Applied Mathematical Modelling
Manuscript Draft

Manuscript Number: APM-D-16-00719

Title: Flexible job shop scheduling problem with parallel batch processing machine

Article Type: Research Paper

Keywords: FJSP; PBM; MIP; Priority Job; Semiconductor

Abstract: Flexible job-shop scheduling problem (FJSP) with parallel batch processing machine (PBM) is studied. First, a mixed integer programming (MIP) formulation is proposed for the first time. In order to address a NP-hard structure of this problem, we relax the model to selectively schedule jobs. Although there are thousands of jobs in a floor, we are most interested in priority jobs because a special customers promise a significant amount of financial compensation in exchange of an expedited delivery. This relaxation could allow some non-priority jobs remain unscheduled, but it vastly expedites the discovery of improving solutions by branching on integer variables with higher priority jobs first. We then turn job-dependent processing time into machine-dependent by assuming a machine has an equal processing time on different jobs. This assumption is roughly true or acceptable for the sake of the reduced computational time in semiconductor manufacturing. This further relaxed model significantly reduces a computational time compared to the original model when tested on a set of common problem instances from a literature. Computational results show that this proposed model can generate an effective schedule for a large problems. Authors encourage other researchers to propose an improved MIP model.

***Highlights**

- A mathematical formulation of FJSP with batching is proposed.
- Priority job scheduling problem is addressed
- Proposed models are tested on a set of common problem instances.

Flexible job shop scheduling problem with parallel batch processing machine

Andy Ham

Andy.Ham@TAMUC.edu

Abstract – Flexible job-shop scheduling problem (FJSP) with parallel batch processing machine (PBM) is studied. First, a mixed integer programming (MIP) formulation is proposed for the first time. In order to address a NP-hard structure of this problem, we relax the model to *selectively* schedule jobs. Although there are thousands of jobs in a floor, we are most interested in priority jobs because a special customers promise a significant amount of financial compensation in exchange of an expedited delivery. This relaxation could allow some non-priority jobs remain unscheduled, but it vastly expedites the discovery of improving solutions by branching on integer variables with higher priority jobs first. We then turn job-dependent processing time into machine-dependent by assuming a machine has an equal processing time on different jobs. This assumption is roughly true or acceptable for the sake of the reduced computational time in semiconductor manufacturing. This further relaxed model significantly reduces a computational time compared to the original model when tested on a set of common problem instances from a literature. Computational results show that this proposed model can generate an effective schedule for a large problems. Authors encourage other researchers to propose an improved MIP model.

***Index Terms*— FJSP; PBM; MIP; Priority Job; Semiconductor.**

1. INTRODUCTION

In semiconductor industry, researchers have exploited a performance of local production areas such as lithography, diffusion, etch, and implanter for the last decades by using advanced scheduling/dispatching systems. Now, there is a growing need of orchestrating a whole factory to seek a global optimization. While flexible job shop scheduling problem

(FJSP) of 3000-job (assuming 50K monthly wafers output and 45 days cycle time), 1000-machine, 500-step, 40-product is unlikely to be solved in reasonable time, linking and orchestrating multiple consecutive steps seems to be approachable.

One of applications is wet-diffusion area scheduling problem which has 2-4 consecutive steps with parallel batching machines [1-3]. Another applications is to schedule jobs having a time constraints between consecutive process steps [4-7]. Last application which has yet been studied in the literature is to schedule a priority jobs in order to meet a due date. Industry calls it *priority job scheduler* (PJS). Due to non-preemptive nature of machines in semiconductor manufacturing, a floor supervisor often takes an extreme measure letting some ports of machine empty as a priority jobs are approaching from an upstream steps which results in a productivity loss. We tackle this PJS problem in the context of FJSP with batching.

Considerable research has been devoted to FJSP in the literature. However there is no earlier work about FJSP with parallel batch processing machine (PBM) at best knowledge. Hereby, the motivations of this paper are to compose a MIP formulation for FJSP with batching constraint for the first time and make a necessary customization in order to implement the theory into a large scale real-world problem. The rest of this paper is organized as follows: a literature review is presented in Section 2 and the proposed MIP model is developed in Section 3. Computational results are reported in Section 4 and finally Section 5 covers the conclusion.

2. PREVIOUS RELATED WORK

2.1 FJSP

The classical JSP schedules a set of jobs on a set of machines with the objective to minimize a maximum completion time over all jobs (*Cmax*), subjected to the constraint that each job has an ordered set of operations, each of which must be processed on a predefined machine, whereas FJSP allows an operation to be processed on a machine out of a set of alternatives, which adds another dimension of complexity.

Researchers have addressed the FJSP mostly using heuristics. Despite the fact that those heuristics may generate fast and effective solutions, they are usually tailor-made. 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 framework. In contrast, mathematical modelling approach divides the problem representation and the solution strategy in an exact MIP model and solving mechanics respectively [8]. Furthermore, as computer hardware and solvers have improved, practitioners have been able to formulate increasingly detailed and complex problems. Therefore, we explorer a mathematical modelling approach.

Table 1 shows an overview of FJSP mathematical models in the literature. The overview table created by Demir and İşleyen [29] is slightly modified. A vast number of researches has addressed FJSP and its variants such as plan flexibility, setup, overlapping, preventive maintenance, etc. However, to the best of our knowledge to date, no published work has dealt with the FJSP with batching.

Table 1
The articles related FJSP mathematical models

Ref.	Year	Highlights	Journal
[9]	1997	with sequence dependent setup times	EJOR
[10]	1999	with process plan flexibility	IJPR
[11]	2001	with alternative process plan	IJPE
[12]	2001	with sequence dependent setup times	IJPR
[13]	2001	with sequence dependent setup times	IEEE
[14]	2002	with process plan flexibility	C&IE
[15]	2005	with homogenous machines	IJPE
[16]	2006	with sequence dependent setup times	IEEE
[17]	2006	with sequence independent setup times	C&OR
[18]	2006	with flexible preventive maintenance	JIM
[19]	2007	-	JIM
[20]	2007	with sequence dependent setup times	IJAMT
[21]	2009	-	C&IE
[22]	2009	with sequence independent setup times	SETP
[23]	2009	with overlapping	AMM
[24]	2010	with process plan flexibility	AMM
[25]	2010	with disturbances	JMST
[26]	2010	-	IJPR
[27]	2011	with preventive maintenance	ESA
[28]	2012	with transportation constraints	C&OR
[29]	2013	evaluation of MIP models	AMM
[30]	2015	with sequence dependent setup times	JIM

They also categorize FJSP mathematical formulations in the literature into the three different types: sequence-position variable based, precedence variable based, and time-indexed. Our proposed model is based on the sequence-position variable.

2.2 Priority job scheduler

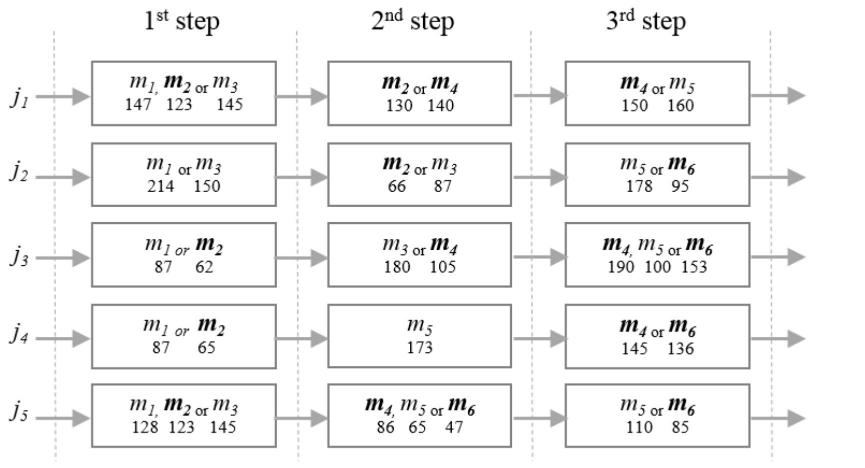
Business requirements drive the need for a small number of jobs to be run through the factory as fast as possible. Various manual and automated schemes have been tried to keep the priority jobs from “queuing at the machine”. These schemes involve idling tools ahead of the arrival of priority jobs and trading machine utilization for priority jobs cycle time [1]. We tackle this PJS problem in the context of FJSP with batching.

The main contributions of this paper can be summarized as follows. We propose a mathematical formulation of FJSP with batching constraint for the first time and make a practical modification in order to implement the theory into the real-world problem encountered at semiconductor manufacturing.

3. Problem Description & Formulation

3.1 FJSP with batching

The FJSP with batch processing machine inherits every complexity of the original FJSP. In addition, it has a set of parallel batch processing machines. Each job has to be processed on one machine out of a set of given compatible machines as it visits a pre-determined series of steps. The batching allows multiple jobs to be simultaneously processed as long as the total size of the batch does not exceed machine capacity. The processing time of a batch is dependent on the individual jobs in the batch, which is the maximum of individual processing times. Figure 1 represents a FJSP instance of 5-job, 6-machine, and 3-step with batching.


1st step
2nd step
3rd step
j_1
m_1, $\boldsymbol{m_2}$ or m_3
147 123 145
$\boldsymbol{m_2}$ or $\boldsymbol{m_4}$
130 140
$\boldsymbol{m_4}$ or m_5
150 160
j_2
m_1 or m_3
214 150
$\boldsymbol{m_2}$ or m_3
66 87
m_5 or $\boldsymbol{m_6}$
178 95
j_3
m_1 or $\boldsymbol{m_2}$
87 62
m_3 or $\boldsymbol{m_4}$
180 105
$\boldsymbol{m_4}$, m_5 or $\boldsymbol{m_6}$
190 100 153
j_4
m_1 or $\boldsymbol{m_2}$
87 65
m_5
173
$\boldsymbol{m_4}$ or $\boldsymbol{m_6}$
145 136
j_5
m_1, $\boldsymbol{m_2}$ or m_3
128 123 145
$\boldsymbol{m_4}$, m_5 or $\boldsymbol{m_6}$
86 65 47
m_5 or $\boldsymbol{m_6}$
110 85

Note: Machines in bold font have a capacity of two.
Numbers under each machine represent a processing time at steps.
Batching requirement is added to original instance MFJS1 by Fattahi *et al.*[19]

Fig. 1. FJSP instance of 5-job, 6-machine, and 3-step with batching.

In this problem, we assume all machines and jobs are available at time 0. We also assume that batch processing machine can process different products simultaneously (compatible job families). The notation used in this paper is summarized in the following:

Sets:

J	jobs (j)
S	steps (s)
M	machines (m)
$K(\lvert J\rvert \times \lvert S\rvert)$	permutation sequences (k)

Parameters:

$par^{ptime}_{j,s,m}$	processing time
par^{capa}_{m}	capacity of machine m
M	$max\left\{\sum_{j,s,m} par^{ptime}_{j,s,m}\right\}$

Decision variables:

X_{jsmk}	1 if job j occupies sequence k in the sequence on machine m at step s; 0 otherwise

Resultant variables:

$J^{arrival}_{j,s}$	arrival time to step s of job j
$M^{start}_{m,k}$	start time of sequence k on machine m
$M^{complete}_{m,k}$	completion time of sequence k on machine m
$M^{ptime}_{m,k}$	processing time of sequence k on machine m
C_{max}	makespan

We name the following model $FJSP^{+Batching}$.

Routing	$\sum_{m,k} X_{jsmk} = 1 \quad \forall j, s$	(1.1)

	$\sum_{j,s}(X_{jsmk}) \leq par_m^{capa} \quad \forall k,m$	(1.2)
Scheduling	$M_{m,k}^{ptime} \geq \left(par_{j,s,m}^{ptime}\right)X_{jsmk} \;\; \forall j,s,m,k$	(1.3)
	$J_{j,s}^{arrival} = par_j^{release} \;\; \forall j, s = 1$	(1.4)
	$M_{m,k}^{start} \geq J_{j,s}^{arrival} + M(X_{jsmk} - 1) \quad \forall j,s,m,k$	(1.5)
	$M_{m,k}^{complete} = M_{m,k}^{start} + M_{m,k}^{ptime} \quad \forall m,k$	(1.6)
	$J_{j,s+1}^{Arrival} \geq M_{m,k}^{complete} + M(X_{jsmk} - 1) \;\; \forall j,m,k : s < \lvert S\rvert$	(1.7)
	$M_{m,k+1}^{start} \geq M_{m,k}^{complete} \; \forall m : k < \lvert K\rvert$	(1.8)
Measuring	$Cmax \geq M_{m,k}^{complete} + M(X_{jsmk} - 1) \; \forall j,s,m,k$	(1.9)
	$Minimize\ CMAX$	(1.10)

Constraint (1.1) ensures that jobs are assigned to one of the available slots. Machine capacity is taken into consideration in Constraint (1.2). Then, Constraint (1.3) defines the processing time of a batch on a machine, which is represented by the longest time of all jobs in the batch. Constraint (1.4) considers the release time of a job. Constraint (1.5) ensures that a batch cannot start its processing until all jobs assigned to the corresponding batch become ready. Constraint (1.6) determines the completion time of a batch. Constraint (1.7) ensures that the available time of a job is greater than or equal to the completion time at the very previous step. Constraint (1.8) ensures the precedence relationship between batches at the same machine. Finally, Constraint (1.9) determines the makespan and Objective (1.10) minimizes it.

Figure 2 represents an optimal solution for the FJSP instance of 5-job, 6-machine, and 3-step when batching is not considered or the capacity of machine is set to 1. The values in rectangle show a job schedule, for instance, *j3s1m1p87* indicates job 3 is scheduled to machine 1 at step 1 with a processing time of 87.

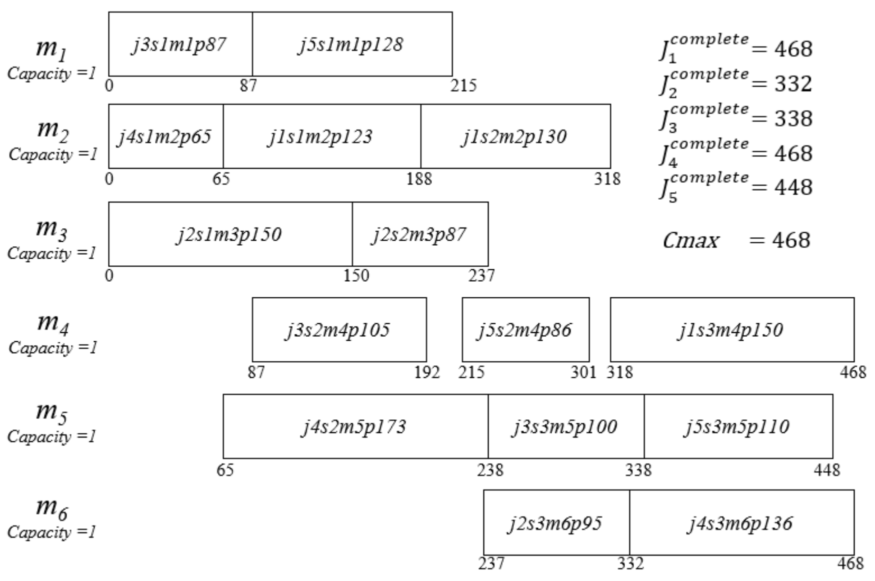


Fig. 2. An optimal solution for the 5-job, 6-machine, and 3-step w/o batching

On the other hand, Figure 3 represents an optimal solution for the same instance when batching is considered.

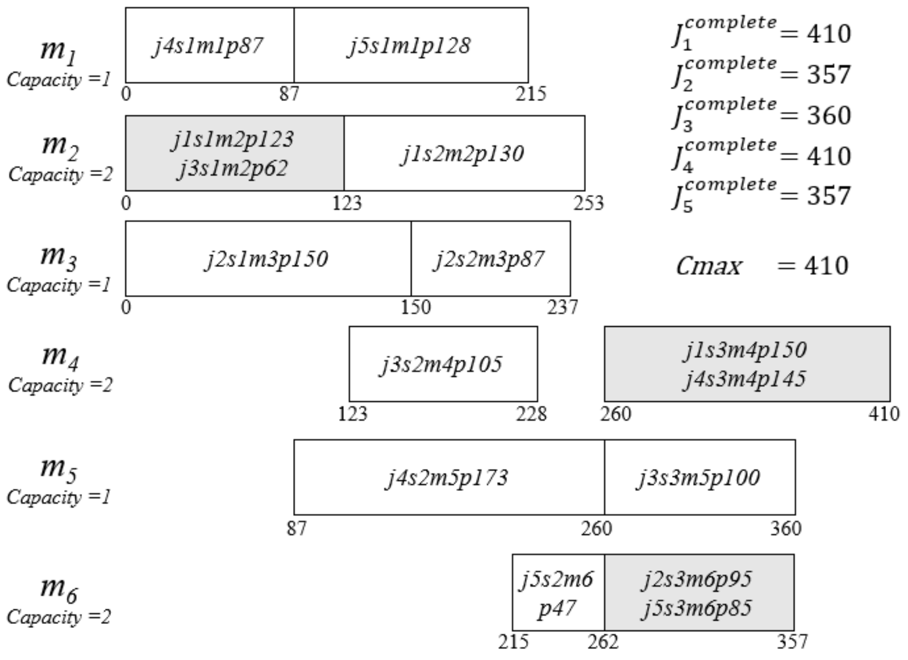


Fig. 3. An optimal solution for the 5-job, 6-machine, and 3-step w/ batching

3.2 FJSP with batching applied to priority job scheduling

During a preliminary experimentation, we found that the $FJSP^{+Batching}$ model can not generate an effective solution for a large instances after several hours of computational time so we explorer an opportunity of practical modification in order to address a real-world scheduling problem. An interview with an industry subject matter expert (SME) provides two key insights:

a) One key insight is about a selective job scheduling. There are thousands of jobs in floor, but we are most interested in priority jobs because a special customers promise a significant amount of financial compensation in exchange of an expedited delivery. To take advantage of this compromise, the proposed model has come under close scrutiny. The model uses lots of binary variables (X_{jsmk}). One of set indexes which caught our eye is *k*, permutation sequence, whose size is currently set to $|J|\times|S|$ assuming all operations could be scheduled to a single machine. This size can be dramatically decreased owing to the compromise which allows some jobs remain unscheduled. Now, the routing Constraint (1.1) which forces all jobs to be assigned is not valid any more so we change the constraint to allow jobs not being scheduled as follows:

$$\sum_{m,k} X_{jsmk} \le 1 \quad \forall j, s \tag{2.1}$$

Constraint (2.2) is added in order to force a job to complete all operations once it is selected for schedule.

$$\sum_{mk} X_{jsmk} = \sum_{mk} X_{j,s+1,m,k} \ \forall j : s < |S| \tag{2.2}$$

Unfortunately, this change on the routing constraint drives a solver to make undesirable results. Namely, a solver drops all jobs from a schedule to minimize *Cmax*. In order to prevent this side effect, Objective (2.3) rewards each job schedule with a large compensation, *M*, which is the theoretical upper bound of *Cmax*.

$$Maximize \sum_{j,s,m,k} (M) X_{jsmk} \tag{2.3}$$

We then calculate a completion time of a job and use it to calculate the makespan as melted into Constraints (2.4) and (2.5), which replace Constraint (1.9).

$J_j^{complete} \geq M_{m,k}^{complete} + M(X_{jsmk} - 1)\ \forall jsmk$	(2.4)
$CMAX \geq J_j^{complete}\ \ \forall j$	(2.5)

This change reduces a number of nodes in branch-and-bound algorithm owing to smaller $|K|$. This variant is named $FJSP_{Selective}^{+Batching}$.

b) Another insight is about a processing time. Although there are minor variations of processing times, a machine has an equal processing time on different jobs in general. This compromise is acceptable for the sake of a reduced computational time. The proposed model has again come under close scrutiny. We modify $FJSP^{+Batching}$ model as follows. Let par_m^{ptime} be the processing time of machine m. Then, Constraint (1.6) can be replaced by Constraint (2.6) and Constraint (1.3) can be removed. This modification tightens the formulation and vastly improves CPLEX run-time performance. We discuss it in computational study section.

$M_{m,k}^{complete} = M_{m,k}^{start} + par_m^{ptime}\ \ \forall m, k$	(2.6)

Hereby, new objectives are to maximize a total count of jobs scheduled while minimizing the last completion time.

4. Computational Study

In this section, we test the effectiveness of our proposed models. We first compare $FJSP^{+Batching}$ with $FJSP_{Selective}^{+Batching}$ in order to understand a computational advantage of the proposed method. Then, we repeat a similar study with the assumption of machine-dependent processing time.

MIP models are generated by IBM OPL and solved by CPLEX 12.6.3 on a personal computer with an Intel Core i5-3470 @ 3.2 Ghz processor and 16 GB RAM.

4.1 Small-size test instances

To test our model, we borrow the same test problems instances created by Fattahi *et al*. [19]. They randomly generated a total of 20 FJSP instances. The instances are divided to two categories: small size problems (SFJS1:10) and medium and large size problems (MFJS1:10). The instances however do not have a batching requirement so we simply assume even (odd) numbered machines have a capacity of two (one). Another change is made to support the machine-dependent processing time assumption. The processing time of machine (par_{m}^{ptime})is calculated as follows: $min\left\{par_{j,s,m}^{ptime}\ \ \forall m\right\}$.

In this study, we make sure $FJSP_{Selective}^{+Batching}$to schedule every jobs for a fair comparison with $FJSP^{+Batching}$. This is made by setting |*K*| value to six and confirmed during a study. The customized CPLEX logs have all details about the scheduling results and the number of jobs scheduled. All the test instances, log files, and results are located at https://dl.dropboxusercontent.com/u/57440903/FJSP_Batching.zip

4.2 Results of small-size test instances

Table 2 summarizes the computational results. Column 1 shows the name of instances used by Fattahi *et al*. [19]. Columns 2–5 contain the best solutions generated by $FJSP^{+Batching}$ and $FJSP_{Selective}^{+Batching}$ which are reported within a 300 seconds. The $FJSP_{Selective}^{+Batching}$finds an optimal solutions of 11 instances out of 20 within 300 seconds compared to 9 out of 20 by $FJSP^{+Batching}$ which demonstrates a reduction of computational time. Another finding is that there is a significant differences in *Cmax* values for the large instances of MFJS 8, 9 and 10, which suggest a superiority of $FJSP_{Selective}^{+Batching}$ for an industry-scale problem instance.

Table 2
Comparison of two different models with the assumption of job-dependent processing time.

	$FJSP^{+Batching}$		$FJSP_{Selective}^{+Batching}$	
Problem	*Cmax*	CPU	*Cmax*	CPU
SFJS1	**66**	0.05	**66**	0.08
SFJS2	**107**	0.05	**107**	0.05
SFJS3	**208**	0.31	**208**	0.20
SFJS4	**272**	0.06	**272**	0.05

SFJS5	**100**	0.14	**100**	0.39
SFJS6	**320**	12.29	**320**	1.37
SFJS7	**397**	9.77	**397**	1.00
SFJS8	**216**	5.34	**216**	6.46
SFJS9	**210**	2.95	**210**	0.87
SFJS10	516	300	**516**	91.96
MFJS1	410	300	410	300
MFJS2	410	300	410	300
MFJS3	420	300	420	300
MFJS4	506	300	506	300
MFJS5	488	300	488	300
MFJS6	631	300	**614**	42.43
MFJS7	916	300	863	300
MFJS8	896	300	808	300
MFJS9	nf	300	955	300
MFJS10	3418	300	1215	300

* bold font indicates an optimal.

Table 3 summarizes the computational results based on the assumption of machine-dependent processing time. Under this assumption, both $FJSP^{+Batching}$and $FJSP^{+Batching}_{Selective}$ find an optimal solutions of 16 instances out of 20 within 300 seconds which demonstrates a reduction of computational time by tightening the formulation. We also find a significant differences in *Cmax* values for the large instances of MFJS8, 9 and 10.

Table 3
Comparison of two different models with the assumption of machine-dependent processing time.

	$FJSP^{+Batching}$		$FJSP^{+Batching}_{Selective}$	
Problem	*Cmax*	CPU	*Cmax*	CPU
SFJS1	**48**	0.03	**48**	**0.03**
SFJS2	**63**	0.01	**63**	0.03
SFJS3	**106**	0.03	**106**	0.03
SFJS4	**126**	0.03	**126**	0.03
SFJS5	**42**	0.02	**42**	0.03
SFJS6	**134**	0.09	**134**	0.12
SFJS7	**194**	0.08	**194**	0.17
SFJS8	**100**	0.11	**100**	0.17
SFJS9	**84**	0.16	**84**	0.25
SFJS10	**314**	0.22	**314**	0.28
MFJS1	**236**	1.00	**236**	4.38
MFJS2	**236**	2.28	**236**	6.58
MFJS3	**250**	4.06	**250**	12.34
MFJS4	**251**	26.43	**251**	71.57
MFJS5	**251**	17.91	**251**	29.44
MFJS6	**254**	23.71	**254**	51.29
MFJS7	325	300	325	300

MFJS8	980	300	344	300
MFJS9	1160	300	435	300
MFJS10	1036	300	457	300

* bold font indicates an optimal.

After confirming both modifications could significantly reduce a computational time, we finally apply the proposed model to the priority job scheduling problem encountered in semiconductor manufacturing.

4.3 Priority job scheduling test instances

At this time, we need a problem instances especially with 3-step according to an interview with industry SME. We could not acquire a real instances due to an intellectual property concern so we randomly generate them. There are 50 jobs. The first 10% of jobs are tagged as *superhot*, the next 10% *hot*, and the remaining 80% *normal*. Jobs with *superhot* or *hot* must be included into a schedule. Remaining *normal* jobs could be selectively scheduled as much as possible to maximize a production output, but it is still optional.

The new performance measures are to maximize a total amount of production (or jobs scheduled) and to meet a target of X-factor [31] which is defined as the ratio of flow time to raw process time. The X-factor is commonly measured in semiconductor manufacturing to assess a level of operational excellence. Table 4 represents a job priority profile with its X-factor target. We need to modify the the model in order to calculate the X-factor of a job and use it as one of objectives. This variant is named $FJSP_{PJS}^{+Batching}$ and the mathematical model is provided in Appendix.

Table 4
Profile of job priority with its X-factor target

Priority levels	Volume	Selectiveness	X-factor target
Superhot	10%	Must	1.1
Hot	10%	Optional	1.3
Normal	80%	Optional	-

Finally, there are 10 machines and 3 steps. Therefore, each machine could be scheduled with 15 operations (=3 steps × 50 jobs / 10 machines) on average if all jobs are scheduled. We set $|K|$ value to 4 and confirm that it provides enough spaces to accommodate all priority jobs.

4.4 Results of priority job scheduling

Table 5 reports the computational results. Column 1 shows the name of instances. Columns 2–4 contain an optimal solutions of *superhot* jobs: number of jobs scheduled, average of X-factors, and maximum of X-factors. Columns 5–7 (8–10) contain the results of *hot* jobs (*normal*). The last Column 11 records the CPU time to reach at an optimal solution.

Table 5
Optimal schedules of priority job scheduling.

	Superhot jobs			Hot jobs			Normal jobs			
Problem	Jobs	AvgX	MaxX	Jobs	AvgX	MaxX	Jobs	AvgX	MaxX	CPU
PJS01	5	1.05	1.14	5	1.28	1.33	10	1.72	2.11	150
PJS02	5	1.06	1.11	5	1.28	1.44	10	1.69	2.63	33
PJS03	5	1.11	1.27	5	1.31	1.45	10	1.57	1.60	113
PJS04	5	1.04	1.11	5	1.33	1.40	10	1.72	2.25	120
PJS05	5	1.09	1.09	5	1.28	1.30	10	1.74	2.18	65
PJS06	5	1.05	1.14	5	1.31	1.38	10	1.66	2.13	119
PJS07	5	1.02	1.10	5	1.26	1.30	10	1.73	2.44	64
PJS08	5	1.09	1.10	5	1.33	1.55	10	1.68	2.63	111
PJS09	5	1.10	1.10	5	1.29	1.50	10	1.60	1.80	35
PJS10	5	1.09	1.10	5	1.32	1.40	10	1.74	2.18	105
Average	5	1.07	1.13	5	1.30	1.41	10	1.69	2.19	91

The $FJSP_{PJS}^{+Batching}$ finds an optimal schedule within 2.5 minutes for all problem instances while accomplishing the X-factor targets. Figure 4 represents an optimal solution of PSJ01. The each box shows a schedule of an operation, for instance, *j1s1 superhot* on *m1* indicates job *1* which has the *superhot* priority is scheduled to machine *1* at step *1* starting at time o and completing at 2.

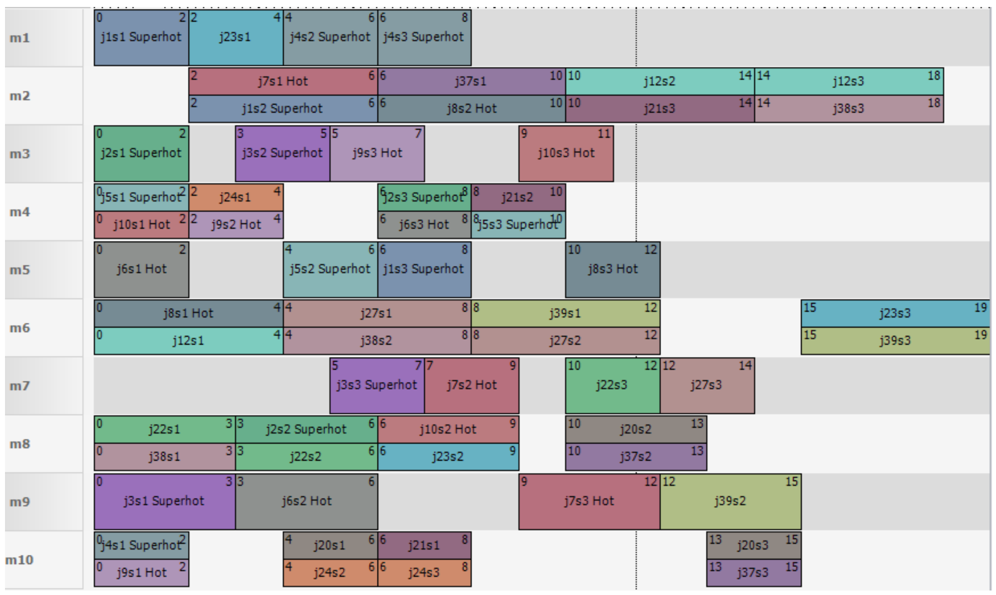


Fig. 4. Gantt chart schedule of test instance of PJS01

5. Conclusion and Future Works

Encountered at semiconductor manufacturing, flexible job-shop scheduling problem (FJSP) with parallel batch processing machine (PBM) is studied as there is a growing need of orchestrating a whole factory to seek a global optimization. We establish a baseline by composing a mixed integer programming (MIP) formulation for the first time. Owing to two critical insights we found during an interview with industry SME, an original mathematical formulation is relaxed to cope with an industry-size instances. The first insight is about a *selective* job scheduling. Although there are thousands of jobs in a floor, industry is most interested in priority jobs because a special customers promise a significant amount of financial compensation in exchange of an expedited delivery. There are three levels of priority: *superhot*, *hot*, and *normal*. Jobs with *superhot* or *hot* priorities must be included into a schedule whereas *normal* jobs could be selectively scheduled as much as possible to maximize a production output, but it is still optional. The second insight is about the machine-dependent processing time. Although there are minor variations of processing times depending on recipes, a machine has a similar processing time on different jobs in general. This compromise is acceptable for the sake of a reduced computational time. This modification tightens the formulation and vastly improves CPLEX run-time performance. Computational results show that the proposed model can schedule 50-job, 10-machine, and 3-step within 2.5 minutes run-time.

This research can be further extended by considering a relatively new approach, constraint programming (CP), which is designed to cope with a complex scheduling problems such as TSP, JSP, and FJSP. Another extension is to improve the proposed MIP model. We can also look into application side as we previously discussed in the introduction section.

Appendix.

$FJSP_{Selective}^{+Batching}$ **model in condense form**

Routing	$\sum_{m,k} X_{jsmk} \leq 1 \quad \forall j,s$	(2.1)
	$\sum_{mk} X_{jsmk} = \sum_{mk} X_{j,s+1,m,k} \ \forall j : s < \lvert S\rvert$	(2.2)
	$\sum_{j,s} (X_{jsmk}) \leq par_m^{capa} \quad \forall k,m$	(1.2)
Scheduling	$J_{j,s}^{arrival} = par_j^{release} \ \forall j, s = 1$	(1.4)
	$M_{m,k}^{start} \geq J_{j,s}^{arrival} + M(X_{jsmk} - 1) \ \forall j,s,m,k$	(1.5)
	$M_{m,k}^{complete} = M_{m,k}^{start} + par_m^{ptime} \quad \forall m,k$	(2.6)
	$J_{j,s+1}^{arrival} \geq M_{m,k}^{complete} + M(X_{jsmk} - 1)$ $\forall j,m,k : s < \lvert S\rvert$	(1.7)
	$M_{m,k+1}^{start} \geq M_{m,k}^{complete} \ \forall m : k < \lvert K\rvert$	(1.8)
Measuring	$J_j^{complete} \geq M_{m,k}^{complete} + M(X_{jsmk} - 1) \ \forall jsmk$	(2.4)
	$CMAX \geq J_j^{complete} \ \forall j$	(2.5)
	$Minimize\ CMAX$	(1.10)
	$Maximize \sum_{j,s,m,k} (M) X_{jsmk}$	(2.3)

$FJSP_{PJS}^{+Batching}$ **model in condense form**

In order to explicitly generate the X-factor of individual job and control it as a knob, the following changes are made.

Parameters:

par_j^{pri}	priority of job *j* (1 for *superhot*, 2 for *hot*, 9 for *normal*)
$Par_j^{Xfactor}$	target X-factor of of job *j*

Par_j^{TCT}	theoretical cycle time of job *j* which is the summation of processing times at each steps

Resultant variables:

$J_j^{\mathrm{X}-}$	amount of X-factor under accomplished against a target
$J_j^{\mathrm{X}+}$	amount of X-factor over accomplished against a target

Routing	$\sum_{m,k} X_{jsmk} = 1 \;\; \forall j,s : par_j^{pri} = 1 \; or \; 2$	(3.1)
	$\sum_{m,k} X_{jsmk} \leq 1 \;\; \forall j,s : par_j^{pri} = 9$	(3.2)
	$\sum_{mk} X_{jsmk} = \sum_{mk} X_{j,s+1,m,k} \; \forall j : s < \lvert S\rvert$	(2.2)
	$\sum_{j,s} (X_{jsmk}) \leq par_m^{capa} \;\; \forall k,m$	(1.2)
Scheduling	$J_{j,s}^{arrival} = par_j^{release} \;\; \forall j, s = 1$	(1.4)
	$M_{m,k}^{start} \geq J_{j,s}^{arrival} + M(X_{jsmk} - 1) \;\; \forall j,s,m,k$	(1.5)
	$M_{m,k}^{complete} = M_{m,k}^{start} + par_m^{ptime} \;\; \forall m,k$	(2.6)
	$J_{j,s+1}^{arrival} \geq M_{m,k}^{complete} + M(X_{jsmk} - 1) \; \forall j,m,k : s < \lvert S\rvert$	(1.7)
	$M_{m,k+1}^{start} \geq M_{m,k}^{complete} \; \forall m : k < \lvert K\rvert$	(1.8)
Measuring	$J_j^{complete} \geq M_{m,k}^{complete} + M(X_{jsmk} - 1) \; \forall jsmk$	(2.4)
	$\frac{J_j^{complete}}{Par_j^{TCT}} = Par_j^{Xfactor} + J_j^{\mathrm{X}-} - J_j^{\mathrm{X}+} \;\; \forall j$	(3.3)
	$Maximize \sum_{j,s,m,k} \left(M/Par_j^{pri}\right) X_{jsmk}$	(3.4)
	$Minimize \sum_j J_j^{\mathrm{X}-}$	(3.5)

Constraints (3.1) and (3.2) differentiate the mandatory and the selective scheduling depending on a priority of a job. Constraints (3.3) calculates the deviation from the target X-factor. Objective (3.4) maximizes the total summation of production weighted by priority. Differentiating jobs by adding appropriate weight factors to cost coefficients in the objective function helps the algorithm distinguish between dominated and dominating solutions, which expedites the discovery of improving solutions [32]. This change instructs CPLEX to branch on integer variables with higher priority jobs first. Objective (3.5) penalizes the amount of X-factor under accomplished compared to a target.

References

[1] Bixby, R., Burda, R., & Miller, D. (2006, May). Short-interval detailed production scheduling in 300mm semiconductor manufacturing using mixed integer and constraint programming. In The 17th Annual SEMI/IEEE Advanced Semiconductor Manufacturing Conference (ASMC-2006) (pp. 148-154).

[2] Yugma, C., Dauzère-Pérès, S., Artigues, C., Derreumaux, A., & Sibille, O. (2012). A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing. International Journal of Production Research, 50(8), 2118-2132.

[3] Jung, C., Pabst, D., Ham, M., Stehli, M., & Rothe, M. (2014). An effective problem decomposition method for scheduling of diffusion processes based on mixed integer linear programming. Semiconductor Manufacturing, IEEE Transactions on, 27(3), 357-363.

[4] Sun, D. S., Choung, Y. I., Lee, Y. J., & Jang, Y. C. (2005, September). Scheduling and control for time-constrained processes in semiconductor manufacturing. In Semiconductor Manufacturing, 2005. ISSM 2005, IEEE International Symposium on (pp. 295-298). IEEE.

[5] Klemmt, A., & Mönch, L. (2012, December). Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing. In Proceedings of the Winter Simulation Conference (p. 194). Winter Simulation Conference.

[6] Sadeghi, R., Dauzere-Peres, S., Yugma, C., & Lepelletier, G. (2015, May). Production control in semiconductor manufacturing with time constraints. In Advanced Semiconductor Manufacturing Conference (ASMC), 2015 26th Annual SEMI (pp. 29-33). IEEE.

[7] Ham, M., Lee, Y. H., & An, J. (2011). IP-Based Real-Time Dispatching for Two-Machine Batching Problem With Time Window Constraints. Automation Science and Engineering, IEEE Transactions on, 8(3), 589-597.

[8] Kopanos, G. M., Méndez, C. A., & Puigjaner, L. (2010). MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. European journal of operational research, 207(2), 644-655.

[9] J. Liu, B.L. MacCarty, A global milp model for FMS scheduling, Eur. J. Oper. Res. 100 (1997) 441–453.

[10] K.-H. Kim, P.J. Egbelu, Scheduling in a production environment with multiple process plans per job, Int. J. Prod. Res. 37 (1999) 2725–2753.

[11] C.S. Thomalla, Job shop scheduling with alternative process plans, Int. J. Production Economics 71 (2001) 125–134.

[12] C.Y. Low, T.H. Wu, Mathematical modelling and heuristic approaches to operation scheduling problems in an FMS environment, Int. J. Prod. Res. 39 (2001) 689–708.

[13] H. Tamaki, T. Ono, H. Murao, S. Kitamura, Modeling and genetic solution of a class of flexible job shop scheduling problems, in: Proceedings of the IEEE Symposium on Emerging Techonologies and Factory Automation, vol. 2, IEEE, 2001, pp. 343–350.

[14] Y.H. Lee, C.S. Jeong, C. Moon, Advanced planning and scheduling with outsourcing in manufacturing supply chain, Comput. Ind. Eng. 43 (2002) 351–374

[15] S.A. Torabi, B. Karimi, S.M.T. Fatemi Ghomi, The common cycle economic lot scheduling in flexible job shops: the finite horizon case, Int. J. Production Economics 97 (2005) 52–65.

[16] N. Imanipour, Modeling & solving flexible job shop problem with sequence dependent setup times, in: Proceedings of the International conference on service systems and service management, October 25–27, 2006, vol. 2, IEEE, 2006, pp. 1205–1210.

[17] C. Low, Y. Yip, T.H. Wu, Modeling and heuristics of FMS scheduling with multiple objectives, Comput. Oper. Res. 33 (2006) 674–694.

[18] J. Gao, M. Gen, L. Sun, Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, J. Intell. Manuf. 17 (2006) 493–507.

[19] P. Fattahi, M.S. Mehrebad, F. Jolai, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, J. Intell. Manuf. 18 (2007) 331–342.

[20] M.S. Mehrabad, P. Fattahi, Flexible job shop scheduling with tabu search algorithms, Int. J. Adv. Manuf. Technol. 32 (2007) 563–570.

[21] G. Zhang, X. Shao, P. Li, L. Gao, An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, Comput. Ind. Eng. 56 (2009) 1309–1318.

[22] L. Lin, H. Jia-zhen, Multi-objective flexible job-shop scheduling problem in steel tubes production, Systems engineering theory practice 29 (8) (2009) 117–126.

[23] P. Fattahi, F. Jolai, J. Arkat, Flexible job shop scheduling with overlapping in operations, Appl. Math. Modell. 33 (2009) 3076–3087.

[24] C. Özgüven, L. Özbakır, Y. Yavuz, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, Appl. Math. Modell. 34 (2010) 1539–1548.

[25] P. Fattahi, A. Fallahi, Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability, CIRP J. Manuf. Sci. Technol. 2 (2010) 114–123.

[26] Ham, M., Lee, Y. H., & Kim, S. H. (2011). Real-time scheduling of multi-stage flexible job shop floor. International Journal of Production Research,49(12), 3715-3730.

[27] E. Moradi, S.M.T. Fatemi Ghomi, M. Zandieh, Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem, Expert Syst. Appl. 38 (2011) 7169–7178.

[28] Q. Zhang, H. Manier, M.-A. Manier, A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times, Comput. Oper. Res. 39 (2012) 1713–1723.

[29] Demir, Y., & İşleyen, S. K. (2013). Evaluation of mathematical models for flexible job-shop scheduling problems. Applied Mathematical Modelling, 37(3), 977-988.

[30] Jalilvand-Nejad, A., & Fattahi, P. (2015). A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. Journal of Intelligent Manufacturing, 26(6), 1085-1098.

[31] Martin, D. P. (1998, September). The advantages of using short cycle time manufacturing (SCM) instead of continuous flow manufacturing (CFM). In Advanced Semiconductor Manufacturing Conference and Workshop, 1998. 1998 IEEE/SEMI (pp. 43-49). IEEE.

[32] Klotz, E., & Newman, A. M. (2013). Practical guidelines for solving difficult mixed integer linear programs. Surveys in Operations Research and Management Science, 18(1), 18-32.