Efficient MILP-based solution strategies for large-scale industrial batch scheduling problems

Pedro Castro[a], Carlos Méndez[b], Ignacio Grossmann[c], Iiro Harjunkoski[d], Marco Fahl[d]

[a]*DMS/INETI, 1649-038 Lisboa, Portugal*
[b]*Chemical Engineering Department-CEPIMA, UPC, E-08028 Barcelona, Spain*
[c]*Dep. Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[d]*ABB Corporate Research Center, Ladenburg, Germany*

Abstract

This paper presents two alternative decomposition approaches for the efficient solution of multistage, multiproduct batch scheduling problems comprising hundreds of batch operations. Both approaches follow the principle of first obtaining a good schedule (constructive stage), by considering only a subset of the full set of orders at a time, and then improving it (improvement stage) by applying a rescheduling technique. The core of both approaches consists on the solution of mixed integer linear programming problems that, on each step, are variations of the scheduling model with global precedence sequencing variables of Harjunkoski & Grossmann (2002). The results for the solution of a 30-order problem show that the proposed decomposition methods are able to obtain solutions that are 35% better than those obtained by the solution of the full problem, on a fraction of the computational time.

Keywords: Decomposition methods, short-term scheduling, rescheduling.

1. Introduction

The increasingly large literature in the scheduling area highlights the successful application of different optimization approaches to an extensive variety of challenging problems (Méndez et al., 2005). This important achievement comes mainly from the remarkable advances in modeling techniques, algorithmic solutions and computational technologies that have been made in the last decade or so. However, there is still a significant gap between theory and practice. New academic developments are mostly tested on relatively small problems whereas real-world applications consist of hundreds of batches, dozens of pieces of equipments and long scheduling periods. In order to make the use of exact methods more attractive for 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. Manageable model sizes may be obtained by applying heuristic model reduction methods, decomposition or aggregation techniques. Once an initial solution is generated with reasonable CPU time, gradual improvement through optimization-based techniques can be achieved with modest computational effort. Although these techniques can no longer guarantee optimality, this may not be so critical in practice due to the following: i) very short time is available to generate a solution; ii) optimality is easily lost due to the dynamic nature of industrial environments; iii) implementing the schedule as such is often limited by the real process; iv) only a subset of the actual scheduling goals are taken into account.

Recent work by Castro & Grossmann (2005) compared the performance of five different approaches for the solution of multistage batch scheduling problems. The results showed that all, besides a uniform time grid continuous-time formulation, can be a valid option depending on the objective function. However, the study was performed for relatively small problems and sequence-dependent changeovers were not considered. Handling industrial-sized problems may quickly lead to an intractable model size. Also, the inclusion of sequence-dependent changeover times may substantially reduce the efficiency of most of the existing models. Following a preliminary evaluation of the most adequate formulation, this paper presents two alternative decomposition strategies that are able to efficiently deal with multistage, multiproduct scheduling problems involving large number of batches, many processing units and substantial cleaning requirements, typically found in the pharmaceutical and fine chemical industry.

2. Problem definition

The industrial multiproduct plant under consideration has the following characteristics:

- 17 machines and 6 stages, allocated in the following way: 2+3+3+3+3+3.
- Unlimited intermediate storage between stages is assumed
- Sequence dependent changeovers in most stages, all of them machine independent. Changeovers are usually of the same order of magnitude or even larger than the processing times.
- Some orders may skip some processing stages.
- Two sets of problem data are considered: i) a 30-order and ii) a 50-order problem.
- The objective is the minimization of the makespan.

3. Selection of the most appropriate formulation

Discrete-time formulations are not appropriate for this specific problem. On the one hand, the objective of makespan minimization can only be achieved through an indirect procedure that involves solving several problems (Maravelias & Grossmann, 2003). On the other hand, sequence dependent changeovers lead to a significant increase in the model size, both directly and indirectly. Directly, due to the increase in the number of constraints to account for the cleaning times. Indirectly, as a result of the larger number of time intervals and the consideration of a finer time grid for an exact representation of the problem data, while keeping the model sensitive to different changeovers.

Continuous-time formulations based on a uniform time grid are generally inefficient for multistage problems without shared resources (Castro & Grossmann, 2005). Continuous-time formulations relying on multiple time grids have more flexibility and have proved to be more competitive for this type of problem. However, sequence dependent changeovers make them less efficient. Continuous-time formulations using explicit sequencing variables are more valid approaches since they hardly require any changes to handle sequence dependent changeovers. The model used in this paper is the one by Harjunkoski & Grossmann (2002), which relies on global precedence sequencing variables. The authors also presented a constraint programming (CP) model, which can be easily adapted to the case of sequence dependent changeovers. One just needs to assign to each unary resource (the machines), the corresponding changeover matrix (using ILOG's OPL Studio).

Table 1 shows the computational statistics for the solution of the 30-order problem with the continuous-time formulation (MILP) and the CP model. As can be seen, the size of the problem is already too large to be handled efficiently. While the MILP finds a fair solution in 1 h (maximum computational limit), it has many binary variables and features a very large integrality gap (the best possible solution at the time of interruption is still only 13.325, which translates into an absolute gap of 34.657 and a relative integrality gap above 72%). CP, one the other hand, was also tested but it was only able to generate a very poor solution at the beginning of the search (in the very first second of calculation). No further improvements were observed in the remaining hour. Hence, the continuous-time MILP will be the only one considered next.

Table 1. Computational statistics for the solution of the full 30-order problem

Model	Bin. vars.	Cont. vars.	Cons.	RMIP	MIP	Best possible	CPUs	Nodes
MILP	2521	2708	10513	7.449	47.982	13.325	3600	62668
CP	-	1050	1300	-	79.989	-	3600	1432

4. Solution Strategy

Since it is impractical to solve the full problem simultaneously, effective decomposition techniques need to be developed. Of a few alternatives tested, two have shown to produce very good results in a relatively small amount of computational time. Both approaches follow a two-step approach. The first, is a constructive step, where the goal is to obtain a good schedule with low computational effort. The second, is an improvement step, where the previous solution is gradually enhanced by applying a rescheduling technique with low computational effort. Bear in mind that the improvement step is common to both approaches.

4.1. Constructive approach 1 (AP1)

The first approach follows a two step decomposition method consisting of a design model and a schedule refinement model. For the design model, we need to define a priori the number of orders to be considered (up to the full set of orders) and the number of orders to be selected, which will be typically around 5 (i.e. for 30 orders, 6 iterations will be required) to ensure a small integrality gap and reaching a good solution very fast in the refinement model. It is very similar to the original model of Harjunkoski & Grossmann (2002) but minor changes were required: i) Introduction of a new set of binary variables, to identify the selection of a particular order from the subset of orders considered in the design problem; ii) Introduction of a new set of constraints, to enforce an even distribution of the most difficult orders (these are high-processing-time orders that can only be assigned to a single machine on a given stage) through the several iterations instead of getting them all in the last iteration, thus avoiding a significant increase in the value of the objective function, the minimization of the makespan.

While the design model generally provides a good solution, it is usually unable to prove optimality within the specified computational limit. If we deal only with the selected orders, a better solution can typically be found. The objective function will minimize the makespan plus a term that accounts for the completion times in all stages, since we need to link two consecutive iterations efficiently. By doing this, we ensure that the resulting schedule is tight for all machines and not just for the ones belonging to the bottleneck path(s) that define the makespan. Following the solution from one iteration, the times at which the several machines end their processing can be determined and used to fix the machines release dates for the subsequent iteration. Furthermore, the last order to be processed can be identified and the information used to account for the correct changeover time.

4.2. Constructive approach 2 (AP2)

The second approach also divides the full set of orders into smaller subsets. Here, the process of assigning orders to iterations does not require any special method, since there is much more freedom to link consecutive iterations. We can choose the set of order(s) to be considered in each iteration by following the lexicographic sequence. Besides the elimination of the design step, there are two important conceptual differences to AP1. i) Orders that have been scheduled in previous iterations (typically 1 or 2 per iteration) are no longer removed from further consideration. Instead, they will be considered over and over again, although with fewer degrees of freedom, until the last iteration; ii) Unit availability is no longer limited in time by using release dates. Previously assigned orders can be left- or right-shifted in time in order to insert the new order(s) to be scheduled. While sequencing and allocation decisions are made for the latter set of orders, for orders belonging to the former set, the binary assignment (they remain allocated to the same machine) and sequencing variables (they maintain their relative position in the sequence) are fixed but the continuous timing variables may change.

This simple method can be illustrated by a simple example. Suppose that the previous iterations resulted in order sequence 4, 3, 2, at a given machine, and that a new order needs to be scheduled (order number 6). Fig. 1 shows that there are only 4 allowed sequences.

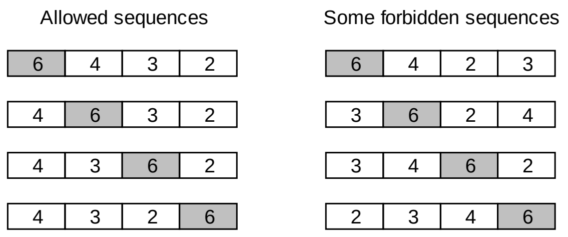

Figure 1. Illustration of the scheduling step whenever a new order (6) is being considered

4.3. Improvement step

The proposed improvement technique is based on the main ideas of the rescheduling model proposed by Roslöf et al. (2001) and Méndez & Cerdá (2003) and can be combined with either of the two previous alternative constructive methods. It involves several iterations, with each being identical to the last iteration of AP2, where all orders except a very small subset of them (e.g. a single order) have fixed assignments and fixed relative positions. The selection of orders among iterations can be made following some kind of sequence or be made randomly. Every time a better makespan is achieved, the allocation (equipment) and position (sequence) tables are updated. The stopping criteria for the rescheduling algorithm can be either a maximum predefined computational limit or a maximum number of iterations without improving the objective function value.

5. Computational results

The computational studies were performed on a Pentium 4-2.8 GHz running GAMS/CPLEX 9.0. Both methods have several parameters that can affect the final result. In AP1, the number of orders to be considered (NPD) and selected (NPS) in the design problem are critical. For AP2, NPS defines the number of orders scheduled simultaneously. Common parameters include the maximum CPU-time for the scheduling (actually in AP1 two values are specified, one for the design step and other for the schedule refinement step) and rescheduling phases and the method of order selection in the latter (direct or random sequence). Table 2 presents some of the most successful runs. The results show that both approaches lead to very good solutions when compared to those obtained using the full MILP (see Table 1). Usually, a better solution from the scheduling phase (6th column), leads to a better final solution after the rescheduling step (7th column for the direct sequence and 8th column for the random sequence). Also, for a given phase, the more the computational time the better the solution (compare rows 3 and 4, and 10 and 11). It was found that AP2 is more robust since the solution obtained is less dependent on the model parameters and is often better. However, the number of orders to be considered increases steadily with the number of iterations, contrary to AP1, so while the first iterations take little time, the last ones can have significantly large integrality gaps when the resource limit is reached, which may eventually compromise the quality of the final solutions. Despite this fact, AP2 performs better for the 50-order problem than for the 30-order problem.

Table 2. Computational results

				CPU-limit (s)	Makespan (h)			CPUs	
Orders	Ap.	NPD	NPS	Per phase	Sch.	Res. Dir.	Res. Ran.	Sch.	Total
30	AP1	30	6	(25;15);(10)	32.439	**30.480**	30.562	176	276
30	AP1	30	5	(20;10);(10)	33.592	31.811	32.316	161	266
30	AP1	30	5	(10;10);(10)	35.578	33.084	33.610	112	212
30	AP1	10	5	(10;10);(10)	34.229	32.192	32.898	110	217
30	AP2	-	1	(10);(10)	33.149	31.175	31.676	86.4	188
30	AP2	-	2	(20);(10)	32.523	31.007	31.678	175	275
30	AP2	-	3	(30);(10)	34.447	31.787	32.485	255	355
50	AP1	30	6	(25;15);(10)	56.082	51.997	54.269	322	422
50	AP1	30	5	(20;10);(10)	53.740	53.409	52.800	283	391
50	AP1	30	5	(10;10);(10)	56.228	53.684	55.131	194	295
50	AP1	30	3	(10;5);(10)	53.453	51.466	51.581	173	275
50	AP2	-	1	(10);(10)	52.911	51.275	**50.721**	240	342
50	AP2	-	2	(15);(10)	52.964	51.080	51.019	321	429
50	AP3	-	3	(20);(10)	55.705	52.960	52.668	306	407

The best solution to the 30-order problem, featuring a makespan of 30.480 h, is given in Fig. 2. Note that the completion times of the last-stage machines (M15-M17) are very similar, with M16 setting the makespan. It is hardly a trivial schedule due to the following reasons: i) the length of changeover periods and idle times exceed by far the lengths of the production times, in the last three stages of production; ii) the order sequence changes often from one stage to the other (e.g. order 17 is processed prior to 27 in stage 1, M1, while the opposite is true in the last stage, M16); iii) despite the plant consisting of the same number of machines per stage after stage 1 (i.e. 3), it is unreasonable to consider independent parallel production lines since the subset of orders assigned to a machine in a particular stage does not remain constant throughout the subsequent stages (e.g. of the 8 orders assigned to M13 in stage 5, 1 (20) goes to M15, 2 (25 and 29) go to M16 and 5 go to M17); iv) even after a thorough examination of the schedule, there is not a clear limiting stage, which means that the bottleneck probably shifts repeatedly between different stages and even between machines of the same stage.

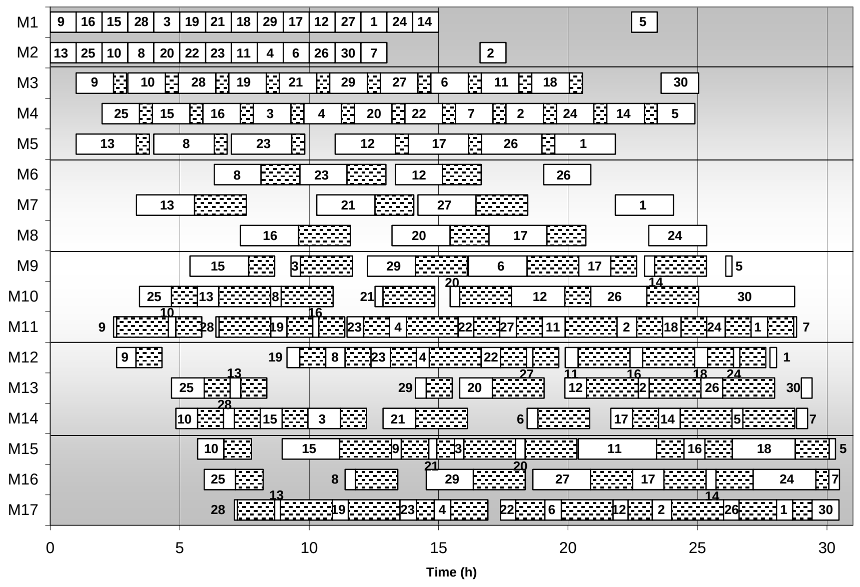

Figure 2. Best solution found for the 30-order problem

6. Conclusions

This paper has described two alternative decomposition approaches for the efficient and fast solution of large industrial scheduling problems. Both use the concept of decomposing the full problem into several subproblems, each featuring a subset of the orders. The main difference lies in linking the consecutive subproblems. While the first approach completely freezes the schedule of the pre-assigned orders and ensures feasibility for the remaining through machine release dates, the second approach allows for more flexibility by only fixing the assignments and relative positions of the previously scheduled orders. The second approach was found to be more robust and seems better suited for the solution of this specific type of problem.

References

Castro, P., Grossmann, I., 2005. New Continuous-Time MILP Model for the Short-Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res. In press.

Harjunkoski, I., Grossmann, I., 2002. Comp. Chem. Eng., 26, 1533.

Maravelias, C., Grossmann, I., 2003. Ind. Eng. Chem. Res., 42, 6252.

Méndez, C., Cerdá, J., 2003. Comp. Chem. Eng., 27, 1247.

Méndez, C., Cerdá, J., Grossmann, I., Harjunkoski, I., Fahl, M., 2005. State-of-the-art Review of Optimization Methods for Short-Term Scheduling of Batch Processes. Submitted to Comp. Chem. Eng.

Roslöf, J., Harjunkoski, I., Björkqvist, J., Karlsson, S., Westerlund, T., 2001. Comp. Chem. Eng., 25, 821.