

Two-machine Shop Floor Scheduling Problem with Multi-Predecessor Constraints*

Xiandong Zhang

Department of Management Science, School of Management, Fudan University,
Shanghai 200433, China, xiandongzhang@fudan.edu.cn

Jatinder N.D. Gupta

College of Administrative Science, The University of Alabama in Huntsville,
Huntsville, AL 35899, USA, guptaj@uah.edu

This paper deals with a two-machine shop floor scheduling problem with multi-predecessor constraints. The problem is proved to be NP-hard in strong sense. The paper proves that under some additional constraints the Algorithm ISPT (Instant Shortest Processing Time first) results an optimal solution. A branch and bound algorithm is developed for the general case and some dominant rules are provided.

Keywords: Shop Floor Scheduling, Multi-predecessor Constraints, Complexity of Scheduling Problems, Branch and bound algorithm

1. Introduction

This paper addresses a two-machine shop floor scheduling problem with multi-predecessor constraints. We are given two machines A and B , and a set $J = \{J_1, J_2, \dots, J_n\}$ of jobs. Each job $J_j \in J$ must be first processed on machine A and then on machine B , therefore it is a kind of flow shop. The operation times are p_{1j} and p_{2j} , respectively, where $p_{1j} + p_{2j} > 0$. Neither of the machines can process more than one job at a time. The objective is to find a schedule that minimizes the makespan C_{max} , which is the completion time of all jobs on both machines. This original problem is denoted by $F2||C_{max}$.

The $F2||C_{max}$ problem can be solved by Johnson's rule very conveniently. But considering multi-predecessor constraints, the problem may be difficult. The multi-predecessor constraints means job J_j may have multiple predecessors, and all the predecessors should be finished first in machine A before job J_j could be started in machine B , i.e. the operation of p_{2j} in machine B could not be started until the operations of p_{1j} and p_{1k} , $J_k \in J^k$, in machine A are finished, where J^k is the set of jobs that precede job J_j .

We write $J_k \xrightarrow{m} J_j$ and say that job J_k is one of the multi-predecessors of job J_j . Job J_j may have more than one predecessors. This type of constraint is defined as *multi-predecessor constraint*. The problem discussed in this paper is denoted by $F2|multi\text{-}predecessor|C_{max}$. Since a numerical example to illustrate where a permutation schedule is not optimal can be easily constructed, we need to find an optimal non-permutation schedule for the problem.

* Supported by China National Nature Science Foundation. Project No.: 70432001.

Traditionally, there are two types of precedence constraints [1]. For two jobs J_j and J_k , we write $J_j \xrightarrow{1} J_k$ and say that job J_j precedes job J_k if and only if job J_k cannot start on any machine until job J_j is completed on all machines it has to be processed on. The relation $\xrightarrow{1}$ will be called the *precedence relation of the first type*. On the other hand, for two jobs J_j and J_k , we write $J_j \xrightarrow{2} J_k$ and say that job J_j precedes job J_k if and only if job J_k cannot start on any machine until job J_j is completed on the same machine. The relation $\xrightarrow{2}$ will be called the *precedence relation of the second type*.

We write “*prec1*” or “*prec2*” in the second position of the standard notation for scheduling problems to indicate that there are precedence constraints either of the first or of the second, respectively. The $F2|prec1|C_{max}$ problem is trivial, and the optimal makespan is the sum of all the operations time. The $F2|prec2|C_{max}$ problem is NP-hard in the strong sense due to Strusevich [1] and Monma [2].

If $J_j \xrightarrow{2} J_k$, the operation of p_{1j} will precede the operation of p_{1k} and operation of p_{2j} will precede the operation of p_{2k} . Therefore, $J_j \xrightarrow{2} J_k$ implies $J_j \xrightarrow{m} J_k$, but the opposite is not true. The multi-predecessor condition is a kind of relaxation of *precedence relation of the second type*.

2. The Complexity of $F2|multi\text{-}predecessor|C_{max}$

In this part, we will prove that $F2|multi\text{-}predecessor|C_{max}$ is strongly NP-hard. The proof is done by reducing $F2|prec2|C_{max}$ to $F2|multi\text{-}predecessor|C_{max}$. We first show that an instance of $F2|prec2|C_{max}$ could be transformed to an instance of $F2|multi\text{-}predecessor|C_{max}$. Then we prove that an optimal schedule of the instance of $F2|multi\text{-}predecessor|C_{max}$ can be used to construct an optimal schedule of the instance of $F2|prec2|C_{max}$.

First, we need the following definition. For the $F2|prec2|C_{max}$ problem, if $J_i \xrightarrow{2} J_j$ and $J_j \xrightarrow{2} J_k$, then $J_i \xrightarrow{2} J_k$. If $J_j \xrightarrow{2} J_k$ and there is no job J_i such that $J_j \xrightarrow{2} J_i \xrightarrow{2} J_k$, then job J_j is said to *directly precede* job J_k , and this is denoted by $J_j \xrightarrow{d2} J_k$.

Then, we use the following algorithm to transform an instance of $F2|prec2|C_{max}$ to an instance of $F2|multi\text{-}predecessor|C_{max}$.

Algorithm T

Step 1 List all the job pairs with the *precedence relation of the second type*, including *directly preceded* job pairs and *indirectly preceded* job pairs in the instance of $F2|prec2|C_{max}$.

Step 2 Change all the *direct and indirect precedence relations of the second type* to the *multi-predecessor constraints*.

Because there are less than n^2 job pairs with precedence relations of the second type, the *algorithm T* is obviously polynomial. We call the original instance as the problem P and the transformed instance as the problem P' . We claim that a feasible schedule S' of problem P' can be modified to a feasible schedule S of problem P without increasing the makespan. The

following *algorithm M* describes the details of the modification. In the *algorithm M*, we will first change the order of operations in machine *B*, and then change the order of operations in machine *A*.

Algorithm M

(Part I. Change the order of operations in machine *B*.)

Step 1 Let $V = \emptyset$. Define a jobs set U which contains jobs without successors in *precedence relations of the second type*. For each job J_k in set U , find all the direct predecessors of the second type $J_j \square J$. If there is a contradiction in machine *B* that operation p_{2k} is scheduled before p_{2j} , then relocate operation p_{2j} just before p_{2k} .

Step 2 Let $V = V \cup U$. If $V = J$, go to Step 4. Otherwise, redefine the jobs set U which contains jobs having direct successors in set V . For each job J_k in set U , find all the direct predecessors of the second type $J_j \square J$. If there is a contradiction in machine *B* that operation p_{2k} is scheduled before p_{2j} , then relocate operation p_{2j} just before p_{2k} .

Step 3 Go to Step 2.

(Part II. Change the order of operations in machine *A*.)

Step 4 Let $N = \emptyset$. Define a jobs set M which contains jobs without predecessors in *precedence relations of the second type*. For each job J_j in set M , find all the direct successors of the second type $J_k \square J$. If there is a contradiction in machine *A* that operation p_{1k} is scheduled before p_{1j} , then relocate operation p_{1k} just after p_{1j} .

Step 5 Let $N = N \cup M$. If $N = J$, stop. Otherwise, redefine the jobs set M which contains jobs having direct predecessors in set N . For each job J_j in set M , find all the direct successors of the second type $J_k \square J$. If there is a contradiction in machine *A* that operation p_{1k} is scheduled before p_{1j} , then relocate operation p_{1k} just after p_{1j} .

Step 6 Go to Step 4.

Because there are at most n^2 job pairs with precedence relations of the second type, the *algorithm M* is also polynomial.

Theorem 2.1 *the F2|multi-predecessor|C_{max} problem is NP-hard in strong sense.*

Proof. The proof is done by reducing F2|prec2|C_{max} to F2|multi-predecessor|C_{max}. According to *Algorithm T*, an instance of F2|prec2|C_{max} could be transferred to an instance of F2|multi-predecessor|C_{max}. If an optimal schedule S^* could be found for the transformed instance of F2|multi-predecessor|C_{max}, we can use *Algorithm M* to change the schedule to a feasible schedule S^* for the original instance of F2|prec2|C_{max}. It is easy to verify that the modification in *Algorithm M* will not increase the makespan. Since F2|multi-predecessor|C_{max} contains less constraints than F2|prec2|C_{max}, the optimal makespan of problem P' is less than or equal to the optimal makespan of problem P . Therefore the modified schedule S^* of problem P' is an optimal schedule of problem P . With the strongly NP-hard property of F2|prec2|C_{max}, we know that the F2|multi-predecessor|C_{max} problem is strongly NP-hard. \square

3. A branch and bound approach

Since the problem of F2|multi-predecessor|C_{max} is NP-hard in strong sense, we will provide a

branch and bound algorithm in this paper. Before we describe the algorithm, some dominant rules will be developed.

Define Φ_j as an operations set which contains operations in machine A that have to be processed before the operation of p_{2j} . The property of set Φ_j can be described as following.

- i) $\Phi_j \neq \emptyset$, because p_{1j} is one of the elements;
- ii) Φ_j and Φ_k have intersection if an operation of p_{1s} precedes both the operations of p_{2j} and p_{2k} .

Define φ_j as a subset of Φ_j . φ_j can be the empty set \emptyset or the whole set Φ_j .

Lemma 3.1 Consider an optimal sequence in machine B . Without loss of generality, we change the subscripts of operations in the optimal sequence such that the sequence is p_{21}, \dots, p_{2n} . Then the optimal sequence in machine A is in the order of $\varphi_1, \varphi_2, \dots, \varphi_n$.

Proof. The proof is done by induction. First, let $j = 1$. Consider operation p_{21} . Since all the operations in Φ_1 have to be finished before p_{21} , we can postpone all the operations not in Φ_1 until all the operations in Φ_1 are finished. Obviously, this will not increase the makespan. Therefore, φ_1 is equal to Φ_1 . Then, let's consider operation p_{2j} . Take the order of $\varphi_1, \varphi_2, \dots, \varphi_{j-1}$ as fixed in machine A . Since all the operations in Φ_j have to be finished before p_{2j} , and some operations in Φ_j have been finished, we can postpone all the rest operations not in Φ_j until all the operations in Φ_j are finished. This will not increase the makespan. \square

From above lemma, we can have a corresponding sets of $\varphi_1, \varphi_2, \dots, \varphi_n$ in machine A , if we have a operations sequence s in machine $B, p_{21}, \dots, p_{2j}, \dots, p_{2n}$.

Define $\Phi_j(k)$ as a subset of Φ_j . $\Phi_j(k)$ contains the all the operations in Φ_j which are not included in $\varphi_1, \varphi_2, \dots, \varphi_{k-1}$. That is to say, if we put job j in the k th position in the sequence, $\Phi_j(k)$ contains all the unprocessed operations in machine A which precede p_{2j} . Clearly, we have $\Phi_j(1) = \Phi_j$.

Define A_j as the total operation time of Φ_j . We have

$$A_j = \sum_{p_{1i} \in \Phi_j} p_{1i}$$

Similarly define $A_j(k)$ as the total operation time of $\Phi_j(k)$.

$$A_j(k) = \sum_{p_{1i} \in \Phi_j(k)} p_{1i}$$

Clearly, $A_j(1) = A_j$.

If $A_j \leq p_{2j}$ for each job J_j in set J , we can form a sequence with the following *Instant Shortest Process Time (ISPT)* algorithm.

Algorithm ISPT

Step 1 Let $k=1$. Find a job with the smallest A_j in machine A. Ties may be broke arbitrarily. Assign this job as the first job in sequence s .

Step 2 Let $k=k+1$. Find the job J_j with the smallest $A_j(k)$. Ties may be broke arbitrarily. Assign job j as the k th job in sequence s .

Step 3 if $k=n$, Stop; else go to **Step2**.

Theorem 3.1 *The sequence s generated from Algorithm ISPT is an optimal sequence for $F2|multi\text{-}predecessor|C_{max}$ if $A_j \leq p_{2j}$ for each job J_j in set J .*

Proof. From Lemma 3.1, we know that the operations sequence in machine A is decided by the operations sequence in machine B , and it is easy to verify that the operations sequences within $\phi_j, j=1, \dots, n$, have no effect on the makespan.

The further proof is by contradiction. Suppose a schedule with contradiction is optimal. In this optimal schedule, there must be a pair of adjacent jobs, say job J_j followed by job J_k , that satisfy the condition of $A_j(l) > A_k(l)$, l is job J_j 's position in the sequence.

It is suffices to show that the makespan is reduced after a pairwise interchange of jobs J_j and J_k in machine B .

Let Θ_{jk} be the intersection of $\Phi_j(l)$ and $\Phi_k(l)$. Because operations in Θ_{jk} are the predecessors for both operations of p_{2j} and p_{2k} , these operations can be finished firstly no matter the order of jobs j and k . Let $\Phi'_j(l)$ and $\Phi'_k(l)$ be the subsets of $\Phi_j(l)$ and $\Phi_k(l)$ without Θ_{jk} , respectively. Let $A'_j(l)$ and $A'_k(l)$ be the operation time of $\Phi'_j(l)$ and $\Phi'_k(l)$. Clearly, we have $A'_j(l) > A'_k(l)$. Since $\Phi'_j(l)$ and $\Phi'_k(l)$ have no intersections, and the operations in $\Phi'_j(l)$ have to be finished before p_{2j} and the operations in $\Phi'_k(l)$ have to be finished before p_{2k} , the shortest makespan for these two jobs can be achieved easily by Johnson's rule. If $A'_j(l) > A'_k(l)$, $A'_j(l) \leq p_{2j}$, and $A'_k(l) \leq p_{2k}$, the sequence of job J_k followed by job J_j will ensure the shortest finish time in machine B for jobs J_j and J_k . \square

Corollary 3.1 *In the optimal sequence of a problem of $F2|multi\text{-}predecessor|C_{max}$, the order of jobs which have $A_j \leq p_{2j}$, follows the ISPT rule.*

Proof. The proof is by contradiction. Suppose another type of schedule is optimal. In this optimal schedule, there must be a pair of jobs, say job J_j followed (may not be immediately followed) by job J_k , that satisfy the condition of $A_j(p) > A_k(p)$, which have $A_j(p) \leq p_{2j}$, $A_k(p) \leq p_{2k}$, p is job J_j 's position in the sequence. This can be illustrated by Fig. 3.1.

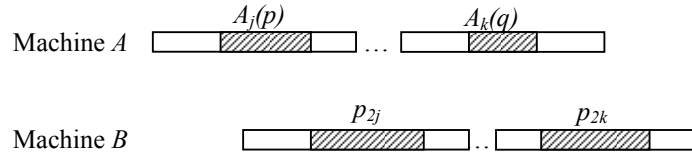


Fig. 3.1 Illustration for Corollary 3.1

It is easy to verify that if we relocate job J_k to the front of job J_j in machine B , and make the corresponding change in machine A based on *Lemma 3.1*, the total makespan will not be increased. \square

Corollary 3.2 *In the optimal sequence of a problem of $F2|multi\text{-}predecessor|C_{max}$, the order of jobs which have condition of $p_{1j} > p_{2j}$ and have no successors will follow the LPT rule in machine B , and all these kind of jobs will be sequenced after the jobs which have $A_j \leq p_{2j}$.*

The proof is by contradiction and similar to the proofs in Theorem 3.1 and Corollary 3.1. \square

The above two corollaries will help us in finding a better branching strategy. To find the bound for the original problem, we need following Lemma.

Lemma 3.2 *The Johnson's rule is the optimal solution for the problem of $F2||C_{max}$.* \square

A branch and bound procedure for $F2|multi\text{-}predecessor|C_{max}$ can be constructed as follows. First, based on *Lemma 3.1*, the sequence in machine A is decided by the sequence in machine B . So, we only need to develop the sequence in machine B . The branching operation may be based on the fact that schedules are developed starting from the beginning of the schedule. There is a single node at level 0 that is the top of the branching tree. At this node, none of the jobs has been put into any position in the sequence.

The branching step In branching level k ($k \geq 1$), there are $k-1$ jobs that have been scheduled. Define \mathcal{J} as the set of jobs that are not scheduled, \mathcal{J}^d as the subset of \mathcal{J} which contains jobs of $A_j(1) \leq p_{2j}$, \mathcal{J}^e as another subset of \mathcal{J} which contains jobs of $p_{1j} > p_{2j}$ and have no successors, and \mathcal{J}^b as the complementary set of \mathcal{J}^d and \mathcal{J}^e in set \mathcal{J} . If $\mathcal{J}^d \neq \emptyset$, find the job J_p in \mathcal{J}^d with the smallest $A_p(k)$, and add job J_p to \mathcal{J}^b . If $\mathcal{J}^d = \emptyset$, and $\mathcal{J}^e \neq \emptyset$, find the job J_q in \mathcal{J}^e with the largest p_{2q} and add job J_q to \mathcal{J}^b . Every job in \mathcal{J}^b can be put in the position k in the sequence. The number of jobs in \mathcal{J}^b is the number of branches in level k . If $\mathcal{J}^b = \emptyset$, we already get a complete sequence; otherwise, we get a partial sequence.

The bounding step Based on each newly generated partial sequence, consecutively append all unscheduled jobs into the partial sequence in the order of *Johnson's rule*, and compute their completion times without considering their precedence constraints. The completion time at machine B for the latest appended job is the lower bound for the makespan of this partial sequence. In level k , the partial sequence with the lowest lower bound will be selected as the next branching node.

The fathoming step At the beginning of the algorithm, the current optimal makespan Z^* is set at infinity, that is, $Z^* = \infty$. A partial sequence will be fathomed if either

- (i) Its lower bound is not smaller than Z^* , or
- (ii) It is a complete sequence.

If a partial sequence is fathomed by (ii), update Z^* with its associated lower bound whenever the latter has a smaller value.

Recursively implementing the branch-and-bound algorithm above will generate an optimal job sequence.

4. Conclusion

In this study, a two-machine shop floor scheduling problem multi-predecessor constraints is studied. In a manufacturing environment, the first machine may be a production stage for different kinds of materials, and the second machine may be the assembly stage using the materials from stage 1. This paper describes the *multi-predecessor constraint*, and proves that the complexity of $F2|multi\text{-}predecessor|C_{max}$ is NP-hard in strong sense. A branch and bound algorithm is put forward to solve the problem. Some dominant rules are established to eliminate branches and improve efficiency.

Concerning jobs with *multi-predecessor constraints*, several issues are worthy of future research. The development of algorithms for the two-machine shop floor with other performance measures can be investigated. Polynomial algorithms for special types of problems can be developed. More dominant rules can be found for the branch and bound algorithm. Application of our results to more complex manufacturing environments, such as the multiple-stage shop floor problem, is another area of research.

References

- [1] V. A. Strusevich (1997), Shop scheduling problems under precedence constraints, *Annals of Operations Research* **69**, 351 – 377.
- [2] C. L. Monma (1979), The two-machine maximum flow-time problem with series-parallel precedence constraints: An algorithm and extension, *Operations Research* **27**, 792-798
- [3] Michael Pinedo (2002), *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, Second Edition.
- [4] A.M.A Hariri and C.N. Potts (1984), Algorithms for two-machine flow shop sequencing with precedence constraints, *European Journal of Operational Research* **17**(2), 238 – 248.
- [5] Graham B. McMahon and Chong-John Lim(1993), The two-machine flow shop problem with arbitrary precedence relations, *European Journal of Operational Research* **64**(2), 249 – 257.
- [6] Bongjin Gim, Guy L. Curry and Bryan L. Deuermeier (1994), Two-machine flow-shop sequencing with sparse precedence constraints, *Computers & Industrial Engineering* **26**(1), 173 – 180.

- [7] Chong John Lim and Graham B. McMahon(1994), The three-machine flow-shop problem with arbitrary precedence relations, *European Journal of Operational Research* **78**(2), 216 – 223.
- [8] Alain Guinet and Marie Legrand(1998), Reduction of job-shop problems to flow-shop problems with precedence constraints, *European Journal of Operational Research* **109**(1), 96 – 110.
- [9] Igor Averbakh, Oded Berman and Ilya Chernykh(2005), The m-machine flowshop problem with unit-time operations and intree precedence constraints, *Operations Research Letters* **33**(3), 263 – 266.
- [10] Johann Hurink and Sigrid Knust (2001), List scheduling in a parallel machine environment with precedence constraints and setup times, *Operations Research Letters* **29**(5), 231 – 239.