MARI

MULTIDISCIPLINARY APPLIED RESEARCH AND INNOVATION
e-ISSN: 2773-4773

Vol. 7 No. 3 (2026) p. 300-307
https://publisher.uthm.edu.my/periodicals/index.php/mari

New Alternate Constructive Heuristic Algorithm (NACHA) in Solving Permutation Flow Shop Scheduling (PFSSP)

Vishal Vengadesan[1], Noor Azizah Sidek[1,2]*, Azli Nawawi[3]

[1] *Department of Mechanical Engineering, Centre for Diploma Studies*
Universiti Tun Hussein Onn Malaysia, Pagoh Higher Education Hub, 84600 Pagoh, Johor, MALAYSIA
[2] *Product Research and Development (ProRed), Centre for Diploma Studies*
Universiti Tun Hussein Onn Malaysia, Pagoh Higher Education Hub, 84600 Pagoh, Johor, MALAYSIA
[3] *Department of Mechanical Engineering Technology, Faculty of Engineering Technology*
Universiti Tun Hussein Onn Malaysia, Pagoh Higher Education Hub, 84600 Pagoh, Johor, MALAYSIA

*Corresponding Author: noorazizah@uthm.com.my
DOI: https://doi.org/10.30880/mari.2026.07.03.038

Article Info

Received: 01 September 2026
Accepted: 15 October 2026
Available online: 01 December 2026

Keywords

Permutation Flow Shop Scheduling (PFSSP), NEH Heuristic, NACHA, makespan optimization, bottleneck identification

Abstract

Flow shop scheduling is a common challenge in manufacturing because every job must follow the same machine order, and even a small delay can slow down the entire process. When jobs are not arranged properly, machines end up waiting, the workflow becomes unbalanced, and the total completion time increases. This study focuses on the Permutation Flow Shop Scheduling Problem (PFSSP) and evaluates two constructive heuristics, NEH and the New Alternate Constructive Heuristic Algorithm (NACHA). NEH is well known for producing strong baseline solutions, while NACHA aims to improve job insertion decisions by analysing dominance patterns and sequence behaviour more closely. Using a selected dataset consisting 10 jobs 5 machines with predefined processing times, both algorithms were applied to generate sequences, construct Gantt charts, and calculate makespan and idle time. The results showed that NEH generally outperformed NACHA, with negative PI values observed in six datasets (−0.53% to −3.68%), while NACHA only showed stronger performance in two datasets (+2.11% and +5.29%), and equal performance in the remaining cases. Although the improvement in selected cases is modest, it highlights the potential of refined insertion strategies in enhancing flow shop efficiency. Overall, NEH remains more consistent, while NACHA can serve as a viable alternative heuristic in specific scenarios.

1. Introduction

Scheduling is an important activity in many industrial and service systems because it helps control how jobs, machines, and time are managed. In manufacturing, one common scheduling problem is the Permutation Flow Shop Scheduling Problem (PFSSP). In PFSSP, all jobs must go through the machines in the same order, and the main goal is to reduce the total time needed to complete all jobs, which is known as makespan [1]. If the job sequence is not planned properly, machines may wait unnecessarily, and the overall production time becomes longer.

This is an open access article under the CC BY-NC-SA 4.0 license.

Due to this complexity, many researchers have studied PFSSP and proposed different solution methods. Heuristic and metaheuristic approaches are widely used because finding the exact optimal solution becomes very difficult when the number of jobs and machines increases. One of the most popular heuristic methods is the Nawaz-Enscore-Ham (NEH) algorithm, which creates job sequences based on processing times [2]. Although NEH is simple and effective, it does not always give the best solution, especially for more complicated scheduling cases. To improve this, newer constructive methods have been introduced. One such method is the New Alternate Constructive Heuristic Algorithm (NACHA), which considers machine bottlenecks and early job arrangement to improve scheduling performance [3]. This paper follows the MARI format and presents a comparison between NEH and NACHA, including background, methodology, results, and conclusions.

1.1 Problem Statement

In manufacturing systems, poor job sequencing often causes machine bottlenecks, long waiting times, and inefficient use of resources. When machines remain idle or jobs are not arranged properly, the total production time increases. This problem is commonly seen in flow shop environments where all jobs follow the same machine order. Previous studies have shown that PFSSP is difficult to solve optimally because the number of possible job sequences increases rapidly as the system becomes larger.

Although traditional heuristics such as NEH can produce reasonable solutions, they do not always provide the best job sequence. In many cases, NEH does not fully consider machine bottlenecks, which can lead to unnecessary idle time. The NACHA method was developed to improve job insertion decisions and reduce machine idle gaps. However, the actual performance of NACHA compared to NEH needs to be tested using real datasets and proper analysis. Therefore, this study investigates whether a bottleneck-aware heuristic (NACHA) can outperform NEH under varying dataset structures.

1.2 Objectives, Scope and Significance

The objectives of this study are:

1. To understand how the NEH heuristic works and identify its weaknesses in solving PFSSP.
2. To develop a new heuristic method called NACHA to improve job scheduling performance.
3. To evaluate the robustness of NACHA under varying scheduling conditions using benchmark datasets.

This study looks at the Permutation Flow Shop Scheduling Problem (PFSSP) using simple heuristic methods. The problem involves 10 jobs and 5 machines, and all jobs follow the same machine order. Both the NEH and NACHA methods are tested under the same conditions using Microsoft Visual Basic for Excel (VBE). The results are checked by looking at makespan values, machine idle time, and job sequences shown in Gantt charts. This study is important because it tries to improve existing scheduling methods by focusing more on machine workload and bottleneck problems. The NACHA heuristic aims to reduce makespan and increase scheduling efficiency by considering key characteristics like bottlenecks and idle patterns. The results of this study can help manufacturing companies make better scheduling decisions and use their resources more efficiently. It can also be useful for future research on flow shop scheduling problems.

1.3 Literature Review

The Permutation Flow Shop Scheduling Problem or PFSSP is widely studied because it represents many real manufacturing systems where all jobs must follow the same machine order. In these systems small delays can increase the total production time. Many studies use makespan as the main performance measure because it shows how efficiently jobs and machines are scheduled [1]. PFSSP is known to be difficult to solve especially when the number of jobs and machines increases. Because of this researcher often use heuristic methods to find good solutions within a short time [1], [4]. One of the most popular heuristic methods for PFSSP is the Nawaz Enscore Ham or NEH algorithm. NEH is widely used because it is simple and can produce good results in many cases [2]. The method works by sorting jobs based on total processing time and inserting them step by step to minimise makespan. Many researchers use NEH as a baseline method when testing new scheduling approaches [2],[5]. However, studies also show that NEH does not always perform well when datasets have strong machine bottlenecks or uneven processing times [6], [7]. To overcome these weaknesses many researchers have proposed improvements to NEH and other constructive heuristics. Some studies focus on improving job insertion rules and tie breaking strategies to produce better schedules [3], [5], [8]. Other research highlights that machine workload and job interaction play an important role in scheduling performance. When one or two machines become bottlenecks, they can strongly affect the final makespan [6].

These findings show that scheduling performance is influenced not only by total processing time but also by how jobs are distributed across machines [7]. Benchmark datasets are commonly used in PFSSP research to ensure fair comparison between different algorithms [9]. Using multiple datasets helps researchers understand how an algorithm performs under different conditions. Studies show that a heuristic may perform well for some datasets but not for others depending on data structure and variability [7]. Although metaheuristic methods can

provide better solutions, they often require longer computation time. Because of these constructive heuristics are still widely used in practical scheduling problems [4]. Based on these research findings this study proposes the NACHA method as a constructive heuristic that focuses more on bottleneck behaviour and job dominance. Previous studies have shown that bottleneck-focused strategies and improved job-ordering heuristics can enhance scheduling performance and reduce makespan in permutation flow shop scheduling problems under certain conditions [3], [5], [6], [10]. Therefore, this study addresses the existing gap in evaluating the robustness of NACHA for PFSSP by conducting a comparative analysis with the NEH method using benchmark datasets.

1.4 Strengths and Weaknesses of NEH and NACHA

Both NEH and NACHA have their own strengths when used to solve PFSSP. NEH is widely used because it is simple, easy to implement, and able to generate good baseline solutions quickly. However, in more complex datasets, NEH may result in larger machine idle times because it does not directly consider bottleneck behaviour. NACHA was introduced to address this weakness by focusing more on machine bottlenecks and the influence of jobs on critical machines.

Table 1 *Strengths and weaknesses of NEH and NACHA*

Method	Strengths	Weaknesses
NEH	Simple and fast; provides strong baseline performance; widely used for PFSSP	Not sensitive to bottlenecks; may create unnecessary idle time depending on dataset
NACHA	Considers dual bottlenecks; uses dominance-based sequencing; can produce shorter makespan compared to NEH	Improvement depends on dataset; may not outperform NEH for all cases

As a result, NACHA can produce more balanced schedules in certain situations. The main strengths and weaknesses of both methods are summarised in Table 1. Following the characteristics shown above, both NEH and NACHA have advantages depending on the nature of the scheduling dataset. NEH remains a reliable choice for many standard PFSSP instances, while NACHA offers an alternative heuristic suited to machine-sensitive scheduling conditions. Accordingly, it is hypothesized that NACHA achieves superior performance in bottleneck-dominant datasets.

2. Methodology

The methodology for this project follows a constructive heuristic approach, where both NEH and NACHA generate job sequences directly from processing-time data without relying on iterative optimisation. The methodological framework is presented in Fig. 1, which outlines the complete flow of the study beginning from data preparation, algorithm execution, and ending with computation of makespan and idle-time distribution. this methodology section is explained through three sub-sections: data analysis, algorithm development, and implementation procedure.

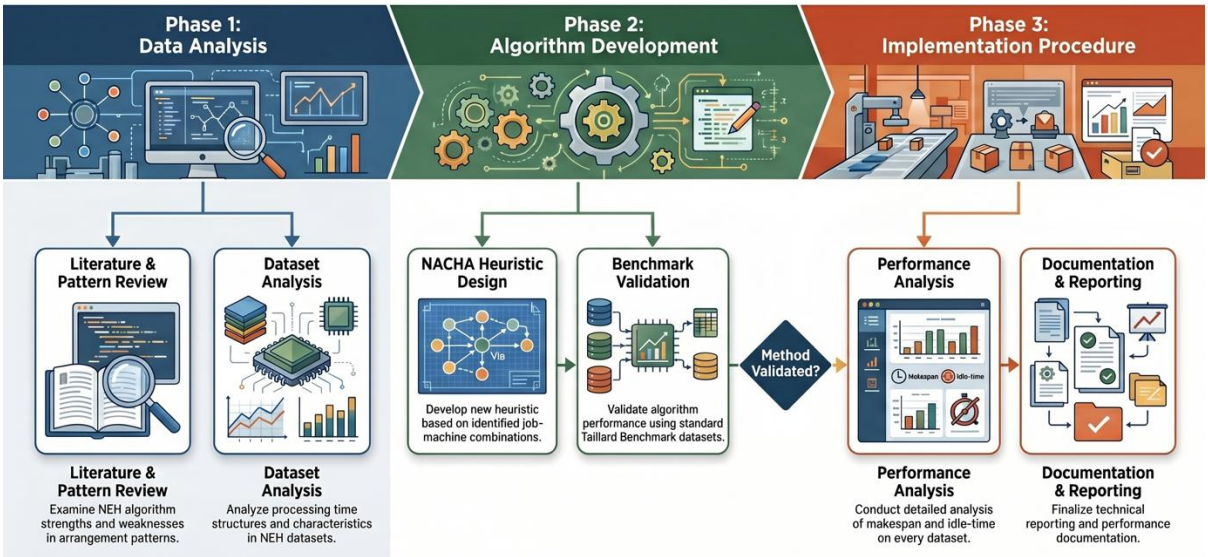

Fig. 1 *Methodological framework: Developing the NACHA Scheduling Heuristic*

2.2 Data Set

This study uses a benchmark dataset that consists of ten jobs and five machines. So, each job must be processed on all machines following the same order. Table 2 shows the processing time values were taken from standard PFSSP datasets that are usually used in scheduling research. These datasets are suitable for evaluating heuristic performance because they represent typical flow shop conditions. All the calculations and simulations in this study were carried out using Microsoft Excel with Visual Basic for Applications (VBA). The VBA program was used to automate job sequencing, calculate completion times and identify machine idle periods for both NEH and NACHA methods. Using Excel-VBA ensures that the same dataset and conditions are applied consistently to both heuristics, allowing a fair comparison between their performances. The use of a small dataset is justified as larger datasets require significantly longer computational time for program execution.

Table 2 *Processing time dataset for 10 jobs and 5 machines*

Job	M1	M2	M3	M4	M5
1	54	79	16	66	58
2	26	59	78	88	69
3	77	39	14	11	83
4	53	93	90	65	64
5	61	27	42	13	55
6	71	13	61	21	91
7	15	28	77	1	45
8	34	5	86	28	8
9	37	59	65	70	94
10	27	79	22	93	38

Table 2 presents the processing time required for 10 jobs (Job 1–Job 10) on five machines (M1–M5). Each row represents a job, while each column shows the processing time of that job on the respective machine. The processing times vary between jobs and machines, ranging from 1 to 94 units. For example, Job 7 has the shortest processing time of 1 unit on M4, while Job 9 has the longest processing time of 94 units on M5. The table is used as the input data for the scheduling analysis in this study.

2.3 Data Analysis

The dataset was first analysed to understand the distribution of processing times across machines and jobs. This step ensures that both NEH and NACHA receive identical and unbiased data inputs. The processing time structure of the dataset directly influences how both heuristics interpret job difficulty and machine load. The consistency of the dataset supports fair comparison between the two algorithms, especially when evaluating makespan and idle time differences. Data analysis also includes verifying the processing time patterns that may lead to bottleneck identification, an important aspect of NACHA's design.

2.4 Algorithm Development

NEH serves as the baseline constructive heuristic for PFSSP. One of the most used heuristics is the Nawaz-Enscore-Ham (NEH) algorithm, which builds job sequences based on processing times. While NEH offers competitive results, it is not always able to produce the optimal solution, especially in more complex scheduling environments [2]. NEH arranges jobs in descending order based on total processing time and inserts them into partial sequences at the position that yields the minimum make span. Its structured simplicity makes it an effective reference method. NACHA was developed to overcome the limitations observed in NEH by introducing bottleneck-aware processing and dominance-based job prioritization. *O*ne such approach is the New Alternate Constructive Heuristic Algorithm (NACHA), which incorporates critical scheduling insights such as dual bottleneck identification and pre-initial arrangement. By strategically managing machine workloads and sequencing jobs before the main process begins, NACHA aims to improve span performance beyond what traditional heuristics like NEH can achieve [3]. NACHA assigns dominance scores to jobs based on their influence on machine congestion, and this scoring mechanism helps identify jobs that should be sequenced earlier or held for later insertion.

3. Result and Discussion

This section shows the results obtained from the experiment conducted using 10 datasets to compare the performance of NEH and NACHA heuristics. The findings are discussed based on makespan performance, performance index (PI), and specific improvement patterns observed in selected datasets. All results were generated using consistent machines and job parameters to ensure fairness in comparison.

3.1 Results

Fig. 2 shows a visual comparison of the makespan (second) values. The chart clearly shows that NEH maintains consistent performance, while NACHA demonstrates both improvements and declines depending on the dataset. Although NACHA does not perform better in all cases, but it shows notable improvements in certain datasets.

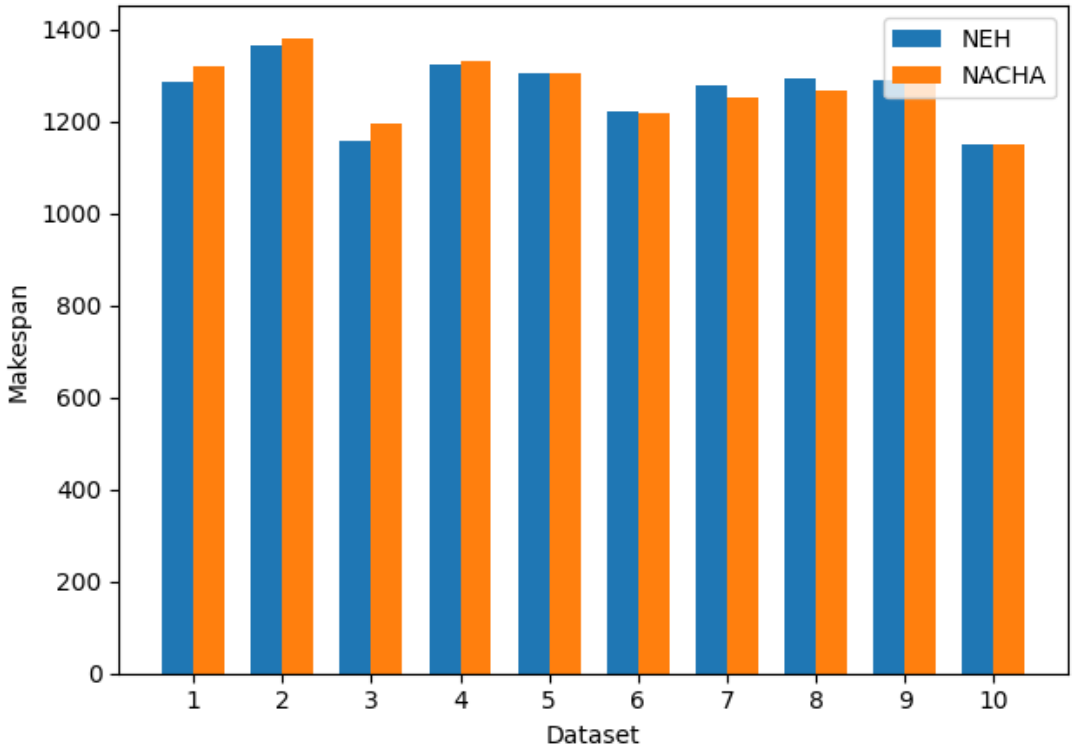

Fig. 2 *Makespan (second) comparison for NEH and NACHA across all datasets*

3.2 Performance Index Analysis

To evaluate the relative performance of NACHA compared to NEH, the performance index (PI) was calculated for each dataset. A positive PI indicates improvement by NACHA, while a negative PI shows that NEH performs better.

Table 3 *Performance Index (PI) Trend*

Data Set	PI (%)	Interpretation
1	-2.80	NEH better
2	-1.24	NEH better
3	-3.10	NEH better
4	-0.53	NEH better
5	0	Equal
6	+5.29	NACHA stronger
7	+2.11	NACHA stronger
8	-3.68	NEH better
9	-1.09	NEH better
10	0	Equal

The Table 3 shows that NACHA is better than NEH only in Dataset 6 and Dataset 7. In these two cases, the PI values are positive, so NACHA gives better results. These datasets have strong machine imbalance, which is why

NACHA works well there. For datasets where the processing times are more balanced, NEH performs better. Dataset 5 and Dataset 10 give the same result because both methods produce the same makespan.

Fig. 3 shows the Performance Index (PI) trend for all datasets when comparing NEH and NACHA. The horizontal line at 0 is equal performance between the two methods. Values above zero indicate that NACHA performs better and values below zero indicate NEH performs better. From the figure, most datasets have PI values below zero which shows that NEH generally gives better results in most cases. However, Dataset 6 and Dataset 7 clearly stand out with positive PI values. This indicates that NACHA performs better than NEH for these 2 datasets. Dataset 6 shows the high positive PI value, so NACHA achieves its best improvement in this case. Dataset 5 and Dataset 10 lie exactly on the zero-line meaning both methods produce the same makespan and perform equally well. Overall, the figure shows that NACHA performs well only in specific situations, especially when machine imbalance is present, while NEH remains more consistent across most datasets.

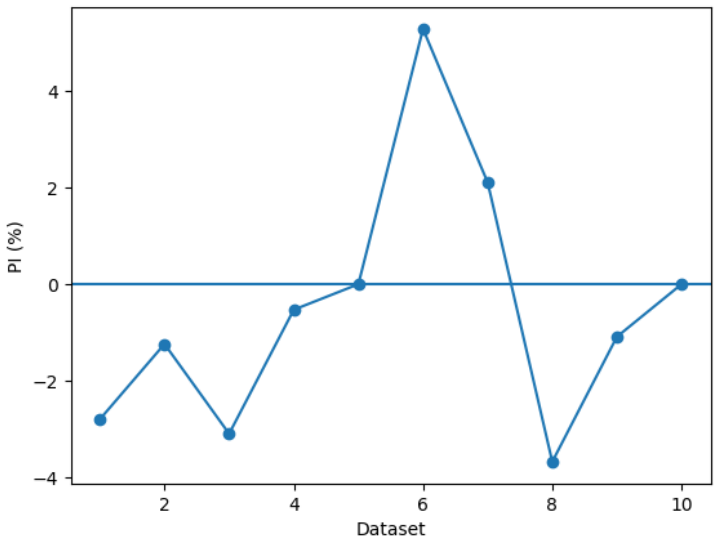

Fig. 3 *Performance Index (PI) comparison between NEH AND NACHA*

Dataset 6 and Dataset 7 are highlighted because they are the only two datasets where NACHA demonstrates clear improvement over the NEH heuristic. These datasets show positive PI values and reduced makespan compared to NEH, indicating that the job–machine structure aligns well with NACHA's dominance-based logic. For Dataset 6**,** NACHA achieves a makespan of 1220**,** outperforming NEH's 1228. Although the improvement is relatively small, it is consistent and meaningful because NEH generally sets a strong performance baseline. The improvement occurs due to better job ordering on bottleneck machines, resulting in reduced waiting time and smoother job flow.

For Dataset 7, NACHA again shows a positive improvement trend with a makespan of 1251, compared to NEH's 1278. The improvement margin here is larger than in Dataset 6, indicating that NACHA's approach is more effective when job processing times create multiple bottlenecks across machines. This allows NACHA to reorder jobs in a way that minimises overall congestion, especially at critical machine stages. Fig. 4 highlights both datasets, showing the points where NACHA outperforms NEH. These results reinforce that NACHA is not universally better but can produce improvements when specific data characteristics favour its prioritization strategy.

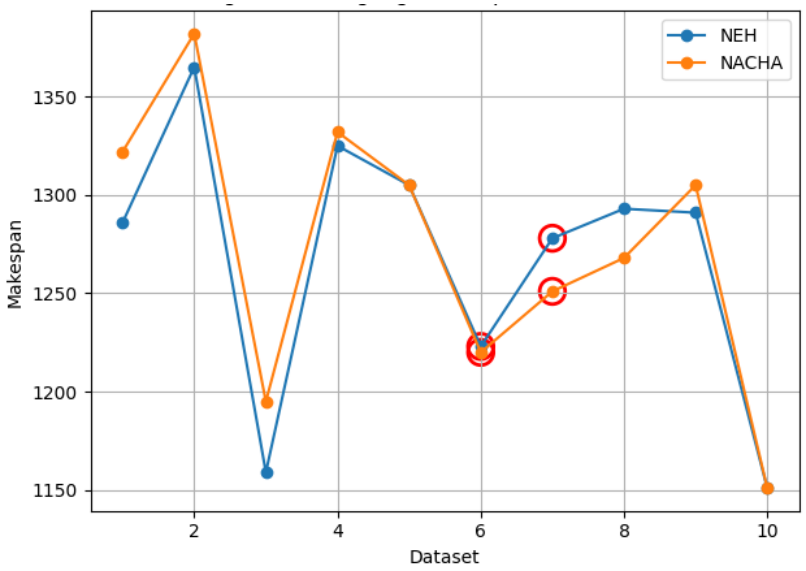

Fig. 4 *Highlighted improvement points for NACHA in Dataset 6 and Dataset 7*

The overall findings indicate that NEH remains one of the most stable heuristics for flow shop scheduling. Across most datasets, NEH consistently produces lower or equal makespan compared to NACHA. However, NACHA demonstrates clear advantages in Dataset 6 and Dataset 7, where the structure of job processing times aligns with its machine dominance scoring.

The improvements observed in these two datasets show that NACHA can reduce makespan under the right conditions. These results suggest that while NACHA is not intended to replace NEH entirely, it serves as a complementary heuristic that can provide better performance in scenarios involving dual bottlenecks or uneven job distributions.

Overall, the analysis supports the conclusion that both heuristics have their strengths. NEH performs reliably across most datasets, while NACHA offers situational improvements that are valuable in selected cases, particularly Dataset 6 and Dataset 7.

4. Conclusion

This study evaluated the performance of two constructive scheduling heuristics NEH and NACHA, using 10 different datasets in a flow shop environment. The analysis showed that NEH continues to be a good and reliable baseline method that producing great makespan values across most datasets. NACHA on the other hand, demonstrated mixed performance with the results varying depending on the structure and distribution of processing times in each dataset.

Although NACHA does not fully outperform the NEH, it achieved clear improvements in Dataset 6 and Dataset 7, where it produced shorter makespan values compared to NEH. These kinds of improvements show that NACHA is effective when the dataset aligns with its dual bottleneck and dominance based sequencing logic. The positive performance index values in these datasets further support the conclusion that NACHA can provide meaningful optimisation under the right scheduling conditions.

Overall, the findings show that NACHA serves as a complementary heuristic rather than a universal replacement for NEH. The improvement patterns observed in specific datasets highlight the potential for further refinement or hybridisation of NACHA to enhance and redefine its stability and performance across a wider range of scheduling scenarios in the world. Future work may explore combining NACHA with adaptive or learning based approaches to strengthen its effectiveness in more complex flow shop settings and situations.

5. Recommendations

Although this study has achieved its main objectives there are still several areas that can be improved in future research. Further studies can focus on testing the proposed method using larger and more complex datasets with a higher number of jobs and machines. This will help to better evaluate its performance in more realistic and challenging scheduling environments. For future research can also compare the proposed approach with other advanced optimization methods such as genetic algorithms or swarm-based techniques to determine its competitiveness. Furthermore, the algorithm can be implemented using different programming platforms to improve computation speed and ease of use. These kinds of improvements may help increase the practicality of the proposed method for real industrial scheduling applications.

Acknowledgement

The authors would like to thank the Centre for Diploma Studies, Universiti Tun Hussein Onn Malaysia for its financial support and facilitation in this project.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Vishal Vengadesan, Noor Azizah Sidek, Azli Nawawi;* ***data collection:*** *Vishal Vengadesan, Noor Azizah Sidek, Azli Nawawi;* ***analysis and interpretation of results:*** *Vishal Vengadesan, Noor Azizah Sidek, Azli Nawawi;* ***draft manuscript preparation:*** *Vishal Vengadesan, Noor Azizah Sidek, Azli Nawawi. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] D. A. Rossit, F. Tohmé, and M. Frutos, "*The non-permutation flow-shop scheduling problem: A literature review*," Omega, vol. 77, pp. 143–153, Jun. 2018, doi: 10.1016/j.omega.2017.05.010.

[2] M. Nawaz, E. E. Enscore, and I. Ham, "*A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*," Omega, vol. 11, no. 1, pp. 91–95, Jan. 1983, doi: 10.1016/0305-0483(83)90088-9.

[3] W. Liu, Y. Jin, and M. Price, "*A new improved NEH heuristic for permutation flowshop scheduling problems*," Int. J. Prod. Econ., vol. 193, pp. 21–30, Nov. 2017, doi: 10.1016/j.ijpe.2017.06.026.

[4] E. G. Talbi, Metaheuristics: From Design to Implementation. Hoboken, NJ, USA: John Wiley & Sons, 2009.

[5] P. J. Kalczynski and J. Kamburowski, "*Tie-breaking strategies for the NEH heuristic*," Eur. J. Oper. Res., vol. 185, no. 2, pp. 698–711, 2008.

[6] N. A. Isa, S. A. Bareduan, and A. S. Zainudin, "*Bottleneck-based heuristic for permutation flowshop scheduling*," in IOP Conf. Ser.: Mater. Sci. Eng., vol. 824, no. 1, Jun. 2020, Art. no. 012020, doi: 10.1088/1757-899X/824/1/012020.

[7] M. M. Yenisey and B. Yagmahan, *"Evaluation of scheduling algorithms for flow shop problems*," Int. J. Prod. Econ., vol. 155, pp. 17–30, 2014.

[8] C. Pacini, A. Subramanian, and R. M. Silva, "*Improved NEH-based heuristics for flow shop scheduling problems*," Algorithms, vol. 13, no. 6, Art. no. 112, 2020.

[9] E. Taillard, "*Benchmarks for basic scheduling problems*," Eur. J. Oper. Res., vol. 64, no. 2, pp. 278–285, Jan. 1993, doi: 10.1016/0377-2217(93)90182-M.

[10] R. Puka, I. Skalna, J. Duda, and A. Stawowy, "*Deterministic constructive vN-NEH+ algorithm to solve permutation flow shop scheduling problem with makespan criterion*," Comput. Oper. Res., vol. 162, Art. no. 106473, 2024, doi: 10.1016/j.cor.2023.106473.