



Novel properties along with solution methods for permutation flowshop scheduling

M. Aminnayeri^a and B. Naderi^{b,*}

a. Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran.

b. Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran.

Received 10 December 2013; received in revised form 1 June 2015; accepted 26 September 2015

KEYWORDS

Scheduling;
 Flowshops;
 Solution methods;
 Pairwise job-ordering;
 Makespan.

Abstract. This paper deals with permutation flow shop scheduling to minimize makespan. Some novel, useful definitions and properties are established. Then, the paper proposes novel solution methods. Particular attention is paid to algorithms based on the orderings of pairs of jobs. The first algorithm is for three-machine problems; it gives an optimal solution in a certain strict sense in the case of the ordering of two jobs. Moreover, the paper extends the three-machine Johnson's rule to the general case of m -machine problems.

© 2016 Sharif University of Technology. All rights reserved.

1. Introduction

In scheduling, there is a set of n jobs that need to be processed by a set of m machines [1]. The problem is how to schedule jobs on machines so that certain objectives, which are mainly time-oriented, are achieved. A Flow-shop Scheduling Problem (FSP) is a certain type in which all of the n jobs follow essentially the same progression from one machine to the next. The machines are numbered in accordance with their appearance in this progression [2].

In FPS, a schedule S is the sequence of n jobs on m machines. A schedule of FPS in which the sequences of all the jobs are the same on all the machines is called a permutation schedule, π . The name “*permutation schedule*” is derived from the fact that, with the n jobs serially labeled, the set of $n!$ permutation of the first n positive integers corresponds to the set of possible permutation schedules.

The completion time of job j , denoted by C_j , is the time at which the process of job j on the last machine is accomplished. The notation $C_j(S)$ or $C_j(\pi)$

is used to denote the completion of job j for a given schedule, S , or permutation schedule, π . Objective functions are often the functions of the $C_j(S)$ or $C_j(\pi)$. This paper is concerned with such a particular objective function, namely, makespan:

$$C_{\max}(S) = \max_j C_j(S).$$

An optimal schedule, S^* , is one such that:

$$C_{\max}(S^*) = \min_S C_{\max}(S).$$

An optimal permutation schedule, π^* , is one such that:

$$C_{\max}(\pi^*) = \min_{\pi} C_{\max}(\pi).$$

It is shown by Conway [3] that when the number of machines is 2 or 3, then S^* and π^* , the optimal schedule and optimal permutation schedule, respectively, are such that:

$$C_{\max}(S^*) = C_{\max}(\pi^*).$$

In other words, in the case of $m = 2$ or 3, an optimal permutation schedule is also an optimal general schedule. In view of this, one may restrict attention to the permutation schedules when $m = 2$

*. Corresponding author. Tel.: +98 26 34551022
 E-mail addresses: mjnayeri@aut.ac.ir (M. Aminnayeri);
 bahman.naderi@aut.ac.ir (B. Naderi)

or 3. In fact, very often, the studies are restricted to permutation schedules also in $m > 3$. This is because the problem of finding S^* is unwieldy. Unless otherwise specified, the term “schedule” is hereinafter used for both a general schedule S and a permutation schedule π .

In recognition of the prototype of such an ordering algorithm proposed by Johnson [4], this paper pays particular attention to the algorithms based on the ordering of pairs of jobs in a permutation PFS. This paper calls such ordering algorithms “*pairwise J-ordering*”.

First, motivated by Johnson’s work, the two abstract properties of “transitivity” and “job-adjunction-robustness” are defined as sufficient for a pairwise J -ordering algorithm to lead to an optimum permutation schedule. Since transitivity are not often realized in practice, result on partial optimality is discussed, when pairwise J -ordering algorithms fail to be transitive. Next, a pairwise J -ordering algorithm for 3-machine permutation FSP is proposed. It is shown that the algorithm coincides with the standard 3-machine adaptation of Johnson’s ordering [4,5]. Finally, this paper extends the 3-machine adaptation of Johnson’s ordering to the general case of m -machine permutation FSP and determines m situations in which the algorithm reaches the optimal permutation schedule.

The rest of the paper is organized as follows. Section 2 is devoted to literature review. Section 3 introduces some definitions and properties. Section 4 proposes pairwise J -ordering algorithms. Section 5 concludes the paper and gives some interesting future research directions.

2. Literature review

For decades, scheduling has been an active research field. Consequently, one can dare to say that the field of scheduling theory is a very large one. Even though it has seen tremendous research efforts, a great many problems still remain unsolved. Johnson’s well-known work [4] is one of the pioneering contributions of the field, and is concerned with scheduling n jobs in a 2-machine flow shops. He also introduces a related algorithm covering special cases in 3-machine flow shops.

From initial papers, one might refer readers to the following papers. Giglio and Wagner [6] analyze the 3-machine scheduling problem empirically by integer programming, linear programming, Monte Carlo, and Johnson’s approach. None of the techniques tested prove to be superior to the others. Dudek and Teuten [7] attempt a general algorithm for the m -machine flowshop for which Karush [8] exhibits a counter example. Smith and Dudek [9] try to correct the algorithm, but the resulting algorithm is inefficient [10]. A comparative study of flow shop

algorithms, reminiscent of the earlier empirical study of Giglio and Wagner [6], is made by Baker [11]. Using a set of test problems, Baker [11] investigates various branch-and-bound and elimination strategies, and combines these in a new efficient algorithm. Dannenbring [12] continues in the tradition set by Giglio and Wagner [6] and Baker [11], and presents a computational experience with eleven heuristics flow shop algorithms.

Smith et al. [13] propose a simple and efficient algorithm for certain special cases of the m -machine flow shop problem. Three special cases of flow shop problems are solved by Szwarc [14] based on the critical path concept [3]. Burns and Rooker [5] relax the condition given by Johnson [4] for solving 3-machine problems and also provide a further condition under which Johnson’s 2-machine flow shop algorithm can be adapted to deriving an optimum permutation schedule for the 3-machine flow shop.

Another research direction is the presentation of approximation algorithms, where they mainly focus on general m -machine cases. Palmer [15] presents a heuristic which assigns an index to every job, and then produces a sequence after sorting the jobs according to the calculated index. Campbell et al. [16] develop another heuristic which is basically an extension of Johnson’s algorithm to the m -machine case. The heuristic constructs $m-1$ schedules by grouping the m original machines into two virtual machines and solving the resulting two-machine problems by repeatedly using Johnson’s rule. Nawaz et al. [17] introduce another heuristic, namely NEH, which is still claimed to be superior to any other available heuristics [18]. The NEH, first, sorts jobs according to their total processing times. Then, it inserts jobs one by one into all possible positions among the previously scheduled jobs. The list of heuristics seems almost endless.

More recent papers have mainly focused on metaheuristics. Among the different algorithms, one can cite the hybrid backtracking search algorithm by Lin et al. [19], the hybrid artificial bee colony algorithm by Li and Pan [20], the variable neighborhood search By Moslehi and Khorasani [21], the discrete artificial bee colony algorithm by Ribas et al. [22], the improved cuckoo search algorithm by Marichelvam et al. [23], the genetic algorithm by Ruiz et al. [24], the ant colony by Lin et al. [25], the simulated annealing by Naderi et al. [26], the tabu search by Eksioğlu et al. [27], the particle swarm optimization by Jarboui et al. [28], the iterated greedy algorithm by Ruiz and Stützle [29], and so on.

3. Some novel definitions and properties

It is said that job i precedes job j in a permutation schedule, π , if job i appears before job j in the

sequence. This is denoted by $i \ll j$. Also, the processing time, including the setup time, of job i on machine 1 is denoted by A_i ; similarly, the processing time, including the setup time, of job I on machines 2 and 3 are denoted by B_i and C_i , respectively, and so on for subsequent machines.

This paper is concerned with techniques to determine optimum permutation schedules, π^* . An alternative method is to utilize an algorithm which tells how to order the jobs as a function of the parameters A_i , B_i , C_i , and so on. A subclass of such algorithms, of which Johnson's algorithm [4] is the outstanding example, consists of algorithms centered on the ordering of pairs of jobs as a function only of their own parameters. Such algorithms are called pairwise J -orderings. Often the simplicity of these algorithms depends on their assumptions on parameters A_i , B_i , and so on.

Definition 1. Pairwise J -ordering: Any ordering of two jobs can be thought of as based on a function $f(A_1, B_1, \dots, A_2, B_2, \dots)$ such that:

$$f(A_1, B_1, \dots, A_2, B_2, \dots) \leq f(A_2, B_2, \dots, A_1, B_1, \dots) \Leftrightarrow 1 \ll 2.$$

In a given parametric problem, for which all parameters of all n jobs are specified, the function f can be replaced by a simpler function g :

$$g(1, 2) = f(A_1, B_1, \dots, A_2, B_2, \dots).$$

The following example shows that it is not, in general, possible to get an optimal ordering of n jobs by means of optimal pairwise J -ordering.

Example 1. Consider an FSP with $n = 3$ and $m = 3$. Table 1 shows the processing time.

If we only consider the problem of finding the optimal ordering of jobs 2 and 3, it is observed that the permutation schedule $\pi' : (2, 3)$ is better than the permutation schedule $\pi'' : (3, 2)$, because $C_{\max}(\pi') = 44$, and $C_{\max}(\pi'') = 45$. On the other hand, we have:

$$\pi_1 : (1, 2, 3) \quad \text{where} \quad C_{\max}(\pi_1) = 59,$$

$$\pi_2 : (1, 3, 2) \quad \text{where} \quad C_{\max}(\pi_2) = 56,$$

$$\pi_3 : (2, 1, 3) \quad \text{where} \quad C_{\max}(\pi_3) = 59,$$

Table 1. The processing time of Example 1.

Job i	A_i	B_i	C_i
1	5	15	11
2	5	14	10
3	9	12	13

$$\pi_4 : (2, 3, 1) \quad \text{where} \quad C_{\max}(\pi_4) = 57,$$

$$\pi_5 : (3, 1, 2) \quad \text{where} \quad C_{\max}(\pi_5) = 60,$$

$$\pi_6 : (3, 2, 1) \quad \text{where} \quad C_{\max}(\pi_6) = 61,$$

therefore, $\pi_2 = \pi^*$. It is clear that the ordering of jobs 2 and 3 in the optimal permutation schedule π_2 violates their ordering in π' .

Definition 2. Schematic tables: Any pairwise J -ordering of n jobs can be portrayed by a "Schematic table" where the symbol \ll or \gg appears in the (i, j) th entry of the table in accordance with whether $i \ll j$ or $j \ll i$.

Figure 1 shows a general schematic table for a pairwise J -ordering.

Schematic tables of this type are also useful for displaying pairwise order relationships among the members of subsets of jobs. Revising Johnson's algorithm, it could be considered as a pairwise J -ordering with the following function:

$$g(i, j) = \min(A_i + B_i, B_j + C_j).$$

The schematic table of Example 1, using the above pairwise J -ordering, is shown in Figure 2.

Definition 3. Transitivity: A pairwise J -ordering is said to be transitive if $i_1 \ll i_2$ and $i_2 \ll i_3$ imply $i_1 \ll i_3$.

	i_1	i_2	i_3	\dots	i_{n-1}	i_n
i_1		\ll	\ll	\dots	\ll	\ll
i_2			\ll	\dots	\ll	\ll
i_3				\dots	\ll	\ll
				\dots	\dots	\dots
i_{n-1}						\ll
i_n						

Figure 1. A general schematic table for a pairwise J -ordering.

	1	2	3
1		\gg	\ll
2			\gg
3			

Figure 2. The schematic table of Example 1.

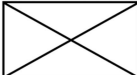
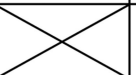
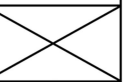
	2	1	3
2		\ll	\ll
1			\ll
3			

Figure 3. Rearranged schematic table under transitivity.

Under transitivity, it is always possible to arrange jobs in such a way that only the sign \ll appears in the schematic table. Rearrangement of the previous schematic table under transitivity, is shown in Figure 3.

Definition 4. Job-Adjunction-Robustness:

A pairwise J -ordering is said to be JAR (Job-Adjunction-Robustness) if the following is true: Let π be any ordering of n jobs such that at least one neighboring pair of jobs is ordered in violation of this pairwise J -ordering. Let π' be an ordering obtained from π by interchanging the positions of the jobs composing this violating neighboring pair. Then, $C_{\max}(\pi') \leq C_{\max}(\pi)$.

The JAR property and transitivity of a pairwise J -ordering are sufficient to insure that the algorithm which orders jobs in accordance with the pairwise J -ordering furnishes an optimal ordering π^* . This is verified in the following theorem.

Theorem 1. Let a pairwise J -ordering be both JAR and transitive, then an ordering, π^* , consistent with this pairwise J -ordering is optimal.

Proof. Suppose not. Without loss of generality, let π^* be $(1, 2, \dots, n)$ and let $\pi_0 : (i_1, i_2, \dots, i_n)$ be any optimal ordering where i_l is the job in l th position in optimal ordering. Let k be the first job in π^* for which we have $i_k \neq k$. Therefore, there is an i_l , where $l > k$, such that $k = i_l$. Clearly, i_l and i_{l-1} are not ordered as in π^* (i.e., i_l precedes i_{l-1}), and are neighbors. Interchanging i_l and i_{l-1} , we get a new ordering π_1 which has, using JAR property, the following relationship with π_0 :

$$C_{\max}(\pi_1) \leq C_{\max}(\pi_0).$$

We can continue the procedure of interchange of i_l with its left neighbors until it replaces i_k , and i_k moves one position to the right in the ordering π_{l-k} , and repeatedly. Moreover, using JAR property, π_{l-k} has the following relationship with π_0 :

$$C_{\max}(\pi_{l-k}) \leq C_{\max}(\pi_0).$$

If π_{l-k} is the same as π^* , then we are done. If not, let k' be the next job in π^* for which $i_{k'} \neq k'$. We can

treat π_{l-k} as we treated π_0 to get a further ordering in $\pi_{l'-k'}$ such that:

$$C_{\max}(\pi_{l'-k'}) \leq C_{\max}(\pi_{l-k}) \leq C_{\max}(\pi_0).$$

Continuing in this way for all the subsequent jobs, we finally reach π^* and find:

$$C_{\max}(\pi^*) \leq \dots \leq C_{\max}(\pi_{l-k}) \leq C_{\max}(\pi_0),$$

which shows that π^* is optimal, thus a contradiction. \square

Definition 5. Transitive skeins: If a pairwise J -ordering is given for ordering n jobs, then a subset of jobs for which the corresponding schematic table exhibits transitivity (i.e., a subset uniquely ordered by the pairwise J -ordering) is called a transitive skein for that pairwise J -ordering.

If the set of all n jobs involved in the problem forms a transitive skein for the pairwise J -ordering, then the pairwise J -ordering is, of course, transitive. In order to establish a relationship between optimal orderings and transitive skeins under a JAR-type property, it is natural to strengthen the latter and to state the following definition.

Definition 6. A Restricted JAR property: A pairwise J -ordering is said to have the strong JAR property if the following is held: In a given ordering, π , if we interchange the order of any two jobs which are ordered in accordance with the schematic table of the pairwise J -ordering to get a new ordering π' , then we have:

$$C_{\max}(\pi') \leq C_{\max}(\pi).$$

The following theorem deduces properties of optimal orderings under the strong JAR property, when transitivity does not necessarily obtain.

Theorem 2. When a pairwise J -ordering has the strong JAR property, then all transitive skeins appear as transitively ordered subsets in at least one of the optimal orderings π^* .

Proof. Suppose not. Without loss of generality, suppose the naturally ordered skein of jobs $1, 2, \dots, k$ appears as a transitively ordered subset in no optimal ordering. Let π_0 be any optimal ordering, for which the jobs $1, 2, \dots, k$ would not appear in the natural order. Clearly, at most, k pairwise interchanges (not necessarily of neighbors) would alter π_0 into a new ordering π^* in which the k jobs appear in their natural order. By the strong JAR property, it will be true that $C_{\max}(\pi^*) \leq C_{\max}(\pi_0)$, so that π^* is optimal and a contradiction is reached. \square

A well-known example of an algorithm based on a pairwise J -ordering, which is transitive and JAR, is Johnson's algorithm for $m = 2$. This algorithm is based on the pairwise J -ordering for which:

$$g(i, j) = \min(A_i, B_j), \quad i \neq j.$$

As defined earlier, A_i is the processing time, including the setup time, of job i on machine 1. B_j is the processing time, including the setup time, of job j on machine 2. We order job i before job j , i.e. $i \ll j$ if $g(i, j) \leq g(j, i)$.

For $m > 2$, it is difficult to find pairwise J -orderings that are transitive and JAR. However, it is sometimes possible to identify a pairwise J -ordering that is both JAR and transitive under suitable conditions on the parameters A_i , B_i , C_i , and so on. An example of this [5], for $m = 3$, is the pairwise J -ordering for which $(i, j) = \min(A_i + B_i, B_j + C_j)$. Conditions under which this pairwise J -ordering is both JAR and transitive are:

- i) $B_i \leq C_j$ for all $i \neq j$;
- ii) $B_i \leq A_j$ for all $i \neq j$;
- iii) $B_i \leq \min(A_i, C_i)$ for all i . (1)

Definition 7. Uniqueness: If, for some pairs (i, j) , we have $g(i, j) = g(j, i)$, pairwise J -ordering algorithms that are both transitive and JAR will lead to several optimal orderings and to a single optimal ordering, when no such pairs exist.

This does not mean, of course, that the latter case is an indication of the uniqueness of the optimal ordering. An example of this in the context of Johnson's algorithm is as follows.

Example 2. Consider a FSP with $n = 3$ and $m = 2$. Table 2 shows the processing time.

Using Johnson's algorithm, we obtain the following ordering: $\pi^* : (2, 1, 3)$, where we have $C_{\max}(\pi^*) = 61$, while another ordering $\pi' : (2, 3, 1)$ has $C_{\max}(\pi') = 61$, which is another optimal ordering.

4. Solution methods

This section provides two solution methods centered on the idea of pairwise J -ordering. The first one is for 3-machine PFS and the second is for general m -machine

PFS. For both algorithms, it will be shown that the cases are both JAR and transitive.

4.1. A pairwise J -ordering for the case of $m = 3$

This section introduces a pairwise J -ordering, called J_3 -ordering whose primary motivation is the ordering of two jobs in a 3-machine PFS. Indeed, the J_3 -ordering is shown in Theorem 3 to be JAR for the case of $n = 2$, i.e. it is shown to provide optimal ordering for that special case. Moreover, it is discussed that J_3 -ordering has also certain further properties in the case of arbitrary n .

As earlier mentioned, in any pairwise J -ordering algorithm, jobs are sequenced in pairs by a function called function g . The function used in J_3 -ordering algorithm is:

$$\begin{aligned} g(i, j) = f(A_i, B_i, B_j, C_j) = & \min(A_i, B_j) \\ & + I(A_i - B_j)I(C_j - B_i) \min(A_i + B_i - B_j, C_j) \\ & + [1 - I(A_i - B_j)I(C_j - B_i)] \min(B_i, C_j), \end{aligned} \quad (2)$$

where:

$$I(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

Theorem 3. J_3 -ordering algorithm gives the optimal ordering for the case of 2-job and 3-machine flow shop scheduling.

Proof. This theorem means job 1 proceeds job 2 if $g(1, 2) \leq g(2, 1)$. Let (A_1, B_1, C_1) and (A_2, B_2, C_2) be the parameters of the problem. Further abbreviating notations to be used are as follows:

- 1. $\ll 2$: Schedule job 1 before job 2,
not job 2 before job 1; (3)
- 2. $\ll 1$: Schedule job 2 before job 1,
not job 1 before job 2; (4)
- 1||2 : Schedule either job first. (5)

We identify twenty exhaustive parametric cases involving the six constants A_1, \dots, C_2 , for which the resolution of the question of optimal precedence between jobs 1 and 2 is made clear. Because of the equalities in the parametric characterization of these twenty cases, the latter are not mutually exclusive. However, it is clear that when the same parametric condition appears more than once, the same ordering conclusion exists.

Table 2. The processing time of Example 2.

Job i	A_i	B_i
1	14	19
2	10	20
3	16	12

- **Case 1.** If $A_1 \leq B_2$, $C_2 \leq B_1$, $A_2 \leq B_1$, and $C_1 \leq B_2$, then 1. $\ll 2$, 2. $\ll 1$, or $1 \parallel 2$, according to whether $A_1 + C_2 \leq A_2 + C_1$, $A_1 + C_2 > A_2 + C_1$, or $A_1 + C_2 = A_2 + C_1$, where the relationship between $A_1 + C_2$ and $A_2 + C_1$ is called the “key relationship” for case 1. To avoid uninformative repetition in the remaining nineteen cases, this last assertion is condensed to:

$$A_1 \leq B_2, \quad C_2 \leq B_1, \quad A_2 \leq B_1, \quad C_1 \leq B_2 :$$

$$1 : 2 :: (A_1 + C_2) : (A_2 + C_1).$$

- **Case 2.** $A_1 \leq B_2$, $B_1 \leq C_2$, $A_2 \leq B_1$, $C_1 \leq B_2$:

$$1 : 2 :: (A_1 + B_1) : (A_2 + C_1).$$

- **Case 3.** $B_2 \leq A_1$, $C_2 \leq B_1$, $A_2 \leq B_1$, $C_1 \leq B_2$:

$$1 : 2 :: (B_2 + C_2) : (A_2 + C_1).$$

- **Case 4.** $A_1 \leq B_2$, $C_2 \leq B_1$, $A_2 \leq B_1$, $B_2 \leq C_1$:

$$1 : 2 :: (A_1 + C_2) : (A_2 + B_2).$$

- **Case 5.** $A_1 \leq B_2$, $B_1 \leq C_2$, $A_2 \leq B_1$, $B_2 \leq C_1$:

$$1 : 2 :: (A_1 + B_1) : (A_2 + B_2).$$

- **Case 6.** $B_2 \leq A_1$, $C_2 \leq B_1$, $A_2 \leq B_1$, $B_2 \leq C_1$:

$$1 : 2 :: (C_2) : (A_2).$$

- **Case 7.** $A_1 \leq B_2$, $C_2 \leq B_1$, $B_1 \leq A_2$, $C_1 \leq B_2$:

$$1 : 2 :: (A_1 + C_2) : (B_1 + C_1).$$

- **Case 8.** $A_1 \leq B_2$, $B_1 \leq C_2$, $B_1 \leq A_2$, $C_1 \leq B_2$:

$$1 : 2 :: (A_1) : (C_1).$$

- **Case 9.** $B_2 \leq A_1$, $C_2 \leq B_1$, $B_1 \leq A_2$, $C_1 \leq B_2$:

$$1 : 2 :: (B_2 + C_2) : (B_1 + C_1).$$

- **Case 10.** $B_2 \leq A_1$, $B_1 \leq C_2$, $B_1 \leq A_2$, $C_1 \leq B_2$, $B_1 + A_1 - B_2 \leq C_2$:

$$1 : 2 :: (A_1) : (C_1).$$

- **Case 11.** $A_1 \leq B_2$, $C_2 \leq B_1$, $B_1 \leq A_2$, $B_2 \leq C_1$, $B_2 + A_2 - B_1 \leq C_1$:

$$1 : 2 :: (A_1 + C_2) : (A_2 + B_2).$$

- **Case 12.** $A_1 \leq B_2$, $B_1 \leq C_2$, $B_1 \leq A_2$, $B_2 \leq C_1$, $B_2 + A_2 - B_1 \leq C_1$:

$$1 : 2 :: (A_1 + B_1) : (A_2 + B_2).$$

- **Case 13.** $B_2 \leq A_1$, $C_2 \leq B_1$, $B_1 \leq A_2$, $B_2 \leq C_1$, $B_2 + A_2 - B_1 \leq C_1$:

$$1 : 2 :: (C_2) : (A_2).$$

- **Case 14.** $B_2 \leq A_1$, $B_1 \leq C_2$, $B_1 \leq A_2$, $B_2 \leq C_1$, $B_1 + A_1 - B_2 \leq C_1$, $B_2 + A_2 - B_1 \leq C_1$:

$$1 : 2 :: (A_1 + B_1) : (A_2 + B_2).$$

- **Case 15.** $B_2 \leq A_1$, $B_1 \leq C_2$, $B_1 \leq A_2$, $B_2 \leq C_1$, $C_2 \leq B_1 + A_1 - B_2$, $B_2 + A_2 - B_1 \leq C_1$:

$$1 : 2 :: (C_2) : (A_2).$$

- **Case 16.** $A_1 \leq B_2$, $C_2 \leq B_1$, $B_1 \leq A_2$, $B_2 \leq C_1$, $C_1 \leq B_2 + A_2 - B_1$:

$$1 : 2 :: (A_1 + C_2) : (B_1 + C_1).$$

- **Case 17.** $A_1 \leq B_2$, $B_1 \leq C_2$, $B_1 \leq A_2$, $B_2 \leq C_1$, $C_1 \leq B_2 + A_2 - B_1$:

$$1 : 2 :: (A_1) : (C_1).$$

- **Case 18.** $B_2 \leq A_1$, $C_2 \leq B_1$, $B_1 \leq A_2$, $B_2 \leq C_1$, $C_1 \leq B_2 + A_2 - B_1$:

$$1 : 2 :: (B_2 + C_2) : (B_1 + C_1).$$

- **Case 19.** $B_2 \leq A_1$, $B_1 \leq C_2$, $B_1 \leq A_2$, $B_2 \leq C_1$, $C_1 \leq B_2 + A_2 - B_1$, $A_1 + B_1 - B_2 \leq C_2$:

$$1 : 2 :: (A_1) : (C_1).$$

- **Case 20.** $B_2 \leq A_1$, $B_1 \leq C_2$, $B_1 \leq A_2$, $B_2 \leq C_1$, $C_2 \leq B_1 + A_1 - B_2$, $C_1 \leq B_2 + A_2 - B_1$:

$$1 : 2 :: (B_2 + C_2) : (B_1 + C_1).$$

We intend to prove the assertion of the theorem in the following steps:

- We consider the condition $g(1,2) < g(2,1)$ in the context of each of the twenty exhaustive parametric cases, and find that, in every case, it leads to the inequality in the key relationship that corresponds to 1. $\ll 2$;
- We consider the condition $g(1,2) > g(2,1)$ in the context of each of the twenty exhaustive parametric cases, and find that, in every case, it leads to the inequality in the key relationship that corresponds to 2. $\ll 1$;
- We consider the condition $g(1,2) = g(2,1)$ in the context of each of the twenty exhaustive parametric cases, and find that, in every case, it leads to the inequality in the key relationship that corresponds to 1 \parallel 2.

Due to the similar procedure to prove, we do not examine parts (a), (b), and (c) for all twenty cases in details, yet only part (a) for cases 1, 6, 7, and 10. The discussion would employ the following notations: $\pi_1 : (1, 2)$ and $\pi_2 : (2, 1)$.

- **Examining Part (a) for Case 1.** Suppose that Relation (3) is held using Relation (2). We get $A_1 + C_2 < A_2 + C_1$. By adding $B_1 + B_2$ to both sides, we have: $A_1 + B_1 + B_2 + C_2 < A_2 + B_1 + B_2 + C_1$. Using the Gantt chart in Figures 4 and 5, we observe that the $A_1 + B_1 + B_2 + C_2 = C_{\max}(\pi_1)$ and $A_2 + B_1 + B_2 + C_1 = C_{\max}(\pi_2)$. Hence, it implies that $C_{\max}(\pi_1) < C_{\max}(\pi_2)$.

Therefore, we have examined part (a). Similarly, we could easily prove parts (b) and (c) by considering $A_1 + C_2 > A_2 + C_1$ and $A_1 + C_2 = A_2 + C_1$, respectively.

- **Examining Part (a) for Case 6.** Consider that Relation (3) is held using Relation (2). Then, we get $B_2 + C_2 < A_2 + B_2$. Adding $A_1 + B_1 + C_1 - C_2$ to both sides, we have: $A_1 + B_1 + C_1 + C_2 < A_2 + A_1 + B_1 + C_1$. Figures 6 and 7 specify that $C_{\max}(\pi_1) < C_{\max}(\pi_2)$.
- **Examining Part (a) for Case 7.** Assume that Relation (3) is held using Relation (2), then we reach $A_1 + C_2 < B_1 + C_1$. By adding $A_2 + B_2$ to both sides, the inequality becomes $A_1 + A_2 + B_2 + C_2 < A_2 + B_2 + B_1 + C_1$. As it is shown by Figures 8 and 9, we know that $C_{\max}(\pi_1) = A_1 + A_2 + B_2 + C_2$ and $C_{\max}(\pi_2) = A_2 + B_2 + B_1 + C_1$ where $C_{\max}(\pi_1) \leq C_{\max}(\pi_2)$.
- **Examining Part (a) for Case 10.** Considering Relation (2), assume that $A_1 < C_1$. Then, we can rewrite the inequality by adding $A_2 + B_2 + C_2$ to both of its sides: $A_1 + A_2 + B_2 + C_2 < A_2 + B_2 + C_2 + C_1$. As it is implied by Figures 10 and 11, we can conclude $C_{\max}(\pi_1) \leq C_{\max}(\pi_2)$.

The same procedure could be applied to all the other cases, and this completes the proof. \square

In the next section, we show that J_3 -ordering is

Machine 1	1	2	
Machine 2		1	2
Machine 3			1 2

Figure 4. The Gantt chart of π_1 for Case 1.

Machine 1	2	1	
Machine 2		2	1
Machine 3			2 1

Figure 5. The Gantt chart of π_2 for Case 1.

Machine 1	1	2	
Machine 2		1	2
Machine 3			1 2

Figure 6. The Gantt chart of π_1 for Case 6.

Machine 1	2	1	
Machine 2		2	1
Machine 3			2 1

Figure 7. The Gantt chart of π_2 for Case 6.

Machine 1	1	2	
Machine 2		1	2
Machine 3			1 2

Figure 8. The Gantt chart of π_1 for Case 7.

Machine 1	2	1	
Machine 2		2	1
Machine 3			2 1

Figure 9. The Gantt chart of π_2 for Case 7.

Machine 1	1	2	
Machine 2		1	2
Machine 3			1 2

Figure 10. The Gantt chart of π_1 for Case 10.

Machine 1	2	1	
Machine 2		2	1
Machine 3			2 1

Figure 11. The Gantt chart of π_2 for Case 10.

equivalent to the well-known 3-machine adaptation of Johnson's pairwise J -ordering under the three parametric conditions in Relations (1) of Section 3 that have been advanced by Burns and Rooker [5]. In other words, under the three parametric conditions in Relations (1) of Section 3, J_3 -ordering is conditionally JAR and transitive, so that J_3 -ordering provides an optimal ordering under these conditions.

Theorem 4. If, in a 3-machine flow shop, the parameters A_i , B_i , and C_i satisfy any of the conditions in Relations (1), then function g in Relation(2) becomes the following function:

$$g(i, j) = \min(A_i + B_i, B_j + C_j).$$

Proof. Suppose that the first condition holds. Function g in Relations (1) becomes:

$$\begin{aligned} g(i, j) &= B_j + I(C_i - B_j) \min(A_i + B_i - B_j, C_j) \\ &\quad + [1 - I(C_j - B_j)] \min(B_i, C_j) \\ &= \begin{cases} B_j + C_j, & \text{if } (C_j \leq B_i) \text{ or } (C_j < B_i \\ & \text{and } C_j \leq A_i + B_i - B_j) \\ A_i + B_i & \text{if } (C_j \leq B_i \\ & \text{and } A_i \leq B_i - B_j < C_j \end{cases} \\ &= \min(A_i + B_i, B_j + C_j). \end{aligned}$$

Suppose that the second condition holds. Function g in Relations (1) becomes:

$$\begin{aligned} g(i, j) &= \min(A_i, B_j) \\ &\quad + I(A_i - B_j) \min(A_i + B_i - B_j, C_j) \\ &\quad + [1 - I(A_i - B_j)] \min(B_i, C_j) \\ &= \begin{cases} A_i + B_i, & \text{if } (A_i \leq B_j) \text{ or } (B_j < A_i \\ & \text{and } A_i + B_i - B_j < C_j) \\ B_j + C_j & \text{if } (B_j \leq A_i \\ & \text{and } C_j \leq A_i + B_i - B_j) \end{cases} \\ &= \min(A_i + B_i, B_j + C_j). \end{aligned}$$

Suppose that the third condition holds. Function g in Relations (1) becomes:

$$\begin{aligned} g(i, j) &= \begin{cases} A_i + B_i, & \text{if } (A_i \leq B_j) \text{ or } (B_j < A_i \\ & \text{and } A_i + B_i - B_j \leq C_j) \\ B_j + C_j & \text{if } (B_j \leq A_i \text{ and } C_j < B_i) \end{cases} \\ &= \min(A_i + B_i, B_j + C_j). \end{aligned}$$

Note that the above function g cannot be $A_i + C_j$, because if $g(i, j) = A_i + C_j$, then we have:

$$B_j > A_i, \quad A_i \geq B_i, \quad \text{and} \quad B_i > C_j,$$

which imply that $B_j > C_j$, a contradiction to the third condition. \square

4.2. Generalization of Johnson's algorithm to the case of $m > 3$

This section extends the Johnson's algorithm to the case of $m > 3$, which generalizes the results obtained by Burns and Rooker [5] for the case of $m = 3$. First,

the paper studies the subject through a theorem for $m = 4$ and provides an example. Then, the paper extends the analysis to the case of $m > 4$.

For the case of $m = 4$, we define the function g used in pairwise J -ordering as follows:

$$g(i, j) = \min(A_i + B_i + C_i, B_j + C_j + D_j). \quad (6)$$

In the following, we investigate the conditions under which an optimal ordering can be found by the pairwise J -ordering algorithm.

Theorem 5. In a 4-machine permutation flow shop with $n > 2$, the pairwise J -ordering based on the function in Eq. (6) achieves the optimal ordering under any of the following four conditions:

- (i) $C_i \leq B_j, B_i \leq A_j$ for all $i \neq j$;
- (ii) $B_i \leq C_j, C_i \leq D_j$ for all $i \neq j$;
- (iii) $B_i \leq A_j, C_i \leq D_j$ for all $i \neq j$;
- (iv) $B_i + C_i \leq \min(A_i, D_i)$ for all i .

Proof. Let π be any permutation schedule. Let us denote the idle time on machines 2, 3, and 4 by X_i , Y_i , and Z_i , respectively, just before the processing of i th job processed. The dependence of the X_i , Y_i , and Z_i on π will be made explicit by adopting the notation $X_i(\pi)$, $Y_i(\pi)$, and $Z_i(\pi)$. Moreover, an analogous notation adopted for the processing times, with $A_i(\pi)$ as an example, denotes the processing time of i th job in π on machine 1. Figure 12 illustrates permutation schedule π and $X_i(\pi)$, $Y_i(\pi)$, $Z_i(\pi)$, $A_i(\pi)$, $B_i(\pi)$, $C_i(\pi)$, and $D_i(\pi)$.

It is readily verified by Figure 12 that we have $C_{\max}(\pi) = \sum_{i=1}^n D_i(\pi) + \sum_{i=1}^n Z_i(\pi)$, so that minimizing $C_{\max}(\pi)$ is equivalent to minimizing $\sum_{i=1}^n Z_i(\pi)$. It is the latter that is now shown to be minimized under conditions (i), (ii), (iii) or (iv) by the pairwise J -ordering based on function g presented in Eq. (6). The proof of the theorem proceeds in several steps:

- **Step 1.** We have:

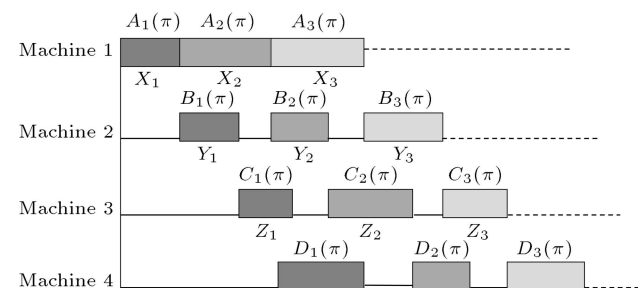


Figure 12. An illustration of typical π .

$$\begin{aligned} Z_1(\pi) &= Y_1(\pi) + C_1(\pi) = X_1(\pi) + B_1(\pi) + C_1(\pi) \\ &= A_1(\pi) + B_1(\pi) + C_1(\pi), \end{aligned}$$

$$\begin{aligned} Z_2(\pi) &= \max(C_1(\pi) + C_2(\pi) + Y_1(\pi) + Y_2(\pi) \\ &\quad - D_1(\pi) - Z_1(\pi), 0), \end{aligned}$$

⋮

$$\begin{aligned} Z_n(\pi) &= \max\left(\sum_{i=1}^n C_i(\pi) + \sum_{i=1}^n Y_i(\pi) - \sum_{i=1}^{n-1} D_i(\pi) \right. \\ &\quad \left. - \sum_{i=1}^{n-1} Z_i(\pi), 0\right), \end{aligned} \quad (7)$$

where:

$$Y_1(\pi) = X_1(\pi) + B_1(\pi) = A_1(\pi) + B_1(\pi),$$

$$\begin{aligned} Y_2(\pi) &= \max(B_1(\pi) + B_2(\pi) + X_1(\pi) + X_2(\pi) \\ &\quad - C_1(\pi) - Y_1(\pi), 0), \end{aligned}$$

⋮

$$\begin{aligned} Y_n(\pi) &= \max\left(\sum_{i=1}^n B_i(\pi) + \sum_{i=1}^n X_i(\pi) - \sum_{i=1}^{n-1} C_i(\pi) \right. \\ &\quad \left. - \sum_{i=1}^{n-1} Y_i(\pi), 0\right), \end{aligned} \quad (8)$$

with:

$$X_1(\pi) = A_1(\pi),$$

$$X_2(\pi) = \max(A_1(\pi) + A_2(\pi) - B_1(\pi) - X_1(\pi), 0),$$

⋮

$$\begin{aligned} X_n(\pi) &= \max\left(\sum_{i=1}^n A_i(\pi) \right. \\ &\quad \left. - \sum_{i=1}^{n-1} B_i(\pi) - \sum_{i=1}^{n-1} X_i(\pi), 0\right). \end{aligned} \quad (9)$$

From Eq. (7), we know:

$$\begin{aligned} \sum_{i=1}^n Z_i(\pi) &= \max\left(\sum_{i=1}^n C_i(\pi) - \sum_{i=1}^{n-1} D_i(\pi) \right. \\ &\quad \left. + \sum_{i=1}^n Y_i(\pi), \sum_{i=1}^{n-1} Z_i(\pi)\right) = \max\left(\sum_{i=1}^n C_i(\pi) \right. \end{aligned}$$

$$\begin{aligned} &\quad \left. - \sum_{i=1}^{n-1} D_i(\pi) + \sum_{i=1}^n Y_i(\pi), \sum_{i=1}^{n-1} C_i(\pi) \right. \\ &\quad \left. + \sum_{i=1}^{n-2} D_i(\pi) + \sum_{i=1}^{n-1} Y_i(\pi), \dots, C_1(\pi) + C_2(\pi) \right. \\ &\quad \left. - D_1(\pi) + Y_1(\pi) + Y_2(\pi), C_1(\pi) + Y_1(\pi)\right). \end{aligned}$$

But, from Eqs. (8) and (9), we have:

$$\begin{aligned} \sum_{i=1}^n Y_i(\pi) &= \max\left[\left(\sum_{i=1}^n A_i(\pi) - \sum_{i=1}^{n-1} B_i(\pi)\right) \right. \\ &\quad \left. + \left(\sum_{i=1}^n B_i(\pi) - \sum_{i=1}^{n-1} C_i(\pi)\right), \left(\sum_{i=1}^{n-1} A_i(\pi) - \sum_{i=1}^{n-2} B_i(\pi)\right) \right. \\ &\quad \left. + \left(\sum_{i=1}^{n-1} B_i(\pi) - \sum_{i=1}^{n-2} C_i(\pi)\right), \dots, \left(\sum_{i=1}^2 A_i(\pi) - B_1(\pi)\right) \right. \\ &\quad \left. + \left(\sum_{i=1}^2 B_i(\pi) - C_1(\pi)\right), (A_1(\pi) + (B_1(\pi)))\right], \end{aligned}$$

and:

$$\begin{aligned} \sum_{i=1}^n X_i(\pi) &= \max\left[\left(\sum_{i=1}^n A_i(\pi) - \sum_{i=1}^{n-1} B_i(\pi)\right), \right. \\ &\quad \left(\sum_{i=1}^{n-1} A_i(\pi) - \sum_{i=1}^{n-2} B_i(\pi)\right), \dots, \\ &\quad \left(\sum_{i=1}^2 A_i(\pi) - B_1(\pi)\right), (A_1(\pi))\right]. \end{aligned}$$

Simplification of $\sum_{i=1}^n Z_i(\pi)$ based on the last two equalities now yields:

$$\begin{aligned} \sum_{i=1}^n Z_i(\pi) &= \max\left[\left(\sum_{i=1}^n C_i(\pi) - \sum_{i=1}^{n-1} D_i(\pi)\right) \right. \\ &\quad \left. + \left(\sum_{i=1}^n A_i(\pi) - \sum_{i=1}^{n-1} B_i(\pi)\right) \right. \\ &\quad \left. + \left(\sum_{i=1}^n B_i(\pi) - \sum_{i=1}^{n-1} C_i(\pi)\right), \left(\sum_{i=1}^{n-1} C_i(\pi) - \sum_{i=1}^{n-2} D_i(\pi)\right) \right. \\ &\quad \left. + \left(\sum_{i=1}^{n-1} A_i(\pi) - \sum_{i=1}^{n-2} B_i(\pi)\right) \right] \end{aligned}$$

$$\begin{aligned}
& + \left(\sum_{i=1}^{n-1} B_i(\pi) - \sum_{i=1}^{n-2} C_i(\pi) \right), \dots, \left(\sum_{i=1}^2 C_i(\pi) - D_1(\pi) \right) \\
& + \left(\sum_{i=1}^2 A_i(\pi) - B_1(\pi) \right) + \left(\sum_{i=1}^2 B_i(\pi) - C_1(\pi) \right), \\
& \left(C_1(\pi) \right) + \left(A_1(\pi) \right) + \left(B_1(\pi) \right) \Big]. \quad (10)
\end{aligned}$$

Let:

$$H_1(\pi) = A_1(\pi),$$

$$K_1(\pi) = B_1(\pi),$$

$$L_1(\pi) = C_1(\pi),$$

$$H_u(\pi) = \sum_{i=1}^u A_i(\pi) - \sum_{i=1}^{u-1} B_i(\pi);$$

$$u = 2, 3, \dots, n,$$

$$K_v(\pi) = \sum_{i=1}^v B_i(\pi) - \sum_{i=1}^{v-1} C_i(\pi);$$

$$v = 2, 3, \dots, n,$$

$$L_w(\pi) = \sum_{i=1}^w C_i(\pi) - \sum_{i=1}^{w-1} D_i(\pi);$$

$$w = 2, 3, \dots, n. \quad (11)$$

Substituting Eq. (11) in Eq. (10), we have:

$$\sum_{i=1}^n Z_i(\pi) = \max[(H_n(\pi) + K_n(\pi) + L_n(\pi)),$$

$$(H_{n-1}(\pi) + K_{n-1}(\pi) + L_{n-1}(\pi)), \dots,$$

$$(H_2(\pi) + K_2(\pi) + L_2(\pi)),$$

$$(H_1(\pi) + K_1(\pi) + L_1(\pi))],$$

which may be condensed to:

$$\sum_{i=1}^n Z_i(\pi) = \max_{1 \leq u \leq v \leq w \leq n} [H_u(\pi) + K_v(\pi) + L_w(\pi)]. \quad (12)$$

- **Step 2.** The next step is to examine the square-bracketed quantity on the R.H.S. of Eq. (12) and to derive certain algebraic conditions (conditions (*), (**), and (***) below) that insure this square-bracketed quantity may be bounded by certain more

elementary expressions. To this end, consider any given ordering π and any integer u , v , and w , such that $1 \leq u \leq v \leq w \leq n$. Then:

$$\begin{aligned}
H_u(\pi) + K_v(\pi) + L_w(\pi) & \leq \max(H_u(\pi) + K_u(\pi) \\
& + L_u(\pi), H_v(\pi) + K_v(\pi) + L_v(\pi), H_w(\pi) \\
& + K_w(\pi) + L_w(\pi)) \Leftrightarrow \{K_u(\pi) + L_w(\pi) \\
& \leq K_u(\pi) + L_u(\pi)\},
\end{aligned}$$

or:

$$\{H_u(\pi) + L_w(\pi) \leq H_v(\pi) + L_v(\pi)\},$$

or:

$$\begin{aligned}
& \{K_u(\pi) + L_w(\pi) \leq K_u(\pi) + L_u(\pi)\} \\
& \Leftrightarrow \left\{ \sum_{i=u+1}^v B_i(\pi) - \sum_{i=v}^w C_i(\pi) \leq C_u(\pi) - \sum_{i=u}^{w-1} D_i(\pi) \right\},
\end{aligned}$$

or:

$$\left\{ \sum_{i=u}^{v-1} B_i(\pi) - \sum_{i=v+1}^w C_i(\pi) \leq \sum_{i=u+1}^v A_i(\pi) + \sum_{i=v}^{w-1} D_i(\pi) \right\},$$

or:

$$\begin{aligned}
& \left\{ \sum_{i=u}^v B_i(\pi) - \sum_{i=v}^{w-1} C_i(\pi) \leq \sum_{i=u+1}^w A_i(\pi) + B_w(\pi) \right\} \\
& \Leftrightarrow \{(*) \text{ or } (**) \text{ or } (***)\},
\end{aligned}$$

where (*), (**), and (***) are as follows:

$$(*) \quad \sum_{i=u+1}^v B_i(\pi) - \sum_{i=v}^w C_i(\pi) \leq C_u(\pi) - \sum_{i=u}^{w-1} D_i(\pi).$$

$$(**) \quad \sum_{i=u}^{v-1} B_i(\pi) - \sum_{i=v+1}^w C_i(\pi) \leq \sum_{i=u+1}^v A_i(\pi) + \sum_{i=v}^{w-1} D_i(\pi).$$

$$(***) \quad \sum_{i=u}^v B_i(\pi) - \sum_{i=v}^{w-1} C_i(\pi) \leq \sum_{i=u+1}^w A_i(\pi) + B_w(\pi).$$

- **Step 3.** In this further step, we verify that (*), (**), and (***) follow any of the four conditions of Theorem 5. It is easily verified that if condition (i) is true, then (***) holds; if condition (ii) is true, then (*) holds; and if condition (iii) is true, then (**) holds. As for condition (iv), (*) and (***) cannot both fail to hold. Therefore, suppose that iv) holds, but (*) and (***) do not. Since (*) is not true, we have:

$$\sum_{i=u+1}^v B_i(\pi) + \sum_{i=v}^w C_i(\pi) > C_u(\pi) + \sum_{i=u}^{w-1} D_i(\pi). \quad (13)$$

And, using condition (iv), the R.H.S. of Relation (13) is such that:

$$C_u(\pi) + \sum_{i=u}^{w-1} D_i(\pi) \geq C_u(\pi) + \sum_{i=u}^{w-1} B_i(\pi) + \sum_{i=u}^{w-1} C_i(\pi). \quad (14)$$

Combining Relations (13) and (14), then simplifying, we have:

$$C_w(\pi) > C_u(\pi) + B_u(\pi) + \sum_{i=v+1}^{w-1} B_i(\pi) + \sum_{i=u}^{v-1} C_i(\pi). \quad (15)$$

Furthermore, since (***) is not true, we have:

$$\sum_{i=u}^v B_i(\pi) + \sum_{i=v}^{w-1} C_i(\pi) > \sum_{i=u+1}^w A_i(\pi) + B_w(\pi). \quad (16)$$

Using condition (iv), the R.H.S. of Relation (16) is such that:

$$\sum_{i=u+1}^w A_i(\pi) + B_w(\pi) \geq \sum_{i=u+1}^w B_i(\pi) + \sum_{i=u+1}^w C_i(\pi) + B_w(\pi). \quad (17)$$

Combining (16) and (17), then simplifying, we have:

$$B_u(\pi) > \sum_{i=v+1}^w B_i(\pi) + \sum_{i=u+1}^{v-1} C_i(\pi) + C_w(\pi) + B_w(\pi). \quad (18)$$

Now, adding Relation (15) to Relation (18), we have:

$$0 > 2 \left[C_u(\pi) + B_w(\pi) + \sum_{i=v+1}^{w-1} B_i(\pi) + \sum_{i=u+1}^{v-1} C_i(\pi) \right],$$

which is a contradiction.

Step 4. As the next-to-last step, in view of Condition III in Theorem 6, we now recognize that any of the four conditions of Theorem 5 implies at least one of the conditions of (*), (**), and (***). And, in view of III, any of the two latter implies that $\sum_{i=1}^n Z_i(\pi)$ may be minimizing:

$$\max_{1 \leq u \leq n} \{H_u(\pi) + K_u(\pi) + L_u(\pi)\}.$$

The last step utilizes this observation to verify that the pairwise J -ordering based on Theorem 4 is conditionally JAR. Since it is clearly transitive, this last step establishes the theorem.

Step 5. Let π' be the permutation schedule formed by interchanging the jobs in positions j and $j+1$ in π . Then,

$$\sum_{i=1}^n Z_i(\pi') = \max_{1 \leq u \leq n} \{H_u(\pi') + K_u(\pi') + L_u(\pi')\},$$

with:

$$H_1(\pi') = A_1(\pi'),$$

$$K_1(\pi') = B_1(\pi'),$$

$$L_1(\pi') = C_1(\pi'),$$

$$H_u(\pi') = \sum_{i=1}^u A_i(\pi') - \sum_{i=1}^{u-1} B_i(\pi'),$$

$$u = 2, 3, \dots, n,$$

$$K_v(\pi') = \sum_{i=1}^u B_i(\pi') - \sum_{i=1}^{u-1} C_i(\pi'),$$

$$u = 2, 3, \dots, n,$$

$$L_w(\pi') = \sum_{i=1}^u C_i(\pi') - \sum_{i=1}^{n-1} D_i(\pi'),$$

$$u = 2, 3, \dots, n,$$

and:

$$\begin{cases} A_i(\pi') = A_i(\pi) \\ B_i(\pi') = B_i(\pi) \\ C_i(\pi') = C_i(\pi) \\ D_i(\pi') = D_i(\pi) \end{cases} \quad \text{if } i \neq j, j+1,$$

$$\begin{cases} A_j(\pi') = A_{j+1}(\pi), & A_{j+1}(\pi') = A_j(\pi) \\ B_j(\pi') = B_{j+1}(\pi), & B_{j+1}(\pi') = B_j(\pi) \\ C_j(\pi') = C_{j+1}(\pi), & C_{j+1}(\pi') = C_j(\pi) \\ D_j(\pi') = D_{j+1}(\pi), & D_{j+1}(\pi') = D_j(\pi) \end{cases}$$

Thus:

$$\sum_{i=1}^n Z_i(\pi') = \sum_{i=1}^n Z_i(\pi),$$

unless, possibly:

$$\max[(H_j(\pi') + K_j(\pi') + L_j(\pi'))],$$

$$(H_{j+1}(\pi') + K_{j+1}(\pi') + L_{j+1}(\pi'))]$$

$$\neq \max[(H_j(\pi) + K_j(\pi) + L_j(\pi)),$$

$$(H_{j+1}(\pi) + K_{j+1}(\pi) + L_{j+1}(\pi))].$$

Let:

$$\begin{aligned} & \max[(H_j(\pi) + K_j(\pi) + L_j(\pi)), \\ & (H_{j+1}(\pi) + K_{j+1}(\pi) + L_{j+1}(\pi))] \\ & < \max[(H_j(\pi') + K_j(\pi') + L_j(\pi')), \\ & (H_{j+1}(\pi') + K_{j+1}(\pi') + L_{j+1}(\pi'))]. \end{aligned} \quad (19)$$

If we subtract:

$$\begin{aligned} & \left[\left(\sum_{i=1}^{j+1} A_i(\pi) - \sum_{i=1}^{j-1} B_i(\pi) \right) + \left(\sum_{i=1}^{j+1} B_i(\pi) - \sum_{i=1}^{j-1} C_i(\pi) \right) \right. \\ & \left. + \left(\sum_{i=1}^{j+1} C_i(\pi) - \sum_{i=1}^{j-1} D_i(\pi) \right) \right], \end{aligned}$$

from both sides of Relation (19), we have:

$$\begin{aligned} & \max[-B_j - C_j - D_j, -A_{j+1} - B_{j+1} - C_{j+1}] \\ & < \max[-B_{j+1} - C_{j+1} - D_{j+1}, -A_j - B_j - C_j], \end{aligned}$$

or:

$$\begin{aligned} & \min[A_j + B_j + C_j, B_{j+1} + C_{j+1} + D_{j+1}] \\ & < \min[A_{j+1} + B_{j+1} + C_{j+1}, B_j + C_j + D_j]. \end{aligned} \quad (20)$$

Substituting g from Theorem 4 in Relation (20), we have:

$$g(j, j+1) < g(j+1, j),$$

which says that the pairwise J -ordering of Theorem 4 is conditionally JAR. \square

The above optimal ordering is further illustrated by applying it to an example with $n = 7$ and $m = 4$. Table 3 shows the processing times. Notice that condition of Theorem 4 is satisfied here. Table 4 shows the parameters $(A_i + B_i + C_i)$ and $(B_i + C_i + D_i)$. Using the rule, we obtain $\pi^* : (1, 2, 5, 7, 4, 3, 6)$, with $F_{\max}(\pi^*) = 102$.

In the following, Theorem 5 is generalized to the case of m -machine permutation flow shops. The parameter is also defined as follows:

p_{ij} : the processing time, including the setup time, of job i on machine j , $i = 1, \dots, n$; $j = 1, \dots, m$.

Table 3. The processing times of the example with $n = 7$ and $m = 4$.

Machines	Jobs						
	1	2	3	4	5	6	7
A	12	10	11	10	18	15	9
B	6	7	9	9	7	8	10
C	7	5	6	5	5	6	4
D	18	9	4	5	8	3	5

Table 4. The processing times of the example with $n = 7$ and $m = 4$.

	Jobs						
	1	2	3	4	5	6	7
$A_i + B_i + C_i$	25	22	26	24	30	29	23
$B_i + C_i + D_i$	31	21	19	19	20	17	19

Theorem 6. In an m -machine where $n > 2$ permutation flowshops if we have:

$$g(i, j) = \min \left(\sum_{k=1}^{m-1} p_{ik}, \sum_{k=2}^m p_{jk} \right),$$

then, under any of the following m conditions, the above pairwise J -ordering yields an optimal ordering of n jobs:

$$1) p_{i,1} \geq p_{j,2}, p_{i,2} \geq p_{j,3}, \dots, p_{i,m-2}$$

$$\geq p_{j,m-1} \quad \text{for all } i \neq j,$$

$$2) p_{i,1} \geq p_{j,2}, p_{i,2} \geq p_{j,3}, \dots, p_{i,m-3}$$

$$\geq p_{j,m-2}, p_{i,m-1} \leq p_{j,m} \quad \text{for all } i \neq j,$$

\vdots

$$k) p_{i,1} \geq p_{j,2}, p_{i,2} \geq p_{j,3}, \dots, p_{i,m-k-1}$$

$$\leq p_{j,m-k}, p_{i,m-k+1} \geq p_{i,m-k+2}, \dots, p_{i,m-1}$$

$$\geq p_{j,m} \quad \text{for all } i \neq j,$$

\vdots

$$m-2) p_{i,1} \geq p_{j,2}, p_{i,3} \leq p_{j,4}, p_{i,4} \leq p_{j,5}, \dots, p_{i,m-1}$$

$$\leq p_{j,m} \quad \text{for all } i \neq j,$$

$$m-1) p_{i,2} \leq p_{j,3}, p_{i,3} \leq p_{j,4}, \dots, p_{i,m-1}$$

$$\leq p_{j,m} \quad \text{for all } i \neq j,$$

$$m) \sum_{j=2}^{m-1} p_{i,j} \leq \min(p_{i,1}, p_{i,m}) \quad \text{for all } i.$$

Proof. Let π be any permutation schedule. Define $x_{i,j}(\pi)$ to be the idle time under π on machine j , for $j = 2, 3, \dots, m$, just before the beginning of the processing of the i th job processed, $i = 1, 2, \dots, n$. Also, let $p_{i,j}(\pi)$ denote the processing time, including setup time, on machine j under π of the i th job processed, for all $j = 1, 2, \dots, m$ and $i = 1, 2, \dots, n$. The permutation schedule π , $x_{i,j}(\pi)$, and $p_{i,j}(\pi)$ are illustrated in Figure 13. It is readily verified, with the

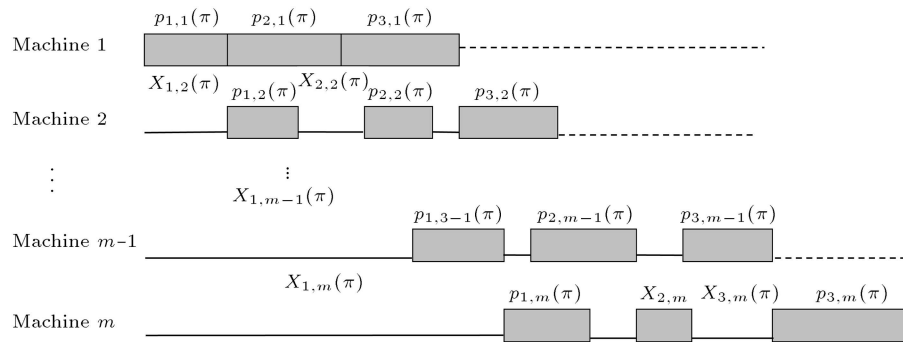


Figure 13. An illustration of a permutation schedule π .

help of Figure 13, that:

$$C_{\max}(\pi) = \sum_{i=1}^n p_{i,m} + \sum_{i=1}^n x_{i,m}(\pi).$$

So that minimizing $F_{\max}(\pi)$ is equivalent to minimizing $\sum_{i=1}^n x_{i,m}(\pi)$. It is the latter that is now shown to be minimized under $1), 2), \dots, m)$, by the pairwise J -ordering based on Theorem 6. The proof is analogous to the proof of Theorem 5, with the same steps. Now, we briefly explain each step.

I. Analogously to Step 1 of the proof of Theorem 5,

$$\sum_{i=1}^n x_{i,m}(\pi) = \max_{1 \leq u_1 \leq u_2 \leq \dots \leq u_{m-1} \leq n} \left[\sum_{j=1}^{m-1} H_{u_j, j}(\pi) \right],$$

where:

$$H_{1,j}(\pi) = p_{1,j}(\pi),$$

and:

$$H_{u_j, j}(\pi) = \sum_{i=1}^{u_j} p_{i,j}(\pi) - \sum_{i=1}^{u_j-1} p_{i,j+1}(\pi),$$

for all $j = 1, 2, \dots, m-1$, and $u_j = 2, 3, \dots, n$.

II. In this step, analogously to Step 2 of the proof of Theorem 5, we can easily see that for all $1 \leq u_1 \leq u_2 \leq \dots \leq u_{m-1} \leq n$,

$$\sum_{j=1}^{m-1} H_{u_j, j}(\pi) \leq \max \left(\sum_{j=1}^{m-1} H_{u_1, j}(\pi), \right.$$

$$\left. \sum_{j=1}^{m-1} H_{u_2, j}(\pi), \dots, \sum_{j=1}^{m-1} H_{u_{m-1}, j}(\pi) \right)$$

$$\{(a_1) \text{ or } (a_2) \text{ or } (a_3) \text{ or } \dots \text{ or } (a_{m-1})\}$$

where:

$$(a_1) : \sum_{i=u_1+1}^{u_2} p_{i,2}(\pi) + \sum_{i=u_2}^{u_3} p_{i,3}(\pi) + \sum_{i=u_3}^{u_4} p_{i,4}(\pi)$$

$$+ \dots + \sum_{i=u_{m-2}}^{u_{m-1}} p_{i,m-1}(\pi)$$

$$\leq p_{u_1,3}(\pi) + p_{u_1,4}(\pi) + \dots$$

$$+ p_{u_1,m-1}(\pi) + \sum_{i=u_1}^{u_{m-1}-1} p_{i,m}(\pi),$$

$$(a_2) : \sum_{i=u_1}^{u_2-1} p_{i,2}(\pi) + \sum_{i=u_2+1}^{u_3} p_{i,3}(\pi) + \sum_{i=u_3}^{u_4} p_{i,4}(\pi)$$

$$+ \dots + \sum_{i=u_{m-2}}^{u_{m-1}} p_{i,m-1}(\pi)$$

$$\leq \sum_{i=u_1+1}^{u_2} p_{i,1}(\pi) + p_{u_2,4}(\pi) + \dots$$

$$+ p_{u_2,m-1}(\pi) + \sum_{i=u_2}^{u_{m-1}-1} p_{i,m}(\pi),$$

$$(a_3) : \sum_{i=u_1}^{u_2} p_{i,2}(\pi) + \sum_{i=u_2}^{u_3-1} p_{i,3}(\pi) + \sum_{i=u_3+1}^{u_4} p_{i,4}(\pi)$$

$$+ \sum_{i=u_4}^{u_5} p_{i,5}(\pi) + \dots + \sum_{i=u_{m-2}}^{u_{m-1}} p_{i,m-1}(\pi)$$

$$\leq \sum_{i=u_1+1}^{u_3} p_{i,1}(\pi) + p_{u_3,2}(\pi)$$

$$+ p_{u_3,5}(\pi) + \dots + p_{u_3,m-1}(\pi)$$

$$\begin{aligned}
& + \sum_{i=u_3}^{u_{m-1}-1} p_{i,m}(\pi), \\
& \vdots \\
(a_{m-1}) : & \sum_{i=u_1}^{u_2} p_{i,2}(\pi) + \sum_{i=u_2}^{u_3} p_{i,3}(\pi) \\
& + \cdots + \sum_{i=u_{m-2}}^{u_{m-1}-1} p_{i,m-1}(\pi) \\
& \leq \sum_{i=u_1+1}^{u_{m-1}} p_{i,1}(\pi) + p_{u_{m-1},2}(\pi) \\
& + \cdots + p_{u_{m-1},m-2}(\pi).
\end{aligned}$$

- III. It may be verified in a manner analogous to corresponding step in Theorem 5 that $(a_1), (a_2), \dots$, or (a_{m-1}) follow any of the m conditions of Theorem 6.
- IV. As the next-to-last step, we now recognize that, in view of step III, any of the m conditions of Theorem 6 implies at least one of conditions $(a_1), (a_2), \dots$, or (a_{m-1}) . And, in view of step II, any of the latter implies that $\sum_{i=1}^n x_{i,m}(\pi)$ may be minimized by minimizing:

$$\max_{1 \leq u \leq n} \left\{ \sum_{j=1}^{m-1} H_{u,j}(\pi) \right\}.$$

The last step utilizes this observation to verify that the pairwise J -ordering based on Theorem 6 is conditionally JAR. Since it is clearly transitive, this last step establishes the theorem.

- V. Let π' be the permutation schedule formed by interchanging the jobs in position i and $i+1$ in π . Then:

$$\sum_{i=1}^n x_{i,m}(\pi') = \max_{1 \leq u \leq n} \left\{ \sum_{j=1}^{m-1} U_{u,j}(\pi') \right\}.$$

We may use an argument similar to the corresponding one in Theorem 4.2.1, to see that the comparison of:

$$\min \left(\sum_{j=1}^{m-1} p_{i,j}, \sum_{j=1}^m p_{i+1,j} \right),$$

and:

$$\min \left(\sum_{j=1}^{m-1} p_{i+1,j}, \sum_{j=1}^m p_{i,j} \right),$$

which is equivalent to the comparison of $g(i, i+1)$ and $g(i+1, i)$, determines the relative worth of π and π' in a manner analogous to the case $m=4$, which shows that the pairwise J -ordering based on Theorem 6 is conditionally JAR. \square

The above generalization does involve generalizing the parametric restrictions of Burns and Rooker [5], which become increasingly restrictive with increasing m .

5. Conclusion and future research

A subclass of algorithms for ordering n jobs in a flow shop, of which Johnson's algorithm [4] is the outstanding example, consists of algorithms based on the ordering of pairs of jobs as a function only of the parameters of the pair. Such an ordering is called a "pairwise J -ordering" in this paper. A "schematic table" for a pairwise J -ordering is an $n \times n$ table with (i, j) th entry that shows whether job i precedes or follows job j in the ordering. If, for a pairwise J -ordering, we can find a schematic table with the same entries, then that pairwise J -ordering is said to be transitive. The concept of Job-Adjunction-Robustness (JAR) of a pairwise J -ordering is also introduced. We may note that this concept is implicitly used, but not identified in [4]. If a pairwise J -ordering is both JAR and transitive, then it leads to an optimal ordering. In the absence of transitivity of a pairwise J -ordering, the concept of "transitive skein", along with certain restricted JAR property, serves to partially identify an optimal ordering.

A certain pairwise J -ordering, the J_3 -ordering, has the following properties. It provides an optimal permutation schedule in the case $n=2$. Under the conditions of Burns and Rooker [5], it is equivalent to Johnson's adaptation of his algorithm to the 3-machine flow shop. Finally, the extension of Johnson's 3-machine adaptation [5] is generalized to the case of m machines. This generalization involves generalizing the parametric restrictions of Burns and Rooker [5], which becomes increasingly restrictive with increasing m . To counter this problem, the author suggests that it may be possible, in given practical situations, to "aggregate" neighboring machines with a single machine to meet at least one of the conditions of Theorem 6, creating an $m-k$ machine problem. Typically, the resulting permutation schedule will call for left-shift machine when machines are "disaggregated" under the $m-k$ optimal permutation schedule.

A different possibility for coping with the severity of the parametric restrictions is contained in the following conjecture: if a k partitioning of n jobs ($k \leq n$) exists, such that each of the k partition elements satisfies one of the m conditions of Theorem 6,

then the pairwise J -ordering given by Theorem 6 may conditionally be JAR and transitive.

References

1. Xie, Z., Zhang, C., Shao, X., Lin, W. and Zhu, H. "An effective hybrid teaching-learning-based optimization algorithm for permutation flow shop scheduling problem", *Advances in Engineering Software*, **77**, pp. 35-47 (2014).
2. Juan, A.A., Barrios, B.B., Vallada, E., Riera, D. and Jorba, J. "A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times", *Simulation Modelling Practice and Theory*, **46**, pp. 101-117 (2014).
3. Conway, R.W., Maxwell, W.L. and Miller, L.W., *Theory of Scheduling*, 1st Ed., Cambridge, Mass., Addison-Wesley (1967).
4. Johnson, S.M. "Optimal two- and three-stage production schedules with setup times included", *Naval Research Logistic Quarterly*, **1**(1), pp. 61-68 (1954).
5. Burns, F. and Rooker, J. "3-stage flowshops with recessive second stage", *Operations Research*, **26**(1), pp. 207-208 (1978).
6. Giglio, R. and Wagner, H. "Approximate solutions to the three-machine scheduling problems", *Operations Research*, **12**(2), pp. 305-324 (1964).
7. Dudek, R.A. and Teuton, J.R. "Development of m -stage decision rule for scheduling n jobs through n machines", *Operations Research*, **12**(3), pp. 471-497 (1964).
8. Karush, W. "A counterexample to a proposed algorithm for optimal sequencing of jobs", *Operations Research*, **13**(2), pp. 323-325 (1965).
9. Smith, M.L. and Dudek, R.A. "A general algorithm for solution of the n -job, m -machine sequencing problem of the flowshop", *Operations Research*, **15**(1), pp. 71-81 (1967).
10. Ashour, S. "A decomposition approach for the machine scheduling problems", Thesis, University of Iowa (1967).
11. Baker, K.R. "Elimination method for flowshop problems", *Operation Research*, **23**(1), pp. 159-162 (1975).
12. Dannenbring, D.G. "Evaluation of flowshop sequencing heuristics", *Management Science*, **23**(1), pp. 1174-1182 (1977).
13. Smith, M.L., Panwalkers, S.S. and Dudek, R.A. "Flowshop sequencing problem with ordered processing time matrices - General case", *Naval Research Logistics Quarterly*, **23**(3), pp. 481-486 (1976).
14. Szwarc, W. "Special cases for flowshop problems", *Naval Research Logistics Quarterly*, **24**(3), pp. 483-492 (1977).
15. Palmer, D.S. "Sequencing jobs through a multi-stage process in the minimum total time: a quick method of obtaining a near optimum", *Operational Research Quarterly*, **16**(1), pp. 101-107 (1965).
16. Campbell, H.G., Dudek, R.A. and Smith, M.L. "Heuristic algorithm for N -job, M -machine sequencing problem", *Management Science Series B-Application*, **16**(10), pp. 630-637 (1970).
17. Nawaz, M., Ensore, E.E. and Ham, I. "A heuristic algorithm for the m machine, n job flowshop sequencing problem", *Omega*, **11**(1), pp. 91-95 (1983).
18. Ruiz, R. and Maroto, C. "A comprehensive review and evaluation of permutation flowshop heuristics", *European Journal of Operational Research*, **165**(2), pp. 479-494 (2005).
19. Lin, Q., Gao, L., Li, X. and Zhang, C. "A hybrid backtracking search algorithm for permutation flowshop scheduling problem", *Computers and Industrial Engineering*, **85**, pp. 437-446 (2014).
20. Li, J.Q. and Pan, Q.K. "Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm", *Information Sciences*, **316**, pp. 487-502 (2015).
21. Moslehi, G. and Khorasani, D. "A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion", *Computers and Operations Research*, **52**, pp. 260-268 (2014).
22. Ribas, I., Companys, R. and Tort-Martorell, X. "An efficient discrete artificial bee colony algorithm for the blocking flow shop problem with total flow time minimization", *Expert Systems with Applications*, **42**(15-16), pp. 6155-6167 (2015).
23. Marichelvam, M.K., Prabakaran, T. and Yang, X.S. "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan", *Applied Soft Computing*, **19**, pp. 93-101 (2014).
24. Ruiz, R., Maroto, C. and Alcaraz, J. "Two new robust genetic algorithms for the flowshop scheduling problem", *Omega*, **34**, pp. 461-476 (2006).
25. Lin, B.M.T., Lu, C.Y., Shyu, S.J. and Tsai, C.Y. "Development of new features of ant colony optimization for flowshop scheduling", *International Journal of Production Economics*, **112**(2), pp. 742-755 (2008).
26. Naderi, B., Zandieh, M., Khaleghi Ghoshe Balagh, A. and Roshanaei, V. "An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness", *Expert Systems with Applications*, **36**(6), pp. 9625-9633 (2009).
27. Ekşioğlu, B., Ekşioğlu, S.D. and Jain, P. "A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods", *Computers and Industrial Engineering*, **54**(1), pp. 1-11 (2008).
28. Jarboui, B., Ibrahim, S., Siarry, P. and Rebai, A. "A combinatorial particle swarm optimisation for solving permutation flowshop problems", *Computers and Industrial Engineering*, **54**(3), pp. 526-538 (2008).

29. Ruiz, R. and Stützle, T. “A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem”, *European Journal of Operational Research*, **177**, pp. 2033-2049 (2007).

Biographies

Majid Aminnayeri received BS and MS degrees in Mathematics and Statistics from Shiraz University, Iran, in 1972 and 1974, respectively. He then received a PhD degree (double major) in Industrial Engineering and Statistics from Iowa State University, USA, in 1981. He has served as an academic member at Iowa State University, USA, and at Kerman University, Iran.

He is presently an Associate Professor in Amirkabir University of Technology, Tehran, Iran. His research interest includes statistical quality control and scheduling as well as general modeling.

Bahman Naderi completed his BSc degree in Industrial Engineering from Mazandaran University of Science and Technology (Iran), MSc and PhD from Amirkabir University of Technology Tehran, Iran, and post-doctoral program at University of Windsor, Canada. Currently, he is an Assistant Professor at Kharazmi University. His research interests are applied operations research (mathematical modeling and solution methods).