The Eighth International Symposium on Operations Research and Its Applications (ISORA'09)
Zhangjiajie, China, September 20–22, 2009
Copyright © 2009 ORSC & APORC, pp. 132–139

MIP-Based Approaches for Solving Scheduling Problems with Batch Processing Machines

Udo Buscher[*] Liji Shen[†]

Chair of Industrial Management, Department of Business Administration and Economics, Dresden University of Technology, 01062 Dresden, Germany

Abstract In this paper, we address the scheduling problem involving batch processing machines. The presented mixed integer programming formulation first provides an elegant model for the problem under study. Furthermore, it enables solutions to the problem instances beyond the capability of exact methods developed so far. In order to alleviate computational burden, we propose MIP-based approaches which balance solution quality and computing time.

Keywords Scheduling; Batching decisions; Mixed integer programming

1 Introduction

In the last decades, numerous researchers have developed significant interest in scheduling problems with *batching* decisions, the benefits of which consist in a remarkable reduction of setup times as well as improved throughput rate [6]. In this regard, one distinct situation for applying batching is characterized by the adoption of batch processing machines in the production system, which is also known as *parallel batching*. A *batch processing machine* can accommodate and process several jobs simultaneously in a batch. *Batch processing times* usually depend on the jobs constituting the corresponding batch. These machines are integral parts of many industrial environments such as burn-in operations in a steel foundry, environmental stress screening chambers, electrical circuits tests, and chemical processes performed in tanks or kilns [3]. As another example, the wafer fabrication process being one of the most important components in semiconductor manufacturing can also be modelled as shop floor process with various inter-connected batch processing machines, so that wafer-lots are transferred through stages including oxidation, deposition, diffusion, etching and ion implantation.

To date, parallel batching related publications mostly focus on single machine cases. There are several reference works in the literature considering flow shop problems with limited number of batch processing machines. [1] addresses two- or three-stage flow shop problems with one batch processing machine involved. A full complexity classification with the performance measures of the total completion times and the makespan is provided as well. [7] considers the problem of scheduling independent jobs on two batch processing machines in an open shop, a job shop and a flow shop environment. There is

[*]buscher@rcs.urz.tu-dresden.de
[†]liji.shen@mailbox.tu-dresden.de

either no restriction on the size of a batch on a machine, or a machine can process only a bounded number of operations in one batch. For most of the possible combinations of restrictions, the authors provide the complexity status of the associated problem.

Motivated by practical relevance, the purpose of our study is to solve the scheduling problem involving a flow shop of batch processing machines. The coming section first presents two MIP formulations complete with a precise evaluation. In solving larger instances with limited computational resource, various MIP-based approaches are proposed and tested. Some concluding remarks are given in section 5.

2 MIP Formulations

[4] and [5] present MIP formulations for the flow shop problem with two batching machines, where solutions are restricted to consistent batches and permutation schedules. That is, the operations incorporated in a batch remain unchanged over all machines. In both studies, a binary variable is used which indicates whether or not a batch is assigned to a specific position in the sequence. However, this is unsuitable while considering multi-machine systems with non-permutation schedules. Therefore, a new binary variable is introduced with

$$\gamma_{bck} = \begin{cases} 1 \text{ , if batch } b \text{ is processed prior to batch } c \text{ on } M_k, \\ 0 \text{ , otherwise.} \end{cases}$$

Other related parameters and variables are given in the following:

$b, c = 1, \ldots, \bar{b}$	batch indices
$i = 1, \ldots, n$	job index
$k = 1, \ldots, m$	machine index
B_{bk}	batch b on machine k
C_{max}	makespan
p_{ik}	processing time of job i on machine k
S	machine capacity (the maximal number of jobs incorporated in a batch)
t_{bk}	start time of batch b on machine k
t'_{ik}	start time of job i on machine k
T_{bk}	end time of batch b on machine k
δ_{ibk}	binary variable, takes the value 1 if job i belongs to batch b on machine k

2.1 Adopting Consistent Batches

Flow shop problems involving batch processing machines can thus be formulated as follows:

$$\min C_{max}. \tag{1}$$

Subject to:

$$\sum_{i=1}^{n} \delta_{ib} \leq S \quad \forall b, \tag{2}$$

$$\sum_{b=1}^{\bar{b}} \delta_{ib} = 1 \quad \forall i, \tag{3}$$

$$T_{bk} \geq t_{bk} + \max_{i} \{\delta_{ib} \cdot p_{ik}\} \quad \forall b, k, \tag{4}$$

$$t_{bk} \geq T_{ck} - \gamma_{bck} H \quad \forall b \neq c, k, \tag{5}$$

$$t_{ck} \geq T_{bk} - (1 - \gamma_{bck}) H \quad \forall b \neq c, k, \tag{6}$$

$$t_{b(k+1)} \geq T_{bk} \quad \forall b, k < m, \tag{7}$$

$$C_{max} \geq T_{bm} \quad \forall b. \tag{8}$$

As a result of requiring consistent batches, the binary variable δ_{ibk} reduces to δ_{ib}. Constraints (2) indicate that the maximal number of jobs incorporated in a batch does not exceed machine capacity S. Constraints (3) ensure that each job is assigned to one and only one batch. Since batch formation remains identical over all stages, the emphasis is then placed on determining batch sequences. According to (4), the completion time of batch b is determined by its start time and the largest processing time of all jobs belonging to batch B_{bk}. Constraints (5) and (6) are then employed to prevent batch overlapping. Constraints (7) describe the flow shop scheduling environment, where the processing of jobs follows the same technological order. In terms of constraints (8), makespan is defined as the largest completion time of all batches on the last machine m.

2.2 Adopting Inconsistent Batches

Based on the model presented above, a generalized formulation adopting inconsistent batches is given in the following:

$$\min C_{max} \tag{9}$$

$$\sum_{i=1}^{n} \delta_{ibk} \leq S \quad \forall b, k, \tag{10}$$

$$\sum_{b=1}^{\bar{b}} \delta_{ibk} = 1 \quad \forall i, k, \tag{11}$$

$$T_{bk} \geq t_{bk} + \max_{i} \{\delta_{ibk} \cdot p_{ik}\} \quad \forall b, k, \tag{12}$$

$$t_{(b+1)k} \geq T_{bk} \quad \forall k, b < \bar{b}, \tag{13}$$

$$t_{bk} \geq t'_{ik} \cdot \delta_{ibk} \quad \forall b, k, \tag{14}$$

$$t'_{i(k+1)} \geq T_{bk} \cdot \delta_{ibk} \quad \forall i, b, k < m, \tag{15}$$

$$C_{max} \geq T_{\bar{b}m}. \tag{16}$$

Constraints (10)–(12) are the simple extensions of constraints (2), (3) and (4). Moreover, binary variable γ_{bck} can be safely dropped, since batches with the same index (b) on different machines are allowed to accommodate different jobs. In order to avoid overlapping, constraints (5) and (6) can be reduced to (13). Although the sequence of batches is predetermined here, this formulation does not lose its generality since the contents of a batch may vary over stages.

Special attention must be drawn to the start time of batches. By the very nature of inconsistent batches, the start time of batch $B_{b(k+1)}$ is not necessarily restrained by the completion time of B_{bk}. Therefore, the start time of each batch depends on all jobs included in the corresponding batch. Constraints (15) determine the start time of each job on the successive machine. Combing with constraints (14), a batch thus cannot begin its processing until all jobs assigned to this batch become available. According to (16), makespan is then defined by the last batch scheduled on the last machine $(B_{\bar{b}m})$. In addition, the formulation can be applied to single machine cases by omitting machine index k.

2.3 Evaluating MIP formulations

An exact evaluation of the formulation with batching machines presented in the previous subsection can be found in table 1, in terms of the number of variables, constraints and nonzeros. Examples of several problem classes are given in table 2.

Table 1: Evaluation of the formulation with batching machines

Variables	Binary	$\bar{b}mn$	Real	$2\bar{b}m+mn+1$
Total	$\bar{b}mn+2\bar{b}m+mn+1$			
Constraints	(10)	$\bar{b}m$	(14)	$\bar{b}mn$
	(11)	mn	(15)	$\bar{b}n(m-1)$
	(12)	$\bar{b}m$	(16)	1
	(13)	$(\bar{b}-1)m$		
Total	$2\bar{b}mn+(3\bar{b}-1)m+mn-\bar{b}n+1$			
Nonzeros	(10)	$\bar{b}mn$	(14)	$3\bar{b}mn$
	(11)	$\bar{b}mn$	(15)	$3\bar{b}n(m-1)$
	(12)	$\bar{b}m(n+2)$	(16)	2
	(13)	$2(\bar{b}-1)m$		
Total	$9\bar{b}mn+(4\bar{b}-2)m-3\bar{b}n+2$			

3 MIP-Based Approaches

Since commercial solvers are only capable of handling small instances, it is of interest to generate satisfying solutions using limited computational resource. Based on the developed MIP-formulations, various relaxations and heuristic rules can be incorporated to reduce solution spaces and to accelerate solution progress.

Considering Permutation Schedules

Note that one of the most important decision variables in these formulations is the binary variable γ_{bck}. Therefore, a promising method consists in reducing the number of

Table 2: Examples of problem sizes

$m \cdot n$	$(\bar{b})$	Variables	Constraints	Nonzeros	Density
$3 \cdot 3$	(3)	55	81	252	0.0566
$4 \cdot 4$	(4)	113	176	592	0.0298
$5 \cdot 5$	(5)	201	325	1150	0.0176
$6 \cdot 6$	(6)	325	540	1980	0.0113
$10 \cdot 3$	(10)	391	620	2520	0.0104
$10 \cdot 4$	(10)	521	860	3460	0.0077
$20 \cdot 5$	(20)	2301	4000	17200	0.0019
$20 \cdot 6$	(20)	2761	4880	20880	0.0015

binary variables to be determined. In a permutation flow shop environment, job sequences remain consistent throughout all stages. Thus, imposing this assumption in the formulation reduces γ_{bck} to γ_{bc}. Actually, flow shop scheduling is greatly desired in numerous manufacturing industries [8]. The permutation schedule requirement in flow shops is also fairly realistic since it can be costly to change batch composition between machines [2].

Incorporating Upper and Lower Bounds

Since most commercial solvers are based on the principle of branch and bound method, upper and lower bounds can be introduced in the formulation so that solution space is effectively reduced. The efficiency of this approach is also confirmed in the literature [5].

Upper bounds can be calculated by applying heuristic rules or constructive algorithms, such as shortest processing time (SPT), longest processing time (LPT), earliest due date (EDD). Assigning jobs to batches may follow the first-first (FF) rule, in which the first available job is grouped into the first batch with sufficient capacity. Generally, such approach can be executed in $O(mn\log n)$ time. Large instances can also be solved in fraction of second.

On the other hand, dropping certain constraints can attain lower bounds on the underlying problem. For example, batch size violation may first be ignored. Preemption of jobs can be introduced by discarding constraints determining operation sequences. In addition, a multi-machine problem can be decomposed into single machine cases with properly defined parameters. Solving each single machine problem then provides a valid lower bound on the original problem.

Simplifying the Batching Phase

The batch formation problem is actually equivalent to a bin-packing problem which is also known to be $\mathcal{NP}$-hard [4]. It is therefore helpful to simplify the procedure of batching. Note that utilizing as much space as possible in a batch may maximize efficiency in flow shops. Therefore, jobs can be sorted in SPT order and assigned to batches according to their priorities. A new batch is constructed once the previous one is fully occupied. In consequence, jobs with similar processing durations are grouped into the same batch, which can effectively reduce machine idle times.

Simplifying the Scheduling Phase

Note that variable δ_{ibk} is predetermined due to the given partition of batches. This significantly reduces the complexity of the problem under study. Since job sequences within each batch are not decision relevant, the problem is actually transformed to finding an ordered collection of batches – the batch sequencing problem. The structure of the model is as follows:

$$t_{bk} \geq T_{ck} - \gamma_{bck} \cdot H \tag{17}$$

$$t_{ck} \geq T_{bk} - (1 - \gamma_{bck}) \cdot H. \tag{18}$$

4 Computational Results

The developed MIP formulation is implemented in Lingo 9.0. Although only small instances can be solved to optimality, the solvable problem size already exceeds the exact methods such as branch and bound, dynamic programming proposed in the literature.

As mentioned earlier, most existing studies focused on single machine cases, for which an optimal solution can be obtained efficiently using the proposed MIP formulation. Furthermore, the models available in the literature are only capable of handling two machine permutation flow shops. Our formulation, however, can be applied to general multi-machine problems.

First, consider a single machine example where 10 jobs are to be scheduled. Processing times are given in table 3. As illustrated in figure 1, if a batch is allowed to contain

Table 3: Data (example with 1 batching machine)

Job index	i	1	2	3	4	5	6	7	8	9	10
Processing time	p_i	2	4	6	6	8	2	4	10	4	2

at most 3 jobs, the maximal and total completion times are 24 and 90, respectively. If 4 jobs can be incorporated in a batch, the associated objective values are reduced to 16 and 88. The solution structure also becomes apparent through this example, where jobs are processed in SPT order. To maximize machine utilization, it is obviously preferable to group jobs with similar processing durations into the same batch. This complies exactly with the principle of simplifying the batching phase.

Next, a flow shop example with 4 jobs and 4 batching machines is examined. Table 4 gives the corresponding processing times. Assume that one batch can include at most

Table 4: Processing times (example with batching machines in flow shop)

Job index	i	1	2	3	4
Processing time	p_{i1}	1	9	8	3
	p_{i2}	6	10	9	5
	p_{i3}	3	8	2	7
	p_{i4}	7	5	8	9

2 jobs. The test result is a permutation schedule as shown in figure 2. Due to the limited

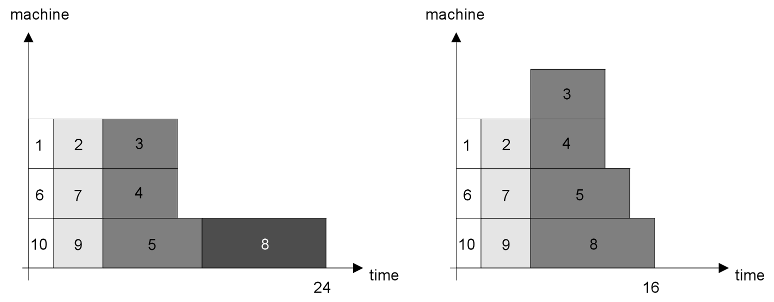

Figure 1: Example with 1 batching machine (total completion time)

capacity of batches, jobs to be scheduled must be split into two batches. To minimize processing delay, the first batch on the first machine is thus constructed with batch processing time smallest possible.

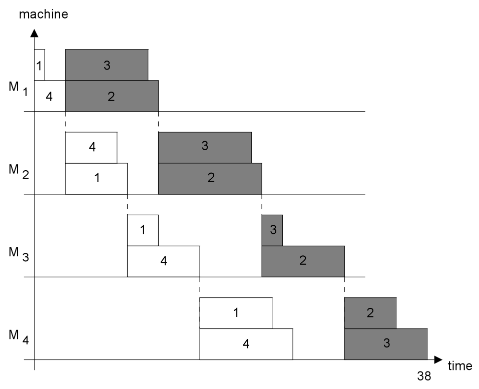

Figure 2: Flow shop example with batch processing machines ($S = 2$)

On the basis of the MIP formulation with flow shop batching machines, the approaches proposed above are implemented and tested on specific problem sets with Lingo 9.0. Each problem class contains 10 randomly generated instances and the average computing time required is reported in table 5.

Note that the problem class with 10 jobs on 4 machines is not solvable in reasonable amount of computing time. By limiting solutions to permutation flow shops, moderate-sized instances can be solved efficiently. An optimal schedule with 10 jobs on 4 batch processing machines can be obtained within 1 minute. For example, a solution with the makespan of 44 is shown in figure 3 where sequences remain unchanged over all stages. In comparison, figure 4 depicts a non-permutation schedule with the makespan of 46, which is the current best one after running Lingo 9.0 for 41 hours. Jobs belonging to the same batch on machine 1 are highlighted with the same shade, which clearly shows the variation of batches on different machines.

In summary, MIP-based approaches enable solutions to moderate-sized problem in-

Table 5: Computing time for MIP-based approaches (in s)

$n \cdot m$	Simplifying scheduling	Simplifying batching	Consistent batches and permutation schedules
$6 \cdot 4$	14	10	5
$10 \cdot 4$	64	48	30
$10 \cdot 5$	201	70	33
$10 \cdot 10$	371	244	93

stances which remain unsolvable so far. Due to the integration of heuristic rules that are widely applied and favourable in the practice, a balance between solution quality and computing time can be achieved.

5 Conclusions

This paper deals with the flow shop scheduling problem involving batch processing machines. The presented mixed integer programming formulation not only extends the existing models in the literature, but also enables solutions to the problem instances beyond the capability of exact methods developed so far. We further propose MIP-based approaches combining some well-performing heuristic rules. Computational results also confirm the substantial reduction of computing time. For future studies, more sophisticated methods, such as a hierarchical structure, can be utilized to develop MIP-based approaches.

References

[1] J.H. Ahmadi, R.H. Ahmadi, S. Dasu, and C.S. Tang. Batching and scheduling jobs on batch and discrete processors. Operations Research, 39:750–763, 1992.

[2] T.C.E. Cheng, B.M.T. Lin, and A. Toker. Makespan minimization in the two-machine flowshop batch scheduling problem. Naval Research Logistics, 47:128–144, 2000.

[3] P. Damodaran, P.K. Manjeshwar, and K. Srihari. Minimizing makespan on a batch processing machine with non-identical job sizes using genetic algorithms. International Journal of Production Economics, 103:882–891, 2006.

[4] P. Damodaran and K. Srihari. Mixed integer formulation to minimize makespan in a flow shop with batch processing machines. Mathematical and Computer Modelling, 40:1465–1472, 2004.

[5] C. Liao and L. Liao. Improved MILP models for two-machine flowshop with batch processing machines. Mathematical and Computer Modelling. In press.

[6] C.N. Potts and M.Y. Kovalyov. Scheduling with batching: A review. European Journal of Operational Research, 120:228–249, 2000.

[7] C.N. Potts, V.A. Strusevich, and T. Tautenhahn. Scheduling batches with simultaneous job processing for two-machine shop problems. Journal of Scheduling, 4:25–51, 2001.

[8] C.S. Sung, Y.H. Kim, and S.H. Yoon. A problem reduction and decomposition approach for scheduling for a flowshop of batch processing machines. European Journal of Operational Research, 121:179–192, 2000.