# A new Nawaz–Enscore–Ham-based heuristic for permutation flow-shop problem with bicriteria of makespan and machine idle-time

Liu, W., Jin, Y., & Price, M. (2016). A new Nawaz–Enscore–Ham-based heuristic for permutation flow-shop problem with bicriteria of makespan and machine idle-time. *Engineering Optimization*, *48*(10), 1808-1822. https://doi.org/10.1080/0305215X.2016.1141202

**Published in:**
Engineering Optimization

**Document Version:**
Early version, also known as pre-print

# A New Nawaz–Enscore–Ham based Heuristic for Permutation Flowshop Problem with Bicriteria of Makespan and Machine Idle-time

Weibo Liu, Yan Jin*, Mark Price

*School of Mechanical and Aerospace Engineering, Queen's University Belfast, Ashby Building, Belfast, BT9 5AH, UK*

y.jin@qub.ac.uk

# A New Nawaz–Enscore–Ham based Heuristic for Permutation Flowshop Problem with Bicriteria of Makespan and Machine Idle-time

A new heuristic based on Nawaz–Enscore–Ham (NEH) algorithm is proposed for solving permutation flowshop scheduling problem in this paper. A new priority rule is proposed by accounting for the average, mean absolute deviation, skewness and kurtosis, in order to fully describe the distribution style of processing times. A new tie-breaking rule is also introduced for achieving effective job insertion for the objective of minimizing both makespan and machine idle-time. Statistical tests illustrate better solution quality of the proposed algorithm, comparing to existing benchmark heuristics.

Keywords: heuristic; flow shop; scheduling; makespan; idle-time

## 1 Introduction

Flow shop scheduling is an active research area in manufacturing, as it has many interesting industrial applications and is also an attractive field of theoretical study (Yenisey and Yagmahan 2014). Industrial applications can be found in automotive manufacturing (Xu and Zhou 2009; Framinan et al. 2014), integrated circuit fabrication (Liu and Chang 2000), photographic film production (Aghezzaf and Van Landeghem 2002), pharmaceutical and agro-food industries ( Boukef, Benrejeb, and Borne 2007). The Flow shop scheduling problem has been proved as NP-hard when the machine number is larger than 2 (Garey, Johnson, and Sethi 1976). It has become a rather challenging problem not only in research, but also for industrial practice. To simplify the problem, permutation flow shop scheduling problem (PFSP), in which the order of jobs passing through every machine is always kept the same, is often used as it is a special case of flow shop problem. PFSP is also proved to be NP-hard (Garey and Johnson 1979) and many methods have been proposed to solve PFSP with a criterion of minimizing makespan or maximum job lateness. Some successes have been obtained

(Allahverdi 2004; Chandra, Mehta, and Tirupati 2009).

The NEH (Nawaz, Enscore Jr., and Ham 1983) heuristic has been regarded as the best algorithm to solve PFSP and many heuristics based on NEH have been proposed with the objective of minimizing makespan or total flow time, and have demonstrated improved performance. NEHKK1 (Kalczynski and Kamburowski 2008), NEH-D (Dong, Huang, and Chen 2008), NEHKK2 (Kalczynski and Kamburowski 2009) and NEHFF (Fernandez-Viagas and Framinan 2014) are currently popular constructive heuristic algorithms, representing the state of the art in the field. Single objective scheduling is employed in most existing heuristics and the objectives only focus on makespan (Gao and Pan 2011; Liu, Fang, and Lin 2013), total flow time (Pan and Ruiz 2013; Msakni et al. 2015), or total tardiness (Yenisey and Yagmahan 2014). However, a single objective may not be good enough to represent reality as most of real life scheduling problems naturally involve multiple objectives. The current trend is to apply multiple objectives while further improving existing heuristics by new approaches. To start to address this problem, this paper studies the PFSP with two objectives, i.e., minimization of both makespan and machine idle-time. Minimizing makespan is to deliver orders as soon as possible, while minimizing idle-time could help improve machine utilization. In the past, minimizing makespan has been mistakenly regarded as equivalent to minimizing machine idle-time, but recent research by (Liu, Jin, and Price 2014) has shown that although they are related, they are clearly different and in fact can be in conflict with each other. Herein, a novel heuristic algorithm based on NEH approach is proposed by utilizing a new priority rule and a new tie breaking method with the two objectives. Its effectiveness is validated through statistical tests with common benchmarks (Taillard 1993; Vallada, Ruiz, and Framinan

2015) by comparing to existing dominating algorithms including NEH, NEHKK1, NEH-D, NEHKK2 and NEHFF.

In literature, many heuristic methods have been introduced to solve PFSP. Early examples can be found in (Johnson 1954; Palmer 1965; Gupta 1971; Campbell *et al*. 1970; Gupta 1976; Dannenbring 1977). Nawaz *et al.* (1983) proposed a ground-breaking algorithm on which many heuristics were introduced for PFSP (Framinan *et al*. 2004; Ruiz and Maroto 2005; Gupta and Stafford 2006). Recent advance lies in the proposition of NEHKK1 (Kalczynski and Kamburowski 2008), NEH-D (Dong et al. 2008), NEHKK2 (Kalczynski and Kamburowski 2009) and NEHFF (Fernandez-Viagas and Framinan 2014), all of which demonstrated improved performance.

Typically, two key steps are required in these NEH-based heuristic algorithms. The first step is to sort all jobs with one priority rule to form the initial partial sequence, and the second step is to insert the rest jobs one by one to the existing sequence for achieving certain objective. Since ties often occur in the second step, the tie-breaking method is also crucial to the performance of the heuristic algorithm. In NEH algorithm, the priority rule is based on the non-increasing sum of processing times, and job insertion is with the objective to minimize makespan, but no tie-breaking method is used. Once a tie occurs, usually the first feasible position is selected. Based on the first or the second step, many improved NEH heuristics were developed. Nagano and Moccellin (2002) developed a priority rule according to the non-increasing difference between total processing times and job waiting time, and it was competitive compared to NEH. Low *et al*. (2004) developed the MNEH algorithm by introducing a priority rule according to the descending sum of artificial processing times and a tie-breaking rule that chose the position with the least idle-time on the bottleneck machine. Kalczynski and Kamburowski (2007) developed a series of NEH modifications

including NEHKK, NEHKK1 and NEHKK2 based upon the concept of Johnson's rule. Dong et al. (2008) introduced the NEH-D heuristic, which includes a priority rule by taking account of processing time variation combined with mean value and a tie-breaking rule choosing the position with the least machine utilization variation. Fernandez-Viagas and Framinan (2014) proposed a tie-breaking rule aiming to minimize total idle-time in system in his NEHFF heuristic. Table 1 summarizes some recent popular heuristic algorithms.

Table 1. Some NEH-based heuristics in terms of priority rule and tie-breaking rule

| Heuristics | Priority rule | Tie-breaking rule |
|---|---|---|
| NEH (1983) | Descending sum of operation times | Usually the first position is selected when ties exist |
| NEHNM (2002) | Descending difference between total processing time and lower bound of job waiting time | Same as NEH |
| MNEH (2004) | Descending sum of artificial processing times | The position with the least idle-time on bottleneck machine is selected |
| NEHKK (2007) | Same as NEH | The position is selected with the least maximum completion time of the sequence between two tie positions |
| NEHKK1 (2007) | Non-increasing sum of weighted processing times $\min(a_i, b_i)$ where $a_i = \sum_{k=1}^{m} \left[\frac{(m-1)(m-2)}{2} + m - k\right] t_{i,k}$, $b_i = \sum_{k=1}^{m} \left[\frac{(m-1)(m-2)}{2} + k - 1\right] t_{i,k}$ | Job x is inserted into the first (or last) position if $a_x \leq b_x$ (or $a_x \geq b_x$) |
| NEH-D (2008) | Descending sum of mean and standard deviation of processing times | The position with more balanced machine utilization is selected |
| NEHKK2 (2009) | Non-increasing sum of weighted processing times $\min(a_i, b_i)$ where $a_i = \sum_{k=1}^{m} t_{i,k} + \sum_{h=1}^{s} \left(\frac{h-\frac{3}{4}}{s-\frac{3}{4}} -$ | Same as NEHKK1 with corresponding $a_x$ and $b_x$ |

| | | |
|---|---|---|
| | $\varepsilon\Big)(t_{s+1-h,i} - t_{t+h,i})$ , $b_i = \sum_{k=1}^{m} t_{i,k} - \sum_{h=1}^{s}\left(\frac{h-\frac{3}{4}}{s-\frac{3}{4}} - \varepsilon\right)(t_{s+1-h,i} - t_{t+h,i})$ and $s = \lfloor m/2 \rfloor$, $t = \lceil m/2 \rceil$ | |
| NEHFF (2014) | Same as NEH | Choose the position with the least front delay and idle-time |

Machine idle-time has been rarely utilized in the literature for PFSP, but it is an important performance measure in manufacturing enterprises, and many companies are using it to drive their operators' behaviour on the shop floor. As shown in Fig.1, apart from machine operations, the empty space is categorized as front delay, idle-time (IT), and back delay (Spachis 1978). Front delay could be occupied by production prior to the current batch, while back delay could be filled in by the subsequent operations, but the idle-time is a real waste, which should be minimized. The idle-time on machine k generated by the $i^{th}$ job can be computed by $IT_{[i],k} = \max\{C_{[i],k-1} - C_{[i-1],k}, 0\}$. In the literature, minimizing machine idle-time was adopted as a strategy to minimize makespan instead of the objective. The most widely accepted objective is to minimize idle-time on the last machine with a makespan criterion. Jobs which do not generate idle-time on last machine are chosen and added to schedule one by one (Sarin and Lefoka 1993), which is similar to MINimum Idle-time (MINT) algorithm which intends to minimize idle-time on the last machine (Gupta 1972). Liu and Reeves (2001) introduced an idle-time based index for composite heuristics for PFSP by calculating the fitness of unscheduled jobs to the last job of partial schedule. Few studies have focused on idle-time minimization directly.

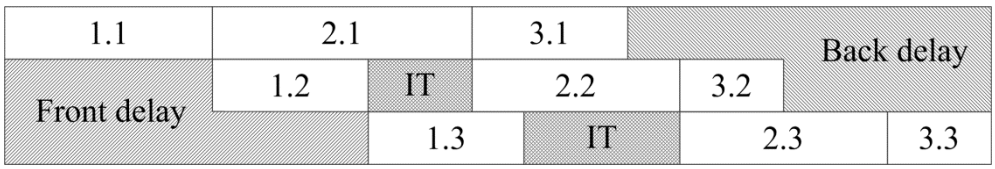


Figure 1. Front delay, idle-time (IT) and back delay of schedule (i.k indicates the processing time of job i on machine k)

A single criterion has been widely used in the heuristics for PFSP, but a single criterion is not good enough for describing complex practical problems. Therefore, researchers have begun to consider applying multiple criteria to solve many real-world scheduling problems (T'kindt and Billaut 2006). Framinan *et al*. (2002) developed two constructive heuristics by using the NEH search strategy for the two-objectives of makespan and flowtime. Braglia and Grassi (2009) developed a heuristic MOGI by integrating the technique for ordered preference by similarity of ideal solution (TOPSIS) with NEH for the objective of makespan and maximum tardiness. The multi-criteria decision making technique TOPSIS is adopted for each step of the NEH heuristic. Chandra et al. (2009) proposed a heuristic procedure for PFSP with objective of earliness and tardiness for a bulk-drugs manufacturer with wide range of due date. In the recent literature review (Yenisey and Yagmahan 2014) the consideration of PFSP with multiple objectives concluded that much attention was paid to multi-objective heuristics for PFSP but idle-time criterion has rarely been taken into account for multi-objectives scheduling. Therefore, herein we consider idle-time minimization together with makespan criterion for PFSP.

The remainder of this paper is organized as follows. In section 2, the example problem studied is described. The newly proposed heuristic is developed in Section 3. In section 4, test cases and computational results are presented, assessing the effectiveness of the proposed algorithm. Final conclusions are presented in section 5.

## 2. Problem definition

In this example ofa permutation flowshop, **n** jobs are to be processed consecutively on **m** machines and the order of jobs on every machine keep the same. Since the

movements of jobs are not in the same pace, buffers exist between machines when jobs wait, and machines may be idle if no job is ready. In order to complete jobs as soon as possible and maximize machine utilization, makespan and machine idle-time are to be minimized. Following convention (T'kindt and Billaut 2006) the problem can be categorized as $F_m|prmu|C_{max},IT$, where $F_m$ represents an m machine flow shop, *prmu* stands for permutation, and $C_{max}$,*IT* represent makespan and idle time. Table 2 shows the notation used in this paper.

Table 2. Notation adopted in this paper

| Parameter | Description |
|---|---|
| n | Total number of jobs |
| m | Total number of machines |
| i | Index for job, $1 \leq i \leq n$ |
| k | Index for machine, $1 \leq k \leq m$ |
| [i] | The $i^{th}$ job of schedule |
| $t_{i,k}$ | Processing time of job i on machine k |
| $IT_{[i],k}$ | Idle-time of machine k generated by the $i^{th}$ job |
| $C_{[i],k}$ | Completion time of the $i^{th}$ job on machine k |

The objective function can be expressed as

$$\text{Min: } F = w_1 * C_{[n],m} + (1 - w_1) \sum_{i=1}^{n} \sum_{j=1}^{m} IT_{[i],j} \quad (1)$$

where $w_1$ is the weight of $C_{[n],m}$. The assumptions used in this paper are described below.

(1) All jobs are available at time zero and start as soon as possible.

(2) Processing time is known and deterministic.

(3) Setup time is included in processing time.

(4) Machines are continuously available but cannot process two or more jobs simultaneously.

(5) Job pre-emption is not permitted.

(6) Buffers' capacity between machines is infinite.

(7) Only permutation schedules are allowed.

## 3. The proposed algorithm

### *3.1 $PR_{LJP}$: New priority rule*

All jobs are sorted according to the descending sum of

$$(\mathrm{AVG_i} + \mathrm{MAD_i} + \mathrm{abs}(\mathrm{SKE_i})^{1/3} + 1/\mathrm{KUR_i}^{1/4}),$$

where $\mathrm{AVG_i}$ represents the average processing time of job i on all machines, $\mathrm{MAD_i}$ is the mean absolute deviation of processing times of job i, $\mathrm{SKE_i}$ and $\mathrm{KUR_i}$ represents skewness and kurtosis of processing times of job i respectively. Mathematically, they are expressed as follows.

$$\mathrm{AVG_i}\,\frac{1}{\mathrm{m}}\sum_{\mathrm{k=1}}^{\mathrm{m}}\mathrm{t_{i,k}}, \tag{2}$$

$$\mathrm{MAD_i} = \frac{1}{\mathrm{m}}\sum_{\mathrm{k=1}}^{\mathrm{m}}\left|\mathrm{t_{i,k}} - \mathrm{AVG_i}\right|, \tag{3}$$

$$\mathrm{SKE_i} = \frac{\frac{1}{m}\sum_{\mathrm{k=1}}^{\mathrm{m}}\left(\mathrm{t_{i,k}}-\mathrm{AVG_i}\right)^3}{\left(\sqrt{\frac{1}{m}\sum_{\mathrm{k=1}}^{\mathrm{m}}\left(\mathrm{t_{i,k}}-\mathrm{AVG_i}\right)^2}\right)^3}, \tag{4}$$

$$\mathrm{KUR_i} = \frac{\frac{1}{m}\sum_{\mathrm{i=1}}^{\mathrm{m}}\left(\mathrm{t_{i,k}}-\mathrm{AVG_i}\right)^4}{\left(\frac{1}{m}\sum_{\mathrm{k=1}}^{\mathrm{m}}\left(\mathrm{t_{i,k}}-\mathrm{AVG_i}\right)^2\right)^2}. \tag{5}$$

The general idea of the new priority rule is the same as NEH, i.e., priority is given to the job with the largest total processing time. But further components are added to more accurately describe the distribution of processing times on every machine, which would have far more effect on the scheduling solution. In the literature, $\mathrm{STD_i}$ is used for this purpose in NEH-D (Dong, Huang, and Chen 2008), which demonstrated improved performance. But the square effect of the difference between sample values

and mean value in computing $STD_i$ could not be eliminated after taking the square root and it can easily enlarge data deviation. To overcome the shortcoming of standard deviation, herein, the mean absolute deviation, $MAD_i$, is considered.

Although the average $AVG_i$ and the $MAD_i$ are considered, they are not sufficient to describe the whole distribution of processing times. The average is used to measure central tendency and the deviation $MAD_i$ depicts the degree of variation from the average. However, as shown in Fig. 2 and Fig. 3, different distributions may share the same average and deviation. But they should be differentiated when scheduling. For this purpose, skewness and kurtosis are proposed to allow this differentiation. A large skewness (positive and negative) indicates high frequency of operation times deviated from the average. Therefore the corresponding job should be given a higher priority. Kurtosis defined in Eq. (5) measures the relative peakedness of a distribution. Low kurtosis means high processing time dispersion. So the job with a large kurtosis of processing times on each machine should be allocated high priority. Therefore, two new hypotheses are proposed here:

1. a job with a large skewness of processing times should be given a higher priority;
2. a job with a low kurtosis should be given a higher priority.

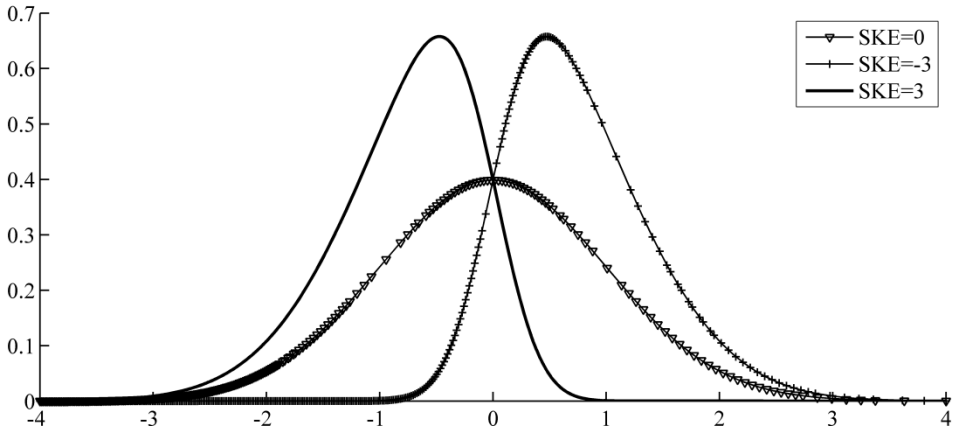


Figure 2. Distributions A, B and C with the same average and deviation but different skewnesses

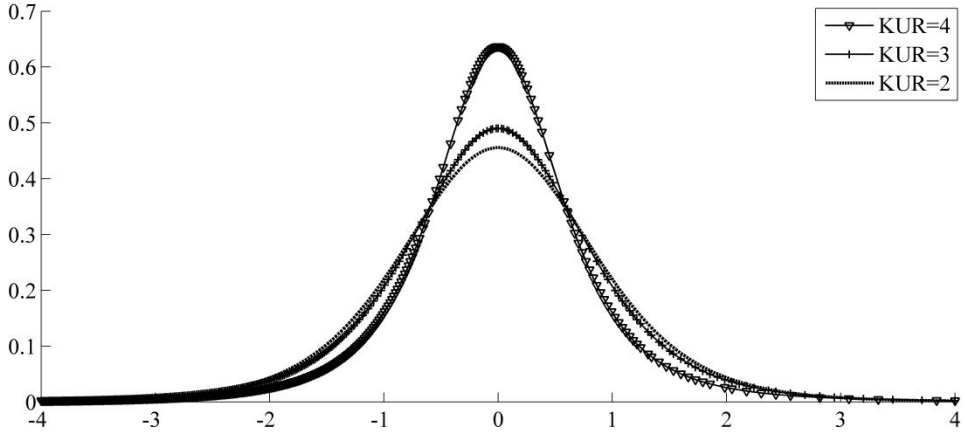



Figure 3. Distributions A, B and C with the same average, deviation and skewness but different kurtoses

Actually, the average is the first moment of a set of data while the mean absolute deviation is used to stand for the second central moment. The skewness and kurtosis represent the 3rd and 4th central moments respectively. To make them dimensional, the 3rd order root of $SKE_i$ and 4th order root of $KUR_i$ are adopted together with $AVG_i$ and $MAD_i$ when allocating priorities.

### *3.2 $TB_{LJP}$: New tie-breaking for job insertion*

In the new tie-breaking rule, named as $TB_{LJP}$, the average and standard deviation of job flow times are used to choose the position for the newly inserted job. Mathematically, the position associated with the largest $P = (\bar{f} - std(f))/C_{max}$ , where $\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$ represents the mean dynamic flow times of all jobs, $f_i = C_{[i],m} - C_{[i-1],1}$ , $C_{[0],1} = 0$, and $std(f) = \sqrt[2]{\frac{1}{n-1}\sum_{i=1}^{n}(f_i - \bar{f})^2}$ represents the standard deviation of flow times.

Usually minimizing flow times leads to makespan minimization and machine idle-time increase, and vice versa. The example in Fig. 4 illustrates this point. In Fig.

4(b), the average dynamic flow time is lower than the one in Fig. 4(a) but it has more idle-time in the schedule. Given the same sum of idle time and makespan, a large average flow time makes the schedule tight with less idle-time in schedule at the cost of makespan. In order to balance the two objectives, large average flow time divided by makespan is pursued. Notably, small flow time variation keeps the schedule smooth. Consequently, large average and low variation of job flow time are adopted as a combined tie-breaking rule while keeping small makespan.

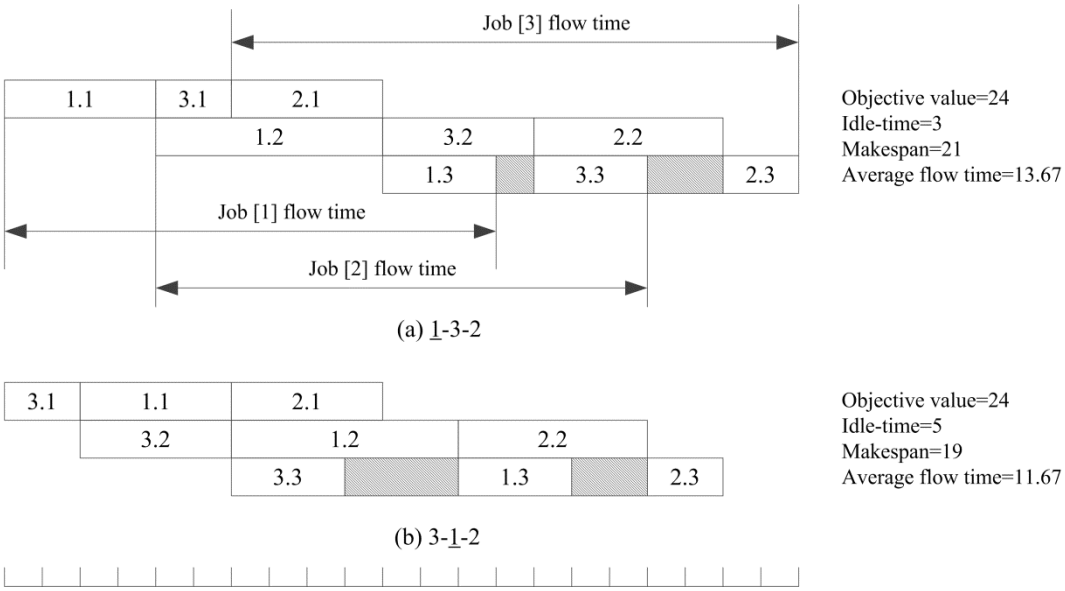



Figure 4. Tie-breaking rule: (a) job 1 added into the 1st position; (b) job 1 added into the 2nd position

For example in Fig. 4, job 1 is going to be inserted into partial sequence 3-2 and two new partial sequences with the same objective value 24 are generated, (a) 1-3-2 and (b) 3-1-2. From Fig. 4 (a) it can be seen that the flow times of job 1, 3 and 2 are 13, 13 and 15. The average is 13.67, maximum completion is 21 and standard deviation is 1.15 while in sequence (b) the flow times are 9, 13 and 13 respectively with average of 11.67, maximum completion time of 19 and deviation of 2.31. Therefore, sequence 1-3-2 is selected with P of (13.67-1.15)/21=0.60, larger than that of sequence 3-1-2 with P of (11.67-2.31)/19=0.49 according to the new tie-breaking method.

### *3.3 NEHLJP: New heuristic based on $PR_{LJP}$ & $TB_{LJP}$*

Based on the above proposed rules, a new heuristic named NEHLJP is proposed as follows.

(1) Sort all jobs according to descending sum of $(AVG_i+MAD_i+abs(SKE_i)^{1/3}+1/KUR_i^{1/4})$;

(2) Take the first two jobs and determine the 2-job partial sequence;

(3) For the rest of jobs, insert it into each possible position and retain the partial sequence associated with the least objective value, as defined in Eq. (1). If ties exist, tie-breaking method is applied;

(4) Repeat step 3 until every job is scheduled.

## 4. Tests and results

To evaluate the performance of the new proposed heuristic, Taillard (Taillard 1993) test bed, VRF benchmark (Vallada, Ruiz, and Framinan 2015) and a randomly generated test bed were used. The test bed presented by Taillard includes 120 instances, 12 different size problems ranging from small size problem, n=20 and m=5 to large size problem n=500 and m=20. Each problem includes 10 instances. It is widely used for PFSP with the makespan criterion, and it has also been applied to criterion of total or mean flow time (Sarin and Lefoka 1993). VRF benchmark is the newest hard test bed including 480 instances ranging from n=10 and m=5 problem to n=800 and m=60 problem. Apart from these two benchmarks, a set of randomly generated instances were also used due to the fluctuated performance of each exiting heuristics on different test beds. There are 450 instances randomly generated with $n \in \{10, 20, 40, 80, 120, 160,$

200, 300, 400} and m ∈ {10, 20, 30, 40, 50}, 10 replications for each combination. The processing times are set uniformly distributed in the range of [1, 99]. So three test beds were used and 1050 instances were tested in total. The Relative Percentage Deviation (RPD) is employed as a performance measure where $\mathrm{RPD_q} = \frac{\mathrm{HS_q} - \mathrm{RS_q}}{\mathrm{RS_q}} \cdot 100\%$, $\mathrm{HS_q}$ represents the value obtained by heuristic of problem instance q and $\mathrm{RS_q}$ is the objective function value of the best approach of the instance. All algorithms are run in Matlab R2013b on a PC with CORE i5 -3210M CPU 2.50 GHz and 4.00GB memory.

To fully depict the performance of the new proposed heuristic NEHLJP, 11 sets of tests were conducted under different weights of makespan with $w_1 = 0, 0.1, 0.2, \ldots, 1$ respectively. Figure 5 shows performance of each heuristic under different weights of objectives in terms of average RPD (ARPD) on Taillard test bed. As the weight of makespan varyies in the objective, the ARPD values of reference heuristics fluctuate. For example, NEH-D is better than the others when only one criterion is applied, but it is worse when bicriteria is given. When idle-time and makespan carry equal weight, NEHLJP, NEHKK1 and NEHKK2 show better performance than NEH-D. This indicates that idle-time and makespan are different although they may be related.

To coordinate the two objectives, it is reasonable to define $w_1$ as 0.5. Because makespan and idle-time are both measures of time which are limited resource for industry. Taking the results of NEH-D heuristic on 20 jobs, 50 jobs and 100 jobs of Taillard test bed as examples, the average of makespan and idle-time are 1868, 3491, 6096 and 1853, 2873, 3944 respectively. Makespan and idle-time have the same magnitude in the objective function. Therefore the weighted objective function is feasible.

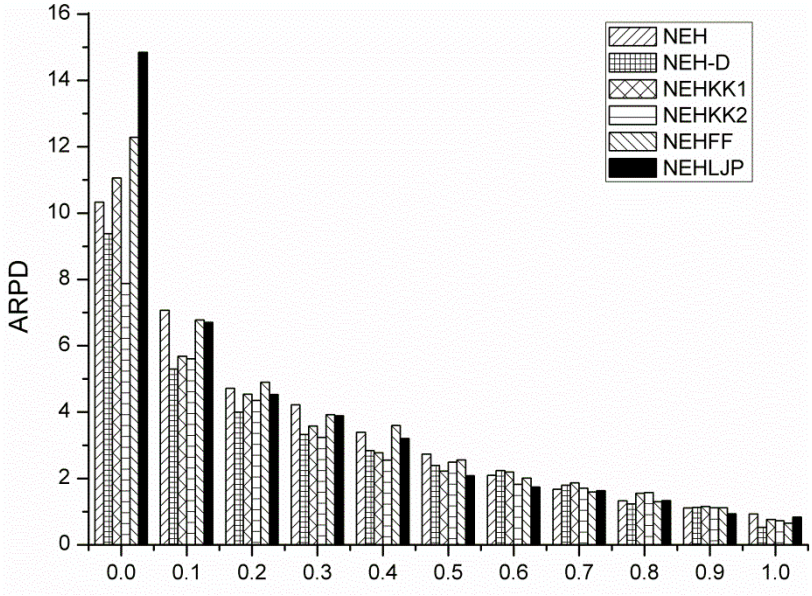



Figure 5. Performance of each heuristic with different $w_1$ on Taillard test bed

To further verify the effectiveness of the proposed NEHLJP heuristic, the new sorting rule and tie-breaking rule were tested in the following sections respectively by comparing to existing heuristics on Taillard benchmark.

### *4.1 Test results of using the new priority rule*

The new priority rule is adopted in NEH heuristic combined with other four existing priority rules. As shown in Table 3, the new priority rule in $PR_{LJP}$ was superior to that of existing heuristics in terms of ARPD, 22.41%, 10.34%, 7.04% and 13.85% better than NEH, $PR_D$, $PR_{KK1}$ and $PR_{KK2}$ respectively.

Table 3. ARPDs of each priority rule adopted in NEH heuristic (%)

| Instance | NEH | $PR_D$ | $PR_{KK1}$ | $PR_{KK2}$ | $PR_{LJP}$ |
|---|---|---|---|---|---|
| 20\|5 | 2.07 | 2.51 | 1.47 | 1.95 | 2.15 |
| 20\|10 | 4.80 | 5.38 | 3.67 | 6.18 | 3.11 |
| 20\|20 | 4.04 | 4.37 | 4.32 | 2.44 | 4.09 |
| 50\|5 | 0.79 | 1.36 | 0.55 | 0.55 | 1.16 |
| 50\|10 | 3.69 | 3.66 | 3.33 | 3.79 | 3.08 |
| 50\|20 | 3.52 | 2.03 | 2.10 | 4.33 | 2.14 |
| 100\|5 | 0.80 | 0.45 | 0.54 | 0.66 | 0.70 |
| 100\|10 | 2.30 | 2.06 | 2.05 | 1.60 | 1.72 |
| 100\|20 | 3.73 | 2.05 | 2.89 | 3.44 | 2.33 |

| | | | | | |
|---|---|---|---|---|---|
| 200\|10 | 1.12 | 1.10 | 0.92 | 0.76 | 0.57 |
| 200\|20 | 3.45 | 1.66 | 2.77 | 2.93 | 2.53 |
| 500\|20 | 2.00 | 1.34 | 2.36 | 0.48 | 1.49 |
| AVG | 2.69 | 2.33 | 2.25 | 2.42 | 2.09 |

### *4.2 Test results of using the tie-breaking rule*

The test results of tie-breaking rule are shown in Table 4. Each tie-breaking rule is adopted in NEH heuristic. The ARPDs of reference heuristics are 0.95, 0.92, 0.90, 0.97, 0.78 and 0.89. The results show that $TB_{FF}$ outperforms the others, followed by $TB_{LJP}$, $TB_{KK1}$,$TB_{KK2}$, $TB_{D}$ and NEH. Significant differences between $TB_{LJP}$ and other four tie-breaking rules can be observed. However, the tie-breaking rule $TB_{LJP}$ achieves better performance with the new proposed priority rule $PR_{LJP}$ than $TB_{FF}$. Due to the superiority of the new priority rule, every tie-breaking rule is applied together with the new priority rule and tested on Taillard benchmark. As depicted in Table 5, by adopting the new priority rule, the new tie-breaking rule $TB_{LJP}$ resulted in the best solution quality with an ARPD value of 0.87, better than $TB_{FF}$.

Table 4. Tie-breaking rule adopted in NEH heuristic (%)

| Instance | NEH | $TB_{D}$ | $TB_{KK1}$ | $TB_{KK2}$ | $TB_{FF}$ | $TB_{LJP}$ |
|---|---|---|---|---|---|---|
| 20\|5 | 0.31 | 0.35 | 0.52 | 0.53 | 0.78 | 0.65 |
| 20\|10 | 0.55 | 0.14 | 0.55 | 0.55 | 0.29 | 0.22 |
| 20\|20 | 0.00 | 0.10 | 0.00 | 0.00 | 0.10 | 0.10 |
| 50\|5 | 0.10 | 0.57 | 0.13 | 0.16 | 0.40 | 0.42 |
| 50\|10 | 2.66 | 3.06 | 3.11 | 3.11 | 1.03 | 2.67 |
| 50\|20 | 0.14 | 0.18 | 0.16 | 0.16 | 0.13 | 0.20 |
| 100\|5 | 0.23 | 0.20 | 0.21 | 0.22 | 0.28 | 0.29 |
| 100\|10 | 1.31 | 0.76 | 0.77 | 0.78 | 1.07 | 0.59 |
| 100\|20 | 1.73 | 0.84 | 1.09 | 1.09 | 1.12 | 0.60 |
| 200\|10 | 1.04 | 0.74 | 0.73 | 0.60 | 0.91 | 0.96 |
| 200\|20 | 1.87 | 2.36 | 1.77 | 1.86 | 2.02 | 2.57 |
| 500\|20 | 1.50 | 1.79 | 1.78 | 2.63 | 1.25 | 1.39 |
| AVG | 0.95 | 0.92 | 0.90 | 0.97 | 0.78 | 0.89 |

Table 5. Each tie-breaking rule implemented with $PR_{LJP}$ (%)

| Instance | NEH | $TB_{D}$ | $TB_{KK1}$ | $TB_{KK2}$ | $TB_{FF}$ | $TB_{LJP}$ |
|---|---|---|---|---|---|---|
| 20\|5 | 0.42 | 0.29 | 0.16 | 0.15 | 0.60 | 0.18 |
| 20\|10 | 0.53 | 0.42 | 1.26 | 1.26 | 1.02 | 1.10 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 20\|20 | 0.07 | 0.00 | 0.00 | 0.00 | 0.05 | 0.22 |
| 50\|5 | 0.17 | 0.28 | 0.13 | 0.08 | 0.07 | 0.16 |
| 50\|10 | 2.21 | 1.32 | 1.75 | 1.75 | 1.54 | 0.68 |
| 50\|20 | 0.70 | 1.54 | 0.59 | 0.59 | 0.52 | 0.82 |
| 100\|5 | 0.29 | 0.31 | 0.46 | 0.53 | 0.11 | 0.29 |
| 100\|10 | 0.97 | 1.12 | 0.69 | 0.50 | 0.91 | 1.17 |
| 100\|20 | 1.94 | 3.27 | 1.04 | 1.04 | 2.53 | 2.15 |
| 200\|10 | 0.79 | 0.99 | 0.99 | 0.90 | 0.95 | 0.88 |
| 200\|20 | 1.74 | 1.90 | 2.79 | 3.26 | 1.34 | 1.88 |
| 500\|20 | 1.69 | 1.75 | 1.34 | 1.50 | 1.40 | 1.70 |
| AVG | 0.89 | 1.04 | 0.90 | 0.92 | 0.88 | 0.87 |

### *4.3 Test results of the new heuristic algorithm with the new priority and tie-breaking rule*

Table 6 shows the test results of each heuristic with respect to bicriteria of makespan and idle-time on Taillard benchmark. The ARPD of NEHLJP is 2.09, the best among all reference heuristics, followed by NEHKK1, NEH-D, NEHKK2, NEHFF and NEH. The ARPD results of all reference heuristics on VRF benchmark are 3.18, 2.46, 3.12, 2.56, 3.06 and 2.41 respectively. NEHLJP demonstrates that the best solution quality is that with the lowest ARPD value. It can be seen from Table 6 and 7, that the performance of existing heuristics fluctuates largely, indicating that the two objectives makespan and machine idle-time conflict sometimes. It can be concluded that the new heuristic NEHLJP performs the best on both test beds.

Table 6. ARPD of each heuristic with bicriteria using Taillard benchmark (%)

| Instance | NEH | NEH-D | NEHKK1 | NEHKK2 | NEHFF | NEHLJP |
|---|---|---|---|---|---|---|
| 20\|5 | 2.12 | 2.44 | 1.67 | 2.22 | 2.61 | 1.96 |
| 20\|10 | 5.10 | 5.31 | 4.04 | 6.16 | 4.82 | 4.04 |
| 20\|20 | 3.96 | 4.28 | 4.23 | 2.35 | 4.06 | 4.16 |
| 50\|5 | 0.66 | 1.51 | 0.43 | 0.55 | 0.96 | 1.02 |
| 50\|10 | 3.80 | 3.30 | 3.37 | 3.81 | 2.15 | 1.66 |
| 50\|20 | 3.61 | 1.87 | 2.38 | 4.25 | 3.59 | 2.36 |
| 100\|5 | 0.77 | 0.41 | 0.52 | 0.72 | 0.83 | 0.68 |
| 100\|10 | 2.02 | 1.33 | 0.90 | 0.47 | 1.78 | 1.65 |
| 100\|20 | 4.65 | 3.32 | 3.48 | 4.22 | 4.04 | 3.46 |
| 200\|10 | 1.15 | 1.06 | 0.81 | 0.95 | 1.02 | 0.68 |
| 200\|20 | 2.48 | 2.19 | 1.71 | 1.77 | 2.64 | 1.72 |
| 500\|20 | 2.16 | 1.33 | 2.78 | 2.09 | 1.90 | 1.64 |

| AVG | 2.71 | 2.36 | 2.19 | 2.46 | 2.53 | 2.09 |
|---|---|---|---|---|---|---|

Table 7. ARPD of each heuristic with bicriteria on VRF benchmark (%)

| Problem | | NEH | | NEH-D | | NEHKK1 | | NEHKK2 | | NEHFF | | NEHLJP | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | L | S | L | S | L | S | L | S | L | S | L | S | L |
| 10×5 | 100×20 | 0.76 | 3.61 | 2.05 | 2.32 | 0.67 | 4.32 | 0.07 | 2.43 | 1.09 | 4.47 | 1.37 | 4.08 |
| 10×10 | 100×40 | 2.10 | 2.38 | 1.09 | 0.60 | 1.80 | 2.34 | 2.72 | 1.69 | 2.10 | 2.29 | 1.55 | 2.81 |
| 10×15 | 100×60 | 4.28 | 3.23 | 1.54 | 1.58 | 3.09 | 3.42 | 3.01 | 2.66 | 4.28 | 3.23 | 2.13 | 2.24 |
| 10×20 | 200×20 | 3.90 | 3.82 | 2.41 | 2.88 | 4.17 | 2.91 | 1.97 | 3.03 | 3.90 | 2.88 | 2.46 | 2.23 |
| 20×5 | 200×40 | 3.07 | 3.06 | 5.30 | 2.43 | 3.33 | 3.31 | 4.23 | 2.49 | 3.63 | 3.15 | 2.21 | 2.68 |
| 20×10 | 200×60 | 4.88 | 3.31 | 5.11 | 2.80 | 5.17 | 2.69 | 1.74 | 2.47 | 5.02 | 3.29 | 4.79 | 2.14 |
| 20×15 | 300×20 | 5.94 | 3.20 | 2.30 | 1.85 | 6.64 | 2.14 | 4.92 | 3.00 | 5.88 | 2.67 | 4.66 | 1.61 |
| 20×20 | 300×40 | 3.54 | 3.21 | 3.49 | 2.45 | 2.97 | 3.13 | 2.77 | 2.21 | 3.54 | 2.24 | 3.34 | 1.98 |
| 30×5 | 300×60 | 2.65 | 1.83 | 1.51 | 2.23 | 1.19 | 1.63 | 2.38 | 1.24 | 2.56 | 2.11 | 2.82 | 1.05 |
| 30×10 | 400×20 | 5.59 | 3.10 | 4.24 | 1.68 | 6.48 | 3.04 | 4.61 | 2.20 | 5.37 | 2.53 | 2.87 | 1.93 |
| 30×15 | 400×40 | 4.27 | 2.45 | 4.75 | 0.65 | 3.41 | 1.79 | 7.21 | 1.99 | 4.17 | 2.15 | 5.32 | 0.84 |
| 30×20 | 400×60 | 4.06 | 2.12 | 3.17 | 1.55 | 4.22 | 2.57 | 2.36 | 1.67 | 4.06 | 2.24 | 2.74 | 2.61 |
| 40×5 | 500×20 | 2.20 | 3.30 | 2.21 | 1.38 | 2.21 | 3.17 | 2.24 | 1.35 | 2.42 | 2.04 | 2.29 | 2.76 |
| 40×10 | 500×40 | 4.67 | 2.82 | 2.56 | 1.98 | 5.12 | 3.10 | 5.90 | 2.69 | 6.35 | 1.77 | 2.38 | 2.60 |
| 40×15 | 500×60 | 4.84 | 2.40 | 5.67 | 1.68 | 4.61 | 1.70 | 3.82 | 1.89 | 4.14 | 1.79 | 3.41 | 2.02 |
| 40×20 | 600×20 | 3.80 | 2.23 | 5.46 | 1.14 | 6.45 | 2.20 | 3.11 | 0.34 | 4.46 | 2.13 | 3.28 | 2.03 |
| 50×5 | 600×40 | 1.28 | 2.52 | 0.90 | 1.29 | 0.90 | 1.59 | 1.82 | 0.75 | 0.51 | 1.23 | 1.29 | 0.92 |
| 50×10 | 600×60 | 3.65 | 2.41 | 1.58 | 1.60 | 3.85 | 2.45 | 4.08 | 1.10 | 2.57 | 1.44 | 2.15 | 1.96 |
| 50×15 | 700×20 | 3.02 | 1.57 | 4.15 | 1.27 | 4.55 | 1.86 | 3.76 | 1.12 | 3.03 | 1.87 | 5.00 | 2.00 |
| 50×20 | 700×40 | 4.60 | 2.44 | 5.96 | 1.81 | 5.01 | 1.95 | 5.16 | 1.57 | 5.20 | 3.00 | 2.41 | 1.71 |
| 60×5 | 700×60 | 1.91 | 2.02 | 1.97 | 1.21 | 1.53 | 2.20 | 1.65 | 1.11 | 3.26 | 2.03 | 1.77 | 1.95 |
| 60×10 | 800×20 | 3.28 | 1.80 | 2.86 | 1.20 | 2.31 | 1.96 | 2.30 | 0.51 | 2.64 | 1.74 | 2.44 | 1.23 |
| 60×15 | 800×40 | 3.83 | 3.11 | 1.97 | 1.53 | 4.19 | 2.36 | 4.11 | 1.64 | 4.69 | 2.62 | 3.13 | 1.52 |
| 60×20 | 800×60 | 5.62 | 2.93 | 5.40 | 1.17 | 6.22 | 1.89 | 3.88 | 1.85 | 5.20 | 1.88 | 2.73 | 0.44 |
| AVG | | 3.18 | | 2.46 | | 3.12 | | 2.56 | | 3.06 | | 2.41 | |

In order to further confirm the superiority of NEHLJP, the test on randomly generated instances is conducted and the results are shown in Table 8. The ARPD value of NEHLJP was 2.02, ascendingly followed by NEH-D, NEHKK2, NEHFF, NEHKK1 and NEH.

Table 8. ARPD of each heuristic with bicriteria on randomly generated benchmark (%)

| Instance | NEH | NEH-D | NEHKK1 | NEHKK2 | NEHFF | NEHLJP |
|---|---|---|---|---|---|---|
| 10\|10 | 3.27 | 1.65 | 4.37 | 5.05 | 3.27 | 1.88 |
| 10\|20 | 2.67 | 3.53 | 2.67 | 2.60 | 2.67 | 3.07 |
| 10\|30 | 1.78 | 1.23 | 3.41 | 3.44 | 1.78 | 0.82 |
| 10\|40 | 3.44 | 1.29 | 2.19 | 3.05 | 3.44 | 1.80 |
| 10\|50 | 1.46 | 3.21 | 1.46 | 1.64 | 1.46 | 2.20 |
| 20\|10 | 3.55 | 3.17 | 3.28 | 4.52 | 3.38 | 2.02 |
| 20\|20 | 5.10 | 5.01 | 5.32 | 2.60 | 5.10 | 2.79 |
| 20\|30 | 3.78 | 3.90 | 4.37 | 5.07 | 3.78 | 3.10 |
| 20\|40 | 3.88 | 2.60 | 3.67 | 2.86 | 3.88 | 2.20 |
| 20\|50 | 2.22 | 4.50 | 2.46 | 3.61 | 2.35 | 4.21 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 40\|10 | 3.07 | 3.27 | 1.69 | 3.75 | 2.54 | 2.43 |
| 40\|20 | 3.58 | 2.90 | 1.89 | 4.31 | 3.29 | 3.58 |
| 40\|30 | 4.28 | 3.09 | 4.86 | 3.12 | 4.28 | 2.94 |
| 40\|40 | 3.09 | 2.03 | 3.77 | 3.31 | 3.33 | 1.78 |
| 40\|50 | 2.57 | 2.98 | 2.34 | 2.79 | 2.58 | 0.96 |
| 80\|10 | 1.52 | 1.61 | 1.99 | 2.12 | 1.23 | 2.34 |
| 80\|20 | 5.73 | 2.85 | 3.26 | 1.79 | 5.04 | 2.53 |
| 80\|30 | 4.85 | 3.90 | 4.08 | 4.51 | 4.84 | 1.84 |
| 80\|40 | 3.51 | 3.24 | 3.14 | 3.54 | 3.26 | 1.57 |
| 80\|50 | 1.79 | 1.80 | 2.85 | 1.88 | 1.79 | 2.20 |
| 120\|10 | 1.18 | 1.52 | 1.16 | 1.41 | 1.15 | 1.42 |
| 120\|20 | 4.82 | 2.56 | 3.49 | 2.62 | 4.66 | 3.10 |
| 120\|30 | 2.23 | 2.42 | 2.86 | 2.73 | 2.33 | 3.75 |
| 120\|40 | 2.37 | 2.46 | 2.47 | 3.19 | 2.51 | 1.75 |
| 120\|50 | 1.86 | 2.96 | 3.03 | 2.23 | 1.64 | 2.13 |
| 160\|10 | 1.48 | 1.19 | 1.46 | 0.63 | 0.97 | 1.09 |
| 160\|20 | 3.37 | 3.33 | 2.72 | 3.37 | 2.62 | 0.77 |
| 160\|30 | 3.23 | 2.14 | 3.15 | 3.15 | 2.46 | 2.05 |
| 160\|40 | 3.46 | 2.35 | 3.02 | 2.41 | 2.15 | 2.08 |
| 160\|50 | 2.88 | 1.15 | 3.10 | 2.07 | 2.74 | 2.03 |
| 200\|10 | 0.91 | 0.86 | 0.92 | 0.70 | 0.50 | 0.91 |
| 200\|20 | 2.67 | 2.11 | 2.37 | 1.79 | 2.84 | 2.84 |
| 200\|30 | 2.72 | 2.16 | 3.24 | 1.41 | 2.20 | 2.30 |
| 200\|40 | 2.18 | 1.65 | 2.99 | 2.90 | 2.06 | 1.14 |
| 200\|50 | 2.19 | 1.52 | 1.79 | 1.81 | 1.72 | 1.16 |
| 300\|10 | 0.59 | 0.21 | 0.62 | 0.45 | 0.38 | 0.72 |
| 300\|20 | 2.82 | 0.94 | 2.15 | 2.44 | 1.92 | 1.08 |
| 300\|30 | 2.43 | 3.13 | 1.27 | 2.79 | 2.88 | 3.44 |
| 300\|40 | 1.88 | 1.44 | 2.06 | 1.83 | 2.58 | 1.84 |
| 300\|50 | 2.21 | 1.80 | 2.12 | 1.99 | 2.97 | 1.65 |
| 400\|10 | 0.36 | 0.23 | 0.37 | 0.22 | 0.50 | 0.18 |
| 400\|20 | 1.96 | 1.09 | 1.61 | 1.74 | 1.75 | 1.57 |
| 400\|30 | 3.77 | 2.61 | 3.50 | 1.35 | 3.90 | 2.06 |
| 400\|40 | 2.37 | 2.03 | 1.52 | 1.14 | 2.08 | 2.13 |
| 400\|50 | 2.37 | 2.73 | 2.22 | 1.41 | 1.67 | 1.40 |
| AVG | 2.74 | 2.32 | 2.63 | 2.52 | 2.59 | 2.02 |

In order to check if the differences of ARPD between heuristics are statistically significant, a multifactor analysis of variance (ANOVA) was carried out. All three hypotheses (normality, homocedasticity and independence of the residuals) were carefully checked but were not found to be suitable. The normality is rejected since all variables focus on the right side of zero according to the adopted performance measure. Therefore a non-parametric Mann-Whitney test was conducted.

When the confidence level was set as 0.05, no significant differences were observed on Taillard benchmark among all reference heuristics. The reason is due to the small size of benchmark (Kalczynski and Kamburowski 2008), (Fernandez-Viagas and Framinan 2014). While on VRF benchmark, NEHLJP showed no significant difference from other heuristics except for NEH. The reason is both benchmarks are developed for makespan criterion instead of bicriteria of makespan and idle-time. No significant performance differences among all heuristics can be observed. While on the randomly generated test bed, it can be concluded that NEHLJP is significant better than existing heuristics. The non-parametric test results can be seen in Table 9.

Table 9. Non-parametric test of Mann-Whitney on the new random test bed

| Algorithm | P |
|---|---|
| NEH – NEH-D | 0.003 |
| NEH – NEHKK1 | 0.315 |
| NEH – NEHKK2 | 0.034 |
| NEH – NEHFF | 0.319 |
| NEH – NEHLJP | 0.000 |
| NEH-D – NEHKK1 | 0.064 |
| NEH-D – NEHKK2 | 0.497 |
| NEH-D – NEHFF | 0.059 |
| NEH-D – NEHLJP | 0.022 |
| NEHKK1 – NEHKK2 | 0.277 |
| NEHKK1 – NEHFF | 0.964 |
| NEHKK1 – NEHLJP | 0.000 |
| NEHKK2 – NEHFF | 0.245 |
| NEHKK2 – NEHLJP | 0.005 |
| NEHFF – NEHLJP | 0.000 |

With respect to computation times of each heuristic, extra computation time is needed for calculating two objectives comparing to with single criterion of makespan. For example, it will take 0.3 secs to complete the computation for a 50 jobs 20 machines problem, and 240 secs for 200 jobs and 20 machines problem for NEHLJP. The computational time is relatively small, within the manageable scale even for a large

size problem, given the computer's average speed. In practice, the schedule quality is the key factor to be considered for industry instead of computation time.

## 5. Conclusions

In this paper, a new heuristic named NEHLJP is proposed by incorporating one new priority rule and one new tie-breaking rule, with bicriteria of both makespan and idle-time. In order to validate the NEHLJP performance, the Taillard test bed, Vallada test bed and a randomly generated test bed are used. The test results show that NEHLJP heuristic can provide a high quality solution on all test beds, outperforming all existing heuristics.

The main contribution of this paper lies in the new proposed priority rule $PR_{LJP}$ taking account of skewness and kurtosis representing the $3^{rd}$ and the $4^{th}$ moment of a distribution. By this new rule, processing time distribution can be differentiated so as to be sequenced. Additionally, a new tie-breaking mechanism $TB_{LJP}$ for minimizing makespan and idle-time simultaneously is introduced. The effectiveness of both $PR_{LJP}$ and $TB_{LJP}$ are validated through statistical tests.

## Reference


Aghezzaf, E.-H., and H. Van Landeghem. 2002. "An Integrated Model for Inventory and Production Planning in a Two-Stage Hybrid Production System." *International Journal of Production Research* 40 (17): 4323–4339.

Allahverdi, Ali. 2004. "A New Heuristic for M-Machine Flowshop Scheduling Problem with Bicriteria of Makespan and Maximum Tardiness." *Computers & Operations Research* 31 (2): 157–180.

Boukef, H., M. Benrejeb, and P. Borne. 2007. "A Proposed Genetic Algorithm Coding for Flow-Shop Scheduling Problems." *International Journal of Computers, Communications & Control* 2 (3): 229–240.

Campbell, H. G., R. A. Dudek, and M. L. Smith. 1970. "A Heuristic Algorithm for the N Job, M Machine Sequencing Problem." *Management Science* 16 (10): B630–637.

Chandra, P., P. Mehta, and D. Tirupati. 2009. “Permutation Flow Shop Scheduling with Earliness and Tardiness Penalties.” *International Journal of Production Research* 47 (20): 5591–5610.

Dannenbring, D. G. 1977. “An Evaluation of Flow Shop Sequencing Heuristics.” *Management Science* 23 (11): 1174–1182.

Dong, X., H. Huang, and P. Chen. 2008. “An Improved NEH-Based Heuristic for the Permutation Flowshop Problem.” *Computers & Operations Research* 35 (12): 3962–3968.

Fernandez-Viagas, V., and J. M. Framinan. 2014. “On Insertion Tie-Breaking Rules in Heuristics for the Permutation Flowshop Scheduling Problem.” *Computers & Operations Research* 45: 60–67.

Framinan, J. M., J. N. D. Gupta, and R. Leisten. 2004. “A Review and Classification of Heuristics for Permutation Flow-Shop Scheduling with Makespan Objective.” *Journal of the Operational Research Society* 55 (12): 1243–1255.

Garey, M. R., D. S. Johnson, and R. Sethi. 1976. “The Complexity of Flowshop and Jobshop Scheduling.” *Mathematics of Operations Research* 1 (2): 117–129.

Garey, Michael Randolph, and David Stifler Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and Company.

Gupta, J. N. 1976. “A Heuristic Algorithm for the Flowshop Scheduling Problem.” *Revue Francaise d’Automatique, Informatique, Recherche Operationnelle* 10: 63–73.

Gupta, J. N. D. 1972. “Heuristic Algorithms for Multistage Flowshop Sdcheduling Problem.” *AIIE Transactions* 4 (1): 11–18.

Gupta, J. N. D. 1971. “A Functional Heuristic Algorithm for the Flowshop Scheduling Problem.” *Operational Research Quarterly* 22 (1): 39–47.

Gupta, J. N. D., and E. F. Stafford. 2006. “Flowshop Scheduling Research after Five Decades.” *European Journal of Operational Research* 169 (3): 699–711.

Johnson, S. M. 1954. “Optimal Two and Three-Stage Production Schedules with Setup Times.” *Naval Research Logistics Quarterly* 1: 61–68.

Kalczynski, P. J, and J Kamburowski. 2008. “An Improved NEH Heuristic to Minimize Makespan in Permutation Flow Shops.” *Computers & Operations Research* 35 (9): 3001–3008.

Kalczynski, P.J., and J. Kamburowski. 2009. “An Empirical Analysis of the Optimality Rate of Flow Shop Heuristics.” *European Journal of Operational Research* 198 (1): 93–101.

Liu, C. Y., and S. C. Chang. 2000. “Scheduling Flexible Flow Shops with Sequence-Dependent Setup Effects.” *IEEE Transactions on Robotics and Automation* 16 (4): 408–419.

Liu, H., L. Gao, and Q. Pan. 2011. “A Hybrid Particle Swarm Optimization with Estimation of Distribution Algorithm for Solving Permutation Flowshop Scheduling Problem.” *Expert Systems with Applications* 38 (4): 4348–4360.

Liu, J., and C. R. Reeves. 2001. “Constructive and Composite Heuristic Solutions to the P//∑Ci Scheduling Problem.” *European Journal of Operational Research* 132 (2): 439–452.

Liu, W., Y. Jin, and M. Price. 2014. “A New Heuristic to Minimize System Idle Time for Flowshop Scheduling.” In *Poster Presented at the 3rd Annual EPSRC Manufacturing the Future Conference*. Glassogw, September 23-24.

Liu, Y. C., K. T. Fang, and B. Lin. 2013. “A Branch-and-Bound Algorithm for Makespan Minimization in Differentiation Flow Shops.” *Engineering Optimization* 45 (12): 1397–1408.

Low, C., J. Y. Yeh, and K. I. Huang. 2004. “A Robust Simulated Annealing Heuristic for Flow Shop Scheduling Problems.” *The International Journal of Advanced Manufacturing Technology* 23: 762–767.

Msakni, M. K., W. Khallouli, M. Al-Salem, and T. Ladhari. 2015. “Minimizing the Total Completion Time in a Two-Machine Flowshop Problem with Time Delays.” *Engineering Optimization*: 1–18. doi:10.1080/0305215X.2015.1099639.

Nagano, M. S., and J. V. Moccellin. 2002. “A High Quality Solution Constructive Heuristic for Flow Shop Sequencing.” *Journal of the Operational Research Society* 53 (12): 1374–1379.

Nawaz, M., E. E. Enscore Jr., and I. Ham. 1983. “A Heuristic Algorithm for the M-Machine, N-Job Flow-Shop Sequencing Problem.” *Omega* 11 (1): 91–95.

Palmer, D. S. 1965. “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): 101–107.

Pan, Q. K., and R. Ruiz. 2013. “A Comprehensive Review and Evaluation of Permutation Flowshop Heuristics to Minimize Flowtime.” *Computers & Operations Research* 40 (1): 117–128.

Ruiz, R., and C. Maroto. 2005. “A Comprehensive Review and Evaluation of Permutation Flowshop Heuristics.” *European Journal of Operational Research* 165 (2): 479–494.

Sarin, S., and M. Lefoka. 1993. “Scheduling Heuristic for the N-Job M-Machine Flow Shop.” *Omega* 21 (2): 229–234.

Spachis, A. S. 1978. “Job-Shop Scheduling with Approximate Methods.” PhD diss., Imperial College London (University of London).

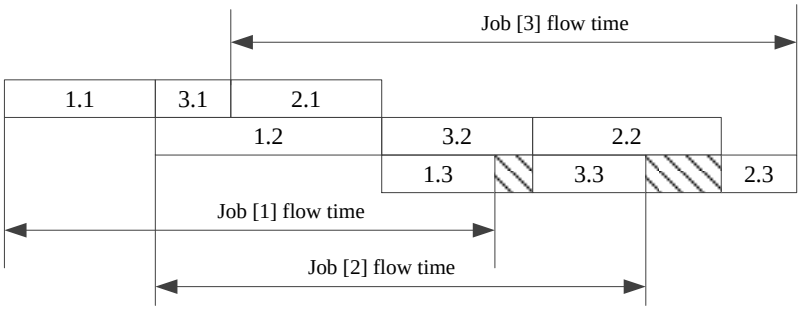
T’kindt, V., and J. C. Billaut. 2006. *Multicriteria Scheduling Problems: Theory, Models and Alogrithms*. Berlin: Springer.

Taillard, E. 1993. “Benchmarks for Basic Scheduling Problems.” *European Journal of Operational Research* 64: 278–285.

Vallada, E., R. Ruiz, and J. M. Framinan. 2015. “New Hard Benchmark for Flowshop Scheduling Problems Minimising Makespan.” *European Journal of Operational Research* 240 (3): 666–677.

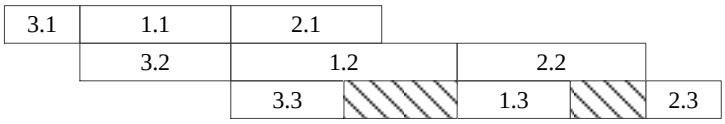
Xu, J., and X. Zhou. 2009. “A Class of Multi-Objective Expected Value Decision-Making Model with Birandom Coefficients and Its Application to Flow Shop Scheduling Problem.” *Information Sciences* 179 (17): 2997–3017.

Yenisey, M. M., and B. Yagmahan. 2014. “Multi-Objective Permutation Flow Shop Scheduling Problem: Literature Review, Classification and Current Trends.” *Omega* 45: 119–135.




Objective value=24
Idle-time=3
Makespan=21
Average flow time=13.67

(a) 1-3-2




Objective value=24
Idle-time=5
Makespan=19
Average flow time=11.67

(b) 3-1-2