Journal of Manufacturing Systems 40 (2016) 2–7

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

An effective heuristic for no-wait flow shop production to minimize makespan

Honghan Ye [1], Wei Li [1,*], Enming Miao [2]

[1] University of Kentucky, Lexington, KY, USA
[2] Hefei University of Technology, Hefei, China

ARTICLE INFO

Article history:
Received 1 December 2015
Received in revised form 9 March 2016
Accepted 21 March 2016
Available online 21 May 2016

Keywords:
Makespan minimization
No-wait flow shop
Heuristics
Computational complexity

ABSTRACT

In no-wait flow shop production, each job must be processed without any interruption from its start time on the first machine to its completion time on the last machine. To minimize makespan in no-wait flow shop production is one of the main concerns in industry. In this paper, we propose an average departure time (ADT) heuristic for minimizing makespan in no-wait flow shop production. Based on the computational experiment with a large number of instances of various sizes, the ADT heuristic performs better than three existing best-known heuristics in the same computational complexity environment.

1. Introduction

No-wait flow shop production is an important production mode in many manufacturing systems such as petrochemical processing [1], steel rolling [2], and plastic molding [3]. For no-wait flow shop scheduling, the orders of n jobs processed on m machines are the same, and all jobs are available to start at time zero. Furthermore, each job must be processed continuously from the start to the end, i.e., no waiting time allowed on intermediate machines from the first machine to the last. Consequently, the start time on the first machine could be postponed to avoid waiting time on any intermediate machine. Take food processing as an example; the quality of food will change as time goes by. Therefore, there should be no waiting between operations during the processing; otherwise the quality of food will be jeopardized, even causing safety issues. For more details about applications of no-wait flow shop production, please refer to Hall and Sriskandarajah [4].

To minimize maximum completion time or makespan, min ($C_{\max}$), is one of the most meaningful objectives for no-wait flow shop production [5]. Makespan is the completion time of the last job on the last machine. There are also several other objectives commonly used to optimize the performance of no-wait flow shop production, such as to minimize total completion time [6], to minimize weighted mean completion time [7], and to minimize total tardiness [8].

To minimize $C_{\max}$ is NP-hard for no-wait flow shop production when the number of machines is larger than 2 [9]. Due to the NP-hardness of no-wait flow shop production to min ($C_{\max}$), it is extremely time consuming for exact algorithms to seek optimal solutions, even for moderate-scale problems [10]. Therefore, it is practical to use heuristics to seek optimal or near-optimal solutions in a reasonable time, especially for large-scale production problems in industry.

The NEH heuristic [11] has been widely regarded as the best constructive heuristic for permutation flow shop production to min ($C_{\max}$) [12] and also has been applied in no-wait flow shop scheduling [13]. The NEH heuristic initially sequences jobs in a non-ascending order by the sum of processing times of a job on all machines. The first two jobs are then selected from the initial sequence, and the partial sequence of these two jobs is fixed by the one with better makespan. The remaining unsequenced jobs, each in turn in the order of the initial sequence, are used to create a set of temporary sequences by inserting each job one-by-one at each position in the current sequence and calculating its $C_{\max}$. The temporary sequence whose job position has the minimum $C_{\max}$ is selected, the job positions are then frozen as the current sequence, and the next job in the initial sequence is examined. A final sequence is generated until all jobs are sequenced.

Gangadharan and Rajendran [14] proposed their GR heuristic for n-job m-machine no-wait flow shop production to min ($C_{\max}$). Given processing times of job j on machine i, $p_{j,i}$, where $j = 1, \ldots, n$, and $i = 1, \ldots, m$, we can calculate the sum of processing times of job j on all machines by $T_j = \sum_{i=1}^{m} p_{j,i}$, and an index for job j by $P_j = (\sum_{i=1}^{m} j \times p_{j,i})/T_j$. The GR heuristic has three steps to construct

* Corresponding author.
E-mail address: wei.mike.li@uky.edu (W. Li).

http://dx.doi.org/10.1016/j.jmsy.2016.05.001
0278-6125/

a sequence. First, the GR heuristic groups all jobs into two sets $\{A\}$ and $\{B\}$, where set $\{A\}$ consists of jobs whose $P_j \geq (1+m)/2$ and set $\{B\}$ consists of jobs whose $P_j < (1+m)/2$. Second, the GR heuristic generates an initial sequence by sequencing jobs in set $\{A\}$ in a non-descending order according to T_j and for jobs in set $\{B\}$ in a non-ascending order. An initial sequence can be $[A'B']$, where $[A']$ is the partial sequence for set $\{A\}$, and $[B']$ for set $\{B\}$. Third, the GR heuristic generates a final sequence by an insertion scheme, in which an index is set for the kth job in the initial sequence $[A'B']$, $k = 1, \ldots, n$; the kth job is removed from and inserted into the rest $n-1$ positions of the sequence $[A'B']$; the initial sequence is updated if improvement of makespan is found through the insertion, and such procedure continues until $k = n$. Rajendran [15] proposed the RAJ heuristic for n-job m-machine no-wait flow shop production to min $(C_{\max})$. The RAJ heuristic groups all jobs into two sets and generates an initial sequence in the same way as the GR heuristic, except sequencing jobs in each set according to that $T_j = \sum_{i=1}^{m}(m-j+1)p_{j,i}$, and uses a different insertion scheme to generate the final sequence. The RAJ heuristic uses two indexes. The index k is for the kth job in the initial sequence, and h for the number of jobs in a temporary sequence. Initially the temporary sequence is empty. The RAJ initially selects the first two jobs from the initial sequence, and the temporary sequence of these two jobs is fixed by the one with the better makespan. Currently, there are two $(h=2)$ jobs in the temporary sequence. As k changes from 3 to n, the kth job in the initial sequence is inserted into the $(h+1)/2$ to $h+1$ positions in the temporary sequence, where $\lfloor \cdot \rfloor$ is the floor function. For example, if $h=2$, then $(h+1)/2 = 1.5$, and $(h+1)/2 = 1$. As k increases from 3 to n, h increases from 3 to n as well, and the final sequence is constructed through the insertion scheme of the RAJ heuristic. According to the results of case studies [14,15], the GR and RAJ heuristics perform better on makespan minimization than the heuristics proposed by Bonney and Gundry [16] and King and Spachis [17].

Some heuristics in the literature can get better makespan than the GR or RAJ heuristics for no-wait flow shop scheduling, but have a longer computation time. Aldowaisan and Allahverdi [18] proposed an AA insertion scheme to improve the performance of the final sequence, which has a number of iterations much more than the insertion scheme in the GR, RAJ, or NEH heuristics. Such AA insertion scheme was applied to the genetic algorithm (GA) heuristic [19], and the simulated annealing (SA) heuristic [20]. Both the GA and SA heuristics perform better than the GR or RAJ heuristic, but take a much longer computation time. For a 600-job 160-machine instance, it took 3265.4 s of CPU time for the SA heuristic to generate a solution on a computer with a 2.0-GHz CPU and 256 M RAM [5]. Moreover, Laha and Chakraborty [21] proposed a heuristic, whose performance on min $(C_{\max})$ is better than the GR heuristic for large-scale instances, but worse in 40% of small scale instances, where the number of jobs is 6, 7, or 8.

In this paper, we propose an average departure time (ADT) heuristic for no-wait flow shop production to min $(C_{\max})$, and compare it with the GR, RAJ and NEH heuristics. The remainder of this paper is organized as follows. Section 2 provides necessary notations in the analysis and describes the problem of no-wait flow shop production. Section 3 presents our ADT heuristic. Section 4 presents the results of case studies and analysis. Section 5 draws conclusion and proposes future work.

2. Problem description

The following notations are used in problem description and formulation.

π	a sequence of n jobs, $\pi = [J_1, J_2, \ldots, J_{j-1}, J_j, \ldots, J_n]$;
n	the number of jobs;
m	the number of machines;
$p_{j,i}$	the processing time of job j on machine i, where $j = 1, \ldots, n$ and $i = 1, \ldots, m$;
$ST_{j,i}$	the start time of job j on machine i;
$CT_{j,i}$	the completion time of job j on machine i;
$d^i_{j-1,j}$	the potential distance between completion time of job j-1 and start time of job j on machine i;
$D_{j-1,j}$	the distance between two adjacent jobs' completion times on the last machine.

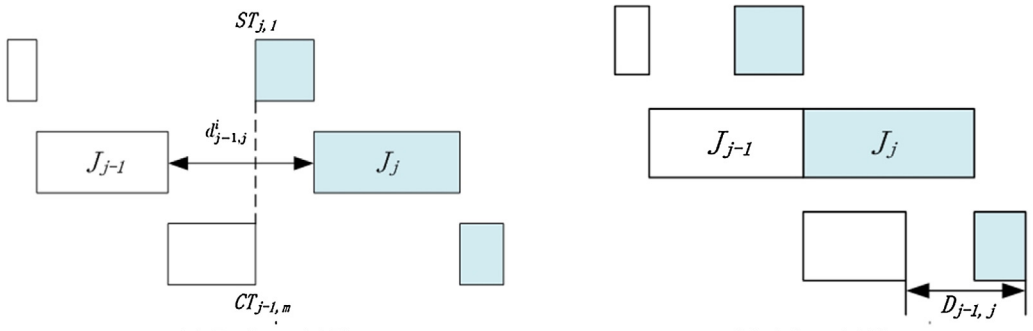
The calculation of $C_{\max}$ for no-wait flow shop production will be illustrated as follows. First we assume the start time of job j on the first machine equals to the completion time of job $j-1$ on the last machine as shown in Fig. 1(a) and Eq. (1). Meanwhile, there is no waiting time on intermediate machines for each job. Accordingly, the start time of job j on machine i and the completion time of job $j-1$ on machine i in Fig. 1(a) can be formulated by Eqs. (2) and (3).

Given initial conditions that $CT_{0,m} = 0$, $p_{j,0} = 0$, $p_{0,i} = 0$, $\sum_{k=1}^{0} p_{j,k}$, and $\sum_{k=m+1}^{m} p_{j,k} = 0$

$$ST_{j,1} = CT_{j-1,m} \quad \text{where} \quad j = 1, 2, \ldots, n \tag{1}$$

$$ST_{j,i} = ST_{j,1} + \sum_{k=1}^{i-1} p_{j,k} \quad \text{where} \quad j = 1, 2, \ldots, n \quad \text{and} \quad i = 1, 2, \ldots, m \tag{2}$$

$$CT_{j,i} = CT_{j,m} - \sum_{k=i+1}^{m} p_{j,k} \quad \text{where} \quad j = 1, 2, \ldots, n \quad \text{and} \quad i = 1, 2, \ldots, m \tag{3}$$



(a) Before shifting (b) After shifting

Fig. 1. Distance between two adjacent jobs.