Computers & Industrial Engineering 60 (2011) 426–432

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Tabu search heuristic for two-machine flowshop with batch processing machines

Li-Man Liao *, Ching-Jen Huang

Department of Industrial Engineering and Management, National Chin-Yi University of Technology, Taichung 411, Taiwan

ARTICLE INFO

Article history:
Available online 20 March 2010

Keywords:
Batch processing machines
MILP model
Flowshop
Tabu search

ABSTRACT

Batch processing machines are frequently encountered in many industrial environments. A batch processing machine is one which can process several jobs simultaneously as a batch. The processing time of a batch is equal to the largest processing time of any job in the batch. This study deals with the problem of scheduling jobs in a flowshop with two batch processing machines such that the makespan is minimized. A heuristic based on Tabu search (TS) technique is proposed. The proposed heuristic is compared with a heuristic based on mixed integer linear programming (MILP). Because the complexity of the MILP-based heuristic is depended on the number of job batches, the comparison is under up-to-eight batches problem. In order to measure the proposed TS-based heuristic in larger batch problem, the relative error percentage with the lower bound (REP_{LB}) is used. The results show that the proposed heuristic is efficient and effective for the problems with relative large job sizes.

1. Introduction

Batch processing machines (BPMs) are encountered in many different environments such as chemical processes performed in tanks or kilns and burn-in operations in semiconductor industrials. The applications of batch processing machines could be found in Uzsoy (1994), Damodaran, Srihari, and Lam (2007), Damodaran and Srihari (2004), and Manjeshwar, Damodaran, and Srihari (2009). Uzsoy described an application for burn-in operations in semiconductor manufacturing. Damodaran et al. (2007) and Manjeshwar et al. (2009) provided applications of batch processing machines used in the chemically treat stage in a rim (for bike) manufacturing facilities and in chambers for the environmental stress screening (ESS) in the printed circuit board assembly environment. This paper deals with the problem of scheduling batch processing machines in a two-machine flowshop which is the same as Damodaran and Srihari (2004) and Manjeshwar et al. (2009).

ESS has evolved from burn-in techniques and requires the entire product to be tested at specified conditions. The chambers employed in an ESS are batch processing time with restricted capacities. The capacities are defined by number of job that can be held. Jobs loaded with different types may be processed in the same chamber. However, once processing is started, no job can be added to or removed from the chamber until the processing of the batch is complete. The batch processing time is set by the longest processing time among those of all the jobs contained in the batch.

This paper considers a scheduling problem for a flowshop with two batch processing machines (two-BPM) in which a finite number of varied size of jobs are handled. The problem is defined as follows. There are n jobs to be grouped into batches. The batches of jobs are processed on the first machine, they move together to the second machine with negligible transportation time. Each job has a processing time and a size on each machine. All jobs in a batch begin processing at the same time, and the processing time of a batch is determined by the longest processing time of all the jobs in the batch. Each machine can process a batch of jobs simultaneously as long as the total size of the batch does not exceed the machine capacity.

Makespan or maximal completion time is equivalent to the completion time of the last job leaving the last work center of production system. The smaller makespan implies a higher utilization. The utilization for flowshop is closely related to productivity of the production system. Therefore, the makespan performance measure is selected as the objective.

The rest of this paper is organized into five sections. Sections 2 and 3 reviews related papers and describes the proposed two-BPM flowshop problems with MILP model, respectively. The proposed heuristic based on Tabu search, TSH is presented in Section 4. Extensive computational experiments are conduct in Section 5 to evaluate the performance of the heuristic. Finally, concluding remarks are made in Section 6.

2. Related works

Problems related to scheduling batch processing machines, in which machine has the ability of processing a number of jobs

* Corresponding author. Tel.: +886 4 24363048; fax: +886 4 24363039.
E-mail address: Liao507@cjhuang.idv.tw (L.-M. Liao).

0360-8352/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2010.03.004

simultaneously has received much attention in scheduling literature in recent years. The main classification division is research done for batch processing machines with incompatible job families vs. those with compatible job families (Perez, Fowler, & Carlyle, 2005). The first model concerns the case of incompatible product families. According to this model, only products belonging to the same family may be processed simultaneously. Kempf, Uzsoy, and Wang (1998), Uzsoy (1995) and Dobson and Nambimadon (2001) developed deterministic algorithms to schedule batch processing machines with incompatible job families. The second model concerns the batch processing of compatible product families. This model is assumed that products belonging to alternative families may be processed simultaneously. For example, Lee, Uzsoy, and Martin-Vega (1992) model burn-in oven as batch processing machine. The role of batch machines is used as a bottleneck for resources, and can process a number of jobs simultaneous. The different jobs can be batched together but the processing time of the batch is given by the longest processing time among all jobs in the batch (Lee et al., 1992; Uzsoy, 1994; Lee & Uzsoy, 1999; Damodaran & Srihari, 2004; Damodaran et al., 2007, and Manjeshwar et al., 2009). This paper considers the second model.

In a batching machine with compatible job families, Uzsoy (1995) firstly proposed heuristics to minimize the makespan and total flow time. Brucker et al. (1998) proposed dynamic programming algorithms to optimize several different criteria both for unrestricted batch sizes, and for batches that can contain at most n jobs. The dynamic programming algorithms are further extended to identical parallel batching machines for unrestricted batch sizes (Van Der Zee, 2007). Dupont and Dhaenens-Flipo (2002) presented some dominance properties for a general enumeration scheme and for the makespan criterion, and provided a branch and bound method. Damodaran et al. (2007) and Kashan, Karimi, and Jenabi (2008) proposed a simulated annealing (SA) heuristic and a hybrid genetic heuristic to minimize makespan, respectively.

From Perez et al. (2005), we know that the scheduling research for batch processing machines almost focus on single and parallel batch machines. Due to the problem of single or parallel batching processing machines is complicate, some efficient heuristics based on hybrid genetic algorithm are proposed. In flowshop batch processing machines, there is little reported research deal with it. Sung and Kim (2003) considers a scheduling problem in a two batch processing machines in a flowshop (two-BPM flowshop) and three efficient polynomial time algorithms are proposed. The paper assumes that the processing time of a batch depends on the individual machine, but not on the jobs in the batch. That is, all the jobs sizes are one. Damodaran and Srihari (2004) proposed two mixed integer linear programming (MILP) models which are for two-BPM flowshop when the buffer capacity is unlimited or zero. Liao and Liao (2008) considered the same problem as Damodaran and Srihari and proposed an improved MILP models for the problem. An effective heuristic based on the modified MILP model was developed to solve near-optimal solutions in less computation time. From Liao and Liao (2008), it reveals that the MILP-based heuristic is still an integer program, where the computation time grows exponentially with problem size. The heuristic only solved problem size about 15 jobs or 8 batches. Afterwards, Liao and Huang. (2008) applied local search (LS) method to develop a polynomial time heuristic to solve the larger problems, and also obtained good solutions. Therefore, this paper refers to Liao and Huang (2008) LS method to construct a heuristic based on Tabu search, named TSH heuristic.

3. Problem statement

This paper considers two-BPM flowshop problem with the following assumptions.

(1) Given n jobs to be processed in a two-machine flowshop. Each job j is available at time 0 and the processing time of job j on machine m denotes by p_{jm}. Because the processing times of jobs are known a priori from production system, all data are assumed to be deterministic.
(2) Capacities of all machines are Q and job j has a size s_j. The size of a job and the sum of the sizes of jobs in batch b cannot exceed Q. That is $\sum_{j\in b} s_j \leqslant Q$, $\forall b$.
(3) Once the processing of a batch is initiated, it cannot be interrupted and other jobs cannot be introduced into the machine until the processing is completed. The processing time of batch b is given by the largest processing time of the jobs in the batch. That is $P_{bm} = \max_{j\in b} p_{jm}$.
(4) The objective is to minimize the makespan, that is the maximum completion time of jobs, $C_{\max}$.

The related sets and variables of problem addressed by Liao and Liao (2008) are as follows:

Sets
J: jobs
M: machines
B: batches
K: positions
Decision variable
$X_{jb} = 1$, if job j is in batch b; 0 otherwise
Dependent variables
$C_{\max}$: makespan
P_{bm}: processing time of batch$b$ on machine m
C_{km}: completion time of the kth batch on machine m

Liao and Liao (2008) proposed the MILP model for the unlimited intermediate storage. The complexity of MILP model is based on batch number, $|B|$. The computational time increases rapidly as $|B|$ increases. The model is described as follows:

$$\text{Minimize} \quad C_{\max} \tag{1}$$

$$\text{Subject to} \quad \sum_{b\in B} X_{jb} = 1 \ \forall j \tag{2}$$

$$\sum_{j\in J} s_j X_{jb} \leq Q, \ \forall b \tag{3}$$

$$P_{bm} \geq p_{jm} X_{jb} \ \forall j \in J, \ b \in B, \ m \in M \tag{4}$$

$$C_{b1} = \sum_{k=1}^{b} P_{k1} \ \forall b \in B \tag{5}$$

$$C_{12} = C_{11} + P_{12} \tag{6}$$

$$C_{b2} = C_{b-1,2} + P_{b2} \ \forall b \in B/\{1\} \tag{7}$$

$$C_{b2} - C_{b1} \geq P_{b2} \ \forall b \in B \tag{8}$$

$$C_{\max} \geq C_{n2} \tag{9}$$

$$X_{jb} = 0 \text{ or } 1 \tag{10}$$

$$C_{bm}, P_{bm}, C_{\max} \geq 0 \tag{11}$$

4. Proposed Tabu search heuristic

This paper adopts Tabu search (TS) technique to develop a heuristic named as TSH for the two-BPM flowshop problem. The objective is to minimize makespan. Fig. 1 illustrates the computational procedure of TSH.

4.1. Initial solution

Su (2003) proposed an algorithm which minimizes the number of bathes formed for minimizing the makespan. For the two-BPM machines flowshop problem, minimizing the number of batches

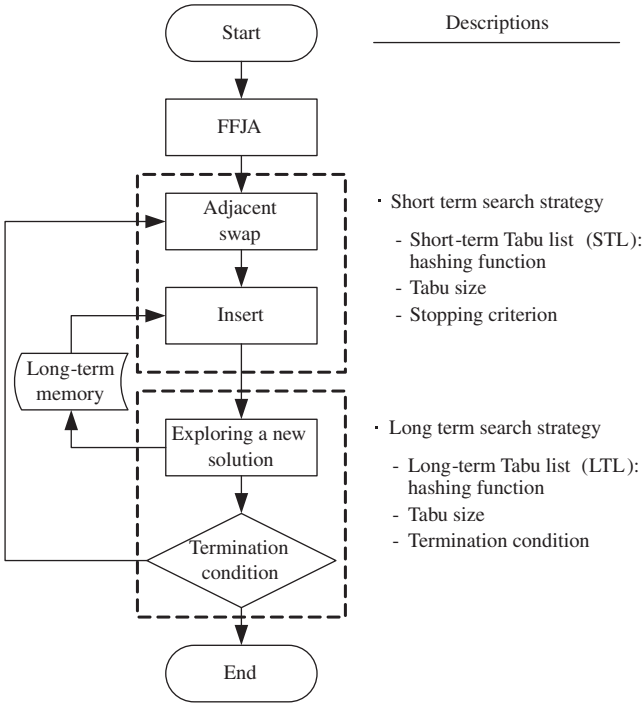

Fig. 1. The computational procedure of TSH.

need not minimize the makespan (Manjeshwar et al., 2009). Therefore, don't enforce batch number minimized on the initial solution. In Liao and Liao (2008) study, it has observed that the optimal number of batch is always near the number of batch minimized, but equation is not a certainty.

Johnson's algorithm (JA) provided an optimal schedule for minimizing makespan in a two-machine flowshop that do not require sequence changes between machines (Pinedo, 2002). Therefore, this paper adopts the procedure of First Fit (FF) (Uzsoy, 1994) and JA to construct the initial solution, called FFJA algorithm. The procedure of FFJA is described as follows:

Step 1. All jobs are arranged by JA, and obtain an initial sequence $\Phi_0 = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, where σ_i is the ith job of Ψ_0. The batches are formed only when each batch accommodates exactly one job. Determine the makespan, $C_{\max}$.

Step 2. Batches formed procedure

2.1 Let Ψ_B denote a set of job batches, and Φ denote a set of jobs unassigned. The Ψ_B set only includes the first job of Ψ_0 i.e. $\Psi_B = \{\sigma_1\}$, and now $\Phi = \{\sigma_2, \ldots, \sigma_n\}$. The current batch is 1, $b = 1$, and job σ_1 is included in the batch.

2.2 If $\Phi \neq \phi$, then try to assign a job of Φ to batch b of Ψ_B; else go to step 3.

2.3 While forming the batches the machine capacity should not be exceeded. Sequentially select a job of Φ, named job j, which satisfy $s_j \leqslant Q - \sum_{i \in b} s_i$. If none job satisfies the conditions, then creates a new batch, $b = b + 1$, and assign the first job of Φ to batch b of Ψ_B and go to step 2.2; else try to combine it to current batch b. Once the batches are formed, each batch is treated as an artificial job and by applying JA to obtain a new batch sequence of $\Psi_B \cup \Phi$.

2.4 Determine the makespan $C'_{\max}$ of the new batch sequence. If $C'_{\max} < C_{\max}$, then assign job j from Φ to batch b of Ψ_B and go to step 2.2.

Step 3 Applying JA to obtain the best batch sequence Ψ^*_B, and determine the makespan $C^*_{\max}$.

4.2. Short-term search strategy

Most flowshop heuristics based on neighborhood search techniques need excessive computing effort when dealing with large-size problems. Therefore, this paper proposes a set of alternative Tabu search procedures by combining two different neighborhood structures, search strategies, and stopping conditions.

4.2.1. Neighborhood structures and selection

Two neighborhood structures are considered, namely adjacent swap (or adjacent interchange) and insert neighborhoods. An adjacent swap neighborhood is obtained by exchanging two jobs which respectively belong to two adjacent batches. An insert neighborhood is obtained by inserting a job to another batch but the batch size cannot exceed machine capacity. The most sizes of the adjacent swap neighborhood and insert neighborhood for an arbitrary sequence with B batches wherein they have n jobs are $n^2(1/|B| - 1/|B|^2)$ and $n(n-1)/2$, respectively. The entire neighborhoods are examined and the best move but not Tabu is taken. This paper adopts adjacent swap neighborhood structure.

4.2.2. Tabu list and size

Hashing function is used to create Tabu list. The type of hashing can be achieved by a function of the form (Carlton & Barnes, 1996; Woodruff & Zemel, 1993):

$$h = \left[\sum_{i=1}^{n} z(x_i) \times z(x_{i+1})\right] \bmod [\text{MAXINT} + 1] \tag{12}$$

where $z(x_i)$ is an integer drawn from the range [1, 32,767], MAXINT = 65,535, and $x_{n+1} = x_1$.

This paper considers batch processing machines, in which machine has the ability of processing a number of jobs simultaneously. Therefore, the jobs within a batch are processed simultaneously and the optimal batch sequence is unconcerned with the jobs' positions within a batch. In order to ensure a batch sequence with only one corresponding hashing value, the hashing function is modified and described as follows:

$$h = \sum_{b=1}^{B} g(b) \tag{13}$$

where

$$g(b) = \left[\sum_{x_i, x_{i+1} \in b} z(x_i) \times z(x_{i+1})\right] \bmod [\text{MAXINT} + 1] \tag{14}$$

$z(x_i)$ is an integer drawn from the range [1, 32,767], MAXINT = 65,535, and the values of the last job and first job in a batch are equivalent.

In a Tabu search heuristic, the size of Tabu list is an important parameter.

4.2.3. Stopping criterion for short-term search

The iterations of adjacent swap (I_S) and insert (I_i) are the stopping criteria.

4.2.4. Aspiration criterion

This paper uses the simplest form of aspiration criterion. A Tabu move is accepted if it produces a solution better than the current best one.

4.3. Long-term search strategy

An intensification scheme often takes the form of reinforcing attributes of good solutions, while a diversification scheme typi-

cally jumps to a new region not yet explored. This paper adopts insert to exploring a new solution. The parameter of termination condition is repeated iteration (Li).

The above procedure of TSH can be divided into three stages. The first stage is to generate an initial batch sequence by applying FFJA method. If the processing times of the jobs in a batch are significant difference, the neighborhood search method should be applied to decrease the difference. For this reason, in the second stage, this paper proposes adjacent swap and insert techniques to combine jobs with closer processing times on each machine to form a batch for making the makespan reduction. That is, the methods reassign jobs to another batch but not Tabu and find a better batch sequence. Finally, in the third stage, this paper tries to explore a new solution region to avoid trapping into local optimum. If the termination condition is not satisfied, then go back to the second stage for another search interaction, otherwise termination of TSH heuristic.

For explaining the advantage of FFJA, a numerical example in Manjeshwar et al. (2009) is used. The example is described as follows:

Example 1. The processing times and sizes of jobs are given in Table 1. The machine capacities are 10.

Although the least batch of example 1 is 2, it is not the optimal batch number. By applying FFJA, an initial solution $\Psi_B^* = \{2,3|1,5|4\}$ is obtained, where '|' denotes a division between two adjacent batches. Obviously, there are three batches in the sequence and it means that FFJA does not enforce on batch number minimized. The sequence $\{2,3|1,5|4\}$ is the optimal sequence and the optimal makespan is 35, $C_{\max}^* = 35$.

For explaining the procedure of TSH, another example in Damodaran and Srihari (2002) is demonstrated and described as follows:

Example 2. The processing times and sizes of jobs are given in Table 2. The machine capacities are assumed to be 10. Applying the above procedures yield the following result.

4.4. Initial solution

Step 1 All jobs are arranged by JA, and obtain initial sequence $\Phi_0 = \{2, 7, 3, 10, 5, 1, 8, 9, 6, 4\}$, and its makespan $C_{\max} = 79$. (See Table 3)
Step 2.1 $b = 1$, $\Psi_B = \{2\}$, $\Phi = \{7, 3, 10, 5, 1, 8, 9, 6, 4\}$.
Step 2.2 $\Phi \neq \phi$, then try to assign a job of Φ.
Step 2.3 $Q - s_2 = 10 - 2 = 8 > 0$ and $i = 2 < 10$. $s_7 = 5 \leqslant 10-2$.
Step 2.4 $\Psi_B = \{2, 7\}$, $\Phi = \{3, 10, 5, 1, 8, 9, 6, 4\}$. Use JA to schedule Ψ_B and Φ, and obtain $C'_{\max} = 77$. Because $C'_{\max} < C_{\max}$, update $C_{\max} = 77$ and go to Step 2.2. (See Table 4a).
Step 2.2 $\Phi \neq \phi$, then try to assign a job of Φ.
Step 2.3 $Q - (s_2 + s_7) = 10 - (2 + 5) = 3 > 0$, and $i = 3 < 10$. $s_3 = 3 \leqslant 3$.
Step 2.4 $\Psi_B = \{2, 7, 3\}$, $\Phi = \{10, 5, 1, 8, 9, 6, 4\}$. Use JA to schedule Ψ_B and Φ, and obtain $C'_{\max} = 74$. Because $C'_{\max} < C_{\max}$, update $C_{\max} = 74$ and go to Step 2.2. (See Table 4b).
Step 2.2 $\Phi \neq \phi$, then try to assign a job of Φ.
Step 2.3 Because none job satisfies the machine capacity conditions (that is, batch 1 is done), then creates a new batch, $b = b + 1 = 2$. $\Psi_B = \{2, 7, 3|10\}$and $\Phi = \{5, 1, 8, 9, 6, 4\}$, and go to step 2.2 (a new batch is starting).

Repeat the Step 2.2–2.5, and finally, all jobs is combined as four batches.

Step 3 Applying JA to obtain the best batch sequence of $\Psi_B, \Psi'_B = \{2,3,7|5,10|1,8,9|4,6\}$ and the makespan, $C_{\max}^* = 46$. It is shown in Table 5.

4.5. Short-term search strategy

Swap two jobs wherein two adjacent batches are. The best neighborhood is the swap of jobs 1 and 10, and $C'_{\max} = 45$. Because $C'_{\max} < C_{\max}^*$, swap the two jobs, and update $\Psi_B^* = \{2,3,7|1,5|8,9,10|4,6\}, C_{\max}^* = 45$. Repeat the short-term search strategy, until stopping criterion is satisfied. The result is shown in Table 6 and Fig. 2.

4.6. Long-term search strategy

A new solution is obtained by insert but not belong to long-term Tabu list and go back to execute short-term strategy. Repeat Li times (terminal condition), the search procedure is termination. The best batch sequence is $\Psi_B^* = \{2,3,7|1,5|8,9,10|4,6\}$ and the $C_{\max}^* = 45$. This solution is optimal in this example (See Fig. 2).

5. Computational experiments

In this section, we evaluate the proposed heuristic by using the same problem generating scheme as Uzsoy (1994) and Liao and Liao (2008). Job processing times were randomly generated from a discrete uniform distribution $U(1,100)$. The capacities of both machines were assumed to be 10. Job sizes were generated from discrete uniform distributions between $a_{\min}$ and $a_{\max}$. Three different

Table 1
Data for Example 1.

Job j	1	2	3	4	5
p_{j1}	10	2	6	15	7
p_{j2}	14	9	10	1	12
s_j	5	2	3	4	5

Table 2
Data for Example 2.

Job j	1	2	3	4	5	6	7	8	9	10
p_{j1}	10	2	6	15	7	9	3	10	10	6
p_{j2}	14	9	10	1	12	4	5	8	5	9
s_j	5	2	3	4	5	3	5	1	4	4

Table 3
Result of JA for Example 2.

Job j	2	7	3	10	5	1	8	9	6	4
p_{j1}	2	3	6	6	7	10	10	10	9	15
p_{j2}	9	5	10	9	12	14	8	5	4	1
s_j	2	5	3	4	5	5	1	4	3	4

Table 4
The first batch formed for Example 2.

Job j	2, 7	3	10	5	1	8	9	6	4
(a)									
p_{j1}	3	6	6	7	10	10	10	9	15
p_{j2}	9	10	9	12	14	8	5	4	1
s_j	7	3	4	5	5	1	4	3	4
Job j	2, 3, 7	10	5	1	8	9	6	4	
(b)									
p_{j1}	6	6	7	10	10	10	9	15	
p_{j2}	10	9	12	14	8	5	4	1	
s_j	10	4	5	5	1	4	3	4	

Table 5
The all batches formed for Example 2.

Job j	2, 3, 7	5, 10	1, 8, 9	4, 6
p_{j1}	6	7	10	15
p_{j2}	10	12	14	4
s_j	10	9	10	7

Table 6
The best batch sequence of Example 2.

Job j	2, 3, 7	1, 5	8, 9, 10	4, 6
p_{j1}	6	10	10	15
p_{j2}	10	14	9	4
s_j	10	10	9	7

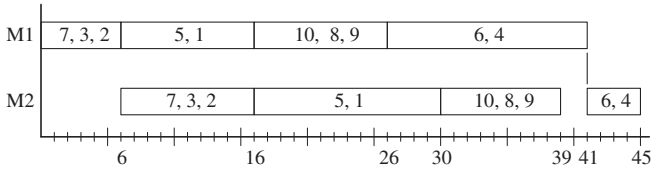

Fig. 2. Gantt chart of the best batch sequence of Example 2.

distributions of job sizes are designed as shown in Table 7. The MILP models, generated by a computer program, were solved by LINGO 10.0. The proposed heuristic and MILP model run on a Pentium IV 3.2 GHz PC.

This paper evaluates the performance of the proposed heuristic by solving the problem with job $n = 10, 20, \ldots, 100$. There are 10 independent instances generated for each combination. The total instance is 300.

In order to calibrate the TSH, the heuristic runs 10 instances of 50-job in distribution I, II, III, respectively. Preliminary experiments are conducted to determine TL = 35 and show that the ranges of Stopping criterions. Therefore, a full factorial design is chosen, in which all combinations of the following factors are tested:

- Stopping criterion of adjacent swap: 5 levels (20, 40, 60, 80 and 100 iterations);
- Stopping criterion of insert: 4 levels (50, 100, 150, and 200)

All the cited factors result in total of 20 different combinations. The set of instances comprises 27 combinations of job number and distributions of job sizes, being $n = \{20, \ldots, 100\}$ and three distributions. There are 10 replicates for each combination thus summing up for 270 instances.

The first experiment is designed for stopping criterion of adjacent swap. The ANOVA shows that the number of job, distributions of job sizes and stopping criterion of adjacent swap are significant. The brief summary of ANOVA table is shown in Table 8. The interaction plot of job size distributions and adjacent swap iteration was shown in Fig. 3. Based on Fig. 3, 80, 60 and 40 are chosen as adjacent swap iterations in job size distributions I, II, III problems, respectively.

The second experiment is designed for stopping criterion of insert. The ANOVA shows that stopping criterion of insert is not significant (P value = 0.57). Therefore, 100 is selected as stopping criterion of insert.

Table 7
Three different distributions of job sizes.

Dist.	(a_{min}, a_{max})	Description
I	(1, 5)	Job sizes are relatively small
II	(4, 10)	Job sizes are relatively large
III	(1, 10)	Job sizes are distributed widely

Table 8
Design of experiment for adjacent swap.

Source	F	P
n	4.41	**0**
Distribution	145.91	**0**
Swap iteration	4.44	**0.004**

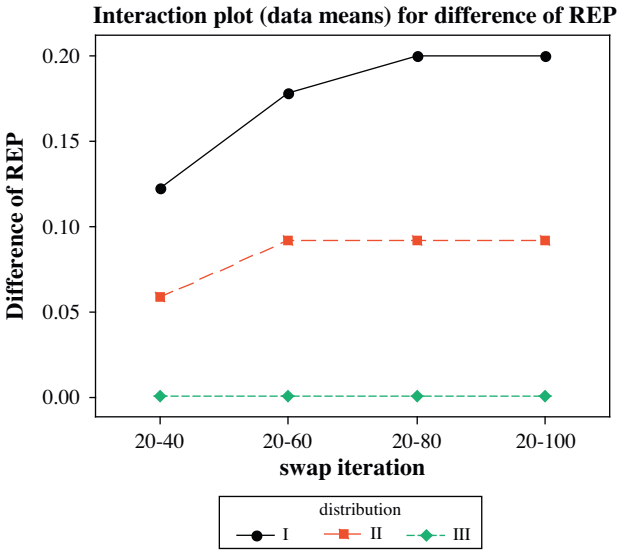

Fig. 3. Interaction plot of Dist. I, II, III and swap iterations.

After the short-term search strategy is done, the next experiment will decide a proper long-term search strategy. The termination condition of TSH has six levels of repeated iterations, i.e., 5, 10, 15, 20, 25, 30. There are 90 instances in each job size distribution problems. The improved instances from iterations k to l are shown in Table 9. As an illustration, in case of Dist. I, there are 81 improved instances in all 90 instances from 1 iteration to 5 iterations. The improve rate is 90% ($81/90 \times 100\%$). Similarly, there are 45 improved instances from 5 iterations to 10 iterations. Consequently, the number of improved instances is in decreasing order of iterations k to l from Table 9. In tradeoff of time and solution quality, the proper choice for distributions, I, II, III are 30, 20, 10 iterations, respectively.

From above experiments, the important parameters of TSH are shown in Table 10.

The further experiments are concerned with the solution quality. Firstly, the problem with $n = 10$ is solved by the MILP model described in Section 3 and the optimal solution is obtained. The solutions of the proposed TSH heuristic compare with the opti-

Table 9
Improved instances of long-term repeated iterations.

From iterations k to l	# of improved instances		
	Dist. I	Dist. II	Dist. III
1–5	81	71	46
5–10	45	22	9
10–15	27	16	5
15–20	24	9	1
20–25	16	6	0
25–30	10	6	0

Table 10
Parameters setting of TSH.

Parameters	Dist. I	Dist. II	Dist. III
Tabu size	35	35	35
Iteration of swap	80	60	40
Iteration of insert	100	100	100
Termination condition	30	20	10

mum. The relative error percentage (REP) is used to evaluate the performance and described as follows. The computational results are shown in Table 11.

$$REP = \frac{C_{\max}^{TSH} - C_{\max}^{*}}{C_{\max}^{*}} \times 100 \quad (12)$$

In $n = 20$ problems, few instances can be solved by MILP model in 24 h. Therefore, the proposed heuristic solves for 20-, 30-, ..., 100-job problems, and then the results compare with the lower bounds (LB) (Liao and Liao, 2008). The LB substitutes the optimal solution, $C_{\max}^{*}$ in formulation (12). The performance index is named REP_{LB} and described as follows. The results are shown in Table 12.

$$REP_{LB} = \frac{C_{\max}^{TSH} - LB}{LB} \times 100 \quad (13)$$

Table 11 presents the performance of the proposed TSH heuristic for $n = 10$ problems. In distributions I and II, the percentages of optimum are both 100%. There are only 3 instances is not optimal in distributions III and the average of REP is 1.00. One relation between LB and $C_{\max}^{*}$ deserves to be mentioned: The average relative error percentages of LBs and $C_{\max}^{*}$ are respectively 13.05, 0.99 and 6.28 for distributions I, II and III. It presents that the LB of distribution II is the closest to the optimum and distribution I is the farthest from optimum.

Table 11
Computational results of instances as $n = 10$.

Inst.	Dist.	LB	MILP model		$\frac{C_{\max}^{*}-LB}{C_{\max}^{*}} \times 100$	Proposed TSH	
			$C_{\max}^{*}$	CPU time (s)		$C_{\max}$	REP
1	I	223	256	81	14.80	256	0
2		193	214	11	10.88	214	0
3		188	213	11	13.30	213	0
4		208	241	12	15.87	241	0
5		216	246	7	13.89	246	0
6		223	251	3	12.56	251	0
7		204	242	7	18.63	242	0
8		243	249	43	2.47	249	0
9		252	269	16	6.75	269	0
10		173	210	5	21.39	210	0
Avg.					13.05		0
11	II	423	423	549	0	423	0
12		469	471	1371	0.43	471	0
13		379	402	1333	6.07	402	0
14		451	467	700	3.55	467	0
15		454	456	1298	0.44	456	0
16		636	636	–	0	636	0
17		482	491	1027	1.87	491	0
18		545	545	440	0	545	0
19		476	476	274	0	476	0
20		468	468	1463	0	468	0
Avg.					0.99		0
21	III	365	393	442	7.67	393	0
22		314	349	844	11.15	349	0
23		279	325	249	16.91	325	2.46
24		467	467	585	0	467	0
25		336	363	514	8.04	363	1.10
26		477	477	170	0	477	0
27		375	375	730	0	375	6.40
28		406	406	280	0	406	0
29		239	283	93	18.41	283	0
30		336	338	177	0.60	338	0
Avg.					6.28		1.00

Table 12
Computational results of $n = 10, 20, \ldots, 100$.

n	Mean of REP_{LB}		
	Dist. I	Dist. II	Dist. III
20	10.00	4.42	4.06
30	13.39	7.04	2.05
40	13.27	6.10	3.24
50	13.40	6.82	4.00
60	12.99	5.50	3.19
70	12.48	6.96	5.15
80	13.04	7.64	6.27
90	13.42	7.64	4.21
100	13.26	8.10	6.30
Avg.	12.80	6.69	4.27

Table 12 presents the performance of the proposed TSH heuristic for $n = 20, 30, \ldots, 100$ problems. The averages of REP_{LB} in distributions II and III only are respectively 6.69 and 4.27. They are significant smaller than the averages of REP_{LB} of distribution I. The main reason is that the LB is closer to the optimum than distributions I.

6. Conclusions

For two-BPM problem, as the number of batch rapidly increases, the computational time of MILP model becomes more exhausted. As job batch is larger than 8, few instances can be solved. In $n = 20$ problems with distribution II, job sizes are relatively large; no instance can be solved by MILP-based heuristic in 12 h. This paper proposes a heuristic based Tabu search for flowshop with two-BPM such that the makespan is minimized. Because the complexity of the MILP-based heuristic is depended on the number of job batches, the comparison of the TSH heuristic with the MILP-based one is under up-to-eight batch problem. As job size is relatively large, the proposed TSH heuristic is significant efficiency.

The TSH heuristic is used to solve large problems and only consumes little time, e.g., a 100-job problem can be solved within 60 s, to obtain a good solution. The performance is evaluated by the REP_{LB}. Although the mean of REP_{LB} of distribution I is 12.28, we can conjecture that the REP of TSH heuristic is very close to optimum based on the average relative error percentages of LB and $C_{\max}^{*}$, $(C_{\max}^{*} - LB)/C_{\max}^{*} \times 100$, of Table 11. The proposed TSH heuristic is more significant efficient and effective than the proposed one by Liao and Huang (2008).

All the experimentation clearly highlights the superior of the proposed TSH heuristic as opposed to Lingo commercial solvers. This research can be extended to optimize other meta-heuristics. The authors are currently applying SA technique for the problem. However, other novel approaches, such as branch-and-bound technique, can also be developed for further research.

Acknowledgment

This research is funded by the National Science Council under Grants NSC 96-2221-E-167-003. And special thanks to all who have helped to make this study.

References

Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M. Y., Potts, C. N., Tautenhahn, T., et al. (1998). Scheduling a batching machine. *Journal of Scheduling, 1*, 31–54.
Carlton, W. B., & Barnes, J. W. (1996). A note on hashing functions and tabu search algorithms. *European Journal Operational Research, 95*, 237–239.

Damodaran, P., & Srihari, K. (2004). Mixed integer formulation to minimize makespan in a flow shop with batch processing machines. *Mathematical and Computer Modeling, 40*, 1465–1472.

Damodaran, P., Srihari, K., & Lam, S. (2007). Scheduling a capacitated batch-processing machine to minimize makespan. *Robotics and Computer – Integrated Manufacturing, 23*(2), 208–216.

Dobson, G., & Nambimadon, R. S. (2001). The batch loading and scheduling problem. *Operations Research, 49*(1), 52–65.

Dupont, L., & Dhaenens-Flipo, C. (2002). Minimizing the makespan on a batch machine with non-identical job sizes: An exact procedure. *Computers and Operations Research, 29*, 807–819.

Kashan, A. H., Karimi, B., & Jenabi, M. (2008). A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Computers and Operations Research, 35*, 1084–1098.

Kempf, K. G., Uzsoy, R., & Wang, C. S. (1998). Scheduling a single batch processing machine with secondary resource constraints. *Journal of Manufacturing Systems, 17*(1), 37–51.

Lee, C. Y., & Uzsoy, R. (1999). Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research, 37*(1), 219–236.

Lee, C. Y., Uzsoy, R., & Martin-Vega, L. A. (1992). Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research, 40*(4), 764–775.

Liao, L. M., & Huang, C. J. (2008). An effective heuristic for two-machine flowshop with batch processing machines. In The 38th conference on computers and industrial engineering (ICCIE2008). Beijing, China.

Liao, C. J., & Liao, L. M. (2008). Improved MILP models for two-machine flowshop with batch processing time. *Mathematical and Computer Modelling, 48*(7-8), 1254–1264.

Manjeshwar, P. K., Damodaran, P., & Srihari, K. (2009). Minimizing makespan in a flow shop with two batch-processing machines using simulated annealing. *Robotics and Computer – Integrated Manufacturing, 25*, 667–679.

Perez, I. C., Fowler, J. W., & Carlyle, W. M. (2005). Minimizing total weighted tardiness on a single batch process machine with incompatible job families. *Computers and Operations Research, 32*, 327–341.

Pinedo, M. (2002). *Scheduling: Theory, algorithms, and systems* (2nd ed.). New Jersey: Prentice-Hall.

Su, L. H. (2003). A hybrid two-stage flowshop with limited waiting time constraints. *Computers and Industrial Engineering, 44*, 409–424.

Sung, C. S., & Kim, Y. H. (2003). Minimizing due date related performance measures on two batch processing machine. *European Journal of Operational Research, 147*, 644–656.

Uzsoy, R. (1994). Scheduling a single batch processing machine with non-identical job sizes. *International Journal of Production Research, 32*(7), 1615–1635.

Uzsoy, R. (1995). Scheduling batch processing machines with incompatible job families. *International Journal of Production Research, 33*(10), 2685–2708.

Van Der Zee, D. J. (2007). Dynamic scheduling of batch-processing machines with non-identical product sizes. *International Journal of Production Research, 45*, 2327–2349.

Woodruff, D. L., & Zemel, E. (1993). Hashing vectors for tabu search. *Anneal Operations Research, 41*, 123–137.