www.toc.ui.ac.ir

**Transactions on Combinatorics**
ISSN (print): 2251-8657, ISSN (on-line): 2251-8665
Vol. 11 No. 1 (2022), pp. 15-27.


www.ui.ac.ir

# AN EFFECTIVE NEW HEURISTIC ALGORITHM FOR SOLVING PERMUTATION FLOW SHOP SCHEDULING PROBLEM

SHAHRIAR FARAHMAND RAD

Abstract. The deterministic permutation flow shop scheduling problem with makespan criterion is not solvable in polynomial time. Therefore, researchers have thought about heuristic algorithms. There are many heuristic algorithms for solving it that is a very important combinatorial optimization problem. In this paper, a new algorithm is proposed for solving the mentioned problem. The presented algorithm chooses the weighted path that starts from the up-left corner and reaches the down-right in the matrix of jobs processing times and calculates the biggest sum of the times in the footprints of this path. The row with the biggest sum permutes among all the rows of the matrix for locating the minimum of makespan. This method was run on Taillards standard benchmark and the solutions were compared with the optimum or the best ones as well as 14 famous heuristics. The validity and effectiveness of the algorithm are shown with tables and statistical evaluation.

## 1. Introduction

In recent years, the number of efficient and good heuristic algorithms for solving $n$ jobs, $m$ machines deterministic permutation flow shop scheduling problems (PFSP) with the makespan criterion has been reduced. About 70 years have passed from the beginning of studying PFSP. Unfortunately, it has not been solved exactly. In spite of all these issues, it was proved by Garey et al. that PFSP for $m$ greater than three is NP-complete in the strong sense [11]. Therefore, that was motivated into solving the problem with heuristic methods.

Communicated Mohammad Reza Darafsheh.

MSC(2010): Primary: 90C27; Secondary: 90C59, 90B35.

Keywords: Heuristic, flowshop, scheduling, makespan, total completion time.

Received: 10 December 2020, Accepted: 30 January 2021.

Manuscript Type: Research Paper.

http://dx.doi.org/10.22108/toc.2021.126406.1795

.

In 1954, an exact algorithm was presented by Johnson for solving PFSP with $m = 2$ [18]. However, sometimes problems with three or more machines have converted to some with two machines.

The most famous heuristic algorithms for solving PFSP have been proposed by Page [26], Palmer [27], Campbell et al. [5], Gupta [13], Bonney and Gundry [3], Dannenbring [6], King and Spachis [20], and Stinson and Smith [34].

In 1983, a heuristic algorithm was presented by Nawaz, Enscore, and Ham (NEH) for solving PFSP, which is the lord of heuristic algorithms [24]. In 2005, Kalszynski and Kamburowski as well as Ruiz and Maroto showed that the quality and running time of NEH were better than those of all the others [19, 30]. Afterward, there have been many works about solving PFSP for example, Hundal and Rajgopal [16], Ho and Chang [15], Sarin and Lefoka [31], Koulamas [21], and Suliman [35]. After these years, studying and researching about the heuristic algorithms for solving PFSP with makespan criterion have demonstrated that proposed algorithms are not better than NEH.

Also, the summary of all these subjects and papers for solving PFSP with heuristic algorithms can be found in Framinan et al. [10], Ruiz and Maroto [30], Hejazi and Saghafian [14], T'kindt and Billaut [38], and Pott and Strusevich [28].

Rad et al. [29] proposed five new algorithms. In 2012, Ancau proposed a constructive algorithm [1]. Singhal et al. [33] proposed a method that was obtained by modifying the NEH algorithm.

Malik and Dhingra [23] presented a comparative analysis of heuristics for minimizing $C_{\max}$ in PFSP. Xu et al. [40] improved an algorithm based on a dynamic neighborhood for the PFSP. Valada et al. [39] discovered a new benchmark for PFSP with minimizing makespan criterion. Liu et al. [22] proposed a new NEH based heuristic. Fernandez et al. [9] presented a new vision of approximate methods for the problem. Brum and Ritt [4] studied the application of automatic algorithm configuration. Nurdiansyah et al. [25] proposed an improved differential evaluation algorithm. Sauvey and Sauer [32] discovered two new heuristic improvements to the original NEH.

There are a large number of algorithms that were proposed for solving PSFP with makespan criterion, but the results of these algorithms are studied on problems with different measures and only on little amounts of $m$ and $n$. For comparing the validity and effectiveness of this category of algorithms, it is convenient that the number of machines is equal and then the number of jobs is even in them as far as possible. Furthermore, the jobs processing times are alike for all algorithms and these specified uniformed of $m$ and $n$. All the above algorithms were run on Taillard's benchmark. Collecting both traditional and contemporary conclusions of numerousness of other algorithms is impossible because they were not run on the same standard benchmark.

This paper includes five sections. In the remainder, four sections are presented. Section two provides assumptions that are in an arbitrary PFSP with minimizing makespan denoted by $C_{\max}$. In section three, we study the proposed algorithm step by step. Next, we explain the running of this algorithm on Taillard's benchmark. Finally, the results are given plus the comparison of 15 heuristics, including ours and 14 others.

## 2. **Problem Description**

A PFSP deals with finding the best queue of $n$ jobs that are to be processed on m machines. Processing of jobs on machines has the same order. Let $p_{ij}$ denotes the processing time of job $j$ at machine $i$. We suppose that each $p_{ij}$ is strictly positive and deterministic. After studying partial sequences of jobs, our objective is to determine a sequence that minimizes the maximum completion time. This objective is called makespan or $C_{\max}$.

In PFSP, we suppose that:

(1) All jobs and machines are ready for starting and continuing the process.
(2) All jobs are separately processed.
(3) There is a one to one corresponding between jobs processing and machines at any time.
(4) Each job continuously is processed at a machine.
(5) Processing times are included the set-up times.
(6) In processing, between any two machines buffer storage is infinite.

In general, we have many possible job orders. Since every order can change from one machine to another, there are $(n!)^m$ schedules. As for all machines the order is the same and fixed, there are $n!$ possible schedules. This problem is denoted as $Fm|Prmu|C_{\max}$ by Graham et al. [12].

## 3. **Heuristic Algorithm**

Let $M = [p_{ij}],\ n \geq i \geq 1,\ m \geq j \geq 1$, be the matrix of jobs processing times in PFSP. From the scheduling literature, we have the following directed graph.

$$\begin{array}{ccccccccccc}
p_{11} & \to & p_{12} & \to & p_{13} & \to & \cdots & \to & p_{1,m-1} & \to & p_{1m} \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
p_{21} & \to & p_{22} & \to & p_{23} & \to & \cdots & \to & p_{2,m-1} & \to & p_{2m} \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
p_{31} & \to & p_{32} & \to & p_{33} & \to & \cdots & \to & p_{3,m-1} & \to & p_{3m} \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\vdots & & \vdots & & \vdots & & & & \vdots & & \vdots \\
\downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
p_{n1} & \to & p_{n2} & \to & p_{n3} & \to & \cdots & \to & p_{n,m-1} & \to & p_{nm}
\end{array}$$

http://dx.doi.org/10.22108/toc.2021.126406.1795

As shown in Bellman et. al. [2], the PFSP from another perspective is a permutation of rows in $M$ so that the length of the longest path is reduced. Additionally, in the base of problem conditions in a given solution one path that starts from $p_{11}$ must end in $p_{nm}$ only with passes to right and to below. This path crosses from $m + n - 1$ $p_{ij}s$ which are called footprints. The sum of all footprints in a path is called weight of this path.

Based on the above ideas, each row in $M$ is chosen and the possible nodes at it are evaluated while we try to minimize $C_{\max}$.

We present a new algorithm that is divided into six steps:

(1) All the paths that start from the up-left corner and reach the down-right one are held.
(2) The weighted path in step one is chosen.
(3) In the previous step, the weighted path's footprints are followed in each row and sum all the times in these footprints is obtained in it. Then, the biggest one in these sums is determined.
(4) The row to which the biggest sum in the previous step was related, is specified. This row is permuted among all the rows of M and, in each case, the minimum of makespans is calculated.
(5) The row that is obtained in step 4 is determined and its place that causes the least of minimums is obtained. The row is stabilized in this place and then step 1 is repeated.
(6) The above process is repeated and the obtained rows are stabilized. Finally, the minimum of makespans of the obtained matrix is determined.

This algorithm is denoted in abbreviation by FRS.

## 4. **Computationl Evaluation**

The heuristic was implemented in Visual Basic and all the problems were carried on a Pentium IVPC/AT computer running at 3.2 GHz with 2 GB RAM memory.

The proposed algorithm was tested on the standard benchmark set by Taillard [36]. The results of these tests are shown in Table 1. The set of Taillard's problems includes 120 instances divided into 12 groups with 10 replicates each. The sizes range from 20 jobs and 5 machines to 500 jobs and 20 machines. This benchmark has been used extensively in these years. For each problem, a very tight lower bound and upper bound are known. All ten in $50 \times 20$, nine in $100 \times 20$, six in $200 \times 20$, and three in $500 \times 20$ instances are still open. All other instances have optimum solutions.

In each case, the relative percentage deviation (RPD) was used:

$$\mathrm{RPD} = \frac{\mathrm{Heu_{sol}} - \mathrm{Best_{sol}}}{\mathrm{Best_{sol}}} \times 100.$$

In this formula, $\mathrm{Heu_{sol}}$ is the solution of the discussed heuristics for a given case and $\mathrm{Best_{sol}}$ is the optimum solution or the lowest known upper bound for Taillard problems.

http://dx.doi.org/10.22108/toc.2021.126406.1795

Table 1. Makespans and RPD's for Taillard's benchmark problems

| *Instance* | *ID* | *Heuristic Makespan* | *RPD* | *Instance* | *ID* | *Heuristic Makespan* | *RPD* |
|---|---|---|---|---|---|---|---|
| $20 \times 5$ | | | | $100 \times 5$ | | | |
| | 1 | 1324 | 0.03 | | 1 | 5564 | 0.01 |
| | 2 | 1383 | 0.01 | | 2 | 5385 | 0.02 |
| | 3 | 1246 | 0.15 | | 3 | 5325 | 0.02 |
| | 4 | 1446 | 0.11 | | 4 | 5058 | 000 |
| | 5 | 1355 | 0.09 | | 5 | 5439 | 0.03 |
| | 6 | 1268 | 0.06 | | 6 | 5235 | 0.01 |
| | 7 | 1296 | 0.05 | | 7 | 5373 | 0.02 |
| | 8 | 1283 | 0.06 | | 8 | 5270 | 0.03 |
| | 9 | 1313 | 0.06 | | 9 | 5506 | 0.01 |
| | 10 | 1167 | 0.05 | | 10 | 5386 | 0.01 |
| $20 \times 10$ | | | | $100 \times 10$ | | | |
| | 1 | 1731 | 0.09 | | 1 | 5975 | 0.03 |
| | 2 | 1821 | 0.09 | | 2 | 5678 | 0.06 |
| | 3 | 1647 | 0.10 | | 3 | 5875 | 0.03 |
| | 4 | 1505 | 0.09 | | 4 | 6143 | 0.06 |
| | 5 | 1718 | 0.21 | | 5 | 5774 | 0.05 |
| | 6 | 1501 | 0.07 | | 6 | 5552 | 0.04 |
| | 7 | 1635 | 0.10 | | 7 | 5924 | 0.05 |
| | 8 | 1689 | 0.09 | | 8 | 5887 | 0.04 |
| | 9 | 1730 | 0.08 | | 9 | 6198 | 0.05 |
| | 10 | 1831 | 0.15 | | 10 | 6114 | 0.04 |
| $20 \times 20$ | | | | $100 \times 20$ | | | |
| | 1 | 2451 | 0.06 | | 1 | 6715 | 0.09 |
| | 2 | 2245 | 0.06 | | 2 | 6768 | 0.09 |
| | 3 | 2492 | 0.07 | | 3 | 6838 | 0.04 |
| | 4 | 2484 | 0.11 | | 4 | 6679 | 0.06 |
| | 5 | 2435 | 0.06 | | 5 | 6892 | 0.10 |

http://dx.doi.org/10.22108/toc.2021.126406.1795

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 6 | 2346 | 0.05 | | 6 | 6930 | 0.09 |
| | 7 | 2410 | 0.06 | | 7 | 6881 | 0.11 |
| | 8 | 2325 | 0.05 | | 8 | 6923 | 0.09 |
| | 9 | 2381 | 0.06 | | 9 | 6815 | 0.09 |
| | 10 | 2360 | 0.08 | | 10 | 6885 | 0.07 |
| $50 \times 5$ | | | | $20 \times 10$ | | | |
| | 1 | 2774 | 0.01 | | 1 | 11281 | 0.03 |
| | 2 | 2934 | 0.03 | | 2 | 10705 | 0.02 |
| | 3 | 2710 | 0.03 | | 3 | 11159 | 0.02 |
| | 4 | 2973 | 0.08 | | 4 | 11210 | 0.02 |
| | 5 | 2935 | 0.02 | | 5 | 10897 | 0.03 |
| | 6 | 2881 | 0.01 | | 6 | 10692 | 0.03 |
| | 7 | 2888 | 0.05 | | 7 | 11165 | 0.02 |
| | 8 | 2808 | 0.04 | | 8 | 10961 | 0.02 |
| | 9 | 2654 | 0.03 | | 9 | 10736 | 0.02 |
| | 10 | 2850 | 0.02 | | 10 | 10921 | 0.02 |
| $50 \times 10$ | | | | $200 \times 20$ | | | |
| | 1 | 3288 | 0.09 | | 1 | 11952 | 0.07 |
| | 2 | 3136 | 0.09 | | 2 | 12025 | 0.06 |
| | 3 | 3154 | 0.11 | | 3 | 12051 | 0.06 |
| | 4 | 3270 | 0.06 | | 4 | 12034 | 0.06 |
| | 5 | 3262 | 0.09 | | 5 | 11825 | 0.05 |
| | 6 | 3252 | 0.08 | | 6 | 11925 | 0.06 |
| | 7 | 3462 | 0.11 | | 7 | 12175 | 0.07 |
| | 8 | 3253 | 0.07 | | 8 | 11978 | 0.05 |
| | 9 | 3261 | 0.12 | | 9 | 11971 | 0.07 |
| | 10 | 3375 | 0.10 | | 10 | 12081 | 0.07 |
| $50 \times 20$ | | | | $500 \times 20$ | | | |
| | 1 | 4161 | 0.10 | | 1 | 26957 | 0.03 |
| | 2 | 4117 | 0.12 | | 2 | 27391 | 0.03 |
| | 3 | 4010 | 0.11 | | 3 | 27337 | 0.03 |

http://dx.doi.org/10.22108/toc.2021.126406.1795

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4 | 4099 | 0.12 | | 4 | 27390 | 0.03 |
| | 5 | 4087 | 0.15 | | 5 | 27227 | 0.03 |
| | 6 | 4035 | 0.10 | | 6 | 27541 | 0.04 |
| | 7 | 4016 | 0.09 | | 7 | 27051 | 0.02 |
| | 8 | 3964 | 0.09 | | 8 | 27446 | 0.03 |
| | 9 | 4110 | 0.12 | | 9 | 27101 | 0.04 |
| | 10 | 4073 | 0.10 | | 10 | 27336 | 0.03 |

In $20 \times 5$, $20 \times 10$, and $20 \times 20$ problems, the average RPD was obtained as 6.7%, 10.7%, and 6.6%, respectively. These were desired results for 20 job problems. In $50 \times 5$, $50 \times 10$, and $50 \times 20$ problems, the average RPD was obtained as 3.2%, 9.3%, and 11%, respectively. In $100 \times 5$, $100 \times 10$ and $100 \times 20$ problems, average RPD is 1.6%, 4.5% and 8.4% respectively, that are favorable results.

For $200 \times 10$, $200 \times 20$, and $500 \times 20$ problems, the average RPD is 2.3%, 6.2%, and 3.1%, respectively, which are excellent. Thus, it seems that the proposed algorithm is suitable for large scale problems than others. It is also observed that, for a number of Taillard problems, the results of the proposed algorithm are close to the optimum. The average percentage increase over the best solution known for the proposed algorithm is shown in Table 2. It is worth mentioning that authors rarely have calculated

Table 2. Average percentage increase over the best solution known for the FRS

| *Problem* | *heuristic* |
|---|---|
| $20 \times 5$ | 6.6 |
| $20 \times 10$ | 10.7 |
| $20 \times 2$ | 6.8 |
| $50 \times 5$ | 3.4 |
| $50 \times 10$ | 9.2 |
| $50 \times 20$ | 11 |
| $100 \times 5$ | 1.6 |
| $100 \times 10$ | 4.5 |
| $100 \times 20$ | 8.3 |
| $200 \times 10$ | 2.3 |
| $200 \times 20$ | 6.2 |
| $500 \times 20$ | 3.1 |

the complexity of their algorithms for PFSP in previous decades. In Table 3, the complexity and acronyms of some algorithms are collected.

http://dx.doi.org/10.22108/toc.2021.126406.1795

Table 3. Complexity and Acronyms of Compared Algorithms

| *year* | *Author* | *Acronym* | *Complexity* |
|---|---|---|---|
| 1954 | *Johnson* | *Johns* | $O(n \log n)$ |
| 1961 | *Page* | *Page* | – |
| 1965 | *Palmar* | *Palme* | $O(mn + n \log n)$ |
| 1970 | *Campbell* | *CDS* | $O(m^2 n + n \log n)$ |
| 1972 | *Gupta* | *Gupta* | – |
| 1977 | *Dannenloring* | *RA* | $O(mn + n \log n)$ |
| | | *RACS* | – |
| | | *RAES* | – |
| 1983 | *Nawaz et al.* | *NEH* | $O(n^2 m)$ |
| 1988 | *Hunda and Rajgopal* | *HunRa* | $O(mn + n \log n)$ |
| 1991 | *Ho and Chang* | *Hocha* | – |
| 1998 | *Koulamas* | *Koula* | $O(m^2 n^2)$ |
| 2000 | *Suliman* | *Sulim* | – |
| 2001 | *Davoud Pour* | *Pour* | – |
| 2012 | *Ankau* | *CG* | $O(n^3)$ |
| | | *SG* | $O(n^2)$ |

Our results were compared with the information about famous heuristics in the literature in Tables 1 and 3 of Ruiz and Maroto [30]. However, all the algorithms have not been executed in the same computer environment. We see that our heuristic is really better than the other 13 methods. Unfortunately, the results of SG and CG are not completely mentioned in Ankau's paper [1].

As it mentioned before in many last papers, NEH algorithm is the best one among heuristic algorithms for solving PFSP. We just compare the results of the above FRS with the other 14 ones in Table 4. For every problem size, the solutions have averaged the 10 corresponding instances.

As can be seen in Table 4, the RPD results of Gupta are usually larger than the others, so they are not suitable. However, in the comparison of algorithms, both complexity and running time shouldn't be ignored. The results of Table 4 also lead to the conclusion that FRS is much better than the 13 prior algorithms' to NEH. The presented algorithm i.e. FRS outperforms Johns, Page, CDS, Gupta, RA, Racs, RAES, HunRa, Hocha, kula, Sulim, and Pour. It has better solutions in 141 cases and gives worse in 15 instances than them. In the 13 algorithms prior to FRS, Sulim is better than the others. On the other hand, FRS is about 91% better than the 13 prior algorithms but in the instances group $20 \times 5$, $20 \times 10$, $20 \times 20$, $50 \times 5$, $50 \times 10$, $50 \times 20$ and $100 \times 5$ is worse in 2, 4, 2, 3, 1, 2 and 1 cases respectively.

http://dx.doi.org/10.22108/toc.2021.126406.1795

TABLE 4. Average percentage increase over the best solution known for the heuristic algorithms

| J/M | Johns | Page | Palme | CDS | Gupta | RA | RACS | RAES | NEH | HunRa | Hocha | Koula | Sulim | Pour | FRS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 × 5 | 12.78 | 15.15 | 10.58 | 9.54 | 12.45 | 8.86 | 7.71 | 4.95 | 3.35 | 9.35 | 6.94 | 7.68 | 4.46 | 12.05 | 6.60 |
| 20 × 10 | 19.97 | 20.43 | 15.28 | 12.13 | 24.48 | 15.40 | 10.66 | 8.62 | 5.02 | 13.34 | 10.51 | 11.82 | 7.84 | 12.34 | 10.70 |
| 20 × 20 | 16.47 | 16.18 | 16.34 | 9.64 | 22.53 | 16.35 | 8.16 | 6.41 | 3.73 | 13.47 | 8.30 | 11.89 | 6.69 | 10.71 | 6.80 |
| 50 × 5 | 10.43 | 10.14 | 5.34 | 6.10 | 6.30 | 6.30 | 5.40 | 3.28 | 0.84 | 4.21 | 3.33 | 4.03 | 2.20 | 6.58 | 3.40 |
| 50 × 10 | 21.90 | 20.47 | 14.01 | 12.98 | 22.05 | 15.05 | 12.19 | 10.41 | 5.12 | 13.35 | 11.29 | 12.13 | 8.46 | 14.44 | 9.20 |
| 50 × 20 | 22.81 | 23.12 | 15.99 | 13.85 | 23.66 | 18.88 | 12.86 | 10.00 | 6.20 | 14.99 | 12.40 | 14.93 | 9.62 | 14.87 | 11.00 |
| 100 × 5 | 6.84 | 7.98 | 2.38 | 5.01 | 6.02 | 3.54 | 4.58 | 3.25 | 0.46 | 1.99 | 2.70 | 3.12 | 1.28 | 4.06 | 1.60 |
| 100 × 10 | 15.1 | 15.79 | 9.20 | 9.15 | 15.12 | 10.48 | 8.72 | 7.31 | 2.13 | 8.12 | 7.96 | 7.50 | 5.75 | 7.82 | 4.50 |
| 100 × 20 | 21.15 | 21.68 | 14.41 | 13.12 | 22.68 | 16.96 | 12.48 | 10.56 | 5.11 | 13.65 | 11.10 | 14.04 | 9.28 | 13.18 | 8.30 |
| 200 × 10 | 11.47 | 12.74 | 5.13 | 7.38 | 11.80 | 6.17 | 7.11 | 6.16 | 1.43 | 4.50 | 5.11 | 5.09 | 4.09 | 5.52 | 2.30 |
| 200 × 20 | 18.93 | 19.43 | 13.17 | 12.08 | 19.67 | 12.67 | 11.83 | 10.39 | 4.37 | 12.59 | 9.99 | 11.60 | 8.85 | 11.50 | 6.20 |
| 500 × 20 | 14.07 | 14.05 | 7.09 | 8.55 | 13.66 | 8.34 | 8.38 | 7.77 | 2.24 | 6.75 | 7.14 | 6.82 | 6.06 | 7.69 | 3.10 |
| Average | 15.99 | 16.43 | 10.74 | 9.96 | 17.21 | 11.58 | 9.17 | 7.43 | 3.33 | 9.69 | 8.06 | 9.22 | 6.21 | 10.07 | 6.14 |

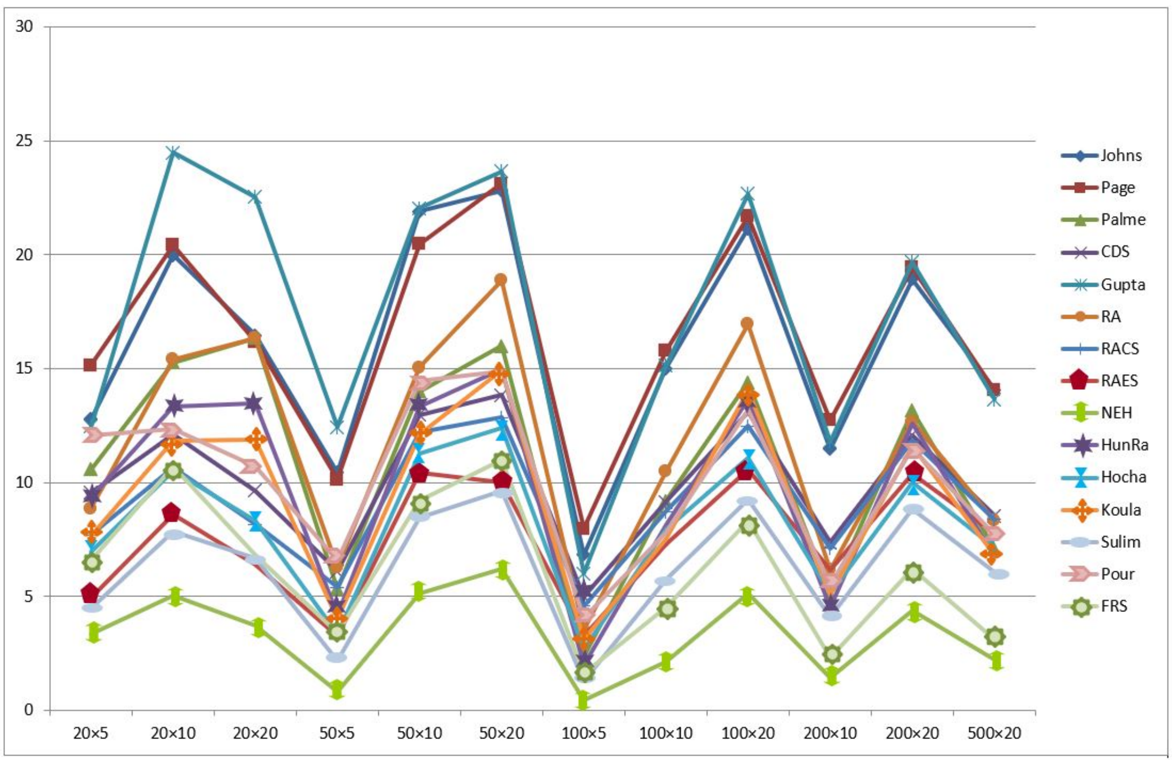


FIGURE 1. Plot of RPDs of Heuristics on Taillard problem Indtances

For a suitable visualization from statistical considerations, we have made a line graph. This graph confirms that the results of FRS are better than 13 well-known heuristics. In Figure 1, the plots depict the average RPD for all heuristics.

Means plots for the compared algorithms show the grade of all 15 algorithms in Figure 2. In this Figure, Y-axis declares average RPD obtained for all instances.

http://dx.doi.org/10.22108/toc.2021.126406.1795

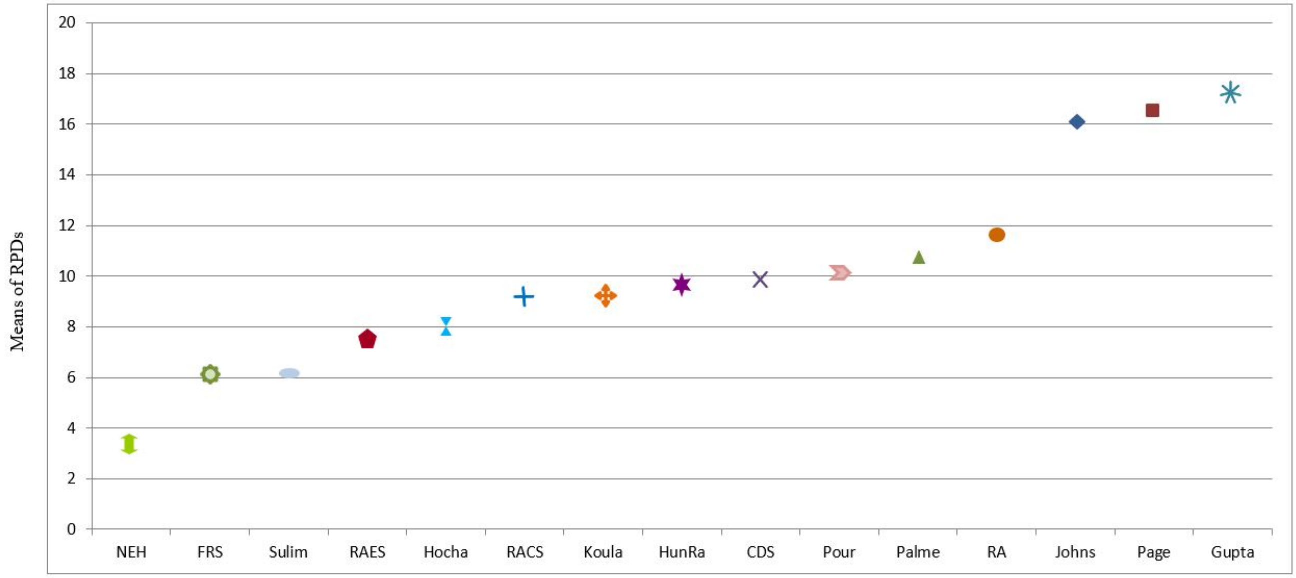


FIGURE 2. Means Plots for the Compared Algorithms

As it mentioned in the comparison, NEH solutions are better than the others and FRS solutions are close to NEH even in the problems number 26 and 92, FRS provides much better solutions. On the contrary, Jin et al. proved that the classical NEH spends a lot of time in order to find the final results in large-scale problems [17] whereas FRS gets more effective in big-size problems. First of all, in NEH the completion time of all jobs on machines could be obtained. Secondly, job sequences might be ordered on the base of the decreasing times of completion. In third place, the already jobs of partial sequences would be made by jobs and initial order iteratively. This happens in two steps to minimize the makespan

(1) Two jobs are chosen and then an ordered sequence is found by these two.
(2) For $k = 3, 4, \dots, n$, $k$th job is inserted among the terms of sequence without changing the order of $k - 1$ remaining jobs.

In this regard, there are $k$ different places to make the partial makespan minimize. The $k$th job is inserted in ordered sequence with $k - 1$ sentences. Additionally, the minimum of partial makespans is kept. The ordered sequence includes $k + 1$ jobs and minimized makespan. These repetition schedules ordered sequence with the objective of minimizing the makespan of n jobs. This type of NEH is called classic and in order to decrease the quantity of makespan, redundant sequences are inserted too.

In many efficient heuristic methods, inserting job among the others in neighborhood is used for finding high quality solution just like NEH. Thus, the key point is to employ the techniques that accelerate the evaluation of objective function. The earliest of these were devised by Taillard for improvement and also acceleration of NEH [37]. The presented method, called Taillard acceleration,

http://dx.doi.org/10.22108/toc.2021.126406.1795

is usable for PFSP problems with makespan criterion provided that the job is inserted among the other ones. Fernandez et al. discovered that, unfortunately, Taillard acceleration is not applicable to a PFSP problem unless it has the objective of optimizing makespan [8]. FRS algorithm also inserts jobs to evaluate the optimum of $C_{\max}$, that's why running time is saved by Taillard acceleration. Fernandez et al. depicted that Taillard acceleration is not suitable in PFSP with NEH-based algorithms as long as the objective is minimizing makespan [8]. Consider the case of NEHedd famous algorithm. It is intended in improvement heuristics to optimize total completion time i.e. sum of completion times of all jobs in Fernandez et. al. [7]. What gives FRS a distinct advantage over NEH is its hopeful capability of using Taillard acceleration in FRS-based heuristics. The aim in this heuristics is to minimize total completion time.

## 5. **Conclusion**

In this paper, a new algorithm (FRS) was proposed that could outperform many famous heuristics for PFSP under the minimization of the makespan on Taillard's benchmark and has the highest rank among them. The validity and effectiveness of the algorithm are shown with tables and statistical evaluations. The line graphs and scatter plots provide a clear mathematical visualization. Both FRS and NEH have brilliant performances. Their easy adoptions to similar problems are going to apply in scheduling problems.

## References


[1] M. Ancău, On Solving Flowshop Scheduling Problems, *Proc. Rom. Acad. Ser. A Math. Phys. Tech. Sci. Inf. Sci.*, **13** (2012) 71–79.

[2] R. E. Bellman, A. O. Esogbue and I. Nabeshima, *Mathematical aspects of scheduling and applications*, International Series in Modern Applied Mathematics and Computer Science, **4** (2014) Elsevier.

[3] M. C. Bonney and S. W. Gundry, Solutions to the constrained flowshop sequencing problem, *Journal of the Operational Research Society*, **27** (1976) 869–883.

[4] A. Brum and M. Ritt, Automatic Design of Heuristics for Minimizing the Makespan in Permutation Flow Shops, *In 2018 IEEE Congress on Evolutionary Computation (CEC)*, (2018) 1–8. IEEE.

[5] H. G. Campbell, R. A. Dudek and M. L. Smith, A heuristic algorithm for the n job, m machine sequencing problem, *Management Sci.*, **16** (1970) B–630.

[6] D. G. Dannenbring, An evaluation of flow shop sequencing heuristics, *Management Sci.*, **23** (1977) 1174–1182.

[7] V. Fernandez-Viagas and J. M. Framinan, NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness, *Comput. Oper. Res.*, **60** (2015) 27–36.

[8] V. Fernandez-Viagas, J. M. Molina-Pariente and J. M. Framinan, Generalised accelerations for insertion-based heuristics in permutation flowshop scheduling, *European J. Oper. Res.*, **282** (2020) 858–872.

[9] V. Fernandez-Viagas, R. Ruiz and J. M. Framinan, A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation, *European J. Oper. Res.*, **257** (2017) 707–721.

[10] J. M. Framinan, J. N. Gupta and R. Leisten, A review and classification of heuristics for permutation flow-shop scheduling with makespan objective, *Journal of the Operational Research Society*, **55** (2004) 1243–1255.


http://dx.doi.org/10.22108/toc.2021.126406.1795

[11] M. R. Garey, D. S. Johnson and R. Sethi, The complexity of flowshop and jobshop scheduling, *Math. Oper. Res.*, **1** (1976) 117–129.

[12] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Ann. Discrete Math.*, **5** (1979) 287–326. Elsevier.

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

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

[15] J. C. Ho and Y. L. Chang, A new heuristic for the n-job, M-machine flow-shop problem, *European J. Oper. Res.*, **52** (1991) 194–202.

[16] T. S. Hundal and J. Rajgopal, An extension of Palmer's heuristic for the flow shop scheduling problem, *International Journal Of Production Research*, **26** (1988) 1119–1124.

[17] F. Jin, S. Song and C. Wu, An improved version of the NEH algorithm and its application to large-scale flow-shop scheduling problems, *IIE Trans.*, **39** (2007) 229–234.

[18] Johnson, S. M. Optimal twoand threestage production schedules with setup times included, *Naval Res. Logist. Quart.*, **1** (1954) 61-68.

[19] Kalczynski, P. J., & Kamburowski, J. On the NEH heuristic for minimizing the makespan in permutation flow shops, *Omega*, **35** (2007) 53-60.

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

[21] Koulamas, C. A new constructive heuristic for the flowshop scheduling problem, *European J. Oper. Res.*, **105** (1998) 66-71.

[22] Liu, W., Jin, Y., & Price, M. A new NawazEnscoreHam-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time, *Eng. Optim.*, **48** (2016) 1808-1822.

[23] Malik, A., & Dhingra, A. K. Comparative analysis of heuristics for make span minimizing in flow shop scheduling, *International Journal of Innovations in Engineering and Technology*, **2** (2013) 263-269.

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

[25] Nurdiansyah, R., Rijanto, O. A. W., Santosa, B., & Wiratno, S. E. An Improved Differential Evolution Algorithm for Permutation Flow Shop Scheduling Problem, *Int. J. Oper. Res. (Taichung)*, **16** (2019) 37-44.

[26] Page, E. S. An approach to the scheduling of jobs on machines, *J. Roy. Statist. Soc. Ser. B (Methodological)*, **23** (1961) 484-492.

[27] Palmer, D. S. 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** (1965) 101-107.

[28] Potts, C. N., & Strusevich, V. A. Fifty years of scheduling: a survey of milestones, *Journal of the Operational Research Society*, **60** (2009) S41-S68.

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

[30] Ruiz, R., & Maroto, C. A comprehensive review and evaluation of permutation flowshop heuristics, *European J. Oper. Res.*, **165** (2005) 479-494.

[31] Sarin, S., & Lefoka, M. Scheduling heuristic for the n-jobm-machine flow shop, *Omega*, **21** (1993) 229-234.

[32] Sauvey, C., & Sauer, N. Two NEH Heuristic Improvements for Flowshop Scheduling Problem with Makespan Criterion, *Algorithms*, **13** (2020) 112.

[33] Singhal, E., Singh, S., & Dayma, A. An Improved Heuristic for Permutation Flow Shop Scheduling, In *(NEH ALGORITHM). International Journal of Computational Engineering Research*, **2** (2012) 95-100.

http://dx.doi.org/10.22108/toc.2021.126406.1795

[34] STINSON, J. P., & SMITH, A. W. A heuristic prorammin procedure for sequencin the static flowshop, *The International Journalof Production Research*, **20** (1982) 753-764.

[35] Suliman, S. M. A. A two-phase heuristic approach to the permutation flow-shop scheduling problem, *International Journal of Production Economics*, **64** (2000) 143-152.

[36] Taillard, E. Benchmarks for basic scheduling problems, *European J. Oper. Res.*, **64** (1993) 278-285.

[37] Taillard, E. Some efficient heuristic methods for the flow shop sequencing problem, *European J. Oper. Res.*, **47** (1990) 65-74.

[38] T'kindt, V., & Billaut, J. C. *Multicriteria scheduling: theory, models and algorithms*, Springer Science & Business Media, (2006).

[39] E. Vallada, R. Ruiz and J. M. Framinan, New hard benchmark for flowshop scheduling problems minimising makespan, *European J. Oper. Res.*, **240** (2015) 666-677.

[40] J. Xu, Y. Yin, T. C. E. Cheng, C. C. Wu and S. Gu, An improved memetic algorithm based on a dynamic neighbourhood for the permutation flowshop scheduling problem, *Int. J. Prod. Res.*, **52** (2014) 1188–1199.

**SHahriar Farahmand Rad**

Department of Mathematics, Payame Noor University of ABCD, P.O.Box 19395-3697, Tehran, Iran

Email: Sh_fmand@pnu.ac.ir