/4	40/3
I10	26	119	32/5	39/4	50/2	42/3	39/4
I11	46	229	33/4	48/2	49/2	53/2	34/4
I12	78	189	31/5	57/1	49/2	52/2	42/3
I13	88	159	53/2	40/3	42/3	44/3	36/4
I14	53	219	28/5	55/1	29/5	28/5	57/1
I15	46	281	51/2	58/1	33/4	53/2	40/3
I16	95	269	43/3	57/1	32/5	39/4	44/3
I17	94	200	38/4	39/4	45/3	37/4	49/2
I18	12	258	55/1	34/4	58/1	56/1	40/3
I19	72	142	54/2	53/2	49/2	44/3	38/4
I20	12	184	28/5	57/1	38/4	43/3	51/2
I21	66	294	33/4	55/1	36/4	43/3	48/2
I22	29	184	54/2	58/1	49/2	47/2	31/5
I23	2	295	48/2	54/2	49/2	33/4	31/5
I24	99	156	50/2	29/5	37/4	40/3	45/3
I25	81	142	43/3	53/2	41/3	33/4	38/4
I26	3	270	42/3	50/2	33/4	52/2	37/4
I27	45	277	41/3	54/1	57/1	43/3	49/2
I28	2	134	49/2	50/2	40/3	37/4	45/3
I29	16	170	54/2	37/4	48/2	48/2	43/3
I30	75	157	57/1	43/3	57/1	52/2	37/4

Table 2. Overview of computational performance (CPU s) for total cost minimization

Type of Model	Discrete-time MILP	Continuous-time MILP			CP	Hybrid MILP/CP
Problem/Model	F1	F2	F3	F4	F5	F6
S1A (12 orders, 3 machines)	18.9	3600†,‡	10.7	15.2	0.98	0.44
S1B (12 orders, 3 machines)	5.67	0.13	0.10	0.05	0.13	0.03
S1C (15 orders, 4 machines)	13.9	-	1.16	33.3	97.3	0.63
S1D (15 orders, 4 machines)	5.48	-	0.14	5.32	6.53	0.45
S1E (20 orders, 5 machines)	7.25	-	62.1	421	3600†,*	0.99
S1F (20 orders, 5 machines)	35.6	-	1.96	3600†,*	2326	0.77
S1G (25 orders, 5 machines)	2.47	-	471	3600†,*	518	172
S1H (25 orders, 5 machines)	4.70	-	115	3600†,*	4133	968
S1I (30 orders, 5 machines)	13.5	-	45.5	107.8	183	46.8
S1J (30 orders, 5 machines)	27.2	-	3600†,*	3600†,*	3600†,*	3600†,‡

†Maximum resource limit
‡ No solution found
*Suboptimal solution returned

5.1.1. Problems S1A-S1F

The computational statistics for the first six problems are given in Table 3 through Table 5. Each table features the two problems of similar complexity, i.e. same number of orders and machines. The analysis of the results is performed model by model.

In the discrete-time formulation (F1), in order to match the exact problem data, we must set $\delta = 1$ and use a total of 380 time intervals in problems S1A/B/E/F and 370 intervals in problems S1C/D. As a consequence, large MILPs are generated. Despite their size, the resulting MILPs are solved rather fast, in part due to their low integrality gaps, and usually on the first nodes of the search tree. Furthermore, increasing the complexity of the problem from 12 orders on 3 machines to 20 orders on 5 machines has little effect on the computational effort.

The constraint programming model (F5), like the discrete-time formulations, is also limited to integer data. This is the only resemblance to F1 since CP uses much fewer variables and constraints and its performance is highly dependent on the problem size. As the size increases so does the number of choice points and the computational effort. While problems S1A/B are solved in less than one second, problems S1C/D take several seconds to solve, S1F takes almost 40 minutes to solve and most importantly, the optimal solution cannot be found for S1E in one hour of computational time

(Harjunkoski & Grossmann, 2002, report 189,244 s to find the optimum), even though a good solution is found in less than one minute.

The results in Table 3 show that the uniform time grid continuous-time formulation (F2) has the worst performance of all models tested. Interestingly, it has completely distinct performances for problems S1A and S1B. While the former is intractable, the latter is solved in just 0.13 s. This difference in behaviour is better understood by recalling that the computational effort increases substantially with an increase in the number of event points ($|T|$) and/or the maximum number of time intervals that a particular task can span (Δt). While S1A requires $|T|=12$ and $\Delta t=5$ to find the global optimum (this can be confirmed by using the solution from F3 to fix the binary variables of model F2), S1B requires only $|T|=9$ and $\Delta t=2$. Since an increase in the number of orders leads to an increase in the number of event points required to solve the problem, there is no point in solving the other problems of this section with model F2.

When going from the uniform to the multiple time grid continuous-time formulation (F3) the size of the resulting MILPs is significantly smaller, the integrality gap is lower and, above all, the computational performance is greatly reduced (S1A is solved in 10.7 s). When using multiple time grids all orders last only one time interval, which means that we only need to be concerned with specifying $|T|$ (one starts at $|I|/|M|+1$ and continues to increase the number of time points until no improvement is found on the value of the objective function). The other advantage becomes apparent when comparing the results for problems S1A/B and S1C/D. Although the size roughly doubled, due to the fact that S1C/D consider three more orders and two more machines, the number of orders per machine, which has a direct influence on the number of required time points and hence on the computational effort, decreased. As a consequence, problems S1C/D took less time to solve by (F3) than S1A/B, while for the continuous-time formulation with sequencing variables (F4) the computational time increased substantially. Furthermore, the MILPs resulting from (F3) have slightly lower integrality gaps than those resulting from (F4), require a larger number of binary variables, but are solved in significantly less time. Note also that the continuous-time formulation with sequencing variables (F4) cannot find the optimal solution for S1F in one hour of computational time, while the proposed formulation takes less than two seconds to find and prove optimality.

Table 3. Computational statistics for problems S1A-S1B

Problem	S1A						S1B					
Model	F1	F2	F3	F4	F5	F6	F1	F2	F3	F4	F5	F6
|T|	381	12	6				381	9	9			
Discrete variables	6948	1620	198	168			8914	540	315	168		
Single variables	8092	1669	217	181	72		10058	577	343	181	72	
Constraints	1156	250	91	358	84		1156	133	136	358	84	
RMIP	101.86	84.67	98.10	97.89			84.92	84.54	84.9	84.88		
Obj	104	-	104	104	104	104	85	85	85	85	85	85
CPU	18.9	3600†	10.7	15.2	0.98	0.44	5.67	0.13	0.10	0.05	0.13	0.03
Nodes	153	864752	20570	10469			7	0	0	0		
Choice points					3593						818	
Major iterations						3						1
Total cuts						109						31

†Resource limit exceeded (best possible solution= 88.01, total tree size= 590 MB)

Table 4. Computational statistics for problems S1C-S1D

Problem	S1C					S1D				
Model	F1	F3	F4	F5	F6	F1	F3	F4	F5	F6
|T|	371	6				371	8			
Discrete variables	13320	405	285			16950	565	285		
Single variables	15176	436	301	90		18806	606	301	90	
Constraints	1871	146	771	135		1871	196	771	135	
RMIP	116	113.03	112.73			104	104	103.78		
Obj	116	116	116	116	116	105	105	105	105	105
CPU	13.9	1.16	33.3	97.3	0.63	5.48	0.14	5.32	6.53	0.45
Nodes	15	1305	10356			1	0	2704		
Choice points				424969					22791	
Major iterations					2					2
Total cuts					302					174

The hybrid MILP/CP model (F6) has the best performance overall, being beaten only by the proposed formulation for problem S1D. The cuts proposed by Maravelias & Grossmann (2004b) make the hybrid model very efficient, since few assignment problems (the number of major iterations) need to be solved (the maximum, 3 iterations, was found for problem S1A). Without the knapsack constraints of Maravelias & Grossmann (2004b), S1A requires a total of 27 major iterations and 42 integer cuts to find the optimal solution in 9.0 CPUs (instead of 0.44 s, see Table 3), while S1E requires 22 major iterations and 38 integer cuts for a total computational effort of 18.0 CPUs, instead of 0.99 s, see Table 5.

Table 5. Computational statistics for problems S1E-S1F

Problem	S1E					S1F				
Model	F1	F3	F4	F5	F6	F1	F3	F4	F5	F6
|T|	381	7				381	9			
Discrete variables	18790	635	480			23320	845	480		
Single variables	20696	671	501	120		25226	891	501	120	
Constraints	1926	176	1376	180		1926	226	1376	180	
RMIP	158.5	155.79	155.46			142.94	143	142.11		
Obj	159	159	159	169	159	144	144	145	144	144
CPU	7.25	62.1	421	57.0†	0.99	35.6	1.96	3600†	2326	0.77
Nodes	0	74653	100399			82	1238	573844		
Choice points				207504					6580032	
Major iterations					1					1
Total cuts					592					411

†Resource limit exceeded (best possible solution= 143, total tree size= 91 MB)

5.1.2. Problems S1G-S1J

The purpose of the last four problems is to identify which model is more adequate for significantly larger problems, since S1G and S1H deal with 25 orders and S1I and S1J with 30 orders on five machines (see Table 1). The results in Table 6 and Table 7 are conclusive since only the discrete-time formulation (F1) remains quite efficient for all cases (the most difficult problem, S1J takes only 27.2 s to solve). The continuous-time formulation with multiple time grids (F3), the constraint programming model (F5) and the hybrid model (F6) can all find the optimal solution of problems S1G-S1I in reasonable time, while the continuous model with sequencing variables (F4) only finds a reasonable solution. Making the problems more constrained, by increasing the processing times (when going from S1G to S1H and S1I to S1J), besides increasing the value of the objective function, generally increases the computational effort, the only exception being the continuous multiple grid model (F3), which solves problem S1H in about one fourth of the time required to solve S1G. Problem S1J is by far the most difficult instance and some models (F4 and F6) are even unable to find a feasible solution to the problem. A more detailed analysis of the performance of each formulation follows.

The discrete-time formulation is surprisingly efficient. Even considering that the MILPs resulting from F1 exhibit the lowest integrality gap of all three MILPs and that the size of the resulting MILPs are now smaller than in the previous two problems, because the time horizon as been reduced (problems S1G-S1J require 294 time points instead of 381), it is difficult to explain why so few nodes are required. One could think that this is because the solution of the relaxed problem has most of the binary variables set to 0 or 1, but this is completely false. It just seems that every decision has a large impact on the model, so CPLEX is able to find the optimal solution very rapidly. The solution for problem S1J is shown in Figure 4. Notice that the 30 orders have been divided equally between the five machines and that there are few waiting periods.

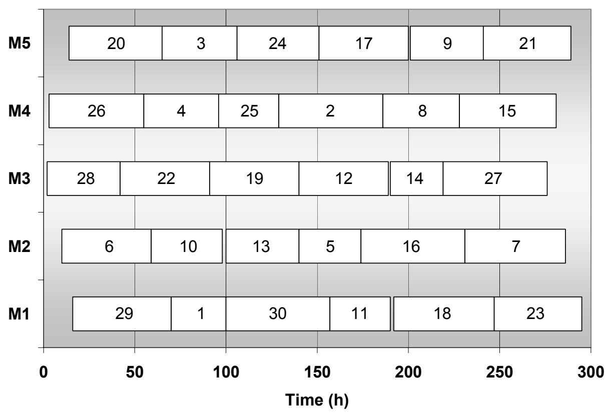

Figure 4. Optimal solution for problem S1J

The performance of the multiple time grid continuous-time formulation (F3) is also very good. Even though it is only the third best performer for problem S1G, for the other three problems (S1H-S1J) is only beaten by the discrete-time formulation (F1). Notice that for S1J, the number of event points was set to the minimum possible value, 7 (|I|/|M|+1=30/5+1), but the problem complexity is already very large (the tree size continued to increase rapidly, after one hour of computational time), even though the solver finds a very good solution. Interestingly, 7 event points are enough to find the global optimum solution (if one uses the solution from F1 to fix the assignments and sequence on machine M1, i.e. order 29 starts at the first event point, order 1 at the second event point and so on, the problem is solved in less than 19 s, even without fixing the absolute times of the several event points). The large influence on computational effort resulting from just a few assignments is a characteristic of these types of continuous-time formulations and it can be used to derive more efficient algorithms (see for instance work by Castro et al., 2004b).

Table 6. Computational statistics for problem S1G-S1H

Problem	S1G					S1H				
Model	F1	F3	F4	F5	F6	F1	F3	F4	F5	F6
|T|	294	6				294	6			
Discrete variables	13548	655	725			13273	655	725		
Single variables	15019	686	751	150		14744	686	751	150	
Constraints	1496	156	2156	225		1496	156	2156	225	
RMIP	50.25	43.28	46.72			53.71	46.87	46.98		
Obj	51	51	54	51	51	54	54	60	54	54
CPU	2.47	471	3600†	518	172	4.70	115	3600‡	4133	968
Nodes	0	311077	398140			0	64070	487058		
Choice points				829542					6538160	
Major iterations					50					53
Total cuts					933					954

†Resource limit exceeded (best possible solution= 47.13, total tree size= 176 MB)
‡Resource limit exceeded (best possible solution= 47.26, total tree size= 240 MB)

Table 7. Computational statistics for problem S1I-S1J

Problem	S1I					S1J				
Model	F1	F3	F4	F5	F6	F1	F3	F4	F5	F6
|T|	294	7				294	7			
Discrete variables	18063	935	1020			16753	935	1020		
Single variables	19534	971	1051	180		18224	971	1051	180	
Constraints	1501	186	3111	270		1501	186	3111	270	
RMIP	53	50	51.44			73.85	69.32	59.62		
Obj	53	53	53	53	53	75	76	98	83	-
CPU	13.5	45.5	107.8	183	46.8	27.2	3600†	3600‡	107*	3600$^{\blacklozenge}$
Nodes	8	17345	11470			3	1081000	68467		
Choice points				228701					259260	
Major iterations					12					1
Total cuts					1150					1163

†Resource limit exceeded (best possible solution= 72.75, total tree size= 344 MB)
‡Resource limit exceeded (best possible solution= 61.55, total tree size= 57 MB)
*Computational time required to find the solution reported (solver terminated at 3600 CPUs)
$^{\blacklozenge}$Resource limit exceeded for first MILP (integer solution= 75, best possible solution= 67.15)

The failure of the hybrid MILP/CP model (F6), the best performer so far (see Table 2), for S1J, is the most surprising result. Until now, the global optimal solution could be found in few iterations and this was done in less than one second. The problems considered in this section are however more complex and more iterations are required to find the global optimal solution. Furthermore, each iteration takes more time to complete, for example problem S1G and S1I average 3.4-3.9 s per iteration, while S1H averages more than 18 s. These two combined factors significantly decrease the performance of the hybrid MILP/CP model. Nevertheless, it is still the second best formulation for problem S1G and the third best for problems S1H and S1I. The major drawback of the hybrid model is highlighted when solving problem S1J. The MILP with assignment and sequencing variables is intractable (see results for model F4) mainly due to its very large integrality gap. The simplified MILP generated from the hybrid model suffers from a similar problem and even though the integrality gap is lower (the solution of the relaxed problem equals 64.4) and a feasible solution is found, the solver is still far from proving optimality in one hour of computational time. Interestingly, the best solution found has the same objective as the global optimum solution, the difference being that for the simplified MILP we are not sure that that solution is even feasible. It is now clear that the decision of dividing the problem in two, assignment and feasibility problems, is good but has two important limitations. The first was already known, this decomposition strategy is only effective when the objective function depends solely on the assignment variables. The second is that several assignment problems may need to be solved to find the optimal solution. If the resulting MILPs have a small size and a low integrality gap, this approach works fine, but if at least one of the conditions fails, then we have a problem, as seen for problem S1J. And as the hybrid model only reaches the feasible region when it finds the optimal solution, we are even without a feasible solution to the problem. Thus, replacing the pure MILP or constraint programming models with the hybrid approach by Jain & Grossmann (2001) is not a good idea for large problems.

Finally, the constraint programming model (F5) also has reasonable performance. It is particularly interesting to see that for problem S1G it performs almost as well as the multiple time-grid continuous-time formulation (F3), a behaviour that was observed only in the first problems (S1A/S1B). For S1J, the CP model finds a good solution in less than two minutes of computational time but one more hour of computational time does not lead to any improvements. Nevertheless, it is significantly better than the solution found by the continuous-time formulation with sequencing variables (F4) in the same computational time.

5.2. Total earliness vs. total cost

Changing the objective function from total cost to total earliness does not affect the model constraints in the case of the discrete-time formulation (F1) (see section 3.1.1) or the constraint programming model (F5). However, the continuous-time formulations need some changes. All three models require a new set of variables, the orders delivery dates (DD_i), and some changes in the model constraints. While in the time grid models two additional sets of constraints are required (eqs 12 and 13 for F2, and eqs 23 and 24 for F3), in the one with sequencing variables (F4) only one adjustment is needed (in equation 17 of Jain & Grossmann, 2001, replace d_i by DD_i and turn the inequality into equality). To see the impact of the objective function change on the computational effort, we have solved problems S1A-S1B for minimizing earliness. The results are given in Table 8.

When comparing Table 8 to Table 3 it is clear that only the discrete-time model (F1) performs better for total earliness than for total cost. The reason for this behaviour can be explained by the fact that now, the time point at which each order is executed also affects the objective function, meaning fewer degenerate solutions and better performance. The constraint programming model (F5) has the second best performance, but the computational effort for S1B has increased by two orders of magnitude (19 vs. 0.13 s), despite requiring fewer variables than for total cost minimization (60 vs. 72). Nevertheless, it still performs much better than the two continuous-time models tested (F2 even has a worse performance than F3, so it was not considered). It is interesting to see that the continuous-time model with sequencing variables (F4) is much more efficient than the multiple time grid model (F3), behaviour that is the exact opposite from that found for minimizing cost. The reason for this was already given in the previous paragraph. While (F4) originates very similar MILPs for both objectives, (F3) must consider two more sets of complex, big-M constraints, which are known to increase the integrality gap and make the MILPs more difficult to solve (note that while F3 and F4 have the same integrality gap, optimality can be proved in a reasonable time for problems resulting from F4, whereas the absolute gaps for those resulting for F3 were still very large after one hour of computational time).

Table 8. Problems S1A-S1B revisited. Computational statistics for total earliness minimization.

Problem	S1A				S1B			
Model	F1	F3	F4	F5	F1	F3	F4	F5
|T|	381	6			381	6		
Discrete variables	6948	198	168		8914	198	168	
Single variables	8092	229	193	60	10058	229	193	60
Constraints	1156	436	820	84	1156	436	820	84
RMIP	733	0	0		97	0	0	
Obj	770	770	770	770	98	98	98	98
CPU	5.61	3600*	1663	4.56	1.56	3600†	95.0	19.0
Nodes	107	859548	463155		3	686730	62867	
Choice points				4570				93048

*Resource limit exceeded (best possible solution= 380.5, total tree size= 111 MB)
†Resource limit exceeded for first MILP (best possible solution= 0, total tree size= 97 MB)

5.3. An efficient strategy for total earliness minimization

The results of the previous section have shown that the size of the problems that can be solved efficiently by the continuous-time or constraint programming models is smaller for earliness minimization than for cost minimization. The discrete-time formulation is very efficient but has one important disadvantage: due to the discretization of the time horizon, a very large number of time intervals may be required to consider the exact problem data, inevitably leading to problem intractability. However, as mentioned in section 3.1, one can always consider an approximation of the problem by rounding the problem data to integer multiples of the interval length (δ). Rounding up the data ensures that if the approximate problem is feasible so is the real problem. Furthermore, the optimal solution of the former will be a very good upper bound to the optimal solution of the latter, with better approximations resulting from lower rounding errors.

The optimal solution from the discrete-time formulation (approximate problem) can then be used to find a very good solution to the exact scheduling problem by using one of the two continuous-time formulations. The proposed strategy is the following: i) solve the approximate problem by the discrete-time formulation (F1) to find the assignments of orders to machines. Although it is desirable to solve the problem to optimality, a feasible solution is enough to proceed to the next step; ii) solve a simplified version of the continuous-time model with multiple time grid (F3) or sequencing variables (F4), by considering in sets I_m and M_i, only the orders assigned to machine *m* and the machine assigned to order *i*, or only the variables $x_{i,m}$ with $i \in I_m$, respectively. Note that once the assignments are fixed, each machine is totally independent from the others so it is better to solve |M| single machine problems instead of a more complex parallel machines problem. For model (F3), the number of event points to use for solving the single machine problem, for machine *m*, is given by $|T|=|I_m|+1$. iii) if the complexity of one or more single machine problems in the previous step is still too much to handle, further reduce the complexity by also fixing the event point at which each order is processed (the first order allocated to a particular machine will start at the first event point, the second order at the second and so on, for model F3), or by fixing the sequencing variables $y_{i,i'}$ (F4). In this case, there are fewer degrees of freedom and hence the exact schedule corresponding to the near optimal solution found in the first step is generated.

Overall, the proposed strategy does not guarantee global optimality, but is capable of generating better solutions than those reported so far in the literature for two complex example problems, as will be seen in the next section.

5.4. Problem set 2: minimize total earliness

The data for the set of problems considered in this section was taken from Méndez & Cerdá (2003) and is given in Table 9. Notice that there are no release dates, meaning that all orders can start to be processed from the beginning of the time horizon (time zero). The complete problem and two subproblems will be solved in increasing order of complexity: problem S2A will consider the first 12 orders; problem S2B the first 29 orders and problem S2C the full set of orders (40). Note that a particular order can only be processed on a subset of the available machines, which allows us to use fewer variables and constraints. This is the reason why we consider a 40 order problem, which is more than the maximum number of orders considered when minimizing total cost, even after knowing from section 5.2 that the resulting problems are more difficult to solve.

Table 9. Data for second set of problems (S2A-S2C)

		$p_{i,m}$ (day)						$p_{i,m}$ (day)			
Order	d_i (day)	*M1*	*M2*	*M3*	*M4*	Order	d_i (day)	*M1*	*M2*	*M3*	*M4*
I1	15	1.718			1.424	*I21*	30	7.497		3.614	
I2	30	1.680			1.019	*I22*	20			0.864	
I3	22	1.787			1.048	*I23*	12			3.624	
I4	25			1.564	2.373	*I24*	30			2.667	4.230
I5	20			0.736	1.247	*I25*	17	6.132		3.448	5.132
I6	30	5.443			3.430	*I26*	20	4.004			1.987
I7	21	5.045		3.025	3.444	*I27*	11	6.590			4.167
I8	26			1.500	1.670	*I28*	30	5.680			3.465
I9	30			1.869	2.689	*I29*	25				4.516
I10	29		1.457			*I30*	26				2.384
I11	30		3.925		3.230	*I31*	22		1.744		1.593
I12	21		6.971	7.000	5.830	*I32*	18			2.698	3.884
I13	30	11.430			6.946	*I33*	15		2.322		
I14	25	2.812			1.757	*I34*	10	3.445		2.658	
I15	24	5.180			3.215	*I35*	10	3.660			2.780
I16	30	1.430			1.013	*I36*	14		2.433		
I17	30	4.654			3.266	*I37*	24		2.320	2.194	
I18	30		1.604			*I38*	16	2.545			2.080
I19	13		3.305		2.917	*I39*	22	2.210			
I20	19	2.604		1.074	1.830	*I40*	23		2.065		

5.4.1. Results of standalone models for problems S2A-S2C

The discrete-time formulation (F1) requires an exceedingly large number of time intervals to consider the exact problem data (30/0.001+1=30001). This is too much, so one needs to round the problem data as described in section 3.1. The smaller the interval length (δ), the more difficult it is to solve a particular problem. Thus, an increase in problem complexity (e.g. increase in the number of orders) should be compensated by an increase in interval length, in order to maintain problem tractability. For these set of problems we have chosen δ=0.01, 0.05 and 0.1 days for problems S2A, S2B and S2C, which means considering 3001, 601 and 301 time points, respectively. Despite the decrease in problem size, the computational effort increases significantly from S2A (7.35 s) to S2B (300 s) to S2C (3600 s), with the latter representing the maximum resource limit (integer solution within 2.63% of the optimum). The solutions from the discrete-time model (F1) are good upper bounds on the true optimal solution, with the quality of the approximation generally increasing with an increase in the number of time intervals. For S2A, the solution of 1.03 days is very close to the optimal solution found by the continuous-time models, 1.026 (see Table 10). For S2B and S2C the solutions are better than the best solutions found by the continuous-time models, and the solution of S2B is even better than the best solution reported in the literature, 62.377 days (see Méndez & Cerdá, 2003).

The constraint programming model (F5) like the discrete-time formulation (F1) uses integer data for the processing times. This means that we need to multiply the values in Table 9 by 1000 before we solve the problem in ILOG (the objective function is then divided by 1000, to find the total earliness). Problem S2A is solved in 3266 CPUs, which is already too much time. To overcome this problem we can use a smaller basis by multiplying the problem data by 100 and then rounding it to the next integer. Doing this ensures that the problem data is exactly the same as that considered in the discrete-time formulation (F1), thus allowing for a more valid comparison. The computational statistics in Table 10 clearly show that the CP model (F5) performs worse than model (F1). Since a significantly worse solution was found for S2B, there is no point on solving S2C by the CP model.

The uniform time grid continuous-time formulation (F2) can find the global optimal solution rather fast but due to the 100% relative integrality gap it is very difficult to prove optimality (about 1.5 hours of computational time). The other disadvantage is that a few problems needed to be solved to find the minimum number of event points (10) and the minimum value of Δt (5) to get to the optimal solution. Thus, like with the first set of problems, the uniform time grid continuous-time formulation has a poor performance and is only useful for small problems.

The multiple time grid continuous-time formulation (F3) has very good performance for S2A (1.1 s). A minimum number of 4 event points ($|I|/|M|+1$) is required to get a feasible solution of 2.457 days in 0.14 s. A single increase in the number of event points allows us to find a better solution (1.026 days), which we know is the global optimum. A further increase in the number of event points leads to a degenerate solution in 6.5 s. Contrary to the first set of problems, where they improved the computational performance, the time matching constraints (eq 19) were not included.

For problem S2B, the multiple time grid continuous-time formulation (F3) generates the worst solution of the four models tested. Note, that is very difficult to reduce the integrality gap (after one hour of computational time, the best possible solution, 1.151, was still very distant to the best integer solution found, 90.182). Furthermore, this solution is for 9 event points (the minimum number that ensures feasibility), a number that is probably insufficient to find the optimal solution (the results for machine M3 indicate the necessity of 11 event points, see Table 11).

The MILP with assignment and sequencing variables (F4) has the best performance of the three continuous-time models since it can always find reasonably good solutions. When compared to the multiple time grid formulation (F3), it has the advantage of considering the time grid implicitly (in the model constraints), which means that it only needs to be solved once. For S2A, the problem is solved in less than one third of the time despite using a larger number of variables and constraints, due to a faster increase in the objective of the relaxed model (i.e. it does a better job at reducing the gap between the relaxed model and the MILP). This seems to suggest that fixing a particular binary variable has a greater impact on model (F4) than in model (F3).

Table 10. Computational statistics for problems S2A-S2C

Problem	S2A					S2B				S2C	
Model	F1	F2	F3	F4	F5	F1	F3	F4	F5	F1	F4
|T|	3001	10	5			601	9			301	
Discrete variables	53955	875	120	157		23190	492	869		14189	1634
Single variables	66000	938	153	182	60	25595	558	928	145	15394	1715
Constraints	12017	445	265	512	84	2434	1042	2845	203	1245	5088
RMIP	1.03	0	0	0	-	60.183	0	0	-	125.49	0
Obj	1.03	1.026	1.026	1.026	1.03	61.35	90.182	64.441	68.05	133.9	143.78
CPU	7.35	5256	1.10	0.30	300	300	3600†	3600‡	3355*	3600♦	3600◊
Nodes	0	1508390	1283	22	-	892	210026	550451		29878	240332
Choice points					20931				12935204		

†Resource limit exceeded (best possible solution= 1.151, total tree size= 90 MB)
‡Resource limit exceeded (best possible solution= 23.137, total tree size= 263 MB)
*Computational time required to find the solution reported (solver terminated at 3600 CPUs)
♦Resource limit exceeded (best possible solution= 130.47, total tree size= 371 MB)
◊Resource limit exceeded (best possible solution= 26.711, total tree size= 204 MB)

5.4.2. Results of proposed strategy for problems S2B-S2C

In the previous section we have seen that the solutions from the discrete-time formulation (F1), which are valid upper bounds on the true optimal solution, are better than the best solutions found by the continuous-time formulation and the constraint programming model. Thus, the strategy presented in section 5.3 can be applied. First, we fix the assignments of orders to machines to the optimal assignments of the approximated problem (solution of F1). Then, we solve the four single machine problems to find an optimal solution to the real problem. Both the continuous-time model with multiple time grids and the one with sequencing variables (F4) are used to find out which one is the best.

The results for S2B (see Table 11) show that both (F3) and (F4) solve the single machine problems very efficiently with model (F4) showing a better performance for machine M3, which has more orders assigned to. This trend is confirmed for S2C (see Table 12). While the multiple time grid formulation (F3) performs similarly to (F4) for machines M1 and M2, the computational effort increases roughly by two orders of magnitude for the other two machines. This is a typical behaviour of time-grid(s) based continuous-time formulations (see Castro et al. 2004a): at high values of |T|, a single increase in the number of event points causes a one order of magnitude effect on computational effort. This significant difference in computational effort is even more interesting when one realizes that the MILPs generated by (F3) and (F4) are identical in terms of the number of discrete variables, constraints and integrality gap, which are classic performance indicators.

The optimal solution for problem S2B is given in Figure 5. The first thing to note is that there is not a limiting machine throughout the time horizon since all have waiting periods after they start to be used (M1 stops 0.955 days between orders 1 and 7 and 0.104 days between 7 and 14; M2 stops 1.026 days between orders 19 and 12 and 2.014 days between 12 and 11; M3 stops 0.664 days between orders 23 and 25; M4 stops 0.206 days between orders 27 and 29 and 0.065 days between orders 3 and 28). Overall, M3 and M4 can be viewed as more limiting machines since they stop for less time. Curiously, all orders that were assigned to M3 and M4 have the lowest processing time on these machines, which seems to indicate that the minimization of total processing time would also be an indirect way to achieve the goal of minimizing total earliness.

Pinto & Grossmann (1995) were the firsts to propose minimizing the total processing time to find the assignments of orders to machines for this same problem, before solving the sequencing problem. They found that with this objective the MILP problems resulting from their formulation became much easier to solve, due to their zero integrality gap. The same approach was tried with the multiple time grid continuous-time formulation (F3) and we also found that by changing the objective function, which allows us to neglect the hard constraints 23 and 24, the resulting MILP becomes much easier to solve. Unfortunately, and although we were able to find reasonable solutions, they were much worse than those found by using the discrete-time formulation first. One order in the optimal schedule of Figure 5 that goes completely against the objective of minimizing total processing time is order 13, which is assigned to M1, where it has a processing time of 11.43 days (see Table 9), while in M4 it would take only 6.946 days to complete.

Preordering of orders based on their due dates is a heuristic that as also been used to decrease the complexity of the single stage parallel machine problem. Pinto & Grossmann (1995) and Ierapetritou et al. (1999) used the earliest due date (EDD) method on this problem to find solutions of 82.202 and 94.814 days, respectively. Mendéz & Cerdá (2003) used the increasing slack times (MST) rule to simplify their MILP formulation and found a better solution (62.377 days), which was further improved by using a very efficient rescheduling MILP to re-sequence orders assigned to a particular machine. Models (F3) and (F4), when applied to the single stage problem, can be viewed as more general formulations than their re-sequencing problem. The schedule of Figure 5, corresponds to a solution that has a total earliness of 59.896 days, 2.481 days better (almost 4%) than the best solution found by Méndez & Cerdá (2003), clearly showing that is better to use a discrete-time formulation with approximate problem data instead of a heuristic procedure. Order 13 is again a good example why the preordering heuristics fail: it is only the 19th order according to the EDD rule and the 15th order according to the MST rule but is the 4th order to be completed.

The best solution found for problem S2C has a total earliness of 126.949 days and is shown in Figure 6. It is 4.4% better than the solution reported by Méndez & Cerdá (2003), which has a total earliness of 132.727. When compared to Figure 5, the machines have fewer waiting periods (M1 stops for 0.263 days between orders 38 and 39; M2 stops for 1.871 days between orders 33 and 37 and 0.014 days between orders 40 and 11; and M3 and M4 never stop) and maintain most of the assignments for the first 29 orders. Of the 40 orders, only 6 orders are delivered exactly at their due dates (orders 2, 9, 16, 18, 38 and 40).

Table 11. Computational statistics for single machine problems (problem S2B)

Model	|T|	discrete variables	single variables	constraints	RMIP	MIP	CPUs	nodes
F1+F3	-	-	-	-	-	59.896	303	-
M1	7	42	63	98	0	19.232	0.22	93
M2	6	30	48	72	0	3.665	0.09	2
M3	11	110	143	242	0	16.907	2.5	3121
M4	9	72	99	162	0	20.092	0.55	583
F1+F4	-	-	-	-	-	59.896	301	-
M1	-	36	49	74	0	19.232	0.07	2
M2	-	25	36	52	0	3.665	0.10	0
M3	-	100	121	202	0	16.907	0.24	202
M4	-	64	81	130	0	20.092	0.14	97

Table 12. Computational statistics for single machine problems (problem S2C)

Model	|T|	discrete Variables	single variables	constraints	RMIP	MIP	CPUs	nodes
F1+F3	-	-	-	-	-	126.949	4798	-
M1	8	56	80	128	0	30.849	0.24	149
M2	10	90	120	200	0	21.616	0.80	870
M3	14	182	224	392	0	42.880	1178	552483
M4	12	132	168	288	0	31.604	13.3	9932
F1+F4						126.949	3612	-
M1	-	49	64	100	0	30.849	0.24	28
M2	-	81	100	164	0	21.616	0.14	88
M3	-	169	196	340	0	42.880	8.8	34145
M4	-	121	144	244	0	31.604	0.23	612

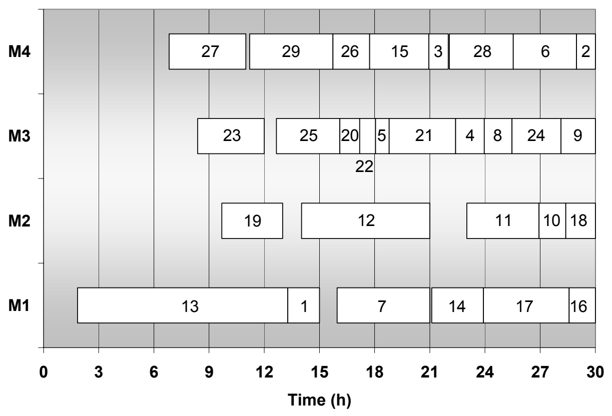

Figure 5. Optimal solution for problem S2B

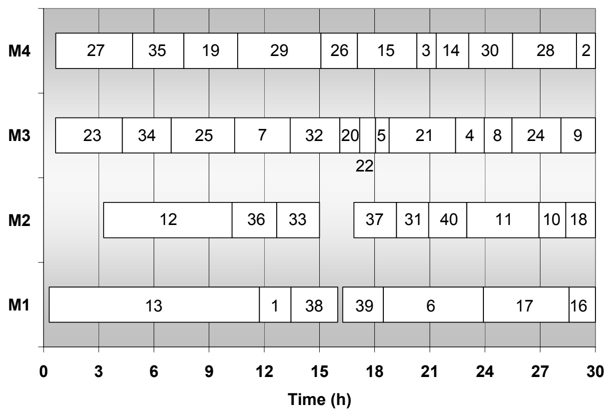

Figure 6. Optimal solution for problem S2C

6. Conclusions

This paper presents a new continuous-time formulation for the short-term scheduling of single-stage parallel machine plants. It can be viewed as an improvement from the general continuous-time formulation for multipurpose plants of Castro et al. (2004), which proved to be extremely limited for the specific type of problem considered. Multiple time grids are used, one for each equipment resource, instead of a single time grid, to take advantage of the lack of common resources like manpower or utilities, and intermediate materials. This, allows us to consider each machine independently, which in turn makes it possible to restrict the duration of all batch tasks to a single time interval. The advantage is that both the number of event points required to find the optimal solution as well as the integrality gap, two commonly used performance indicators for MILPs and for continuous-time formulations, are greatly reduced.

Other novel features presented in this paper, concerning the continuous-time formulations, are the introduction of hard constraints to model release and due dates as well as a set of constraints to determine the release dates of the final products. Although restricted here for single stage problems, these can easily be adapted for the general multipurpose plant. Also, some characteristics of the general multipurpose formulation of Castro et al. (2004) such as its ability to handle variable duration tasks can easily be implemented on the single stage, multiple time grid formulation, but this is beyond the scope of this paper.

The other goal of the paper has been to provide a critical review of other existing approaches that are able to solve this type of scheduling problem. These included a RTN-based discrete-time formulation (Pantelides, 1994), a continuous-time formulation that relies on sequencing variables instead of event points and a constraint programming formulation (Jain & Grossmann, 2001), and the hybrid MILP/CP model of Maravelias & Grossmann, 2004b. The analysis was performed for two widely used objective functions: minimization of total cost and minimization of total earliness. A total of 15 example problems were solved with the aim of finding the limit of applicability of all six approaches. The results were very conclusive and allowed us to arrive at the following conclusions.

The discrete-time RTN formulation was shown to be the best formulation. It has a very consistent performance for different problem sizes and for the two different objective functions tested. As is well known, its most important limitation is that it cannot handle variable duration tasks. Another disadvantage is that it may need to use a large number of time intervals to consider the exact problem data. We disagree with this argument because fewer time intervals can always be used if the problem data is rounded to the next integer multiple of the interval length, to decrease the size and complexity of the problem. In that case, the solution obtained will be an upper bound on the true global optimum. This important feature of the discrete-time formulation has been used to propose an efficient optimization strategy for total earliness minimization. It consists on finding the assignments of orders to machines by solving the parallel machine single stage scheduling problem with the discrete-time formulation and subsequent solution of |M| single machine problems with one of the two most efficient continuous-time formulations. This optimization strategy was shown to be better than preordering heuristics as we were able to find better solutions than those reported in the literature for problems S2B and S2C.

The two problem specific continuous-time formulations (F3) and (F4) follow the discrete-time formulation in terms of computational performance. While the multiple time grid formulation performs better for the minimization of total cost, the one with sequencing variables performs better for the minimization of total earliness both for the parallel and single machine problems. The worsening in the performance of the multiple time grid continuous-time formulation when going from cost minimization to earliness minimization is due to the addition of two sets of big-M constraints, whereas the continuous-time formulation with sequencing variables only requires minor changes. The latter formulation is also easily adapted in order to deal with sequence dependent changeovers, while the former requires additional sets of variables and constraints to model the cleaning tasks and the different equipment states, like all RTN-based formulations.

Finally, there are the constraint programming and the hybrid MILP/CP approach by Jain & Grossmann (2001). The former always gives a reasonably good solution, even though it may be impossible to find the global optimal solution for medium sized problems. The latter has the best performance of all 6 models tested for small to medium sized problems, but it failed to find a solution for the larger problem since the assignment part of the hybrid model (the simplified MILP) becomes intractable like the complete MILP from which it originates. Another disadvantage of the hybrid approach is that the only feasible solution it generates is the global optimum so there are no intermediate feasible solutions. Furthermore, it is only efficient when the objective function depends solely on the assignment variables, e.g. total cost minimization.

Acknowledgments

The authors gratefully acknowledge financial support from Fundação Calouste Gulbenkian.

Nomenclature

Sets/Indices
$I/i, i'$= process orders
I_m= orders to be processed on machine m
$I_{t,m}$= orders that can start on time point t on machine m
M/m= process equipments (machines)
M_i=machines that can process order i
$T/t, t',t''$=Points of the time grid
$T_{i,m}$=Time points where order i can start to be processed on machine m

Parameters
$c_{i,m}$=cost of processing order i on machine m
d_i=due date of order i
H=time horizon
$p_{i,m}$=processing time of order i on machine m
r_i=release date of order i
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
$N_{i,m,t}$=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
T_t=absolute time of event point t

References

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). Constrained-based Scheduling: Applying Constraint Programming to Scheduling Problems. Kluwer Academic Publishers.

Bockmayr, A., & Pisaruk, N. (2003). Detecting Infeasibility and Generating Cuts for MIP using CP. Proceedings CPAIOR'03, 24.

Castro, P.M., Barbosa-Póvoa, A.P., & Novais, A.Q. (2005). Simultaneous Design and Scheduling of Multipurpose Plants using RTN-based Continuous-time Formulations. Ind. Eng. Chem. Res., 343.

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

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

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

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

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

Ierapetritou, M.G., Hené, T.S. & Floudas, C.A. (1999). Effective Continuous-Time Formulation for Short-Term Scheduling. 3 Multiple Intermediate Due Dates. Ind. Eng. Chem. Res., 38, 3446.

Jain, V., & Grossmann, I.E. (2001). Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. Informs Journal on Computing, 13, No. 4, 258.

Janak, S.L., Lin, X.; & Floudas, C.A. (2004). 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., 43, 2516.

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

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

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

Maravelias, C.T., & Grossmann, I.E. (2004a). A Hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. Comp. Chem. Eng., 28, 1921.

Maravelias, C.T., & Grossmann, I.E. (2004b). Using MILP and CP for the Scheduling of Batch Chemical Processes. Proceedings CPAIOR'04, 1.

Méndez, C., & Cerdá, J. (2000). Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. Comp. Chem. Eng., 24, 369.

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

Méndez, C., & Cerdá, J. (2003). Dynamic scheduling in multiproduct batch plants. Comp. Chem. Eng., 27, 1247.

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

Pinedo, M. (2001). Scheduling: Theory, Algorithms and Systems. Prentice Hall.

Pinto, J.M., & Grossmann, I.E. (1995). A Continuous Time Mixed Integer Linear Porgramming Model for Short Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res., 34, 3037.

Sadykov, R., & Wolsey, L. (2005). Integer Programming and Constraint Programming in Solving a Multi-Machine Assignment Scheduling Problem with Deadlines and Release Dates. To appear in INFORMS Journal on Computing.

Van Hentenryck, P. (1999). The OPL Optimization Programming Language. MIT Press, Cambridge, MA.