Two-Machine Flowshop Scheduling with Conditional Deteriorating Second Operations*

B.M.T. Lin[1,#] and T.C.E. Cheng[2]

[1]Department of Information and Finance Management

Institute of Information Management

National Chiao Tung University, Hsinchu, Taiwan

[2]Department of Logistics

The Hong Kong Polytechnic University, Kowloon, Hong Kong

Abstract: This paper considers a flowshop scheduling problem with a waiting time constraint imposed to restrict the processing of the two operations of each job. If the second operation of a job cannot start within a specified waiting time after the completion of its first operation, then an extra processing time will be incurred for its second operation as a penalty. We first show that is even a greatly restricted version of the problem is strongly $\mathcal{NP}$-hard. We then develop an $O(n^2)$ algorithm to determine the makespan of a processing sequence of the jobs.

Keywords: Flowshop scheduling, waiting time, makespan, NP-hardness

---

*: The first author was partial supported by the National Science Council of the R.O.C. under grant number NSC-93-2416-H-009-026, and the second author was supported in part by The Hong Kong Polytechnic University under a grant from the Area of Strategic Development in China Business Services.

#: Corresponding author: Prof. B.M.T. Lin; E-mail: bmtlin@mail.nctu.edu.tw; Tel: +886-3-5131472.

1 Introduction

Flowshop scheduling, first considered by Johnson (1954), has been one of the most extensively studied problems in scheduling research. Numerous variants, properties and solution methods are available in the literature (Lawler et al. 1993; Reisman 1997). In this paper we consider a two-machine flowshop scheduling problem with possible penalties imposed on the second operations of the jobs. In most deterministic scheduling problems, the job parameters are known in advance and fixed throughout the solution process. Since the last decade, scheduling problems with time- or position-dependent processing times have received considerable research attention (Cheng et al. 2003).

The problem under study in this paper is defined as follows. There is a set of jobs $N = \{1, 2, \ldots, n\}$ to be processed on a two-machine flowshop from time zero onwards. No preemption is allowed. The processing times of the first (or machine-one) and the second (or machine-two) operations of job i are denoted by p_i and q_i, respectively. Due to the characteristics of the manufacturing environment, the machine-two operation of job i has a basic processing time a_i and must start processing no later than l_i time units after the completion of its machine-one operation. If the time lag constraint is violated, the processing time on machine two will be lengthened by b_i time units as a penalty. In other words, the processing time on machine two will become $a_i + b_i$ as a result. We assume that all the parameters are non-negative integers, and that the processing sequences on the two machines are the same. The problem seeks to compose a schedule such that the makespan is minimum. Following the standard three-field notation (Graham et al. 1979), we use $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ to denote the problem under study. The terms sequence and schedule are used interchangeably in some scheduling studies. In this paper a sequence simply means a processing order of the jobs, while a schedule precisely specifies the starting time of each job.

If $l_i = \infty$ or $b_i = 0$, the studied problem reduces to the classical two-machine flowshop scheduling problem, which is solvable in $O(n \log n)$ time by Johnson's algorithm (Johnson 1954). On the other hand, if $l_i = 0$ and $b_i = \infty$, then the constraint is hard and the $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ problem is equivalent to the problem with no-wait constraints, $F2/\text{no-wait}/C_{max}$, which is equivalent to a special version of the traveling

salesperson problem and can be solved in $O(n \log n)$ time (Gilmore and Gomory 1964; Hall and Sriskandarajah 1996). The $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ problem is similar to the scheduling problem introduced by Mitten (1958) where a time lag D_i is specified for the two operations of each job such that the second operation cannot start in less than D_i time units after the first operation is started and the second operation cannot complete in less than D_i time units after the first operation is completed. An $O(n \log n)$ algorithm was also proposed by Mitten (1958). Addressing the same issue considered in Mitten's paper, Johnson (1959) provided an alternative proof and presented several properties for general cases. Later, Mitten (1959) investigated a case where the time lags for the start and completion of each job's two operations may be different. To the best of our knowledge, the $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ problem has not been studied in the scheduling literature.

In this paper, we will first show that the $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ is strongly $\mathcal{NP}$-hard even if restricted conditions are imposed. Then, we explore a property that can be deployed to determine the makespan of a fixed processing sequence. We will also discuss potential developments.

2 Computational Complexity

In this section we present $\mathcal{NP}$-hardness results by a reduction from 3-Partition, which is $\mathcal{NP}$-hard in the strong sense (Garey and Johnson 1979).

3-Partition**:** Given a non-negative integer B and a set of $3m$ non-negative integers $A = \{x_1, x_2, \ldots, x_{3m}\}$ with $1/4 < x_i < 1/2$ for each x_i and $\sum_{i=1}^{3m} x_i = mB$, is there a partition $A_1, A_2, \ldots, A_m$ of set A such that for each subset $A_j, \sum_{x_i \in A_j} x_i = B$?

Theorem 1 *The $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ problem is strongly $\mathcal{NP}$-hard even if all the jobs have the same machine-one processing time, lag and penalty.*

Proof: It is easy to see that the decision version of $F2/q_i = a_i$ or $a_i + b_i/C_{max}$with the specified constraints belongs to $\mathcal{NP}$. Given an instance of 3-Partition, we create an instance of $4m + 1$ jobs for $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ as follows:

Ordinary jobs: $a_i = B + x_i, 1 \leq i \leq 3m$;

Enforcer jobs: $a_{3m+i} = 0, 1 \leq i \leq m+1$.

All the jobs have machine-one processing time $p_i = B$, time lag $l_i = B$ and penalty $b_i = 1$.

With the above two instances, we show that there is a partition as specified for set A if and only if there is a schedule for $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ whose makespan is no greater than $(4m+1)B$. Without loss of generality we assume $m \geq 2$.

(**IF**) Let subsets $A_1, A_2, \ldots, A_m$ constitute a partition of set A as specified in 3-PARTITION. We compose a schedule for $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ as follows: The three jobs corresponding to the elements of A_1 come first and job $3m+1$ follows. Then, the three jobs corresponding to the elements of A_2 and job $3m+2$ are scheduled. Repeat the scheduling pattern for the remaining jobs and schedule jobs $4m$ and $4m+1$ last. The Gantt chart in Figure 1 illustrates the arrangement. It can easily be verified that the makespan of the schedule is $(4m+1)B$.

Insert Figure 1 here.

(**ONLY IF**) Assume S is a schedule for $F2/q_i = a_i$ or $a_i + b_i/C_{max}$with makespan no greater than $(4m+1)B$. Because the sum of machine-one processing times of all the jobs is exactly $(4m+1)B$, no idle time is allowed on machine one. The sum of the machine-two processing times of all the jobs is exactly $4mB$ and an idle time of B time units is inevitable for the first operation on machine two. Therefore, no other idle time can be incurred on the second machine. Moreover, no penalty can be incurred by any job.

We start the analysis from the job scheduled in the first position of the schedule. If any enforcer job occupies the first position, an idle time of $2B$ time units will be incurred for the first two jobs on machine two. As a result, the first job must be an ordinary one. Because all the enforcer jobs are homogeneous, we assume without loss of generality that they are arranged in ascending order of their indices.

Let N_1 denote the set of jobs preceding job $3m+1$. We consider the following two cases:

(1) $|N_1| < 3$:

On the two machines, the completion times of job $3m+1$ are $|N_1|B+B$ and $B+|N_1|B+\sum_{i\in N_1} x_i$. Consider the difference $(B + |N_1|B + \sum_{i\in N_1} x_i) - (|N_1|B + B) = \sum_{i\in N_1} x_i$. The inequality $\sum_{i\in N_1} x_i < B$ must hold because $x_i < B/2$ and $|N_1| \leq 2$. Therefore, on machine two a non-zero idle time will be introduced to the job that immediately follows job $3m + 1$, and a contradiction arises.

(2) $|N_1| \geq 3$:

Let $N_1 = \{i_1, i_2, \ldots, i_{N_1}\}$ and consider the first three jobs i_1, i_2, i_3. If $x_{i_1} + x_{i_2} + x_{i_3} > B$, then the difference between the completion times of job i_3 on both machines is

$$(B + (B + x_{i_1}) + (B + x_{i_2}) + (B + x_{i_3})) - 3B,$$

which is greater than $2B$. Thus, the fourth job i_4, or the enforcer job $3m + 1$ if $|N_1| = 3$, needs to be deferred or penalized and a contradiction arises. On the other hand, if $x_{i_1} + x_{i_2} + x_{i_3} < B$ and $|N_1| \geq 4$, then the difference between the completion times of job i_4 on both machines is

$$\begin{aligned} & (B + (B + x_{i_1}) + (B + x_{i_2}) + (B + x_{i_3}) + (B + x_{i_4})) - 4B \\ = \;& B + x_{i_1} + x_{i_2} + x_{i_3} + x_{i_4}, \end{aligned}$$

which is greater than $2B$ because $x_i > B/4$ for all i. Similarly, a contradiction arises. if $x_{i_1} + x_{i_2} + x_{i_3} < B$ and $|N_1| = 3$, then the difference between the completion times of job $3m + 1$ on both machines is

$$(B + (B + x_{i_1}) + (B + x_{i_2}) + (B + x_{i_3})) - 4B,$$

which is smaller than B, implying non-zero idle time for the immediate successor of $3m+1$ on machine one. Excluding the possibilities discussed above, we establish that the equality $x_{i_1} + x_{i_2} + x_{i_3} = B$ must hold. However, we still need to show that N_1 contains exactly three elements. If $|N_1| > 3$, then when considering the fourth job i_4, we come up with the same contradiction concerning the successor of job i_4.

From the above discussion, we have that $|N_1| = 3$ and $x_{i_1} + x_{i_2} + x_{i_3} = B$. We let the elements corresponding to the jobs of N_1 constitute subset A_1. Continuing the same line of reasoning, we can come up with $A_2, A_3, \ldots$, and A_m as required. Therefore, we have derived a partition for set A as specified in 3-PARTITION. The proof is thus accomplished.

Treating the problem in a mirror way from the other side of the time horizon, we can also show that the $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ problem remains strongly $\mathcal{NP}$-hard even if all the jobs have the same basic machine-two processing time, lag and penalty. A polynomial-time reduction from 3-PARTITION can be carried out by creating a job set of $4m + 1$ jobs as follows:

$p_i = x_i, 1 \leq i \leq 3m$;

Enforcer jobs: $p_{3t+i} = 3B, 1 \leq i \leq m + 1; p_{4m+1} = 0$.

All the jobs have the same $a_i = B, l_i = 2B$ and $b_i = M^2$.

Given the instance transformation, we claim that the desired partition of set A exists if and only if there is a schedule for $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ whose makespan is no greater than $(4m + 1)B$. The optimal job sequence is configured as

$$4m + 1, < A_1 >, 3m + 1, < A_2 >, 3m + 2, \cdots, < A_{m-1} >, 4m,$$

where $< A_i >$ denotes the jobs corresponding to the elements in subset A_i.

3 Determining the Makespan of a Job Sequence

In most flowshop scheduling problems, the makespan is readily determined when the order of the jobs is known. The $F2/q_i = a_i$ or $a_i + b_i/C_{max}$ problem, however, exhibits a very intriguing characteristic about the determination of the *optimal* makespan of a given sequence. This difficulty arises mainly from the decision about whether or not to postpone some machine-one operations to prevent the possible incurring of penalties. In the worst case, there will be $O(2^n)$ combinations of decisions of whether or not to defer each of the n jobs. In this section, we develop a polynomial time procedure to determine the optimal makespan for a give job sequence.

Assume a sequence $[1], [2], \cdots, [n]$ is given, where $[i]$ denotes the job in position i of the sequence. We call a job *deferred* if idle time is inserted before its machine-one operation so as to avoid the penalty that might be incurred by its machine-two operation. It is the variability caused by deferred jobs that makes the determination of the makespan difficult. A key to overcoming this obstacle is based upon the observation that the last

deferred job will hedge the variability. Therefore, we focus on schedules with the last deferred job fixed. Consider any two schedules S_1 and S_2 for the first j jobs with job $[i], 1 \leq i \leq j \leq n$, as the last deferred job. Let $C^1_{S_k}(i,j)$ and $C^2_{S_k}(i,j)$ denote the completion times of job $[i]$ on machine one and machine two in schedule S_k, $k = 1$ or 2. Due to the role of job $[i]$, the differences between the completion times on the two machines are the same for the two batches. In other words, although it is possible that $C^1_{S_1}(i,j) \neq C^1_{S_2}(i,j)$ and $C^2_{S_1}(i,j) \neq C^2_{S_2}(i,j)$, the differences are the same, i.e., $C^2_{S_1}(i,j) - C^1_{S_1}(i,j) = C^2_{S_2}(i,j) - C^1_{S_2}(i,j)$. Furthermore, in both schedules, no idle time is inserted on machine one for any successor of job $[i]$. Therefore, the successor jobs of job $[i]$ have the same processing mode as, in both schedules, they are both either normal or penalized. As shown in Figure 2, the shaded part of both schedules are the same.

Insert Figure 2 here.

Based upon the above observation, we may then develop an algorithm for calculating the optimal makespan of a given sequence. Let $F(i,j)$ denote the optimal makespan of the job sequence $[1], [2], \ldots, [j]$ given that job $[i], 0 \leq i \leq j \leq n$, is the last deferred job. Let $C^1(i,j)$ and $C^2(i,j)$ respectively, denote the completion times on the two machines for the schedule associated with $F(i,j)$. Initially, we set $C^1(0,1) = p_1, C^2(0,1) = p_1 + a_1, F(0,1) = C^2(0,1)$ and $F(1,1) = \infty$. The first three terms reflect that a dummy job is added for initialization. The term $F(1,1) = \infty$ indicates that it is not necessary to defer the machine-one operation of the first job. For any i and j, $1 < i \leq j \leq n$, $F(i,j) = \infty$ means that a schedule with job $[i]$ as the last deferred job cannot be optimal for the first j jobs.

For a fixed j, we want to calculate the values of $F(i,j)$ for all $1 \leq i \leq j$. To compute the value of $F(i,j)$, we need to know the completion times on the two machines corresponding to $F(i,j-1)$. Based upon the information conveyed in the schedule associated with $F(i,j-1)$, we have three possible cases: (1) job j can be normally processed without deferring or incurring a penalty; (2) job j is processed with a penalty incurred; and (3) job j is deferred. Therefore, considering $F(i,j-1)$ and job j will result in two entries

$F(i,j)$ and $F(j,j)$. The latter one dictates the fact that job j is deferred. Moreover, for any specific i and j, $1 < i < j$, $F(i,j) = \infty$ if and only if $F(i,j') = \infty$, $j < j' \leq n$. The detailed computation is outlined in the following algorithm.

Algorithm Find_Makespan

1: Initialization: $C^1(0,1) = p_1, C^2(0,1) = p_1 + a_1, F(0,1) = C^2(0,1)$; and $F(1,1) = \infty$;
2: For j=2 to n do
3: $F(j,j) = \infty$;
4: For $i = 0$ to $j-1$
5: If $F(i,j-1) = \infty$ then $F(i,j) = \infty$ else do the following steps
{
6: Case 1: $C^2(i,j-1) - C^1(i,j-1) - p_j \leq l_j$ /* Job j is normally processed */
{
7: $C^1(i,j) = C^1(i,j-1) + p_j$;
8: $C^2(i,j) = \max\{C^1(i,j), C^2(i,j-1)\} + a_j$;
9: $F(i,j) = C^2(i,j)$;
}
10: Case 2: $C^2(i,j-1) - C^1(i,j-1) - p_j > l_j$;
{
/* Job j is penalized; */
11: $C^1(i,j) = C^1(i,j-1) + p_j$;
12: $C^2(i,j) = C^2(i,j-1) + a_j + b_j$;
13: $F(i,j) = C^2(i,j)$;
/* Job j is deferred; */
14: If $F(j,j) = \infty$ or $C^2(i,j-1) + a_j < F(j,j)$ then
{
15: $C^1(j,j) = C^2(i,j-1) - l_j$;
16: $C^2(j,j) = C^2(i,j-1) + a_j$;
17: $F(j,j) = C^2(i,j)$;
}

}

}

18: Else $F(i,j) = \infty$;

19: Output $\min_{i=0}^{n}\{F(i,n)\}$.

The algorithm comprises a loop of $O(n^2)$ iterations, each of which takes $O(1)$ time. Therefore, we conclude the results with the following theorem.

Theorem 2 *Given a processing sequence of the jobs, the makespan can be determined in* $O(n^2)$ *time.*

4 Conclusion

In this paper we considered a flowshop scheduling problem with the consideration of lag penalties. The problem was shown to be strongly $\mathcal{NP}$-hard. To determine the makespan of a job sequence is not as straightforward as in the case of classical flowshop scheduling problems. We elaborated on a property and applied it to develop an $O(n^2)$ algorithm for calculating the makespan of a given job sequence. For further research, it is interesting to extend the study to the general flowshop with m machines, where m is an input variable. For example, settling the complexity status of determining the makespan of a fixed sequence in the general flowshop setting is a worthy topic.

References

Cheng, T.C.E., Ding, Q., Lin, B.M.T., 2003. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, 152, 1-13.

Garey, M.R., Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*, Freedman, San Francisco, CA.

Gilmore, P.C., Gomory, R.E., 1964. Sequencing a one-state variable machine: a solvable case of the traveling salesman problem. *Operations Research*, 12, 655-679.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, 5, 287-326.

Hall, N., Sriskandarajah, C., 1996. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44, 510-525.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1, 61-68.

Johnson, S.M., 1959. Discussions: Sequencing n jobs on two machine with arbitrary time lags. *Management Science*, 5, 299-303.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., ShmoysD.B., 1993. Sequencing and scheduling: algorithms and complexity. In *Logistics of Production and Inventory*, (S.C. Graves, A.H.G. Rinnooy Kan and P.H. Zipkin, Eds.) North Holland, Amsterdam, the Netherlands, 445-522.

Mitten, L.G., 1958. Sequencing n jobs on two machine with arbitrary time lags. *Management Science*, 5, 293-298.

Mitten, L.G., 1959. A scheduling problem: An analytical solution based upon two machines, n jobs, arbitrary start and stop lags, and common sequences. *Journal of Industrial Engineering*, 10, 131-135.

Reisman, A., Kumar, A., Motwani, J., 1997. Flowshop scheduling/sequencing research: A statistical review of the literature, 1952-1994. *IEEE Transactions on Engineering Management*, 44, 316-329.

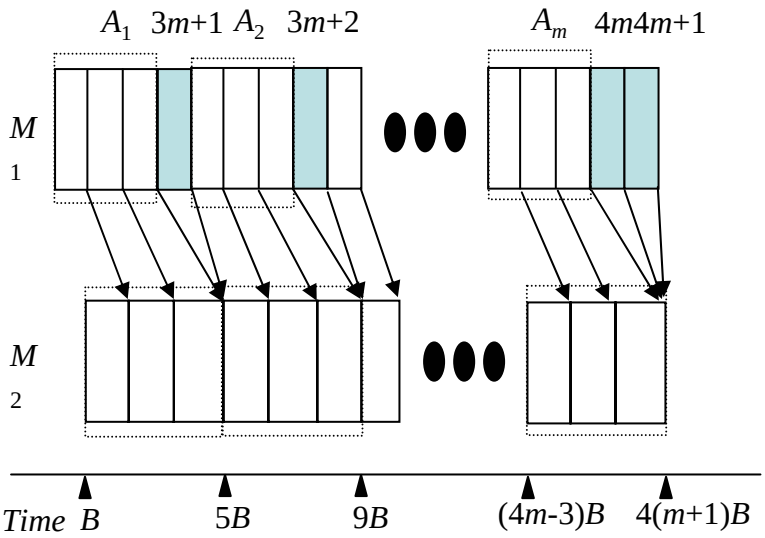

Figure 1: The schedule used in the proof of Theorem 1.

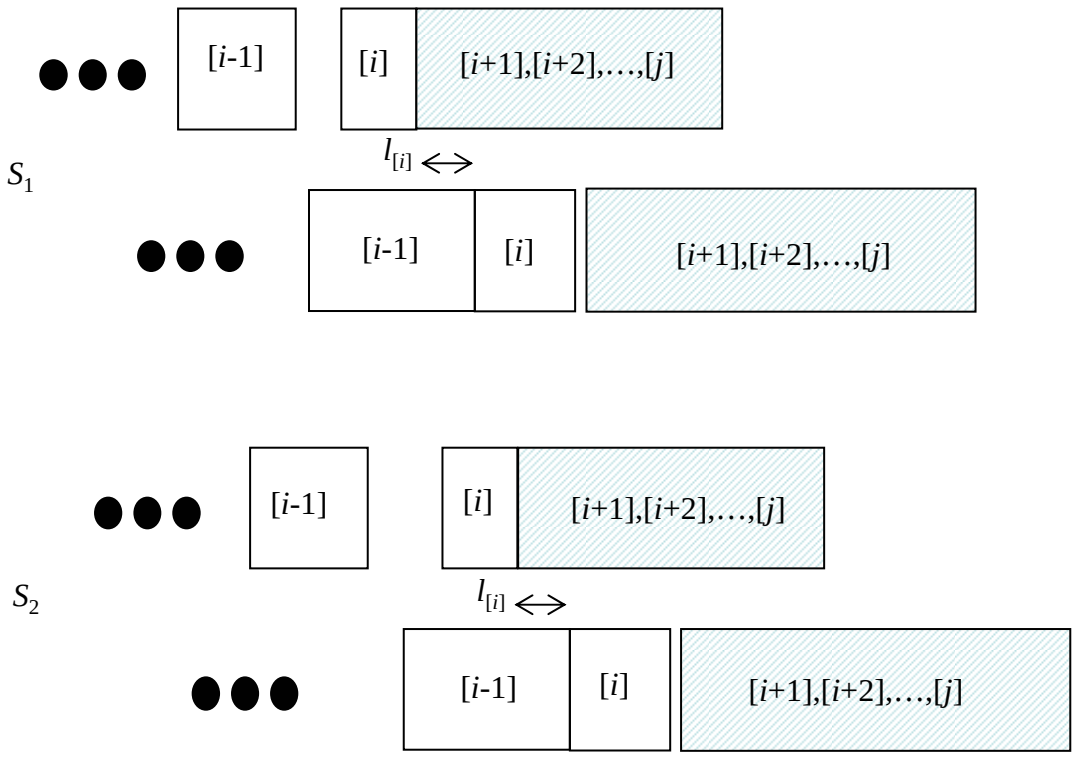

Figure 2: Two schedules with the same job [i] as the last deferred job.