Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

An improved NEH heuristic to minimize makespan for flow shop scheduling problems

Meenakshi Sharma[a], Manisha Sharma[a] and Sameer Sharma[b*]

[a]*Panjab University Chandigarh, India*
[b]*DAV College Jalandhar, India*

C H R O N I C L E

Article history:
Received December 28, 2020
Received in revised format: January 30, 2021
Accepted February 27 2021
Available online February 27, 2021

Keywords:
Scheduling
Flow shop
Makespan
Heuristic
Processing time

A B S T R A C T

Flow shop scheduling problems with rudimentary criteria of minimum makespan are the most important investigated problems in the field of scheduling. Generally during the process of generating an optimal sequence, multiple partial sequences claiming the optimal value of makespan are observed. In this paper a novel tie-breaking rule to select one of the best optimal sequences out of all possible partial sequences is developed which then applied to Nawaz-Enscore-Ham (NEH) heuristic to solve the scheduling problems in permutation flowshop without increasing the computational complexity. The performance of proposed heuristic is tested with other existing tie-breaking heuristics of similar complexity over Taillard and VRF's instances. Computational results reveal that in terms of solution quality, the proposed heuristic outperforms over the other NEH based heuristics of the same complexity reported in literature.

1. Introduction

Scheduling is a decision making process that plays a momentous role in production and manufacturing sectors. It is a process of taking decisions regarding when, where, how and how much workload can be distributed among various resource requirements. With a large scale advancement in the manufacturing sector, the practical importance of scheduling has raised to a prodigious level. Scheduling involves different branches of machine job environments such as single machine system, two machine system, multi machine system and open machine system, etc. The main objective of these scheduling problems is to schedule the jobs in an available machine environment in such a way that certain scheduling criteria can be successfully optimized. These criteria may be defined as measures of performance and generally categorized as: efficiency related (includes makespan, flow time, mean flow time, waiting time, idle time, etc.), cost related (includes transportation cost, maintenance cost, hiring cost etc.) and due date related (includes lateness, tardiness, number of tardy jobs, etc.). Flow shop scheduling problems (FSSP) are the special class of scheduling problems to obtain the fix processing order of n jobs on a system of m machines, while each machine can perform a single operation on these jobs and all jobs have the same sequence of machines. Due to practical significance and real life existence, FSSP is the most studied scheduling problem. Flow shop scheduling problem with the criteria of minimum makespan has attracted the attention of various researchers and practitioners for being the tool to measure efficiency rate of both production and service sectors. Garey et al. (1976) proved strong NP- completeness of flow shop scheduling problems related to the system of n jobs and m machines, when $m>2$. Literature reveals that in the last five decades different heuristic and metaheuristic algorithms have been developed by various researchers to solve the large scale job machine flow shop scheduling problems with the criteria of minimum makespan. Johnson (1954) was the first to investigate two and three stage flow shop scheduling problems with makespan criteria. Ignall and Schrage (1965) proposed a branch and bound method to obtain a sequence of jobs which, when processed on a system of m machines results in minimum makespan. Page (1961), Palmar (1965) proposed the simple index based heuristics to arrange the jobs in ascending or descending order of specified weights with the objective of minimum makespan. Campbell et al. (1970), Koulamas (1998) proposed constructive heuristics using Johnson's two machine approach for flow shop scheduling problems. Gupta (1971) proposed a functional heuristic algorithm to solve large size

* Corresponding author.
E-mail address: samsharma31@gmail.com (S. Sharma)

doi: 10.5267/j.dsl.2021.2.006

flow shop scheduling problems. Bonney and Grundy (1976), Dannenbring (1977), King and Spachis (1980) successfully studied the performance of each constructive algorithm for minimum makespan criteria. Some of the pioneering studies on makespan are given by Stinson and Smith (1982), Nawaz et al. (1983), Taillard (1990), Hundal and Rajgopal (1987). Framinan et al. (2003) constructed 177 initial job ordering procedures and observed that the NEH heuristic outperforms over all other methods in flow shop scheduling problems with minimum makespan. Hejazi and Saghafian (2005) gave a complete literature review of flow shop scheduling problems with makespan criteria. Ruiz and Maroto (2005) validated that among all the constructive heuristics NEH heuristic is the best heuristic for Taillard's benchmarks (1990).

The NEH algorithm can be processed under two phases: sorting and reinsertion. In the first phase, the jobs are sorted in descending order of total processing time and initial feasible schedule is obtained. In the second phase, jobs from the initial sequence are picked up one by one and are arranged corresponding to a minimum value of makespan, to obtain an optimal schedule. Kalczynski and Kamburowski (2007) studied the shortcoming of NEH heuristic that is the tie in job schedules with minimum makespan while arranging the jobs in the second phase. Chakraborty and Laha (2007) proposed the heuristic algorithm for minimizing makespan in permutation flow shop scheduling environment. Dong et al. (2007) proposed NEH-D heuristic with a new initial priority rule and tie breaking strategy based on balanced utilization in the machine system. Kamburowski and Kalczynski (2008, 2009) marked NEH-KK1 and NEH-KK2 heuristics with tie breaking strategy TB_{KK1} and TB_{KK2} of complexity $O(mn^3)$ based on Johnson's heuristic to schedule jobs in system of machines to minimize makespan by giving weightage to the processing time. Rad et al. (2009) proposed new insertion methods that outperform NEH on comparatively large numbers of instances when the comparison is carried over to Taillard Benchmarks (1990). Yin and Lin (2013) proposed constructive heuristic with the definition of tie breaking strategy that involves minimum system idle time priority rule, to solve makespan related flow shop scheduling problems. Vasiljevic and Danilovic (2015) studied different methods for handling ties in an NEH-heuristic FRB1-FRB5 of complexity $O(m^2 n^2)$ for permutation flow shop scheduling problems with makespan criteria. Liu et al. (2017) studied the effects of the first four moments of processing time on the initial job sequence and proposed novel tie breaking rule NEH-LPJ1 of complexity $O(mn^2)$ for NEH heuristic by minimizing front delay time and idle time before tie position.

In this paper an effective tie breaking strategy TB_{SMM} of complexity $O(mn^3)$ is proposed to select the best sequence out of possible partial sequences that correspond to minimum makespan. The efficiency of proposed tie breaking rule is compared with existing tie breaking strategies over Taillard and VRF's benchmarks for permutation flow shop scheduling problems.

The rest of the paper is organized as follows: In section 2, the mathematical model for the considered problem with various notations is developed. The existing tie breaking rules developed on NEH heuristic to solve criteria of makespan are discussed in section 3 and the proposed tie breaking mechanism is elaborated in section 4. In section 5 the test cases and computational results are discussed to demonstrate the effectiveness of the proposed heuristic. Finally conclusions and future developments are presented in section 6**.**

2. Problem definition

Flow shop scheduling is the branch of scheduling in which jobs undergo the available system of machines in a fixed order without any preemption and each of the jobs follow the same route of machines without any interruption. In this section, a mixed integer programming model is developed to obtain processing order of jobs corresponding to minimum makespan.

The considered problem is based on the following assumptions:

1. All the jobs are available for processing at time zero.
2. No two jobs can be processed on the same machine simultaneously.
3. No two machines can process the same job at a time.
4. Processing of jobs is continuous, no machine breakdown is considered.
5. No job can be processed on the same machine twice.
6. Job preemption is not allowed.
7. Job processing time is predetermined.
8. No job passing over is allowed.
9. Set up time is taken as part of processing time not as an independent factor.

The following notations are used in the progress of the paper:

Table 1
Notations

Parameters	Definition
n	number of jobs
m	number of machines
i	index for i^{th} job
j	index for j^{th} machine
p_{ij}	processing time of i^{th} job on j^{th} machine
C_{ij}	completion of i^{th} job on j^{th} machine

T_i	total processing time of ith job
$\prod$	initial sequence of jobs
σ	partial optimal sequence
σ_i	ith job in optimal sequence σ
r	an unscheduled job of initial sequence $\prod$ to be inserted at various positions of partial optimal sequence σ
$C_{\sigma i\, j}$	completion time of ith job of scheduled sequence on jth machine
$C_{\sigma\, j}$	completion time of optimal partial sequence σ on jth machine
g_j	time gap between completion time of sequence σ on machines j and j-1
l	number of jobs in optimal partial sequence σ
$y_{i\,k}$	takes value 1 if job i is scheduled at location k in partial optimal sequence, zero otherwise
$z_{i\,j\,t}$	takes value 1 if job i is processed on machine j at time t, zero otherwise
C_{max}	maximum total elapsed time

Objective function

$$\min C_{max} = \sum_{i=1}^{n} C_{i\,m} \times y_{in} \tag{1}$$

subject to

$$C_{max} - C_{i\,j} \geq 0 \tag{2}$$

$$C_{i\,j} - C_{i\,(j-1)} \geq \sum_{k=1}^{n} p_{i\,j} \times y_{i\,k} \;\; \forall i = 1,2,\dots,n;\; j = 1,2,\dots,m \tag{3}$$

$$C_{i\,j} - C_{(i-1)\,j} \geq \sum_{k=1}^{n} p_{i\,j} \times y_{j\,k} \;\; \forall\, i = 1,2,\dots,n;\; j = 1,2,\dots,m \tag{4}$$

$$\sum_{k=1}^{n} y_{i\,k} = 1 \;\; \forall\, i = 1,2,\dots,n \tag{5}$$

$$\sum_{i=1}^{n} y_{i\,k} = 1 \;\; \forall\, k = 1,2,\dots,n \tag{6}$$

$$\sum_{j=1}^{m} z_{i\,j}(t) = 1 \;\; \forall\, i = 1,2,\dots,n \tag{7}$$

$$\boldsymbol{z_{i\,j}(t)} \leq \sum_{k=1}^{n} y_{i\,k} \;\; \forall\, i = 1,2,\dots.,n, j = 1,2,\dots.,m \tag{8}$$

$$C_{o\,j} = 0 = C_{i\,0} \;\; \forall\, j = 1,2,\dots,m;\, i = 1,2,\dots,n \tag{9}$$

$$z_{i\,j}(t), y_{i\,k} \in \{0,1\} \;\; \forall\, i,k = 1,2,3,\dots,n;\, j = 1,2,3,\dots,m \tag{10}$$

$$C_{i\,j} \geq 0 \;\; \forall i = 0,1,2,\dots.,n;\, j = 0,1,2,3,\dots.m \tag{11}$$

Constraint **(2)** represents the relationship between maximum completion time (C_{max}) and completion time of every job *i* on each of the machine *j*. Constraint (**3**) depicts that the job *i* can't be shifted to machine *j* unless the operation is completed on machine `*j-1*'. Constraint **(4)** guarantees that if the job `*i-1*' processed before job `*i*' on a particular machine `*j*' then processing of job `*i*' can not be started before the processing of job `*i-1*' is completed. Eq. **(5)** assures that the job scheduling is permutation in nature .i.e. each job *i* is scheduled at a unique position `*k*' in the final schedule. Eq. **(6)** depicts that one and only one job is scheduled at position *k* in the final sequence .i.e. no two jobs can be scheduled at the same position. Equation **(7)** clarifies that at a time *t* the job `*i*' can be processed on a single machine. Constraint **(8)** depicts the relation between decision variables of model. Dummy variables introduced in the model to handle the constraint **(3)** and constraint **(4)**, for *j=1* and *i=1* respectively, presented in equation **(9)**. Constraints **(10)** defines the domain of decision variables. Non-negativity restriction on completion time of each job on each machine is presented as output variables of the scheduling model in constraint **(11)**.

3. Tie breaking mechanism

The flow shop scheduling problem with criteria of makespan is NP-hard (Garey 1976). Therefore many heuristics have been developed to solve this problem over the last few decades. Among all the constructive heuristics addressed in literature Nawaz Enscore Ham (NEH) heuristic (Nawaz et al. (1983)) is the most popular and effective heuristic to solve the referent problem. This heuristic involves following steps:

One main limitation of NEH heuristic comes into play when an unscheduled job *r* of initial sequence ($\prod$) is to be inserted at different positions of optimal partial sequence (σ) and the multiple partial sequences giving the same optimal value of makespan are observed .i.e. a situation of tie occurs. For illustration, consider the problem given in Table **2**. Here, when job 3 is inserted at different positions in partial sequence 6-7, then resultant 2 partial sequences 6-3-7 and 6-7-3 have the same minimum makespan. In this situation considering all the partial optimal sequences altogether for inserting the next job at all possible locations may lead to high computation complexity of the algorithm. Therefore, to select one of the best possible optimal partial sequences among all the obtained partial sequences an effective tie breaking technique is needed to be developed. Moreover, due to high complexity of large size FSSP, it is not possible to have a single tie breaking rule to achieve desired optimality, yet by applying a well defined and appropriate tie breaking mechanisms with NEH algorithm, the better results can be expected. The existing tie breaking mechanisms in literature are discussed in following subsections:

3.1 Tie breaking mechanism by Kalczynski and Kamburowski I

TB_{KK1} is the tie breaking mechanism developed in heuristic denoted by NEHKK1. This heuristic is due to Kalczynski and Kamburowski (2007) in which priority rule to obtain initial feasible sequence is completely different from NEH priority rule. In this tie breaking rule the first index of inserted job r for which the minimum value of makespan achieved is considered if $min(a_\sigma, b_r)) \geq min\ (a_r, b_\sigma)$, otherwise the last index for which the minimum value obtained is considered. The parameter $a_r, b_r\ b_\sigma\ and\ a_\sigma$, are given by:

$$a_r = \sum_{j=1}^{m} p_{r\,j} - p_{r\,m} \tag{12}$$

$$b_r = \sum_{j=1}^{m} p_{r\,j} - p_{r\,1} \tag{13}$$

$$a_\sigma = C_{max}(\sigma) - \sum_{i\in\sigma} p_{i\,m} \tag{14}$$

$$b_\sigma = C_{max}(\sigma) - \sum_{i\epsilon\sigma} p_{i\,1} \tag{15}$$

3.2 Tie breaking mechanism by Kalczynski and Kamburowski II

The tie breaking rule denoted by TB_{KK2} is proposed by Kamburowski and Kalczynski (2008) in heuristic NEHKK2 to solve referent problems. In this case whenever the tie occurs the first index is chosen as the optimal position of the inserted job r if $\bar{a}_r \geq \bar{b}_r$, otherwise the last index is considered to be the optimal position. The parameters $\bar{a}_r$ and $\bar{b}_r$ of this tie breaking rule are represented as:

$$\bar{a}_r = \sum_{j=1}^{m} \left[\frac{(m-1)(m-2)}{2} + m - j\right] p_{r\,j} \tag{16}$$

$$\bar{b}_r = \sum_{j=1}^{m} \left[\frac{(m-1)(m-2)}{2} + j - 1\right] p_{r\,j} \tag{17}$$

4. Proposed tie breaking mechanism

The tie breaking rule denoted by TB_{SMM} is proposed in this paper which is related to minimum mean completion time. According to the proposed tie-breaking rule the optimal sequence is chosen with least mean completion time of the jobs on the given system of machines. Mathematically, the proposed tie breaking rule is defined as:

$$TB_{SMM} = \frac{C_{\sigma\,1} + C_{\sigma\,2} + C_{\sigma\,3} + \dots\dots\dots\dots\dots. + C_{\sigma\,m}}{m} \tag{18}$$

where $C_{\sigma j}$ defined as completion time of the job present at the last position of sequence σ on machine `*j'*. The main idea behind the new tie breaking rule is the time gap: $g_j = C_{\sigma\,j} - C_{\sigma\,j-1}$, between completion times of partial optimal sequence σ on machines *j* and *j-1*. When an unscheduled job *r* of initial sequence $\prod$ is appended at *(k+1)th* position of sequence σ of length *k*, the completion time of resultant partial sequence on machine *j-1* is given by $C_{\sigma_k\,j-1}$. Sometimes with new completion time, there arises a situation when $C_{\sigma_{k+1}\,j-1} > C_{\sigma\,j}$, which leads to the idle time on machine *j*. Since there is a direct relation between makespan and idle time, therefore if this situation is continued on all the machines, then the idle time of the machine *m* may lead to an increase in the makespan. However, if the new job in sequence σ with $C_{\sigma_{k+1}\,j-1} < C_{\sigma\,j}$ is appended then makespan will be comparatively less. For large values of g_j the existing gap between the completion times will persist even after appending the job r to the optimal partial sequence σ, i.e. there will be small or even no idle time on machine j. Thus completion time of resultant sequence (obtained by appending the unscheduled job r), on each machine has great influence on makespan. Although this criteria is not related to inserting the job r in optimal partial sequence (σ) rather is related to appending the particular job to sequence, however is the basis for developing a new tie breaking strategy.

4.1 Algorithm proposed

Algorithm *NEHSMM* is proposed, by combining the tie breaking strategy TB_{SMM} with *NEH* heuristic to obtain the optimal sequence of jobs on an available system of machines with the objective of minimum makespan in the FSS environment. Further, a high level flow chart of proposed heuristic is presented in **Fig. 1.**

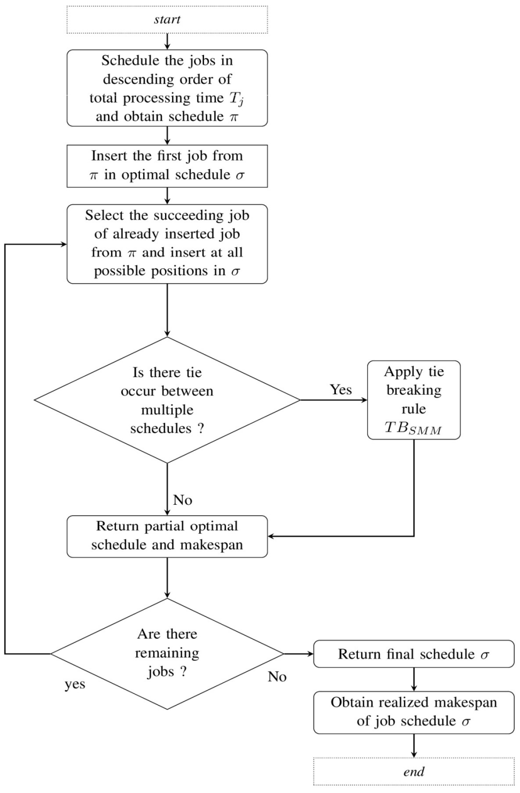
start
Schedule the jobs in descending order of total processing time T_j and obtain schedule π
Insert the first job from π in optimal schedule σ
Select the succeeding job of already inserted job from π and insert at all possible positions in σ
Is there tie occur between multiple schedules ?
Yes
Apply tie breaking rule TB_{SMM}
No
Return partial optimal schedule and makespan
Are there remaining jobs ?
yes
No
Return final schedule σ
Obtain realized makespan of job schedule σ
end

Fig. 1. Flow chart of proposed heuristic

The following steps are involved in overall computation by proposed heuristic:

Step 1: Calculate total processing time $T_i = \sum_{j=1}^{m} p_{i\,j}\ \forall\ i = 1,2, \dots, n.$

Step 2: Arrange all the jobs in descending order of T_i and obtain the initial feasible sequence $\prod$ of jobs.

Step 3: Consider the first two jobs from sequence $\prod$ and arrange them corresponding to minimum makespan to obtain partial sequence σ. In case of tie use the proposed tie breaking strategy TB_{SMM}, if tie still exist then the position of the jobs corresponding to first obtained partial sequence σ with minimum makespan is considered. Set length of resultant sequence $l=2$.

Step 4: Insert the next unscheduled job from initial feasible sequence $\prod$ in current optimal partial sequence σ and again arrange the jobs corresponding to minimum makespan (elapsed time), without violating the order of already scheduled jobs. When the choice is to be made among the multiple optimal partial sequences corresponding to minimum makespan, the proposed tie breaking strategy (**4.1**) is implemented to determine the best partial sequence. After insertion of new job increase the length l of sequence σ by 1.

Step 5: Repeat Step 4 until $l \geq n$.

4.2 Computation complexity

In this paper the tie breaking mechanism is introduced by calculating the mean completion time of all the jobs after the insertion of job at all the possible positions which clearly does not alter the computation complexity of NEH heuristic and it remains $O(mn^3)$.

4.3 Numerical illustration

In this section we have shown implementation of proposed tie breaking rule with a simple numerical illustration as discussed below:

Table 2
Data of processing time of 10 jobs on 5 machines

	Machines				
Jobs	M_1	M_2	M_3	M_4	M_5
J_1	79	67	10	48	52
J_2	40	40	57	21	54
J_3	48	93	49	11	79
J_4	16	23	19	2	38
J_5	38	90	57	73	3
J_6	76	13	99	98	55
J_7	73	85	40	20	85
J_8	34	6	27	53	21
J_9	38	6	35	28	44
J_{10}	32	11	11	34	27

The first step of the algorithm mentioned in Section **4.1**, is to arrange the jobs in descending order of total processing time T_i. Here, for each job total processing time is given by: ***$T_1 = 256$, $T_2 = 212$, $T_3 = 280$, $T_4 = 98$, $T_5 = 261$, $T_6 = 341$, $T_7 = 303$, $T_8 = 141$, $T_9 = 151$, $T_{10} = 115$.***
Therefore corresponding initial feasible solution is **:**

J_6-J_7-J_3-J_5 -J_1-J2-J_9-J_8-J_{10}-J_4.

To obtain the final optimal schedule σ follow the Step 2 to Step 5. Following step 2 first two jobs J_6 and J_7, are considered. arranging these jobs simultaneously two possible schedules **(J_6 , J_7)** and **(J_7, J_6)** with corresponding makespan **426 units** and **450 units**, respectively. So, the partial optimal schedule **(J_6 , J_7)** is selected. Now consider the next job **J_3** from the initial feasible job schedule and insert it in all the possible positions of partial optimal schedule σ. The obtained job schedules are **(J_3,J_6,J_7)**, **(J_6,J_3,J_7)** and **(J_6,J_7,J_3)**. Here, schedules **(J_6,J_3,J_7)** and **(J_6,J_7,J_3)** are optimal partial schedules with minimum makespan **505 units**. Following the proposed procedure when the tie occurs between the schedules **(J_6,J_3,J_7)** and **(J_6,J_7,J_3)**, the value of TB_{SMM} obtained, **341.6 units** and **358.4 units,** respectively. Therefore, the resultant schedule obtained by inserting job **J_3**, into second position is considered as optimal partial schedule. Clearly computation of TB_{SMM} does not lead to an increase in complexity of the original NEH heuristic as no extra loop is required to be implemented in original code. Following the various steps of proposed heuristic the final optimal job schedule is: **(J_4, J_2, J_{10}, J_6, J_3, J_1, J_7, J_8, J_9, J_5)** with makespan **713 units** which is comparatively less than that by NEH i.e. **726 units.** Implementation of tie breaking mechanism in generating the final optimal job schedule is presented in **Table 3**.

Table 3
The best partial sequence constructed by the improved NEH based heuristic after inserting each job of π

Iteration	Inserting job	σ	C_{max}	Tie situation
1	J_7	J_6-J_7	426	No
2	J_3	J_6- J_3-J_7	505	Yes
3	J_5	J_6- J_3-J_7-J_5	525	No
4	J_1	J_6- J_3- J_1-J_7-J_5	592	Yes
5	J_2	J_2-J_6-J_3-J_1-J_7-J_5	632	Yes
6	J_9	J_2-J_6-J_3-J_1-J_7-J_9-J_5	652	No
7	J_8	J_2-J_6-J_3-J_1-J_7-J_8-J_9-J_5	673	Yes
8	J_{10}	J_2-J_{10}-J_6-J_3-J_1-J_7-J_8-J_9-J_5	697	No
9	J_4	J_4-J_2-J_{10}-J_6-J_3-J_1-J_7-J_8-J_9-J_5	713	No

5. Computational experiments

In this section, the computational experiments are carried out in MATLAB over Intel(R) core(TM) i5 CPU @ 2.20 GHZ computer with 8GB RAM. To confirm the effectiveness of proposed tie breaking rule and to test the performance of proposed heuristic, two sets of tests, including comparison tests of the tie-breaking rule and proposed heuristic on Taillard (Taillard, 1990) and VRF (Vallada et al., 2015) benchmarks are performed. These benchmarks consist of 600 problem instances with varying sizes of number of jobs and number of machines. The processing time is uniformly distributed [1,99] for each job *i* on each machine *j*. Further, for the comparison test of tie breaking rule two most important tie breaking mechanism TB_{KK1} (Kalczynski & Kamburowski, 2007) and TB_{KK2} (Kalczynski & Kamburowski, 2008) with same complexity $O(n^3m)$ have been considered, however many other improved heuristics with different complexities are available in the literature. To validate performance of proposed heuristic in solving the referred problem the results obtained for Taillard and VRF benchmarks of scheduling are compared against existing well known heuristics including NEH (Nawaz et al. (1984)), DE (Onwubolu & Davendra (2006)), NEHKK1 (Kalczynski & Kamburowski (2007)), NEHKK2 (Kalczynski & Kamburowski, 2008), MOD (Semanco & Modrak (2012)), CL_{WTS} (Ying & Lin (2013)), GC (Gupta and Chauhan (2015)), NEHAB1 (Baskar (2016)). The different parameters used in complete evaluation are given as:

Table 3
Parameters and Measurements

Parameters	Measurements
Number of jobs	10, 20, 30, 40, 50, 60, 100, 200, 300, 400, 500, 600, 700, 800
Number of machines	5, 10, 15, 20, 40, 60
Processing duration	Taillard's and VRF standard benchmarks

In order to measure the solution quality, the average relative percentage deviation *(ARPD)* is given by,

$$ARPD = \sum_{k=1}^{10} \frac{HS_k - BS_k / BS_k}{10} \times 100\% \tag{19}$$

is considered as a tool to measure performance, where HS_k is makespan computed by particular heuristic for problem instance k and BS_k is the best solution value for this particular instance provided by Taillard (1990) and VRF (Vallada et. al. 2015). The detail of these tests is discussed in the following subsections.

۵/۱ *Comparison test of tie-breaking rule*

The pursuance of proposed tie breaking rule is compared with tie breaking rules in reference, i.e. TB_{KK1} and TB_{KK2} by implementing them in an NEH heuristic whenever time

Table 4
ARPD values of different tie breaking rules on Taillard instances

Problem	NEH	TB_{KK1}	TB_{KK2}	TB_{SMM}
20×5	3.31	2.65	2.73	**2.40**
20×10	4.60	**4.31**	**4.31**	4.45
20×20	3.73	**3.41**	**3.41**	3.77
50×5	0.72	**0.59**	0.66	0.66
50×10	5.07	4.83	4.87	**4.69**
50×20	6.65	6.37	6.41	**6.18**
100×5	0.53	0.42	0.41	**0.41**
100×10	2.21	2.27	**1.77**	2.04
100×20	5.34	5.31	**5.28**	5.69
200 ×10	1.26	**1.12**	1.17	1.28
200×20	4.41	4.24	4.17	**3.92**
500×20	2.07	**2.00**	2.02	**2.00**
Average	**3.33**	**3.13**	**3.11**	**3.09**

Table 5
ARPD values for different tie breaking rules on VRF benchmarks

Problem		NEH		TB_{KK1}		TB_{KK2}		TB_{SMM}	
Small	Large	Small	Large	Small	Large	Small	Large	Small	Large
10×5	100× 20	**2.18**	5.71	2.21	5.80	2.23	5.76	2.47	**5.67**
10× 10	100× 40	**1.63**	5.67	1.74	5.52	1.74	5.44	2.03	**5.07**
10× 15	100× 60	1.53	4.95	1.54	4.92	1.44	4.86	**1.31**	**4.77**
10× 20	200× 20	1.99	4.23	1.54	**4.00**	1.54	4.09	1.98	4.04
20× 5	200× 40	1.51	4.71	**1.19**	4.55	2.09	4.51	1.31	4.53
20× 10	200× 60	4.82	4.55	4.71	4.40	4.71	4.40	4.79	**4.24**
20× 15	300× 20	4.33	2.99	4.09	2.80	4.09	2.79	**4.04**	2.88
20× 20	300× 40	4.12	4.08	3.69	3.89	**3.64**	3.87	3.89	**3.79**
30× 5	300× 60	1.43	3.93	**1.09**	**3.73**	1.24	3.76	1.22	3.94
30× 10	400× 20	5.26	2.58	5.87	2.33	5.18	2.40	**5.44**	**2.24**
30× 15	400× 40	5.83	3.66	5.75	3.50	5.76	3.45	**5.32**	**3.43**
30× 20	400× 60	5.48	3.56	5.43	**3.31**	5.43	3.33	**5.30**	3.44
40× 5	500× 20	1.09	2.27	0.73	1.95	0.81	1.91	**0.72**	**1.87**
40× 10	500× 40	4.97	3.21	5.07	3.05	5.02	3.01	**4.38**	3.03
40× 15	500× 60	6.05	3.12	6.14	3.13	6.05	3.09	**5.54**	**3.03**
40× 20	600× 20	**5.14**	1.57	5.68	1.43	5.61	1.56	5.25	**1.26**
50× 5	600× 40	0.55	3.13	0.47	2.86	**0.32**	2.92	0.59	**2.76**
50× 10	600× 60	4.58	2.93	4.43	2.82	**4.22**	2.87	4.62	**2.86**
50× 15	700× 20	6.52	1.41	6.46	1.27	6.44	1.24	**6.20**	**1.23**
50× 20	700× 40	**5.96**	2.77	5.98	2.76	5.99	2.65	6.20	**2.48**
60× 5	700× 60	0.89	2.75	0.70	2.76	**0.67**	2.81	0.86	**2.72**
60× 10	800× 20	3.96	1.23	**3.89**	1.19	3.96	1.19	4.03	**1.11**
60× 15	800× 40	5.79	2.43	5.76	2.45	5.78	2.50	**5.15**	**2.34**
60× 20	800× 60	6.45	2.70	**6.25**	2.69	6.42	2.67	6.49	**2.66**
Average		3.83	3.34	3.72	3.21	3.76	3.21	**3.71**	**3.15**
		3.59		3.48		3.49		**3.43**	

It has been found that the performance of the proposed tie breaking rule does not remain consistent on all the benchmarks due to random data but it remains favorable in the final evaluation. From Table **5**, the average ARPD value for original

NEH heuristic is **3.33** for Taillard's benchmarks where as an implementing the tie breaking rules TB_{KK1} and TB_{KK2} the average ARPD values reduce to 3.13 and 3.11 respectively, whereas proposed tie breaking strategy outperforms these referred tie breaking rules with average ARPD 3.09 and minimum value of ARPD on 6/12 problem instances. From Table **6**, we observed that by applying a proposed heuristic on VRF-test beds, the significantly better results are obtained on **27/48** problem instances with average ARPD value **3.43** (**3.71** on small instances and **3.15** on large instances) which are comparatively lesser than another tie breaking rules taken in reference.

5.2 Comparison test for heuristics

For comparison the NEH (Nawaz et.al.(1984)), DE (Onwubolu and Davendra (2006)), NEHKK1 (Kalczynski and Kamburowski (2007)), NEHKK2 (Kalczynski and Kamburowski (2008)), MOD (Modrak (2012)), CL_{WTS} (Ying and Lin (2013)), GC (Gupta and Chauhan (2015)), NEHAB1 (Baskar (2016)) and proposed heuristic NEHSMM are implemented first on Taillard test beds and result is reported in Table **7. NEHSMM** heuristic dominates over other referred heuristics on **6/12** problem instances with ARPD **3.09**, whereas it is **3.17** for **NEHKK1**, **3.14** for **NEHKK2**, **8.35** for **GC**, **10.49** for **MOD**, **3.18** for **CL_{WTS}, 3.14** for **NEHAB1** and **9.25** for **DE.**

Table 6
ARPD values of different heuristics on Taillard's test bed

Problem	NEH	NEHKK1	NEHKK2	GC	MOD	CL_{WTS}	NEHAB1	DE	NEHSMM
20×5	3.31	2.81	2.48	7.75	8.89	2.84	2.47	3.98	**2.40**
20×10	4.60	4.43	**4.17**	10.62	13.76	4.54	4.50	5.86	4.45
20×20	3.73	**3.37**	3.57	8.76	13.41	3.77	3.03	4.53	3.77
50×5	0.72	0.67	**0.44**	4.09	6.27	0.61	0.64	4.28	0.66
50×10	5.07	5.46	5.38	11.49	13.12	5.17	5.07	11.48	**4.69**
50×20	6.65	6.25	6.22	12.62	15.41	6.32	6.25	14.73	**6.18**
100×5	0.53	0.41	0.22	2.93	4.08	0.41	0.52	4.27	0.41
100×10	2.21	2.08	2.28	7.69	9.29	2.07	1.99	10.42	**2.04**
100×20	5.34	**5.23**	11.66	14.35	9.29	5.47	5.39	16.08	5.39
200 ×10	1.26	1.32	1.32	5.35	6.92	1.61	1.39	8.34	**1.28**
200×20	4.41	4.17	4.25	7.95	12.52	4.04	4.37	15.44	**3.92**
500×20	2.07	**1.95**	2.08	6.64	7.86	1.96	2.09	11.58	2.00
Average	3.33	3.17	3.14	8.35	10.49	3.23	3.14	9.25	**3.09**

On VRF instances comparison is carried out only among the **NEH, NEHKK1, NEHKK2** and proposed ***NEHSMM*** heuristics. The reason behind taking only these heuristics for evaluation is that only these heuristics compete with our proposed heuristic. As shown in Table **8**, the proposed heuristic ***NEHSMM*** outperforms with ARPD **3.45** (it is **3.71** for small instances and **3.15** for large instances) with significant performance on **21/48** problem instances, whereas it is **14/48** problems, **13/48** and **2/48** problems for ***NEHKK1***, ***NEHKK2*** and ***NEH*** heuristics respectively.

Table 7
ARPD values of different heuristics on VRF instances

Problem		NEH		TB_{KK1}		TB_{KK2}		TB_{SMM}	
Small	Large	Small	Large	Small	Large	Small	Large	Small	Large
10×5	100× 20	2.18	5.71	**1.99**	**5.37**	2.34	5.70	2.42	5.67
10× 10	100× 40	1.63	5.67	**1.59**	5.46	2.39	5.42	2.03	**5.07**
10× 15	100× 60	1.53	4.95	1.48	4.72	1.86	4.86	**1.31**	4.77
10× 20	200× 20	1.99	4.23	1.54	4.18	**1.51**	4.15	1.98	**4.04**
20× 5	200× 40	1.51	4.71	1.59	**4.49**	2.43	4.76	**1.31**	4.53
20× 10	200× 60	4.82	4.55	5.07	4.31	4.77	4.31	4.79	**4.24**
20× 15	300× 20	4.33	2.99	4.23	2.90	4.10	**2.82**	**4.04**	2.88
20× 20	300× 40	4.12	4.08	3.91	3.80	4.22	3.99	**3.89**	**3.79**
30× 5	300× 60	1.43	3.93	**0.92**	**3.84**	1.07	4.24	1.22	3.94
30× 10	400× 20	5.26	2.58	**5.09**	2.39	5.22	2.27	5.44	**2.24**
30× 15	400× 40	5.83	3.66	6.02	3.70	5.53	3.61	**5.32**	**3.43**
30× 20	400× 60	5.41	3.56	5.44	**3.42**	5.61	3.49	**5.30**	3.44
40× 5	500× 20	1.09	2.27	0.61	1.84	**0.45**	**1.73**	0.72	1.87
40× 10	500× 40	4.97	3.21	4.78	3.09	5.01	3.02	**4.38**	**3.01**
40× 15	500× 60	6.05	3.12	6.14	3.09	6.33	3.19	**5.54**	**3.03**
40× 20	600× 20	**5.14**	1.57	5.35	**1.50**	5.37	1.52	5.25	1.56
50× 5	600× 40	0.55	3.13	0.45	3.00	**0.37**	2.88	0.59	**2.76**
50× 10	600× 60	4.58	2.93	4.30	2.88	**3.48**	2.94	4.62	**2.86**
50× 15	700× 20	6.52	1.41	6.36	1.33	**6.19**	**1.11**	**6.19**	1.23
50× 20	700× 40	5.96	2.77	6.38	2.68	**6.10**	**2.45**	6.20	2.48
60× 5	700× 60	0.89	2.75	0.77	**2.68**	0.88	2.76	**0.76**	2.72
60× 10	800× 20	**3.96**	1.23	4.06	1.13	4.02	**0.95**	4.03	1.11
60× 15	800× 40	5.79	2.43	5.69	2.44	5.96	**2.32**	**5.15**	2.34
60× 20	800× 60	6.45	2.70	**6.08**	2.58	6.68	**2.56**	6.49	2.66
Average		3.83	3.34	3.74	3.20	3.80	3.20	**3.71**	**3.15**
		3.59		3.49		3.50		**3.45**	

5.3 Test for significance

To test whether the difference in makespan and ARPD values for the referred heuristic algorithms and proposed heuristic is significant on Taillard and VRF benchmarks, the paired sample t-test is carried out at 95% level of significance and the results are shown in Table **10** and Table **11**. The test is also carried over the combined data of both referred test beds in Table **9**. As depicted in Table **9** for confidence level $\alpha = 0.05$, the value of p less than 0.05 demonstrates that there is significant difference present between the proposed heuristic and referred heuristics. It is clear that the difference between pairs **NEHSMM** and **NEH**, **NEHSMM** and **NEHKK1**, **NEHSMM** and **NEHKK2** is significant. Similarly, two sided paired sample t-test is carried over ARPD values of **NEHSMM** and for all other referred heuristics. This test is carried over both the VRF and Taillard's benchmarks and significance values are presented in the significance column of Table **10** and Table **11**, respectively.

Table 8
Paired sample t-test for combined VRF and Taillard benchmarks

Parameters	Pair	Mean	SE_m	CI-lower	CI-upper	t	P
Makespan	NEH-NEHSMM	16.19	2.534	12.01	21.96	6.703	0.000
	KK1-NEHSMM	4.938	2.337	0.346	9.530	2.113	0.035
	KK2-NEHSMM	1.489	2.49	-3.409	6.388	0.597	0.049
ARPD	NEH-NEHSMM	0.166	0.031	0.106	0.228	5.468	0.000
	KK1-NEHSMM	0.031	0.035	-0.038	0.102	0.922	0.036
	KK2-NEHSMM	0.559	0.044	-0.034	0.146	1.247	0.021

Table 9
Paired sample t-test result on Taillard's Benchmarks

Parameters	Pair	Mean	SE_m	CI-lower	CI-upper	t	P
Makespan	NEH-NEHSMM	10.40	3.577	3.324	17.491	2.91	0.004
	KK1-NEHSMM	1.125	3.316	-5.442	7.692	0.339	0.035
	KK2-NEHSMM	1.275	3.907	-6.461	9.011	0.326	0.047
	GC-NEHSMM	337.5	31.01	276.08	398.9	10.88	0.000
	MOD-NEHSMM	254.7	2099.9	1611.1	6705.2	11.08	0.000
	CL_{WTS}-NEHSMM	9.967	3.494	-1.88	5.95	1.277	0.024
	NEHAB1-NEHSMM	7.250	3.804	-0.282	14.78	1.906	0.049
	DE-NEHSMM	447.2	40.76	366.5	527.9	10.97	0.000
ARPD	NEH-NEHSMM	0.225	0.082	0.405	0.456	2.761	0.019
	KK1-NEHSMM	0.051	0.091	-0.148	0.031	1.973	0.058
	KK2-NEHSMM	0.012	0.084	-0.172	0.196	0.148	0.050
	GC-NEHSMM	5.253	0.497	4.158	6.349	1.056	0.000
	MOD-NEHSMM	6.968	0.603	5.641	8.295	11.56	0.024
	CL_{WTS}-NEHSMM	0.082	0.184	0.013	0.151	2.620	0.050
	NEHAB1-NEHSMM	0.012	0.084	-0.172	0.196	0.148	0.000
	DE-NEHSMM	6.149	1.091	3.748	8.551	5.635	0.0000

Table 10
Paired sample t-test results on VRF benchmarks

Parameters	Pair	Mean	SE_m	CI-lower	CI-upper	t	p
Makespan	NEH-NEHSMM	22.95	3.011	17.035	28.86	7.622	0.000
	KK1-NEHSMM	6.791	2.799	1.290	12.29	2.462	0.016
	KK2-NEHSMM	3.292	2.999	-2.605	9.186	1.097	0.027
ARPD	NEH-NEHSMM	0.152	0.033	0.087	0.214	4.701	0.000
	KK1-NEHSMM	0.027	0.038	-0.048	0.103	0.725	0.047
	KK2-NEHSMM	0.067	0.052	-0.038	0.172	1.278	0.208

5.4 Computation efficiency

In order to test computation efficiency of the proposed heuristic over the other heuristics taken in reference, average relative percentage deviation (ARPD), average CPU time (ACPU), average relative computation time (ARCT) is computed simultaneously. The ACPU and ARCT (Fernandez et al., 2017) are defined as follows:

$$ACPU = \frac{\sum_t CPU_{t\,h}}{I} \tag{20}$$

$$ARCT = \frac{\sum_t {CPU_{t\,h} - ACT_t}/{ACT_t}}{I} + 1 \tag{21}$$

where, I is the total number of instances, $CPU_{t\,h}$ is computation time taken by heuristic h to solve the instance t, $ACT_{\ t} = \frac{\sum_h CPU_{t\,h}}{H}$, H=number of heuristics considered.

ACPU and *ARCT* are two performance measures of computation efficiency of heuristic whereas *ARPD* is used to verify the solution quality of heuristic h. To get the unbiased results, each of the heuristic is run over the both Taillard and VRF benchmarks independently five times and then mean is taken as final *CPU* time. The results of each heuristic on Taillard's and VRF benchmarks are presented in Table **12**. From Table **12**, it is clear that the proposed heuristic **NEHSMM** is outperforming the other referred heuristics significantly, however is not efficient on a scale of computation. This shows as the more time is spent on sequencing the jobs better results can be obtained.

Table 11
Comparison on basis of ARPD, ACPU, ARCT

Heuristic	**Taillard**			**VRF**		
	ARPD	**ACPU**	**ARCT**	**ARPD**	**ACPU**	**ARCT**
NEH	3.325	2.046	0.945	3.592	19.345	0.784
NEHKK1	3.171	2.046	1.016	3.502	19.395	1.052
NEHKK2	3.142	2.094	1.039	3.502	19.365	1.035
GC	8.352	2.165	1.210	-	-	-
MOD	10.49	2.861	1.732	-	-	-
NEHAB1	3.148	2.974	0.526	-	-	-
NEHSMM	3.099	2.092	0.999	3.446	19.503	1.081

Also computation effectiveness of each heuristic competitive to proposed heuristic is depicted in Fig. **2** to Fig. **5**.

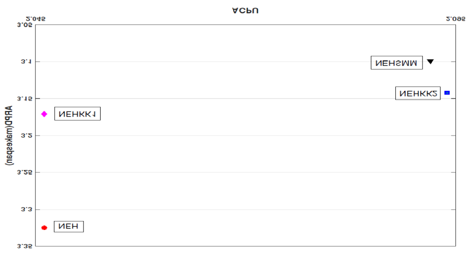

Fig. 2. ARPD vs ACPU on Taillard's benchmarks

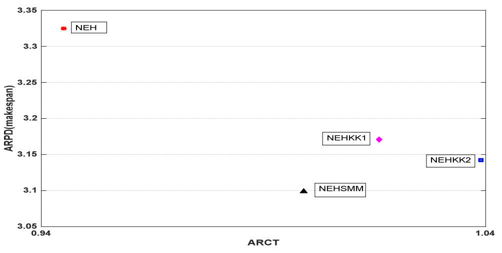

Fig. 3. ARPD vs ARCT on Taillard's benchmarks

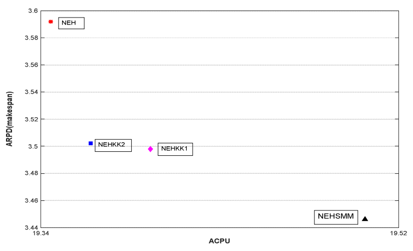

Fig. 4. ARPD vs ACPU on VRF benchmarks

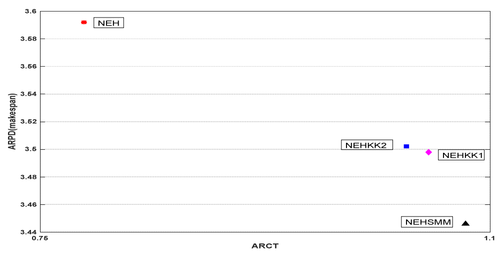

Fig. 5. ARPD vs ARCT on VRF benchmarks

6. Conclusion

In this work a novel tie-breaking rule for NEH heuristic without increasing its computational complexity is introduced. The computation tests are carried over the Taillard's and VRF-hard's benchmarks for permutation flowshop scheduling. Test results show that the proposed heuristic ***NEHSMM*** outperforms the other referred constructive heuristics significantly. Computation efficiency results depict that, however the use of tie breaking strategies with the NEH heuristic may lead to increase in computation time yet the best results of makespan can be expected. In conclusion, a new heuristic ***NEHSMM*** is proposed for the permutation flowshop scheduling problems with minimum makespan criteria. The effectiveness of the proposed heuristic has also been validated on both Taillard and VRF test beds. The future work on this strategy is to define more effective strategies and tie breaking techniques for NEH technique so as to obtain the best results of makespan in flow shop scheduling environment. Moreover the technique developed here can also be used to solve the tie breaking strategy occurs is bicriteria flow shop scheduling problem with makespan as primary objective so as to get better results of this parameter.

Acknowledgement

One of authors acknowledges the financial support provided by council of scientific and industrial research, New Delhi, India, in the form of SRF through grant number 09/135(0766)/2017-EMR-I.

References

Baskar, A. (2016). Revisiting the NEH algorithm-the power of job insertion technique for optimizing the makespan in permutation flow shop scheduling. *International Journal of Industrial Engineering Computations*, *7*(2), 353-366.

Bonney, M. C., & Gundry, S. W. (1976). Solutions to the constrained flowshop sequencing problem. *Journal of the Operational Research Society*, *27*(4), 869-883.

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management science*, *16*(10), B-630.

Chakraborty, U. K., & Laha, D. (2007). An improved heuristic for permutation flowshop scheduling. *International Journal of Information and communication technology*, *1*(1), 89-97.

Dannenbring, D. G. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, *23*(11), 1174-1182.

Dong, X., Huang, H., & Chen, P. (2008). An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research*, *35*(12), 3962-3968.

Fernandez-Viagas, V., Ruiz, R., & Framinan, J. M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research*, *257*(3), 707-721.

Framinan, J. M., Leisten, R., & Rajendran, C. (2003). Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. *International Journal of Production Research*, *41*(1), 121-148.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*(2), 117-129.

Gupta, J. N. (1971). A functional heuristic algorithm for the flowshop scheduling problem. *Journal of the Operational Research Society*, *22*(1), 39-47.

Gupta, A., & Chauhan, S. (2015). A heuristic algorithm for scheduling in a flow shop environment to minimize makespan. *International Journal of Industrial Engineering Computations*, *6*(2), 173-184.

Hundal, T. S., & Rajgopal, J. (1988). An extension of Palmer's heuristic for the flow shop scheduling problem. *International Journal of Production Research*, *26*(6), 1119-1124.

Ignall, E., & Schrage, L. (1965). Application of the branch and bound technique to some flow-shop scheduling problems. *Operations research*, *13*(3), 400-412.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, *1*(1), 61-68.

Kalczynski, P. J., & Kamburowski, J. (2009). An empirical analysis of the optimality rate of flow shop heuristics. *European Journal of Operational Research*, *198*(1), 93-101.

Kalczynski, P. J., & Kamburowski, J. (2008). An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers & Operations Research*, *35*(9), 3001-3008.

King, J. R., & Spachis, A. S. (1980). Heuristics for flow-shop scheduling. *International Journal of Production Research*, *18*(3), 345-357.

Koulamas, C. (1998). A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research, 105*(1), 66-71.

Liu, W., Jin, Y., & Price, M. (2017). A new improved NEH heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics*, *193*, 21-30.

Nawaz, M., Enscore Jr, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

Onwubolu, G., & Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research*, *171*(2), 674-692.

Page, E. S. (1961). An approach to the scheduling of jobs on machines. *Journal of the Royal Statistical Society: Series B (Methodological)*, *23*(2), 484-492.

Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. *Journal of the Operational Research Society*, *16*(1), 101-107.

Rad, S. F., Ruiz, R., & Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *Omega*, *37*(2), 331-345.

Reza Hejazi*, S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, *43*(14), 2895-2929.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, *165*(2), 479-494.

Semanco, P., & Modrak, V. (2012). A comparison of constructive heuristics with the objective of minimizing makespan in the flow-shop scheduling problem. *Acta Polytechnica Hungarica*, *9*(5), 177-190.

STINSON, J. P., & SMITH, A. W. (1982). A heuristic programming procedure for sequencing the static flowshop. *The International Journal Of Production Research*, *20*(6), 753-764.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278-285.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational research*, *47*(1), 65-74.

Vallada, E., Ruiz, R., & Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, *240*(3), 666-677.

Vasiljevic, D., & Danilovic, M. (2015). Handling ties in heuristics for the permutation flow shop scheduling problem. *Journal of Manufacturing Systems*, *35*, 1-9.

Ying, K. C., & Lin, S. W. (2013). A high-performing constructive heuristic for minimizing makespan in permutation flowshops. *Journal of Industrial and Production Engineering*, *30*(6), 355-362.