Scheduling of Three-Operation Jobs in a Two-Machine Flow Shop with mean flow time measure

Hee Jin Ha, Chang Sup Sung

Department of Industrial Engineering, KAIST, rainwhite-@hanmail.net

Abstract

This paper considers a two-machine flow-shop scheduling problem for minimizing mean flow time. Each job has three non-preemptive operations, where the first and third operations must be processed on the first and second machines, respectively, but the second operation can be processed on either machine. A lower bound based on SPT rule is derived, which is then used to develop a branch-and-bound algorithm. Also, an efficient simple heuristic algorithm is developed to generate a near-optimal schedule. Numerical experiments are performed to evaluate the performances of the proposed branch-and-bound and the heuristic algorithm.

I. Introduction

This paper considers a two-machine flowshop scheduling problem for minimizing mean flow time. The problem is, however, different from classical problems in the sense that each job has three operations. The first operation must be processed on the first machine, the second operation can be processed on either machine, and the third operation must be processed on the second machine.

The three-operation jobs in a two-machine scheduling problem can be widely considered in various applications. For example, in SCM environment, the first and second machines can be a factory and a market respectively. The three operations can be manufacturing, transporting (or delivering) and selling, respectively. The manufacturing must be processed in the factory and the selling must be processed in the market. But the transporting, the second operation, can be processed on either factory or market.

As another application, it may be interesting to consider such a situation where a machine-independent setup operation between the two operations is needed on each job. The setup time is job-dependent and both machines are equipped with some tools requiring setups. Consequently, based on the coded instructions loaded on each machine prior to the processing of a set of jobs, the setup of an individual job is performed either after completing the first operation or prior to the start of the second operation.

In SCM environment, all the involved stages in the supply chain need to be managed in integration manner, which is a very competitive issue for each company to provide with quick and good services to customers. Therefore, this paper considers the scheduling measure, flowtime, which represents the time duration for each job to stay in the system, so that its minimization amounts to improving customer service in terms of responsiveness.

II. Problem description

There are n jobs, denoted by $J_1, J_2, \ldots, J_n$, to be processed on two machines M1 and M2. Job J_i consists of three operations J_{iA}, J_{iB} and J_{iC} for i=1,2…,n. The processing times of J_{iA}, J_{iB} and J_{iC} are denoted by P_{iA}, P_{iB} and P_{iC}, respectively. These operations are to be processed in the order J_{iA}, J_{iB} and J_{iC}. Operations J_{iA} and J_{iC} are to be processed on machines M1 and M2, respectively. On the other hand, operation J_{iB} may be processed on either machine M1 or machine M2 and is always processed after operation J_{iA} and before operation J_{iC}. It is assumed that preemption is not allowed, i.e., any operation once started must be completed without interruption. All the jobs are available to be processed at the time 0. The objective is to minimize the mean flow time.

For notational convenience, the following notation will be considered in this thesis: $J_i=(P_{iA},P_{iB},P_{iC})$ for job i. For example, $J_2=(1,2,3)$ implies that $P_{2A}=1$, $P_{2B}=2$ and $P_{2C}=3$. Denote by M1 the first stage machine, and denote by M2 the second stage machine.

III. Solution Analysis

The proposed problem is a modification of the classical two-machine flow-shop scheduling problem. Namely if there are only two operations, then it is seen as the two-machine flow-shop problem. Garey, Johnson, and Sethi.[14] have proved that $F2||\sum C_j$ problem is strongly NP-hard. Thus, the proposed problem is NP-hard in the strong sense.

The first solution property shows that proposed three-operation two-machine flow-shop problem reduces to the classical two-machine flow-shop problem once it is known which jobs have their second operations assigned to machine A and which ones assigned to machine B.

1) Property1

If the second operations of job i and job j are assigned to machine 1 or 2, then the processing time of job i at machine 1 is αi and at machine 2 is βi. Also the processing time of job j at machine 1 is αj and at machine 2 is βj.

If $\alpha i \leq \alpha j$, $\beta i \leq \beta j$ and $\alpha i \geq \beta i$ or $\alpha j \leq \beta j$, then job i precedes job j.

Proof)

Consider a schedule S with job j preceding job i. Now construct a new schedule S', in which jobs i and j are interchanged in the schedule and all other jobs are completed at the same time as in S. Denote by t_B denotes the point in time at which job i begins in S and at which job j begins in S'. Also, denote by B the set of jobs that precede jobs i and j in both schedules and denote by A the set of jobs that follow jobs i and j in both schedules. $F_k(S)$ represents the flowtime of job k under schedule S.

It will suffice to deal with $\sum_{k=1}^{n} F_k$ as a criterion, since this differs from $\overline{F}$ only in division by a constant.

$$\sum_{k=1}^{n} F_k(S) = \sum_{k\in B} F_k(S) + (t_B + \alpha_j + \beta_j) + (t_B + \alpha_j + \alpha_i + \beta_i) + \sum_{k\in A} F_k(S)$$

$$\overset{or}{=} \sum_{k\in B} F_k(S) + (t_B + \alpha_j + \beta_j) + (t_B + \alpha_j + \beta_j + \beta_i) + \sum_{k\in A} F_k(S)$$

$$\sum_{k=1}^{n} F_k(S') = \sum_{k\in B} F_k(S') + (t_B + \alpha_i + \beta_i) + (t_B + \alpha_i + \alpha_j + \beta_j) + \sum_{k\in A} F_k(S')$$

$$\overset{or}{=} \sum_{k\in B} F_k(S') + (t_B + \alpha_i + \beta_i) + (t_B + \alpha_i + \beta_i + \beta_j) + \sum_{k\in A} F_k(S')$$

$$\sum_{k=1}^{n} F_k(S) - \sum_{k=1}^{n} F_k(S') \geq 0$$

In other words, the interchange of jobs i and j reduces the value of $\overline{F}$. Therefore, job i precedes job j. This completes the proof. □

Corollary 1

If the relations $P_{iA} < P_{jA}$, $P_{iB} < P_{jB}$ and $P_{iC} < P_{jC}$ hold, then the job i precedes job j.

Proof)

If the second operations of job i and j are assigned to the same machine, then the results of Property 2 will hold. This completes the proof. □

2) Property2

If jobs i and j have the same average of their these operation processing times, but how different operation processing times, and if the starting time on machine 2 is shorter than the completion time of the first time of job i or longer than completion time of the first operation of job j.

And if $P_{iB}=P_{jB}$ and $P_{iA} < P_{jA}$, then job i precedes job j.

Proof)

Consider a schedule S with job j preceding job i. Now construct a new schedule S', in which jobs i and j are interchanged in schedule. Denote by B denotes the set of jobs that precede jobs i and j in both schedules and denote by A denotes the set of jobs that follow jobs i and j in both schedules. $F_k(S)$ represents the flowtime of job k under schedule S.

It will suffice to deal with $\sum_{k=1}^{n} F_k$ as a criterion, since this differs from $\overline{F}$ only in division by a constant.

$$\sum_{k=1}^{n} F_k(S) = \sum_{k\in B} F_k(S) + (t_B + \alpha_j + \beta_j) + (t_B + \alpha_j + \alpha_i + \beta_i) + \sum_{k\in A} F_k(S)$$

$$\overset{or}{=} \sum_{k\in B} F_k(S) + (t_B + \alpha_j + \beta_j) + (t_B + \alpha_j + \beta_j + \beta_i) + \sum_{k\in A} F_k(S)$$

$$\sum_{k=1}^{n} F_k(S') = \sum_{k\in B} F_k(S') + (t_B + \alpha_i + \beta_i) + (t_B + \alpha_i + \alpha_j + \beta_j) + \sum_{k\in A} F_k(S')$$

$$\overset{or}{=} \sum_{k\in B} F_k(S') + (t_B + \alpha_i + \beta_i) + (t_B + \alpha_i + \beta_i + \beta_j) + \sum_{k\in A} F_k(S')$$

$$\sum_{k=1}^{n} F_k(S) - \sum_{k=1}^{n} F_k(S') \geq 0$$

In other words, the interchange of jobs i and j reduces the value of $\overline{F}$. Therefore, job i precedes job j. This completes the proof. □

IV. Branch-and-Bound and Heuristic Algorithm

4.1 Branching rule

Since the jobs should be processed in the same order on all machines, the proposed problem can be considered as finding an optimum from among n! possible permutation sequences. Therefore, a forward branching tree can be used, where each node corresponds to a sub-problem which is defined by a partial sequence of a subset of jobs that are to be placed at the front of the whole sequence. In each branching stage, a sub-problem (i.e., node) is selected and partitioned into one or more sub-problems that are defined by attaching one more job associated with the sub-problem being partitioned. To select a node for branching, a depth first rule is adapted in the proposed algorithm. In the depth first rule, a node with the most jobs in the corresponding partial sequence is selected for branching. In case of ties, the branching rule selects a node with the minimum job index number among them. In the branching a node list will be kept, which is ranked by the depth levels and lower bounds for active nodes, and the first node in the list will be selected for branching.

4.2 Initial upper bound (Heuristic Procedures)
It is known that the SPT rule minimizes the mean flow time in single machine problem. Thus, the SPT rule is going to be adapted to derive a lower bound. The average of total processing time is used as single machine problem measure.
In order to save the search effort in the proposed branch-and-bound algorithm, branching will start with an upper bound corresponding to a feasible solution value which is found prior to the implementation of the *branch-and-bound algorithm* by a separate heuristic approach. In this regard, a SPT-based with total processing-time heuristic mechanism is considered.

3) Property3
The second operation of a job i (i=1,2,...,n) is assigned to the earlier available machine among the machines.
Proof)
Denote by C_{iA} the completion time of the first operation of a job i. Denote by C_{iB} and C_{iC} in a similar way. The starting time of the J_{iB} can be C_{iA} in the machine 1 or $C_{(i-1)C}$ in the first machine 2 so that C_{iB} is equal to $C_{iA} + P_{iB} + P_{iC}$ or $C_{(i-1)C} + P_{iB} + P_{iC}$.
If $C_{iA} < C_{(i-1)C}$ holds, then $C_{iA} + P_{iB} + P_{iC} < C_{(i-1)C} + P_{iB} + P_{iC}$ holds. Therefore, a assignment of the J_{iB} on machine 1 reduces the completion time of a job i as much as $C_{(i-1)C} - C_{iA}$.
The case of $C_{iA} > C_{(i-1)C}$ can be proved in a similar way. This completes the proof. □

4) Property 4
If the both machines are available at once, the second operation of a job i (i=1,2,...,n) is assigned to M2.
Proof)
Consider a sequence S that the second operation of a job i (i=1,2,...,n) is assigned to M1 and a new sequence S' that the second operation of a job i (i=1,2,...,n) is assigned to M2. The situation is depicted in Figure 1, where B denotes the set of jobs that precede job i and A denotes the set of jobs that follow i. $C_k(S1)$ represents the flowtime of job k in M1 under schedule S.
In the both of sequence S and S', the completion time of a job i is not changed but the completion time of a job ($i+1$) is different. Since the starting time of job ($i+1$) is different. Under the sequence S, the starting time of job ($i+1$) is $\sum_{k\in B} C_k(S1) + P_{iA} + P_{iB}$. On the other hand, the starting time of job ($i+1$) is $\sum_{k\in B} C_k(S1) + P_{iA}$ in the sequence S'.
Therefore, the assignment of the second operation of a job i (i=1,2,...,n) reduces the flowtime. This completes the proof. □

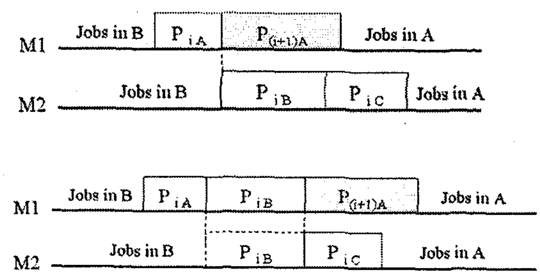

Fig. 1. Assignment of the second operation

4.2 Lower bound
In order to derive the lower bound, it is assumed that a slitting of every operation is possible. This assumption means every operation can be processed concurrently in both of machines. But still preemption is not allowed.
In optimal schedule, J_{jC} can be started when the $J_{(j-1)C}$ is completed or the J_{jB} is completed. But under this assumption, J_{jC} can be started when the half of sum of the completion time of $J_{(j-1)C}$ and the completion time of J_{jB}. So the assumption reduces the optimal mean flow time definitely.
The sequence follows SPT rule with the average of total processing time. Because the SPT rule make the *shortest mean flow time under the assumption.* Therefore, it is can be a lower bound of the proposed problem.

5) Property 5
A SPT sequence with slitting operation provides a lower bound.

$$LB = n \times (\frac{P_{1A}}{2} + \frac{P_{1B}}{2}) + \sum_{j=2}^{n}[(n-j+1)(\frac{P_{(j-1)C}}{2} + \frac{P_{jA}}{2} + \frac{P_{jB}}{2})] + \sum_{j=1}^{n} P_{jC}$$

Proof)
S* means the optimal sequence and $SPT_{Ave.}$ means SPT sequence with the average of total processing time.

$$\begin{aligned}
opt. &= P_{1A}(S^*) + P_{1B}(S^*) + P_{1C}(S^*) \\
&+ \sum_{j=2}^{n}[\max(C_{(j-1)C}(S^*), C_{jA}(S^*) + P_{jB}(S^*)) + P_{jC}(S^*)] \\
&\geq \frac{P_{1A}(S^*)}{2} + \frac{P_{1B}(S^*)}{2} + P_{1C}(S^*) \\
&+ \sum_{j=2}^{n}[\frac{C_{(j-1)C}(S^*) + C_{jA}(S^*) + P_{jB}(S^*)}{2} + P_{jC}(S^*)] \\
&\geq \frac{P_{1A}(SPT_{Ave.})}{2} + \frac{P_{1B}(SPT_{Ave.})}{2} + P_{1C}(SPT_{Ave.}) \\
&+ \sum_{j=2}^{n}[\frac{C_{(j-1)C}(SPT_{Ave.}) + C_{jA}(SPT_{Ave.}) + P_{jB}(SPT_{Ave.})}{2} + P_{jC}(SPT_{Ave.})] \\
&= LowerBound
\end{aligned}$$

The expression of the lower bound with not considering the sequence is as follow. And it has the minimal value of with $\overline{F}$ $SPT_{Ave.}$ sequence

$$\frac{P_{1A}}{2}+\frac{P_{1B}}{2}+P_{1C}+\sum_{j=2}^{n}[\frac{C_{(j-1)C}+C_{jA}+P_{jB}}{2}+P_{jC}]$$
$$=n\times[\frac{P_{1A}}{2}+\frac{P_{1B}}{2}]+\sum_{j=2}^{n}[(n-j+1)\frac{P_{(j-1)C}+P_{jA}+P_{jB}}{2}]+\sum_{j=1}^{n}P_{jC}$$
$$=n\times[\frac{P_{1A}}{2}+\frac{P_{1B}}{2}]$$
$$+(n-1)\times\frac{P_{1C}}{2}+(n-1)\times[\frac{P_{2A}+P_{2B}}{2}]$$
$$+(n-2)\times\frac{P_{2C}}{2}+(n-2)\times[\frac{P_{3A}+P_{3B}}{2}]$$
$$+(n-3)\times\frac{P_{3C}}{2}+\cdots+$$
$$+1\times[\frac{P_{nA}+P_{nB}}{2}]+0\times\frac{P_{nC}}{2}+\sum_{j=1}^{n}P_{jC}$$

In order to show that this expression has the minimal value with $SPT_{Ave.}$ sequence, a pairwise interchange method is used. Consider a sequence S that is not the $SPT_{Ave.}$ sequence. That is, somewhere in S there must exist a pair of adjacent jobs, *i* and *j*, with *j* following *i*, such that $P_{iA}+P_{iB}+P_{iC}>P_{jA}+P_{jB}+P_{jC}$. Now, construct a new sequence, S', in which job *i* and *j* are interchanged in sequence. $F_k(S)$ represents the flowtime of job *k* under schedule S.

$$\sum_{j=1}^{n}F_k(S)=n\times(\frac{P_{1A}}{2}+\frac{P_{1B}}{2})+(n-1)\times\frac{P_{1C}}{2}+(n-1)\times(\frac{P_{2A}}{2}+\frac{P_{2B}}{2})+\cdots$$
$$+(n-i+1)\times(\frac{P_{jA}}{2}+\frac{P_{jB}}{2})+(n-i)\times\frac{P_{jC}}{2}$$
$$+(n-j+1)\times(\frac{P_{iA}}{2}+\frac{P_{iB}}{2})+(n-j)\times\frac{P_{iC}}{2}+\cdots$$
$$+1\times(\frac{P_{nA}}{2}+\frac{P_{nB}}{2})+0\times\frac{P_{nC}}{2}+\sum_{j=1}^{n}P_{jC}$$
$$\sum_{j=1}^{n}F_k(S')=n\times(\frac{P_{1A}}{2}+\frac{P_{1B}}{2})+(n-1)\times\frac{P_{1C}}{2}+(n-1)\times(\frac{P_{2A}}{2}+\frac{P_{2B}}{2})+\cdots$$
$$+(n-i+1)\times(\frac{P_{iA}}{2}+\frac{P_{iB}}{2})+(n-i)\times\frac{P_{iC}}{2}$$
$$+(n-j+1)\times(\frac{P_{jA}}{2}+\frac{P_{jB}}{2})+(n-j)\times\frac{P_{jC}}{2}+\cdots$$
$$+1\times(\frac{P_{nA}}{2}+\frac{P_{nB}}{2})+0\times\frac{P_{nC}}{2}+\sum_{j=1}^{n}P_{jC}$$

$$\sum_{j=1}^{n}F_k(S)-\sum_{j=1}^{n}F_k(S')=(P_{iA}+P_{iB}+P_{iC})-(P_{jA}+P_{jB}+P_{jC})>0$$

In other words, the interchange of jobs *i* and *j* reduces the value of $\overline{F}$. Therefore any sequence that is not the $SPT_{Ave.}$ sequence can be improved with respect to $\overline{F}$ by such an interchange of an adjacent pair of jobs. It follows that the expression has the minimal value with $SPT_{Ave.}$ sequence.

V. Concluding Remarks

*This thesis considers three-operation jobs in a two-*machine flowshop scheduling problem to minimize the mean flowtime. In the proposed problem, each job has three operations where the second operation is flexible in that it can be assigned to either the first or the second machine. In the analysis, some solution properties are characterized to derive a solution bound, which are used to derive a SPT-based with total processing-time heuristic algorithm and a depth-frist branch-and-bound algorithm. The effectiveness and efficiency of the derived algorithms are evaluated through several computational experiments which are performed with various numerical instances.

The results of this thesis may immediately be applied to many corresponding production/service systems for improving the associated logistic service quality.

As a further study, more than 2 machines may be considered in a multi-stage flowshop problem.