Iran J Sci Technol Trans Sci (2022) 46:181–188
https://doi.org/10.1007/s40995-021-01202-4

RESEARCH PAPER

A New Ant Algorithmic Approach for Solving PFSP

Shahriar Farahmand Rad[1]

Received: 13 February 2021 / Accepted: 5 August 2021 / Published online: 3 October 2021

Abstract
In this paper, a new ant algorithmic approach is presented for solving n-job, m-machine permutation flow shop scheduling problem. The main objective is to find a permutation of n given jobs, i.e., $\sigma: \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$. This permutation minimizes the maximum completion time of the schedule arising from σ. An illustration of using the presented heuristic algorithm for finding a good initial sequence of jobs is given. The proposed method is an ant-based approach to permutation flow shop scheduling problem by the behavior of real ants, but it is different with the pheromone trail concept. The presented model is compared against the one by NEH which has been considered the best constructive algorithm so far. Regarding the quality of results, the superiority of the proposed method over NEH is demonstrated by computational evaluation. The comparison is produced on generated random test problems. This comparison is drawn in domain of feasible instances. It is easy to implement the produced method as a metaheuristic.

Keywords Scheduling · Heuristics · Permutation flow shop · Makespan · NEH algorithm · Initial sequence

1 Introduction

A permutation flow shop scheduling problem (PFSP) is known to determine a sequence of n jobs that are processed on m independent machines such that the makespan or $C_{\max}$ is minimized. Makespan is the distance in time that elapses from the start of processing to the end.

Although the result of $n \times m$ PFSP for $m = 2$ was obtained in polynomial time, it was proved that problem is NP-complete in the strong sense for $m \geq 3$ (Garey et al. 1976). Therefore, when m and n are increased, the methods giving the optimum solutions are impractical. That is why there is a preference for heuristic algorithm.

Initial sequences of jobs are derived in many heuristic algorithms; thereafter, the heuristic approach consists of its performance evaluation. Framinan et al. (2003) proposed 177 different initial orders in NEH-insertion approach. The execution of Kurz and Askin (2004), Leung et al. (2005), Allahverdi and Al-Anzi (2006), Alaykýran et al. (2007), Kalczynski and Kamburowski (2008), Rad et al. (2009), Ancău (2012), Malik and Dhingra (2013), Xu et al. (2014), Liu et al. (2016), Brum and Ritt (2018), Nurdiansyah et al. (2019) and Sauvey and Sauer (2020) heuristics was proposed after obtaining suitable initial sequences.

A PFSP is supposed in conditions below:

1. All jobs are processed on the same machines.
2. Each job is processed at most once on machines number 1 to m with the same order.
3. All jobs are independent and ready for processing in zero time.
4. There is one-to-one correspondence between the processing of jobs and machines.
5. Buffering storage between every two machines is infinite.
6. There are not parallel machines.
7. Machines are available continuously.
8. The processing of jobs cannot pass each other.

Since all jobs are processed on the same machines, the number of results for a $n \times m$ problem is reduced from $(n!)^m$ to $n!$. Let a_{ij} be the processing time of ith job on jth machine.

✉ Shahriar Farahmand Rad
sh_fmand@pnu.ac.ir

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

Suppose the matrix of processing times is $M = \left[a_{ij}\right]_{n\times m}$. Consider the following weighted directed graph (Fig. 1).

In a PFSP for minimizing the $C_{\max}$, a suitable ordering of jobs is searched. According to the mentioned conditions, the columns of matrix M or the graph cannot be replaced, because columns are corresponding machines. The rows must be permuted such that $C_{\max}$ becomes minimized. The weight of the vertex situated in ith row and jth column is equal to a_{ij}, $(i = 1, 2, \ldots, n)$ and $(j = 1, 2, \ldots, m)$. The weight of every path in the graph is the total of weights of its vertices.

For every permutation denoted by π, $\pi(j)$ is the job in jth row. Then

$$C_{i,\pi(j)} = \max\left\{C_{i-1,\pi(j)}, C_{i,\pi(j-1)}\right\} + a_{i\pi(j)},$$

$$C_{\max} = C_{m,\pi(n)}.$$

In these relations, C_{ji} is the completion time of ith job on jth machine. It can also be written for every π and $j = 1, 2, \ldots, m$

$$C_{0\pi(j)} = 0,\ C_{1\pi(j)} = a_{1\pi(1)} + \cdots + a_{1\pi(j)}$$

$$C_{i,\pi(0)} = 0,\ C_{i\pi(1)} = a_{1\pi(1)} + \cdots + a_{i\pi(1)}.$$

2 Bellman, Esogbue, Nabeshima Theorem

In every PFSP with the time matrix $M = \left[a_{ij}\right]$, makespan is equal to the weight of the heavy path that is traced between a_{11} and a_{nm} in corresponding directed graph (Bellman et al. 2014).

According to this theorem, let every vertex a_{ij} be the food with a_{ij} weight in the mentioned graph. An ant wants to go from a_{11} to a_{nm} after collecting the heaviest foods with knowing all the weight of a_{ij}s.

$$\begin{tikzcd}
a_{11} \arrow[r] \arrow[d] & a_{12} \arrow[r] \arrow[d] & a_{13} \arrow[r] \arrow[d] & \cdots \arrow[r] & a_{1,m-1} \arrow[r] \arrow[d] & a_{1m} \arrow[d] \\
a_{21} \arrow[r] \arrow[d] & a_{22} \arrow[r] \arrow[d] & a_{23} \arrow[r] \arrow[d] & \cdots \arrow[r] & a_{2,m-1} \arrow[r] \arrow[d] & a_{2m} \arrow[d] \\
a_{31} \arrow[r] \arrow[d] & a_{32} \arrow[r] \arrow[d] & a_{33} \arrow[r] \arrow[d] & \cdots \arrow[r] & a_{3,m-1} \arrow[r] \arrow[d] & a_{3m} \arrow[d] \\
\vdots \arrow[d] & \vdots \arrow[d] & \vdots \arrow[d] & & \vdots \arrow[d] & \vdots \arrow[d] \\
a_{n1} \arrow[r] & a_{n2} \arrow[r] & a_{n3} \arrow[r] & \cdots \arrow[r] & a_{n,m-1} \arrow[r] & a_{nm}
\end{tikzcd}$$

Fig. 1 The weighted directed graph of processing times

The objective is giving an order of rows in M or graph such that the collected food be minimized.

Based on this intuition, the heavy entries must be out of access. Indeed, whenever ant obtains food in a vertex, other heavy foods get out of reach. Hence, if ant reaches a_{ij}, it does not attain the vertex with less column and row numbers. In fact, there will be no loop. Therefore, as the following figure is shown, if the line that connects two vertices has positive slope, ant does not receive both of them (Fig. 2).

With due attention to this idea, the heavy entries are considered and rows are permuted such that the line that connects these vertices has positive slope. This is certainly impossible for all vertices. In this situation, the best permutation of rows is resulted from the amount of makespans. These are the motivation of the presented algorithm.

3 The Proposed Algorithm

All of the prior concepts lead to the following steps. These steps will result in a good initial order of jobs.

1. The greatest entry in M is chosen, and the corresponding row and column of it are deleted.
2. The above step is repeated on the rest of entries, and the greatest chosen entries in total make a finite sequence of m members.
3. All of m elected members in the last step are deleted in M.
4. The loop is continued n times.

For each sequence of entries, m corresponding rows are arranged. Suppose the sequence $a_{i_1 j_1}, \ldots, a_{i_m j_m}$ is the resulted one in the above process. Since $m \leq n$ and $j_1, \ldots, j_m$ are distinct, $j_1, j_2, \ldots, j_m$ is a permutation of $1, 2, \ldots, m$. The indices $j_1, j_2, \ldots, j_m$ are distinct plus they are not all of the $1, 2, \ldots, n$.

The rows $r_{i_1}, \ldots, r_{i_m}$ are arranged such that the slope of lines connecting each elected two entries is positive.

The furthest row is r_{i_h} with $j_{i_h} = m$. The next row is $r_{i_{h-1}}$ with $j_{i_{h-1}} = m - 1$, and it will be repeated in this manner for the next rows until the last one being r_{i_1} with $j_{i_1} = 1$. Since the column numbers of elected entries in a group are all of the numbers $1, 2, \ldots, m$, these entries can be numerated in the from $a_{i_1 1}, \ldots, a_{i_m m}$. Hence, smaller row means the row that is under the greater one.

A distinguished ordering among the rows of elected entries has been constructed. But for every such ordering, a value can be allocated. For determining the value of ordering $r_{i_l} < r_{i_h}$, the profit of this ordering must be checked. Based on Bellman's theorem (2014), only the bigger one in the $a_{i_l l}, a_{i_h h}$ is chosen. In the inverse ordering, both of entries can be chosen. Thus, the profit of this ordering can be defined $\min\{a_{i_h h}, a_{i_l l}\}$. This means that the inverse ordering, i.e., $r_{i_h} < r_{i_l}$ has a disadvantage that is equal to $\min\{a_{i_l l}, a_{i_h h}\}$. The negative of this number is proposed as the profit of $r_{i_l} < r_{i_h}$ ordering. Now for each group that has been obtained, there are m rows with distinguishing ordering. Then, for every two rows in it, the obtained profit is constructed from the order of them. In the inverse ordering of these two rows, the profit becomes negative. Therefore, the square matrix $P = [p_{ij}]$ is defined. Every p_{ij} denotes the sum of $r_i < r_j$ ordering profits $(\hat{P}_{ij})$ and $r_j < r_i$ ordering disadvantages $(\hat{\hat{P}}_{ij})$. Indeed, for every two indices i and j, it is possible that r_i and r_j arise in the some of the mentioned corresponding sequences. Clearly $p_{ij} = -p_{ji}$, $P_{ij} = \hat{P}_{ij} + \hat{\hat{P}}_{ij}$.

For every arrangement of rows in M that is obtained from a permutation $\sigma: \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, matrix P denotes the amount of correctness for place and order of each row. For example, suppose there is $r_{\sigma(i)}$ of M in ith row. Then, the amount of correctness for place of $r_{\sigma(i)}$ in ith row is

$$p_{ij} = p_{\sigma(n)\sigma(i)} + \cdots + p_{\sigma(i+1)\sigma(i)} + p_{\sigma(i)\sigma(i-1)} + \cdots + p_{\sigma(i)\sigma(1)}, \ (i = 1, 2, \ldots, n), \ (j = 1, 2, \ldots, n).$$

Obviously $p_{ij} = 0$ when $i = j$.

4 Example for Constructing Matrix *P*

Let the processing times of four jobs on three machines be as the following:

$$\begin{array}{c} (1) \\ (2) \\ (3) \\ (4) \end{array} \begin{bmatrix} 50 & 13 & 62 \\ 5 & 84 & \textcircled{89} \\ 1 & 44 & 22 \\ 36 & 14 & 19 \end{bmatrix} = M \quad . \ \text{Then}$$

$$\text{I)}\, M \xrightarrow{89} \begin{bmatrix} \boxed{50} & 13 & 0 \\ 0 & 0 & 0 \\ 1 & 44 & 0 \\ 36 & 14 & 0 \end{bmatrix} \xrightarrow{50} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \langle 44 \rangle & 0 \\ 0 & 14 & 0 \end{bmatrix} \xrightarrow{44} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \to (89, 50, 44)$$

$$\text{II)} \begin{bmatrix} 0 & 13 & 62 \\ 5 & \textcircled{84} & 0 \\ 1 & 0 & 22 \\ 36 & 14 & 19 \end{bmatrix} \xrightarrow{84} \begin{bmatrix} 0 & 0 & \boxed{62} \\ 0 & 0 & 0 \\ 1 & 0 & 22 \\ 36 & 0 & 19 \end{bmatrix} \xrightarrow{62} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ \langle 36 \rangle & 0 & 0 \end{bmatrix} \xrightarrow{36} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \to (84, 62, 36)$$

$$\text{III)} \begin{bmatrix} 0 & 13 & 0 \\ 5 & 0 & 0 \\ 1 & 0 & \textcircled{22} \\ 0 & 14 & 19 \end{bmatrix} \xrightarrow{22} \begin{bmatrix} 0 & 13 & 0 \\ 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \boxed{14} & 0 \end{bmatrix} \xrightarrow{14} \begin{bmatrix} 0 & 0 & 0 \\ \langle 5 \rangle & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{5} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \to (22, 14, 5)$$

$$\text{IV)} \begin{bmatrix} 0 & \textcircled{13} & 0 \\ 0 & 0 & 0 \\ \langle 1 \rangle & 0 & 0 \\ 0 & 0 & \boxed{19} \end{bmatrix} \to (19, 13, 1).$$

Consider the lines connecting two entries of (59, 80, 44) and other resulted sequences. The slopes of these must be positive, therefore;

$$\text{I)}\ (89,50,44)\rightarrow \begin{array}{c} (2)\\ (3)\\ (1)\end{array}\begin{bmatrix} 5 & 84 & 89\\ 1 & 44 & 22\\ 50 & 13 & 62\end{bmatrix}=A$$

$$\text{II)}\ (84,62,36)\rightarrow \begin{array}{c} (1)\\ (2)\\ (4)\end{array}\begin{bmatrix} 0 & 13 & 62\\ 5 & 84 & 0\\ 36 & 14 & 19\end{bmatrix}=B$$

$$\text{III)}(22,14,5)\rightarrow \begin{array}{c} (3)\\ (4)\\ (2)\end{array}\begin{bmatrix} 1 & 0 & 22\\ 0 & 14 & 19\\ 5 & 0 & 0\end{bmatrix}=C$$

$$\text{IV)}(19,13,1)\rightarrow \begin{array}{c} (4)\\ (1)\\ (3)\end{array}\begin{bmatrix} 0 & 0 & 19\\ 0 & 13 & 0\\ 1 & 0 & 0\end{bmatrix}=D$$

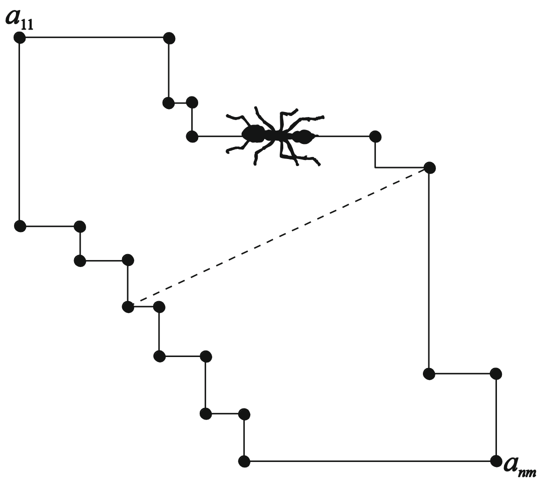

Fig. 2 The path of the ant

$$A \Rightarrow r_1 < r_3 < r_2 \Rightarrow \hat{P}_{13} = \min\{50, 44\} = 44,$$
$$\hat{P}_{12} = \min\{50, 89\} = 50,$$
$$\hat{P}_{32} = \min\{44, 89\} = 44 \Rightarrow \hat{\hat{P}}_{31} = -44, \hat{\hat{P}}_{21} = -50,$$
$$\hat{\hat{P}}_{23} = -44$$
$$B \Rightarrow r_4 < r_2 < r_1 \Rightarrow \hat{P}_{42} = \min\{36, 84\} = 36,$$
$$\hat{P}_{41} = \min\{36, 62\} = 36,$$
$$\hat{P}_{21} = \min\{84, 62\} = 62 \Rightarrow \hat{\hat{P}}_{24} = -36,$$
$$\hat{\hat{P}}_{14} = -36, \hat{\hat{P}}_{12} = -62$$
$$C \Rightarrow r_2 < r_4 < r_3$$
$$\Rightarrow \hat{P}_{24} = \min\{5, 14\} = 5, \hat{P}_{23} = \min\{5, 22\} = 5,$$
$$\hat{P}_{43} = \min\{14, 22\} = 14 \Rightarrow \hat{\hat{P}}_{42} = -5,$$
$$\hat{\hat{P}}_{32} = -5, \hat{\hat{P}}_{34} = -14$$
$$D \Rightarrow r_3 < r_1 < r_4 \Rightarrow \hat{\hat{P}}_{31} = \min\{1, 13\} = 1,$$
$$\hat{\hat{P}}_{34} = \min\{1, 19\} = 1,$$
$$\hat{P}_{14} = \min\{13, 19\} = 13 \Rightarrow \hat{\hat{P}}_{13} = -1,$$
$$\hat{\hat{P}}_{43} = -1, \hat{\hat{P}}_{41} = -13\,.$$
$$P_{ij} = \hat{P}_{ij} + \hat{\hat{P}}_{ij},\ 1 \le i \le 4,\ 1 \le j \le 4$$
$$P = [p_{ij}] =$$
$$\begin{bmatrix} 0 & 50-62 & 44-1 & -36+13\\ 62-50 & 0 & -44+5 & -36+5\\ -44+1 & 44-5 & 0 & -14+1\\ 36-13 & -5+36 & 14-1 & 0\end{bmatrix}$$
$$P = \begin{bmatrix} 0 & -12 & 43 & -26\\ 12 & 0 & -39 & -31\\ -43 & 39 & 0 & -13\\ 26 & 31 & 13 & 0\end{bmatrix}.$$

It is possible that $p_{ij} > 0$ or $p_{ij} < 0$. The best place for a row is where the corresponding p_{ij} has the greatest amount, and thus, the row with the least amount of p_{ij} is in the worst place.

After constructing the matrix $P = [p_{ij}]$, the row with the worst place is determined, and the best place for it is obtained with a local search. Then, the penultimate row can be chosen, and the mentioned operations are repeated.

These operations are continued till the last row. At the end, a well ordering is found in which every replacement of its rows does not provide a better solution. A situation improves when a better makespan is found.

A suitable initial sequence for processing n jobs on m machines can be obtained from $P = [p_{ij}]$. The Bellman

Table 1 The frequency of the superior results in 100 random problems

NEH \ New Heuristic	m=10		m=20		m=30		m=40		m=50	
	NEH	New Heuristic	NEH	New Heuristic	NEH	New Heuristic	NEH	New Heuristic	NEH	New Heuristic
n=10	29	71	33	67	24	76	30	70	28	72
n=20	27	73	25	75	15	85	21	79	26	74
n=30	27	73	13	87	17	83	20	80	15	85
n=40	23	77	15	85	8	92	17	83	22	78
n=50	25	75	17	83	7	93	11	89	17	83
n=60	27	73	16	84	8	92	7	93	11	89
n=70	29	71	15	85	9	91	13	87	10	90
n=80	40	60	17	83	8	92	15	85	10	90
n=90	36	64	16	84	16	84	8	92	11	89
n=100	35	65	19	81	10	90	8	92	7	93

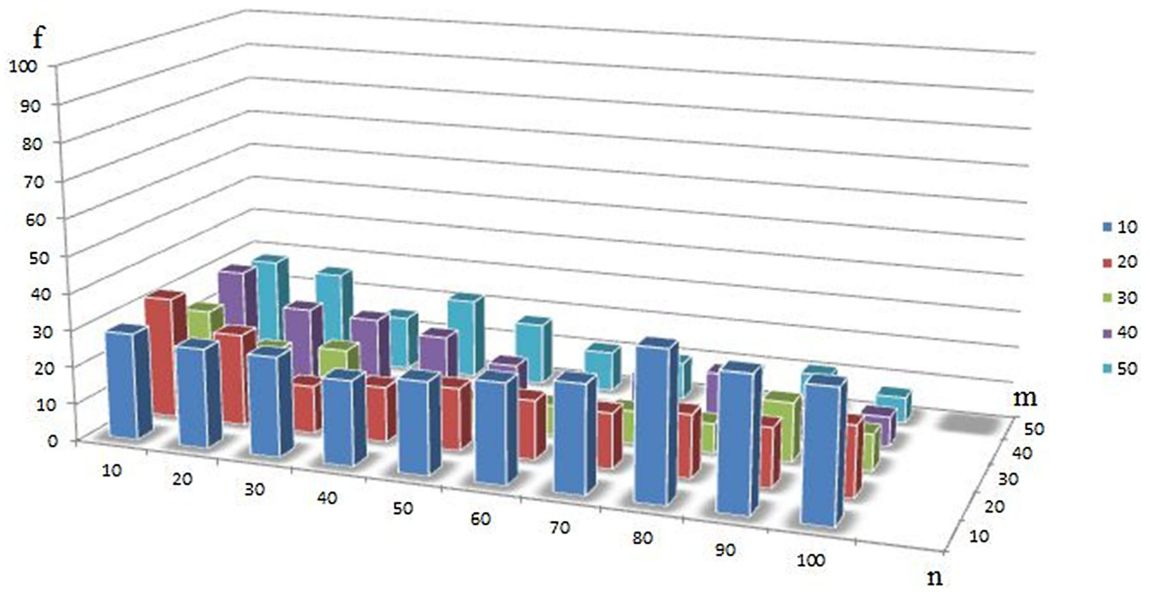

Fig. 3 Three-dimensional histogram of NEH superior results

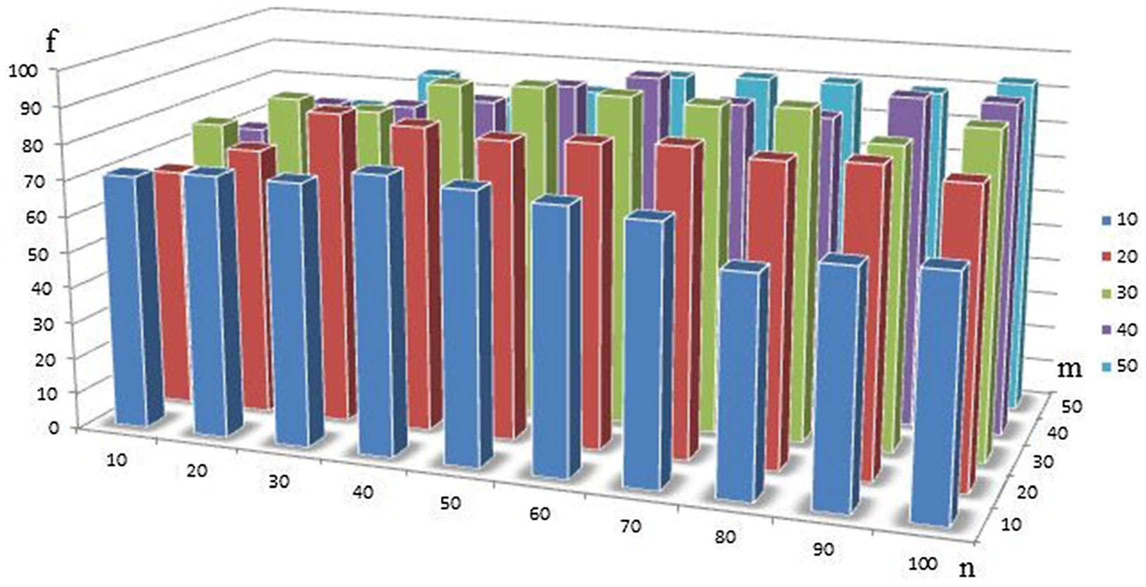

Fig. 4 Three-dimensional histogram of heuristic algorithm's superior results

et al. theorem (Bellman et al. 2014) is used to determine makespans.

5 Computational Evaluation

The heuristic algorithm was implemented in Visual Basic and carried on a Pentium IV PC/AT computer running at 3.2 GHz with 2 GB RAM memory. The proposed algorithm was tested on 100 generated random problems with the sizes $n = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$ and $m = 10, 20, 30, 40, 50$ in the feasible domain. The results of these tests are shown in Table 1 and were compared with the results of NEH [the best heuristic algorithm in classic methods (Nawaz et al. 1983)] on the same random problems. There are two numbers in each table cell. These numbers represent how many times corresponding algorithm has better results than the other.

The main conclusion to be drawn from this table is that in 82% of times, the heuristic algorithm has better results than NEH. Also in large-scale problems, this improvement is increased.

The results are compared in the following figures (Figs. 3, 4).

In Table 2, the running times of the problems are in second.

The approximate relation $T_{\text{Heu}} \cong 0/15 \times m\,T_{\text{NEH}}$ is resulted on the basis of the data in Table 2. Notations T_{Heu} and T_{NEH} are, respectively, referred to the running times of heuristic algorithm and NEH in each allocation of the table.

6 Time Complexity

The complexity of sorting mn entries of the matrix M is $O(m^2n^2)$. There is less running time for dividing these entries into m groups and counting the number of operations. The calculation of p_{ij} s has less computational used time in the construction of matrix $P = [p_{ij}]$. Therefore, the complexity of the presented heuristic algorithm is $O(m^2n^2)$.

7 Conclusions

In this paper, a reasonably good initial sequence was obtained for solving PFSP with minimizing makespan criterion. After running the heuristic algorithm on 100 random problems, the results in 82% of times were better than NEH. For future researching lines, the presented algorithm with due attention to its mathematical properties could be used as a metaheuristic.

Table 2 Running times of heuristic algorithms

Nos	$n \times m$	New algorithm	NEH
1	10 × 10	0.0006	0
2	10 × 20	0.0018	0
3	10 × 30	0.0042	0
4	10 × 40	0.0065	0
5	10 × 50	0.0082	0
6	20 × 10	0.0034	0.0156
7	20 × 20	0.0088	0
8	20 × 30	0.0167	0
9	20 × 40	0.0299	0
10	20 × 50	0.0437	0.0078
11	30 × 10	0.0065	0
12	30 × 20	0.0212	0.0156
13	30 × 30	0.418	0.0078
14	30 × 40	0.685	0
15	30 × 50	0.1013	0.0078
16	40 × 10	0.0127	0
17	40 × 20	0.0374	0.0078
18	40 × 30	0.0807	0.0018
19	40 × 40	0.1308	0.0078
20	40 × 50	0.1943	0.0156
21	50 × 10	0.0196	0.0078
22	50 × 20	0.0682	0.0234
23	50 × 30	0.1309	0.0234
24	50 × 40	0.2079	0.0312
25	50 × 50	0.3109	0.0468
26	60 × 10	0.0294	0.0117
27	60 × 20	0.987	0.0195
28	60 × 30	0.1949	0.3225
29	60 × 40	0.3053	0.0429
30	60 × 50	0.4634	0.0585
31	70 × 10	0.0373	0.0195
32	70 × 20	0.138	0.0507
33	70 × 30	0.281	0.0507
34	70 × 40	0.448	0.0898
35	70 × 50	0.748	0.1718
36	80 × 10	0.0511	0.0234
37	80 × 20	0.1870	0.0703
38	80 × 30	0.3869	0.1093
39	80 × 40	0.7514	0.1914
40	80 × 50	1.1805	0.2382
41	90 × 10	0.0639	0.0390
42	90 × 20	0.2411	0.0898
43	90 × 30	0.5737	0.1914
44	90 × 40	1.0782	0.2695
45	90 × 50	1.5966	0.3281
46	100 × 10	0.0706	0.0585
47	100 × 20	0.2932	0.1210
48	100 × 30	0.7044	0.2695
49	100 × 40	1.4287	0.3710
50	100 × 50	2.0510	0.4531

Authors' contribution All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Shahriar Farahmand Rad. The first draft of the manuscript was written by Shahriar Farahmand Rad and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

Alaykýran K, Engin O, Döyen A (2007) Using ant colony optimization to solve hybrid flow shop scheduling problems. Int J Adv Manuf Technol 35(5):541–550

Allahverdi A, Al-Anzi FS (2006) A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. Comput Oper Res 33(4):1056–1080

Ancău M (2012) On solving flowshop scheduling problems. Proc Roman Acad Ser A 13(1):71–79

Bellman R, Esogbue AO, Nabeshima I (2014) Mathematical aspects of scheduling and applications: modern applied mathematics and computer science, vol 4. Elsevier, Amsterdam

Brum A, Ritt M (2018) Automatic design of heuristics for minimizing the makespan in permutation flow shops. In: 2018 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8

Framinan JM, 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. Int J Prod Res 41(1):121–148

Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1(2):117–129

Kalczynski PJ, Kamburowski J (2008) An improved NEH heuristic to minimize makespan in permutation flow shops. Comput Oper Res 35(9):3001–3008

Kurz ME, Askin RG (2004) Scheduling flexible flow lines with sequence-dependent setup times. Eur J Oper Res 159(1):66–82

Leung JYT, Li H, Pinedo M (2005) Order scheduling in an environment with dedicated resources in parallel. J Sched 8(5):355–386

Liu W, Jin Y, Price M (2016) A new Nawaz–Enscore–Ham-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time. Eng Optim 48(10):1808–1822

Malik A, Dhingra AK (2013) Comparative analysis of heuristics for makespan minimising in flow shop scheduling. Int J Innov Eng Technol 2(4):263–269

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

Nurdiansyah R, Rijanto OAW, Santosa B, Wiratno SE (2019) An improved differential evolution algorithm for permutation flow shop scheduling problem. Int J Oper Res 16(2):37–44

Rad SF, Ruiz R, Boroojerdian N (2009) New high performing heuristics for minimizing makespan in permutation flowshops. Omega 37(2):331–345

Sauvey C, Sauer N (2020) Two NEH heuristic improvements for flowshop scheduling problem with makespan criterion. Algorithms 13(5):112

Xu J, Yin Y, Cheng TCE, Wu CC, Gu S (2014) An improved memetic algorithm based on a dynamic neighbourhood for the permutation flowshop scheduling problem. Int J Prod Res 52(4):1188–1199