USING REVERSIBILITY PROPERTY TO SOLVE PERMUTATION FLOW SHOP SCHEDULING PROBLEM BY MEANS OF ALGORITHMS IMPLEMENTING N-LIST TECHNIQUE

[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.2022.4540

Abstract

One of the most popular algorithms for solving the permutation flow shop scheduling problem (PFSP) for the makespan minimization criterion is the NEH algorithm. Many research papers have been devoted to the NEH algorithm and its improvements. One of the most interesting improvements is the N-list technique, which can be implemented to other algorithms based on the NEH algorithm. This paper is devoted to analyzing the effect of the reversibility property on the results obtained by the selected algorithms. The analyses were performed on three example algorithms for which the N-list technique was implemented and which used the reversibility property. The analyses were performed using the two most popular benchmarks for the PFSP problem: the Taillard's benchmark and the VRF benchmark. The results clearly indicate that the use of the reversibility property can significantly improve the results obtained by the algorithm.

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

1. INTRODUCTION

The permutation flow shop scheduling problem (PFSP) is one of the cases of flow shop scheduling problem (FSP). The PFSP can be defined as a problem of finding the sequence of production, that minimizes the selected goal function (e.g. makespan, tardiness [1]). In the PFSP problem, production of each from n jobs is realized on m machines, always on the same order. In this paper, the makespan was chosen as the optimization criterion. The makespan (C_{max}) is defined as the time of finish production of the last job on the last machine. The permutation flow shop problem with makespan as a minimization criterion is denoted as $F_m|prmu|C_{max}$ [1].

The PFSP problem is NP-hard for the number of machines greater than 2 [2], and therefore to solve the problem heuristics and metaheuristics algorithms are used. One of the most popular constructive heuristic algorithm for solving the PFSP problem is NEH algorithm [3]. The NEH algorithm 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.

Since its publication, the NEH algorithm has become the focus of much research. Many modifications of the NEH algorithm retained the constructive and deterministic nature. The most important improvements of the NEH can be divided into two classes:

1) Changing the sorting criteria in the initial order (first step of the NEH algorithm).

2) Implementing the tie-breaking mechanism for a partial sequence with the same makespan (third step of the NEH algorithm).

Of course, some algorithms implement both of the modifications (e.g. [4]). Both classes of improvements are derived from the ambiguity of the construction of the NEH algorithm e.g. in the case of equal the total processing time of jobs (the importance of this aspect has been analyzed in the paper [5]). Another improvement to the NEH algorithm is to use the reversibility property of the PFSP problem [6]. According to the reversibility property, the solution obtained using the modified duration time of each job (see Equation 1) is the inverse of the solution of the base problem. Importantly, the solutions (base and inverse permutations) have the same C_{max} value.

$$d'_{j,i} = d_{m-j+1,i} \quad j = 1, \dots, m \quad i = 1, \dots, n \tag{1}$$

where $d_{j,i}$ is the duration time *i*-th job on the j-th machine, $d'_{j,i}$ - duration time for the inverse solution.

Another valuable approach is presented in the paper [7]. The authors proposed a new technique selection of candidate jobs called the N-list technique. The N-list contains the list of candidate jobs that can be scheduled in the third step of the NEH algorithm, taking into account the optimization criterion (C_{max} in this case). The job with a minimum makespan value is added to the partial sequence and removed from the N-list. If possible (not all jobs have been scheduled or are in the N-list), the N-list is completed with the next job from the sorted list of all jobs.

In this paper, the authors analyze the impact of using the reversibility property in the algorithm implementing the N-list technique. To make the analysis more reliable, have been chosen algorithms that represent the most important improvements of NEH algorithm. The authors analyze three algorithms: N-NEH, N-NEHSKK (implementing own sorting criteria), and N-NEHKK (with the tie-breaking mechanism).

2. ALGORITHMS IMPLEMENTS N-LIST

The N-NEH algorithm is the NEH algorithm that implements N-list. The N-list may have various lengths and therefore it is necessary to specify which length has been used in the given case. For example, the notation N-NEH(4) denotes the N-list with a length of four. The plus sign after the algorithm name (e.g. N-NEH+(4)) denotes, that the result of the algorithm was obtained using the N-list of length (1, ..., N) and selecting the result with the minimum makespan value.

The next selected algorithm is the NEHSKE algorithm based on the proposition Liu et al. [4]. The algorithm uses a more advanced (than the NEH algorithm) sorting jobs strategy in the initial phase – all jobs are sorted by the non-increasing sum:

$$AVG_i + STD_i + abs(SKE_i) \tag{2}$$

where AVG_i denotes the average processing times of *i*-th job, STD_i – the standard deviation of i-th job, abs(SKE_i) - value of the skewness of i-th job.

An example of an algorithm that implements a tie-breaking mechanism is the NEHKK algorithm [8]. The tie-breaking mechanism is only used when the minimum value of makespan has been achieved for more than one sequence. Kalczynski and Kamburowski [8] propose that when the makespan is equal for several sequences, the sequence with minimum makespan for the subsequence needs to be selected.

The reversibility property has been implemented in all presented algorithms (N-NEH, N-NEHSKE, N-NEHKK). In the results, the letter „r" was added to the algorithm names in which the reversibility property was used (e.g. rNEH). In the „r" algorithms both the base and the inverse permutations were calculated, and the permutation with a lower makespan value was retained.

The comparison of the performance of algorithms is described in the next section.

3. COMPUTATIONAL EXPERIMENTS

Two benchmarks were selected to compare the effectiveness of the algorithms: Taillard's benchmark (with 120 instances) [9], and VRF benchmark (with 480 instances divided into two sets: Small and Large) [10]. The algorithms were implemented in C#. Computations were performed on a computer with Intel i5-8265U CPU (4 cores, each with 1.6 GHz base clock speed). To reduce the time complexity, Taillard's acceleration [11] was implemented. The measure of solution quality for a given instance (*ic*) is the relative percentage deviation (RPD) calculated as:

$$RPD_{alg,i} = \frac{M_{alg,ic} - M_{best,ic}}{M_{best,ic}} \quad (3)$$

where *alg* indicates the algorithm used, $M_{alg,ic}$ is the makespan value of the algorithm *alg* for instance *ic* and $M_{best,ic}$ is the best known value of makespan for instance *ic*.

For the benchmarks that contain multiple instances, the average relative percentage deviation (ARPD) measure is used:

$$ARPD_{alg} = \frac{1}{I}\sum_{ic=1}^{I} RPD_{alg,ic} \quad (4)$$

where *I* is the number of instances in the benchmark.

The second type of measure that can be used for the quality of the solution is the average CPU time (ACPU) computed as follows:

$$ACPU_{alg} = \frac{1}{I}\sum_{ic=1}^{I} CPU_{alg,ic} \quad (5)$$

where $CPU_{alg,ic}$ is the CPU time of algorithm *alg* on instance *ic*.

The results achieved by the analyzed algorithms for the measures ARPD and ACPU are presented in **Tables 1 and 2**. The results are divided in terms of the benchmark for which they were achieved, as well as the length of the N-list. Five N-list lengths were analyzed: 1, 2, 4, 8, 16.

Table 1 ARPD[%] for benchmarks: Taillard's, VRF Small and VRF Large

Benchmark	N	N-Neh	N-rNeh	N-NEHSKE	N-rNEHSKE	N-NEHKK	N-rNEHKK
Taillard's	1	3.33	3.03	3.37	2.99	3.08	2.97
	2	2.95	2.67	2.94	2.63	2.71	2.61
	4	2.55	2.39	2.57	2.34	2.49	2.40
	8	2.32	2.17	2.34	2.11	2.23	2.15
	16	2.20	2.06	2.18	2.00	2.09	2.04
VRF Small	1	3.84	3.46	3.84	3.42	3.67	3.58
	2	3.32	2.99	3.36	3.00	3.16	3.09
	4	2.95	2.68	2.89	2.61	2.82	2.75
	8	2.64	2.43	2.62	2.37	2.57	2.52
	16	2.47	2.28	2.45	2.20	2.40	2.35
VRF Large	1	3.33	3.18	3.36	3.18	3.22	3.10
	2	3.00	2.88	3.01	2.85	2.92	2.80
	4	2.67	2.57	2.63	2.54	2.59	2.52
	8	2.39	2.28	2.38	2.30	2.32	2.25
	16	2.14	2.05	2.12	2.06	2.10	2.04

Based on the results presented in **Table 1**, there is a clear difference between algorithms that use reversibility property and those that do not**. Figure 1** shows the percentage improvement ratio of algorithms implementing reversibility property to those not using this method.

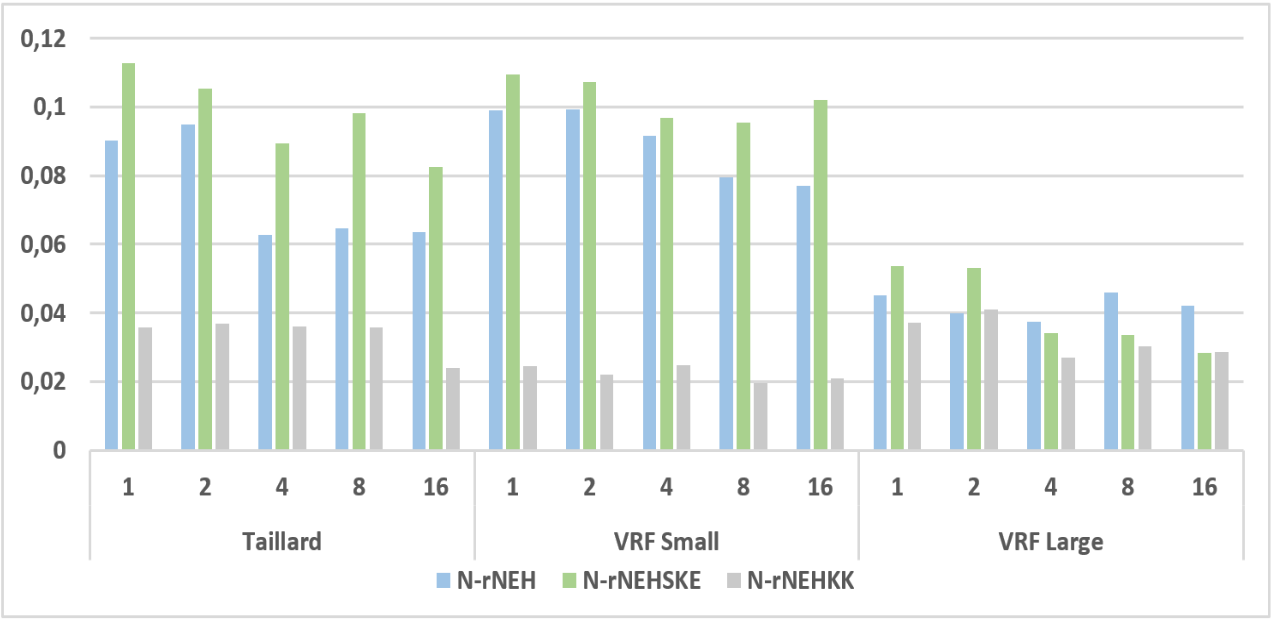

Figure 1 The percentage improved ARPD value by use of reversibility property

As can be seen in **Figure 1**, using the reversibility property method allows for a significant improvement in the score. The amount of improvement depends on:

- the algorithm: much larger improvements can be observed for the N-rNEH AND N-rNEHSKE algorithms than for the N-rNEHKK algorithm,
- the benchmark used to analyze the algorithms: the least improvement can be observed for the VRF Large benchmark,
- length of the N-list: it should be noted, however, that in this case, it is not possible to indicate a clear trend of change.

The small improvement values for the N-rNEHKK algorithm are noteworthy. In part, this may be due to the low ARPD value for the N-NEHKK algorithm, making it relatively more difficult to improve the result. Certainly, the fact that this algorithm uses a tie-breaking mechanism, which is a kind of local optimization, is also important.

The ACPU (**Table 2**) values show that using the reversibility property is associated with virtually doubling the execution time of the algorithm. At the same time, doubling the length of the N-list more than doubles the time - the longer the N-list, the higher the multiplicity of time increase for doubling the N-list. Undoubtedly, an important aspect in the analysis of computation time is the available time, i.e., in the absence of a time constraint, it is more advantageous to use the reversibility property method.

The results presented in **Tables 1 and 2** are compared with the results of selected deterministic heuristic algorithms for the Taillard's benchmark (**Table 3**) [12]. Algorithms using the N-list method produce some of the best results in this set. The only algorithms that obtain better results are the FRB algorithms. [13]. However, they are not constructive algorithms.

Table 2 ACPU[ms] for benchmarks: Taillard's, VRF Small and VRF Large

Benchmark	N	N-Neh	N-rNeh	N-NEHSKE	N-rNEHSKE	N-NEHKK	N-rNEHKK
Taillard's	1	21.9	44.2	22.3	44.4	36.5	70.7
	2	52.9	106.7	53.6	106.7	105	206.4
	4	141.4	285.7	143.3	285.2	280.6	555
	8	424.9	856.3	430.2	854.7	769.6	1644.1
	16	1409.2	2833.1	1424.7	2831.4	2162.4	4450.5
VRF Small	1	0.8	1.6	0.8	1.6	1	1.9
	2	1.9	4	1.9	3.9	2.4	4.9
	4	5.2	10.6	5.3	10.5	7	13.8
	8	15.3	30.8	15.4	30.7	19.7	39
	16	46.4	93.8	46.4	92.8	56.9	112.7
VRF Large	1	1158.6	2159.5	1172.6	2200.1	1588.4	3020.4
	2	2646.5	5080.7	2646.5	5158.2	3694.1	7327.6
	4	6644.6	13132.6	7002.7	13412.2	10170.2	19825.3
	8	19180.7	38303.6	19906.3	38906.2	27481.9	54351.6
	16	63569.3	128062.5	63251.5	125118.8	86096.1	167857.9

Table 3 ARPD and ACPU values of different heuristics for Taillard's benchmark

Algorithm	ARPD	ACPU	Algorithm	ARPD	ACPU	Algorithm	ARPD	ACPU
rRAER	3,53	159,7	rKKER	2,86	150	N-NEH+(16)	2,20	1409,2
N-NEHSKE	3,37	22,3	rNEHR	2,85	157,8	N-NEHSKE+(16)	2,18	1424,7
N-NEH	3,33	21,9	rNEHD	2,84	198,1	N-rNEH+(8)	2,17	856,3
NEMR	3,16	123,8	N-NEHKK+(2)	2,71	105	N-rNEHKK+(8)	2,15	1644,1
KKER	3,15	73	N-rNEH+(2)	2,67	106,7	FRB4_4	2,13	218,6
rNEHKK1	3,15	44,6	N-rNEHSKE+(2)	2,63	106,7	N-rNEHSKE+(8)	2,11	854,7
NEHKK2	3,09	22,6	N-rNEHKK+(2)	2,61	206,4	N-NEHKK+(16)	2,09	2162,4
N-NEHKK	3,08	36,5	N-NEHSKE+(4)	2,57	143,3	N-rNEH+(16)	2,06	2833,1
NEHR	3,05	76,7	N-NEH+(4)	2,55	141,4	N-rNEHKK+(16)	2,04	4450,5
N-rNEH	3,03	44,2	N-NEHKK+(4)	2,49	280,6	N-rNEHSKE+(16)	2,00	2831,4
CL_WTS	3,02	1038,1	N-rNEHKK+(4)	2,40	555	FRB4_8	1,95	368,4
N-rNEHSKE	2,99	44,4	N-rNEH+(4)	2,39	285,7	FRB2	1,93	771,4
N-rNEHKK	2,97	70,7	N-NEHSKE+(8)	2,34	430,2	FRB4_6	1,91	294,6
rNEMR	2,97	222,6	N-rNEHSKE+(4)	2,34	285,2	FRB4_10	1,87	441
N-NEH+(2)	2,95	52,9	FRB4_2	2,33	135,4	FRB4_12	1,79	507
N-NEHSKE+(2)	2,94	53,6	N-NEH+(8)	2,32	424,9	FRB3	1,61	6105,9
NEHFF	2,90	24,2	N-NEHKK+(8)	2,23	769,6	FRB5	1,48	17545,7

4. CONCLUSION

This paper has analyzed the possibility of using reversibility property in three sample algorithms implementing the N-list technique. On the basis of conducted analysis using the most popular benchmarks for PFSP problem, it was shown that using reversibility property improves the results achieved by algorithms using N-list method. The use of the reversibility property results in nearly doubling the execution time of the algorithms. Therefore, as with most heuristics, there is a trade-off between the level of results (ARPD) and their computation time (ACPU).

From the results obtained and analyses performed, the N-rNEHSKE algorithm seems to be the most promising algorithm, which for most benchmarks and different N-list lengths achieved the best or very close to the best results in terms of ARPD values. It should also be noted that over the second best performing algorithm, N-rNEHKK shows a definite advantage in terms of significantly shorter ACPU time.

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*. 1979, vol. 5, pp. 287-326. Elsevier.

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

[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] 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.

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

[6] RIBAS, I., COMPANYS, R., TORT-MARTORELL, X. Comparing three-step heuristics for the permutation flow shop problem. *Computers & Operations Research*. 2010, vol. 37, no. 12, pp. 2062-2070.

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

[8] 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.

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

[10] VALLADA, E., RUIZ, R., FRAMINAN, J. New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*. 2015, 240, vol. 3, pp. 666–677.

[11] TAILLARD, E. Some efficient heuristic methods for the flow shop sequencing problem. European Journal of Operational Research. 1990, vol. 47, no. 1, pp. 65-74.

[12] FERNANDEZ-VIAGAS, V., RUIZ, R., FRAMINAN, J. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research*. 2017, vol. 257, no. 3, pp.707–721.

[13] RAD, S.F., RUIZ, R., BOROOJERDIAN, N. New high performing heuristics for minimizing makespan in permutation flowshops. *Omega*. 2009, vol. 37, no. 2, pp. 331-345.