European Journal of Operational Research 207 (2010) 644–655

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry

Georgios M. Kopanos [a], Carlos A. Méndez [b], Luis Puigjaner [a,*]

[a] Department of Chemical Engineering, Universitat Politècnica de Catalunya, ETSEIB, Av. Diagonal 647, 08028 Barcelona, Spain
[b] INTEC (UNL-CONICET), Güemes 3450, 3000 Santa Fe, Argentina

ARTICLE INFO

Article history:
Received 26 October 2009
Accepted 1 June 2010
Available online 16 June 2010

Keywords:
Scheduling
Large scale optimization
Mixed integer programming
Decomposition strategy
Pharmaceutical industry

ABSTRACT

An efficient systematic iterative solution strategy for solving real-world scheduling problems in multiproduct multistage batch plants is presented. Since the proposed method has its core a mathematical model, two alternative MIP scheduling formulations are suggested. The MIP-based solution strategy consists of a constructive step, wherein a feasible and initial solution is rapidly generated by following an iterative insertion procedure, and an improvement step, wherein the initial solution is systematically enhanced by implementing iteratively several rescheduling techniques, based on the mathematical model. A salient feature of our approach is that the scheduler can maintain the number of decisions at a reasonable level thus reducing appropriately the search space. A fact that usually results in manageable model sizes that often guarantees a more stable and predictable optimization model behavior. The proposed strategy performance is tested on several complicated problem instances of a multiproduct multistage pharmaceuticals scheduling problem. On average, high quality solutions are reported with relatively low computational effort. Authors encourage other researchers to adopt the large-scale pharmaceutical scheduling problem to test on it their solution techniques, and use it as a challenging comparison reference.

1. Introduction

Nowadays, it is widely recognized that the current gap between practice and theory in the area of short-term scheduling needs to be bridged, as clearly remarked in Méndez et al. (2006) and Ruiz et al. (2008). New academic developments are mostly tested on relatively small problems whereas current real-world industrial applications consist of hundreds of batches, numerous multiple units available for each task and long sequence of processing stages. Additionally, there exist a wide range of operational constraints which should be taken into account in order to guarantee the feasibility of the proposed solution. Most industrial problems are very hard-constrained, thus optimization solvers have to find optimal or near-optimal solutions in a huge search space with a relatively small feasible region. This fact may result in unstable and unpredictable computational performance of optimization models, which is definitely not suitable for industrial environments.

Since most industrial scheduling problems are highly combinatorial and complex decision-making processes, they rarely can be solved to optimality within a reasonable amount of computational time. In addition, the computational effort to find a good solution tends to be as important as the scheduling problem itself; since industry demands solutions that are both optimal, or at least close-optimal, and quick to be reached. As a result, the development of efficient heuristic or metaheuristic techniques has emerged as a promising option to face the inherent computational burden of this problem. For instance, dispatching rules, genetic algorithms, graphs theory, simulated annealing, tabu search, particle swarm and ant colony optimization methods have been widely used in a variety of scheduling problems. Some excellent contributions in this direction can be found in Franca et al. (1996), Raaymakers and Hoogeveen (2000), Pacciarelli (2002), Ruiz and Maroto (2006), Ruiz and Stutzle (2008), and Venditti et al. (2010), among many others. Despite the fact that the aforementioned methods may generate fast and effective solutions for complex problems, they are usually tailor-made and cannot systematically estimate the degree of quality of the solution generated. Moreover, the efficiency of these techniques strongly depends on the proper implementation and fine tuning of parameters since they combine the problem representation and the solution strategy into the same optimization framework.

In contrast, mathematical approaches divide the problem representation and the solution strategy in an exact MIP model and

* Corresponding author. Tel.: +34 934 016 678; fax: +34 934 010 979.
E-mail address: luis.puigjaner@upc.edu (L. Puigjaner).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.06.002

Nomenclature

Indices/sets

$i, i', i'' \in I$	product orders
$j \in J$	processing units
$s \in S$	processing stages

Subsets

I^{in}	set of product orders i that are included into the optimization
J_i	available processing units j to process product i
J_s	available processing units j to process stage s
S_i	set of stages s for each product order i
S_i^{last}	last processing stage for product order i

Parameters

α_i	weighing coefficient for earliness for product i
β_i	weighing coefficient for tardiness for product i
$\gamma_{ii'j}$	sequence-dependent setup time between orders i and i' in unit j
δ_i	due date for order i
ε_j	time point that unit j is available to start processing
M	a big number
μ_{s-1s}	batch transfer time between two consecutive stages $s-1$ and s
$\xi_{ii'j}$	sequence-dependent setup cost between orders i and i' in unit j
o_i	release time for product i
π_{ij}	sequence-independent setup time of product i in unit j
τ_{isj}	processing time for stage s of product i in unit j
ψ	operating cost of production facility per time unit

Continuous variables

C_{is}	completion time of stage s of product i
C_{max}	makespan
E_i	earliness for product i
T_i	tardiness for product i
W_{is}	the time that stage s of a product i is stored (waits) before proceeding to the following processing stage $s+1$
$Z_{ii'j}$	position difference between orders i and i' when both are allocated to the same unit j

Binary variables

$X_{ii'j}$	if product i is processed before product i', when both are allocated to the same unit j, then $X_{ii'j}=1$, otherwise $X_{ii'j}=0$
$\overline{X}_{ii'j}$	if product i is processed exactly before product i', when both are assigned on the same unit j, then $\overline{X}_{ii'j}=1$, otherwise $\overline{X}_{ii'j}=0$
Y_{isj}	if stage s of product i is allocated to unit j, then $Y_{isj}=1$, otherwise $Y_{isj}=0$

an optimization code. Therefore, a general MIP model can be solved by using alternative commercial optimization solvers, such as CPLEX, XPRESS, OSL, and the more recently developed GUROBI, which have been evolved significantly in last years. Although small and medium size models can be usually solved to optimality by using default values in code parameters, large size problems are generally unmanageable by mathematical formulations. Therefore, in order to make the use of exact methods more attractive in real-world applications, increasing effort has been oriented towards the development of systematic techniques that allow maintaining the number of decisions at a reasonable level, even for large-scale problems. A reduced search space usually results in manageable model sizes that often guarantee a more stable and predictable optimization model behavior. Furthermore, once the best possible feasible solution has been generated in a short time, optimization-based methods could be employed to gradually enhance a non-optimal solution with low computational effort. Following this trend, the work of Castro et al. (2009) have been recently emerged as alternative solution strategies to these challenging problems. An apparent drawback of these techniques is that optimality can no longer be assured. Nevertheless, from a practical point of view, guaranteeing global optimality may not be relevant in many industrial scenarios mainly due to the following features: (i) a very short time is just available to generate a solution and send it to the plant floor, (ii) optimality is easily lost because of the highly dynamic nature of industrial environments, (iii) implementing the schedule as such is limited by the real process, and (iv) only a part of the real scheduling goals are generally taken into account in the model since not all scheduling objectives can be quantified. Heuristic model reduction methods, decomposition/aggregation techniques, and improvement optimization-based techniques constitute the principal methods that are embedded in exact mathematical models to face large-scale scheduling problems. A brief description of the aforementioned methods follows. For a detailed state-of-the-art refer to Méndez et al. (2006).

(i) Heuristic model reduction methods usually take into advantage an empirical solution tactic or a particular problem feature and incorporate this knowledge into the mathematical problem representation. As a result, good solutions can be generated in a reasonable time. Simple or combined dispatching (preordering) rules are usually adopted. The contributions by Pinto and Grossmann (1995), Cerdá et al. (1997), Blömer and Günther (2000), and Méndez et al. (2001) are some representative works of heuristic model reduction methods.

(ii) Approaches based on spatial or temporal decomposition, such as the works by Graves (1982) and Gupta and Maranas (1999), usually rely on Lagrangian decomposition. Aggregation techniques aggregate later time periods within the specified time horizon in order to reduce the dimensionality of the problem, or to aggregate the scheduling problem so that it can be considered as part of a planning problem (see Bassett et al., 1997; Birewar and Grossmann, 1990).

(iii) Improvement optimization-based techniques can be interpreted as a special case of rescheduling where an initial solution is partially adjusted with the only goal of enhancing a particular scheduling criterion. These techniques use the current schedule as the initial point of a procedure that iteratively enhances the existing solution in a systematic manner. The works by Röslof et al. (2001) and Méndez and Cerdá (2003), which followed this direction, have shown promising results with relatively low computational cost.

In this work, a hybrid and systematic solution strategy, having a Mixed Integer Programming (MIP) scheduling formulation as its core, is introduced to address large-scale scheduling problems in multiproduct multistage batch plants. The applicability of the methodology is demonstrated by solving a challenging real-world scheduling problem arising in a pharmaceutical process. The manuscript is organized as follows. Section 2 describes the major problem characteristics of the problem addressed. In Section 3, two MIP

models for solving scheduling problems in multiproduct multistage batch plants are presented. Afterwards, in Section 4, the proposed MIP-based solution strategy is explained in detail. Then, a benchmark large-scale scheduling problem of a pharmaceuticals plant is introduced to Section 5 and a set of problem instances is solved in Section 6 in order to validate the performance of the proposed solution strategy and highlight its potential benefits. Finally, the article concludes with some discussion and remarks in Section 7.

2. Problem statement

This study considers the short-term scheduling problem of industrial-scale multiproduct multistage batch processes with the following features:

- A set of product orders $i \in I$ should be processed by following a predefined sequence of processing stages $s \in S$ with, in general, unrelated processing units $j \in J$ working in parallel.
- Each product order i comprises a single batch that must follow a set of processing stages $s \in S_i$.
- Some orders i may skip some processing stages $s \notin S_i$, since different production recipes are considered.
- Product order i can be processed in a specific subset of units $j \in J_i$. Similarly, processing stage s can be processed in a specific subset of units $j \in J_s$.
- Transition times between consecutive product orders are expressed as the sum of two terms. One depends on both the unit and the order being processed (π_{ij}) while the other also varies with the order previously manufactured in that unit ($\gamma_{ii'j}$). Transition times must be explicitly taken into account in the schedule generation process since they are usually of the same order of magnitude or even larger than the processing times. Consequently, they become a very critical feature when scheduling real-world pharmaceutical processes.
- Model parameters like order due dates (δ_i), processing times (τ_{isj}), sequence-dependent changeover times ($\gamma_{ii'j}$) and costs ($\xi_{ii'j}$), unit-dependent setup times (π_{ij}), order release times (o_i), unit available times (ε_j), and operating cost (ψ) are all deterministic.
- Once the processing of an order in a given stage is started, it should be carried out until completion without interruption (non-preemptive mode).
- Mixing or splitting of product orders is not allowed.

The key decision variables are:

- the allocation of products i to units $j \in J_i$ per stage, Y_{isj};
- the relative sequence for any pair of products i, i' at unit $j \in (J_i \cap J_{i'})$, $X_{ii'j}$;
- the completion time of products i at stage $s \in S_i$, C_{is}.

Alternative objective functions can be considered, such as the minimization of makespan, total weighted lateness or total operating and changeovers cost.

3. Mathematical formulations

In this section, two batch-oriented mathematical models are presented for solving scheduling problems in multiproduct multistage batch plants. Both models are based on a continuous-time domain and utilize sequencing variables. The first model is based on the general (global) precedence sequencing concept and the latter one is based on the unit-specific general precedence sequencing concept, recently introduced by Kopanos et al. (2009). Global precedence formulations result in models with small model size and they are computationally faster on average. However, a drawback of these models is that they cannot optimize objectives containing sequence-dependent setup issues (e.g., minimization of changeover costs). For this reason, a unit-specific general precedence model, for scheduling multiproduct multistage batch plants, able to cope with any objective function, is also presented and is proposed as a more general mathematical formulation.

It is worth noticing that the MIP models, presented in this work, are not claimed to be either the fastest or the tightest. However, in order to present the proposed MIP-based solution strategy, the MIP models adopted were entirely developed by the authors rather than using some models from other sources. Otherwise, other mathematical formulations found in the literature could be used as core MIP models as well in the proposed solution strategy. The description of the mathematical frameworks used in this work follows.

3.1. The general precedence multistage scheduling framework

The problem under study can be formulated by the following sets of constraints using the general precedence notion:

$$\sum_{j \in (J_i \cap J_s)} Y_{isj} = 1 \quad \forall i \in I^{in},\ s \in S_i, \tag{1}$$

$$C_{is} \geqslant \sum_{j \in (J_i \cap J_s)} (max[\varepsilon_j, o_i] + \pi_{ij} + \tau_{isj}) Y_{isj} \quad \forall i \in I^{in},\ s \in S_i : s = 1, \tag{2}$$

$$C_{is} - \sum_{j \in (J_i \cap J_s)} (\pi_{ij} + \tau_{isj}) Y_{isj} = C_{is-1} + W_{is-1} + \mu_{s-1s} \quad \forall i \in I^{in},\ s \in S_i : s > 1, \tag{3}$$

$$\begin{aligned} C_{is} + \gamma_{ii'j} \leqslant\ & C_{i's} - \pi_{i'j} - \tau_{i'sj} + M(1 - X_{ii'j}) \\ & + M(2 - Y_{isj} - Y_{i'sj}) \quad \forall i \in I^{in},\ i' \in I^{in}, \\ & s \in S_i,\ j \in (J_s \cap J_i \cap J_{i'}) : i' > i, \end{aligned} \tag{4}$$

$$\begin{aligned} C_{i's} + \gamma_{i'ij} \leqslant\ & C_{is} - \pi_{ij} - \tau_{isj} + MX_{ii'j} + M(2 - Y_{isj} - Y_{i'sj}) \\ & \forall i \in I^{in},\ i' \in I^{in},\ s \in S_i,\ j \in (J_s \cap J_i \cap J_{i'}) : i' > i, \end{aligned} \tag{5}$$

$$\begin{aligned} & Y_{isj} \in \{0,1\} \quad \forall i \in I^{in},\ s \in S_i,\ j \in (J_s \cap J_i), \\ & X_{ii'j} \in \{0,1\} \quad \forall i \in I^{in},\ i' \in I^{in},\ j \in (J_i \cap J_i') : i' \neq i, \\ & W_{is} \geqslant 0 \quad \forall i \in I^{in},\ s \in S_i : s < S, \\ & C_{is} \geqslant 0 \quad \forall i \in I^{in},\ s \in S_i. \end{aligned} \tag{6}$$

Constraint set (1) assures that every product order goes through one unit $j \in (J_s \cap J_i)$ at each stage $s \in S_i$. Constraint set (2) defines the completion time of the first stage for every product. Notice that this set of constraints takes into account possible release order o_i and available unit ε_j times. Constraint set (3) gives the timing for every product order between to consecutive stages. This set of constraints allows for the consideration of possible transferring times between two sequential stages. Positive variable W_{is-1} reflects the wait time of every product batch before proceeding to the following processing stage. Note that in Zero Wait (ZW) storage policy W_{is-1} is set to zero. In Unlimited Intermediate Storage (UIS) policy, W_{is-1} is left free or, alternatively, it can be eliminated and substitute the equality by a greater-or-equal inequality. In order to model storage policies like Non Intermediate Storage (NIS) and Finite Intermediate Storage (FIS), appropriate sets of constrains found in the literature can be easily added to the current model. To continue with, constraint sets (4) and (5) give the relative sequencing of product batches at each processing unit. These sets of big-M constraints force the starting time of an order i' to be greater than the

completion time of whichever order i processed beforehand. Note that $X_{ii'j}$ corresponds to the global sequencing binary variable. Have in mind that $X_{ii'j}$ is active (i.e., $X_{ii'j}=1$) for all product batches i' that are processed after product batch i. Finally, the decision variables are defined by (6). Henceforth, we will refer to the MIP model that constitutes by constraint sets (1)–(6) as *GP*.

3.2. The unit-specific general precedence multistage scheduling framework

The following sets of constraints are proposed for coping with scheduling problems where the optimization of sequence-dependent issues are the optimization goal or are a part of it. The constraints are:

$$\sum_{j\in(J_i\cap J_s)} Y_{isj}=1 \quad \forall i\in I^{in},\ s\in S_i, \tag{7}$$

$$C_{is}\geqslant \sum_{j\in(J_i\cap J_s)}(max[\varepsilon_j,o_i]+\pi_{ij}+\tau_{isj})Y_{isj}+\sum_{i'\neq i}\sum_{j\in(J_s\cap J_i\cap J_{i'})}\gamma_{i'ij}\overline{X}_{i'ij} \quad \forall i\in I^{in},\ s\in S_i: s=1, \tag{8}$$

$$C_{is}-\sum_{j\in(J_i\cap J_s)}(\pi_{ij}+\tau_{isj})Y_{isj}=C_{is-1}+W_{is-1}+\mu_{s-1s} \quad \forall i\in I^{in},\ s\in S_i: s>1, \tag{9}$$

$$C_{is}+\gamma_{ii'j}\overline{X}_{ii'j}\leqslant C_{i's}-\pi_{i'j}-\tau_{i'sj}+M(1-X_{ii'j}) \quad \forall i\in I^{in},\ i'\in I^{in},\ s\in S_i, j\in(J_s\cap J_i\cap J_{i'}): i'\neq i, \tag{10}$$

$$Y_{isj}+Y_{i'sj}\leqslant 1+X_{ii'j}+X_{i'ij} \quad \forall i\in I^{in},\ i'\in I^{in},\ s\in S_i, j\in(J_s\cap J_i\cap J_{i'}): i'>i, \tag{11}$$

$$2(X_{ii'j}+X_{i'ij})\leqslant Y_{isj}+Y_{i'sj} \quad \forall i\in I^{in},\ i'\in I^{in},\ s\in S_i, j\in(J_s\cap J_i\cap J_{i'}): i'>i, \tag{12}$$

$$Z_{ii'j}=\sum_{i''\in I^{in}: i''\neq[i,i']}(X_{ii''j}-X_{i'i''j})+M(1-X_{ii'j}) \quad \forall i\in I^{in},\ i'\in I^{in}, j\in(J_i\cap J_{i'}): i'\neq i, \tag{13}$$

$$Z_{ii'j}+\overline{X}_{ii'j}\geqslant 1 \quad \forall i\in I^{in},\ i'\in I^{in},\ j\in(J_i\cap J_{i'}): i'\neq i, \tag{14}$$

$$\begin{aligned}
&Y_{isj}\in\{0,1\} \quad \forall i\in I^{in},\ s\in S_i,\ j\in(J_s\cap J_i),\\
&X_{ii'j}\in\{0,1\} \quad \& \quad \overline{X}_{ii'j}\in\{0,1\} \quad \forall i\in I^{in},\ i'\in I^{in},\ j\in(J_i\cap J_i'): i'\neq i,\\
&Z_{ii'j}\in\Re \quad \forall i\in I^{in},\ i'\in I^{in},\ j\in(J_i\cap J_i'): i'\neq i,\\
&W_{is}\geqslant 0 \quad \forall i\in I^{in},\ s\in S_i: s<S,\\
&C_{is}\geqslant 0 \quad \forall i\in I^{in},\ s\in S_i.
\end{aligned} \tag{15}$$

Constraint set (7) forces that every product order goes through one unit $j\in(J_s\cap J_i)$ at each stage $s\in S_i$. Constraint set (8) gives the completion time of the first stage for every product. Notice that $\overline{X}_{ii'j}$ is the unit-specific immediate precedence binary variable. Constraint set (9) gives the timing for every product order between to consecutive stages and is similar to constraint set (3) of the GP model. Constraint sets (10)–(12) give the relative sequencing of product batches at each processing unit. Big-M constraint set (10) forces the starting time of a product batch i' to be greater than the completion time of whichever product batch i processed beforehand at the same unit. Constraint sets (11) and (12) state that when two-product batches are allocated to the same unit (i.e., $Y_{isj}=Y_{i'sj}=1$), one of the two global sequencing binary variables $X_{ii'j}$ and $X_{i'ij}$ should be active. If the two-product batches are not allocated to the same unit then $X_{ii'j}=X_{i'ij}=0$. To continue with, obviously, two orders i and i' are consecutive only in the case that $X_{ii'j}=1$ and, moreover, when there is no other order i'' between them. In other words, two-product batches i and i' are consecutive *if and only if* the total number of batches that are processed after batch i, if batch i' is excluded, is equal to the total number of batches that are processed after batch i', when batch i is excluded; see Fig. 1. Constraint sets (13) and (14) formulate this concept. Note that the auxiliary variable $Z_{ii'j}$ is zero whenever two products i and i' are sequentially processed in the same unit. The RHS of Eq. (14) can be substituted by $X_{ii'j}$; in some instances this reduces the computational time. Refer to Kopanos et al. (2009) for more details regarding the unit-specific general precedence concept. Finally, the decision variables are defined by (15). Henceforth, we will refer to the MIP model that constitutes by constraint sets (7)–(15) as *USGP*.

3.3. Objective functions

In this section, the different optimization goals used in this study, for solving the short-term scheduling in a real-life multiproduct multistage pharmaceutical batch plant, are presented.

3.3.1. Makespan

The time point at which all product orders are accomplished corresponds to the makespan, which can be calculated by Eq. (16). The makespan objective is closely related to the throughput objective. For instance, minimizing the makespan in a parallel-machine environment with sequence-dependent setup times forces the scheduler to balance the load over the various machines and to minimize the sum of all the setup times in the critical bottleneck path (Pinedo and Chao, 1999).

$$\text{minimize} \quad C_{max}\geqslant C_{is} \quad \forall i\in I^{in},\ s\in S_i^{last}. \tag{16}$$

3.3.2. Total weighted lateness

The minimization of a combined function of earliness and tardiness, as given in Eq. (17), is one of the most widely used objective functions in the scheduling literature. It is also known as weighted lateness. The weighing coefficients α_i and β_i are used to specify the significance of every product order earliness or tardiness, respectively.

$$\text{minimize} \quad \sum_{i\in I^{in}}(\alpha_i E_i+\beta_i T_i). \tag{17}$$

Earliness and tardiness for every product order i are estimated by constraint set (18) and (19), respectively.

$$E_i\geqslant \delta_i-C_{is} \quad \forall i\in I^{in},\ s\in S_i^{last}, \tag{18}$$

$$T_i\geqslant C_{is}-\delta_i \quad \forall i\in I^{in},\ s\in S_i^{last}. \tag{19}$$

This objective, in a sense, accounts for minimizing storage and handling costs while maximizing service and customer satisfaction level.

3.4. Operating and changeovers costs

Operating and changeovers costs goal constitutes a reasonable goal in production environments where sequence-dependent setup costs are significant. The operating cost is denoted by ψ and is defined as the cost for operating the production facility per time unit. Obviously, makespan corresponds to the total operating time. Parameter $\xi_{ii'j}$ stands for the changeover cost from product order i to i' in processing unit j

$$\text{minimize} \quad \left(\psi C_{max}+\sum_{i\in I^{in}}\sum_{i'\in I^{in}, i'\neq i}\sum_{j\in(J_i\cap J_{i'})}\xi_{ii'j}\overline{X}_{ii'j}\right) \tag{20}$$

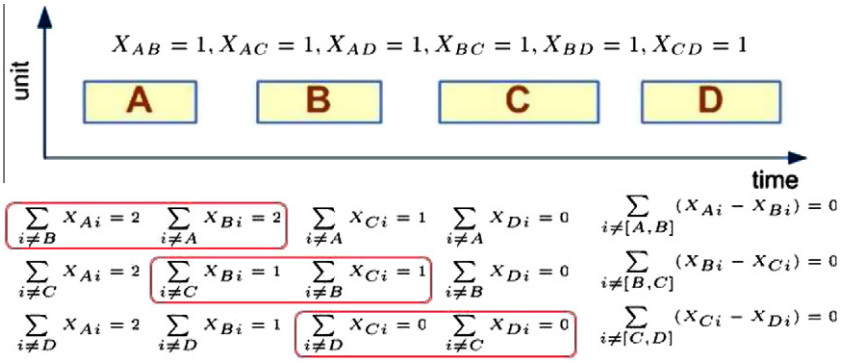

Fig. 1. Unit-specific general precedence concept.

4. The MIP-based solution strategy

Although these mathematical formulations are able to describe a large number of scheduling problems, in practice, they can only solve problems of modest size in a reasonable computational time. Have in mind that the combinatorial complexity strongly increases with the number of product orders considered thus precluding the resolution of real-life scheduling problems by exact methods. According to Herrmann (2006), algorithms that can find optimal solutions to these hard problems in a reasonable amount of time are unlikely to exist.

In a nutshell, the proposed MIP-based solution strategy has as a core a MIP scheduling framework and consists of two major procedure steps: (i) the constructive step, and (ii) the improvement step. The objective in the constructive step is the generation of a feasible schedule in short amount of time. Afterwards, this schedule is gradually improved by implementing some elaborate rescheduling techniques, in the improvement step. As a sequence, the generation of feasible and fairly good schedules in reasonable computational time is favored. A description of the proposed solution strategy steps follows (see Fig. 2).

4.1. Constructive step

In the constructive step, the large-scale scheduling problem is decomposed by scheduling, in an iterative mode, a subset of the involved product orders. That way the MIP solver search space is reduced and the resolution of the problem is favored. Concretely, a predefined number of product orders ($i \in I^{in}$) are scheduled (by solving the MIP model) at each iteration, until all product orders are finally scheduled. User defines the number of product order for each iteration. Note that the number of orders inserted into each iteration should be small enough to ensure the quick MIP model resolution for every iteration, and thus generating a feasible schedule in short time. In this study, it is proposed to insert (schedule) product orders one-by-one, since it has been observed, after a series of experiments, that insertion of a higher number of products per iteration: (i) does not guarantee a better constructive step solution, and (ii) is more computationally expensive.

To continue with, user should also specify the order that products are inserted into the constructive step procedure. An insertion criterion could be adopted in order to decrease the possibility of obtaining a bad constructive step solution. Here, it is proposed to insert first the products will less unit-stage allocation flexibility. In other words, products with less alternative units should be scheduled first. By doing so, unit allocation decisions are first taken for the less unit-stage-flexible products. Consider a single-stage two-product (A and B) batch plant with two parallel processing units (J1 and J2). The product's A processing time in unit J1 is 3 hours and in unit J2 equals to 2 hours. Product B can be only processed in unit J2 in 3 hours. The minimization of makespan is the optimization goal. Consider the following insertion sequences: the case I that first is inserted product A and afterwards product B, which opposes our proposed insertion criterion, and the case II that

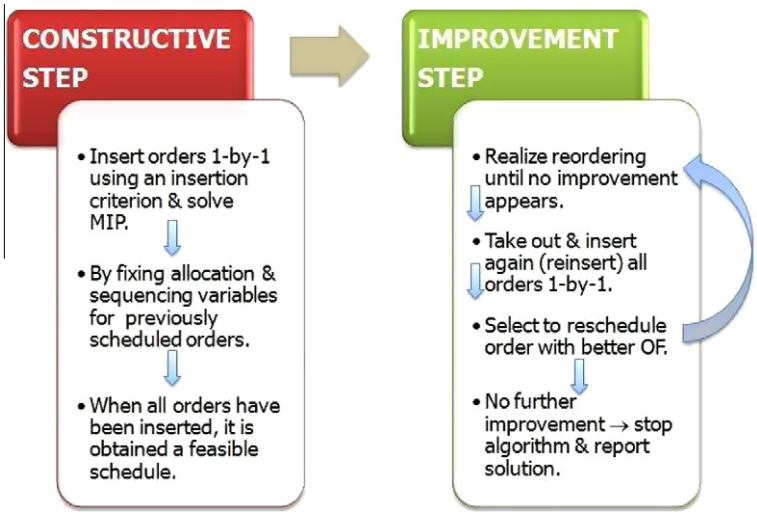

Fig. 2. Representative scheme of the proposed MIP-based solution strategy.

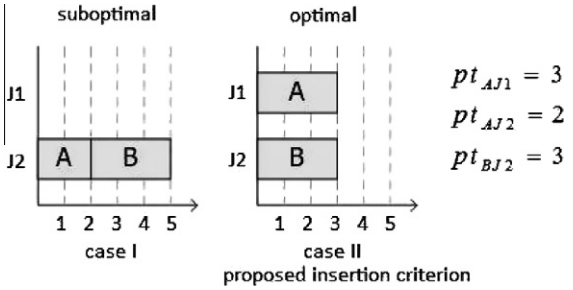

Fig. 3. Illustrative example for insertion criterion.

first is inserted product B and then product A, in accordance with our proposed insertion criterion. As one can observe, the first insertion strategy (case I) results in a makespan of 5 hours. Note that both products are allocated to unit J2. Following our insertion criterion (case II) a makespan of 3 hours is obtained. Fig. 3 illustrates the schedules for both cases.

After the resolution of the MIP model at each iteration, allocation and global sequencing binary variables for the already scheduled product orders are fixed. In other words, unit allocation decisions and relative sequencing relations between the already scheduled products cannot be modified in the following iterations. However, timing decisions may change thus permitting the insertion of new product orders among the previously scheduled product orders. Fig. 4 delineates an illustrative example (single-stage products and single-unit) of the allowed sequences when a product D is inserted to a current schedule containing products A, B, and C. Note that just 4 sequences are permitted, instead of the 24 possible sequences, thus reducing significantly the computational effort. When all product orders have been inserted, a feasible schedule can be finally obtained in relatively short time.

Similar insertion methods have also been implemented to other types of scheduling problems by Nawaz et al. (1983), Werner and Winkler (1995), Röslof et al. (2001), and Röslof et al. (2002). It is pointed out that the insertion order of the product orders influences the quality of the solution. Therefore, a more detailed study and the development of other insertion criteria seems a promising future research direction for enhancing the proposed approach.

4.2. Improvement step

The initial feasible schedule provided by the constructive step can be systematically improved through reordering and/or reassignment MIP-based operations; in accordance with the main rescheduling concepts introduced by Röslof et al. (2001) and Méndez and Cerdá (2003). The improvement step is a two-stage closed loop procedure that consists of the reordering and the reinsertion stage, which are performed sequentially until no improvement is observed. A description of the improvement step follows.

4.2.1. Reordering stage

In this stage, unit allocation decisions, are fixed. Reordering actions are iteratively applied on the initial schedule, by solving a MIP model, until no further improvement is observed. A full unit reordering tactic results impractical due to the large number of batches and processing units in real-world industrial scheduling problems. Instead, the alternative of limited reordering operations may usually improve the current schedule with relatively low computational effort. It is common sense that there exists a strong trade-off between the degrees of freedom and the solution time. In an industrial environment, the scheduler should appropriately define the reordering tactic/limitations, followed in this step, depending on the complexity of the scheduling problem. A local reordering tactic is adopted in this study. Concretely, in an attempt to maintain manageable model sizes, reordering of batches with their direct predecessor or successor is only allowed. An illustrative example is included here to highlight the local reordering computational benefits. Consider the reordering scheduling problem of 4 single-stage products (A, B, C, and D) on a single-unit. As Fig. 5 shows, a local reordering policy will only examine 4 potential sequences instead of the 23 total possible sequences. On the one hand, solution quality is probably decreased since one of the 19 unexplored sequences may yield a better solution. On the other hand, the optimization search space is significantly reduced. Keep in mind that considering the whole set of possible sequences impacts drastically the computational performance of the reordering step. Other less-limited reordering tactics could be also easily applied. The interested reader is referred to the contribution work of Méndez and Cerdá (2003).

ALLOWED SEQUENCES

A	B	C	D
A	B	D	C
A	D	B	C
D	A	B	C

inserted product order

previously scheduled product order

FORBIDDEN SEQUENCES

A	C	B	D
A	C	D	B
A	D	B	C
B	A	C	D
B	A	D	C
B	C	A	D
B	C	D	A
B	D	A	C
B	D	C	A
C	A	B	D
C	A	D	B
C	B	A	D
C	B	D	A
C	D	A	B
C	D	B	A
D	A	C	B
D	B	A	C
D	B	C	A
D	C	A	B
D	C	B	A

Fig. 4. Illustrative example for allowed sequences in constructive step.

A	B	C	D
A	B	D	C
A	C	B	D
A	C	D	B
A	D	B	C
A	D	C	B
B	A	C	D
B	A	D	C
B	C	A	D
B	C	D	A
B	D	A	C
B	D	C	A

C	A	B	D
C	A	D	B
C	B	A	D
C	B	D	A
C	D	A	B
C	D	B	A
D	A	B	C
D	A	C	B
D	B	A	C
D	B	C	A
D	C	A	B
D	C	B	A

local reordering possible sequence current schedule sequence

Fig. 5. Illustrative example for local reordering.

4.2.2. Reinsertion stage

The schedule of the reordering step constitutes the initial schedule in the reinsertion stage. Here, unit allocation and relative sequencing decisions for a small number of product orders are left free by the scheduler. Let us refer to these product orders as reinserted orders. Allocation and relative sequencing decisions, among the non-reinserted orders, are fixed. In other words, some products orders are extracted from the current schedule, and they are reinserted aiming at improving the actual schedule. Note that the reinsertion stage is quite similar to the last iteration of the constructive step (see Fig. 4). Since our scope is to propose a general standard algorithm for large-scale industrial scheduling problems, we adopt the lowest number of reinsertion orders (i.e., one at a time) in order to favor low solution times. However, the scheduler could set the number of reinserted orders depending on the specific scheduling problem. In the standard reinsertion stage, the number of iterations (reinsertions) equals the number of product orders. The solutions of all reinserted orders (iterations) are compared, and the best one is finally chosen as the solution of the reinsertion stage. Note that if the number of product orders is too high, someone could have preferred to end the reinsertion stage once a better solution (comparing it with the previous stage) is reached. That way is saved computational time. If the best solution of this stage is better than the solution of the reordering stage, the algorithm goes to the reordering stage again. Otherwise, the solution algorithm terminates and reports the best solution found.

In the Appendix, some illustrative pseudo-codes can be found for the constructive and the improvement stage of our MIP-based solution strategy.

5. Pharmaceutical production process

A real-world multiproduct multistage pharmaceutical batch plant is studied in the current work. Recently, Castro et al. (2009) have also studied this pharmaceutical facility. Concretely, they solved two problem instances (for 30 and 50 product orders) minimizing the makespan under UIS policy. In this work, we use partially different sets of data (e.g., we introduce due dates, changeover costs) and we deal with more objective functions.

In the current study, the short-term scheduling problem of a considerably high number of multistage product orders (30–60) at the 17 processing units of the production plant is addressed. The production process has 6 processing stages, as Fig. 6 depicts. Some products bypass the third processing stage S3. Processing times can be found in the Supplementary material. Sequence-dependent setup times are also explicitly considered thus increasing the complexity of the problem. An interesting feature of the production process is that in some processing stages changeover times are higher than the processing times. Changeover times are zero in the first stage S1 and 0.45 hours in the second stage S2 among all products. Changeover times for the remaining stages (S3–S6) can be found in the Supplementary material. Finally, changeover costs are defined as the multiplication of the impact factors, in Table 1, and the corresponding changeover time.

6. Experimental studies

In this section, the problem instances details are firstly introduced and the results of these experimental studies are presented and discussed afterwards.

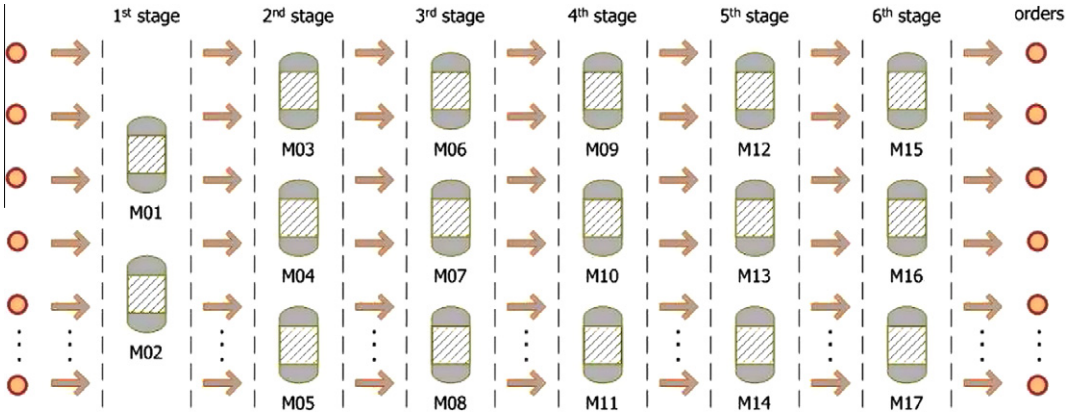

Fig. 6. Pharmaceutical multistage process.

Table 1
Changeover costs' impact factors per time unit (10^3\$/hour).

Products	P01–P10	P11–P20	P21–P30
P01–P10	0.36	0.27	0.27
P11–P20	0.27	0.45	0.27
P21–P30	0.27	0.27	0.54

Table 2
Due dates for product orders (hours).

P01	30.6	P16	20.7	P31	34.2	P46	54.0
P02	16.2	P17	14.4	P32	37.8	P47	49.5
P03	23.4	P18	23.4	P33	37.8	P48	46.8
P04	18.0	P19	20.7	P34	48.6	P49	52.2
P05	27.0	P20	27.0	P35	40.5	P50	54.0
P06	16.2	P21	30.6	P36	34.2	P51	48.6
P07	28.8	P22	9.0	P37	46.8	P52	52.2
P08	20.7	P23	18.0	P38	54.0	P53	27.0
P09	18.0	P24	23.4	P39	46.8	P54	52.2
P10	10.8	P25	23.4	P40	49.5	P55	41.4
P11	30.6	P26	18.0	P41	52.2	P56	49.5
P12	14.4	P27	14.4	P42	40.5	P57	54.0
P13	30.6	P28	9.0	P43	54.0	P58	52.2
P14	14.4	P29	18.0	P44	36.0	P59	40.5
P15	27.0	P30	10.8	P45	52.2	P60	36.0

6.1. Details of problem instances

In this work, 12 different problem instances have been solved. These case studies differ in: (i) the optimization goal (makespan, weighted lateness, and operating and changeover costs), (ii) the number of product orders (30 products, and 60 products), and (iii) the storage policy type (ZW, and UIS).

Since, all the aforementioned data tables provide data about 30 products, two batches for every product are considered in order to address the 60-product cases. Therefore, for instance, the product order P31 has the same processing characteristics with product order P01, the product order P32 has the same processing data with product order P02, and so on. Moreover, notice that the changeover times/costs between P01 and P31 are equal to the changeover times/costs between P01 and P01, as they are given in the data tables, and so on. For the problem instances where the optimization goal is the minimization of weighted lateness (i.e., I.05 to I.08), due dates for every product order are considered; see Table 2. Observe that due dates for two batches of the same product (e.g., P01 and P31) may be different. Additionally, the weighing coefficient for earliness, α_i, equals to 0.9 and the weighing coefficient for tardiness, β_i, is set to 4.5 for all products. Regarding the problem instances that have as an objective the simultaneous minimization of operating and changeovers costs (i.e., I.09 to I.12), the operating cost per time unit, ψ, is considered equal to 0.9×10^3\$/hour.

At this point, it is worth mentioning that GP model has been used to solve the problem instances that involves the minimization of the makespan or the weighted lateness (i.e., I.01 to I.08), and USGP model has been employed to cope with the operating and changeovers costs objective (i.e., I.09 to I.12). Besides, it is emphasized that the 30-product problem instances (I.01, I.02, I.05, I.06, I.09 and I.10) deal with the complex scheduling of 168 product batches, and the 60-product problem instances (I.03, I.04, I.07, I.08, I.11 and I.12) tackle the intricate scheduling problem of 336 product batches.

Table 3
Problem instances: best schedules found within the maximum predefined time limit (3600 CPU seconds).

Problem instance	Objective function	Products (batches)	Storage policy	1st stage solution	1st stage CPU seconds	Best solution	Total CPU seconds	Impovement percent
I.01	C_{max}	30 (168)	UIS	28.507	38	26.559	542	6.83
I.02	C_{max}	30 (168)	ZW	31.520	7	30.532	187	3.14
I.03	C_{max}	60 (336)	UIS	49.161	155	48.548	1502	1.25
I.04	C_{max}	60 (336)	ZW	58.104	106	56.061	1718	3.52
I.05	W.L.	30 (168)	UIS	48.613	22	19.085	720	60.74
I.06	W.L.	30 (168)	ZW	115.016	15	84.438	262	26.59
I.07	W.L.	60 (336)	UIS	118.683	403	87.943	3600	25.90
I.08	W.L.	60 (336)	ZW	629.672	356	515.876	1478	18.07
I.09	O.& C.C	30 (168)	UIS	66.158	94	62.910	3600	4.91
I.10	O.& C.C	30 (168)	ZW	72.318	58	70.209	3600	2.92
I.11	O.& C.C	60 (336)	UIS	119.759	1780	117.909	3600	1.54
I.12	O.& C.C	60 (336)	ZW	139.104	880	134.624	3600	3.22

† W.L. = Weighted Lateness, and O. & C.C = Operating & Changeovers Costs.

Table 4
Comparison between original MIP model and proposed MIP-based strategy best solutions found within the maximum predefined time limit (3600 CPU seconds).

	Original mathematical model						Our strategy	
Problem instance	Constrains	Binary variables	Continuous variables	Gap (%)	Best solution	Total CPU seconds	Best solution	Total CPU seconds
I.01	10,230	5326	295	57.7	34.810	3600	26.559	542
I.02	10,230	5326	295	–	–	3600	30.532	187
I.03	40,988	20916	589	90.1	109.960	3600	48.548	1502
I.04	40,988	20916	589	–	–	3600	56.061	1718
I.05	10,261	5326	355	100.0	428.146	3600	19.085	720
I.06	10,261	5326	355	–	–	3600	84.438	262
I.07	41,049	20916	709	100.0	23453.744	3600	87.943	3600
I.08	41,049	20916	709	–	–	3600	515.876	1478
I.09	39,858	10264	20286	–	–	3600	62.910	3600
I.10	39,858	10264	20286	–	–	3600	70.209	3600
I.11	161,828	41056	81626	–	–	3600	117.909	3600
I.12	161,828	41056	81626	–	–	3600	134.624	3600

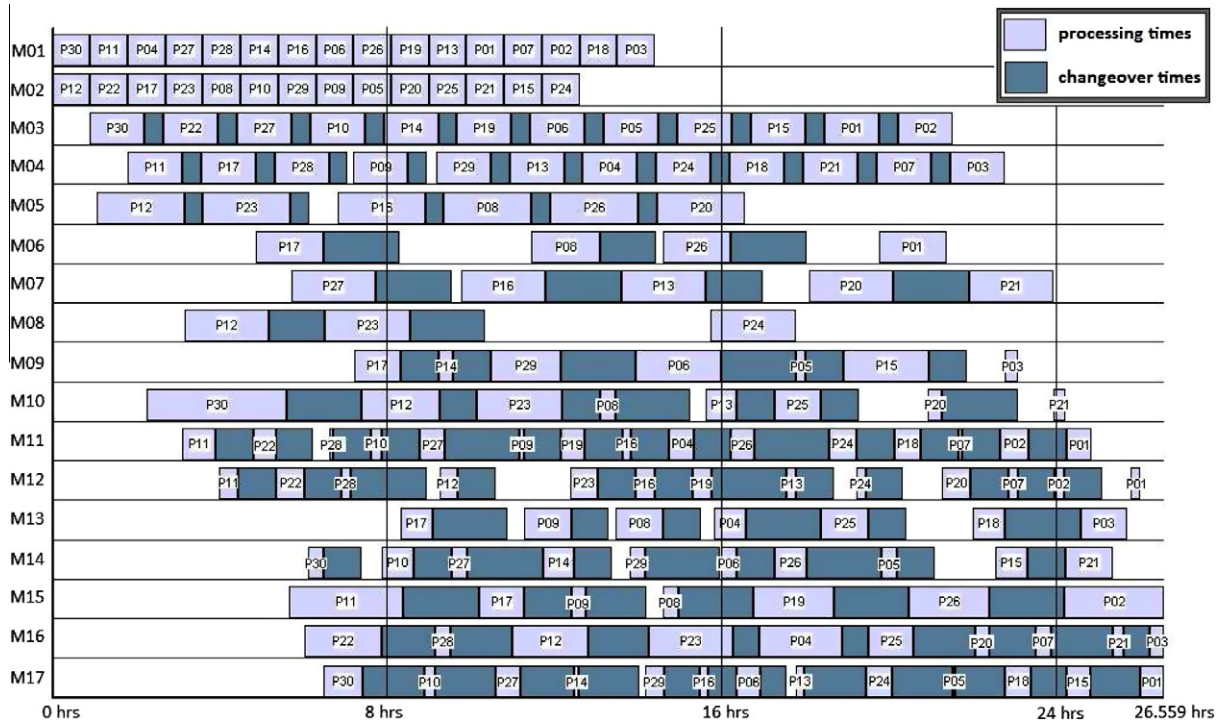

Fig. 7. Best schedule for I.01 (30-product case: min. makespan under UIS policy).

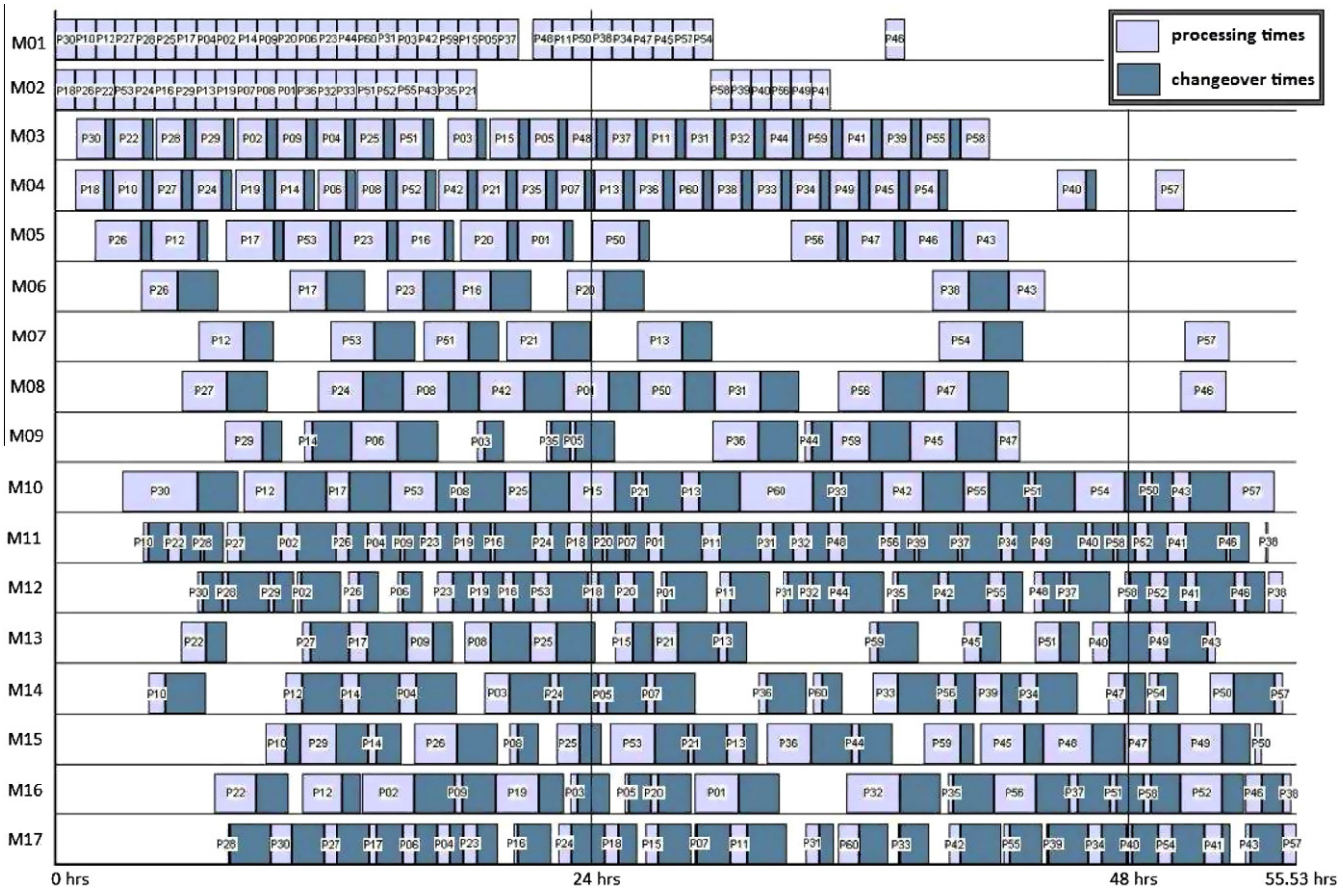

Fig. 8. Best schedule for I.07 (60-product case: min. total weighted lateness under UIS policy).

6.2. Results and discussion

The proposed solution strategy has been tested on a total number of twelve complex problem instances in order to validate its performance. A time limit of 1 CPU hour has been imposed on the solution of every problem instance. All problem instances have been solved in a Dell Inspiron 1520 2.0 GHz with 2 GB RAM using CPLEX 11 via a GAMS 22.8 interface (Brooke et al., 1998).

Table 3 presents the constructive step's solution (initial solution) and the best solution found for every problem instance. The computational time for the constructive step (1st stage) as well as the total computation time are also included to the same table. Note that feasible schedules are obtained in a short amount of time in most cases. Problem instance I.11 is the most time-demanding problem instance since almost half a CPU hr was needed in order to obtain a feasible solution. The remaining problem instances reached a feasible solution in relatively low computational time; from some CPU seconds and no more than 7 CPU minutes.

The MIP-based solution strategy is able to quickly generate feasible solutions and then gradually enhance these solutions. It was observed that the necessary computational time to improve a given solution mainly depends on: (i) the total number of batches to be scheduled, (ii) the objective function, (iii) the storage policy, and (iv) the core mathematical model. Obviously, the lower the total number of bathes the faster the problem is solved. It has been observed that the case studies considering ZW storage policy are solved faster comparing them with the problem instances under UIS policy. Finally, the mathematical model used depends on the optimization goal. Roughly speaking, the more complicated the objective function the bigger the size of the model; such is the case of minimizing operating and changeovers costs.

All problem instances were also solved by using the original undecomposed mathematical formulations in order to underline the high complexity of the problems addressed and to highlight the practical benefits of our proposed solution approach. Table 4 contains the computational features of the original mathematical models and the best solution found, for all problem instances, within the predefined time limit of 1 CPU hour. It is worth mentioning that the number of the sequencing binary variables is strongly augmented by increasing the number of product orders. Note that the least complex problem instances (I.01 and I.02) result into a MIP model of 10,230 equations, 5326 binary variables, and 295 continuous variables, while, the most sophisticated problem instances (I.11 and I.12) result into a huge MIP model of 161,828 constraints, 41,056 binary variables, and 81,626 continuous variables. Observe that a feasible solution was not found by the original mathematical models in 8 of the 12 problem instances. In the remaining problem instances, feasible but very bad solutions (i.e., with big integrality gap) were obtained. For example, the original GP model in problem instance I.01 reported a makespan equal to 34.810 hours, with an integrality gap of 57.71%, after 1 CPU hr while our solution approach gave a makespan of 26.559 hours in just 542 CPU seconds. The solution found by the original GP model is 31.07% worse than that of our approach. It is worth mentioning that all problem instances were also solved by the original MIP models without setting a time limit. However, in all cases the MIP solver terminated because memory capacity was exceeded.

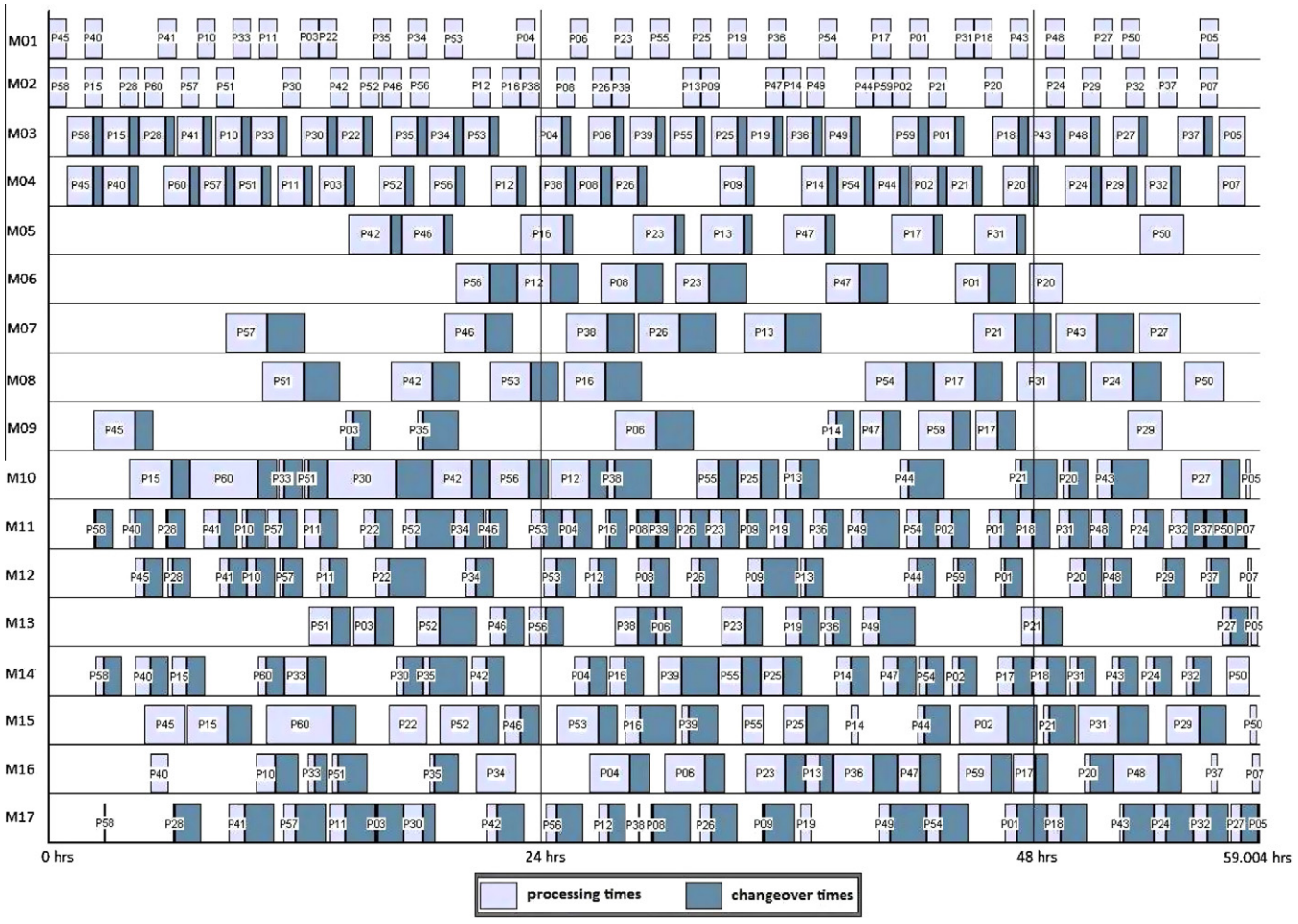

Fig. 9. Best schedule for I.12 (60-product case: min. total operating and changeovers costs under ZW policy).

According to Table 4, it is evident that the proposed MIP-based solution strategy overwhelms the original MIP models. By using our approach, highly complicated scheduling problems in multiproduct multistage batch plants can be solved, as the current experimental study reveals. Although optimality cannot be guaranteed, feasible solutions can be obtained in relatively short computational time. Bear in mind that feasibility is the principal goal in scheduling. To the best of our knowledge, no other standard solution methods nor heuristics exist for tackling the studied scheduling problem efficiently.

To conclude, some representative Gantt charts of the best schedules for some of the problem instances are included in order to provide the reader with a visual demonstration of the complexity of the addressed problems. Concretely, Fig. 7 presents the best solution found for solving the 30-product case by minimizing C_{max} under UIS policy (problem instance I.01). Fig. 8 shows the best schedule reported for solving the 60-product case by minimizing total weighted lateness under UIS policy (problem instance I.07). Finally, Fig. 9 depicts the best schedule found for solving the 60-product case by minimizing total operating and changeovers costs under ZW storage policy (problem instance I.12).

7. Final considerations

An iterative two-step MIP-based solution strategy has been presented for the resolution of large-scale scheduling problems in multiproduct multistage batch plants. Besides, a benchmark scheduling problem in a multiproduct multistage pharmaceutical batch plant has been introduced in this work. The proposed solution technique is able to generate feasible and good solutions in relatively short time, as the several problem instances of the pharmaceutical scheduling problem reveal. Also, have in mind that the user can appropriately define the degrees of freedom of the decision variables by balancing the trade-off between computational time and solution quality. The solver search space is reduced by degenerating the degrees of freedom of the decision variables thus favoring the faster resolution of the scheduling problem; reducing probably the solution quality. The proposed solution strategy can be also applied to other types of scheduling problems by adopting a different MIP model that describes the particular scheduling problem. Moreover, this work aims to be a step towards reducing the gap between scheduling theory and practice, since it has clearly demonstrated that real-world industrial problems can be solved by using effective MIP-based optimization solution strategies. The results obtained are promising and the further enhancement of the proposed solution method seems a challenging future research task. Other authors are encouraged to validate and compare their solution methods in the large-scale benchmark pharmaceuticals scheduling problem. In particular, comparisons of the proposed solution method with elaborate metaheuristics would be of great interest.

Acknowledgments

Financial support received from the Spanish Ministry of Education (FPU Grant) and the European Community (FEDER) (Projects DPI2006-05673; DPI2009-09386) from AECID under Grant PCI-D/024726/09, from FONCYT-ANPCyT under Grant PICT 2006-01837, from CONICET under Grant PIP-2221 and from UNL under Grant PI-66-337 is fully appreciated. Authors would also like to express their gratitude to Dr. Iiro Harjunkoski from ABB Corporate Research Center for providing them with valuable suggestions and information to improve this contribution.

Appendix

Algorithm 1. Pseudo-code for iterative procedure in constructive step

Set step = 1, initial = 1 & pos(i) parameter. Also, set $I^{in} = \emptyset$
FOR z = initial to card (i) by step
  LOOP i
    IF pos (i) $\leqslant$ z
      I^{in} = yes
    END IF
  END LOOP
  SOLVE MIP model
  fix Y_{isj} & $X_{ii'j}$ binary variables $\forall i \in I^{in}$
END FOR

Algorithm 2. Pseudo-code for iterative procedure in improvement step

⟨⟨ Refer to Méndez and Cerdá (2003) for an explanation of parameter n, subsets IS_i, $ISS_{ii'}$, and Reordering-MIP model.⟩⟩
Set $iter^{max}$, n = 1, order (i), reins = 1 parameters & IS_i subset
$fixY_{isj} = Y_{isj}$, $fixX_{ii'j} = X_{ii'j}$ (solution of constructive step)
iteration = 1
WHILE (OF^{reins} is better than OF^{reord} or iteration = 1)
  → *Reordering Stage*
  $Y_{isj} = fixY_{isj}$
  $iter$ = 1
  WHILE (OF_{iter} better than OF_{iter-1} and $iter \leqslant iter^{max}$)
    CLEAR $ISS_{ii'}$ subset
    assess $ISS_{ii'}$ subset
    CLEAR all variables apart from Y_{isj}
    SOLVE Reordering-MIP model
    $iter = iter + 1$
  END WHILE
  save best solution of reordering stage:
  CLEAR $fixX_{ii'j}$, and set $fixX_{ii'j} = X_{ii'j}$ & $OF^{reord} = OF_{iter-1}$
  → *Reinsertion Stage*
  iter = 1
  FOR z = reins to card (i) by reins
    LOOP i
      IF order (i) $\leqslant$ z-reins + 1
        CLEAR all variables related to i (e.g., $Y_{isj}, X_{ii'j}, C_{is}$, etc.)
      ELSE
        $Y_{isj} = fixY_{isj}$ & $X_{ii'j} = fixX_{ii'j}$
      END IF
    END LOOP
    SOLVE MIP model
    IF OF_{iter} is better than OF^{reord}
Save Solution (iter) (e.g., save $OF, Y_{isj}, X_{ii'j}, C_{is}$, etc.)
    END IF
    $iter = iter + 1$
  END FOR
  save best solution of reinsertion stage:
  CLEAR $fixY_{isj}, fixX_{ii'j}$ & OF^{reins}
  OF^{reins} is equal to the best OF_{iter}
  set $fixY_{isj} = Y_{isj}$, $fixX_{ii'j} = X_{ii'j}$ only for $OF_{iter} = OF^{reins}$
  iteration = iteration + 1
END WHILE

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2010.06.002.

References

Bassett, M.H., Pekny, J.F., Reklaitis, G.V., 1997. Using detailed scheduling to obtain realistic operating policies for a batch processing facility. Industrial and Engineering Chemistry Research 36, 1717–1726.

Birewar, D., Grossmann, I.E., 1990. Simultaneous planning and scheduling of multiproduct batch plants. Industrial and Engineering Chemistry Research 29, 570–580.

Blömer, F., Günther, H., 2000. LP-based heuristics for scheduling chemical batch processes. International Journal of Production Research 38, 1029–1051.

Brooke, A., Kendrik, D., Meeraus, A., Raman, R., Rosenthal, R.E., 1998. GAMS-A User's Guide. GAMS Development Corporation, Washington.

Castro, P.M., Harjunkoski, I., Grossmann, I.E., 2009. Optimal short-term scheduling of large-scale multistage batch plants. Industrial and Engineering Chemistry Research 48, 11002–11016.

Cerdá, J., Henning, G.P., Grossmann, I.E., 1997. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Industrial and Engineering Chemistry Research 36, 1695–1707.

Franca, P.M., Gendreau, M., Laporte, G., Muller, F.M., 1996. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. International Journal of Production Economics 43, 78–89.

Graves, S.C., 1982. Using Lagrangean techniques to solve hierarchical production planning problems. Management Science 28, 260–275.

Gupta, A., Maranas, C.D., 1999. A hierarchical Lagrangean relaxation procedure for solving midterm planning problems. Industrial and Engineering Chemistry Research 38, 1937–1947.

Herrmann, J., 2006. Handbook of Production Scheduling. Springer Science + Business Media, Inc., New York, NY 10013, USA.

Kopanos, G.M., Laínez, J.M., Puigjaner, L., 2009. An efficient mixed-integer linear programming scheduling framework for addressing sequence-dependent setup issues in batch plants. Industrial and Engineering Chemistry Research 48, 6346–6357.

Méndez, C.A., Cerdá, J., 2003. Dynamic scheduling in multiproduct batch plants. Computers and Chemical Engineering 27, 1247–1259.

Méndez, C.A., Henning, G.P., Cerdá, J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Computers and Chemical Engineering 25, 701–711.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. Review: state-of-the-art of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering 30, 913–946.

Nawaz, M., Enscore, E.E., Ham, I., 1983. A heuristic algorithm for the *m*-machine *n*-job flow-shop sequencing problem. OMEGA, The International Journal of Management Science 11, 91–95.

Pacciarelli, D., 2002. The alternative graph formulation for solving complex factory scheduling problems. International Journal of Production Research 40, 3641–3653.

Pinedo, M., Chao, X., 1999. Operations scheduling with applications in manufacturing and services. McGraw Hill International Editions.

Pinto, J.M., Grossmann, I.E., 1995. A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. Industrial and Engineering Chemistry Research 34, 3037–3051.

Raaymakers, W.H.M., Hoogeveen, J.A., 2000. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. European Journal of Operational Research 126, 131–151.

Röslof, J., Harjunkoski, I., Björkqvist, J., Karlsson, S., Westerlund, T., 2001. An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. Computers and Chemical Engineering 25, 821–828.

Röslof, J., Harjunkoski, I., Westerlund, T., Isaksson, J., 2002. Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure. European Journal of Operational Research 138, 29–42.

Ruiz, R., Maroto, C., 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 169, 781–800.

Ruiz, R., Stutzle, T., 2008. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. European Journal of Operational Research 187, 1143–1159.

Ruiz, R., Serifoglu, F.S., Urlings, T., 2008. Modeling realistic hybrid flexible flowshop scheduling problems. Computers and Operations Research 35, 1151–1175.

Venditti, L., Pacciarelli, D., Meloni, C., 2010. A tabu search algorithm for scheduling pharmaceutical packaging operations. European Journal of Operational Research 202, 538–546.

Werner, F., Winkler, A., 1995. Insertion techniques for the heuristic solution of the job shop problem. Discrete Applied Mathematics 58, 191–211.