MULTIPLE BOTTLENECKS SORTING CRITERION AT INITIAL SEQUENCE IN SOLVING PERMUTATION FLOW SHOP SCHEDULING PROBLEM

N.A. Isa[1], S.A. Bareduan[1], A.S. Zainudin[2] and M. Marsudi[3]

[1]Faculty of Mechanical and Manufacturing Engineering,
Universiti Tun Hussein Onn Malaysia,
86400 Batu Pahat, Johor, Malaysia.

[2]Faculty of Engineering,
Universiti Teknologi PETRONAS,
32610 Bandar Seri Iskandar, Perak, Malaysia.

[3]Faculty of Engineering,
Islamic University of Kalimantan,
70123 South Kalimantan, Indonesia.

Corresponding Author's Email: [1]saleh@uthm.edu.my

Article History: Received 13 March 2020; Revised 9 September 2020;
Accepted 2 February 2021

ABSTRACT: This paper proposes a heuristic that introduces the application of bottleneck-based concept at the beginning of an initial sequence determination with the objective of makespan minimization. Earlier studies found that the scheduling activity become complicated when dealing with machine, m greater than 2, known as non-deterministic polynomial-time hardness (NP-hard). To date, the Nawaz-Enscore-Ham (NEH) algorithm is still recognized as the best heuristic in solving makespan problem in scheduling environment. Thus, this study treated the NEH heuristic as the highest ranking and most suitable heuristic for evaluation purpose since it is the best performing heuristic in makespan minimization. This study used the bottleneck-based approach to identify the critical processing machine which led to high completion time. In this study, an experiment involving machines (m =4) and n-job (n = 6, 10, 15, 20) was simulated in Microsoft Excel Simple Programming to solve the permutation flowshop scheduling problem. The overall computational results demonstrated that the bottleneck machine M4 performed the best in minimizing the makespan for all data set of problems.

KEYWORDS: *Bottleneck-Based; Flow Shop; Scheduling; Makespan; Heuristic; Algorithm*

1.0 INTRODUCTION

Scheduling is a planning activity with certain objectives within the time constraints. Scheduling is an important factor in manufacturing industry especially for production planning. Efficient scheduling helps to increase the production efficiency, utilization and also profitability. Many researchers have been focusing on the study of flow shop scheduling in which each job is processed by a series of machine with the same sequence, even though the processing time may be different [3-7, 13-21]. One of the assumptions or limitations identified for developing the intended flow shop scheduling is no machine can process more than one job at a time. It means that each machine can only process and finish one job at a time before continuing with other jobs. Other assumptions include pre-emption is not allowed, which means that the job must continuously be processed without any interruption. All setup time is included into the job processing time where the machine setup time to remove tools, jigs and fixtures is included into the job processing times. There is unlimited storage between the machines. There is no blocking job (job remains at machine after finishing) to avoid interruptions of other jobs, thus there is available space for finished job between the machines. All machines are continuously available which mean that there is no machine breakdown that will affect the total completion time.

Researchers found that scheduling activities become more complicated when the production scheduling system is involved with machines, $m > 2$. It is becoming an NP-hard problem. NP-hard is informally said as "at least as hard as the hardest problems in NP" in a group of problems, defined in computational complexity theory [1]. Herrmann and Lee [23] studied three objectives of the strong NP hardness which are makespan, number of tardy jobs and total completion, and the continuation is done by Lin and Hwang [1] in minimizing the total completion time using dynamic programming algorithm with designed matrices. Implicit enumeration technique searches were also developed, however it was known to have large constraint when dealing with large problem size where it consumed very long time [4]. This is the answer to the questions on why it is difficult to produce a new heuristic which is capable to produce near optimal solution in solving large sizes permutation flow shop scheduling problem.

Nawaz et al. [14] introduced the NEH heuristic more than three decades ago where it was declared as the best performing method in solving makespan criterion. Since then, the permutation flow shop scheduling problem has become an interesting and famous topic

among researchers. Most of them modify or extend the NEH heuristic procedure in order to improve the scheduling solutions. Among the researchers are [2-7, 16, 18]. Framinan et al. [7] recommended further studies to be conducted to the NEH by employing different indicator values at the first step and choosing different sorting criterion at the second step. Abedinnia et al. [18] suggested additional option of extending the NEH by performing a local search of partial sequence at each iteration, applying different tie-breaking at second step, and choosing different decision criterion for selecting the best k-job partial sequence.

The bottleneck phenomenon often occurs in production scheduling systems [11]. In ensuring the feasibility and effectiveness of scheduling result, it is important to identify the bottleneck resources so that the jobs can be rationally scheduled. It also helps in reducing the difficulty of follow-up scheduling [10]. Shifting bottleneck model has been widely used in sorting the machine with the highest increase of objective function value [9]. Basically, the main idea is to schedule the jobs at the bottleneck stage where it may affect the performance of a heuristic scheduling jobs at all stages [11]. Bottleneck machines are detected based on the machine workload, utilization rate or idle time length, and bottleneck involvement significantly affects the quality of the final solution [12]. Based on the study by Zhang and Wu [12], the local search effort for the bottleneck machines has generated higher quality of solution result at reasonable short CPU time. The study of shifting bottleneck performance is also supported by the work by Mönch and Zimmermann [24], where they investigated stochastic settings in a semiconductor manufacturing environment. Most bottleneck studies combined with other simple priority rule, genetic algorithm, neighbourhood search algorithm, iterated local search, dispatch rules and many more stabilize the heuristic, and solve their objective function problem [9, 13, 25-28].

Our recent study found an interesting modification to the NEH that can boost scheduling performance with the objective of minimizing the makespan. This paper proposes a new heuristic for permutation flow shop scheduling using bottleneck-based concept consisting of three important implementation phases in minimizing the makespan.

Further designs of this paper are structured as follows. Section 2.0 describes the heuristic of NEH and possible modification for its extension. In Section 3.0, the methodology on the techniques and procedure of the proposed heuristic are highlighted in detail. A comprehensive comparison of the proposed heuristic and NEH,

together with a detailed evaluation of the effect of the proposed modifications on NEH, are given in Section 4.0, and finally Section 5.0 concludes the paper.

2.0 LITERATURE REVIEW

Permutation flow shop scheduling problem (PFSP) is one of the best known production scheduling problems with the same job order on all machines which have an intense engineering background [18]. Makespan minimization is one of the most attractive objective where it leads to the development of many approximate algorithms [14-16, 22]. Nawaz, Enscore and Ham's (NEH) heuristics which appeared more than three decades ago are known as the best performing method in minimizing the makespan for permutation flow shop scheduling problem [14]. The NEH heuristic procedure consists of two phases which are; (a) Sorting phase/ prioritizing phase and (b) Insertion phase.

Sorting phase is the phase where the jobs are sorted in descending order of their total processing time, and the sorted list is used in the insertion phase to determine the sequence in which jobs are added to the existing partial sequences. This algorithm gives the highest attention on the job with the largest total processing time where it should have been the priority to be processed first. For n-job PFSP, the insertion phase consists of n iterations. The k-th job is successively assigned to the k possible slots in the current partial sequence obtained from previous iteration consisting of k – 1 jobs. The partial sequence with the lowest objective function is used as current k-job partial sequence for the next iteration.

Extensions of NEH heuristic have been reported by many researchers with multiple purposes of study and with different objectives [5, 8, 12, 15-17, 19, 21]. The latest study by Viagas and Framinan [17] considered the total tardiness in a permutation flow shop scheduling problem. The analysis showed that the improved NEH, that is NEHedd heuristic applies the sorting phase by arranging the jobs according to the Earliest Due Date (EDD). Then the partial sequence with the lowest total tardiness is selected. Several tie breaking methods were proposed, and extensive computational experiment was conducted to handle the ties based on machine idle time as well as the Taillard's acceleration. This Taillard's acceleration mechanism is used in order to reduce the complexity of the NEH heuristic [20].

3.0 METHODOLOGY

The bottleneck-based (BNB) heuristic firstly identifies the bottleneck stage, and then determines the schedule of the jobs in the bottleneck stage [11]. The job processing time is used as an indicator to determine the bottleneck stage. The heuristic proposed here uses three important phases to execute this BNB heuristic. The phases must be followed in sequence as follows:

i. Bottleneck Identification Phase
ii. Initial Sequence Arrangement Phase
iii. Job Insertion Phase

3.1 Bottleneck Identification Phase

Based on the four machines' problem, there are four possible bottleneck machines and each of them has its own initial sequence. The initial sequence for each of the bottleneck machine was arranged based on the result of bottleneck machine identification. Bottleneck identification is important to classify the criticality of process machines. The machine criticality or bottleneck machine is classified based on the processing time of job being processed in each machine. All steps to identify the machine bottleneck are explained below:

Step 1: The average processing time for all jobs on all machines was calculated.

Step 2: The machine dominance was then determined for each processing time.

$$X = \begin{cases} 1, & \text{if } t > t_{avg} \\ 0, & \text{if } t \leq t_{avg} \end{cases} \quad (1)$$

where; X= Dominance value, t = Processing Time and t_{avg} = Average Processing Time.

Step 3: The total dominance value was calculated for each machine. The machine with the highest dominance value was identified as the bottleneck machine.

3.2 Initial Sequence Phase

Framinan et al. [7] found that initial job arrangement and the opportunity of inserting a job in any possible position are the strengths of NEH heuristic. These are the reasons why the NEH are known as the best heuristic in solving the permutation flowshop problem with the

objective of makespan minimization. In the proposed BNB heristic, the initial sequence was arranged based on the result of machine dominant identification. The cases were studied using the scheduling Gantt charts that manage to obtain the best solution. The initial sequence arrangements depended on the number of total jobs processed as follows:

i. Six-jobs and ten-jobs
   a. Machine 1 was the bottleneck machine (BM1), thus the processing time of jobs for machines 1 and 2 was summed up (M1+M2).
   b. Machine 2 was the bottleneck machine (BM2), thus the processing time of jobs for machines 1 and 2 was summed up (M1+M2).
   c. Machine 3 was the bottleneck machine (BM3), thus the processing time of jobs for machines 3 and 4 was summed up (M3+M4).
   d. Machine 4 was the bottleneck machine (BM4), thus the processing time of jobs for machines 4 was used (M4).

ii. Fifth teen-jobs and twenty-jobs
   a. Machine 1 was the bottleneck machine (BM1), thus the processing time of jobs for machines 1 and 2 was summed up (M1+M2).
   b. Machine 2 was the bottleneck machine (BM2), thus the processing time of jobs for machines 1 and 2 was summed up (M1+M2).
   c. Machine 3 was the bottleneck machine (BM3), thus the processing time of jobs for machines 1, 2 and 3 was summed up (M1+M2+M3).
   d. Machine 4 was the bottleneck machine (BM4), the processing time of jobs for machines 1, 2 and 3 was summed up (M1+M2+M3).

The sequence guide selects a job with the highest processing time to be processed first, so that there is not much idle time towards the end of the overall sequence. This is followed with the job with second highest processing time, and continued with the next job until the last job with the lowest processing time. The summary of machine processing time used to determine the initial sequence arrangement for each of the bottleneck machine is shown in Table 1.

Table 1: Processing time used for initial sequence arrangement

Job	Sum Up of Machine Processing Time			
	Bottleneck Machine BM1	Bottleneck Machine BM2	Bottleneck Machine BM3	Bottleneck Machine BM4
6	M1+ M2	M1+ M2	M3+M4	M4
10	M1+ M2	M1+ M2	M3+M4	M4
15	M1+ M2	M1+ M2	M1+M2+M3	M1+M2+M3
20	M1+ M2	M1+ M2	M1+M2+M3	M1+M2+M3

3.3 Insertion Phase

Next, for the kth job, k = 3,.., n, insert the job into the place, among the positions 1, 2,…, k job of partial sequence, while keeping the relative sequence of partial sequence. Choose the best sequence out of k sequence as partial sequence for the next iteration. The example of generalization sequence of this heuristic can be described as follows:

Suppose that a standard permutation flowshop problem with m = 4 machines, n = 10 jobs, processing time Pmn and the objective function is makespan minimization, Cmax.

Step 1: Compute the sum of processing time, and calculate the average processing time of 10-job for 4-machine.

Step 2: Calculate the total dominance value and determine the bottleneck machine using equation in Step 2 at bottleneck identification phase. The machine having the highest total dominance value are called the bottleneck machine.

Step 3: Sort the jobs according to descending sums of processing time depending on the machine bottleneck cases as in Table 1.

Step 4: The initial sequence arrangement was made based on the sorted list from Step 3.

Step 5: Take the first two jobs from the job arrangement in Step 4, and schedule their start-stop time. Reverse the position of jobs. Determine the first partial sequence with the lowest objective function, Cmax.

Step 6: For the kth job, k = 3,…, 10, insert the job into place, among the positions 1, 2,…, k job of partial sequence, while keeping the relative partial sequence. Choose the best Cmax out of k sequence as partial sequence for the next iteration until the schedule is completed.

4.0 RESULTS AND DISCUSSION

This section examines the performance of the heuristic proposed in Section 3. As mentioned before, Nawaz et al. [14] showed that using the

jobs sorted in descending order leads to better results. Therefore, we compare the results of the proposed heuristic in this paper with the original NEH heuristic. The performance of both heuristic was evaluated respectively to the effectiveness (the quality and goodness solution obtained) and efficiency in different job sizes.

In our computational experiment, BNB and NEH heuristic were coded in Excel VBA and run on 1.8GHz Intel Core i3 processor with 4.00 GB RAM. We considered the small size problems with a number of jobs equal to 6 and 10, and large size problems with a number of jobs equal to 15 and 20. The machines were equal to 4 and the potential bottleneck machines were also equal to 4. A hundred instances were generated for each combination of jobs and machines. The processing time was generated randomly using a U(1,40) distribution. In addition, to verify the heuristic performance, ten replications of 100 set of random data were done for each job stage (n = 6, 10, 15, 20). The purpose of this process was to verify the capability of the proposed heuristic to produce an optimum result at various processing time.

4.1 Comparing BNB and NEH for Optimum Solution

This section compares and critically discusses the performance of BNB and NEH. As mentioned before, both algorithms have almost the same complexity. Table 2 summarizes the results of BNB and NEH performance against the optimum solution for four-machine and six-job problem. The average makespan ratio represents the average ratio of makespan from BNB and NEH heuristic to the optimum makespan from the complete enumeration, and are classified into four bottleneck machines. The optimum result column represents the percentage of both BNB and NEH makespan equals to optimum makespan. Makespan ratio is defined as heuristic makespan/optimum makespan. The percentage of optimum result is defined as the percentage of heuristic makespan ratio equals to the optimum makespan among the available data set.

According to Table 2, the results indicated that the BNB heuristic produced overall makespan ratio that was 0.90% above the optimum with overall average makespan ratio of 1.0090. Lower makespan ratio indicated less error of makespan when compared to the optimum answer. NEH heuristic produced overall makespan solution that was 1.17% above the optimum with the overall average makespan ratio of 1.0117. Moreover, bottleneck machines M1, M2, and M4 achieved lower error compared to the NEH solution. BNB led to the optimal solution in 67.99% of the cases. NEH showed lower performance than the BNB

by obtaining the optimal solution of 58.15%. The higher percentage of optimum result indicated high number of optimum solution obtained.

Table 2: Comparing BNB and NEH for optimum solution

Bottleneck Machine	Average Makespan Ratio		Optimum Result (%)	
	BNB	NEH	BNB	NEH
M1	1.0098	1.0159	63.86	48.57
M2	1.0124	1.0145	57.89	57.89
M3	1.0105	1.0085	77.78	62.96
M4	1.0032	1.0079	68.42	63.16
Overall	1.0090	1.0117	67.99	58.15

Figure 1 shows the dotplot graph of heuristics performance for six-job problem. The graph shows the trend of BNB and NEH behaviour against the optimum result. It can be clearly seen that the frequency at value 1.00 for BNB makespan was higher compared to the NEH makespan. Thus, it indicated that the number of optimum solutions obtained by the BNB were higher than the NEH solution, since each symbol represented up to 3 observations. From the dotplot graph in Figure 1, it can be seen that one dot of NEH makespan ratio almost achieved 1.080, and the NEH distributions of makespan ratio were a bit distant from the optimum solution ratio of 1.0. While, the BNB distributions of makespan ratio were closer to the optimum solution where the highest value is recorded at lower than 1.068. Thus, it showed that the proposed BNB algorithm outperformed the NEH for four-machine and six-job problem.

4.2 Evaluating the BNB/NEH Performance over the Problem Sizes

This section analyses and discusses the performance of the proposed BNB algorithm with NEH for variety of job size problems. Table 3 shows the overall performance of the BNB heuristic performance over job sizes, classified into each bottleneck machine. Makespan ratio is defined as BNB makespan/ NEH makespan.

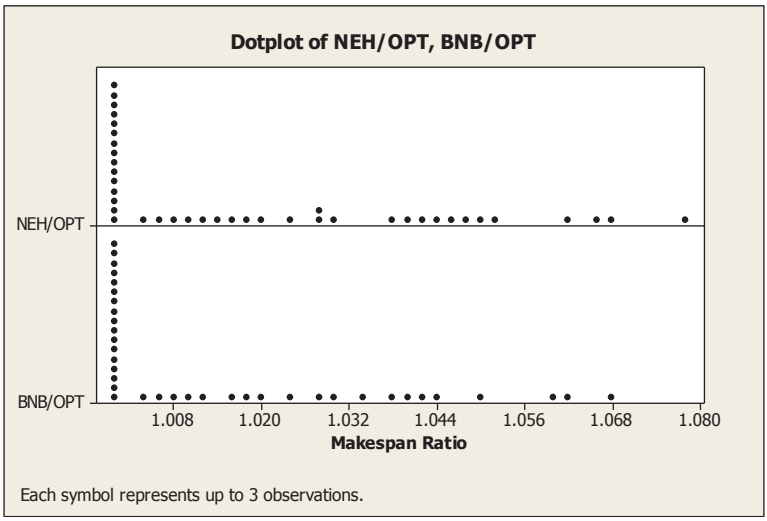

Figure 1: Dotplot graph of heuristics performance for six-jobs

From Table 3, the results show a good performance for each bottleneck machine on all jobs. The average makespan ratio showed values of less than 1.00, or very close to 1.00. Bottleneck machines M1, M3, and M4 produced overall makespan better than the NEH makespan. Bottleneck machine M2 produced overall makespan equal with the NEH makespan. Bottleneck machines M1, M3 and M4 produced makespan solutions better than the NEH makespan solutions at least on three job sizes, respectively, while bottleneck machine M2 produced two makespan solutions better than the NEH that were for six-job and twenty-job problem sizes. The results indicated that the solutions obtained by M1, M3 and M4 were better than NEH due to suitable arrangements of initial sequence with the set of processing time used during the test. The initial sequence used was good in producing the makespan near to the optimal results.

Figure 2 shows the dotplot graph of BNB makespan ratio for all job problem sizes. The graph shows that the patterns of performance for all job sizes were almost the same. It can be seen that the number of dot at value 1.00 was decreasing with the increase of job problem size. For six-job problem, the graph showed higher frequency of makespan ratio of 1.00 since the column was higher at value 1.00. There were more dots located at the makespan ratio below 1.00. This means that more BNB solutions were better than the NEH solutions. This can also be seen in the problem size on 10-job and 20-job.

However, the frequency difference between makespan below and higher than 1.00 was very small. For 15-job problem, slightly more dots

were located at the makespan ratio higher than 1.00. This means that slightly more BNB solutions were worse than the NEH solution.

Table 3: BNB/NEH performance over job sizes

Problem Sizes(Job)	Average Makespan Ratio			
	M1 Bottleneck	M2 Bottleneck	M3 Bottleneck	M4 Bottleneck
6	0.9942	0.9980	0.9981	0.9955
10	0.9955	1.0010	1.0038	0.9973
15	0.9969	1.0018	0.9995	1.0013
20	1.0038	0.9994	0.9972	0.9988
Overall	0.9976	1.0000	0.9997	0.9982

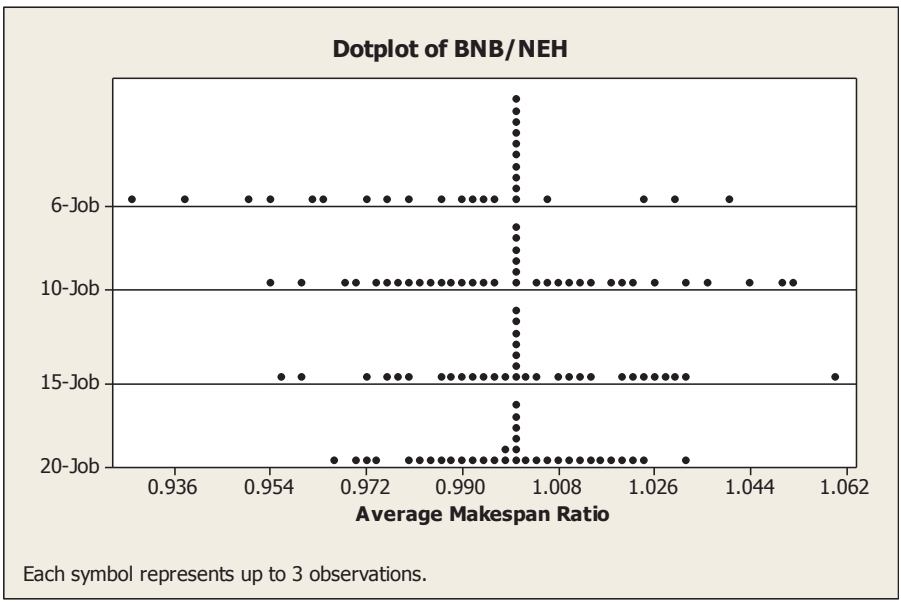

Figure 2: Dotplot graph for overall performance of BNB average makespan ratio for all job sizes

4.3 Verification of BNB /NEH Performance on Ten Replication

This section evaluates and discusses the results of BNB/NEH performance on ten replications for each job size. The purpose of this replication was to ensure the effectiveness of proposed heuristic over a variety of processing time sets. Technically, each job size was tested with 1000 sets of generated random data of processing time. Table 4 shows the result on overall performance of BNB/NEH for all job sizes.

Based on Table 4, the results show that bottleneck machine M4 performs the best in overall verification test with 0.03% below the NEH makespan result. The other three bottleneck machines were less performing, since they produced a little higher makespan solution over

the NEH result. This happened because the arrangement of initial sequence was less suitable with the variety set of processing time used during the verification test. Sometimes, the results are good at small sized problems only, and sometimes are good at both, small and large sized problems. This was indicated by the makespan ratio results which fluctuated over the bottleneck machines and job sizes.

Table 4: BNB/NEH performance on ten replication

Problem Sizes(Job)	Average Makespan Ratio for Ten Replication			
	M1 Bottleneck	M2 Bottleneck	M3 Bottleneck	M4 Bottleneck
6	0.9999	1.0028	1.0017	0.9979
10	1.0018	1.0017	1.0053	1.0009
15	1.0001	1.0019	1.0003	1.0003
20	1.0022	1.0014	1.0009	0.9997
Overall	1.0010	1.0020	1.0021	0.9997

Figure 3 shows the bar graph for BNB/NEH performance on ten replications. The graph obtained was based on the result shown in Table 3. The graph shows that the patterns of performance fluctuated. Some of the BNB answer was good enough to compete with NEH answer. However, on certain job sizes, the effectiveness of the proposed heuristic was not strong enough to produce answer, which was better than NEH heuristic. Among the result analysis for 1000 sets of random data outcomes, only the overall result performance of bottleneck M4 succeeded in producing the answer better than the NEH.

Comparing the results of the first computational experiment using 100 sets of data against the second experiment using 1000 sets of other random data, the findings can be critically analyzed as the followings. In the first experiment for 100 sets of data, the observation suggested that the developed BNB heuristic has the capability to produce better results than the NEH specifically for M1, M3 and M4 bottlenecks as shown in Table 3. This indicated that within these three bottleneck groups, the selected initial sequence arrangement and the inserting methods used was effective in producing good results. This indeed agree with the report by Framinan et al. [7]. These preliminary findings were then further evaluated using computational experiment involving 10 times of the previous data quantity which total to 1000 sets of random data. The results showed that the BNB heuristic only consistently produced better overall result for the M4 bottleneck group.

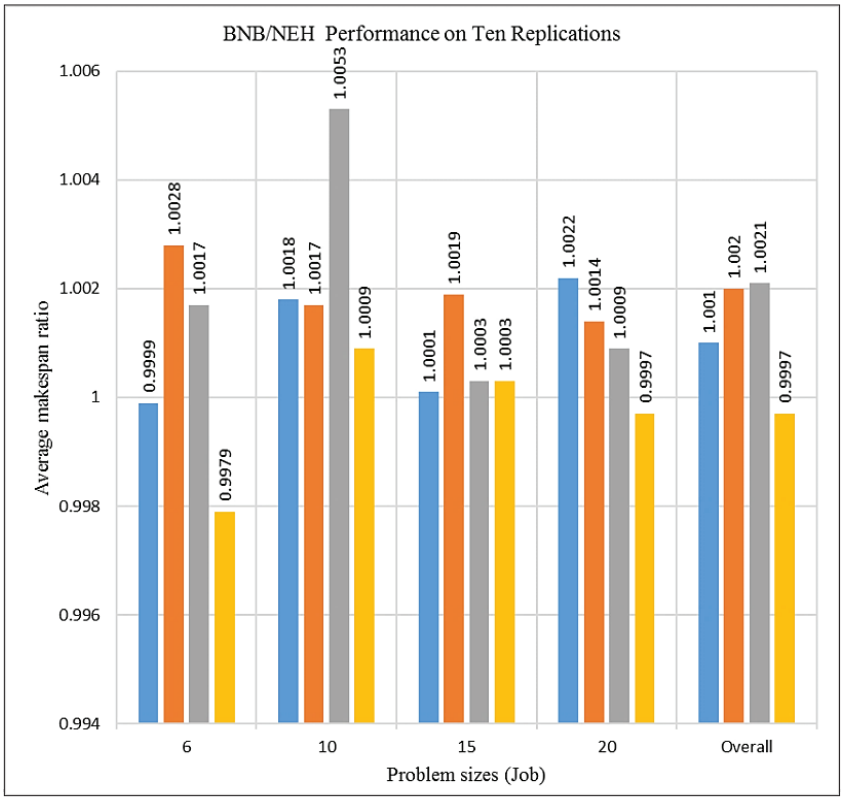

Figure 3: Bar graph for BNB/NEH performance on ten replication

This is shown in Table 4. However, the BNB still manage to produce better result at the M1 bottleneck group specifically for small problem size involving six job problem. In other bottleneck groups, the NEH is still the best performing heuristic. Nevertheless, the findings in this experiment showed that the BNB initial sequence arrangement approach has the potential to produce better result than the NEH at some specific bottleneck groups and problem sizes. The BNB potential has to be further polished by conducting detail in-depth investigation in the area of bottleneck classification, initial sequence arrangement and insertion methods.

5.0 CONCLUSION

The best known simple constructive heuristic algorithm for optimizing the permutation flowshop scheduling problem with the objective of makespan minimization was modified in this paper by employing different sorting criteria at initial sequence. A computational experiment was conducted to examine the effect of the modification on

the makespan result. Our statistical result analyses demonstrated that the modification proposed and evaluated in this paper leads to the better performance from the NEH heuristic only at specific bottleneck groups which are at last machine or M4. In modifying the NEH algorithm to improve its performance, we introduce a new idea for classifying the bottleneck machine based on machine dominance value. This was followed by selecting the initial solution based on combination of specific machines processing time and the bottleneck groups. The numerical studies showed that using alternative sorting methods has potential to improve the algorithm performance in some specific bottleneck groups and problem size. Another contribution of the paper is the utilization of newly-defined decision criterion at initial sequence before selecting the first two jobs in earlier iteration of the NEH heuristic. This numerical studies showed that using the bottleneck grouping criteria has the potential to lead better results. The idea of using bottleneck identification phase shows that the method used can improve the NEH heuristic result. None of the previous studies has discovered this concept, where the improvement can be made with the combination approach of machine dominance and sorting criteria based on bottleneck groups.

ACKNOWLEDGMENTS

This research is funded by Ministry of Education of Malaysia (MOE) and Universiti Tun Hussein Onn Malaysia (UTHM) under Fundamental Research Grant Scheme (FRGS, Vot K088).

REFERENCES

[1] B.M.T. Lin and F.J. Hwang, "Total completion time minimization in a 2-stage differentiation flowshop with fixed sequences per job type", *Information Processing Letters*, vol. 111, no. 5, pp. 208–212, 2011.

[2] H.S. Woo and D.S. Yim, "A heuristic algorithm for mean flowtime objective in flowshop scheduling", *Computers & Operations Research*, vol. 25, no. 3, pp. 175–182, 1998.

[3] P.J. Kaczynski and J. Kamburowski, "On the NEH heuristic for minimizing the makespan in permutation flow shops", *Computers & Operations Research*, vol. 35, no. 9, pp. 3001–3008, 2008.

[4] A. Allahverdi and T. Aldowaisan, "New heuristics to minimize total completion time in m-machine flowshops", *International Journal Production Economics*, vol. 77, no. 1, pp. 71–83, 2002.

[5] X. Dong, H. Huang and P. Chen, "An improved NEH-based heuristic for the permutation flowshop problem", *Computers & Operations Research*, vol. 35, no. 12, pp. 3962–3968, 2008.

[6] N.W. Saleh, "Modified NEH Heurictic Scheduling by Using Weighted Idle Time," Bachelor thesis, Faculty of Mechanical Engineering, Universiti Tun Hussein Onn Malaysia, Johor, 2014.

[7] J. M. Framinan, R. Leisten and C. Rajendran, "Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem", *International Journal of Production Research*, vol. 41, no. 1, pp. 121–148, 2003.

[8] A. Allahverdi and F.S. Al-Anzi, "A branch-and-bound algorithm for three-machine flowshop scheduling problem to minimize total completion time with separate setup times", *European Journal of Operational Research*, vol. 169, no. 3, pp. 767–780, 2006.

[9] R. Braune and G. Zapfel, "Shifting bottleneck scheduling for total weighted tardiness minimization-A computational evaluation of subproblem and re-optimization heuristics", *Computers & Operations Research*, vol. 66, pp. 130–140, 2016.

[10] B. Zhenqiang, W. Weiye, W. Peng and Q. Pan, "Research on production scheduling system with bottleneck based on multi-agent", *Physics Procedia*, vol. 24, pp. 1903–1909, 2012.

[11] C.L. Chen and C.L. Chen, "A bottleneck-based heuristic for minimizing makespan in a flexible flow line with unrelated parallel machines", *Computers & Operations Research*, vol. 36, no. 11, pp. 3073–3081, 2009.

[12] R. Zhang and C. Wu, "Bottleneck machine identification method based on constraint transformation for job shop scheduling with genetic algorithm", *Information Sciences*, vol. 188, pp. 236–252, 2012.

[13] R. Braune, G. Zäpfel and M. Affenzeller, "An exact approach for single machine sub-problems in shifting bottleneck procedures for job shops with total weighted tardiness objective", *European Journal of Operational Research*, vol. 218, no. 1, pp. 76–85, 2012.

[14] 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, 1983.

[15] T. Aldowaisan and A. Allahverdi, "New heuristics for no-wait flowshops to minimize makespan", *Computers & Operations Research*, vol. 30, no. 8, pp. 1219–1231, 2003.

[16] P.J. Kalczynski and J. Kamburowski, "A heuristic for minimizing the makespan in no-idle permutation flow shops", *Computers & Industrial Engineering*, vol. 49, no. 1, pp. 146–154, 2005.

[17] V.F. Viagas and J.M. Framinan, "NEH-based heuristic for the permutation flowshop scheduling problem to minimise total tardiness", *Computers & Operations Research*, vol. 60, pp. 27–36, 2015.

[18] H. Abedinnia, C.H. Glock and A. Brill, "New simple constructive heuristic algorithms for minimizing total flow-time in the permutation flowshop scheduling problem", *Computers & Operations Research*, vol. 74, pp. 165–174, 2016.

[19] J.M. Framinan, and R. Leisten, "A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness", *International Journal of Production Economics*, vol. 99, no. 1–2, pp. 28–40, 2006.

[20] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem", *Europe Journal of Operation Research*, vol. 47, no. 1, pp. 65–74, 1990.

[21] I. Wegener, *Complexity Theory; Exploring the Limits of Efficient Algorithms*. New York: Springer Verlag, 2005.

[22] B.C. Choi, J.Y.T. Leung and M.L. Pinedo, "A note on makespan minimization in proportionate flow shops", *Information Processing Letters*, vol. 111, no. 2, pp. 77–81, 2010.

[23] W. Herrmann and C.Y. Lee, "Three-machine look-ahead scheduling problems," Department of Industrial Engineering, University of Florida, Gainesville, Research Report No. 92-93, 1992.

[24] L. Mönch and J. Zimmermann, "A computational study of a shifting bottleneck heuristic for multi-product complex job shops", *Production Planning & Control*, vol. 22, no. 1, pp. 25–40, 2011.

[25] S.J. Mason, J.W. Fowler and W.M. Carlyle, "A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops", *Journal of Scheduling*, vol. 5, no. 3, pp. 247–262, 2002.

[26] L. Mönch, R. Schabacker, D. Pabst and J.W. Fowler, "Genetic algorithm-based sub-problem solution procedures for a modified shifting bottleneck heuristic for complex job shops", *Europe Journal of Operation Research*, vol. 177, no. 3, pp. 2100–2118, 2007.

[27] B. Scholz-Reiter, T. Hildebrandt and Y. Tan, "Effective and efficient scheduling of dynamic job shops combining the shifting bottleneck procedure with variable neighborhood search", *CIRP Annals*, vol. 62, no. 1, pp. 423–426, 2013.

[28] A. Bilyk and L. Mönch, "Variable neighborhood search-based sub-problem solution procedures for a parallel shifting bottleneck heuristic for complex job shops", in IEEE International Conference on Automation Science and Engineering, Seoul, South Korea, 2012, pp. 419–424.