Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

JOURNAL OF INFORMATION SYSTEM AND TECHNOLOGY MANAGEMENT (JISTM)
www.jistm.com

OPTIMIZATION OF NEW CONSTRUCTIVE HEURISTIC ALGORITHMS FOR PERMUTATION FLOW SHOP SCHEDULING PROBLEM

Noor Amira Isa[1], Noor Azizah Sidek[2*], Salleh Ahmad Bareduan[3], Azli Nawawi[4], Muhammad Marsudi[5]

[1] Department of Manufacturing Engineering, Faculty of Mechanical and Manufacturing Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia.
Email: nooramira1992@gmail.com
[2] Department of Mechanical Engineering, Centre for Diploma Studies, Universiti Tun Hussein Onn Malaysia, Pagoh Higher Education Hub, Jalan Panchor, 84600 Panchor, Johor, Malaysia.
Email: noorazizah@uthm.edu.my
[3] Department of Manufacturing Engineering, Faculty of Mechanical and Manufacturing Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia.
Email: saleh@uthm.edu.my
[4] Department of Mechanical Engineering Technology, Faculty of Engineering Technology, Universiti Tun Hussein Onn Malaysia, Pagoh Higher Education Hub, Jalan Panchor, 84600 Panchor, Johor, Malaysia
Email: azle@uthm.edu.my
[5] Department of Industrial Engineering, Faculty of Engineering, Islamic University of Kalimantan, 70123 South Kalimantan, Indonesia
Email: sholeh.marsudi1984@gmail.com
[*] Corresponding Author

Article Info:

Article history:
Received date: 27.10.2024
Revised date: 11.11.2024
Accepted date: 15.12.2024
Published date: 24.12.2024

To cite this document:

Isa, N. A., Sidek, N. A., Bareduan, S. A., Nawawi, A., & Marsudi, M. (2024). Optimization Of New Constructive Heuristic Algorithms For Permutation Flow Shop Scheduling Problem. *Journal of Information System and Technology Management, 9* (37), 203-219.

Abstract:

This paper presents a new heuristic designed specifically for minimizing makespan in scheduling problems. The proposed approach incorporates a dual bottleneck phase combined with a pre-initial arrangement to enhance optimization of new heuristic. By introducing both major and minor bottleneck identification phases, the heuristic effectively identifies critical processing machines with significant completion times. To evaluate the performance, this study employed the Taillard benchmark and the upper bound (UB) makespan as comparative tools, assessing the new heuristic against the well-known NEH heuristic. Computational results clearly demonstrate that the new heuristic significantly outperforms the NEH heuristic in reducing the total completion time. The consistent lower RPD values and negative percentage errors indicate that ICHA is more effective in approaching the optimal makespan as indicated by the Taillard UB.

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

DOI: 10.35631/JISTM.937016

Keywords:

Scheduling, Heuristic, Flow shop, Bottleneck, Makespan, Algorithm

Introduction

The Flowshop Scheduling Problem (FSP) is a widely used scheduling approach that finds applications in various industries, aiming to optimize the sequencing of tasks and processes (Abedinnia et al., 2016; Dong et al., 2008; Kalczynski & Kamburowski, 2007; Woo & Yim, 1998). FSP plays a crucial role in optimizing operations and improving efficiency which is extensively used in manufacturing industries such as automotive, aerospace, electronics, and consumer goods production (Azami et al., 2018; König et al., 2023; Sinthamrongruk et al., 2019). It helps optimize the scheduling of machines and assembly lines to minimize production time, reduce inventory, and improve resource utilization. Flowshop scheduling is also employed in food processing industries to schedule the sequence of tasks involved in food production, packaging, and distribution (Akkerman & van Donk, 2009).

Scheduling helps in reducing lead times, minimizing waste, and ensuring timely delivery of products. While in hospitals and healthcare facilities, flowshop scheduling used to optimize patient flow through various departments, such as outpatient clinics, diagnostic labs, operating rooms, and inpatient wards (Abdalkareem et al., 2021). Efficient scheduling improves resource allocation, reduces patient waiting times, and enhances overall service quality. Besides that, flowshop scheduling is also applied in logistics and transportation industries to optimize the movement of goods and vehicles through distribution centers, warehouses, and transportation networks (Abosuliman & Almagrabi, 2021). It helps in minimizing delivery times, reducing transportation costs, and improving overall supply chain efficiency. Lastly, flowshop scheduling is utilized in IT industries for tasks such as job scheduling in data centers, scheduling software builds and deployments and optimizing workflow in software development processes (Aladwani, 2020). It helps in maximizing computing resources, meeting deadlines, and improving software delivery efficiency. However, the specific challenges and objectives may vary depending on the industry and application context.

Permutation flowshop scheduling problem (PFSP) is one of most discussed problem among the authors in FSP (Abedinnia et al., 2016; Dong et al., 2008; Kalczynski & Kamburowski, 2007). PFSP refers to the different sequences in which the jobs can be processed on the machines. Each permutation represents a different order in which the jobs can be scheduled. In PFSP, the goal is to schedule a set of jobs to be processed on a series of machines in a way that minimizes the total completion time or makespan. While, in a flowshop environment, jobs must go through a series of machines, with each machine performing a specific task in a predefined order. There are no branching or alternative paths; each job follows the same sequence of machines. The objective is typically to minimize the makespan, which is the total time taken to complete all jobs where makespan is known as the time at which the last job completes processing. The main challenge in PFSP is to find the optimal sequence of jobs on the machines that minimizes the makespan. This problem is often NP-hard (Non-deterministic Polynomial-time hard), meaning that finding the optimal solution for large instances becomes computationally infeasible in a reasonable amount of time. Various algorithms are employed to solve permutation flowshop scheduling problems, including heuristic approaches like Johnson's algorithm, Nawaz, Enscore, and Ham (NEH) algorithm, genetic algorithms, and

Journal of Information System and Technology Management
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016
simulated annealing, among others (Fernandez-Viagas & Framinan, 2015; W. Liu et al., 2016; Low et al., 2004; Sidek et al., 2023). These algorithms aim to find near-optimal solutions within a reasonable computational time. Overall, PFSP is a complex optimization problem that requires balancing the sequencing of jobs on different machines to achieve efficient utilization of resources and minimize overall processing time.

In this study, the author focused on reducing the makespan of the total processing times in PFSP. This study introduced development of new performance criterion for partial sequence or insertion process and end up with makespan for the final sequence selection have been supported by the study of (Abedinnia et al., 2016) which some optional was proposed in extending the NEH heuristic. One of the suggested methods is to develop new performance criterion by employing different indicator values besides the processing time and choosing different sorting criterion for the selection of the best k-job in partial sequence. The k-job value shows the size of the subset of jobs that is being calculated and deployed during the partial sequencing process. The initial job arrangement and the opportunity of job insertion are the strength of NEH heuristic (Framinan et al., 2003). Thus, the improvement method for insertion process will give an effect to the makespan performance. Moreover, this study used Taillard's benchmark of processing time since lots of researchers used it as their standard data set to solve the PFSP especially in comparing the results with the other heuristic algorithms (Fernandez-Viagas & Framinan, 2015; Pan & Ruiz, 2013). The use of standard data set makes the researchers easier to evaluate the effectiveness newly proposed heuristic since the size of the problems contained in the set are representative for the real industrial problem (Taillard, 1990).

The research introduces a new scheduling method, ICHA, which improves upon existing techniques. ICHA identifies critical processing stages and optimizes the initial job sequence, leading to a more efficient and faster production schedule. The further designs of this paper are structured as follows. Section 2.0, 3.0 and 4.0 highlighted the methodology on the techniques and procedure of the proposed heuristic, comprehensive comparison of the proposed heuristic and NEH also the detailed results along with the graph, and conclusion of the paper respectively.

Methodology

This section focuses on developing a new heuristic known as Intelligent Constructive Heuristic Algorithm (ICHA) in solving permutation flow shop scheduling problem. The author divided this section into three main steps as follow:

i. Identification of the strength and weakness of NEH arrangement pattern using Gantt chart.
ii. Development of new heuristic identified as Intelligent Constructive Heuristic Algorithm (ICHA) in optimizing PFSP.
iii. Validation performance of ICHA against NEH heuristic.

This study started with the identification and application of NEH heuristic in PFSP. Then, development of this heuristic was simulated in Microsoft Excel spreadsheet with built-in Visual Basic for Application (VBA). NEH arrangement pattern was studied in finding the strength and weakness so that the parameters can be manipulated and applied in improving the solution of ICHA heuristic. A detailed study was carried out to find significant characteristics of scheduling data which affect scheduling performance. The significant character will be either

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016
processing idle time, jobs arrangement, processing time arrangement, or any of their combinations. Identification of NEH dataset which fails to produce the optimum schedule based on Taillard's benchmark upper bound was used to identify the NEH weaknesses. Analysis on failed dataset was conducted and investigated by using scheduling Gantt chart pattern to locate the important characteristics of dataset that lead to the NEH failure. Thus, the milestone for the first step of study (Identification of the strength and weakness of NEH arrangement pattern using Gantt chart) was achieved.

The next phase of this study was the development of new heuristics identified as Intelligent Constructive Heuristic Algorithms (ICHA) in optimizing PFSP. This study identified the several combinations of the observed scheduling Gantt chart that can be used to eliminate or reduce the NEH failure. The use of Microsoft Excel where all the data will be tabulated in a spreadsheet provides more visible analyses on the evaluation performance of the new intelligent algorithm in solving PFSP. Modification and improvement were added to the new intelligent algorithm steps until better results than the NEH are obtained to confirm the successful validation of the new Intelligent Constructive Heuristic Algorithm (ICHA).

Lastly, the final phase of this study focused on the validation performance of ICHA against NEH heuristic. The comparison was done using Taillard's benchmark. The comparison stage was made using ten datasets of Taillard's benchmark and it was used in evaluating the performance of ICHA and NEH heuristic. The study proceeded with the detailed analysis performance for each of the datasets. The datasets pattern was visualized in generated comparison tables for each flowshop setting in form of makespan value and the percentage improvements compared to the NEH.

Identification Of The Strength And Weakness Of NEH Arrangement Pattern Using Gantt Chart

This section focused on the detailed steps in the investigation of the NEH algorithm in PFSP. NEH algorithm needs to be programmed first before the NEH arrangement was studied.

Development of NEH Algorithm

The NEH algorithm was proposed by (Nawaz et al., 1983) appeared to be the best heuristic when makespan criterion was considered. There are two phases of NEH heuristic consists as follow:

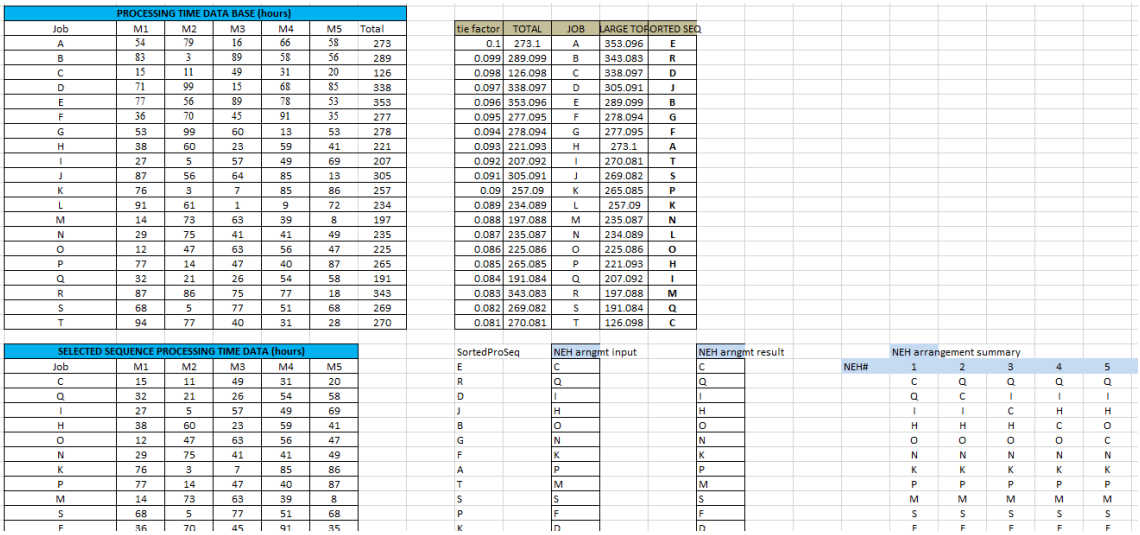
i. Sorting phase / Prioritizing phase
ii. Insertion phase

In NEH algorithm, sorting phase is where the jobs are sorted in descending order of their total processing times. The sorted list was then used in the insertion phase to determine the sequence in which jobs are added to the existing partial sequences. NEH algorithm gave a highest attention to the job with larger total processing time. The job should have higher priority to be processed first. For n-job of PFSP, the insertion phase consists of n iterations and 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 job. Then, the lowest makespan of partial sequence was used as a current k-jobs partial sequence in next iteration.

For this study, the author started the construction of Excel spreadsheet and the VBA coding for NEH by creating the spreadsheet interface. This interface consists of the processing time database table, sorted sequence table, NEH arrangement summary table, start-stop data table,

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

selected sequence data table and NEH arrangement input and output. Figure 1 shows the Excel spreadsheet interface for NEH algorithm.



PROCESSING TIME DATA BASE (hours)

Job	M1	M2	M3	M4	M5	Total
A	54	79	16	66	58	273
B	83	3	89	58	56	289
C	15	11	49	31	20	126
D	71	99	15	68	85	338
E	77	56	89	78	53	353
F	36	70	45	91	35	277
G	53	99	60	13	53	278
H	38	60	23	59	41	221
I	27	5	57	49	69	207
J	87	56	64	85	13	305
K	76	3	7	85	86	257
L	91	61	1	9	72	234
M	14	73	63	39	8	197
N	29	75	41	41	49	235
O	12	47	63	56	47	225
P	77	14	47	40	87	265
Q	32	21	26	54	58	191
R	87	86	75	77	18	343
S	68	5	77	51	68	269
T	94	77	40	31	28	270

tie factor	TOTAL	JOB	LARGE TO	RTED SEQ
0.1	273.1	A	353.096	E
0.099	289.099	B	343.083	R
0.098	126.098	C	338.097	D
0.097	338.097	D	305.091	J
0.096	353.096	E	289.099	B
0.095	277.095	F	278.094	G
0.094	278.094	G	277.095	F
0.093	221.093	H	273.1	A
0.092	207.092	I	270.081	T
0.091	305.091	J	269.082	S
0.09	257.09	K	265.085	P
0.089	234.089	L	257.09	K
0.088	197.088	M	235.087	N
0.087	235.087	N	234.089	L
0.086	225.086	O	225.086	O
0.085	265.085	P	221.093	H
0.084	191.084	Q	207.092	I
0.083	343.083	R	197.088	M
0.082	269.082	S	191.084	Q
0.081	270.081	T	126.098	C

SELECTED SEQUENCE PROCESSING TIME DATA (hours)

Job	M1	M2	M3	M4	M5
C	15	11	49	31	20
Q	32	21	26	54	58
I	27	5	57	49	69
H	38	60	23	59	41
O	12	47	63	56	47
N	29	75	41	41	49
K	76	3	7	85	86
P	77	14	47	40	87
M	14	73	63	39	8
S	68	5	77	51	68
F	36	70	45	91	35

SortedProSeq
E
R
D
J
B
G
F
A
T
S
P
K

NEH arngmt input
C
Q
I
H
O
N
K
P
M
S
F
D

NEH arngmt result
C
Q
I
H
O
N
K
P
M
S
F
D

NEH arrangement summary

NEH#	1	2	3	4	5
	C	Q	Q	Q	Q
	Q	C	I	I	I
	I	I	C	H	H
	H	H	H	C	O
	O	O	O	O	C
	N	N	N	N	N
	K	K	K	K	K
	P	P	P	P	P
	M	M	M	M	M
	S	S	S	S	S
	F	F	F	F	F

Figure 1: Excel Spreadsheet Interface for NEH Algorithm

After the spreadsheet interface was done, the Taillard's benchmark dataset was imported into the interface where the coding was then created. The sorting phase was done on the spreadsheet while the insertion phase was covered in VBA windows.

NEH Arrangement on Gantt Chart

In the past, minimizing makespan has been mistakenly regarded as equivalent to minimizing machine idle time, however the recent research by (Nawaz et al., 1983) has shown that although they are related, they are clearly different and in fact can conflict with each other. This is supported by the study of (W. Liu et al., 2016) which introduced an idle-time based index for composite heuristics for PFSP by calculating the unscheduled jobs fitness to the last job of partial schedule. Machine idle time has been rarely utilized in the literature for PFSP, but it is an important performance measure in manufacturing enterprises (Nawaz et al., 1983). Thus, this machine idle time minimization will be adopted as a strategy to minimize the makespan. As shown in Figure 2, apart from the machine operations, the empty space is categorized as front delay, idle time (IT), and back delay (J. Liu & Reeves, 2001). Front delay could be occupied by production prior to the current batch, while back delay could be filled in by the subsequent operations. But idle time is a real waste which should be minimized.

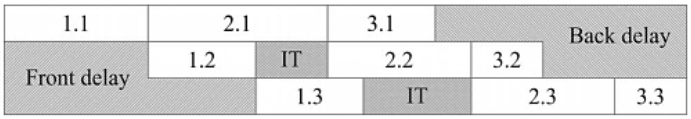


Figure 2: Front Delay, Idle Time (IT) and Back Delay of Schedule

Before the author proceeds with the analysis of NEH arrangement, the development of NEH heuristic was done in Excel VBA as shown in the previous section. In this study, the strength and weakness identification of NEH arrangement was done on problem size of 20 jobs and 5 machines. Another Excel spreadsheet was created to present the pattern of NEH arrangement. The Excel interface for 20 jobs and 5 machines arrangement was shown in Figure 3. This Excel

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

interface collects the sequence arrangement from the previous section of makespan result. All 10 sequences of Taillard's benchmark datasets were present in the form of Gantt chart.

START-STOP DATA

JOB	M1		M2		M3		M4		M5	
	START	STOP	START	STOP	START	STOP	START	STOP	START	STOP
C	0	15	15	26	26	75	75	106	106	126
Q	15	47	47	68	75	101	106	160	160	218
I	47	74	74	79	101	158	160	209	218	287
H	74	112	112	172	172	195	209	268	287	328
O	112	124	172	219	219	282	282	338	338	385
N	124	153	219	294	294	335	338	379	385	434
K	153	229	294	297	335	342	379	464	464	550
P	229	306	306	320	342	389	464	504	550	637
M	306	320	320	393	393	456	504	543	637	645
S	320	388	393	398	456	533	543	594	645	713
F	388	424	424	494	533	578	594	685	713	748
D	424	495	495	594	594	609	685	753	753	838
E	495	572	594	650	650	739	753	831	838	891
R	572	659	659	745	745	820	831	908	908	926
A	659	713	745	824	824	840	908	974	974	1032
B	713	796	824	827	840	929	974	1032	1032	1088
J	796	883	883	939	939	1003	1032	1117	1117	1130
G	883	936	939	1038	1038	1098	1117	1130	1130	1183
T	936	1030	1038	1115	1115	1155	1155	1186	1186	1214
L	1030	1121	1121	1182	1182	1183	1186	1195	1214	1286

GANTT CHART DATA

	1	C	2	Q	3	I	4	H	5	O	6	N	7	K	8	P	9	M	10	S	11
M5	106	20	34	58	0	69	0	41	10	47	0	49	30	86	0	87	0	8	0	68	0
M4	75	31	0	54	0	49	0	59	14	56	0	41	0	85	0	40	0	39	0	51	0
M3	26	49	0	26	0	57	14	23	24	63	12	41	0	7	0	47	4	63	0	77	0
M2	15	11	21	21	6	5	33	60	0	47	0	75	0	3	9	14	0	73	0	5	26
M1	0	15	0	32	0	27	0	38	0	12	0	29	0	76	0	77	0	14	0	68	0
Total	222		55		6		47		48		12		30		9		4		0		26

remarks:

front delay	222
idle time	592
back delay	463

Figure 3: Excel Interface for 20 Jobs and 5 Machines NEH Arrangement

Gantt chart was then used in visualizing the analysis of weak and best makespan of NEH. The data was visualized in form of Gantt chart so that the author can clearly see the idle time of dataset and how NEH schedule the job on that dataset. Figure 4 shows the constructed Gantt chart of Taillard's dataset for 20 jobs and 5 machines problem size. The total front delay for this scheduling dataset is 222 hours, while the idle time is 592 hours, and the back delay is 463 hours.

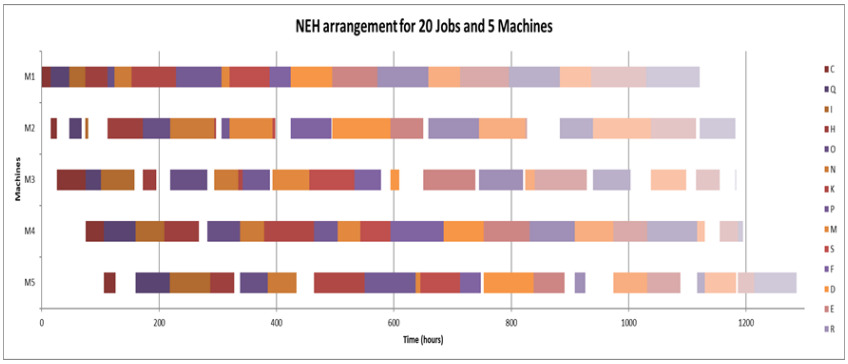


Figure 4: Gantt Chart of NEH Arrangement for 20 Jobs and 5 Machines

Development of ICHA in Optimizing Permutation Flow Shop Problem

In this sub-section, the author proceeds with the construction of ICHA involving the built-up new spreadsheet of Excel VBA. This spreadsheet is a bit differ to the NEH algorithm spreadsheet since there is modification and special combination phases made to improve the makespan result.

Development of ICHA

The process of ICHA development in spreadsheet Excel VBA is presented in Figure 5. There are four important phases to develop successful ICHA which are i) Bottleneck Identification

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

Phase (Major), ii) Bottleneck Identification Phase (Minor), iii) Initial Partial Sequence Phase, iv) Job Insertion Phase

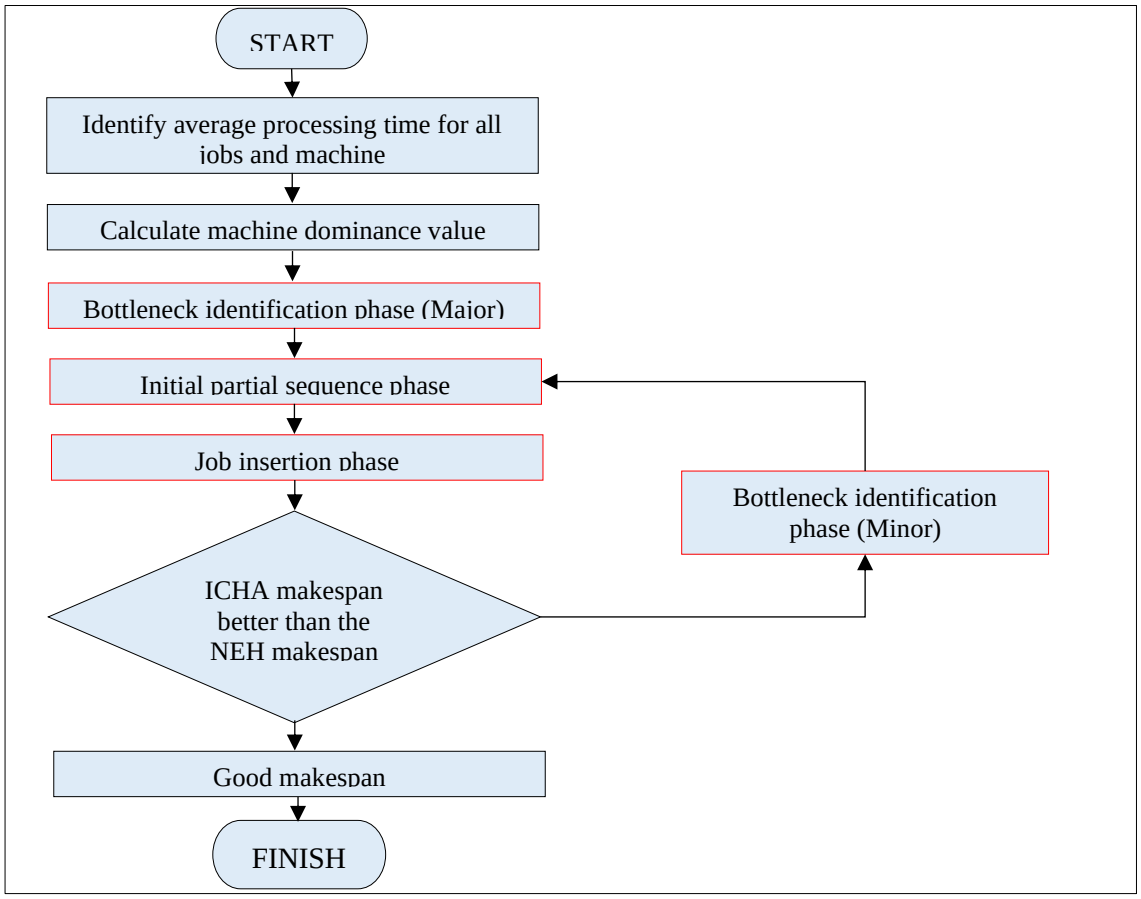


Figure 5: Flowchart of ICHA Procedure

All the phases were included in the development of Excel spreadsheet and VBA coding. Initial partial sequence phase was done in the Excel spreadsheet while the bottleneck identification phase and job insertion phase were done in VBA window. The Excel interface for ICHA was shown in Figure 6. The NEH algorithm spreadsheet was improve with a modification of dominance calculation and the bottleneck-base rule while the rest maintain the same.

Journal of Information System and Technology Management
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

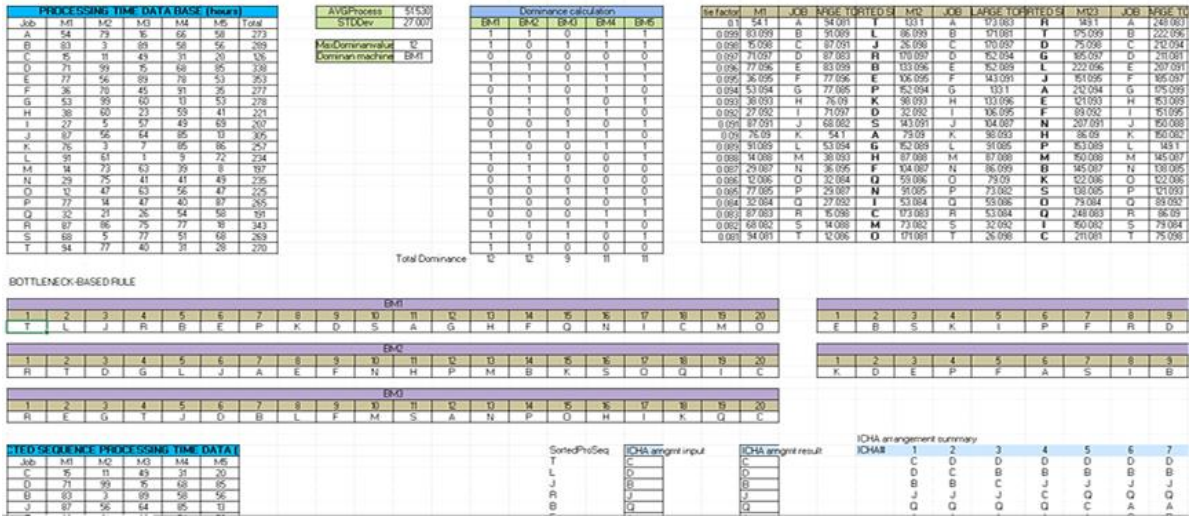


Figure 6: Excel Spreadsheet Interface for ICHA

Bottleneck Identification Phase

Bottleneck identification phase is the phase where bottleneck-based analysis was used to identify the bottleneck stage which is dominant machine that contributes to large total completion time. Then, it will be used to determine the jobs schedule in the bottleneck stage (Spachis, 1978). Dominance values are used to decide the dominant machine based on their job processing time on each machine. Average processing time for all machines will be calculated and used as indicator in deciding the dominance value. The value of one (1) is used for processing time higher than the average processing time for all machines, while value zero (0) is used for processing time lower than the average processing time for all machines. For major phase, the machine with highest value will be identified as bottleneck machine 1 (BM1). While the rest of the machines are identified as bottleneck machine 2 (BM2) on the second highest value respectively until the lowest value and total bottleneck machines are depending on the problem sizes. When there are same highest values, the first highest value of machine was chosen. While for minor phase, the machine with second highest value was chosen as the first bottleneck machine. This bottleneck identification phase is important to classify the processes machines criticality (bottleneck machines) before proceeding to the next phase. Table 1 and Table 2 shows example of processing times and dominance values dataset for 20 jobs and 5 machines.

Table 1: Example of Processing Times Dataset for 20 Jobs and 5 Machines

PROCESSING TIME DATA BASE (hours)						
Job/Machine	M1	M2	M3	M4	M5	Total
A	27	79	22	93	38	259
B	92	23	93	22	84	314
C	75	66	63	64	62	330
D	94	5	53	81	10	243
E	18	15	30	94	11	168
F	41	51	34	97	93	316
G	37	2	27	54	57	177
H	58	81	30	82	81	332

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

I	56	12	54	11	10	143
J	20	40	77	91	40	268
K	2	59	24	23	62	170
L	39	32	47	32	49	199
M	91	16	39	26	90	262
N	81	87	66	22	34	290
O	33	78	41	12	11	175
P	14	41	46	23	81	205
Q	88	43	24	34	51	240
R	22	94	23	87	21	247
S	36	1	68	59	39	203
T	65	93	50	2	27	237

Table 1 shows the processing times for the problem size of 20 machines and 5 jobs. The dataset was taken from Taillard's benchmark.

Table 2: Dominance Value Results for 20 Jobs and 5 Machines

	Dominance calculation				
	M1	M2	M3	M4	M5
	0	1	0	1	0
	1	0	1	0	1
	1	1	1	1	1
	1	0	1	1	0
	0	0	0	1	0
	0	1	0	1	1
	0	0	0	1	1
	1	1	0	1	1
	1	0	1	0	0
	0	0	1	1	0
	0	1	0	0	1
	0	0	0	0	1
	1	0	0	0	1
	1	1	1	0	0
	0	1	0	0	0
	0	0	0	0	1
	1	0	0	0	1
	0	1	0	1	0
	0	0	1	1	0
	1	1	1	0	0
Total	9	9	8	10	10

From the calculation of processing times for 20 jobs and 5 machines, the average processing time for all jobs on all machines is 47.78. Thus, the dominance value is depending on the job processing times. Table 2 shows the dominance value results for the given processing times. Total dominance values were summed up from the values on each machine. From this dataset, M4 was chosen as the bottleneck machine since it has the first greater value compared to the other machines. If the makespan result failed to beat the NEH result for the major bottleneck

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016
identification phase, M5 will then be chosen to be bottleneck machine for minor bottleneck identification phase.

Initial Partial Sequence Phase

Initial partial sequence phase is an initial job arrangement which is one of the strengths of NEH heuristic (Framinan et al., 2003). This job sorting with one priority is used to form the initial partial sequence which brings the success of NEH heuristic. NEH heuristic is known as the best heuristic in solving permutation flow shop problem with makespan minimization objective. NEH heuristic gives a large attention on the job with larger total processing time where it should have a higher priority to process first. The jobs are arranged in decreasing order of the total processing time and then, the first two jobs are picked from the job arrangement list and were scheduled. The lowest makespan value was chosen as current partial sequence. This study also focuses on the initial sequence arrangement to obtain the best arrangement which is the nearest Taillard lower bound. In this study, the initial sequence arrangements depend on the result of machine bottleneck where it have been classified into multiple bottleneck machines based on problem sizes. This was supported by the latest study of (Chen & Chen, 2009) in which the usage of bottleneck affects the final job sequence which most of it leads to a better result. There are few variables used in determining the initial partial sequence for each bottleneck machine which is called as pre-initial arrangement.

Table 3: Pre-initial Arrangement for 5 Machines Problem

Case study	Bottleneck Machines	Processing times summation
5M	BM1	M1, (M2+M3+M4+M5)
	BM2	M1, (M1+M2), M2, (M2+M3+M4+M5), (M3+M4+M5)
	BM3	(M1+M2), (M1+M2+M3), M3, (M3+M4+M5), (M4+M5)
	BM4	(M1+M2+M3), (M1+M2+M3+M4), M4, (M4+M5), M5
	BM5	(M1+M2+M3+M4), M5

Table 3 shows the pre-initial arrangement for 5 machines. All the variables were tested to obtain the minimum makespan. In this study, each bottleneck machine considered the processing times of machine before, middle and after the bottleneck machine. The variable tests are crucial in identifying the possibilities of any lowest makespan. On the previous study of (Chen & Chen, 2009), the observation found that on the small sample (m = 4, n = 6, 10, 15, 20), the earlier machines (M1 and M2) are criticality giving the large completion times in scheduling process, thus the job with higher sum of processing times on machines must be processed first. The sequence guide of NEH were still applied in this study which are the job with highest processing time must be processed first and then proceed with the job with second highest processing time and continue with the next job until the last job with lowest processing time. It is important to ensure that there is not much idle time towards the end of the overall sequence.

Job Insertion Phase

Job insertion phase is the phase where the k-th job is assigned to the k possible slots in the current partial sequences which was obtained from the previous iteration consisting of k - 1 jobs. The partial sequence with the lowest objective function will be used as a current k-jobs

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

partial sequence for the next iteration (Taillard, 1990). This job insertion phase introduced in NEH heuristic was shown to be effective in producing a sequence near optimal solution and with smaller total completion time (Framinan et al., 2003). This attracts researcher to make some modifications to obtain better near optimal solution by optimizing the partial sequences at the end of each iteration of NEH insertion phase (Framinan et al., 2003; Isa et al., 2020).

Abedinnia et al., 2016 continued the study of Laha and Sarin in optimizing the total flowtime in a permutation flow shop manufacturing system (Abedinnia et al., 2016). Priority orders were proposed effectively in the insertion phase by initiating the local search with the most important job before proceeding to the less important ones and then replacing the current sequence which leads to an improvement. The idea behind these modifications is to give a chance for position changes of important jobs which has higher impact on the final solution than a change in the position of less important jobs. Example of insertion phase technique can be seen in Figure 7. The current partial sequences are AB and job C as the new job. Thus, Job C will be inserted at the front of job A, in the middle jobs A and B, and after job B. So, the next possible partial sequences will be CAB, ACB and ABC. Then, the sequence with lowest completion time was chosen as the next partial sequence. This insertion phase was applied at each stage of job sequencing.

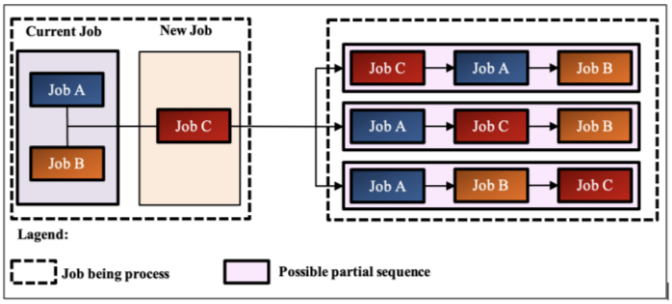


Figure 7: Insertion Phase Technique

Tie-breaking Rule

During job insertion phase, a tie always occurred where at each sequencing stage, the makespan values are the same. Hereby, this brings to a new rule in solving a tie problem known as tie-breaking rules. In NEH algorithm, no specific tie-breaking method is used, however, once a tie occurs, the first feasible position is usually selected. Researchers focused much more attention on the second type of ties which is proposing insertion tie-breaking rules (Laha & Sarin, 2009). The NEH job priority order was shown to be superior when tested to 177 different examined orders (Framinan et al., 2003). Thru these analysis, large number of published papers suggesting a new priority order thus introducing the tie-breaking rules in the insertion phase (Laha & Sarin, 2009).

Kalczynski and Komburowski suggested a better priority order with a combination of simple tie breaking in the insertion phase which secures the optimality of two machine case and improving the general performance (Kalczynski & Kamburowski, 2008). Low developed MNEH that introduced a tie-breaking rule that chooses the position with the least idle-time on the bottleneck machine (Low et al., 2004). In 2008, Dong et. al introduced the NEH-D heuristic with tie-breaking rule that choosing the position with the least machine utilization variation (Dong et al., 2008). In addition, Viagas and Framinan presented a new tie breaking rule based on an estimation of idle times of the different subsequences which is by choosing the position

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

with the least front delay and idle time (Fernandez-Viagas & Framinan, 2014). Liu selected the position with minimum completion times and balanced workloads at all machines to solve ties problem (J. Liu & Reeves, 2001). Thus, in this study, a tie-breaking rule that will be used is whether the first feasible position or the last feasible position which the results lead to minimum completion time.

Performance comparison of heuristics

In this study, NEH heuristic have been set as the benchmark performance test since NEH is the most succeeded heuristic in makespan minimization. To validate the statistical tests of ICHA effectiveness, the processing time of common benchmark (Taillard's benchmark) will be used in this study. This Taillard's benchmark is widely used for permutation flow shop problem (PFSP) with the makespan criterion (Sidek et al., 2023). Taillard datasets have been chosen in compatibly this study with the other researchers in the same field. Besides that, both NEH and ICHA will be compared to identify the capability of both solutions (in producing minimum makespan) towards the upper bound of Taillard's benchmark. The datasets sample can be seen in Figure 8 for 20 jobs 5 machines problem size and 20 jobs 10 machines problem size.

```
number of jobs, number of machines, initial seed, upper bound and lower bound :
         20           5   873654221        1278        1232
processing times :
 54 83 15 71 77 36 53 38 27 87 76 91 14 29 12 77 32 87 68 94
 79  3 11 99 56 70 99 60  5 56  3 61 73 75 47 14 21 86  5 77
 16 89 49 15 89 45 60 23 57 64  7  1 63 41 63 47 26 75 77 40
 66 58 31 68 78 91 13 59 49 85 85  9 39 41 56 40 54 77 51 31
 58 56 20 85 53 35 53 41 69 13 86 72  8 49 47 87 58 18 68 28
number of jobs, number of machines, initial seed, upper bound and lower bound :
         20           5   379008056        1359        1290
processing times :
 26 38 27 88 95 55 54 63 23 45 86 43 43 40 37 54 35 59 43 50
 59 62 44 10 23 64 47 68 54  9 30 31 92  7 14 95 76 82 91 37
 78 90 64 49 47 20 61 93 36 47 70 54 87 13 40 34 55 13 11  5
 88 54 47 83 84  9 30 11 92 63 62 75 48 23 85 23  4 31 13 98
 69 30 61 35 53 98 94 33 77 31 54 71 78  9 79 51 76 56 80 72
```

Figure 8: Taillard's Benchmark Sample Datasets

To measure the solution quality of each algorithm, the relative percentage deviation (RPD) is employed as the performance measure. HS represents the value obtained by heuristics on problem size while UB is the upper bound of Taillard's benchmark (W. Liu et al., 2017; Sidek et al., 2019).

Computational Results

Efficiency of heuristics for 20 Jobs and 5 Machines (20J5M)

Performance of both heuristics was evaluated based on their makespan value (percentage error) and the RPD for 20 jobs and 5 machines. The ICHA makespan result for both major n minor phase of bottleneck identification and pre-initial arrangement were shown in Table 4. While Table 5 shows makespan result for both heuristics. The heuristics performance result was then compared to Taillard upper bound (UB).

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

Table 4: ICHA Makespan Result and ICHA Pre-initial Arrangement for 20J5M

DATA	NEH Makespan	ICHA B1 Makespan	Bottleneck Machine 1 (B1)	ICHA B1 Pre-Initial Arrangement	ICHA B2 Makespan	Bottleneck Machine 2 (B2)	ICHA B2 Pre-Initial Arrangement
1	1286	1301	BM1	M2-M5	1301	BM2	M2-M5
2	1365	1373	BM5	M1-M4	1372	BM2	M3-M5
3	1159	1145	BM1	M2-M5	1329	BM2	M1-M2
4	1325	1323	BM5	M5	1323	BM1	M2-M5
5	1305	1305	BM5	M2-M5	1256	BM1	M1
6	1228	1212	BM3	M4-M5	1210	BM4	M1-M4
7	1278	1251	BM4	M4	1251	BM3	M1-M3
8	1223	1226	BM5	M1-M4	1226	BM1	M2-M5
9	1291	1305	BM1	M1	1255	BM3	M4-M5
10	1151	1144	BM4	M5	1144	BM5	M5

Based on the results above, ICHA B1 achieves a better makespan in 6 out of 10 instances compared to the NEH makespan. Since identifying the major bottleneck did not suffice, minor bottleneck identification was conducted to further improve the makespan solution. This process revealed that B2 achieves an even better makespan than B1 in 4 instances, outperforming another NEH makespan. The table demonstrates that the pre-initial arrangement is not fixed, but it is evident that pre-initial arrangement variables in Table 3 contribute to achieving a better makespan.

Table 5: NEH and ICHA Makespan Result for 20J5M

DATA	NEH makespan	ICHA (B1 +B2) makespan	Status
1	1286	1301	Lose
2	1365	1372	Lose
3	1159	1126	Win
4	1325	1323	Win
5	1305	1256	Win
6	1228	1220	Win
7	1278	1251	Win
8	1223	1226	Lose
9	1291	1255	Win
10	1151	1144	Win

The frequency of evidence suggesting that ICHA is better than NEH is displayed in Table 6. The table shows that ICHA achieves a better makespan in 7 out of 10 instances compared to NEH. From this result, the author concludes that combining major and minor bottleneck identification leads to an improved makespan.

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

Table 6: Performance Comparison of Both Heuristics

DATA	ICHA (B1+B2) Makespan	NEH Makespan	Taillard Upper Bound (UB)	Percentage Error ICHA vs NEH	ICHA Relative Percentage Deviation (RPD_{ICHA})	NEH Relative Percentage Deviation (RPD_{NEH})
1	1301	1286	1278	1.15%	1.80%	0.63%
2	1372	1365	1359	0.51%	0.96%	0.44%
3	1126	1159	1081	-2.93%	4.16%	7.22%
4	1323	1325	1293	-0.15%	2.32%	2.47%
5	1256	1305	1236	-3.90%	1.62%	5.58%
6	1220	1228	1195	-0.66%	2.09%	2.76%
7	1251	1278	1239	-2.16%	0.97%	3.15%
8	1226	1223	1206	0.24%	1.66%	1.41%
9	1255	1291	1230	-2.87%	2.03%	4.96%
10	1144	1151	1108	-0.61%	3.25%	3.88%
			Average Percentage	-1.14%	2.09%	3.25%

In this study, the author used Taillard's Upper Bound as a comparative tool for both the ICHA and NEH heuristics (Fernandez-Viagas & Framinan, 2014). The percentage error indicates how much better or worse the ICHA makespan is compared to the NEH makespan. Positive values indicate that ICHA makespan is worse than NEH (data sets 1, 2, 8), while negative values indicate that ICHA makespan is better than NEH (data sets 3, 4, 5, 6, 7, 9, 10). On average, the percentage error is -1.14%, which means that ICHA generally outperforms NEH by 1.14%.

From Table 6, RPDICHA ranges from -3.90% to 4.16%, while RPDNEH ranges from 0.44% to 7.22%. The average RPD for ICHA is 2.09%, while for NEH it is 3.25%. This shows that ICHA's makespan is closer to the UB, on average, compared to NEH. In data set 3, ICHA significantly outperforms NEH with a -2.93% percentage error and a lower RPDICHA of 4.16% compared to NEH's 7.22%. In data set 5, ICHA again significantly outperforms NEH with a -3.90% percentage error and a lower RPDICHA of -3.90% compared to NEH's 5.58%. The only data sets where NEH slightly outperforms ICHA (positive percentage error) are 1, 2, and 8, but the differences are minimal (1.15%, 0.51%, and 0.24%, respectively). The ICHA (B1+B2) makespan values are consistently lower than or comparable to the NEH makespan values across all data sets. Specifically, ICHA achieves a lower makespan in 7 out of 10 data sets (as seen in data sets 1, 3, 5, 6, 7, 9, and 10). ICHA heuristic demonstrates superior performance compared to the NEH heuristic in minimizing makespan. The consistent lower RPD values and negative percentage errors indicate that ICHA is more effective in approaching the optimal makespan as indicated by the Taillard Upper Bound. The results suggest that the ICHA heuristic is a viable and more efficient alternative to the NEH heuristic for scheduling problems focused on makespan minimization.

From the time series plot graph in Figure 9, ICHA makespan (blue circles and solid line) generally achieves lower or comparable makespan values compared to the NEH makespan (red squares and dashed line). In 7 out of the 10 data sets, ICHA performs better (i.e., has a lower makespan) than NEH. These data points are particularly noticeable at indices 1, 3, 5, 6, 7, 9,

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

and 10. The makespan difference is especially notable in data sets 1, 3, 5, 7, and 9, where the ICHA makespan is significantly lower than the NEH makespan while the Taillard Upper Bound (green diamonds and dotted line) serves as a benchmark for the best possible makespan. Both ICHA and NEH makespan values are consistently higher than the Taillard Upper Bound, as expected, since the upper bound represents an ideal benchmark. The gap between ICHA and the Taillard Upper Bound is generally smaller compared to the gap between NEH and the Taillard Upper Bound, indicating that ICHA is closer to the optimal solution.

ICHA heuristic demonstrates a clear advantage over the NEH heuristic, producing lower makespan values in most cases. The average makespan reduction achieved by ICHA compared to NEH is significant, as reflected in the consistent performance across the different data sets. The performance of ICHA relative to the Taillard Upper Bound suggests that it is an effective heuristic for minimizing makespan, though there is still room for improvement to reach the optimal makespan values represented by the upper bound. The graph visually confirms that the ICHA heuristic is more effective than the NEH heuristic in minimizing makespan, providing a more efficient scheduling solution closer to the optimal benchmark provided by the Taillard Upper Bound.

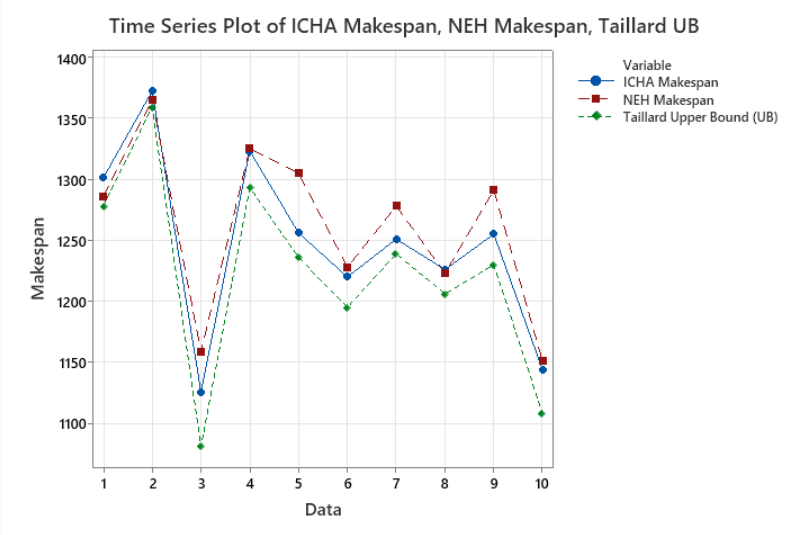


Figure 9: Time Series Plot for Performance Comparison to the Taillard UB

Conclusions

This study focused on reducing the makespan of the processing times in PFSP. Intelligent Constructive Heuristic Algorithm (ICHA) was presented in this paper with some modification of NEH heuristic. Dual bottleneck identification phase was introduced followed by newly pre-initial arrangement of processing times. Our statistical result analyses demonstrated that the modification proposed and evaluated in this paper leads to the better performance compared to the NEH heuristic for all 10 datasets of Taillard benchmarks and outperform the Taillard upper bound (UB) for 2 over 10 datasets. This study shows that ICHA compromising lowest makespan since almost all of 10 datasets shows a huge difference value compared to the NEH. Numerical studies showed that using dual bottleneck identification phase and pre-initial arrangement has potential in improving the algorithm performance. The idea of using dual bottleneck identification phase and pre-initial arrangement shows that the method used can improve the NEH heuristic result. None of the previous studies has discovered this concept before thus the further research should cover this study in various and large range of Taillard

Journal of Information System and Technology Management
EISSN: 0128-1666
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016

benchmark on m-machines (m = 10, 20) and n-jobs (n = 50, 100, 200, 500) to test the effectiveness for large number of jobs and machines especially in large sector industries. To sum up, this research introduces a new scheduling method, ICHA, which improves upon existing techniques. ICHA identifies critical processing stages and optimizes the initial job sequence, leading to a more efficient and faster production schedule.

Acknowledgement
This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM) through Tier 1 (Vot Q131).

References

Abdalkareem, Z. A., Amir, A., Al-Betar, M. A., Ekhan, P., & Hammouri, A. I. (2021). Healthcare scheduling in optimization context: a review. *Health and Technology*, *11*, 445–469.

Abedinnia, H., Glock, C. H., & Brill, A. (2016). New simple constructive heuristic algorithms for minimizing total flow-time in the permutation flowshop scheduling problem. *Computers & Operations Research*, *74*, 165–174.

Abosuliman, S. S., & Almagrabi, A. O. (2021). Routing and scheduling of intelligent autonomous vehicles in industrial logistics systems. *Soft Computing*, *25*, 11975–11988.

Akkerman, R., & van Donk, D. P. (2009). Analyzing scheduling in the food-processing industry: structure and tasks. *Cognition, Technology & Work*, *11*, 215–226.

Aladwani, T. (2020). Types of task scheduling algorithms in cloud computing environment. *Scheduling Problems-New Applications and Trends*.

Azami, A., Demirli, K., & Bhuiyan, N. (2018). Scheduling in aerospace composite manufacturing systems: a two-stage hybrid flow shop problem. *The International Journal of Advanced Manufacturing Technology*, *95*, 3259–3274.

Chen, C.-L., & Chen, C.-L. (2009). A bottleneck-based heuristic for minimizing makespan in a flexible flow line with unrelated parallel machines. *Computers & Operations Research*, *36*(11), 3073–3081.

Dong, X., Huang, H., & Chen, P. (2008). An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research*, *35*(12), 3962–3968.

Fernandez-Viagas, V., & Framinan, J. M. (2014). On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers & Operations Research*, *45*, 60–67. https://doi.org/http://dx.doi.org/10.1016/j.cor.2013.12.012

Fernandez-Viagas, V., & Framinan, J. M. (2015). NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. *Computers & Operations Research*, *60*, 27–36. https://doi.org/http://dx.doi.org/10.1016/j.cor.2015.02.002

Framinan, J. M., 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. *International Journal of Production Research*, *41*(1), 121–148. https://doi.org/10.1080/00207540210161650

Isa, N. A., Bareduan, S. A., & Zainudin, A. S. (2020). Bottleneck-Based Heuristic for Permutation Flowshop Scheduling. *IOP Conference Series: Materials Science and Engineering*, *824*(1), 12020.

Kalczynski, P. J., & Kamburowski, J. (2007). On the NEH heuristic for minimizing the makespan in permutation flow shops. *Omega*, *35*(1), 53–60. https://doi.org/10.1016/J.OMEGA.2005.03.003

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 9 Issue 37 (December 2024) PP. 203-219
DOI: 10.35631/JISTM.937016
Kalczynski, P. J., & Kamburowski, J. (2008). An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers and Operations Research*, *35*(9), 3001–3008. https://doi.org/10.1016/j.cor.2007.01.020

König, S., Reihn, M., Abujamra, F. G., Novy, A., & Vogel-Heuser, B. (2023). Flexible scheduling of diagnostic tests in automotive manufacturing. *Flexible Services and Manufacturing Journal*, *35*(2), 320–342.

Laha, D., & Sarin, S. C. (2009). A heuristic to minimize total flow time in permutation flow shop. *Omega*, *37*(3), 734–739.

Liu, J., & Reeves, C. R. (2001). Constructive and composite heuristic solutions to the P//∑ Ci scheduling problem. *European Journal of Operational Research*, *132*(2), 439–452.

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. *Engineering Optimization*, *48*(10), 1808–1822.

Liu, W., Jin, Y., & Price, M. (2017). A new improved NEH heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics*, *193*, 21–30. https://doi.org/10.1016/j.ijpe.2017.06.026

Low, C., Yeh, J.-Y., & Huang, K.-I. (2004). A robust simulated annealing heuristic for flow shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *23*, 762–767.

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

Pan, Q. K., & Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. In *Computers and Operations Research* (Vol. 40, Issue 1, pp. 117–128). https://doi.org/10.1016/j.cor.2012.05.018

Sidek, N. A., Bareduan, S. A., & Nawawi, A. (2019). Performance Investigation of Artificial Bee Colony (ABC) Algorithm for Permutation Flowshop Scheduling Problem (PFSP). *Journal of Physics: Conference Series*, *1150*(1), 012060. https://doi.org/10.1088/1742-6596/1150/1/012060

Sidek, N. A., Bareduan, S. A., & Nawawi, A. (2023). Development of Guided Artificial Bee Colony (GABC) Heuristic for Permutation Flowshop Scheduling Problem (PFSP). *Journal of Advanced Research in Applied Sciences and Engineering Technology*, *33*(3), 393–406.

Sinthamrongruk, T., Premphet, P., Smutkupt, U., Dahal, K., & Smith, L. (2019). Production plan scheduling on electronic factory. *2019 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON)*, 155–158.

Spachis, A. S. (1978). *Job-shop scheduling with approximate methods*. University of London.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, *47*(1), 65–74.

Woo, H.-S., & Yim, D.-S. (1998). A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers & Operations Research*, *25*(3), 175–182.