IMPACT OF INPUT SEQUENCE ON N-NEH+ ALGORITHM

[1]Radosław PUKA, [1]Bartosz ŁAMASZ

[1]AGH University of Science and Technology, Cracow, Poland, EU,
rpuka@zarz.agh.edu.pl, blamasz@zarz.agh.edu.pl

https://doi.org/10.37904/clc.2022.4556

Abstract

The job scheduling problem is one of the biggest optimization challenges for manufacturing companies. A properly performed planning can reduce both cost and production time. A scheduling problem that has been the subject of research in numerous research papers is the permutation flow shop scheduling problem (PFSP) with makespan as optimization (minimization) criterion. Frequently used algorithms to solve PFSP is the constructive deterministic heuristic algorithm NEH. Many researchers have analyzed the effectiveness of the NEH algorithm. An example of the issues studied is the importance of the order of input data on the results obtained by the NEH algorithm. In the literature, a number of variants of NEH-based algorithms can be found. The N-NEH+ algorithm is considered to be one of the most efficient. Therefore, this paper focuses on analyzing the influence of the input sequence on the results obtained by the N-NEH+ algorithm. Two most popular benchmarks were used to analyze the influence of the input sequence: the Taillard's benchmark and the VRF benchmark. The results obtained confirm that the input sequence has a significant impact on the results obtained by the N-NEH+ algorithm. However, this influence is less than that of the NEH algorithm.

Keywords: PFSP, N-list technique, N-NEH+ algorithm, scheduling, input sequence

1. INTRODUCTION

The permutation flow shop scheduling problem (PFSP) is a production problem for finding the best sequence of jobs to be processed in minimizing a given objective function (e.g. makespan, tardiness, cost or flow time). A characteristic of PFSP is that the order of execution of all of the n jobs is the same for each of the m machines. In this paper, the makespan value (C_{max}) is used as an optimization criterion. The problem thus defined can be written as $Fm|prmu|C_{max}$ [1]. For the number of machines more than 2, $Fm|prmu|C_{max}$ is an NP-hard problem [2].

Due to the complexity of PFSP, heuristic algorithms are applied to solve it. One of the most popular heuristics used is the NEH algorithm [3], whose operation can be described in the following three steps:

1) Sort in non-increasing order the list of jobs according to their total processing time.
2) Add the first job from the list to a partial sequence and deletes the job from the list.
3) Until the list is not empty, add the first job from the list to a partial sequence to get makespan as short as possible, and then delete the scheduled job from the list.

The efficiency of the NEH algorithm is strongly influenced by the first step, which results in the input sequence. Many studies have focused on analyzing the impact of the method of determining the priority for each job (after which jobs will be subsequently sorted) on the final result of the NEH algorithm [4-12]. One of the most efficient modifications of the NEH allogrithm is the N-NEH+ algorithm [13]. The operation of the N-NEH+ algorithm is based on the use of the N-list technique. It is a modification of step 3 of the NEH algorithm, by checking not one but N jobs each time. This paper focuses on verifying whether the input sequence for the N-NEH+ algorithm is as relevant as for the NEH algorithm.

2. INPUT SEQUENCE

Determining the input sequence is the first step of both the NEH and N-NEH+ algorithms. The input sequence is created by assigning a priority (p_j) to each job and then sorting the jobs according to this priority. For testing, selected methods of priority calculation based on the work of [4-11] were used. The jobs for creating the input sequence were sorted non-increasing according to job priorities calculated as follows:

- TPT (NEH): $p_j = \sum_{i=1}^{m} p_{i,j}$

where $p_{i,j}$ is the processing time of job *j* on machine *i*

- TPT1: $p_j = \sum_{i=2}^{m-1} p_{i,j}$
- TPT2: $p_j = \sum_{i=2}^{m} p_{i,j}$
- TPT3: $p_j = \sum_{i=1}^{m-1} p_{i,j}$
- S: $p_j = \sum_{i=1}^{m} (p_{i,j})^2$
- SR: $p_j = \sum_{i=1}^{m} (p_{i,j})^{1/2}$
- SP: $p_j = \sum_{i=1}^{m} i * p_{i,j}$
- WSUM: $p_j = \sum_{i=1}^{m} (m - i + 1) p_{i,j}$
- ABS_DIF: $p_j = \sum_{i=1}^{m} \sum_{j'=1}^{n} |p_{i,j} - p_{i,j'}|$
- WABS_DIF: $p_j = \sum_{i=1}^{m} \sum_{j'=1}^{n} (m - i + 1) |p_{i,j} - p_{i,j'}|$
- SS_SRA: $p_j = \sum_{j'=1, j' \neq j}^{n} \sum_{i=2}^{m} |r_{i,j,j'}|$

where: $r_{i,j,j'} = p_{i,j} - p_{i,j'}$

- SS_WSRA: $p_j = \sum_{j'=1, j' \neq j}^{n} \sum_{i=2}^{m} (m - i + 1) |r_{i,j,j'}|$
- SS_SRS: $p_j = \sum_{j'=1, j' \neq j}^{n} \sum_{i=2}^{m} (r_{i,j,j'})^2$
- SKE: $p_j = AVG_j + STD_j + abs(SKE_j)$

where $AVG_j = \frac{1}{m} \sum_{i=1}^{m} p_{i,j}$, $STD_j = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (p_{i,j} - AVG_i)^2}$, $abs(SKE_j) = \frac{\frac{1}{m} \sum_{i=1}^{m} (p_{i,j} - AVG_i)^3}{\left(\sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (p_{i,j} - AVG_i)^2}\right)^3}$

- SKEx: $p_j = AVG_j + abs(SKE_j)$
- SKEy: $p_j = AVG_j + STD_j$

The presented job priorities were used to analyze the effect of input sequece on the results of the N-NEH+ algorithm. The measure of solution quality for a given benchmark is the average relative percentage deviation (ARPD) calculated as:

$$ARPD_{SP} = \frac{1}{I} \sum_{ic=1}^{I} \frac{M_{SP,ic} - M_{best,ic}}{M_{best,ic}} \quad (1)$$

where *SP* is the sorting priority, *I* indicates the number of instances in the benchmark, $M_{SP,ic}$ is the makespan value of the sorting priority *SP* for instance *ic* and $M_{best,ic}$ is the best known value of makespan for instance *ic.*

3. COMPUTATIONAL EXPERIMENTS

The N-NEH+ algorithm performance with the presented sorting priorities were tested using two benchmarks: Taillard's benchmark with 120 instances [14], and VRF benchmark with 240 Small (S) and 240 Large (L) [15]. The algorithm was implemented in C# and all the computations were carried out on a computer with two Intel Xeon E5-2660 v4 CPUs.

Tables 1-3 show the results of the N-NEH+ algorithm for different N-list lengths for instances, respectively: Taillard (**Table 1**), VRF Small (**Table 2**), and VRF Large (**Table 3**).

Table 1 ARPD[%] values for TAILLARD'S benchmark (the best ARPD value for a given N is bolded).

Sorting priority	N				
	1	**2**	**4**	**8**	**16**
TPT	3,326	2,947	**2,552**	2,317	2,195
TPT1	3,947	3,490	2,989	2,606	2,319
TPT2	3,602	3,130	2,769	2,485	2,258
TPT3	3,464	3,086	2,625	2,364	2,243
S	3,279	2,952	2,642	2,314	2,184
SR	3,462	3,028	2,696	2,414	2,246
SP	3,882	3,345	2,920	2,540	2,284
WSUM	3,633	3,120	2,732	2,386	2,217
ABS_DIF	3,724	3,349	3,011	2,713	2,569
WABS_DIF	4,008	3,436	3,000	2,748	2,512
SS_SRA	3,752	3,336	3,041	2,726	2,532
SS_WSRA	3,808	3,275	2,936	2,692	2,506
SS_SRS	3,777	3,334	3,003	2,701	2,528
SKE	3,282	**2,886**	2,589	2,353	**2,181**
SKEx	**3,201**	2,968	2,597	**2,309**	2,189
SKEy	3,273	2,919	2,644	2,408	2,204

For the Taillard benchmark, three sorting priorities stand out: TPT, SKE, and SKEx. For the NEH algorithm (N=1), the best priority was found to be SKEx, which is clearly outperformed by the others. The original sorting priority of the NEH algorithm was found to be the best for only one analyzed N-list length (N=4).

Table 2 ARPD[%] values for VRF Small benchmark (the best ARPD value for a given N is bolded).

Sorting priority	N				
	1	**2**	**4**	**8**	**16**
TPT	3,845	**3,325**	2,951	2,641	2,473
TPT1	4,231	3,718	3,199	2,870	2,653
TPT2	4,041	3,510	3,111	2,740	2,566
TPT3	3,919	3,469	3,067	2,781	2,604
S	3,923	3,436	2,971	2,657	2,496
SR	3,955	3,541	3,063	2,763	2,548
SP	4,108	3,548	3,108	2,803	2,594

WSUM	4,076	3,521	3,106	2,765	2,571
ABS_DIF	4,299	3,799	3,389	3,161	2,947
WABS_DIF	4,400	3,825	3,456	3,171	2,976
SS_SRA	4,294	3,853	3,374	3,141	2,951
SS_WSRA	4,329	3,876	3,471	3,196	3,025
SS_SRS	4,291	3,797	3,455	3,193	3,024
SKE	3,842	3,356	**2,886**	**2,622**	**2,448**
SKEx	**3,835**	3,360	2,937	2,655	2,493
SKEy	3,880	3,395	2,938	2,669	2,487

For the VRF Small instance, the distinguished sorting priorities again turned out to be: TPT, SKE and SKEx. Again, the SKEx priority was the best for the NEH algorithm. Among the priorities, the SKE priority stands out, producing the best results for three of the five N-list lengths tested.

Table 3 ARPD[%] values for VRF Large benchmark (the best ARPD value for a given N is bolded).

Sorting priority	**N**				
	1	**2**	**4**	**8**	**16**
TPT	3,332	3,003	2,673	2,390	2,142
TPT1	3,470	3,140	2,830	2,525	2,287
TPT2	3,426	3,082	2,764	2,460	2,210
TPT3	3,368	3,034	2,736	2,424	2,202
S	**3,288**	**3,000**	2,689	**2,368**	2,127
SR	3,380	3,061	2,773	2,466	2,232
SP	3,795	3,457	3,081	2,782	2,493
WSUM	3,501	3,193	2,874	2,599	2,335
ABS_DIF	3,446	3,185	2,890	2,629	2,407
WABS_DIF	3,436	3,168	2,895	2,669	2,429
SS_SRA	3,384	3,132	2,851	2,606	2,388
SS_WSRA	3,448	3,184	2,900	2,649	2,427
SS_SRS	3,416	3,143	2,848	2,618	2,386
SKE	3,362	3,013	**2,634**	2,378	**2,122**
SKEx	3,313	3,016	2,669	2,375	2,156
SKEy	3,348	3,008	2,671	2,372	2,132

For the VRF Large instance, the most effective priorities were: S and SKE. For none of the N-lists was the TPT priority the best. Only for the VRF Large instance, the S priority achieved the best result among all priorities. Moreover, priority S turned out to be the most effective sorting priority for as many as three of the five analyzed N-list lengths.

The last aspect that was analyzed is the difference between the best and the worst result of all the sorting methods used. This value was referred to as “Difference” in the following section and was calculated as follows:

$$\text{Difference} = \text{Max} - \text{Min} \tag{2}$$

Table 4 summarizes the Min, Max, and Difference scores for each benchmark and N-list length analyzed.

Table 4 Summary of Min, Max and Difference (max-min) ARPD values for the analyzed sorting priority

Bench.	Value	N				
		1	2	4	8	16
Tai.	Min	3,201	2,886	2,552	2,309	2,181
	Max	4,008	3,49	3,041	2,748	2,569
	Dif.	0,807	0,604	0,489	0,439	0,388
VRF S	Min	3,835	3,325	2,886	2,622	2,448
	Max	4,4	3,876	3,471	3,196	3,025
	Dif.	0,565	0,551	0,585	0,574	0,577
VRF L	Min	3,288	3	2,634	2,368	2,122
	Max	3,795	3,457	3,081	2,782	2,493
	Dif.	0,507	0,457	0,447	0,414	0,371

For both the Taillard and VRF Large instances, it is clear that the Difference value decreases as the N-list length increases. Only for VRF Small instance, the Difference value remains at a similar level. For all instances, however, can be seen a clear decrease in the value of Min and Max. It means that regardless of the input sequence, the N-NEH+ algorithm allows for a clear improvement in the performance of the NEH algorithm. In addition, the value of Difference allows us to conclude that for most instances, it will reduce the discrepancy of results obtained by the N-NEH+ algorithm for different input-sequences.

4. CONCLUSION

In this study, the problem of the influence of input sequence on the results obtained using the N-NEH+ algorithm was analyzed. For calculations 16 sorting priorities were used. Analyses were performed using Taillard's and VRF benchmarks. Different lengths of the N-list were also considered in the analyses. The obtained results confirm that similarly to the NEH algorithm, also for the N-NEH+ algorithm the input sequence is very important for the efficiency of the algorithm. For each of the analyzed sorting priorities N-NEH+ allowed a significant improvement of the result, relative to the NEH algorithm. Both for Taillard's benchmark and VRF Large, as the length of N-list increases, we can observe a decreasing difference between the best and worst results of analyzed sorting priorities. Only in the case of VRF Small instance this difference remains at a similar level. It is also worth noting that for analyzed benchmarks, different sorting priorities allowed to obtain the best results for the NEH algorithm, but in no case was the original sorting priority of the NEH algorithm - i.e. TPT.

ACKNOWLEDGEMENTS

This study was conducted under a research project funded by a statutory grant of the AGH University of Science and Technology in Kraków for maintaining research potential.

REFERENCES

[1] GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K., KAN, A. R. Optimization and approximation in deterministic sequencing and scheduling: a survey. *In Annals of discrete mathematics*. *Elsevier.* 1979, vol. 5, pp. 287-326.

[2] GAREY, M. R., JOHNSON, D. S. *Computers and intractability*. San Francisco: freeman, 1979, vol. 174.

[3] NAWAZ, M., ENSCORE, E., HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *The International Journal of Management Science*. 1983, vol. 11, no. 1, pp. 91–95.

[4] FRAMINAN, J. M., R. LEISTEN, 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.* 2003, vol. 41, no. 1, pp. 121-148.

[5] RAJENDRAN, C. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *International Journal of Production Economics*. 1993, vol. 29, no. 1, pp. 65-73.

[6] STINSON, J. P., SMITH, A. W. A heuristic programming procedure for sequencing the static flowshop. *The International Journal Of Production Research*. 1982, vol. 20, no. 6, pp. 753-764.

[7] LIU, W., JIN, Y., PRICE, M. A new improved NEH heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics*. 2017, vol. 193, pp. 21–30.

[8] DONG, X., HUANG, H., CHEN, P. An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research*. 2008, vol. 35, no. 12, pp. 3962-3968.

[9] KALCZYNSKI, P. J., KAMBUROWSKI, J. On the NEH heuristic for minimizing the makespan in permutation flow shops. *Omega*. 2007, vol. 35, no. 1, pp. 53-60.

[10] KALCZYNSKI, P. J., KAMBUROWSKI, J. An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers & Operations Research*. 2008, vol. 35, no. 9, pp. 3001-3008.

[11] NAGANO, M. S., MOCCELLIN, J. V. A high quality solution constructive heuristic for flow shop sequencing. *Journal of the Operational Research Society*. 2002, vol. 53, no. 12, pp. 1374-1379.

[12] PUKA, R., DUDA, J., STAWOWY, A. Input Sequence of Jobs on NEH Algorithm for PermutationFlowshop Scheduling Problem. *Management and Production Engineering Review*. 2022, vol. 1, no. 13, pp. 32-43, Available from: https://doi.org/10.24425/mper.2022.140874.

[13] PUKA, R., DUDA, J., STAWOWY, A., SKALNA, I. N-NEH+ algorithm for solving permutation flow shop problems. *Computers & Operations Research*. 2021, vol. 132, 105296.

[14] TAILLARD, E. Benchmarks for basic scheduling problems. *European Journal of Operational Research*. 1993, vol. 64, no. 2, pp. 278–285. Project Management anf Scheduling.

[15] VALLADA, E., RUIZ, R., FRAMINAN, J. New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*. 2015, vol. 240, no. 3, pp. 666–677.