Computers & Industrial Engineering 56 (2009) 306–311

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Heuristic algorithms for two-machine flowshop with availability constraints ☆

Li-Man Liao *, Chin-Hui Tsai

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

ARTICLE INFO

Article history:
Received 15 August 2006
Received in revised form 7 January 2008
Accepted 5 June 2008
Available online 13 June 2008

Keywords:
Scheduling
Two-machine flowshop
Machine availability
Makespan

ABSTRACT

The majority of the scheduling studies carry a common assumption that machines are available all the time. However, machines may not always be available in the scheduling period due to breakdown or preventive maintenance. Taking preventive maintenance activity into consideration, we dealt with the two-machine flowshop scheduling problem with makespan objective. The preventive maintenance policy in this paper was dependent on the number of finished jobs. The integer programming model was proposed. We combined two recent constructive heuristics, HI algorithm and H algorithm, with Johnson's algorithm, and named the combined heuristic H&J algorithm. We also developed a constructive heuristic, HD, with time complexities $O(n^2)$. Based on the difference in job processing times on two machines, both H&J and HD showed good performance, and the latter was slightly better. The HD algorithm was able to obtain the optimality in 98.88% of cases. We also employed the branch and bound (B&B) algorithm to obtain the optimum. With a good upper bound and a modified lower bound, the proposed B&B algorithm performed significantly effectively.

1. Introduction

Most studies on scheduling assume that machines are available throughout the planning horizon. However, in practice, machines are not always available (Pinedo, 2002). That is, machines may not be available during the scheduling horizon due to breakdown (stochastic) or preventive maintenance (deterministic). Taking preventive maintenance activity into consideration, we dealt with a flowshop scheduling problem with limited machine availability.

In capital-intensive industry, production generally proceeds on a continuous basis and the availability of production centers at all time is very important. Nevertheless, maintenance activities have to be performed. Possible events that necessitate maintenance operations include: (1) the occurrence of a failure (failure-based maintenance); (2) the elapse of a certain amount of time or usage (use-based maintenance); and (3) the tested condition of a unit (condition-based maintenance) (Art, Knapp, & Lawrence, 1998). For recent surveys of problems with limited machine availability, refer to Sanlaville and Schmidt (1998) and Schmidt (2000). However, research on these problems has started only recently.

Johnson's rule is well known for the case of continuous machine availability, making the problem of minimizing the makespan easy to solve for two machines. Lee (1997) proved the problem to be NP-hard when an interval of non-availability (or hole, for short) occurs, and then developed a pseudo-polynomial dynamic programming algorithm to optimally solve the problem. Lee presented two heuristic algorithms. The first heuristic had a worst-case error bound of 1/2 for the case in which the hole occurred on the first machine. The second heuristic with a worst-case error bound of 1/3 for the case in which the hole occurred on the second machine. Similarly, Cheng and Wang (2000) studied the problem with the holes occurred on the first machine. Their heuristic had a worst-case error bound of 1/3. Breit (2004), studying the holes occurring on the second machine, proposed a heuristic with a worst-case error bound of 1/4. Cheng and Wang (1999) considered a special case where the availability constraint is imposed on each machine, but the two availability constraints are consecutive.

Lee (1999) considered the two-machine flowshop problem under the assumption that if a job cannot be finished before the next down period of a machine, then the job must be restarted partially when the machine becomes available again. His model was called semi-resumable. The model contained two important special cases: resumable where the job can be continued without any penalty and non-resumable where the job must be totally restarted. Lee also developed a pseudo-polynomial dynamic programming algorithm to optimally solve the problem and proposed heuristic algorithms with an error bound analysis.

Blazewicz, Breit, Formanowicz, Kubiak, and Schmidt (2001) studied a two-machine flowshop problem where machines are unavailable in given time intervals. They analyzed two constructive heuristics, Johnson's algorithm and look-ahead heuristic, and a heuristic based on simulated annealing (SA). Blazewicz et al. concluded that the SA-based heuristic is a more effective approach.

☆ This manuscript was processed by Area Editor John W. Fowler.

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

0360-8352/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2008.06.003

Kubiak, Blazewicz, Formanowicz, Breit, and Schmidt (2002) proved that no polynomial time heuristic with a finite worst-case bound may exist when at least two holes are allowed to occur. Their study also showed that makespan minimization becomes NP-hard in the strong sense even if arbitrary number of holes occur on one machine only. Most important, Kubiak et al. proved two important properties of optimal schedules for the two-machine flowshop, a theory which serves as the framework of the current paper. They further developed a branch and bound algorithm based on the proposed properties.

Some papers stated that machines are available in time windows, which is true in computer systems. Aggoune, Mahdi, and Portman (2001) and Aggoune (2004) considered a flowshop problem with availability constraints, and provided two approaches to dealing with the maintenance activities: either starting time of the maintenance tasks are fixed or the maintenance tasks must be performed on a given time window. Aggoune et al. proposed a heuristic based on genetic algorithm to solve the makespan and the total weighted tardiness minimization problems. Aggoune developed a heuristic based on genetic algorithm and tabu search to solve the makespan minimization problem.

Most studies on machine availability take into consideration the elapse of a certain amount of time or usage (use-based maintenance). However, Dell'Amico and Martello (2001) considered a practical assembly line for printed circuit boards. They asserted that the machine is not available after processing a fixed number of jobs to allow for time precision adjustment of the machines. That is, the time periods of preventive maintenance activities are dependent on the number of finished jobs. Liao, Chen, and Lin (2007) provided an algorithm to solve two parallel machines where there are one or more unavailability intervals for each machine. The algorithm had exponential time complexities, but it could optimally solve the various-sized problems in reasonable computation time.

This paper dealt with the two-machine flowshop scheduling problem with makespan objective. The preventive maintenance policy was dependent on "the number of finished jobs". We combined two recent constructive heuristics, HI algorithm (Cheng & Wang, 1999, 2000) and H algorithm (Breit, 2004), with Johnson's algorithm, and named the heuristic H&J algorithm. We also developed a constructive heuristic, HD, which is based on the difference in the jobs' processing times on two machines. In order to evaluate the performance of H&J and HD, we further developed a branch and bound algorithm with a modified lower bound. Compared with the optimum solution, H&J was able to obtain optimality in 1562 out of 1600 instances (97.63%), and HD was able to obtain optimality in 1582 out of 1600 instances (98.88%). Both H&J and HD showed good performance, and the latter was slightly better.

The rest of the paper is organized as follows. Section 2 defines the terminology and constructs an integer programming model. Section 3 addresses basic properties of optimal solution and the development of two constructive heuristics, H&J and HD algorithms. A branch and bound algorithm (B&B algorithm) with a modified lower bound is constructed in Section 4. The performance of HD algorithm is evaluated in Section 5. The final section draws the conclusions of this work.

2. Terminology and integer programming model

Given n jobs to be processed in a two-machine flowshop, we define the following notations:

J_j	job j, $j = 1,\ldots,n$
$J_{[j]}$	the job at the jth position of schedule
M_i	machine i, $i = 1, 2$
$p_{i,j}$	processing time for J_j on M_i
t_i	length of hole on M_i
x_i	the number of finished jobs, the preventive maintenance policy on M_i
$h_{i,[j]}$	the index of M_i is available after the jth job. $h_{i,[j]} = 1$ if M_i is not available (hole) after the jth job; $h_{i,[j]} = 0$ if M_i is available after the jth job, i.e., $h_{i,[j]} = 1$ if j/x_i is integer; $h_{i,[j]} = 0$, otherwise.
$z_{j,k}$	the index of job j is scheduled at the kth position. $z_{j,k} = 1$ if job j is scheduled at the kth position; $z_{j,k} = 0$, otherwise.
$\sum$	schedule of jobs $1,\ldots,n$.
$J_k^{\sum}$	For a given schedule $\sum$ the holes partition jobs into disjoint subsets, the subset contains jobs completed on M_1 between starting points of the kth and the $(k+1)$th holes.
$C_{i,j}$	the completion time for J_j on M_i.
$C_{i,[l]}$	the completion time of the lth ranked job on M_i.
$C_{1,\max}$	makespan', $C_{1,\max} = C_{1,[n]} = \max\{C_{1,j}, j = 1,\ldots,n\}$.
$C_{\max}$	makespan,$C_{\max} = C_{2,[n]} = \max\{C_{2,j}, j = 1,\ldots,n\}$.
S_a	set of jobs before executing forward insert.
S_b	set of jobs before executing backward insert.
d_j	difference in processing time for J_j on M_1 and M_2, $d_j = p_{1,j}$-$p_{2,j}$.

In order to describe the problem clearly, an integer programming model is presented. The decision variables and auxiliary variables are $z_{j,k}$ and $C_{i,[l]}$, respectively. The parameters are $p_{i,j}$, t_i, x_i and $h_{i,[j]}$. The mixed integer programming model with $n^2 + 2n$ variables, including n^2 binary variables and $2n$ variables, and $5n$ constraints is formulated. The model is formulated as follows.

Objective function:

$$\min C_{2,[n]}$$

Subject to:

$$\sum_{j=1}^{n} z_{j,k} = 1; \quad k = 1,2,\ldots,n \tag{1}$$

$$\sum_{k=1}^{n} z_{j,k} = 1; \quad j = 1,2,\ldots,n \tag{2}$$

$$C_{1,[l]} = \sum_{k=1}^{l}\sum_{j=1}^{n}(z_{j,k} \times p_{1,j}) + \sum_{k=1}^{l}(h_{1,[k-1]} \times t_1); \quad l = 1,2,\ldots,n \tag{3}$$

$$C_{2,[l]} \geqslant C_{1,[l]} + \sum_{j=1}^{n}(z_{j,l} \times p_{2,j}); \quad l = 1,2,\ldots,n \tag{4}$$

$$C_{2,[l]} \geqslant C_{2,[l-1]} + h_{2,[l-1]} \times t_2 + \sum_{j=1}^{n}(z_{j,l} \times p_{2,j}); \quad l = 1,2,\ldots,n \tag{5}$$

$$h_{i,[j]} \in \{0,1\}; \quad i = 1,2; j = 1,2,\ldots,n \tag{6}$$

$$z_{j,k} \in \{0,1\}; \quad j,k = 1,2,\ldots,n \tag{7}$$

Constraint (1) specifies that exactly only one job can be scheduled to position k for any job j. Constraint (2) specifies that job j has to be scheduled to exactly one position. Constraint (3) defines the completion time of the lth ranked job on M_1. Constraints (4) and (5) insure that a job's completion time on M_2 is no earlier than that job's completion time on M_1 plus that job's processing time on M_2 and its previous job's completion time on M_2 plus that job's processing time on M_2.

3. The proposed solution methods

In this section, two critical properties of optimal schedules are described and two heuristics, H&J algorithm and HD algorithm, are proposed.

3.1. Basic properties

The two properties of optimal schedules, Lemmas 1 and 2, which were initially provided by Kubiak et al. (2002), were used

in this study. Lemma 1 was applied to HD algorithm to enhance its performance. Both Lemmas 1 and 2 were applied to B&B algorithm to enable a decrease in branching nodes. The B&B algorithm could accordingly effectively increase performance. The two properties are described as follows.

Lemma 1. *There exists an optimal schedule* $\sum$ *such that jobs in sets* $J_k^{\sum}$ *are in Johnson order.*

Lemma 2. *There exists an optimal schedule* $\sum$ *such that if* $p_{1,j'} \leqslant p_{1,j}$, $p_{2,j'} \geqslant p_{2,j}$, $j' \in J_k^{\sum}$ *and* $j \in J_l^{\sum}$, *then* $k \leqslant l$.

3.2. H&J algorithm

The authors proposed the first heuristic (H&J), which combined two heuristics of HI algorithm (Cheng & Wang, 1999, 2000) and H algorithm (Breit, 2004), with Johnson's algorithm. The proposed algorithm required O($n$log$n$) computation time, and is described as follows.

Step 1. Use Johnson's algorithm to schedule the jobs and let the corresponding schedule be S_1.
Step 2. Sequence the jobs in a non-increasing order of $p_{2,j}/p_{1,j}$, and let the corresponding schedule be S_2.
Step 3. Let J_k and J_l be two jobs with the largest and the second largest processing time on M_2, respectively, i.e., $p_{2,k} \geqslant p_{2,l} \geqslant \max p_{2,j}$ for $j = 1,\ldots,n$, $j \neq k$, and $j \neq l$. The jobs are sequenced in the same sequence as that in Step 2 except that J_k and J_l are scheduled as the first two jobs such that the makespan is minimized. Let the corresponding schedule be S_3.
Step 4. Let J_a and J_b be two jobs with the largest and the second largest processing time on M_1 and M_2, respectively, i.e., $p_{1,a} + p_{2,a} \geqslant p_{1,b} + p_{2,b} \geqslant \max\{p_{1,j} + p_{2,j}\}$ for $j = 1, \cdots, n$, $j \neq a,b$. And let J_c be the job with the largest processing time on M_2, other than job J_a, i.e., $p_{2,c} = \max\{p_{2,j} | J_j \in S \backslash J_a\}$.
Step 5. Construct schedule S_4 by moving J_a to the first position in S_2.
Step 6. Construct schedule S_5 by moving J_a to the last position in S_4.
Step 7. Sequence J_a and J_c as the first two jobs such that the maximum completion time of these jobs is minimized, and sequence other jobs after J_a and J_c in the same order as that in S_2. Let the schedule be S_6.
Step 8. Sequence J_a and J_b as the first two jobs such that the maximum completion time of these jobs is minimized, and sequence other jobs after J_a and J_c in the same order as that in S_2. Let the schedule be S_7.
Step 9. Select the best schedule from S_1, S_2, S_3, S_4, S_5, S_6 and S_7, named it $S_{H\&J}$, and its makespan is $C_{\max}^{H\&J} = \min\{C_{\max}(S_1), C_{\max}(S_2), \cdots, C_{\max}(S_7)\}$.

3.3. HD algorithm

The authors developed the second heuristic and named it HD algorithm, which involved two main phases. The first phase sequenced jobs according to differences in processing times of jobs on two machines, and performed local adjustment for jobs between two consecutive holes by Lemma 1. The second phase involved insert procedures including forward and backward insert, and also performed local adjustment for jobs between two consecutive holes by Lemma 1. The proposed algorithm had polynomial time complexities, O(n^2), and is described as follows.

First phase

Step 1. Compute difference in processing time of every job on M_1 and M_2, that is, $d_j = p_{1,j} - p_{2,j}$. Sequence the jobs in a non-decreasing order of d_j.
Step 2. Perform local adjustment for jobs between two consecutive holes by Lemma 1. Let the schedule be S and the corresponding makespan be $C_{\max}(S)$.

Second phase

Step 3. Let l denote the position index and its initial value be 1. Let $S_a = \{J_1,\ldots,J_n\}$.
Step 4. Select J_f, where $p_{1,f} = \min\{p_{1,j}\}$, $j \in S_a$ and $p_{1,f} \leqslant p_{2,f}$, and forward insert J_f to the lth position. Perform Step 2 to obtain S', and its corresponding makespan $C_{\max}(S')$.
Step 5. If $C_{\max}(S') < C_{\max}(S)$ has met or $C_{\max}(S') = C_{\max}(S)$ has not all met in three successive times, then $S = S'$, $C_{\max}(S) = C_{\max}(S')$, and $S_a = S_a - \{J_f\}$; otherwise, go to Step 6. If $l < n$, then let $l = l + 1$ and back to Step 4.
Step 6. Let l denote the position index and its initial value be n. Let $S_b = \{J_1,\ldots,J_n\}$.
Step 7a. Select J_{b1} and J_{b2}, where $p_{2,b1} = \min\{p_{2,j}\}$, $j \in S_b$ and $p_{2,j} \leqslant p_{1,j}$, and $p_{2,b2} = \min\{p_{2,j}\}$, $j \in S_b - \{J_{b1}\}$ and $p_{2,j} \leqslant p_{1,j}$. Backward insert J_{b1} and J_{b2} to the lth position respectively and obtain two different schedules.
Step 7b. The two schedules obtained above are applied to perform Step 2, and then obtain S'_{b1} and S'_{b2}, respectively. Select the better schedule from S'_{b1} and S'_{b2}, and the makespan of the selected schedule is $C'_{\max} = \min\{C_{\max}(S'_{b1}), C_{\max}(S'_{b2})\}$. If $C'_{\max} = C_{\max}(S'_{b1})$, then $J_b = J_{b1}$, and $S' = S'_{b1}$; otherwise, $J_b = J_{b2}$ and $S' = S'_{b2}$.
Step 8. If $C_{\max}(S') < C_{\max}(S)$ has met or $C_{\max}(S') = C_{\max}(S)$ has not all met in three successive times, then $S = S'$, $C_{\max}(S) = C_{\max}(S')$, and $S_b = S_b - \{J_b\}$; otherwise, go to Step 9. If $l > 1$, then let $l = l - 1$ and back to Step 7a.
Step 9. The best schedule is $S_{HD} = S$, and yields its corresponding makespan, $C_{\max}^{HD} = C_{\max}(S)$.

4. Branch and bound algorithm

In order to evaluate the performance of H&J and HD, the authors employed the branch and bound (B&B) algorithm to obtain the optimum. The key elements of the B&B procedures were the branching rule which breaks up the feasible set of solutions, the lower bounding procedure, and the rule for selecting the next subproblem to be solved. We referred to Sule's procedure of B&B (1997) and developed our B&B algorithm.

4.1. Branching step

The branching rule that we proposed to use is the best bound search. Forward branching method employed when the total workload (including processing times and hole's times) of M_2 is larger than that of M_1; otherwise, the backward branching method is used. The tree structure has its root node at level 0 in which none of the jobs are placed in any position of the sequence. Level 1 is constructed from the root by branching at each job which satisfies Lemmas 1 and 2. For example, when there are n jobs, there must be fewer than n nodes at level 1. Next, at each node of level 1, we create fewer than $(n - 1)$ children at level 2. Expanding likewise, we can generate the tree structure where at level k, we have a partial solution with the first k jobs (forward branching) or the last k jobs (backward branching).

4.2. Bounding step

Let the initial upper bound (UB) be the better solution of H&J and HD algorithms, namely, $UB = \min\{C_{\max}^{HD}, C_{\max}^{H\&J}\}$. The bound is updated whenever a node of the search tree results in a makespan

smaller than the current upper bound. Determine a lower bound (LB) at each node of the tree. Since the policy of maintenance is dependent on the number of finished jobs, that is, x_i jobs have been done on M_i, the machine maintenance task occurs immediately. For convenient description, let J_e and J_f be two unscheduled jobs with respectively the shortest and the second shortest processing time on M_1. Moreover, let J_k and J_l be two unscheduled jobs with respectively the shortest and the second shortest processing time on M_2. The LB is derived as follows.

(1) $LB1 = \sum_{j=1}^{n} p_{1,j} + \sum_{j=1}^{n}(h_{1,[j-1]} \times t_1) + p_{2,k} = C_{1,\max} + p_{2,k}$

(2) $LB2 = C_{1,\max} + p_{2,k} + \min\{\max\{(p_{2,l} - p_{1,k} - h_{1,[n-1]} \times t_1),0\}, (p_{2,l} - p_{2,k})\}$

(i) If $p_{2,l} > p_{1,k} + h_{1,[n-1]} \times t_1$ and J_k is placed in the last position of the sequence, then $LB_2^1 = \sum_{j=1}^{n} p_{1,j} + \sum_{j=1}^{n}(h_{1,[j-1]} \times t_1) + p_{2,k} + (p_{2,l} - p_{1,k} - h_{1,[n-1]} \times t_1) = C_{1,\max} + p_{2,k} + (p_{2,l} - p_{1,k} - h_{1,[n-1]} \times t_1)$, as shown in Fig. 1.

(ii) If $p_{2,l} > p_{1,k} + h_{1,[n-1]} \times t_1$ and J_l is placed in the last position of the sequence, then $LB_2^2 = \sum_{j=1}^{n} p_{1,j} + \sum_{j=1}^{n}(h_{1,[j-1]} \times t_1) + p_{2,k} + (p_{2,l} - p_{2,k}) = C_{1,\max} + p_{2,l}$, as shown in Fig. 2.

From (i) and (ii), we know that $\min\{LB_2^1, LB_2^2\} = C_{1,\max} + p_{2,k} + \min\{(p_{2,l} - p_{1,k} - h_{1,[n-1]} \times t_1), (p_{2,l} - p_{2,k})\}$, if $p_{2,l} > p_{1,k} + h_{1,[n-1]} \times t_1$. Therefore, $LB2 = C_{1,\max} + p_{2,k} + \min\{\max\{(p_{2,l} - p_{1,k} - h_{1,[n-1]} \times t_1),0\}, (p_{2,l} - p_{2,k})\}$.

(3) $LB3 = \sum_{j=1}^{n} p_{2,j} + \sum_{j=1}^{n}(h_{2,[j-1]} \times t_2) + p_{1,e}$

(4) $LB4 = \sum_{j=1}^{n} p_{2,j} + \sum_{j=1}^{n}(h_{2,[j-1]} \times t_2) + p_{1,e} + \min\{\max\{(p_{1,f} - p_{2,e}), 0\}, (p_{1,f} - p_{1,e})\}$

(i) If $p_{1,f} > p_{2,e}$, and J_e is placed in the first position of the schedule, then $LB_4^1 = \sum_{j=1}^{n} p_{2,j} + \sum_{j=1}^{n}(h_{2,[j-1]} \times t_2) + p_{1,e} + (p_{1,f} - p_{2,e})$, as shown in Fig. 3.

(ii) If $p_{1,f} > p_{2,e}$, and J_f is placed in the first position of the sequence, then $LB_4^2 = \sum_{j=1}^{n} p_{2,j} + \sum_{j=1}^{n}(h_{2,[j-1]} \times t_2) + p_{1,f}$, as shown in Fig. 4.

From (i) and (ii), we know that $\min\{LB_4^1, LB_4^2\} = \sum_{j=1}^{n} p_{2,j} + \sum_{j=1}^{n}(h_{2,[j-1]} \times t_2) + p_{1,e} + \min\{(p_{1,f} - p_{2,e}), (p_{1,f} - p_{1,e})\}$, if $p_{1,f} > p_{2,e}$. Therefore, $LB4 = \sum_{j=1}^{n} p_{2,j} + \sum_{j=1}^{n}(h_{2,[j-1]} \times t_2) + p_{1,e} + \min\{\max\{(p_{1,f} - p_{2,e}),0\}, (p_{1,f} - p_{1,e})\}$.

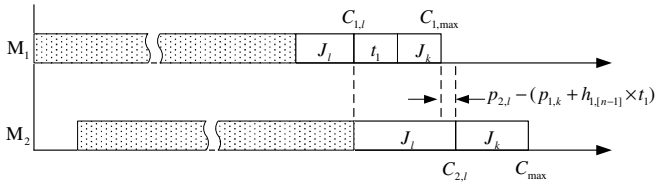

Fig. 1. Scheme of LB_2^1.

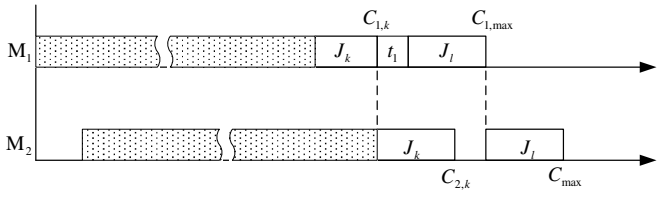

Fig. 2. Scheme of LB_2^2.

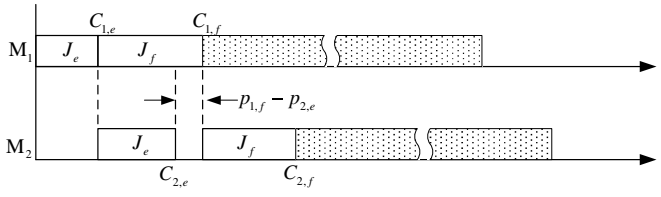

Fig. 3. Scheme of LB_4^1.

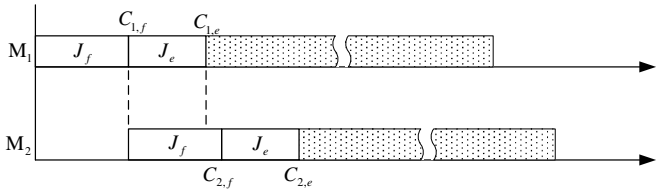

Fig. 4. Scheme of LB_4^2.

(5) $LB5 = C_{1,\max} + p_{2,k} + \min\{\max\{(p_{2,l} + t_2 - p_{1,k} - h_{1,[n-1]} \times t_1),0\}, (p_{2,l} - p_{2,k})\}$, when a hole occurs before the last job on M_2.

(i) If $p_{2,l} + t_2 > p_{1,k} + h_{1,[n-1]} \times t_1$ and J_k is placed in the last position of the sequence, then $LB_5^1 = C_{1,max} + p_{2,k} + (p_{2,l} + t_2 - p_{1,k} - h_{1,[n-1]} \times t_1)$, as presented in Fig. 5.

(ii) If $p_{2,l} + t_2 > p_{1,k} + h_{1,[n-1]} \times t_1$ and J_l is placed in the last position of the sequence, then $LB_5^2 = C_{1,\max} + p_{2,l}$, as presented in Fig. 6.

From (i) and (ii), we know that $\min\{LB_5^1, LB_5^2\} = C_{1,\max} + p_{2,k} + \min\{(p_{2,l} + t_2 - p_{1,k} - h_{1,[n-1]} \times t_1), (p_{2,l} - p_{2,k})\}$, if $p_{2,l} + t_2 > p_{1,\ k} + h_{1,[n-1]} \times t_1$. Therefore, $LB5 = C_{1,\max} + p_{2,k} + \min\{\max\{(p_{2,l} + t_2 - p_{1,k} - h_{1,[n-1]} \times t_1),0\}, (p_{2,l} - p_{2,k})\}$.

(6) $LB = \max\{LB2, LB4, LB5\}$. From above derivation, it can be seen that $LB2 \geqslant LB1$, $LB4 \geqslant LB2$, and $LB5 \geqslant LB1$. Therefore, $LB = \max\{LB2, LB4, LB5\}$.

4.3. Fathoming step

For each new node, we apply the following three fathoming tests to help us prune nodes:

(1) If the lower bound is larger than or equal to *UB*, i.e., $LB \geqslant UB$, then the node is dominated by the best solution so far. (Non-optimal solution.)
(2) If the node has no child, then prune it. (Infeasible solution.)
(3) If the node is a feasible solution and its corresponding makespan is smaller than UB, then update the best solution; otherwise, prune it. (Feasible solution.)

5. Experiment and results

This section presents the evaluation of the performances of H&J and HD algorithms by computational experiments, where all the algorithms were coded in C++ and run on a Pentium 2.0 G PC. The experiments conducted under the following three categories:

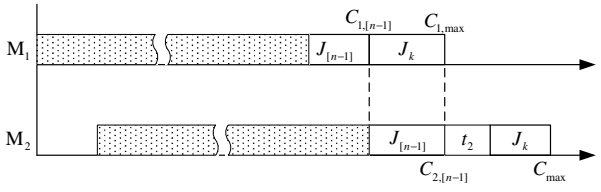

Fig. 5. Scheme of LB_5^1.

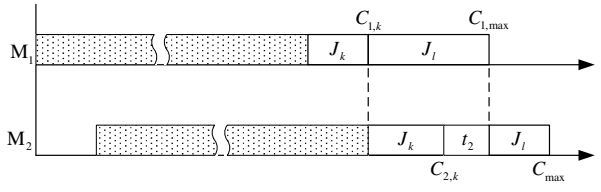

Fig. 6. Scheme of LB_5^2.

Table 1
Results of comparison between HD and H&J (holes on M_2)

n	# of instances			% of $C_{max}^{HD} \leqslant C_{max}^{H\&J}$
	$C_{max}^{HD} < C_{max}^{H\&J}$	$C_{max}^{HD} = C_{max}^{H\ J}$	$C_{max}^{HD} > C_{max}^{H\&J}$	
20	4	76	0	100
30	3	77	0	100
40	0	80	0	100
50	1	79	0	100
60	0	80	0	100
Total	8	392	0	100

Table 2
Performance of H&J (holes on M_2)

n	# of $C_{max}^{H\&J} = C_{max}^{*}$				% of $C_{max}^{H\&J} = C_{max}^{*}$
	$x = 3$	$x = 5$	$x = 7$	$x = 9$	
20	17	20	20	19	95.00
30	20	20	18	19	96.25
40	20	20	20	20	100.00
50	20	20	19	20	98.75
60	20	20	20	20	100.00
Average					98.00

Table 3
Performance of HD (holes on M_1 or M_2)

n	# of $C_{max}^{HD} = C_{max}^{*}$				% of $C_{max}^{HD} = C_{max}^{*}$
	$x = 3$	$x = 5$	$x = 7$	$x = 9$	
20	20	20	20	20	100
30	20	20	20	20	100
40	20	20	20	20	100
50	20	20	20	20	100
60	20	20	20	20	100
Average					100

Table 4
Results of comparison between HD and H&J (holes on M_1 and M_2)

n	# of instances			% of $C_{max}^{HD} \leqslant C_{max}^{H\&J}$
	$C_{max}^{HD} < C_{max}^{H\&J}$	$C_{max}^{HD} = C_{max}^{H\&J}$	$C_{max}^{HD} > C_{max}^{H\&J}$	
20	6	314	0	100.00
30	13	318	2	100.00
40	9	318	3	100.00
50	7	312	2	99.69
60	3	320	0	100.00
Total	38	1555	7	99.94

(1) holes on M_1;
(2) holes on M_2; and
(3) holes on M_1 and M_2.

The experimental instances were generated randomly. The total number of jobs, n, are 20, 30, 40, 50 and 60 while the numbers of block jobs, x, are 3, 5, 7, and 9 (four different blocks). Therefore, experimental instances for the first and second condition were both 400, and experimental instances for the third condition were 1600.

The job processing times ($p_{i,j}$) and the lengths of the holes (t_i) were randomly generated from uniform distribution in [20,50]. With assembly line balancing taken into consideration, the processing times were adjusted as follows. (1) If machine maintenance occurs only on M_1, then $p_{1,j}$ is adjusted as $p_{1,j} \times x_1/(x_1 + 1)$. (2) If machine maintenance occurs only on M_2, then $p_{2,j}$ is adjusted as $p_{2,j} \times x_2/(x_2 + 1)$.

The two proposed algorithms were analyzed in terms of solution quality. First, HD was compared with H&J, and both were evaluated in terms of relative performance. Then, the mean and worst relative performance of H&J and HD were calculated according to the optimum. The process is described as follows.

(1) Mean relative performance of H&J and HD:

$$MR = \frac{1}{N}\left(\sum_{i=1}^{N} \frac{C_{max}(i) - C_{max}^{*}(i)}{C_{max}^{*}(i)}\right) \times 100\%$$

(2) Worst relative performance of H&J and HD:

$$WR = \max_i\left(\frac{C_{max}(i) - C_{max}^{*}(i)}{C_{max}^{*}(i)}\right) \times 100\%$$

For the first category, the solution quality of HD was exactly the same as that of H&J, and both completely equal to optimum. For the second category, the solution quality of HD is slightly superior to that of H&J, as shown in Table 1. H&J and HD obtained optimum solution in all 400 instances, which are 98% and 100% optimality, respectively, as shown in Tables 2 and 3. Apparently, in the first two categories, HD obtained optimum solution in all 800 instances, which was 100% optimality. In other words, HD performs excellently when holes occur either on the first or second machine.

The third category included 16 combinations of $x_1 \times x_2$, where x_1 and x_2 are the number of holes on M_1 and M_2, respectively. For each combination, 100 replications were run, and the total instances were 1600. The results are given in Table 4. As can be seen, HD performed better than H&J. Of the 1600 instances, H&J and HD obtain the optimum solution in 1562 and 1582 instances, respectively. The worst relative error values of H&J and HD, as shown in Tables 5 and 6, are only 1.3100% and 0.2913%, respectively, and the mean are 0.0058% and 0.0017%, respectively.

The average computation times of HD were 0.023 s, 0.060 s, 0.123 s, 0.205 s and 0.343 s for the 20-, 30-, 40-, 50-, and 60-job instance, respectively. The longest computation time of HD is only

Table 5
Performance of H&J (holes onM_1 and M_2)

n	# of $C_{max}^{HD} = C_{max}^{*}$				% of $C_{max}^{H\&J} = C_{max}^{*}$	MR (WR)%
	$3 \times 3,5,7,9$	$5 \times 3,5,7,9$	$7 \times 3,5,7,9$	$9 \times 3,5,7,9$		
20	79	77	78	80	98.13	0.0125 (1.3100)
30	77	73	80	77	95.94	0.0092 (0.5803)
40	78	77	79	77	97.19	0.0022 (0.1380)
50	77	77	80	79	97.81	0.0046 (0.4464)
60	79	78	80	80	99.06	0.0006 (0.0939)
Average					97.63	0.0058 (1.3100)

Table 6
Performance of HD (holes on M_1 and M_2)

n	# of $C_{max}^{HD} = C_{max}^{*}$				% of $C_{max}^{HD} = C_{max}^{*}$	MR (WR)%
	$3 \times 3,5,7,9$	$5 \times 3,5,7,9$	$7 \times 3,5,7,9$	$9 \times 3,5,7,9$		
20	80	79	80	80	99.69	0.0005 (0.1456)
30	80	79	80	79	99.38	0.0006 (0.1003)
40	80	78	78	78	98.13	0.0034 (0.2913)
50	76	77	80	78	97.19	0.0040 (0.2753)
60	80	80	80	80	100.00	0 (0)
Average					98.88	0.0017 (0.2913)

0.600 s. The data indicate that HD can obtain good solution within a very short computation time. The above instances were solved by the B&B algorithm within 0.028 s, 0.070 s, 0.125 s, 0.276 s and 0.348 s, respectively. The longest computation time of B&B is only 6.594 s. The data show that the proposed B&B algorithm had very good upper bound and lower bound, which significantly decrease the computation time.

6. Conclusion

This paper has addressed a special case of a two-machine flowshop problem with availability constraints imposed on the first or/and the second machines. Moreover, we have developed a constructive heuristic, HD, which is based on the difference in processing times of job on two machines. We observe that the HD algorithm is slightly superior to H&J algorithm, which modifies HI algorithm and H algorithm.

When holes only occur on M_1, both the percentages of optimum of H&J and HD are 100%; when holes only occur on M_2, the percentages of optimum of two heuristics are 98% and 100%, respectively. As holes occur on M_1 and M_2, the average percentages of obtaining the optimum of H&J and HD are 97.63% and 98.88%, respectively. The mean relative performances of H&J and HD according to the optimum, MR, are 0.0058% and 0.0017%, and the worst relative performances of H&J and HD according to the optimum, WR, are only 1.3100% and 0.2913%, respectively. The average computation time of HD is only 0.343 s for the 60-job instances. We may, therefore, conclude that the performance of the two heuristics is superior, especially the HD algorithm. HD is not only very close to the optimum but it obtains excellent efficiency. Also, the proposed modified lower bound of the B&B algorithm greatly contributes to the computation efficiency. For future research, it is interesting to develop a good heuristic for hybrid flowshop problem with various machine availability constraints.

Acknowledgments

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

References

Aggoune, R. (2004). Minimizing the makespan for the flow shop scheduling problem with availability constraints. *European Journal of Operational Research, 153*, 534–543.
Aggoune, R., Mahdi, A. H., & Portman, M. C. (2001). Genetic algorithm for the flow shop scheduling problem with availability constraints. *IEEE International Conference on Systems, Man, and Cybernetics, 4*, 2546–2551.
Art, R. H. P. M., Knapp, G. M., & Lawrence, M. J. (1998). Some aspects of measuring maintenance in the process industry. *Journal of Quality in Maintenance Engineering, 4*, 6–11.
Blazewicz, J., Breit, J., Formanowicz, P., Kubiak, W., & Schmidt, G. (2001). Heuristic algorithms for the two-machine flowshop with limited machine availability. *Omega, 29*, 599–608.
Breit, J. (2004). An improved approximation for two-machine flow shop scheduling with an availability constraint. *Information Processing Letters, 90*, 273–278.
Cheng, T. C. E., & Wang, G. (1999). Two-machine flowshop scheduling with consecutive availability constraints. *Information Processing Letters, 71*, 49–54.
Cheng, T. C. E., & Wang, G. (2000). An improved heuristic for two-machine flowshop scheduling with an availability constraint. *Operation Research Letters, 26*, 223–229.
Dell'Amico, M., & Martello, S. (2001). Bounds for cardinality constrained P||Cmas problem. *Journal of Scheduling, 4*, 123–138.
Kubiak, W., Blazewicz, J., Formanowicz, P., Breit, J., & Schmidt, G. (2002). Two-machine flow shops with limited machine availability. *European Journal of Operational Research, 136*, 528–540.
Lee, C. Y. (1997). Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operation Research Letters, 20*, 129–139.
Lee, C. Y. (1999). Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research, 114*, 420–429.
Liao, C. J., Chen, C. M., & Lin, C. H. (2007). Minimizing makespan for two parallel machines with job limit on each availability interval. *Journal of Operation Research Society, 58*, 938–947.
Pinedo, M. (2002). *Scheduling: Theory* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.
Sanlaville, E., & Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica, 35*, 795–811.
Schmidt, C. (2000). Scheduling with limited machine availability. *European Journal of Operational Research, 121*, 1–15.
Sule, D. R. (1997). *Industrial scheduling*. Boston, MA: PWS.