CHEMICAL ENGINEERING TRANSACTIONS Volume 21, 2010
Editor J. J. Klemeš, H. L. Lam, P. S. Varbanov
 ISBN 978-88-95608-05-1 ISSN 1974-9791
DOI: 10.3303/CET1021086

Solving Scheduling Problems in a Multi-stage Multi-product Batch Pharmaceutical Industry

Georgios M. Kopanos[1], Carlos A. Méndez[2], and Luis Puigjaner[1]*
[1]Department of Chemical Engineering, Universidad Politècnica de Catalunya, ETSEIB, Av. Diagonal 647, 08028 Barcelona, Spain
luis.puigjaner@upc.edu
[2]INTEC (UNL-CONICET), Argentina

An iterative two-stage decomposition solution strategy for solving real-world scheduling problems in multi-stage multi-product batch plants is presented. The proposed method has as a core a mixed integer mathematical model, and consists of a constructive step, wherein a feasible and good solution is rapidly generated by following some insertion criteria, and an improvement step, wherein the initial solution is systematically enhanced by adopting several rescheduling techniques. The proposed strategy performance is tested on a number of problem instances of a complicated real-world multi-stage multi-product pharmaceuticals scheduling problem. High quality solutions are reported within reasonable computational time.

1. Introduction

Since most industrial scheduling applications are commonly modelled as large-scale combinatorial and complex optimization problems, they rarely can be solved to optimality within a reasonable amount of computational time. Thus, in industrial environments, computational time becomes an issue as important as the scheduling problem itself; since industrial problems require optimal solutions, or at least close to optimal, that can be reached in the shortest possible time. As a consequence, heuristic or meta-heuristic techniques have been employed in order to reduce the inherent computational burden. For instance, genetic algorithms, simulated annealing, tabu search, particle swarm and ant colony optimization methods have been widely utilized in a variety of scheduling problems. However, although the aforementioned methods may generate solutions in short computational time, they cannot provide the reliable behaviour required for hard restricted industrial environments.
In order to make rigorous mathematical-based methods more attractive for real-world applications, increasing effort has been oriented towards the development of systematic techniques that allow maintaining the number of decision variables at a reasonable low 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 behaviour. Furthermore, once the best possible feasible solution has been generated in short time, systematic optimization-based methods can be employed to

Please cite this article as: Kopanos G. M., Méndez A. and Puigjaner L., (2010), Solving scheduling problems in a multi-stage multi-product batch pharmaceutical industry, Chemical Engineering Transactions, 21, 511-516 DOI: 10.3303/CET1021086

gradually enhance a non-optimal solution with low computational effort. An apparent drawback of these techniques is that optimality can no longer be assured. Nevertheless, achieving rigorous optimality may not be specially significant in many industrial scenarios mainly due to the following: (i) only a short time is usually available to generate a solution and get it launched in the plant floor, (ii) optimality is easily lost because of the highly dynamic nature of industrial environments, (iii) implementing the schedule as such is constrained by the real process, and (iv) the real scheduling goals are only partly taken into account since not all scheduling objectives can be quantified.

2. Mathematical Frameworks

Mixed Integer Programming (MIP) models constitute the core of the proposed solution method. In this study, two different batch-oriented MIP formulations are used. Both models are based on a continuous-time domain and utilize sequencing variables. The first MIP model is based on the global precedence sequencing concept, and can be found in Méndez and Cerdá (2003). Notice that global precedence formulations result in models with small model size and they are computationally faster on average in comparison with the other batch oriented frameworks (Méndez et al., 2006). However, a drawback of global precedence models is that they cannot optimize objectives containing sequence-dependent changeover issues (such as costs, etc.). For this reason, a new unit-specific global-direct precedence framework, for scheduling multi-stage multi-product batch plants, has been recently developed by Kopanos et al. (2009), as a general mathematical formulation, which is able to cope with any objective function.

3. 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. Therefore, in this work we propose a solution strategy for solving large-scale scheduling problems. 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 Figure 1).

3.1 Constructive step

The constructive step is performed in an iterative mode. A predefined number of product orders are scheduled at each iteration with lower degrees of freedom; until all product orders are finally scheduled. The number of orders used at each iteration should be small enough to ensure the quick resolution of each iteration, and thus generating a

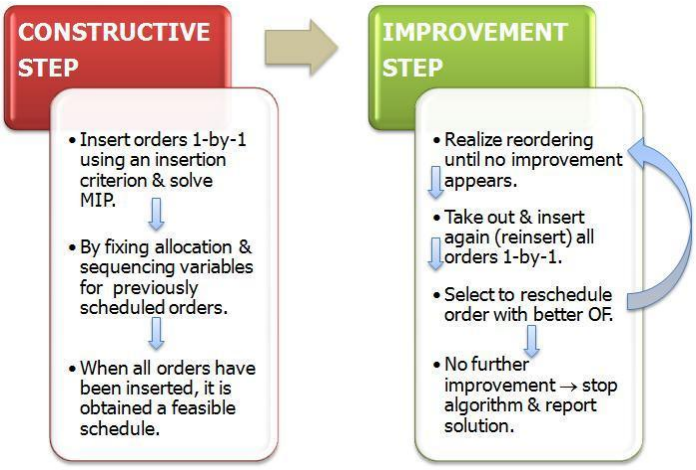

Figure 1: Representative scheme of the proposed MIP-based solution strategy.

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) do not guarantee a better constructive step schedule, and (ii) it is more computationally expensive.
After each iteration, allocation and global sequencing binary variables for the previously scheduled product orders are fixed in order to reduce the computational effort. In other words, unit allocation decisions and relative sequencing relations of the already scheduled product orders cannot be modified in the following iterations. However, timing decisions may change thus permitting the insertion of new inserted product orders among the previously scheduled product orders. When all product orders have been inserted, a feasible schedule can be finally obtained in relatively short time.

3.2 Improvement step

In this step, the initial feasible schedule provided by the constructive step is further improved through reordering and/or reassignment operations; in accordance with the main rescheduling concepts of the work presented by Mendez and Cerdá (2003). The improvement step, which follows a two-stage closed loop procedure, consists of the reordering stage and the reinsertion stage, which are performed sequentially until no improvement is observed.

3.2.1 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. 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.

3.2.2 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 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. 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.

4. Pharmaceutical Production Process

In the current study, the short-term scheduling problem of a considerably high number of multistage product orders (30 to 60) in the 17 processing units of the pharmaceuticals production plant is addressed. The production process has 6 processing stages. Some products bypass the third processing stage. 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.

5. Case Study: Results and Discussion

Twelve problem instances considering a different number of products have been solved: (i) 30 product orders (168 batches), and (ii) 60 products (336 batches). Different storage policies (Zero Wait (ZW), Unlimited Intermediate Storage (UIS)) and objective functions (makespan (MK), weighted (WL) lateness, operating and changeover costs (O&C)) have been also considered. A time limit of 1 CPU hr 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 2GB RAM using CPLEX 11 via a GAMS 22.8 interface (Brooke et al., 1998).

Table 1: Best schedules found within the predefined time limit (3600 CPU s).

problem instance	objective function	batches	storage policy	1st-stage solution	1st-stage CPU s	best solution	total CPU s	impove-ment
I.01	MK	168	UIS	28.507	38	26.559	542	6.83%
I.02	MK	168	ZW	31.520	7	30.532	187	3.14%
I.03	MK	336	UIS	49.161	155	48.548	1502	1.25%
I.04	MK	336	ZW	58.104	106	56.061	1718	3.52%
I.05	WL	168	UIS	48.613	22	19.085	720	60.74%
I.06	WL	168	ZW	115.016	15	84.438	262	26.59%
I.07	WL	336	UIS	118.683	403	87.943	3600	25.90%
I.08	WL	336	ZW	629.672	356	515.876	1478	18.07%
I.09	O&C	168	UIS	66.158	94	62.910	3600	4.91%
I.10	O&C	168	ZW	72.318	58	70.209	3600	2.92%
I.11	O&C	336	UIS	119.759	1780	117.909	3600	1.54%
I.12	O&C	336	ZW	139.104	880	134.624	3600	3.22%

Table 1 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 is 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 s and no more than 7 CPU min.

Note that the original un-decomposed MIP models were unable to solve even small instances of the pharmaceuticals case study, thus highlighting the practical benefits of our approach. It 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.

A representative Gantt chart of the best schedule for problem instance I.12 is shown in Figure 2 in order to provide the reader with a visual demonstration of the complexity of the addressed problems.

6. Conclusions

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 initial 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.

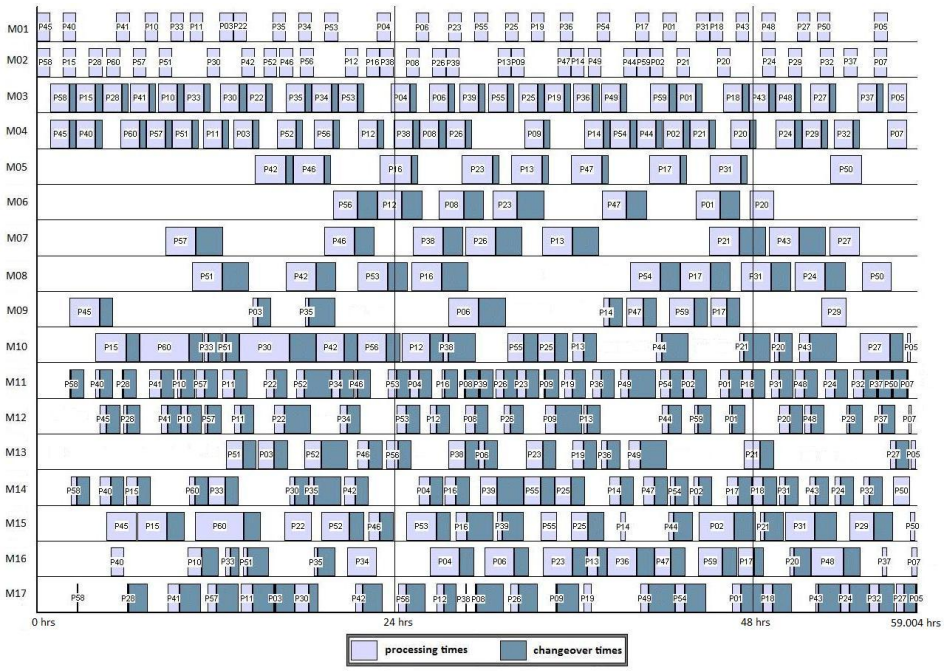

Figure 2. Best schedule for I.12 (60-product case: min. total operating and changeovers costs under ZW policy).

Acknowledgements

Financial support received from the Spanish Ministerio de Ciencia e Innovación (FPU grant and research projects DPI2006-05673 and DPI2009-09386) are gratefully acknowledged. Funding from the European Commission (FEDER) is also appreciated. Support received from AECID under Grant PCI-D/024726/09 is also gratefully acknowledged.

References

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

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

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

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I. and 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.