A Heuristic Method for Job-Shop Scheduling with an Infinite Wait Buffer

- From One-Machine to Multi-Machine Problems

Z. J. Zhao * J. Kim M. Luo H. C. Lau S. S. Ge J.B. Zhang

***Abstract*—Through empirical comparison of classical Job Shop Problems (JSP) with multi-machine consideration, we find that the objective to minimize the sum of weighted tardiness has a better wait property compared with the objective to minimize the makespan. Further, we test the proposed Iterative Minimization Micro-model (IMM) heuristic method with the Mixed Integer Programming (MIP) solution by CPLEX. For multi-machine problems, the IMM heuristic method is faster and achieves a better solution. Finally, for a large problem instance with 409 jobs and 30 types of machines, IMM-heuristic method is compared with ProModel and we find that the heuristic method is slightly better.**

***Index Terms*—Job Shop Scheduling, Multiple Machine, infinite in-process wait buffer**

I. Introduction

Multi-machine shop scheduling problems (job-shop, flow-shop and etc.) receive more and more attention ever since 10 years ago, as it can greatly improve the production rate and reliability. Its application can be found in logistics, semiconductor manufacturing, robotics, and etc.

In [3], there is a broad classification for classical one-machine problem. In [6], the same notation follows and further, an extensive review for multi-machine problems is given. Multi-machine problems are usually studied in one of the following cases:

Case1: There are multiple exchangeable and renewable machines C_k for each machine type k, and each specific machine could perform no more than one operation at any time;

Case2: There is exactly one machine for each type k, which can perform as many as C_k operations at any time, and $C_k \geq 1$;

Case3: There are cases with combination of Case 1 and Case 2.

Z. J. Zhao is with School of Information System, Singapore Management University. He is currently a PhD candidate at the Electrical and Computer Engineering Department of the National University of Singapore, 117576 Singapore. Tel.: +65-68518564. Fax: +65-67791103. Email: zzytgx@yahoo.com.cn

* Corresponding author

J. Kim is with the Institute of Transportation & Logistics, UTAC, Korea

M. Luo and J.B.Zhang are with the Singapore Institute of Manufacturing Technology

H.C. Lau is with School of Information Systems, Singapore Management University

S.S. Ge is with the Electrical and Computer Engineering Department of the National University of Singapore.

This research is partially funded by the A*STAR SERC TSRP GRANT P0520104 and IMSS TSRP 052 116 0075.

From *Formulation's* point of view, the *Differences* between one-machine problem and multi-machine problem are noted as following:

FD-1: One-machine problem can be formulated as disjunctive graph [2], which could be easily mapped to MIP model and the processing time can be non-integers.

FD-2: For multi-machine problem, it is so far not yet nicely formulated in our opinion especially when processing time is non-integer. In [8], a model is proposed with $\delta()$ function, which can only be solved by heuristic methods and the optimality is not guaranteed for all cases. For integer processing time, in [4], the Pritsker's formulation [7] was applied and a 0-1 Integer Problem (IP) is solved.

With respect to the *Solutions*, there are following *Differences*, which will be further explained later.

SD-1: One-machine problem has a solution as the task schedule, while the multi-machine problem's solution includes task schedule and machine dispatching.

SD-2: The solution of task schedule for both one-machine problem and multi-machine problem is unique, while solution for machine dispatching in multi-machine problem is not unique.

In this paper, our work is focused on the multi-machine JSP under Case 1. In order to make benchmark with optimal solutions, we use Pritsker's 0-1 formulation and all processing time are integers. Although we assume infinite wait buffer in the problem formulation, we can see that the wait-in-process is reduced when machine capacity is increased. An iterative minimization micro-model is implemented to solve the overall problem iteratively. The solution performance is bench-marked with the CPLEX solution for small-size problems and compared with ProModel solution for large-size problems. The notation of this paper is summarized in Table I.

II. Sample Problem: MT6, $C_k = 2, \forall k$

For a clear demonstration, we use the MT6 problem [5], where totally there are 6 jobs and each job has 6 tasks on 6 different types of machines. The original MT6 problem in [5] is one-machine problem $C_k = 1, k \in \{1, 2, ..., 6\}$, while our sample problem differs in $C_k = 2, k \in \{1, 2, ..., 6\}$. The solution Gantt chart is shown in Fig. 1 where Fig. 1(a) is represented by job-grouping and Fig. 1(b) is by machine-grouping. Both Fig. 1(a) and Fig. 1(b) are actually from the same solution. Y-axis in Fig. 1(a) is job-id, while in Fig. 1(b), integer represents machine type and fraction represents machine-id of

978–1–4244–1674–5/08/$25.00 © 2008 IEEE

TABLE I
NOTATION FOR JOBS, PROCESSES MACHINES

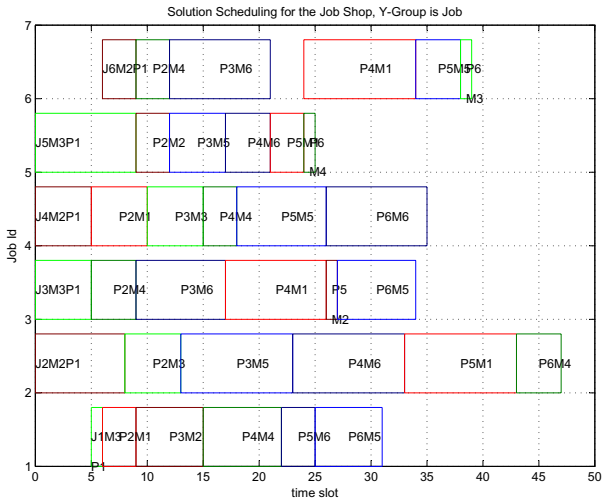
Notation for Jobs and Processes	
N	Total number of jobs
o_i	Total number of operations (i.e. processes) for job i
p_{ij}	processing time of job i process j $i \in \{1, ..., N\}$ and $j \in \{1, ..., o_i\}$
d_i	Due time of job i
W_i	Delay penalty per unit time if job i is completed after its due time
Notation for Machines	
K	Total number of machine types
k	Machine type index, $k \in \{1, ..., K\}$
$\tilde{T}$	Total time slot in discrete time formulation
C_k	Capacity of machine k, i.e. number of operations that can be executed at anytime on machine type k
M_{ij}	Mapping from job i and process j to machine type
Variables in Process Scheduling	
X_{ijt}	discrete decision variable for i^{th} job, j^{th} process at time slot t
T^s_{ij}	start time of process j in job i
T^e_{ij}	completion time of process j in job i
$M_k(t)$	Machine utilization function in continuous time Each k indicates a specific machine type
Notation in the IMM heuristic method	
R	Iterative number
$L_k(R)$	total number of executable tasks on machine type k at iteration R; executable in the sense of precedence constraints
t^a_l	arrive time for task l, $1 \leq l \leq L_k(R)$
t^r_m	release time for machine-id m of type k $1 \leq m \leq C_k$
$mi(l)$	a 1-to-1 mapping from task-id $l \in \{1, 2, ..., L_k(R)\}$ to machine-id $mi \in \{1, 2, ..., C_k\}$ $mi(l) \in \Pi(L_k(R) \rightarrow C_k)$
$\overline{RTAM}$	Mean Release Time of all machines after scheduling
$\overline{CTAJ}$	Mean Completion Time of all jobs
T^M	Makespan of scheduling
$\overline{TW}$	Total Wait count
$\overline{WT}$	Mean Wait Time
WT^M	Max Wait Time
$\overline{AMU}$	Mean percentage of Machine Utilization of all

that type. For the multiple solutions in machine dispatching, which is mentioned previously in **SD-2**, it can be explained that in Fig. 1(b), exchanging the machine-id (fractional part of machine scheduling) will not affect the task scheduling.

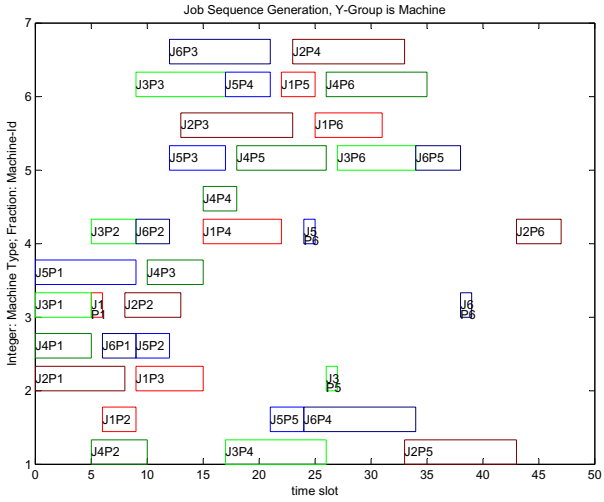
We further group the Gantt chart in colors, for the detailed readers to verify that Fig. 1(a) and Fig. 1(b) are actually the same solution. In Fig. 1(a), the same color represents the same machine type, while in Fig. 1(b), the same color represents the same job-id.

III. A 0-1 FORMULATION

A 0-1 formulation is proposed by [7], which might be the only solvable model for rigorous multi-machine problems.



(a) schedule (by job)



(b) schedule (by machine type (integer) and ID(fraction))

Fig. 1. Sample Schedule

A. Decision variable

Before formulating the problem in discrete time domain, one has to estimate a proper time unit $U_T \in \mathcal{R}$, with which the continuous time domain is discretized to $\tilde{T} \in \mathcal{N}$ time slots in total. This estimation is introduced [4] [8], and the total time period $U_T\tilde{T}$ could be available by heuristic solution. In discrete time domain, $t \in \{1, 2, ..., \tilde{T}\}$ is used to represent each time slot. A binary decision variable is X_{ijt}, where

$$X_{ijt} = \begin{cases} 1 & \text{if process } j \text{ in job } i \text{ starts} \\ & \text{by time } t \text{ inclusively;} \\ 0 & \text{if process } j \text{ in job } i \\ & \text{has not yet started time at } t \end{cases}.$$

There are following constraints by assumption:

$$X_{ijt} - X_{i,j,t+1} \leq 0, \qquad \forall i, j, t \in \{1, 2, ..., \tilde{T} - 1\}. \quad (1)$$

$$X_{ijt} \in \{0, 1\} \qquad \forall i, j, t \in \{1, 2, ..., \tilde{T}\}. \quad (2)$$

Inequality (1) assumes that every task is non-preemptive.

$$\left\{ \begin{array}{ll} X_{i,j,t} - X_{i,j-1,t-p_{i,j-1}} & \text{if } t > p_{i,j-1} \\ X_{i,j,t} & \text{if } t \leq p_{i,j-1} \end{array} \right\} \leq 0, \forall i,j \in \{2,...,o_i\}. \quad (3)$$

$$\sum_{i,j:M_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > p_{ij} & (X_{ijt} - X_{i,j,t-p_{ij}}) \\ \text{if } t \leq p_{ij} & X_{ijt} \end{array} \right\} - C_k \leq 0, \forall k \in \{1,2,...,K\} \quad (4)$$

B. Constraints on task precedence and machine capacity

Task precedence constraint is specified as inequality (3), which means that in-process wait is allowed. We name it to be an *Infinite Wait Buffer* because the model does not specify on:

- wait buffer capacity, which is usually related with machine settings,
- upper and lower bounds on wait time, which is usually from specifications of task schedule.

Machine capacity constraint is stated as the inequality (4), which is well studied in [8].

C. Objective functions

There are usually two types of criterions *CRI-1* and *CRI-2*:

CRI-1: The objective is to minimize the makespan, which by nature, is to minimize the time length from the starting time of very first task to completion time of the very last task.

CRI-2: The objective is to minimize the sum of weighted tardiness, which actually, is to minimize some areas.

Their formulations are following:

CRI-1: For a general job-shop problem, two types of dummy variables $\{X_{S,t}, X_{E,t} : t = 1,2,...,\tilde{T}\}$ are introduced to formulate the *Start* time of very first task, and the *End* time of the very last task. See (5) and the inequality constraints (6) and (7).

$$\text{minimize} \sum_t (1 - X_{E,t}) - \sum_t (1 - X_{S,t}) \quad (5)$$

$$X_{i,j,t} - X_{S,t} \leq 0, \forall i,j,t \quad (6)$$

$$X_{E,t} - X_{i,j,t} \leq 0, \forall i,j,t \quad (7)$$

CRI-2: The most generalized formulation introduces two lists of parameters, $W_i : i = 1,2,...,N$ for each job's weight (or importance) and $d_i : i = 1,2,...,N$ for each job's due time. For a special kind of flow shop with some sequencing constraints like [8], by adjusting the parameter W_i, (8) represents either minimizing the makespan or minimizing the sum of weighted tardiness.

$$\text{minimize} \sum_i (W_i \sum_{t > d_i - p_{i,o_i}} (1 - X_{i,o_i,t})) \quad (8)$$

D. An empirical comparison of above two criterions

Classical jobshop problems are usually to minimize the makespan [1] [2]. However, we observe that the overall final machine release time, which is more important in our opinion especially for multi-machine problems and it is related with the sum of weighted completion time. Some experiments are given to compare the two criterions, the sample problems are chosen to be $\{MT6, MT10\}$ at machine capacity $C_k \in \{1,2,3,...\}$.

Some terms are defined at first. There is wait between i^{th} job's j^{th} process and its precedence process iff. there is $T^e_{i,j-1} \neq T^s_{ij}; i = 1,...,N, j = 2,...,o_i$.

- *Total Wait:* TW - the total element count in $\{(i,j)|T^e_{i,j-1} \neq T^s_{ij}, i = 1,...,N, j = 2,...,o_i\}$;
- *Max Wait Time:* $WT^M = \max\{T^s_{ij} - T^e_{i,j-1} | i = 1,...,N; j = 2,...,o_i\}$;
- *Mean Wait Time:* $\overline{WT} = \frac{TotalWaitTime}{TotalWait} = \frac{\sum_{i,j}\{T^s_{ij} - T^e_{i,j-1} | i=1,...,N; j=2,...,o_i\}}{TW}$.

Relation of continuous variable T^s_{ij} and discrete variable X_{ijt} are studied in [8] and listed as follows:

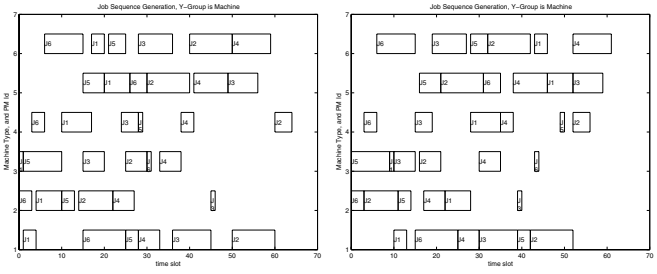
$$X_{i,j,t^*-1} == 0 \quad \cap \quad X_{i,j,t^*} == 1$$
$$\Updownarrow$$
$$T^s_{ij} = t^* - 1, i^{th} \text{ job's } j^{th} \text{ process starts at the time slot } t^*.$$

For the terms related with the machine utilizations, our definition is based on previous work [8] w.r.t. the construction of machine utilization function $M_k(t)$. Their definitions are shown as following.

- *Mean Machine Utilization for type k:* $\frac{\int_t M_k(t)dt}{C_k}$;
- *Mean Utilization All Machines:* $\overline{AMU} = \frac{\sum_k \int_t M_k(t)dt}{\sum_k C_k}$,

where the integration is from the very first time when the machine type k is put to use to the very last time when the machine type k stops using.

- *Mean Complete Time* is defined as $\overline{CTAJ} = \frac{T^e_{i,o_i}}{N}$. For the discrete formulation under assumption that all jobs' first process is arrived at time 0, it is equivalent to $\frac{\sum_i \sum_{t>0}(1 - X_{i,o_i,t})}{N}$.



(a) To minimize sum of completion time (by CRI-2)

(b) To minimize the makespan (by CRI-1)

Fig. 2. Criterion Comparison: One-machine MT6 problem, grouping by machine

Fig. 2 demonstrates a one-machine MT6 optimal solution by CPLEX. Fig. 2(a) is to minimize the sum of completion time (or **criterion-2** with $d_i = 0, W_i = 1, \forall i$), while Fig. 2(b) is to minimize the makespan as **criterion-1**. The authors suggest to focus on only comparison of the length (*makespan*) and the area of last task with x-axis (*sum of weighted tardiness*). Here, we use gray figure because it is a *1-machine MT6* problem, in contrast with Fig. 1 which is *2-machine MT6* problem. It should be noticed that Fig. 1 is for two representation of the same problem (actually criterion-2) while Fig. 2 is for 2 different problems (criterion-2 in Fig. 2(a) and criterion-1 in Fig. 2(b)).

Precisely, Table IV is used to compare {1, 2}-machine by CRI-{1, 2} on the MT6 and the MT10 problems. All schedule solutions are done by CPLEX.

From Table IV, we have following observations:

- When the total number of machines is increasing, both the makespan and the sum of weighted tardiness are dropping. When it reaches some level, both will keep the same value. This is similar to the Infinite Resource Model in [8].
- The sum of weighted tardiness (criterion-2) model is consistently better in the wait property (i.e. less in total wait count and the wait time) and the mean complete time than makespan model (criterion-2)
- For the mean machine release time, the 1-machine problem and the multi-machine problem are contrary to each other, while for the makespan and the machine utilization, there is no consistency.

E. Complete model for comparison with heuristic method

Based on above observation, we will further study our heuristic method for multi-machine job-shop with **CRI-2** as objective to minimize. For simplicity without loss of generality, $d_i = 0, W_i = 1, \forall i$. The complete model to benchmark our IMM heuristic method is shown as follows.

JSP	$\mathbf{MultiMach - InfWait}$
minimize	SumCompleteTime(8)
subject to:	Inequality(1)
	Inequality(3)
	Inequality(4)
	0-1 Variable as(2)

This is the formulation for CPLEX solution of a 2-machine MT6 problem and shown in Fig. 1.

IV. Heuristic Solution: Iterative Minimization of a Micro-model

The micro-model is defined for one particular machine type k with capacity C_k, on which, totally $L_k(R)$ processes are to be done. This model is solved and updated iteratively for the overall multi-machine JSP. R is a counter for iterations, and $L_k(R)$ is the total number of executable tasks on machine type k. "Executable" means satisfying sequencing constraints.

A. Minimization of a micro-model

In the Micro-model, there are C_k exchangeable and renewable machines, serving $L_k(R)$ tasks. Each machine has a release time $t^r_m : m \in \{1, 2, ..., C_k\}$ while each task has an arrival time $t^a_l : l \in \{1, 2, ..., L_k(R)\}$ and processing time $p_l : l \in \{1, 2, ..., L_k(R)\}$.

- t^r_m is a time for a particular machine m to be ready to start a new task.
- t^a_l is a time for task l to be ready to start. It is the completion time of task l's precedence task which is specified in sequencing constraint.
- $p_l, 1 \le l \le L_k(R)$, processing time for task l.

The problem is to minimize the sum of completion (or end) time. The summation is done for $\min\{C_k, L_k(R)\}$ tasks.

$$\text{Minimize: } {}_{mi(l)} \sum_{mi(l)\in\{1,...,C_k\}, l\in\{1,2,...,L_k(R)\}} t^e_l \tag{9}$$

$$\text{subject to: } t^s_l = \max\{t^a_l, t^r_{mi(l)}\} \forall l \tag{10}$$

$$t^e_l = t^s_l + p_l, \forall l \tag{11}$$

$$\text{decision variable: } mi(l) \in \Pi(L_k(R) \to C_k) \tag{12}$$

$$t^s_l, t^e_l \in \mathcal{R}, \forall l = 1, 2, ..., L_k(R) \tag{13}$$

This micro-model is to minimize the sum of end time in schedulable and executable task schedule as in (9) under the constraints of (10) and (11) with variable mapping (12), and the time variable lists (13).

From (9), total number of items in the summation is $\min\{C_k, L_k(R)\}$ where C_k is for scheduleable constraints by the machine availability and $L_k(R)$ is for executable constraints of precedence. In (9), $mi(l)$ is a 1-to-1 mapping from the task-id $l \in \{1, 2, ..., L_k(R)\}$ to the machine-id $m = mi(l) \in \{1, 2, ..., C_k\}$

There is a post-process after solving above minimization micro-model, which is to update the machine release time for next group of tasks, as shown in (14).

$$t^r_{mi(l)} = t^e_l, \forall l \text{ executable \& schedulable} \tag{14}$$

B. Three scenarios and computational complexity

According to the relative value of C_k and $L_k(R)$, there are totally 3 scenarios for consideration:

IMM1:$L_k(R) \ge C_k = 1$, this is actually a one-machine problem. This can be solved optimally by enumeration, which is shown in **Proposition-1**.

IMM2:$C_k \ge L_k(R) \ge 1$, the total number of machines is not less than the total number of tasks. The optimal solution for this case is following in **Proposition-2**.

IMM3:$L_k(R) > C_k > 1$, the total number of machines is less than the total number of tasks.

Proposition-1: For **IMM1**, the single machine is m_1 and the task list is $l \in \{1, 2, ..., L_k(R)\}$. The optimal objective value is following:

$$\min\left(\min_{l:t^a_l \le t^r_{m_1}} \{p_l + t^r_{m_1}\}, \min_{l:t^a_l > t^r_{m_1}} \{p_l + t^a_l\}\right)$$

Proof: Done by enumeration. ■

Proposition-2: For **IMM2**, the optimal solution is to assign earliest available $L_k(R)$ machines for tasks by first come first serve rule.

Proof: We begin with 2 tasks l_1, l_2 on 2 parallel machines m_1, m_2. The processing time and arrival time of the 2 tasks are $\{p_{l_1}, t^a_{l_1}\}$ and $\{p_{l_2}, t^a_{l_2}\}$, while the machine release time of the 2 machines are $t^r_{m_1}, t^r_{m_2}$. Without loss of generality, we assume that $t^a_{l_1} \leq t^a_{l_2}$ and $t^r_{m_1} \leq t^r_{m_2}$. The optimal objective value is $p_{l_1} + \max\{t^a_{l_1}, t^r_{m_1}\} + p_{l_2} + \max\{t^a_{l_2}, t^r_{m_2}\}$ because it can be verified by all the possible cases as following:

$$\begin{array}{l} t^a_{l_1} \leq t^a_{l_2} \leq t^r_{m_1} \leq t^r_{m_2} \\ t^a_{l_1} \leq t^r_{m_1} \leq t^a_{l_2} \leq t^r_{m_2} \\ \vdots \\ t^r_{m_1} \leq t^a_{l_1} \leq t^r_{m_2} \leq t^a_{l_2} \\ t^r_{m_1} \leq t^r_{m_2} \leq t^a_{l_1} \leq t^a_{l_2} \end{array}$$

The first come first serve rule will result as following mapping $mi(l_1) = m_1, mi(l_2) = m_2$, so it is true for 2-task-2-machine case.

For general cases, $C_k \geq 2, L_k(R) \geq 2$, for any mapping solution $mi(l)$, if $\exists\{l_1, l_2, m_1, m_2\}$, s.t. $mi(l_1) = m_1, mi(l_2) = m_2$, and $(t^a_{l_1} - t^a_{l_2})(t^r_{m_1} - t^r_{m_2}) < 0$. By swapping the mapping to first come first serve order will always achieve a better solution. ■

For **IMM3**, one solution strategy is to schedule one machine id after another and repeatedly apply **Proposition-1** to be solved. This is not an optimal solution, but a fast feasible solution. The optimality depends on machine id sequencing.

Computational complexity for **Proposition-1** is $O(L_k(R))$, and for **Proposition-2** is $O\left(L_k(R) \cdot ln(L_k(R))\right)$, while the fast feasible solution of **IMM3** is $O(L_k(R) \cdot C_k)$

V. Report for the Experiments

A. Benchmark on classical problems {MT6, MT10}– a multi-machine version

There are 6 pairs of experiments to test our proposed IMM-heuristic method with CPLEX solution. The job-process-machine setting is exactly same as the MT6 and MT10 problems. First 3 pairs are the MT6 with {1, 2, 3}–machine(s) for each type. Last 3 pairs are the MT10 with {1, 2, 3}-machine(s) for each type. The results are shown in Table V. There are following points for notice:

- CPLEX solver engine is stopped after one hour and return the best feasible solution;
- both CPLEX and IMM-heuristics solver runs on Intel(R) Xeon(R) CPU of X3220 2.4 GHz with 2.4GHz & 4.0 GB RAM;
- IMM-heuristic method runs about 3 seconds for MT10 problems or 1 second for MT6 problems.

From the table, we can see that the optimality of IMM-heuristic method becomes better when total number of machines are increased. Except for MT10 one-machine problem, IMM-heuristic method achieves better solution than CPLEX in other problems.

We observe that for one-machine MT10 problem, CPLEX [1] failed to have a feasible solution within one hour. It is because the 0-1 formulation results as a very large size problem [8].

TABLE II
Sample Job List in a Semiconductor Manufacturing Problem

Product	Total Number of Processes Per Job	Daily Production Throughput (lots)
1	14	12
2	16	161
3	18	117
4	16	119
Subtotal		409

TABLE III
Solution Comparison for above Sample Job List

Methods	IMM	Rules in ProModel		
		FIFO	LIFO	SPT
Makespan	1462	1575	1601	1592

B. Comparison with existing job dispatch rules for a semiconductor manufacturing scheduling problem

The model of multi-machine JSP with an infinite wait buffer is just suitable for schedule application in back-end semiconductor manufacturing which has following features:

- each job represents particular part (or one lot of chips), whose volume is not so large and has standard size for easy storing;
- the temperature of the parts does not need strict control.

A complex manufacturing floor shop in a semiconductor industry is used as a further test bed in this study. Using ProModel, which is an industrial manufacturing modeling software, a simulation model of the floor shop is created. The specifications of the shop floor are summarized in Fig. 3 and Table III. There are 30 types of machines corresponding to 30 different processes and there are 4 product types, each with its own process flow and planed daily production throughput. Such problem is too hard to be solved by the 0-1 IP model as in section III, while it can be solved by our IMM-heuristic method by around 1 minute for all 409 jobs. IMM-heuristic method achieves slightly better performance than the existing dispatching rules (FIFO, SPT and etc.) used in ProModel.

The problem detailed instance and the solution are given from *ftp://imss2:@robotics.nus.edu.sg/Release/data/imss-dram/CISRAM2008/.*

VI. Conclusion and Further Research Suggestions

We empirically evaluated the impact of multi-machine consideration on solution of several classical **JSPs** with an infinite wait buffer. Statistics show that the objective of sum weighted tardiness is more suitable for multi-machine JSP. Furthermore, an IMM heuristic method has been proposed for a fast feasible solution and the performance is bench-marked with CPLEX solutions. Especially, our IMM heuristic method has following features:

- It supports fractional processing time;
- The micro-model considers the machine release time and the task arrival time, which can be extended as an input from the problem of multi-machine JSP;
- It outputs the machine dispatching solution and the machine release time, so the micro-model can be easily applied to construct a huge multi-machine problems;

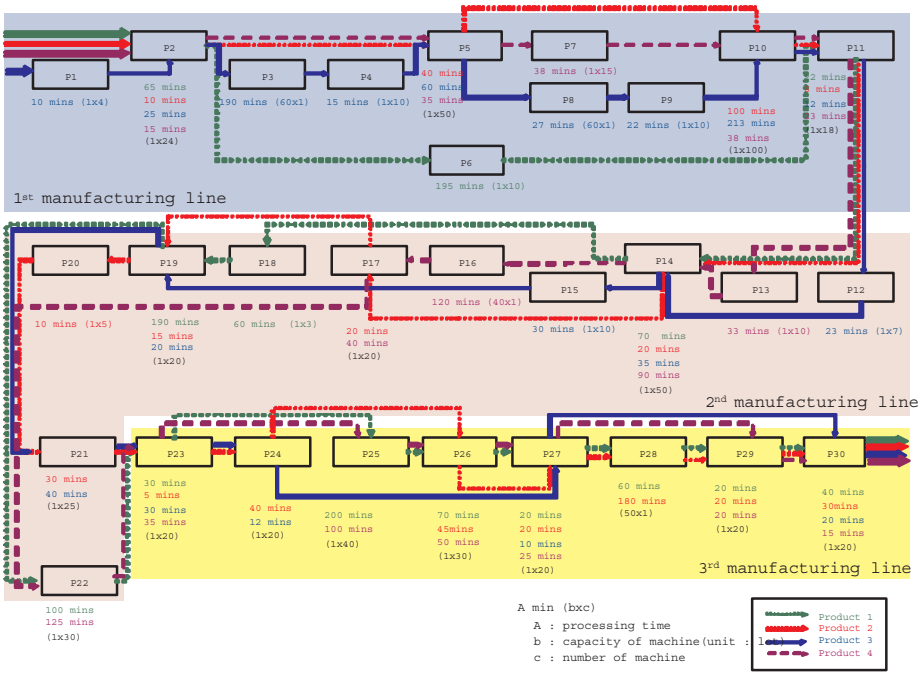


Fig. 3. A Semiconductor Manufacturing Problem: Production Line with 4 Types of Products and 30 Types of Machines

- The overall computational complexity of IMM-heuristic method is not more than $O((N \cdot K \cdot \max_{i=1,2,...,N}\{o_i\})^2)$
- It achieves local optimality for micro-model in 2 scenarios, which are **IMM1** and **IMM2**.
- The optimality of the overall solution is affected by the sequencing in selection of machine type k at iteration R for solving the *Minimization Micro-model* and the sequencing of solving **IMM3** during a specific iteration. This, to our opinion, deserves further research.

In the future, multi-period consideration should come into picture for better machine utilization and dynamics handler. What's more, a fast solution towards optimality of **IMM3** also deserves further study.

Intuitively, the classical model of traveling-salesman is well and deep studied. Afterward, the problems of traveling group with multi agents (or robots), and the case when some parts of the group are ill (or not in best performance) will be further studied.

TABLE IV
COMPARISON OF 1-MACHINE V.S. 2-MACHINE AND CRI-1 (MAKESPAN) V.S. CRI-2 (SUM TARDI.) ON MT6 & MT10 PROBLEMS

	Criterion	$\overline{RTAM}$ time slot	T^M time slot	TW count	WT^M / $\overline{WT}$ time slot	$\overline{AMU}$ per cent	$\overline{CTAJ}$ time slot
MT6-Cap1	1	52.0	61	15	12 / 5.7	72.0%	53.7
	2	53.8	64	8	4 / 2.0	67.9%	44.2
MT6-Cap2	1	43.3	51	4	5 / 2.5	48.7%	47.5
	2	31.0	47	1	3 / 3.0	53.3%	35.2
MT10-Cap2	1	558.2	696	48	113 / 15.3	61.8%	661.3
	2	543.8	764	25	71 / 11.6	52.7%	578.9
MT10-Cap3	1	521.2	691	12	40 / 13.6	44.2%	657.8
	2	443.8	701	5	35 / 10.2	40.0%	532.8

TABLE V
PERFORMANCE BENCH MARK (IMM-HEURISTICS V.S. CPLEX)

Problem	Method	$\overline{RTAM}$ time slot	T^M time slot	TW count	WT^M / $\overline{WT}$ time slot	$\overline{AMU}$ per cent	$\overline{CTAJ}$ time slot
MT6-Cap1	IMM	45.8	63	10	20 / 7.5	79.1%	38.8
	CPLX	53.8	64	8	4 / 2.0	67.9%	44.2
MT6-Cap2	IMM	32.3	48	6	5 / 3.0	51.1%	36.8
	CPLX	31.0	47	1	3 / 3.0	53.3%	35.2
MT6-Cap3	IMM	26.0	47	0	0 / 0.0	36.6%	32.8
	CPLX	28.1	47	0	0 / 0.0	36.6%	32.8
MT10-Cap1	IMM	953.6	1124	29	160 / 39.6	62.8%	784.3
	CPLX	-	930 [1]	-	- / -	-	-
MT10-Cap2	IMM	587.3	754	30	131 / 41.7	54.7%	658.8
	CPLX	543.8	764	25	71 / 11.6	52.7%	578.9
MT10-Cap3	IMM	474.8	690	14	75 / 29.6	40.4%	561.5
	CPLX	443.8	701	5	35 / 10.2	40.0%	532.8

VII. ACKNOWLEDGEMENT

Much appreciation goes to Ireneus Wior for his excellent simulation work which makes ProModel a really efficient tool.

REFERENCES

[1] J.Adams, E.Balas, and D.Zawack. "The shifting bottleneck procedure for job shop scheduling,"*Management Science*, 34, pp.391-401, 1988.
[2] S.D. Wu, E.S. Byeon, R. H. Storer, "A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness," *Operations Research*, vol. 47, no.1, pp.113-124, 1999.
[3] R.L. Graham, E.L. Lawler, J.K.Lenstra and A.H.G.R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annuals of Discrete Mathematics*, pp.287-326, 1979.
[4] E. Kutanoglu, S.D. Wu, "Improving scheduling robustness via preprocessing and dynamic adaptation," in *IIE Transactions*, vol.36, pp.1107-1124, 2004.
[5] J.F. Muth and G.L. Thompson (eds.) *Industrial scheduling*, Prentice-Hall, Englewood Cliffs, NJ. 1963.
[6] M. Drozdowski, "Scheduling multiprocessor tasks - an overview," *European Journal of Operational Research*, vol. 94, pp. 215-230, 1996.
[7] A. Pritsker, L. Watters, and P. Wolfe, "Multiproject scheduling with limited resources: a zero-one programming approach," *Management Science: Theory*, vol.16, no.1, pp.93-108, 1969.
[8] Z.J. Zhao, T.Y.Leong, S.S.Ge, and H.C.Lau, "Bidirectional flow shop scheduling with multi-machine capacity and critical operation sequencing," *Proceedings IEEE-ISIC, International Symposium on Intelligent Control*, 2007, pp. 446 - 451.