Applied Mathematical Modelling
Manuscript Draft

Manuscript Number: APM-D-16-00719R1

Title: Flexible Job Shop Scheduling Problem for Parallel Batch Processing Machine with Compatible Job Families

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, the formulation is relaxed to selectively schedule jobs. Although there are many jobs on a given floor, semiconductor manufacturing is most challenged by priority jobs that promise a significant amount of financial compensation in exchange for an expedited delivery. This relaxation could leave some non-priority jobs unscheduled. However, it vastly expedites the discovery of improving solutions by first branching on integer variables with higher priority jobs. This study then turns job-dependent processing times into job-independent ones 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 the industry. These changes significantly reduce computational time compared to the original model when tested on a set of common problem instances from the literature. Computational results show that this proposed model can generate an effective schedule for large problems. Author encourages 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.

*Detailed Response to Reviewers

Professor Johann (Hans) Sienz
Editor in Chief
Applied Mathematical Modelling

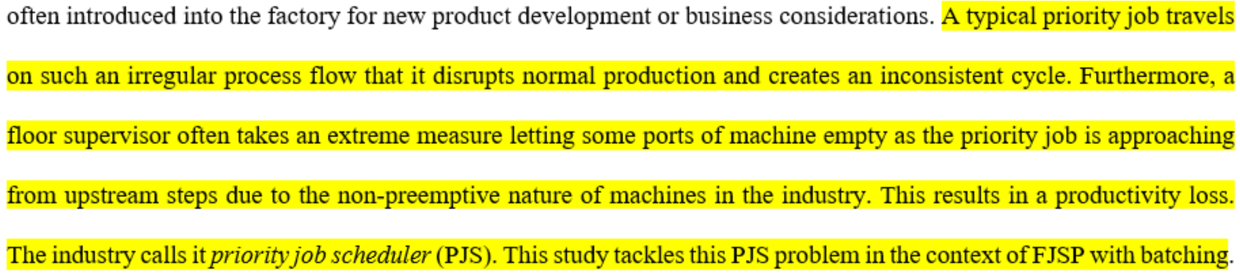
Color Legend
Comment by author
Actual change on paper

Comments from the Editors and Reviewers (if available):

Comments from the Associate Editor:

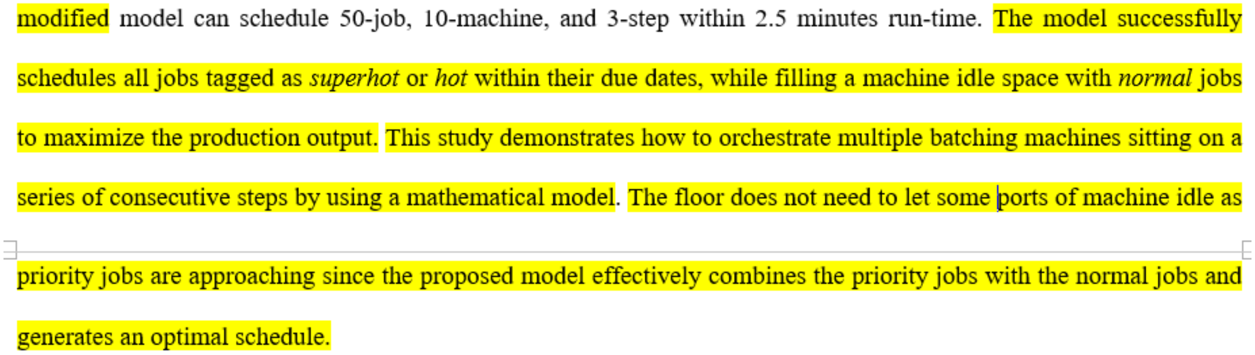
(1) Although the paper reads okay, it can still benefit from language editing (e.g., usage of articles, clauses, punctuation).
➔ The paper has just completed a careful editing.

(2) Managerial implications and insights are missing from the paper. Please emphasize the applicability of your model in a real-life engineering setting.
➔ Revised as follows:



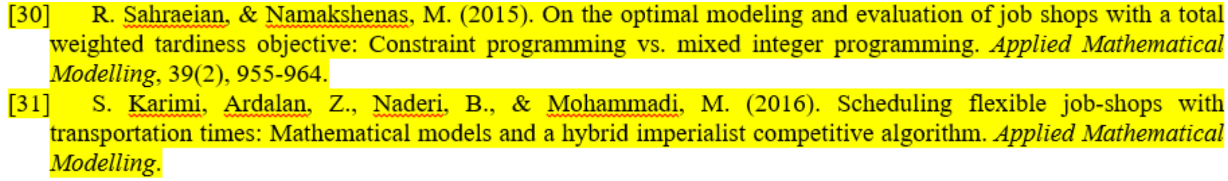
often introduced into the factory for new product development or business considerations. A typical priority job travels on such an irregular process flow that it disrupts normal production and creates an inconsistent cycle. Furthermore, a floor supervisor often takes an extreme measure letting some ports of machine empty as the priority job is approaching from upstream steps due to the non-preemptive nature of machines in the industry. This results in a productivity loss. The industry calls it *priority job scheduler* (PJS). This study tackles this PJS problem in the context of FJSP with batching.

(skip)



modified model can schedule 50-job, 10-machine, and 3-step within 2.5 minutes run-time. The model successfully schedules all jobs tagged as *superhot* or *hot* within their due dates, while filling a machine idle space with *normal* jobs to maximize the production output. This study demonstrates how to orchestrate multiple batching machines sitting on a series of consecutive steps by using a mathematical model. The floor does not need to let some ports of machine idle as priority jobs are approaching since the proposed model effectively combines the priority jobs with the normal jobs and generates an optimal schedule.

(3) cite 1-2 more relevant and recent papers from Applied Mathematical Modelling.
➔ Two more AMM papers have been added. I really appreciate this particular feedback because I found two great papers which provide me with the next research topics: FJSP in constraint programming and FJSP with transportation times. As soon as I complete this paper, I will work on FJSP with transportation times in constraint programming.

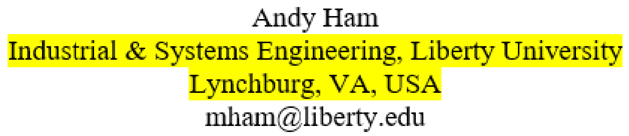


[30] R. Sahraeian, & Namakshenas, M. (2015). On the optimal modeling and evaluation of job shops with a total weighted tardiness objective: Constraint programming vs. mixed integer programming. *Applied Mathematical Modelling*, 39(2), 955-964.

[31] S. Karimi, Ardalan, Z., Naderi, B., & Mohammadi, M. (2016). Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*.

(4) Please provide a complete affiliation on the first page, not just the email.

➔ Revised as follows:



Andy Ham
Industrial & Systems Engineering, Liberty University
Lynchburg, VA, USA
mham@liberty.edu

(5) I may seek the opinion of a third reviewer.

Good luck,

MYJ

Reviewer #1: The paper deals with flexible job-shop scheduling problem with parallel batch processing machine which is encountered at semiconductor manufacturing. Authors first propose a mathematical formulation for related problem and then make a practical modification in order to implement the theory into the real-world problem. The paper is well organized and presented. The effectiveness of proposed mixed integer programming model is tested by authors. So, I suggested that the paper can be accepted after the following comments.

1. Should be explained in notations on page 7.

➔ Revised as follows:

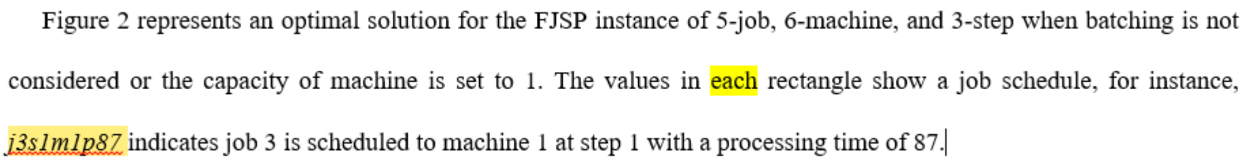
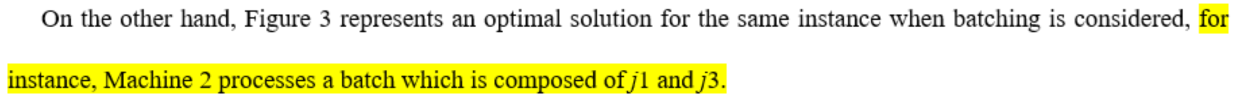


Figure 2 represents an optimal solution for the FJSP instance of 5-job, 6-machine, and 3-step when batching is not considered or the capacity of machine is set to 1. The values in each rectangle show a job schedule, for instance, *j3s1m1p87* indicates job 3 is scheduled to machine 1 at step 1 with a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

… (skip)…



On the other hand, Figure 3 represents an optimal solution for the same instance when batching is considered, for instance, Machine 2 processes a batch which is composed of $j1$ and $j3$.

2. The paper must be written in third person (Avoid using "we" in the paper).

➔The countless errors of “we” are changed into “third person” view throughout the paper.

This study then turns job-dependent

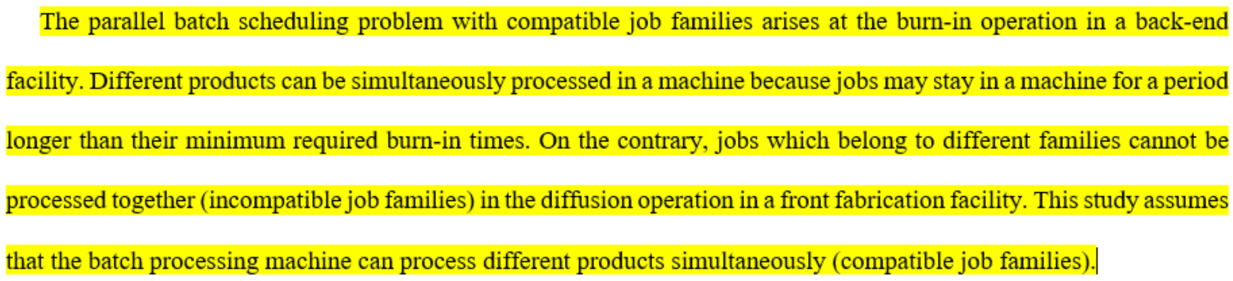
Reviewer #2:

The paper deals with a very interesting and relevant scheduling problem: the flexible job-shop with parallel batch machines. This problem is particularly important in the semiconductor industry.

Although the models and the concepts used through the paper are quite simple, the article is really hard to follow. This is partly due to the poor English, to many typos/errors/imprecisions but also to a lack of description of what the author is trying to achieve with the different versions of the models.

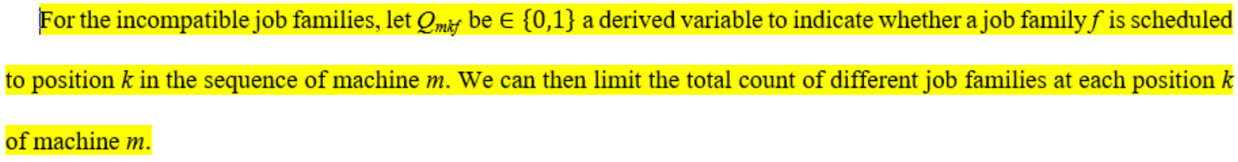
Following a description of the context of the work in section 1 and 2, section 3.1 introduces a MIP model for the FJSP with batching. This section is the most comprehensible one. The proposed MIP model, which to the best of my knowledge is original (even if it appears quite straightforward) is the main contribution of the paper. Still, this section could be improved:
- As mentioned by the author, the problem assumes that all jobs are compatible so that any two steps scheduled on the same machine can be batched together. In the semiconductor problems, there are often different families of jobs (and even steps) so that two steps of different family cannot be batched together. It would be interesting to mention how this feature could be integrated in the proposed MIP model.

➔ Revised as follows:



The parallel batch scheduling problem with compatible job families arises at the burn-in operation in a back-end facility. Different products can be simultaneously processed in a machine because jobs may stay in a machine for a period longer than their minimum required burn-in times. On the contrary, jobs which belong to different families cannot be processed together (incompatible job families) in the diffusion operation in a front fabrication facility. This study assumes that the batch processing machine can process different products simultaneously (compatible job families).

… (skip)…



For the incompatible job families, let Q_{mkf} be $\in \{0,1\}$ a derived variable to indicate whether a job family f is scheduled to position k in the sequence of machine m. We can then limit the total count of different job families at each position k of machine m.

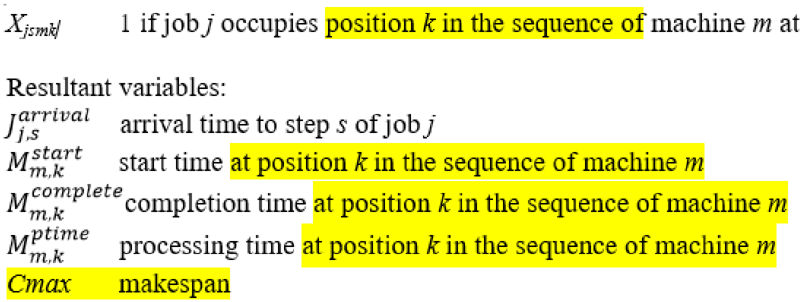
In addition, the title of paper is changed into:

Flexible Job Shop Scheduling Problem for Parallel Batch
Processing Machine with Compatible Job Families

I really appreciate this particular feedback because it clears out a confusion.

- In the description of the MIP model, instead of using the term "sequence" for "k" I would rather use something like "position" in the sequence. For instance: "X_jsmk is 1 if step s of job j occupies position k in the sequence of machine m". This is more clear than the current "X_jsmk is 1 if job j occupies sequence k in the sequence on machine m at step s". This is just an example of the very awkward formulations that, all along the paper, make it difficult to read.

➔ Revised. Thanks for this feedback. The following are some of the similar changes.



X_{jsmk} 1 if job j occupies position k in the sequence of machine m at st

Resultant variables:

$J_{j,s}^{arrival}$ arrival time to step s of job j

$M_{m,k}^{start}$ start time at position k in the sequence of machine m

$M_{m,k}^{complete}$ completion time at position k in the sequence of machine m

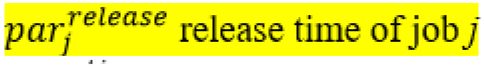
$M_{m,k}^{ptime}$ processing time at position k in the sequence of machine m

$Cmax$ makespan

- Also, pay attention to the notations:

  * For instance parameter par^release_j is used in the MIP but not described in the parameters

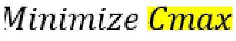
➔ Revised. Thanks for this feedback.



$par_j^{release}$ release time of job j

  * Makespan is sometimes denoted C_max, sometimes Cmax, sometimes CMAX,

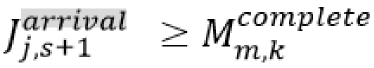
➔ Revised. Thanks for this feedback. The following is one of the similar changes.



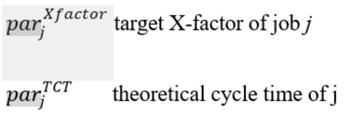
$Minimize\ Cmax$

* J^arrival sometimes J^Arrival, etc.

➔ Revised. Thanks for this feedback. The following is one of the similar changes.



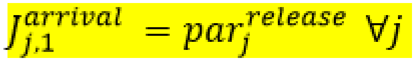
$J_{j,s+1}^{arrival} \geq M_{m,k}^{complete}$



$par_j^{Xfactor}$ target X-factor of job j

par_j^{TCT} theoretical cycle time of j

 * Simplify when possible: for instance "J^arrival_i = par^realease_j forall j,s=1". Why not: "J^arrival_1 = par^realease_j forall j" ?



$$J_{j,1}^{arrival} = par_j^{release} \quad \forall j$$

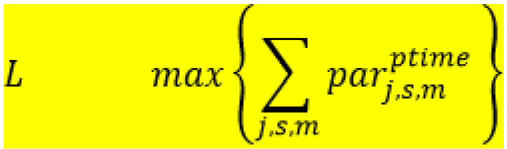
Section 3.2 introduces the notion of optional job: "... we found that the [MIP] model can not generate an effective solution for a large instances (sic) after several hours of computational time so we explorer (sic) an opportunity of practical modification in order to address a real-world scheduling problem". As we will see later, the mentioned "modifications" are not really clear but, what is even more annoying, is that it is never said what is the *nature* of these modifications. Is it "relaxations" of

the original problem (so that even if they provide non feasible solution they can be used to compute a lower-bound on the problem; is it "equivalent" reformulations of the problem that maintains the same set of optimal solutions; is it "over-constrained" modifications that result in a more constrained problem that maintains the feasibility of the produced solutions but no guarantee of optimality; or just some more general "approximations" of the original problem ? The main
advantage of MIP models for scheduling is that they an provide some guarantees on the solution quality (optimality certificates, gap). Except for the case your model modifications are "equivalent" formulations or "relaxations", they cannot bring any guarantee on the solution quality and you lose the advantage of using a MIP compared to alternative approaches as for instance meta-heuristics that can provide high-quality solutions to large instances of the problem.

The first variant described in item (a) in section 3.2 is not clear. I understand that jobs are optional and that the objective is to maximize the number of executed jobs (equation 2.3). But if there are no limitations on the makespan (and no deadlines on the jobs), it is clear that an optimal solution to this model consists in executing all the jobs. What is the added value of this model?

Also, I do not understand the statement that "M is the theoretical upper bound of Cmax". What is the theoretical upper bound of Cmax?

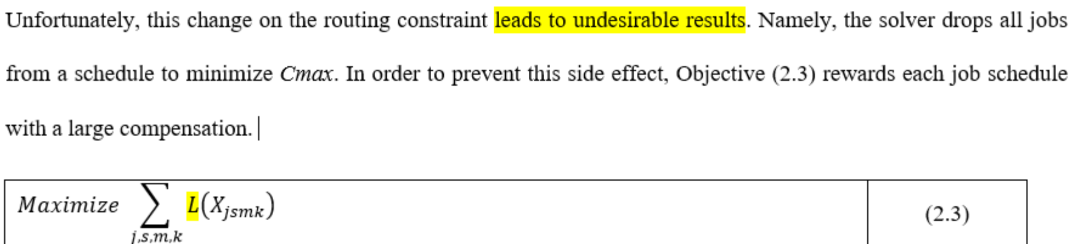
➔ The “M” is user for machine set, so I replaced “M” with “L”. The “L” is defined as a parameter in Notation section. It acts like Big-M.



$$L \qquad max\left\{\sum_{j,s,m} par^{ptime}_{j,s,m}\right\}$$

Why do we need this apparently useless constant factor in the maximization objective "maximize sum M.X_{jsmk}" ?

➔ Quote from the paper



Unfortunately, this change on the routing constraint leads to undesirable results. Namely, the 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.

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

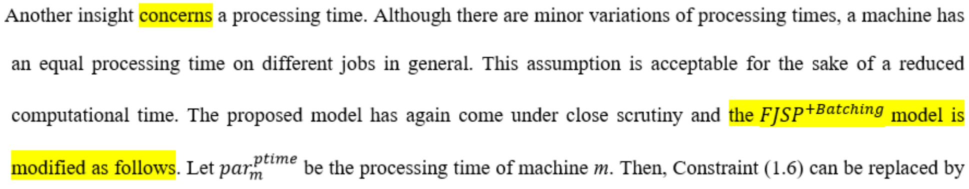
Then "This change reduces a (sic) number of nodes in branch-and-bound algorithm owing to smaller |K|". It reduces the number of nodes compared to what? The model described in this section maximizes the number of executed jobs, how can it result in smaller |K|, how does it relate to the original problem of minimizing the makespan? My feeling is that you first use this model to compute a feasible solution with all the jobs scheduled (but such a feasible solution could be computed using any heuristic) and this solution is used to compute a maximal value for the number of steps executed by a machine, and that this maximal number of steps is used as additional constraint to limit the search space of the original model. In this case, this means that the resulting model FJSP^+Batching_Selective is an over-constrained problem compared to the original problem. But then an optimal solution to FJSP^+Batching_Selective is not necessarily an optimal solution for the original problem and

this model cannot be used to give any solution quality guarantee for the original problem. On the same line, in the experimental section you cannot argument that "FJSP^+Batching_Selective is better than FJSP^+Batching because it finds more optimal solutions", just because (if I understand correctly), because FJSP^+Batching_Selective is over-constrained it can miss optimal solutions of the original problem.

➔ Sorry about the confusion, but you misunderstand what the paper is intended. Let me quote the paper, "The original model uses lots of binary variables (X_{jsmk}). One of the 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."

The second variant is described in item (b): "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." What compromise? I suppose it is the fact to consider that on a machine the processing time does not depend on the job compared to solving the real version of the problem where processing time are job-dependents. But then you should either show that this compromise is indeed acceptable (and the difference between Cmax values between Table 2 and Table 3 seem to suggest that processing times are highly job dependents otherwise you would get similar makespans whereas the optimal makespans computed with min values on Table 3 are about twice smaller that the ones on Table2) or take it for granted from the beginning and then only work with job-independent processing times.

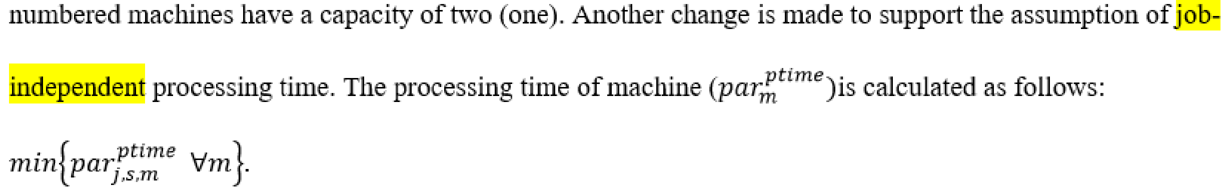
➔ I replaced "compromise" with "assumption". The test problem instances are taken from a literature (Fattahi *et al.* 2007) so its processing times are highly job dependents so that comparing the Cmax in Table 2 with the one in Table 3 is not valid. The assumption of machine-dependent processing times can be made in some of semiconductor manufacturing operations. I understand it would be better if we could use the instances from a real semiconductor manufacturing.



Another insight concerns a processing time. Although there are minor variations of processing times, a machine has an equal processing time on different jobs in general. This assumption is acceptable for the sake of a reduced computational time. The proposed model has again come under close scrutiny and the $FJSP^{+Batching}$ model is modified as follows. Let par_m^{ptime} be the processing time of machine m. Then, Constraint (1.6) can be replaced by

Furthermore, in the description of this variant in item (b) you should say how the "common" processing time is computed for a given machine: is it the min/max/average of the actual job-dependent processing times? Which steps are considered: all the ones potentially executable by the machine? And then, again, if you use a 'min' (as the experimental section seems to suggest) you get a relaxation of the real problem so this can be used to compute lower bounds for the actual problem but there is not any guarantee about the actual quality of the solutions to this relaxed problem.

➔ I use “Min” processing time.



numbered machines have a capacity of two (one). Another change is made to support the assumption of job-independent processing time. The processing time of machine (par_m^{ptime})is calculated as follows: $min\{par_{j,s,m}^{ptime}\ \ \forall m\}$.

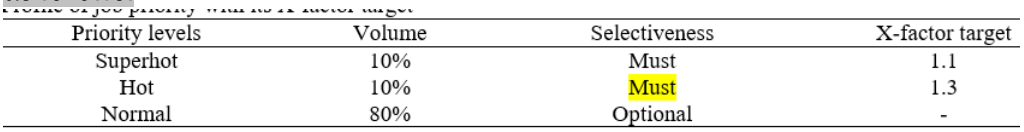
The above response can be applied here too. I changed the term “relaxation” to “modification”. What this paper is trying to say is … Whenever we see the problem with job-independent processing times like some of semiconductor manufacturing operations, we can apply this variant which can reduce a computational time.

The problem with priority jobs suffers the same issues as mentioned above: were job-dependent processing times somehow considered? How the two objectives mentioned in the model where "aggregated" ? What it the guarantee that setting |K| value to 4 does not rule out some optimal solutions?

➔ The condensed mathematical model can be found in the Appendix. The $FJSP_{PJS}^{+Batching}$ uses job-independent processing time. Regarding the K value of 4, please refer to the answer in below.

Also there are some inconsistencies in the description: Table 4 seems to suggest that "hot" jobs are optional whereas the description and the MIP model in appendix says that they "must" be scheduled.

➔Thanks for the feedback. “Superhot” and “Hot” are must be scheduled. It was a typo. It is revised as follows:

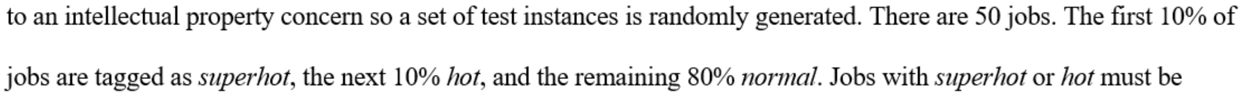


Profile of job priority with its X-factor target

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

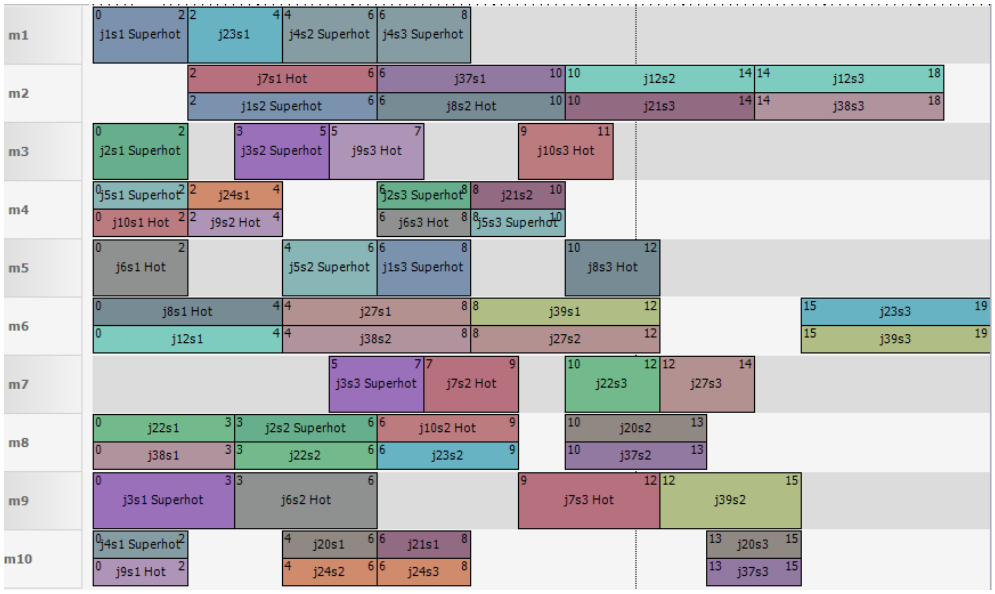
Also, isn't it surprising that among the 50 jobs of the benchmark, for all of the 10 instances it is exactly the same number of super-hot, hot and normal jobs that are effectively scheduled (resp. : 5, 5 and 10) ? Is it just by chance?

➔Among a total of 50 jobs, 5 are superhot, 5 are hot, 40 are normal jobs. The computational study shows the proposed method successfully schedules all superhots and hots while some of the normal jobs remain unscheduled.



to an intellectual property concern so a set of test instances is randomly generated. 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

Now, regarding the 5:5:10, the maximum count of possible operations which can be scheduled is 60 (= 4 (K) * 5 (single machines) + 4 (K) * 5 * 2 (batch machines)). Therefore, a total of 20 jobs having 3 steps for each (60=20*3) is the maximum the solver can schedule. Among the 20 jobs actually scheduled, 5 are superhot, 5 are hot, and 10 are normal as we anticipate. Please refer to the Figure 4.

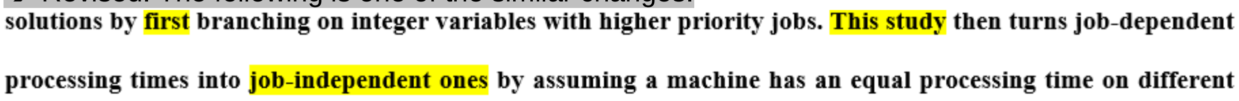


To summarize, I think there are interesting ideas in the MIP model proposed in section 3.1. But the "practical" improvements to this model need to be clarified. Furthermore, the presentation of the paper needs to be improved.

A few more detailed comments:

- In the abstract: "We then turn job-dependent processing time into machine-dependent". This is a bit misleading, I would rather say that you turn job-dependent processing time into job-independent ones". Same in title of Table 3 and probably in other places too.

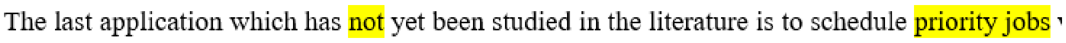
➔ Revised. The following is one of the similar changes.



solutions by first branching on integer variables with higher priority jobs. This study then turns job-dependent processing times into job-independent ones by assuming a machine has an equal processing time on different

- In introduction : "Last application which has yet been studied in the literature". Perhaps you mean "not yet"

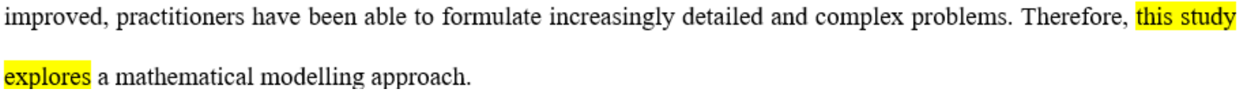
➔ Revised.



The last application which has not yet been studied in the literature is to schedule priority jobs

- Section 2.1: "we explorer" -> "we explore[d]" There are other occurrences of this "explorer"

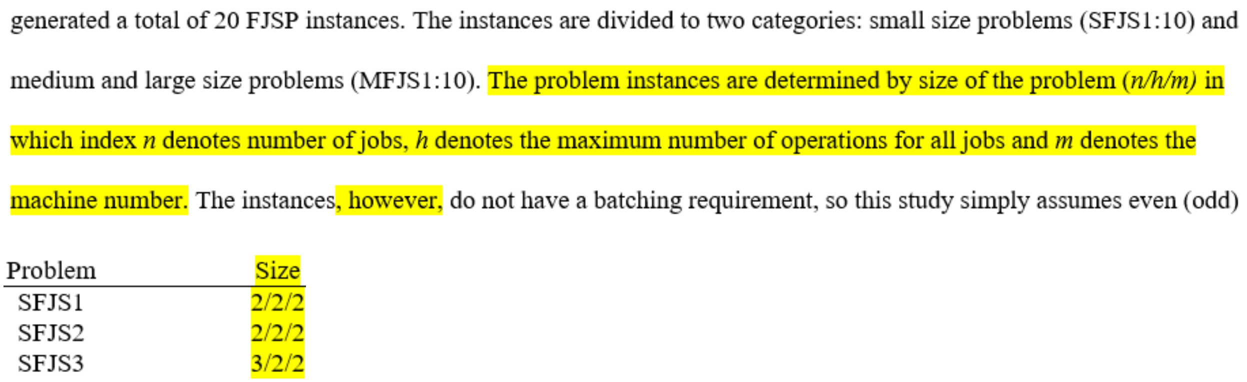
➔ Revised.



improved, practitioners have been able to formulate increasingly detailed and complex problems. Therefore, this study explores a mathematical modelling approach.

- For the benchmark used in section 4.1, you should mention the exact size of the instances (beside only mentioning small, medium and large size problems)

➔ Revised as follows



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 problem instances are determined by size of the problem (*n/h/m*) in which index *n* denotes number of jobs, *h* denotes the maximum number of operations for all jobs and *m* denotes the machine number. The instances, however, do not have a batching requirement, so this study simply assumes even (odd)

Problem	Size
SFJS1	2/2/2
SFJS2	2/2/2
SFJS3	3/2/2

- Section 4.3: "We need to modify the the model ..."

➔ Revised

The model is modified in order to calculate

- In the MIP model about priority jobs in appendix, it seems to be possible to get rid of variable J^{X+}_j by just replacing the equality in equation 3.3 by an inequality

➔ I agree. Since we are not penalizing the over achievement, we don't need J_j^{X+}. A change is made as follows:

$$\frac{J_j^{complete}}{par_j^{TCT}} \leq par_j^{Xfactor} + J_j^{X-} \quad \forall j$$

I really appreciate the detailed feedbacks and I feel sorry about my countless errors. I will take this process as a great lesson and try not to make a similar mistake again.

Best Regards,
Andy

Flexible Job Shop Scheduling Problem for Parallel Batch Processing Machine with Compatible Job Families

Andy Ham
Industrial & Systems Engineering, Liberty University
Lynchburg, VA, USA
mham@liberty.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, the formulation is modified to *selectively* schedule jobs. Although there are many jobs on a given floor, semiconductor manufacturing is most challenged by priority jobs that promise a significant amount of financial compensation in exchange for an expedited delivery. This modification could leave some non-priority jobs unscheduled. However, it vastly expedites the discovery of improving solutions by first branching on integer variables with higher priority jobs. This study then turns job-dependent processing times into job-independent ones 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 the industry. These changes significantly reduce computational time compared to the original model when tested on a set of common problem instances from the literature. Computational results show that this proposed model can generate an effective schedule for large problems. Author encourages other researchers to propose an improved MIP model.

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

1. INTRODUCTION

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

One application is a wet-diffusion area scheduling problem which has 2-4 consecutive steps with parallel batching machines [3,6,28]. Another application is to schedule jobs having the time constraint between consecutive process steps [2,10,20,29]. The last application which has not yet been studied in the literature is to schedule priority jobs which are often introduced into the factory for new product development or business considerations. A typical priority job travels on such an irregular process flow that it disrupts normal production and creates an inconsistent cycle. Furthermore, a floor supervisor often takes an extreme measure letting some ports of machine empty as the priority job is approaching from upstream steps due to the non-preemptive nature of machines in the industry. This results in a productivity loss. The industry calls it *priority job scheduler* (PJS). This study tackles this PJS problem in the context of FJSP with batching.

Considerable research has been devoted to FJSP in the literature. However, in consideration of the fact that there is no earlier work concerning FJSP with parallel batch processing machine (PBM), this paper attempts to propose a MIP model for FJSP with batching constraint for the first time and suggest a couple of modifications to reduce computational time. 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 Flexible job-shop scheduling problem

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 these heuristics may generate fast and effective solutions, they are usually tailor-made. The best combination of parameters, which lead to effective solutions, is difficult to find so researchers conduct extensive experiments solely for that purpose. Namely, the efficiency of these techniques strongly depends on a proper implementation and fine tuning of parameters since they combine a problem representation and a solution strategy into a single framework. In contrast, mathematical modelling approach separates a problem representation from a solution strategy [8]. Furthermore, as computer hardware and solvers have improved, practitioners have been able to formulate increasingly detailed and complex problems. Therefore, this study explores a mathematical modelling approach.

Table 1 shows an overview of FJSP mathematical models in the literature. The overview table created by Demir and İşleyen [33] is slightly modified. A vast number of studies have addressed FJSP and its variants like 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
[17]	1997	with sequence dependent setup times	EJOR
[18]	1999	with process plan flexibility	IJPR
[7]	2001	with alternative process plan	IJPE
[8]	2001	with sequence dependent setup times	IJPR
[15]	2001	with sequence dependent setup times	IEEE
[34]	2002	with process plan flexibility	C&IE
[32]	2005	with homogenous machines	IJPE
[23]	2006	with sequence dependent setup times	IEEE
[4]	2006	with sequence independent setup times	C&OR
[16]	2006	with flexible preventive maintenance	JIM
[26]	2007	-	JIM
[22]	2007	with sequence dependent setup times	IJAMT
[14]	2009	-	C&IE
[19]	2009	with sequence independent setup times	SETP
[25]	2009	with overlapping	AMM
[5]	2010	with process plan flexibility	AMM
[24]	2010	with disturbances	JMST
[21]	2010	-	IJPR
[12]	2011	with preventive maintenance	ESA
[27]	2012	with transportation constraints	C&OR
[33]	2013	evaluation of MIP models	AMM
[1]	2015	with sequence dependent setup times	JIM
[31]	2016	with transportation times	AMM

They also categorize FJSP mathematical formulations into 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

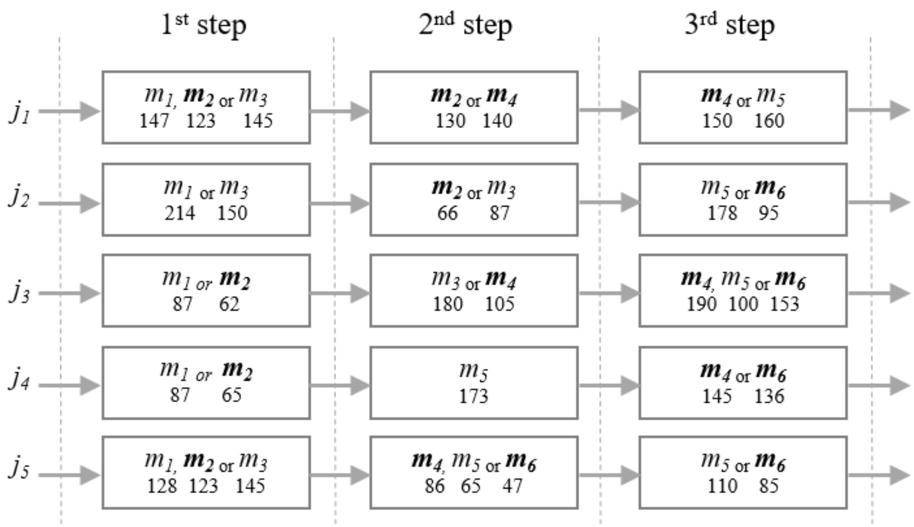
Business requirements drive the need for a small number of jobs to be run through the factory as fast as possible. Various manual and automated schemes have been tried to keep the priority jobs from "queuing at the machine". These schemes involve idling tools ahead of the arrival of priority jobs and trading machine utilization for priority jobs cycle time [28].

The main contributions of this paper can be summarized as follows. This study proposes a mathematical formulation of FJSP with batching constraint for the first time and makes a couple of modifications to reduce computational time for the PJS 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.



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.*[26]

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

The parallel batch scheduling problem with compatible job families arises at the burn-in operation in a back-end facility. Different products can be simultaneously processed in a machine because jobs may stay in a machine for a period longer than their minimum required burn-in times. On the contrary, jobs which belong to different families cannot be processed together (incompatible job families) in the diffusion operation in a front fabrication facility. This study assumes that the 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)
$K(\|J\|\times\|S\|)$	positions (k)

Parameters:

$par_j^{release}$	release time of job j
$par_{j,s,m}^{ptime}$	processing time
par_m^{capa}	capacity of machine m
L	$max\left\{\sum_{j,s,m} par_{j,s,m}^{ptime}\right\}$

Decision variables:

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

Resultant variables:

$J_{j,s}^{arrival}$ arrival time to step s of job j

$M_{m,k}^{start}$ start time at position k in the sequence of machine m

$M_{m,k}^{complete}$ completion time at position k in the sequence of machine m

$M_{m,k}^{ptime}$ processing time at position k in the sequence of machine m

$Cmax$ makespan

The following model is named $FJSP^{+Batching}$.

Routing	$\sum_{m,k} X_{jsmk} = 1 \quad \forall j,s$	(1.1)
	$\sum_{j,s} (X_{jsmk}) \le par_m^{capa} \quad \forall k,m$	(1.2)
Scheduling	$M_{m,k}^{ptime} \ge \left(par_{j,s,m}^{ptime}\right) X_{jsmk} \;\; \forall j,s,m,k$	(1.3)
	$J_{j,1}^{arrival} = par_j^{release} \;\; \forall j$	(1.4)
	$M_{m,k}^{start} \ge J_{j,s}^{arrival} + L(X_{jsmk} - 1) \quad \forall j,s,m,k$	(1.5)
	$M_{m,k}^{complete} = M_{m,k}^{start} + M_{m,k}^{ptime} \quad \forall m,k$	(1.6)
	$J_{j,s+1}^{arrival} \ge M_{m,k}^{complete} + L(X_{jsmk} - 1) \;\; \forall j,m,k : s < \lvert S \rvert$	(1.7)
	$M_{m,k+1}^{start} \ge M_{m,k}^{complete} \;\; \forall m : k < \lvert K \rvert$	(1.8)
Measuring	$Cmax \ge M_{m,k}^{complete} + L(X_{jsmk} - 1) \; \forall j,s,m,k$	(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 step just previous. Constraint (1.8) ensures the precedence relationship between batches on the same machine. Finally, Constraint (1.9) determines the makespan and Objective (1.10) minimizes it.

For the incompatible job families, let Q_{mkf} be $\in \{0,1\}$ a derived variable to indicate whether a job family f is scheduled to position k in the sequence of machine m. We can then limit the total count of different job families at each position k of machine m.

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 each 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.

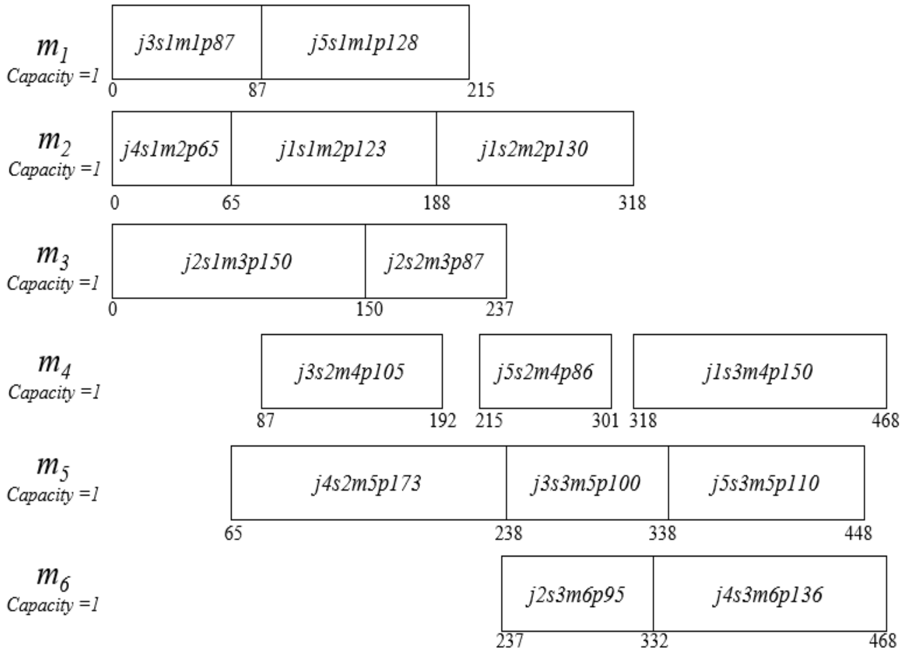


Fig. 2. An optimal solution for the 5-job, 6-machine, and 3-step w/o batching

On the other hand, Figure 3 represents an optimal solution for the same instance when batching is considered, for instance, Machine 2 processes a batch which is composed of *j*1 and *j*3.

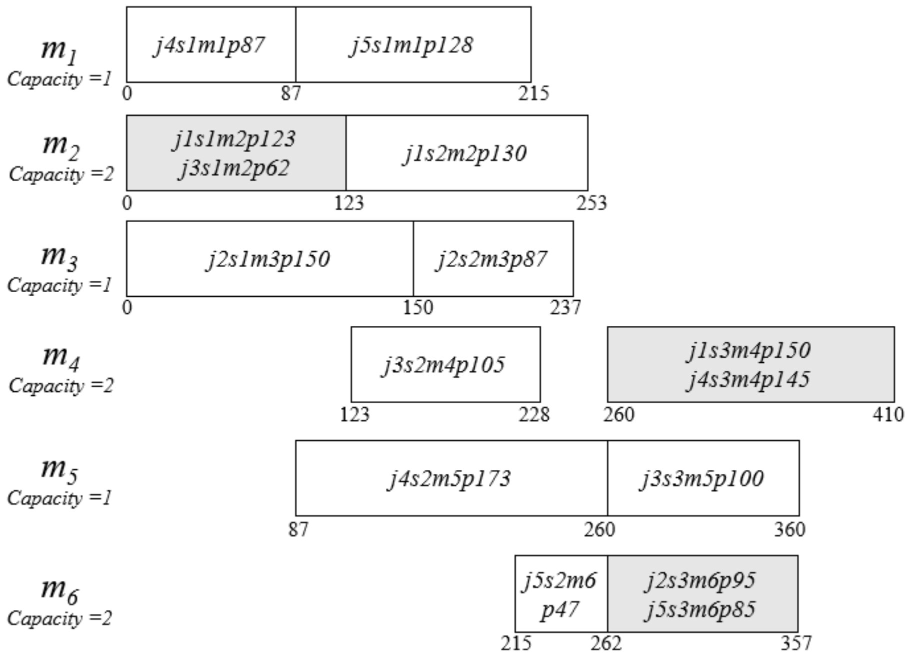


Fig. 3. An optimal solution for the 5-job, 6-machine, and 3-step w/ batching

3.2 FJSP with batching applied to priority job scheduling

The $FJSP^{+Batching}$ model could not generate an effective solution for large instances after several hours of computational time during a preliminary experimentation, so this study explores an opportunity of practical modification. An interview with an industry subject matter expert (SME) provides the following two key insights:

a) One key insight concerns a *selective* job scheduling. Although there are many jobs on a given floor, the industry is most interested in priority jobs that promise a significant amount of financial compensation in exchange for an expedited delivery. To take advantage of this finding, the proposed model has come under close scrutiny. The model uses many binary variables (X_{jsmk}). One of the set indexes which caught our eye is k, positions, 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 if the model allows some jobs to remain unscheduled. Now, the routing Constraint (1.1) is modified as follows:

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

Constraint (2.2) is added in order to force a job to complete all operations once it is selected for the 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 leads to undesirable results. Namely, the 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.

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

This study then calculates a completion time of each individual job and uses the time to calculate the makespan as shown on Constraints (2.4) and (2.5), which replace Constraint (1.9).

$$J_j^{complete} \geq M_{m,k}^{complete} + L(X_{jsmk} - 1) \;\; \forall jsmk \tag{2.4}$$

$$Cmax \geq J_j^{complete} \quad \forall j \tag{2.5}$$

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

b) Another insight concerns a processing time. Although there are minor variations of processing times, a machine has an equal processing time on different jobs in general. This assumption is acceptable for the sake of a reduced computational time. The proposed model has again come under close scrutiny and the $FJSP^{+Batching}$ model is modified 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. It is discussed in the computational study section.

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

New objectives are to maximize the total count of jobs scheduled while minimizing the last completion time.

4. Computational Study

In this section, the effectiveness of our proposed models is tested. Firstly, $FJSP^{+Batching}$ and $FJSP_{Selective}^{+Batching}$ are compared. Then, this study conducts a similar study with the assumption of job-independent processing time.

The proposed 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, the same test problem instances created by Fattahi *et al*. [26] are borrowed. 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 problem instances are determined by size of the problem (*n/h/m)* in which index *n* denotes number of jobs, *h* denotes the maximum number of operations for all jobs and *m* denotes the machine number. The instances, however, do not have a batching requirement, so this study simply assumes even (odd) numbered machines have a capacity of two (one). Another change is made to support the assumption of job-independent processing time. The processing time of machine (par_m^{ptime})is calculated as follows: $min\left\{par_{j,s,m}^{ptime} \quad \forall m\right\}$.

The $FJSP_{Selective}^{+Batching}$ is set to schedule all jobs for a fair comparison with $FJSP^{+Batching}$, although it selectively chooses jobs to schedule. This is made by setting |*K*| value to six. We have confirmed the model schedules all jobs during a study. The customized CPLEX logs have all the 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. Columns 1–2 show the name of instances used by Fattahi *et al*. [26] and size, respectively. Columns 3–6 contain the best solutions generated by $FJSP^{+Batching}$ and $FJSP^{+Batching}_{Selective}$ which are reported within 300 seconds.

The $FJSP^{+Batching}_{Selective}$ finds an optimal solutions of 11 instances out of 20 compared to 9 out of 20 by the $FJSP^{+Batching}$. Also, the average *Cmax* values of $FJSP^{+Batching}_{Selective}$ ($FJSP^{+Batching}$) are 637.1 (899.4) respectively for the large size instances. The statistic does not include the MFJS9 instance since $FJSP^{+Batching}$ fails to report any feasible solution.

In terms of computational time, $FJSP^{+Batching}_{Selective}$ finds the optimal solutions in 1.16 seconds on average compared to 3.44 seconds by $FJSP^{+Batching}$, for the small size problem instances. The statistic does not include the SFJS10 instance since $FJSP^{+Batching}$ fails to reach an optimal.

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

		$FJSP^{+Batching}$		$FJSP^{+Batching}_{Selective}$	
Problem	Size	*Cmax*	CPU	*Cmax*	CPU
SFJS1	2/2/2	**66**	0.05	**66**	0.08
SFJS2	2/2/2	**107**	0.05	**107**	0.05
SFJS3	3/2/2	**208**	0.31	**208**	0.20
SFJS4	3/2/2	**272**	0.06	**272**	0.05
SFJS5	3/2/2	**100**	0.14	**100**	0.39
SFJS6	3/3/2	**320**	12.29	**320**	1.37
SFJS7	3/3/5	**397**	9.77	**397**	1.00
SFJS8	3/3/4	**216**	5.34	**216**	6.46
SFJS9	3/3/3	**210**	2.95	**210**	0.87
SFJS10	4/3/5	516	300	**516**	91.96
MFJS1	5/3/6	410	300	410	300
MFJS2	5/3/7	410	300	410	300
MFJS3	6/3/7	420	300	420	300

MFJS4	7/3/7	506	300	506	300
MFJS5	7/3/7	488	300	488	300
MFJS6	8/3/7	631	300	**614**	42.43
MFJS7	8/4/7	916	300	863	300
MFJS8	9/4/8	896	300	808	300
MFJS9	11/4/8	nf	300	955	300
MFJS10	12/4/8	3418	300	1215	300

* bold font indicates an optimal.

Table 3 summarizes the computational results based on the assumption of job-independent processing time. Under this assumption, both $FJSP^{+Batching}$and $FJSP^{+Batching}_{Selective}$ find optimal solutions of 16 instances out of 20 within 300 seconds which demonstrates a reduction of computational time by tightening the formulation. There are also 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 job-independent 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
Average	309.5	63.8	212.5	68.8

* bold font indicates an optimal.

After confirming both modifications could significantly reduce computational time, the proposed model is applied to the priority job scheduling problem encountered in semiconductor manufacturing.

4.3 Priority job scheduling test instances

An industry SME directed this study to look into the test problem instances especially with 3-step which typically takes 2~3 hours for wafers to flow through in semiconductor manufacturing. A real instance could not be obtained due to an intellectual property concern so a set of test instances was randomly generated. 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 this is optional.

The new performance measures are to maximize a total amount of production (or jobs scheduled) and to meet a target of X-factor [9] 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. The model is modified in order to calculate the X-factor of a job. This variant is named $FJSP_{PJS}^{+Batching}$ and the mathematical model is provided in the 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%	Must	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. The size of $|K|$ is set as 4 and the computational study confirms 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 an optimal solution.

Table 5
Optimal schedules of priority job scheduling.

	Superhot jobs			Hot jobs			Normal jobs			
Problem	Jobs	Avg	Max	Jobs	Avg	Max	Jobs	Avg	Max	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. 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 0 and completing at 2.

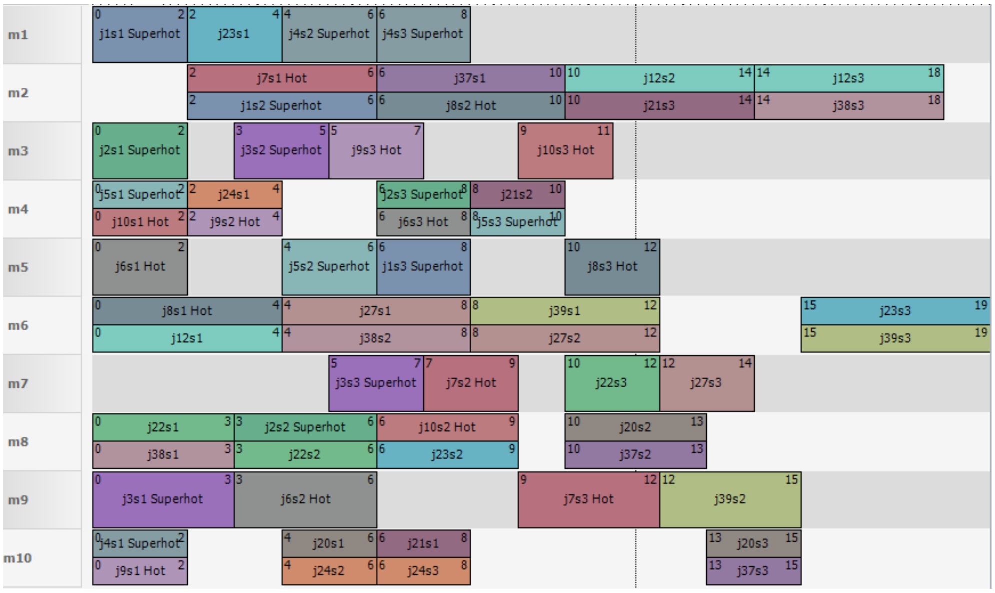


Fig. 4. Gantt chart schedule of test instance of PJS01

5. CONCLUSION AND FUTURE RESEARCH

Encountered at the semiconductor manufacturing, a flexible job-shop scheduling problem with parallel batch processing machine is studied as there is a growing need for orchestrating a whole factory to seek a global optimization. There are several immediate applications of the FJSP with batching in semiconductor manufacturing. One of the applications is a wet-diffusion area scheduling problem which has 2-4 consecutive steps with parallel batching machines. Another application is to schedule jobs having time constraints between consecutive process steps. The last application is to schedule priority jobs, which are often introduced into the factory out of new product development or business considerations. They often follow such an irregular flow of steps that it greatly disrupts normal production, not to mention its high priority over normal jobs. A floor supervisor often takes an extreme measure, letting some ports of machine idle as priority jobs are approaching from upstream steps which results in a productivity loss. This study addresses the priority job scheduling problem in the context of FJSP with batching.

A mixed integer programming (MIP) formulation is composed for the first time. Owing to two critical insights found during an interview with an industry SME, the mathematical formulation is modified to cope with industry-size instances. The first insight is about a *selective* job scheduling. Although there are thousands of jobs on a floor, the industry is most interested in priority jobs because special customers promise a significant amount of financial compensation in exchange for 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 this is optional. The second insight concerns the job-independent processing time. A machine has a similar processing time on different jobs in general, although there are minor variations of processing times depending on recipes. This assumption 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 modified model can schedule 50-job, 10-machine, and 3-step within 2.5 minutes run-time. The model successfully schedules all jobs tagged as *superhot* or *hot* within their due dates, while filling a machine idle space with *normal* jobs to maximize the production output. This study demonstrates how to orchestrate multiple batching machines sitting on a series of consecutive steps by using a mathematical model. The floor does not need to let some ports of machine idle as priority jobs are approaching since the proposed model effectively combines the priority jobs with the normal jobs and generates an optimal schedule.

This research can be further extended by considering a relatively new approach, constraint programming (CP), which is designed to cope with complex scheduling problems such as TSP, JSP, and FJSP [30]. Another extension is to improve the proposed MIP model. Lastly, application opportunities can be further explored as discussed in the introduction section.

Appendix.

$FJSP_{Selective}^{+Batching}$ **model in condensed 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	$J_{j,1}^{arrival} = par_j^{release} \ \forall j$	(1.4)
	$M_{m,k}^{start} \geq J_{j,s}^{arrival} + L(X_{jsmk} - 1) \ \forall j,s,m,k$	(1.5)
	$M_{m,k}^{complete} = M_{m,k}^{start} + par_m^{ptime} \quad \forall m,k$	(2.6)
	$J_{j,s+1}^{arrival} \geq M_{m,k}^{complete} + L(X_{jsmk} - 1)$ $\forall j,m,k : s < \lvert S \rvert$	(1.7)
	$M_{m,k+1}^{start} \geq M_{m,k}^{complete} \ \forall m : k < \lvert K \rvert$	(1.8)
Measuring	$J_j^{complete} \geq M_{m,k}^{complete} + L(X_{jsmk} - 1) \ \forall jsmk$	(2.4)
	$Cmax \geq J_j^{complete} \ \forall j$	(2.5)
	$Minimize\ Cmax$	(1.10)
	$Maximize \sum_{j,s,m,k} L(X_{jsmk})$	(2.3)

$FJSP_{PJS}^{+Batching}$ **model in condensed 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 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

Routing	$\sum_{m,k} X_{jsmk} = 1 \quad \forall j,s : par_j^{pri} = superhot\ or\ hot$	(3.1)
	$\sum_{m,k} X_{jsmk} \le 1 \quad \forall j,s : par_j^{pri} = normal$	(3.2)
	$\sum_{mk} X_{jsmk} = \sum_{mk} X_{j,s+1,m,k} \ \ \forall j : s < \lvert S \rvert$	(2.2)
	$\sum_{j,s} (X_{jsmk}) \le par_m^{capa} \quad \forall k,m$	(1.2)
Scheduling	$J_{j,1}^{arrival} = par_j^{release} \quad \forall j$	(1.4)
	$M_{m,k}^{start} \ge J_{j,s}^{arrival} + L(X_{jsmk} - 1) \quad \forall j,s,m,k$	(1.5)
	$M_{m,k}^{complete} = M_{m,k}^{start} + par_m^{ptime} \quad \forall m,k$	(2.6)
	$J_{j,s+1}^{arrival} \ge M_{m,k}^{complete} + L(X_{jsmk} - 1) \ \forall j,m,k : s < \lvert S \rvert$	(1.7)
	$M_{m,k+1}^{start} \ge M_{m,k}^{complete} \ \ \forall m : k < \lvert K \rvert$	(1.8)
Measuring	$J_j^{complete} \ge M_{m,k}^{complete} + L(X_{jsmk} - 1) \ \forall jsmk$	(2.4)
	$\frac{J_j^{complete}}{par_j^{TCT}} \le par_j^{Xfactor} + J_j^{\mathrm{X}-} \ \ \forall j$	(3.3)
	$Maximize \sum_{j,s,m,k} (L/par_j^{pri}) X_{jsmk}$	(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 [11]. This change instructs CPLEX to branch on integer variables with higher priority jobs first. Objective (3.5) penalizes the amount of X-factor underaccomplished compared to a target.

References

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

[2] A. Klemmt, & M?nch, 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*.

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

[5] C. Özgüven, L. Özbakır, Y. Yavuz, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, *Applied Mathematical Modelling*, 34 (2010) 1539–1548.

[6] C. Yugma, Dauz?re-P?r?s, 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.

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

[8] C.Y. Low, T.H. Wu, Mathematical modelling and heuristic approaches to operation scheduling problems in an FMS environment, *International Journal of Production Research*, 39 (2001) 689-708.

[9] D. P. Martin, (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).

[10] D. S. Sun, 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, IEEE International Symposium on* (pp. 295-298).

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

[12] 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.

[13] G. M. Kopanos, M?ndez, 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] M. Ham, 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.

[21] M. Ham, 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.

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

[23] 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.

[24] 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.

[25] P. Fattahi, F. Jolai, J. Arkat, Flexible job shop scheduling with overlapping in operations, *Applied Mathematical Modelling*, 33 (2009) 3076-3087.

[26] 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.

[27] 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.

[28] R. Bixby, 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 Conferenc*e (ASMC-2006) (pp. 148-154).

[29] R. Sadeghi, 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.

[30] R. Sahraeian, & Namakshenas, M. (2015). On the optimal modeling and evaluation of job shops with a total weighted tardiness objective: Constraint programming vs. mixed integer programming. *Applied Mathematical Modelling*, 39(2), 955-964.

[31] S. Karimi, Ardalan, Z., Naderi, B., & Mohammadi, M. (2016). Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*.

[32] 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.

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

[34] 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.