Discrete and continuous-time formulations for dealing with breaks: preemptive and non-preemptive scheduling

Pedro M. Castro[a,*] *, Iiro Harjunkoski*[b] *and Ignacio E. Grossmann*[c]

[a] Centro de Matemática Aplicações Fundamentais e Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal.

[b] ABB Corporate Research, Wallstadter Strasse 59, 68526 Ladenburg, Baden-Württemberg, Germany

[c] Center for Advanced Process Decision-Making, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

This paper presents new mixed-integer linear programming (MILP) approaches for handling preemption both in discrete and continuous-time scheduling formulations. Preemption refers to the possibility of interrupting the execution of a task when encountering a break, located at a pre-defined time window, assuming that the task continues immediately after the end of the break. We rely on Generalized Disjunctive Programming to derive the constraints for the continuous-time formulations and on a compact convex hull reformulation to make them computationally efficient. We investigate both the general precedence time representation and multiple time grids concepts. Generalization of the discrete-time formulation is simpler, involving a change in the model parameters. Validation and comparison of the mathematical formulations is done through the solution of sixteen benchmark

[*] Corresponding author: pmcastro@fc.ul.pt

problems, involving instances with one to three breaks. The results show that discrete-time formulation is computationally more effective for flexible flowshops while the continuous-time approach prevails for single stage plants with parallel units.

Keywords: *Scheduling, Combinatorial Optimization, Integer Programming, Multiproduct batch plants, Resource-Task Network.*

1. Introduction

Preemptive scheduling has rarely been considered in the context of batch scheduling problems (Pei et al., 2015). In preemptive scheduling, assuming it is operationally feasible, a job may be interrupted and resumed later, even on a different machine, whereas interruptions are not allowed in nonpreemptive scheduling. Preemption can lead to a more efficient use of the production resources, e.g. lower makespan, but there are also cases where performance is not improved. Using the notation of Graham et al. (1979), Brucker et al. (2003) identify the single stage parallel machine scheduling problems for which preemption does not improve the objective value. A key assumption, aside from being operationally feasible, is that all data are assumed to be integer, so all jobs can be split into as many parts as the value of their processing times. It is thus not surprising that the authors found that allowing preemption may turn a polynomially solvable problem into a NP-hard one.

Batsyn et al. (2014) propose a Boolean Linear Programming (an Integer Program with 0-1 variables) model for the preemptive single machine scheduling problem with release dates and integer processing times. Each job is divided into unit parts, while the time horizon of interest is divided into slots of size one, in a number that equals the sum of the processing times. In this discrete-time formulation, the binary variables assign each part of a job to exactly one time slot. To cope with the difficulty of CPLEX to handle problems with 25 jobs, the authors proposed a high-quality (optimality gap <0.1% for the problems that could be solved to optimality) heuristic method capable of handling problems with 5000 jobs in less than one hour.

Pei et al. (2015) extend the work of Batsyn et al. (2014) by considering a two-stage supply chain scheduling problem, with a production stage and a single vehicle in the transportation stage. The first stage features a single batching machine capable of handling multiple jobs in a serial fashion (one batch) up to the given capacity. Each job must be assigned to exactly one batch. The processing of a batch can be interrupted by another batch and resumed later (preemption). Once a batch is finished, it will occupy the machine until the transportation vehicle returns to take it. The authors propose an $O(n \log n)$ exact algorithm to address the problem where the n jobs are available at time zero, and a heuristic with the same time complexity to address the case with different release dates.

Preemption in a context of a scheduling problem with precedence constraints between activities, also appears in the multi-mode resource constraint project scheduling problem (Peteghem and Vanhoucke, 2010). The authors propose a genetic algorithm and evaluate the impact of preemption in the quality of the schedule. Using data instances from the PSPLIB, they get average makespan improvements of about 0.5% for cases with nonrenewable resources, but also obtained runs with worse solutions. This was justified by the larger project networks that are required when introducing activity preemption.

A closely related problem is the multi-skilled resource investment project scheduling problem (Javanmard et al., 2017). Different skills are required to accomplish project activities and the limits on investment of renewable resources are considered decision variables. The authors propose two mixed-integer linear programming (MILP) formulations. In the first, the assigned level of skills was considered fixed, while in the second, the levels could be altered after preemption. Not surprisingly, the second model led to a lower total cost. Two meta-heuristic algorithms were also proposed, with the genetic algorithm outperforming its particle swarm optimization counterpart. They were first validated using PSPLIB instances with up to 30 activities, leading to average optimality gaps below 0.04% (compared to the optimal solutions from the MILP models), and then used to tackle the instances with 90 activities in roughly 1000 CPUs.

Preemption is just one of many aspects that may be encountered when facing a real-life scheduling problem. The multitude and complexity of scheduling problems is increasing and with the growing size and scope it is important to enable various collaboration schemes (Harjunkoski et al., 2014). As different types of methods may need to collaborate, also stemming from various industries and companies, it is important to have a common basis for exchanging information (Harjunkoski, 2016). This is currently strongly driven by the digitalization activities such as Industry 4.0 that promote crossing the traditional solution boundaries.

The standard ISA-95 can support the collaboration and integration of methods by offering a neutral and uniform data structure. The ISA-95 standard (ANSI/ISA-95.00.03-2005), was originally created to act as an interface between business- and control systems. Therefore, it defines most of the required data fields needed for scheduling and furthermore offers an Extended Markup Language (XML)-based implementation, which is extremely useful in a software-context as most programming languages have built-in support for XML. The actual implementation of ISA-95 is realized through XML-schemas and called B2MML (business to manufacturing markup language). The standard can also be flexibly extended to meet future challenges and requirements (e.g. allowing for preemption). The main benefit of using a standard is to remove problems in agreeing on the data design between two entities. The ISA-95 standard has already been widely adopted and accepted by many industrial solutions and a detailed description of its functions is described in part 1 of the standard (ANSI/ISA-S95, 2000).

Apart from standardized data-interfaces, another challenge is how to effectively solve the increasing growing number of heterogenous scheduling problems. The current work was driven by the search for a truly generic scheduling formulation for process plants. Previous work with the Resource-Task Network (RTN) discrete-time formulation of Pantelides (1994) have led us to choose this approach as a basis for further development. Reasons for this decision include: (i) uniform treatment of different types of resources (e.g. equipment, materials, utilities, manpower, etc.); (ii)

modeling paradigm easily understood by business stakeholders (e.g. Wassick and Ferrio (2011) from the Dow Chemical Company); (iii) flexible approach, easily modified when new information becomes available; (iv) the discrete-time model is better at finding good solutions in short computational time, a requirement for a successful commercial software.

In this paper, we extend the RTN discrete-time formulation to preemptive tasks. We also develop new constraints for handling preemption with continuous-time formulations, enabling a comparison between the two types of time representation. It should be highlighted that contrary to the articles discussed in the beginning of this section, preemption is not something that can occur throughout the time horizon of interest. Instead, it is a possibility when a task encounters break periods occurring at predefined time windows (e.g. preventive maintenance of equipment, working shifts, etc.). On the other hand, the current paper is not restricted to integer processing times.

2. Preemptive vs. non-preemptive scheduling

Let br be a break period occurring in a given time window $[b_{br}^{L}, b_{br}^{U}]$ of the scheduling horizon. In most scheduling models, a non-preemptive mode of operation is assumed when facing a break, meaning that a task i can either be completely executed before the start of the break, or begin only after the end of the break. Let Ts_i and Tf_i be non-negative variables representing the start and end times of task i. The two possibilities can be formulated as an exclusive disjunction (Balas, 1979; Raman and Grossmann, 1994), as can be seen in Fig. 1 (Castro et al., 2014).

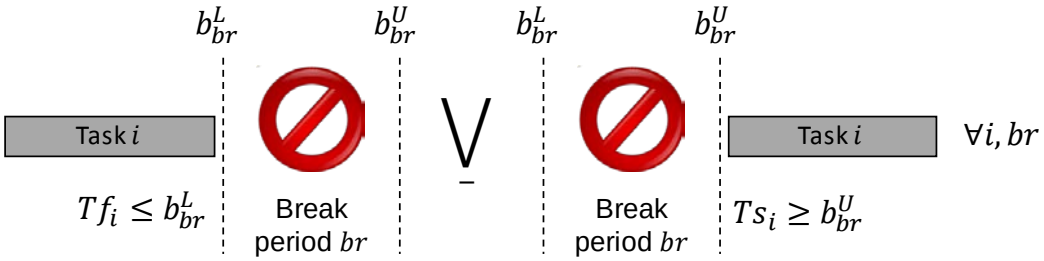

Fig. 1. In non-preemptive scheduling, a task can either end before or start after a break.

The goal of preemptive scheduling is to make the most of the available time, by allowing part of the task to be executed before the break and the remaining part after the break. Assume that the normal (non-preemptive) duration of task i is given by parameter p_i and let the non-negative variable $P_{i,br}$ hold the extended duration due to break br. If the optimization decides to execute the task entirely before or after br, as in Fig. 1, $P_{i,br}$ will be equal to zero. Otherwise, if the optimization decides to split the task, the extended duration will match the duration of the break period, $P_{i,br} = b_{br}^U - b_{br}^L$, as seen in Fig. 2. Equation (1) is valid in both cases and computes the total duration of the task as the sum of the processing time p_i plus the length of all break periods that the task crosses.

$$Tf_i - Ts_i = p_i + \sum_{br} P_{i,br} \quad \forall i \tag{1}$$

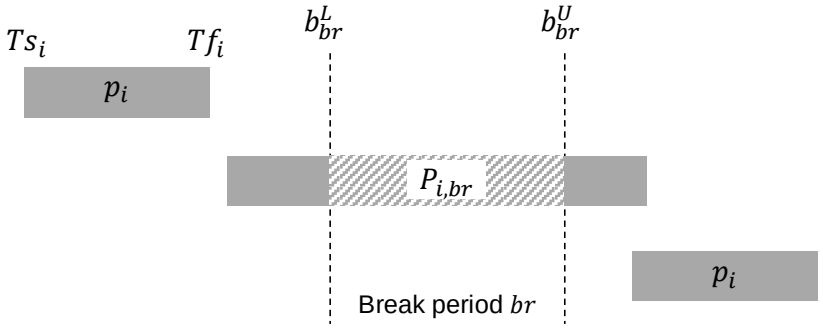

Fig. 2. In preemptive scheduling, it is possible to extend the duration of a task so that it can start before and end after a break period.

3. Preemption with continuous-time formulations

We now propose constraints to handle preemption with continuous-time formulations. As highlighted in the review paper by Harjunkoski et al. (2014), these can rely on the concept of precedence, or use a single or multiple time grids to keep track of activities.

We start by considering the general precedence formulation of Méndez and Cerdá (2002) for multistage batch plants. In this formulation, there are already starting and ending time variables for the orders in the different stages, making it easier to develop the preemption constraints. We then consider the multiple time grid formulation of Castro and Grossmann (2006, 2012) for single stage plants with parallel units, in which the start and end times are known only implicitly, from the timing

of event points that compose the grid of each equipment unit and from the assignments of orders to units.

Note that while problems from single stage plants can also be addressed by the general precedence formulation (such a topology is a special case of a flexible flowshop), they can be solved orders of magnitude faster by the multiple time grid formulation (Castro and Grossmann, 2012).

3.1. New constraints for handling preemption with a general precedence formulation

Let the three alternative decisions in Fig. 2 be identified through binary variables: $Y^B_{i,br}(=1)$, if task i completes execution before the start of break period br; $Y^D_{i,br}$, if the task is executed during the break; $Y^A_{i,br}$, if it starts after the end of break period br. As can be seen in the disjunction in Eq. (2), each decision is associated to three sets of constraints involving the continuous variables $P_{i,br}$, Ts_i and Tf_i, where parameters ts_i^L, tf_i^L, ts_i^U and tf_i^U represent the lower (superscript L) and upper bounds (U) on the starting and ending times of tasks.

$$\begin{bmatrix} Y^B_{i,br} \\ P_{i,br} = 0 \\ ts_i^L \le Ts_i \le b_{br}^L \\ tf_i^L \le Tf_i \le b_{br}^L \end{bmatrix} \veebar \begin{bmatrix} Y^D_{i,br} \\ P_{i,br} = b_{br}^U - b_{br}^L \\ ts_i^L \le Ts_i \le b_{br}^L \\ b_{br}^U \le Tf_i \le tf_i^U \end{bmatrix} \veebar \begin{bmatrix} Y^A_{i,br} \\ P_{i,br} = 0 \\ b_{br}^U \le Ts_i \le ts_i^U \\ b_{br}^U \le Tf_i \le tf_i^U \end{bmatrix} \forall i, br \tag{2}$$

The logic proposition implicit in the exclusive disjunction in Eq. (2) is that exactly one binary must be active, which is translated into Eq. (3).

$$Y^B_{i,br} + Y^D_{i,br} + Y^A_{i,br} = 1 \ \forall i, br \tag{3}$$

For the three rows of constraints inside the disjunctive terms in Eq. (2), we use the hull relaxation reformulation (Balas, 1985; Raman and Grossmann, 1994). Taking the first row as an example, it involves defining new sets of disaggregated variables (one for each term): $\hat{P}^B_{i,br}$, $\hat{P}^D_{i,br}$ and $\hat{P}^A_{i,br}$. These

are related to the original variable $P_{i,br}$ through Eq. (4), and to the corresponding binary variable through Eqs. (5)-(7).

$$P_{i,br} = \hat{P}^B_{i,br} + \hat{P}^D_{i,br} + \hat{P}^A_{i,br} \; \forall i, br \tag{4}$$

$$\hat{P}^B_{i,br} = 0 \cdot Y^B_{i,br} \; \forall i, br \tag{5}$$

$$\hat{P}^D_{i,br} = (b^U_{br} - b^L_{br}) \cdot Y^D_{i,br} \; \forall i, br \tag{6}$$

$$\hat{P}^A_{i,br} = 0 \cdot Y^A_{i,br} \; \forall i, br \tag{7}$$

Ideally, we want to eliminate the disaggregated variables to avoid increasing the problem size. In this case, this is possible since each constraint inside a disjunctive term involves a single variable. We simply need to replace Eqs. (5)-(7) in Eq. (4) to generate a sharp/compact formulation (Jeroslow and Lowe, 1984; Castro and Grossmann, 2012). The result can be seen in Eq. (8).

$$P_{i,br} = (b^U_{br} - b^L_{br}) \cdot Y^D_{i,br} \; \forall i, br \tag{8}$$

Applying the same procedure to the second and third row of constraints in Eq. (2), leads us to Eqs. (9)-(10).

$$ts^L_i \cdot (Y^B_{i,br} + Y^D_{i,br}) + b^U_{br} Y^A_{i,br} \leq Ts_i \leq b^L_{br} \cdot (Y^B_{i,br} + Y^D_{i,br}) + ts^U_i Y^A_{i,br} \; \forall i, br \tag{9}$$

$$tf^L_i \cdot Y^B_{i,br} + b^U_{br} \cdot (Y^D_{i,br} + Y^A_{i,br}) \leq Tf_i \leq b^L_{br} \cdot Y^B_{i,br} + tf^U_i \cdot (Y^D_{i,br} + Y^A_{i,br}) \; \forall i, br \tag{10}$$

In summary, Eqs. (1), (3) and (8)-(10), are required to represent the possible preemption.

3.1.1. Non-preemption as a special case

If preemption is not allowed when encountering a break period (typical mode of operation), the two alternatives in Fig. 1 can be handled simply by setting $Y^D_{i,br} = 0 \; \forall i, br$.

3.1.2. Link to standard formulation for flexible flowshops

We now recall the general precedence formulation of Méndez and Cerdá (2002), which is suitable for flexible flowshops and can easily be adapted to flexible jobshops (see Harjunkoski et al., 2014). In a flexible flowshop (Fig. 3), each order *j* needs to go through the same sequence of stages k to

reach the condition of final product, and on each stage, there are a set of parallel machines $m \in M_k$ to choose from. Assuming that the parallel units of a stage are equivalent, a characteristic of the benchmark problem in section 5, the connection with the preemptive constraints given above can be made by noting that a task i is equivalent to a pair (j, k).

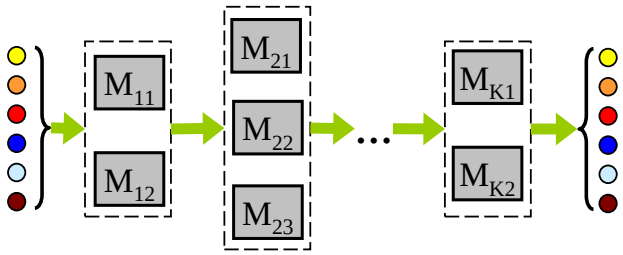

Fig. 3. A flexible flowshop with two parallel machines on the first and last stages ($M_1 = \{M_{11}, M_{12}\}$, $M_K = \{M_{K1}, M_{K2}\}$), and three machines on the second stage ($M_2 = \{M_{21}, M_{22}, M_{22}\}$).

Let binary variables $X_{j,m,k} = 1$ if order j is assigned to machine m in stage k, and $Y_{j,j',k} = 1$ if order j globally precedes order j' in stage k. Equation (11) states that exactly one machine m is assigned to the execution of order j in stage k. Equations (12)-(13) guarantee that orders allocated to the same machine cannot overlap. They can be viewed as super big-M constraints, involving three different binary variables, which are known to yield poor linear relaxations. Relating the execution of an order in consecutive stages is much simpler. Equation (14) states that the start time of order j in stage $k+1$ must be greater than its ending time in stage k.

$$\sum_{m \in M_k} X_{j,m,k} = 1 \; \forall j, k \tag{11}$$

$$Tf_{j,k} \leq Ts_{j',k} + tf^U_{j,k}\left(3 - Y_{j,j',k} - X_{j,m,k} - X_{j',m,k}\right) \; \forall j, j' > j, k, m \in M_k \tag{12}$$

$$Tf_{j',k} \leq Ts_{j,k} + tf^U_{j',k}\left(2 + Y_{j,j',k} - X_{j,m,k} - X_{j',m,k}\right) \; \forall j, j' > j, k, m \in M_k \tag{13}$$

$$Ts_{j,k+1} \geq Tf_{j,k} \; \forall j, k < K \tag{14}$$

3.1.3. Objective function

The objective function considered in this work is to minimize the makespan, MS, the maximum of the last-stage K finishing times, as seen in Eqs. (15)-(16).

$$\min MS \tag{15}$$

$$MS \geq Tf_{j,K} \ \forall j \tag{16}$$

Assuming that the makespan occurs after the end of the last break period, one can use the tightening constraint in Eq. (17). It states that the makespan must be greater than the sum of the processing times of all orders executed in machine m (second term on the left-hand side) plus the minimum processing time over all stages before and after k, plus the total duration of break periods (the new term compared to the constraint in Castro and Grossmann, 2005).

$$\min_j(\sum_{k'<k} p_{j,k'}) + \sum_j p_{j,k} X_{j,m,k} + \min_j(\sum_{k'>k} p_{j,k'}) + \sum_{br}(b_{br}^U - b_{br}^L) \leq MS \ \forall m,k \tag{17}$$

3.2. New constraints for handling preemption with a multiple time grid formulation

We consider a multiple time grid formulation with m grids, one for every unit, comprising an equal number of time slots $|T|$. Let $Ts_{m,t}$ and $Tf_{m,t}$ be continuous variables representing the starting and ending times of slot t in unit m, respectively, which are bounded between zero and the time horizon h. Equation (18) states that the duration of the unit-slot must be greater than the processing time plus the length of all break periods included in the slot (note that while the equality can also be used, preliminary tests have shown a worse computational performance). Compared to Eq. (1), the first term on the right-hand side is now a variable ($P_{m,t}^V$) rather than a parameter, since the processing time will depend on the order assigned to the slot.

$$Tf_{m,t} - Ts_{m,t} \geq P_{m,t}^V + \sum_{br} P_{m,t,br} \ \ \forall m,t \tag{18}$$

Eq. (19) states that the starting time of slot $t+1$ must be greater than the ending time of slot t, while the makespan is the maximum, over all units, of the ending times of the last slot, as seen in Eq. (20).

$$Ts_{m,t+1} \geq Tf_{m,t} \ \forall m, t < |T| \tag{19}$$

$$MS \geq Tf_{m,|T|} \ \forall m \tag{20}$$

Contrary to section 3.1, we now assume that breaks are unit-dependent, representing for instance periods of preventive maintenance. Therefore, the index m is added to the period bounds (e.g. $b^L_{m,br}$). The three alternatives for the location of slot (m,t) with respect to break period (m,br) are modeled through the disjunction in Eq. (21). Besides the changes to the domain and variables indices, the only difference are the weaker bounds for the timing variables. This is because we need to consider a wide variety of cases, ranging from no orders assigned to unit m ($Ts_{m,t} = Tf_{m,t} = 0\ \forall t$), to idle slots t^* towards the end of the time horizon h ($Ts_{m,t^*} = Tf_{m,t^*} \leq h$).

$$\begin{bmatrix} Y^B_{m,t,br} \\ P_{m,t,br} = 0 \\ 0 \leq Ts_{m,t} \leq b^L_{m,br} \\ 0 \leq Tf_{m,t} \leq b^L_{m,br} \end{bmatrix} \veebar \begin{bmatrix} Y^D_{m,t,br} \\ P_{m,t,br} = b^U_{m,br} - b^L_{m,br} \\ 0 \leq Ts_{m,t} \leq b^L_{m,br} \\ b^U_{m,br} \leq Tf_{m,t} \leq h \end{bmatrix} \veebar \begin{bmatrix} Y^A_{m,t,br} \\ P_{m,t,br} = 0 \\ b^U_{m,br} \leq Ts_{m,t} \leq h \\ b^U_{m,br} \leq Tf_{m,t} \leq h \end{bmatrix} \forall m,t,br \tag{21}$$

The hull reformulation of Eq. (21) leads to Eqs. (22)-(25).

$$Y^B_{m,t,br} + Y^D_{m,t,br} + Y^A_{m,t,br} = 1\ \forall m,t,br \tag{22}$$

$$P_{m,t,br} = \left(b^U_{m,br} - b^L_{m,br}\right) \cdot Y^D_{m,t,br}\ \forall m,t,br \tag{23}$$

$$b^U_{m,br} Y^A_{m,t,br} \leq Ts_{m,t} \leq b^L_{m,br} \cdot \left(Y^B_{m,t,br} + Y^D_{m,t,br}\right) + h \cdot Y^A_{m,t,br}\ \forall m,t,br \tag{24}$$

$$b^U_{m,br} \cdot \left(Y^D_{m,t,br} + Y^A_{m,t,br}\right) \leq Tf_{m,t} \leq b^L_{m,br} Y^B_{m,t,br} + h \cdot \left(Y^D_{m,t,br} + Y^A_{m,t,br}\right)\ \forall m,t,br \tag{25}$$

Similarly to section 3.1.1, non-preemption can be handled simply by setting $Y^D_{m,t,br} = 0$.

3.2.1. Link to standard formulation for single stage plants with parallel units

To compute the non-preemptive processing time of slot (m,t), we need to know which order is assigned to it. The exclusive disjunction in Eq. (26) states that there can be at most one order assigned to the slot. If it is order j ($X_{j,m,t} = 1$), then the slot processing time $P^V_{m,t}$ must match the order processing time $p_{j,m}$, the start time must be greater than release date r_j and the end time must be lower than due date d_j (see also illustration in Fig. 4). If no orders are assigned to the slot ($X^{free}_{m,t} =$

1), then the processing time is equal to zero and the timing constraints are relaxed up to the horizon time h.

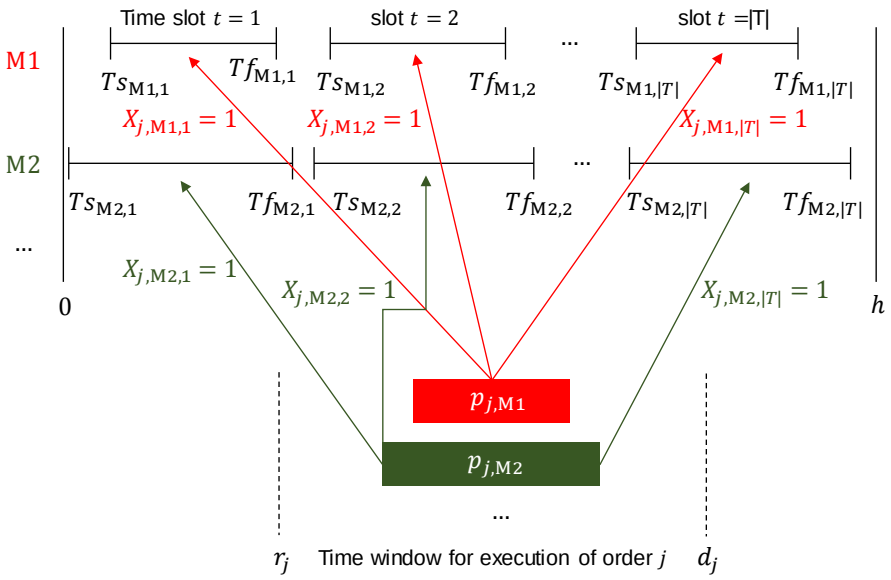

Fig. 4. Multiple time grid formulation for single stage plants. Each order j must be assigned a unit m and exactly one time slot t must be chosen. The time window for execution is the interval between its release r_j and due date d_j and processing times are unit-dependent.

$$\bigvee_{j}\begin{bmatrix} X_{j,m,t} \\ P^V_{m,t} = p_{j,m} \\ Ts_{m,t} \geq r_j \\ Tf_{m,t} \leq d_j \end{bmatrix} \bigvee \begin{bmatrix} X^{free}_{m,t} \\ P^V_{m,t} = 0 \\ Ts_{m,t} \geq 0 \\ Tf_{m,t} \leq h \end{bmatrix} \quad \forall m, t \tag{26}$$

Equation (27) reduces solution symmetry by moving idle slots to the right of active slots. Equation (28) then ensures that every order j is assigned to exactly one slot.

$$X^{free}_{m,t} \Longrightarrow X^{free}_{m,t+1} \; \forall m, t < |T| \tag{27}$$

$$\bigvee_{m}\bigvee_{t} X_{j,m,t} \quad \forall j \tag{28}$$

The reformulation of the logic propositions in Eqs. (26)-(28) leads to the Eqs. (29)-(31), which involve only binary variables.

$$\sum_j X_{j,m,t} + X_{m,t}^{free} = 1 \; \forall m, t \tag{29}$$

$$X_{m,t}^{free} \geq X_{m,t+1}^{free} \; \forall m, t < |T| \tag{30}$$

$$\sum_m \sum_t X_{j,m,t} = 1 \; \forall j \tag{31}$$

The sharp/compact hull reformulation of the constraints inside the disjunction in Eq. (26) generates Eqs. (32)-(35), which involve both binary and continuous variables.

$$P_{m,t}^V = \sum_j p_{j,m} X_{j,m,t} \; \forall m, t \tag{32}$$

$$Ts_{m,t} \geq \sum_j r_j X_{j,m,t} \; \forall m, t \tag{33}$$

$$Tf_{m,t} \leq \sum_j d_j X_{j,m,t} + h \cdot X_{m,t}^{free} \; \forall m, t \tag{34}$$

4. Preemption with a discrete-time formulation

In discrete-time formulations, the time horizon of interest is typically divided into uniform time slots of duration δ (Harjunkoski et al., 2014). To keep the model tractable, it is often needed to approximate the processing times p_i (in minutes or hours) to multiples of δ. More specifically, we compute the duration of task i (in number of time slots in the grid) by rounding up, i.e. $\tau_i = \lceil p_i/\delta \rceil$. All other timing parameters can be converted in a similar manner. In particular, b_{br}^L and b_{br}^U now represent the slots where break br starts and production can resume, respectively.

Let binary variable $N_{i,t} = 1$ indicate that task i starts to be executed at slot t and subset T_i hold the slots where task i can start. Break periods help to reduce the model size by decreasing the elements in T_i, and thus restricting the domain of variables $N_{i,t}$. As an example, consider a task spanning over $\tau_i = 5$ time slots executed in the vicinity of a four-interval break ($b_{br}^L = 9, b_{br}^U = 13$). In non-preemptive scheduling, the domain for task execution is reduced by more than half compared to the case of no break, to $T_i = \{1,2,3,4,13,14, \ldots\}$, as seen in Fig. 5.

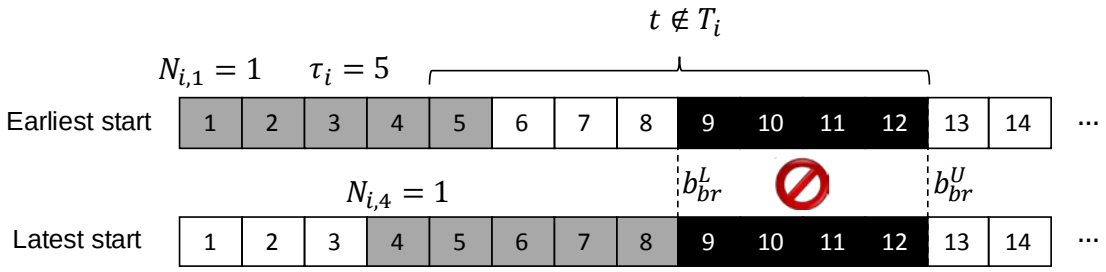

Fig. 5. In non-preemptive scheduling, there is a significant reduction in the domain for task execution before the break.

The domain for task execution is larger when allowing for preemption. As can be seen in Fig. 6, the task can now also start at $t \in \{5,6,7,8\}$. One aspect to consider is that its duration will be longer than if started earlier. In other words, one can no longer refer to the task duration without specifying the starting time slot. Thus, the duration parameter gains a time index: $\bar{\tau}_{i,t}$. In the three alternatives shown in Fig. 6, the duration ranges between the nominal value ($\bar{\tau}_{i,1} = \tau_i = 5$), to 9 and 11 slots, when facing one and two break periods, respectively.

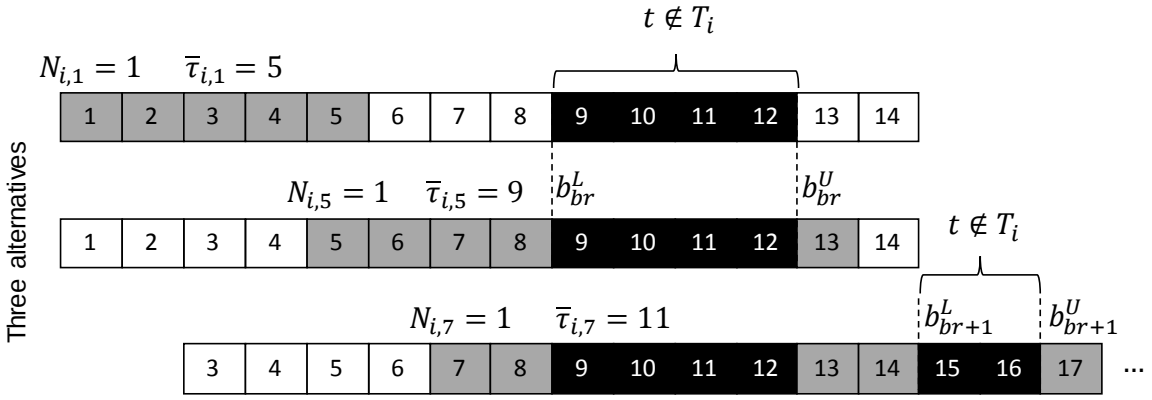

Fig. 6. In preemptive scheduling, domain reduction for task execution is limited to the break periods while task duration varies with the starting time slot.

4.1. Extended formulation for single stage plants

The most successful discrete-time scheduling model in Process Systems Engineering uses the State-Task Network (STN) process representation (Kondili et al., 1993). Here, we are interested in the equipment allocation constraint, which is basically the only constraint needed for single stage plants. The constraint prevents tasks competing for the same unit m to overlap in time and was later tightened

by Shah et al. (1993). Making it suitable for preemptive tasks requires a simple change in the duration parameters (from τ_i to $\bar{\tau}_{i,t}$). In Eq. (35), we show an equivalent constraint (see derivation in Castro et al., 2018).

$$\sum_{i\in I_m}\sum_{\substack{\theta=0:\\ t-\theta\in T_i}}^{\bar{\tau}_{i,t}-1} N_{i,t-\theta} \le 1 \;\forall m,t \tag{35}$$

Each task i is now linked to a pair (j,m), with subsets I_m and I_j respectively holding the tasks that are executed in unit m or correspond to order j. Note that the timing domain of task i (T_i) takes into account the location of breaks as well as release (r_j), due date (d_j) and processing time ($p_{j,m}$).

The execution of every order is enforced through Eq. (36).

$$\sum_{i\in I_j}\sum_{t\in T_i} N_{i,t} = 1 \;\forall j \tag{36}$$

Let $time_t$ be a parameter indicating the starting time of slot t (in minutes or hours). The makespan must then be greater than the ending time of all tasks, as seen in Eq. (37).

$$MS \ge \sum_{t\in T_i} N_{i,t}\cdot time_{t+\bar{\tau}_{i,t}} \;\forall i \tag{37}$$

4.2. Generalized Resource-Task Network (RTN) formulation

In more complex production facilities, we need to keep track of resources other than equipment units. Pantelides (1994) introduced the Resource-Task Network (RTN) process representation to unify the treatment of production resources. The key constraints are the excess resource balances, which relate the availability of resource r at time t (nonnegative continuous variable $R_{r,t}$) with the amount at $t-1$ (or the initial availability R_r^0) plus the amount produced by all tasks finishing at t minus the amount consumed by all tasks starting at t. Assuming tasks with fixed batch size and duration and no interaction with the system boundaries, such as in the flexible flowshop described in section 3.1.2, we give in Eq. (38) the generalized constraint for handling both non-preemptive tasks i' and preemptive tasks i. Note that the differences occur at the level of parameter indices, and so it is straightforward to relax the assumptions and include more terms (one can use as starting point the RTN excess resource balances in Harjunkoski et al. (2014) and Castro et al. (2018)).

$$R_{r,t} = R_r^0|_{t=1} + R_{r,t-1}|_{t>1} + \sum_{i'} \sum_{\substack{\theta=0: \\ t-\theta \in T_{i'}}}^{\tau_{i'}} \mu_{r,i',\theta} N_{i',t-\theta} + \sum_i \sum_{\substack{\theta=0: \\ t-\theta \in T_i}}^{\overline{\tau}_{i,t}} \overline{\mu}_{r,i,t-\theta,\theta} N_{i,t-\theta} \; \forall r, t \qquad (38)$$

In general, the structural parameters $\mu_{r,i,\theta}$ represent the amount of resource r consumed (-) and produced (+) at a time θ relative to the start of the task. Their preemptive counterparts $\overline{\mu}_{r,i,t,\theta}$ gain one index to identify the starting interval t. Let us now consider the specific case of a flexible flowshop, where a task i is linked to a (j, m, k) triplet. The task consumes resource r^-, linked to the output of order j from stage $k-1$, at its start, and produces resource r^+, linked to the output of order j from stage k, at its end. It also holds equipment resource r^* at the start, linked to unit m, releasing it at the end. Returning to the non-preemptive example in Fig. 5, we have $\mu_{r^-,i,0} = \mu_{r^*,i,0} = -1$ and $\mu_{r^+,i,5} = \mu_{r^*,i,5} = 1$. For the preemptive case in Fig. 6, and noting that the largest value of θ is $\overline{\tau}_{i,t}$, we have for the three alternatives (from top to bottom):

- $\overline{\mu}_{r^-,i,1,0}, \overline{\mu}_{r^*,i,1,0} = -1$ and $\overline{\mu}_{r^+,i,1,5}, \overline{\mu}_{r^*,i,1,5} = 1$;
- $\overline{\mu}_{r^-,i,5,0}, \overline{\mu}_{r^*,i,5,0} = -1$ and $\overline{\mu}_{r^+,i,5,9}, \overline{\mu}_{r^*,i,5,9} = 1$;
- $\overline{\mu}_{r^-,i,7,0}, \overline{\mu}_{r^*,i,7,0} = -1$ and $\overline{\mu}_{r^+,i,7,11}, \overline{\mu}_{r^*,i,7,11} = 1$.

Compared to the continuous-time formulations, one can see that the modelling challenge has switched from developing the constraints to generating the parameters of the RTN representation. It should also be emphasized that it is straightforward to consider other discrete time formulations (e.g. Velez et al., 2017).

5. Computational results

The mixed-integer linear programming models were implemented in GAMS 24.9.2 and solved with CPLEX 12.7.1 running in parallel deterministic mode using up to eight threads. We have used default settings except for the termination criteria, a relative optimality tolerance of 10^{-6} or a maximum wall time limit of 3600 CPUs. The hardware consisted of a Windows 10, 64-bit desktop with an Intel i7-4790 (3.6 GHz) processor and 8 GB of RAM.

Two different sets of benchmark problems are considered. The first set is based on a real-life scheduling problem from a steel plant (Castro et al., 2013), a flexible flowshop with $K = 4$ stages, each with two units in parallel. To make the comparison easier between the discrete and continuous-time formulations, we have: (i) relaxed the no interruption constraint in the continuous casting stage (the last) between heats belonging to the same group; (ii) slightly changed the last-stage durations so that the parallel units of a stage are equivalent. The problem involves 24 orders to schedule over one day, with $\delta = 5$ min slots guaranteeing the same data accuracy for the discrete and continuous-time formulations (see processing times in Table 1). We have defined the 3 break periods given in Table 2 and generated sixteen instances of growing complexity by considering a subset of orders, the first up to $|J|$, and break periods. Eq. (39) enforces a maximum transfer time between consecutive stages, with the tr_k^U values being 240, 240 and 120 min, for $k = 1, 2$ and 3, respectively.

$$Ts_{j,k+1} - Tf_{j,k} \leq tr_k^U \ \forall j, k < K \tag{39}$$

Table 1. Processing times $p_{j,k}$ (min) for flexible flowshop problem.

Order/Stage	$k = 1$	$k = 2$	$k = 3$	$k = 4$
$j \in \{1, \dots, 4\}$	80	75	35	50
$j \in \{5, 6\}$	85	80	45	60
$j \in \{7, 8\}$	85	80	20	55
$j \in \{9, \dots, 12\}$	90	95	45	60
$j \in \{13, 14\}$	85	85	25	70
$j \in \{15, 16\}$	85	85	25	75
$j = 17$	80	85	25	75
$j = 18$	80	95	45	60
$j = 19$	80	95	45	70
$j = 20$	80	95	30	70
$j \in \{21, 22\}$	80	80	30	50
$j \in \{23, 24\}$	80	80	30	60

Table 2. Location of break periods (min) for flexible flowshop problem.

Break	b_{br}^L	b_{br}^U
$br = 1$	250	280
$br = 2$	450	475
$br = 3$	700	745

The second set of problems deals with a single stage batch plant comprising $|M| = 5$ units in parallel. The main data for Ex1-Ex4 is taken from Castro and Grossmann (2006) and involve $|J| = 25$ (Ex1-2) to 30 orders. The new information concerns the location of the break periods, which are now unit dependent. Since these problems can be solved in a few seconds, and to test the limits of the models, we have increased the number of orders by 5 per problem until reaching 50 in Ex8. There are three instances for each example, featuring $|BR| = 1, 2$ or 3 break periods. The full set of data is provided as Supplementary Material.

5.1. Flexible flowshop instances

We start the analysis by looking at the results in Table 3 for the discrete-time formulation and the flexible flowshop instances. The first thing to note is that adding break periods leads to an increase in the makespan. This was obviously expected, the main purpose of including the rows for $|BR| = 0$ being the comparison of model size and computational performance to the standard scheduling with no breaks. In this respect, adding breaks leads to a reduction in the number of binary variables in the problem (due to a reduced domain for task execution, recall illustrations in section 4), more so for non-preemptive scheduling, for a computational time roughly in the same order of magnitude. When considering the same breaks for all units, it is easy to generate the preemptive schedule from the schedule with no breaks, one just needs to delay the end of the tasks executed around the break periods for the duration of the break. It is thus no surprise, that the preemptive makespan matches the one for $|BR| = 0$ plus $\sum_{br}(b_{br}^{U} - b_{br}^{L})$, i.e. just add 30, 55 and 100 min for $|BR| = 1, 2$ or 3 break periods (check values in Table 2).

On the other hand, it is difficult to predict the increase in makespan from preemptive to non-preemptive scheduling (note that the lower non-preemptive makespan reported in Table 3 for $|J| = 24$, $|BR| = 2$ is because CPLEX failed to find the optimal preemptive schedule within the time limit of one hour). There are two cases ($|J| = 20$, $|BR| = 1, 2$) for which it is possible to cope with the reduced domain for task execution by changing the production sequence (check Fig. 7 for $|BR| = 2$),

cases with minor (e.g. 5 min in $|J| = 16$, $|BR| = 1$) and major degradation in makespan (e.g. 85 min in $|J| = 12$, $|BR| = 3$, very close to the total duration of breaks, 100 min).

Table 3. Results for flexible flowshop problem and discrete-time formulation (BV= binary variables, LP= makespan of linear relaxation (min), MIP= makespan (min), optimal solutions in bold, maximum computational time in italic).

		Preemptive scheduling				Non-preemptive scheduling			
$\|J\|$	$\|BR\|$	BV	LP	MIP	CPUs	CPUs	MIP	LP	BV
8	0	7664	371.44	**485**	10.6				
	1	7472	393.97	**515**	10.2	6.62	**520**	395.91	7114
	2	7312	400.19	**540**	10.1	8.31	**550**	404.76	6590
10	0	9512	426.38	**575**	32.4				
	1	9272	451.62	**605**	40.0	27.2	**615**	467.26	8820
	2	9072	463.66	**630**	53.4	7.81	**675**	493.87	8148
12	0	11360	477.81	**665**	123				
	1	11072	504.66	**695**	84.2	56.6	**710**	523.16	10526
	2	10832	519.74	**720**	145	16.4	**770**	558.23	9706
	3	10400	525.50	**765**	133	14.3	**850**	572.72	8694
16	0	15128	562.90	**835**	572				
	1	14744	590.47	**865**	285	190	**870**	597.07	14010
	2	14424	608.12	**890**	1087	83.2	**905**	634.88	12912
	3	13848	624.66	**935**	488	91.4	**985**	669.69	11558
20	0	18864	647.82	**1010**	1142				
	1	18384	677.82	**1040**	1399	1150	**1040**	682.10	17462
	2	17984	695.88	**1065**	770	1192	**1065**	709.23	16078
	3	17264	717.52	1115	*3600*	690	**1150**	749.31	14374
24	0	22704	725.41	1310	*3600*				
	1	22128	753.18	1210	*3600*	*3600*	1220	757.67	21026
	2	21648	772.10	1235	*3600*	934	**1225**	781.91	19382
	3	20784	797.49	1285	*3600*	*3600*	1310	829.27	17354

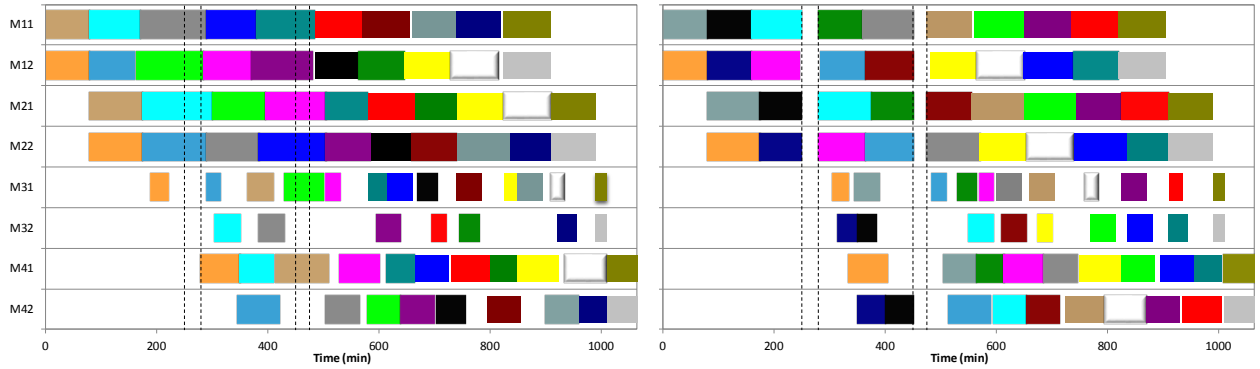

Fig. 7. Optimal solutions for flexible flowshop problem with 20 orders and 2 break points. The makespan (1065 min) for preemptive scheduling (on the left) does not deteriorate when switching to non-preemptive mode (on the right).

Table 4. Results for flexible flowshop problem and continuous-time formulation.

		Preemptive scheduling				Non-preemptive scheduling			
$\|J\|$	$\|BR\|$	BV	LP	MIP	CPUs	CPUs	MIP	LP	BV
8	0	176	485	**485**	0.29				
	1	272	515	**515**	0.93	2.84	**520**	515	240
	2	368	540	**540**	1.97	18.2	**550**	540	304
10	0	260	575	**575**	0.97				
	1	380	605	**605**	7.97	*3600*	**615**	605	340
	2	500	630	**630**	2.94	*3600*	**675**	630	420
12	0	360	665	**665**	496				
	1	504	695	**695**	14.3	*3600*	**710**	695	456
	2	648	720	**720**	112	*3600*	**770**	720	552
	3	792	765	**765**	826	*3600*	**850**	765	648
16	0	608	835	840	*3600*				
	1	800	865	870	*3600*	*3600*	**870**	865	736
	2	992	890	900	*3600*	*3600*	910	890	864
	3	1184	935	950	*3600*	*3600*	**985**	935	992
20	0	920	1010	1020	*3600*				
	1	1160	1040	1060	*3600*	*3600*	1105	1040	1080
	2	1400	1065	**1065**	3316	*3600*	1070	1065	1240
	3	1640	1110	1115	*3600*	*3600*	1165	1110	1400
24	0	1296	1170	1180	*3600*				
	1	1584	1200	1210	*3600*	*3600*	1235	1200	1488
	2	1872	1225	1255	*3600*	*3600*	1235	1225	1680
	3	2160	1270	1280	*3600*	*3600*	1370	1270	1872

Table 4 presents the results for the continuous-time model. Comparing the results between Table 3 and Table 4, one concludes that the discrete-time formulation has much better performance in the flowshop problem than its continuous counterpart for the larger instances. Despite using a number of binary variables that is roughly one order of magnitude larger, it can solve 12 preemptive instances to optimality vs. 8 ($|BR| > 0$). The difference is even bigger for the nonpreemptive case (14 vs. 2). For the latter case, the continuous-time formulation did find the optimal solution in 7 other instances, but it was unable to close the optimality gap, due to the presence of the super big-M constraints in Eqs. (12)-(13). For preemptive scheduling, the best possible solution was equal to the root node LP relaxation in all cases. Furthermore, the tightening constraint in Eq. (17), for the specific problem data, seems to lead to a formulation with zero integrality gap (note that the results for the LP column in Table 4 match those of the MIP column in Table 3 for the proven optimal solutions; it is the reason

why the last 5 rows are not in bold). However, CPLEX found it difficult to guide the search in the right direction, failing to find the optimum in all but one instance with 16 orders and beyond, and often terminating with close to one million open nodes.

5.2. Instances from single stage plants with parallel units

The single stage problems feature order due dates and unit-specific breaks, two characteristics that make it more difficult to generate an optimal preemptive schedule from one without breaks. The results in Table 5 for the discrete-time formulation, show a more profound difference in computational performance, with the non-preemptive mode of operation being often one or two orders of magnitude faster. It is a consequence of the major reduction in problem size achieved by each break period (compare BV columns). In particular, all problems with $|BR| = 3$ can be solved to optimality for non-preemptive scheduling, and there only two suboptimal solutions returned for EX7 and EX8 ($|BR| = 1$) vs. all six cases when considering preemption.

While the discrete-time formulation performs reasonably well for the single stage instances, it is not as fast as the continuous-time formulation. As can be seen in Table 6, all but one instances can be solved to optimality except EX8 for $|BR| = 3$ (preemptive case). By optimality, we mean that the solutions obtained using the number of slots listed in the $|T|$ columns were confirmed for $|T| + 1$, which is the standard heuristic procedure for time-grid based formulations (Harjunkoski et al., 2014). Notice that the LP relaxation for a particular example varies with the number of slots but not with the number of break periods. Since both the makespan and problem size increase with the number of breaks, it is not surprising to observe an increase in computational time.

The values of $|T|$ in Table 6 indicate the maximum number of orders allocated to a unit. Dividing the number of orders in the problem ($|J|$) by the number of units ($|M| = 5$) yields a lower bound on the value of $|T|$ that achieves feasibility, which for EX1-EX8 is always an integer number. It corresponds to a balanced distribution of orders over units, which is enough to ensure optimality in many cases (e.g. EX2 and EX7 for preemptive scheduling). Interestingly, the optimal value of $|T|$ is

often not the same for the two operating modes, an indication of the major changes required in sequencing (already reported for the flowshop instances) and order-unit assignment (processing times now differ between units). As an illustration, we show in Fig. 8 the best-found/optimal schedules for the most difficult problem, EX8 with $|BR| = 3$.

Table 5. Results for single stage plant and discrete-time formulation.

			Preemptive scheduling				Non-preemptive scheduling			
Problem	$\|J\|$	$\|BR\|$	BV	LP	MIP	CPUs	CPUs	MIP	LP	BV
EX1	25	0	13548	139.86	**188**	18.7				
		1	12709	147.72	**198**	22.2	9.48	**215**	161.77	10349
		2	12021	153.50	**208**	366	2.31	**244**	208.98	6555
		3	11428	159.00	**215**	33.5	1.20	**268**	243.14	4350
EX2	25	0	13273	146.10	**197**	19.1				
		1	12434	155.98	**207**	16.4	10.5	**224**	171.89	10025
		2	11748	166.41	**219**	20.3	1.95	**247**	219.00	6169
		3	11156	172	**227**	16.3	1.09	**266**	246.81	4017
EX3	30	0	18063	125.24	**178**	*3600*				
		1	16146	136.53	**195**	*3600*	16.9	**196**	142.27	13541
		2	14261	151.62	**206**	35.5	4.51	**227**	190.74	9281
		3	13432	156.25	**214**	27.9	2.26	**257**	206.92	6923
EX4	30	0	16753	175.39	**220**	*3600*				
		1	15751	172.38	**232**	*3600*	8.53	**242**	189.05	12607
		2	14987	178.77	**238**	2454	2.14	**274**	244.17	7909
		3	14547	186.48	**240**	33.2	1.39	**289**	289	5206
EX5	35	0	25329	167.61	**253**	*3600*				
		1	24330	176.77	**264**	*3600*	*3600*	**279**	188.57	21695
		2	23190	185.27	**271**	*3600*	9.28	**294**	225.88	15005
		3	21858	196.40	**281**	*3600*	5.49	**319**	274.32	9867
EX6	40	0	30453	166.7	237	*3600*				
		1	29451	170.75	**244**	*3600*	*3600*	**250**	173.08	26982
		2	28217	176.90	**252**	*3600*	3511	**267**	195.45	20526
		3	26740	182.17	**260**	*3600*	54.8	**291**	215.82	14995
EX7	45	0	37497	161.66	**256**	*3600*				
		1	36178	169.64	267	*3600*	*3600*	276	176.13	32730
		2	34710	176.41	275	*3600*	*3600*	**286**	188.77	24605
		3	32905	182.39	285	*3600*	53.8	**303**	210.48	18020
EX8	50	0	40885	183.07	289	*3600*				
		1	39429	193.69	298	*3600*	*3600*	302	198.11	35818
		2	37836	200.99	304	*3600*	*3600*	**313**	212.85	27185
		3	35979	210.74	313	*3600*	96.8	**326**	243.13	20087

Table 6. Results for single stage plant and continuous-time formulation.

				Preemptive scheduling				Non-preemptive scheduling				
Problem	$\|J\|$	$\|BR\|$	$\|T\|$	BV	LP	MIP	CPUs	CPUs	MIP	LP	BV	$\|T\|$
EX1	25	0	5	650	187	**188**	0.62					
		1	6	870	173.99	**198**	2.56	3.68	**215**	173.99	840	6
		2	6	960	173.99	**208**	10.0	4.20	**244**	173.99	900	6
		3	6	1050	173.99	**215**	13.3	5.91	**268**	173.99	960	6
EX2	25	0	5	650	196	**197**	1.14					
		1	5	725	196	**207**	1.23	3.78	**224**	183.05	840	6
		2	5	800	196	**219**	2.17	1.99	**247**	183.05	900	6
		3	5	875	196	**227**	3.87	6.22	**266**	183.05	960	6
EX3	30	0	7	1085	166.32	**178**	2.21					
		1	6	1020	180	**195**	1.57	1.46	**196**	180	990	6
		2	6	1110	180	**206**	2.34	8.99	**227**	166.32	1225	7
		3	7	1400	166.32	**214**	13.4	3.14	**257**	180	1110	6
EX4	30	0	7	1085	206.99	**220**	2.32					
		1	7	1190	206.99	**232**	12.1	7.89	**242**	206.99	1155	7
		2	6	1110	224.08	**238**	4.14	6.99	**274**	206.99	1225	7
		3	7	1400	206.99	**240**	28.6	13.0	**289**	206.99	1295	7
EX5	35	0	7	1260	250.53	**253**	0.82					
		1	7	1365	250.53	**264**	3.40	36.9	**279**	245.91	1520	8
		2	8	1680	245.91	**271**	10.6	6.38	**294**	245.91	1600	8
		3	8	1800	245.91	**281**	34.7	8.73	**319**	245.91	1680	8
EX6	40	0	9	1845	226.98	**235**	13.5					
		1	9	1980	226.98	**244**	26.0	18.7	**250**	226.98	1935	9
		2	8	1880	233.46	**252**	8.08	36.8	**267**	226.98	2025	9
		3	9	2250	226.98	**260**	98.5	31.6	**291**	226.98	2115	9
EX7	45	0	9	2070	253.95	**256**	2.71					
		1	9	2205	253.95	**266**	5.28	8.33	**275**	253.95	2160	9
		2	9	2340	253.95	**273**	14.7	5.97	**286**	253.95	2250	9
		3	9	2475	253.95	**283**	61.1	33.2	**303**	251.36	2600	10
EX8	50	0	11	2805	281.80	**286**	49.1					
		1	11	2970	281.80	**296**	41.5	92.8	**300**	281.80	2915	11
		2	11	3135	281.80	**303**	139	24.1	**313**	287.48	2750	10
		3	11	3300	281.80	311	*3600*	103	**326**	281.80	3135	11

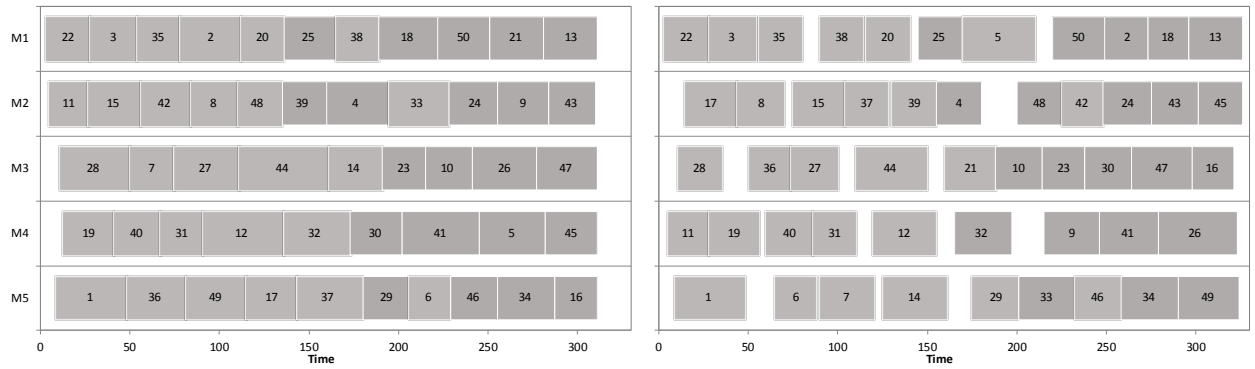

Fig. 8. Optimal solutions for single stage problem with 50 orders and 3 break points (preemptive schedule on the left, makespan=311; non-preemptive schedule on the right, makespan=326).

6. Conclusions

For cases where it is operationally feasible, taking advantage of the possibility of production tasks being interrupted when facing a break, is important since it can improve equipment utilization and reduce the makespan. This paper has expanded the scope of scheduling formulations by showing how to handle preemptive tasks in a computationally efficient way. Both discrete and continuous-time formulations were considered, the latter relying on the concept of general precedence or multiple time grids.

Events occurring at specific points in time, like breaks, release and due dates, are valuable information that can be used in a preprocessing step to reduce the domain of the binary variables of a discrete-time formulation. Preemption makes the duration of the task to be dependent on its starting time, a constraint that can be reflected in the model parameters, which gain one index. This is a very minor difference compared to the standard formulation for non-preemptive tasks.

Coping with break periods in continuous-time formulations, on the other hand, is associated to new sets of binary variables and constraints, which typically make the optimization problem more difficult to solve. We have seen that these can be separated from the model part that handles the assignment of orders to units and sequencing of orders allocated to a unit, either explicitly for the general precedence formulation, or implicitly for multiple time grids. Non-preemptive tasks need to finish

before the start of a break or start after its end, leading to two new sets of binary variables. Preemptive tasks have the additional possibility of being interrupted during the break.

Through the solution of a set of benchmark problems, we have found that the best formulation for a problem depends on the plant topology. The discrete-time formulation was found better for flexible flowshops, despite featuring an integrality gap greater than zero (unlike its continuous-time counterpart for preemptive scheduling) and generating MILPs that are one order of magnitude larger in size. It is primarily because continuous-time formulations cannot avoid using big-M constraints, either to do the sequencing of orders in a unit, as with the general precedence model used for comparison, or to do the transfer from one stage to the next, like in multiple time grid formulations. Benefiting from a sharp/compact hull reformulation of the disjunctions used to derive the model, the multiple time grid formulation solved faster for single stage plants.

Overall, accounting for breaks in a scheduling problem keeps the computational time in the same order of magnitude.

Acknowledgments

This work was supported by Fundação para a Ciência e Tecnologia (projects IF/00781/2013 and UID/MAT/04561/2013); and ABB Corporate Research.

References

ANSI/ISA-S95.00.01-2000, 2000. Enterprise-Control System Integration. Part 1: Models and Terminology, ISBN: 1-55617-727-5.

ANSI/ISA-95.00. 03-2005, 2005. Enterprise-Control System Integration. Part 3: Activity Models of Manufacturing Operations Management. ISA—The Instrumentation, Systems, and Automation Society, ResearchTriangle Park, NC.

Balas, E., 1979. Disjunctive programming. Annals of Discrete Mathematics 5, 3-51.

Balas E., 1985. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J. Algebraic Discrete Methods 6(3), 466-486.

Batsyn, M., Goldengorin, B., Pardalos, P.M., Sukhov, P., 2014. Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time. Optimization Methods & Software 29, 955-963.

Brucker, P., Heitmann, S., Hurink, J., 2003. How useful are preemptive schedules? Operations Research Letters 31, 129-136.

Castro, P.M., Grossmann, I.E., 2005. New Continuous-Time MILP Model for the Short-Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res. 44, 9175-9190.

Castro, P.M., Grossmann, I.E., 2006. An efficient MILP model for the short-term scheduling of single stage batch plants. Comput. Chem. Eng. 30, 1003-1018.

Castro, P.M., Grossmann, I.E., 2012. Generalized Disjunctive Programming as a Systematic Modeling Framework to Derive Scheduling Formulations. Ind. Eng. Chem. Res. 51, 5781-5792.

Castro, P.M., Grossmann, I.E., Veldhuizen, P., Esplin, D., 2014. Optimal Maintenance Scheduling of a Gas Engine Power Plant using Generalized Disjunctive Programming, AIChE J. 60, 2083-2097.

Castro, P.M., Grossmann, I.E., Zhang, Q., 2018. Expanding scope and computational challenges in process scheduling. Comput. Chem. Eng. 114, 14-42.

Castro, P.M., Sun, L., Harjunkoski, I., 2013. Resource-Task Network Formulations for Industrial Demand Side Management of a Steel Plant. Ind. Eng. Chem. Res. 52, 13046-13058.

Graham, R.L., Lawler, E.L., Lenstra, J.K, Rinnooy Kan, 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. Ann. Discrete Math. 5, 287-326.

Harjunkoski, I., 2016. Deploying scheduling solutions in an industrial environment. Comput. Chem. Eng. 91, 127–135.

Harjunkoski, I., Maravelias, C., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial applications of production scheduling models and solution methods. Comput. Chem. Eng. 62, 161–193.

Javanmard, S., Afshar-Nadjafi, B., Niaki, S.T.A., 2017. Preemptive Multi-Skilled Resource Investment Project Scheduling Problem: Mathematical Modelling and Solution Approaches. Comput. Chem. Eng. 96, 55-68.

Jeroslow, R.G., Lowe, J.K., 1984. Modelling with integer variables. Math. Programming Studies 22, North-Holland, Amsterdam, pp 167-184.

Kondili, E., Pantelides, C.C., Sargent, R.W.H., 1993. A General Algorithm for Short-Term Scheduling of Batch Operations – 1. MILP formulation. Comput. Chem. Eng. 17, 211-227.

Méndez, C. A., Cerdá, J., 2002. An MILP framework for short-term scheduling of single-stage batch plants with limited discrete resources. In J. Grievink, & J. van Schijndel (Eds.), Computer-aided chemical engineering (Vol. 10) (p. 721). Amsterdam, The Netherlands: Elsevier.

Pantelides, C.C., 1994. 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, pp 253.

Pei, J., Fan, W., Pardalos, P.M., Liu, X., Goldengorin, B., Yang, S., 2015. Preemptive Scheduling in Two-Stage Supply Chain to Minimize the Makespan. Optimization Methods & Software 30, 727-747.

Peteghem, V.V., Vanhoucke, M., 2010. A Genetic Algorithm for the Preemptive and Non-Preemptive Multi-Mode Resource-Constrained Project Scheduling Problem. European Journal of Operational Research 201, 409-418.

Raman, R., Grossmann, I.E., 1994. Modeling and computational techniques for logic based integer programming. Comput. Chem. Eng. 18, 563-578.

Shah, N., Pantelides, C.C., Sargent, R.W.H., 1993. A General Algorithm for Short-Term Scheduling of Batch Operations – 2. Computational Issues. Comput. Chem. Eng. 17, 229-244.

Velez, S., Dong, Y., Maravelias, C.T., 2017. Changeover Formulations for Discrete-Time Mixed-Integer Programming Scheduling Models. European Journal of Operational Research 260, 949-963.

Wassick, J.M., Ferrio, J., 2011. Extending the resource task network for industrial applications. Comput. Chem. Eng. 35(10), 2124-2140.