Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 1107–1111
DOI: 10.15439/2023F7072
ISSN 2300-5963 ACSIS, Vol. 35

New measures of algorithms quality for permutation flow-shop scheduling problem

Radosław Puka
AGH University,
ul. Gramatyka 10, 30-067 Kraków, Poland
Email: rpuka@agh.edu.pl

Iwona Skalna, Tomasz Derlecki
AGH University,
ul. Gramatyka 10, 30-067 Kraków, Poland
Email: {skalna, derlecki}@agh.edu.pl

***Abstract*—The permutation flow-shop scheduling problem (PFSP) is an important problem in production industry. The problem has been a subject of many research and various algorithms to solve PFSP have been developed over the years. The newly developed algorithms are usually tested on Taillard and VRF benchmarks and their results are compared using various measures that assess the size of error made by an algorithm and the computation time. In this paper, we propose two new measures to assess the quality of results of algorithms for solving PFSP with the makespan criterion. The first ARD.NEH measure gives similar results as the well known ARPD measure but is robust to updates of the best known solutions of benchmark problems. The second ARID measure is an interval-based measure which is able to assess whether the good quality of an algorithm results stems from its good behavior of this algorithm for a few instances or from its good behavior for most instances. The computational experiments confirm the usefulness of the proposed quality measures.**

I. Introduction

THE permutation flow-shop scheduling problem (PFSP) is one of the most studied combinatorial optimization problems, rooted in the manufacturing industry. It can be defined as follows: given a finite set of m machines $\{M_1, \ldots, M_m\}$ and a finite set of n jobs $\{J_1, \ldots, J_n\}$, each of which should go through all the m machines in the same order, the goal is order the jobs so as to minimize the assumed optimization criterion (e.g., makespan, total tardiness, flow time, cost, energy consumption).

The PFSP with makespan criterion, commonly referred to as $Fm|prmu|C_{\max}$ [1], is undoubtedly the most frequently investigated scheduling problem. Garey and Johnson [2] proved that $Fm|prmu|C_{\max}$ is NP-hard if $m \geqslant 3$. Therefore, various heuristics have been developed to solve this problem in a reasonable amount of time. Among them, the Navaz, Enscore and Ham (NEH) construction heuristic [3] plays an important role; for a long time NEH has been regarded as the best heuristic for solving $Fm|prmu|C_{\max}$.

Since optimal solutions are generally not known for some instances, the only way to asses the results of new methods is to compare them with the best solutions known so far. The well-known measure of solution quality, initially referred to as the increase over optimum (IOO) [4] and later as the relative percentage deviation (RPD) [5], is defined as:

$$\mathrm{RPD} = \frac{S - Best}{Best} \times 100\%, \tag{1}$$

where S is the solution of the evaluated algorithm and $Best$ is the best solution known so far for a given instance of the problem. For a group of instances, a synthetic solution quality measure, called the average relative percentage deviation (ARPD), is calculated as:

$$\mathrm{ARPD} = \frac{1}{I}\sum_{i=1}^{I}\frac{S_i - Best_i}{Best_i}, \tag{2}$$

where I is the number of instances, S_i is the solution of the evaluated algorithm on the instance i of a given size, and $Best_i$ is the best solution known so far for this instance.

The quality of solutions is obviously not the only aspect of algorithms evaluation – the running time is also an important feature (we often face the trade-off between the quality of results and computational time). Literature research shows that the computational time is often reported in time units (usually in milliseconds) [6], sometimes, especially in case of simpler algorithms, the computational complexity is provided. Given several algorithms to be compared and various instances, the computational effort is usually measured by using the average CPU time (ACPU) computed as follows:

$$\mathrm{ACPU}_j = \frac{1}{I}\sum_{i=1}^{I}\mathrm{CPU}_{i,j}, \tag{3}$$

where $\mathrm{CPU}_{i,j}$ is the CPU time consumed by algorithm j on instance i. However, the running time scheduling algorithms strongly depends on the size of the problem instance, therefore Fernandez-Viagas and Framinan [7] proposed to measure the average relative percentage time (ARPT) consumed by algorithm j:

$$\mathrm{ARPT}'_j = \frac{1}{I}\sum_{i=1}^{I}\mathrm{RPT}_{i,j}, \tag{4}$$

where $\mathrm{RPT}_{i,j}$ (relative percentage computation time of algorithm j for instance i) is computed as:

$$\mathrm{RPT}_{i,j} = \frac{\mathrm{CPU}_{i,j} - \mathrm{ACT}_i}{\mathrm{ACT}_i}, \tag{5}$$

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

and ACT_i (average computational time for instance i) is computed as:

$$\mathrm{ACT}_i = \frac{\sum_{j=1}^{J} \mathrm{CPU}_{i,j}}{J}. \tag{6}$$

Since ARPT'_j can yield negative values ($\mathrm{ARPT}'_j > -1$), Fernandez-Viagas and Framinan [8] proposed to compute $\mathrm{ARPT} = \mathrm{ARPT}' + 1$, which allows the graphics to be shown in logarithmic scale.

The above described features of ACPU and ARPT make these two measure not very authoritative and quite cumbersome in practice. In [9], we have proposed the ART.NEH (the Average Relative Time over NEH) indicator defined by the following formula:

$$\mathrm{ART.NEH} = \frac{\sum_{i=1}^{I} \frac{\mathrm{CPU}_i}{\mathrm{CPU}_{i,\mathrm{NEH}}}}{I}, \tag{7}$$

where I is the number of considered instances, CPU_i is the CPU time of a considered algorithm for the instance i, and $\mathrm{CPU}_{i,\mathrm{NEH}}$ is the CPU time of NEH for the instance i. ART.NEH indicates how many times, on average, the evaluated algorithm is faster (ART.NEH$<$1) or slower (ART.NEH$>$1) than the classical NEH. Following this idea, we propose in Section II several new measures to compare the quality of results produced by algorithms for solving PFSP with the makespan criterion. Numerical experiments showing the usefulness of the proposed measures are described in Section III. The paper ends with concluding remarks.

II. New measures of algorithms efficiency

New algorithms are expected to be better than existing ones, but a fair comparison of algorithms is quite difficult (due to implementation issues and hardware used). However, most of papers on solving PFSP with the makespan criterion provide the results produced by NEH. So, it seems quite natural to use this well-known heuristic as a computational benchmark.

The ARPD indicator given by formula (2) is by far the most popular measure for assessing the quality of scheduling algorithms taking into account the size of the error. It has, however, some drawbacks which led to the development of alternative measures. An important drawback, we want to emphasize, is that the value of ARPD can change when new better solutions are found for an analyzed instance. In this regard, the ARPD value of an algorithm can change significantly over the years. A good example can be the most known PFSP benchmark – Taillard's benchmark [10] published in 1993. Though it is now 30 years since its publication, better solutions are still found for various instances [11]. Thus, since the ARPD factors change, the whole measure change as well. In that case, it is difficult to compare the new results with existing (published) ones due to different reference values (the results of such a comparison may not be reliable). To get rid of this drawback, in this paper we propose a new measure ARD.NEH (Average Relative Deviation over NEH) which will not change in time thanks to the use of the NEH results as reference results. The proposed ARD.NEH measure is computed from the following formula:

$$\mathrm{ARD.NEH} = \frac{1}{I} \sum_{i=1}^{I} \frac{\mathrm{NEH}_i - S_i}{\mathrm{NEH}_i}, \tag{8}$$

where I is the number of instances, S_i is the solution of the evaluated algorithm on the instance i, and NEH_i is the solution obtained using the NEH algorithm for this instance.

The main reason for developing this measure was to make it easier to compare the results produced by new algorithms with the results available in the literature. The advantage of ARD.NEH over ARPD is that it does not change over time. This particular feature of ARD.NEH is due to the fact that ARD.NEH does not depend on the best solutions known so far, but on the results of NEH. So, the measure is especially useful to deal with those problems for which the optimal solution is not know yet. Since ARD.NEH indicates how far the results of an algorithm are from the results of NEH, the greater is ARD.NEH the better.

Another new measure to assess the quality of the results, we propose in this paper, is the $\mathrm{ARID}(\inf, \sup)$ (Average Relative Interval Deviation) measure. $\mathrm{ARID}(\inf, \sup)$ is different from existing quality measures in that it is based on the interval $[\inf, \sup]$ (it is assumed that the interval $[\inf, \sup]$ can be improper) instead of a single value (reference point). By taking different intervals, we can obtain various quality measures. The concept behind this measure is to equalize the impact of each benchmark instance on the final value of the evaluation measure. The value of $\mathrm{ARID}(\inf, \sup)$ is computed from the following formula:

$$\mathrm{ARID}(\inf, \sup) = \frac{1}{I} \sum_{i=1}^{I} \frac{\max(\inf, \sup) - S_i}{\sup - \inf} \tag{9}$$

where S_i is the solution for the instance i.

Proposition 1: ARPD and ARD.NEH measures are a special case of the ARID measure.

Proof: Let I be the set of instances, and $Best_i$, NEH_i, and S_i the best known solution, the solution produced by NEH, and the solution for the instance i, respectively. It holds that

$$\mathrm{ARID}(\mathit{Best}, 0) = \frac{1}{I} \sum_{i=1}^{I} \frac{\max(Best_i, 0) - S_i}{0 - Best_i} =$$

$$= \frac{1}{I} \sum_{i=1}^{I} \frac{Best_i - S_i}{-Best_i} = \frac{1}{I} \sum_{i=1}^{I} \frac{S_i - Best_i}{Best_i} = \mathrm{ARPD}$$

$$\mathrm{ARID}(0, \mathrm{NEH}) = \frac{1}{I} \sum_{i=1}^{I} \frac{\max(\mathrm{NEH}_i, 0) - S_i}{\mathrm{NEH}_i - 0} =$$

$$= \frac{1}{I} \sum_{i=1}^{I} \frac{\mathrm{NEH}_i - S_i}{\mathrm{NEH}_i} = \mathrm{ARD.NEH}$$

■

In what follows, we set $\inf$ = *Best*, $\sup$ = NEH, where *Best* means that we use the best solutions (makespans) known so far for benchmark instances, and NEH means that we use

the solutions produced by NEH for the respective instances. Then, ARID(*Best*,NEH) (further referred to as simply ARID), similarly as ARPD, uses the best know solutions, so ARID is recommended to be used for problems with $Best = Opt$. ARID allows to equalize the impact of different instances on the final result. For example, the ARPD value for Taillard benchmark is the most influenced by the instances having the best solutions far from the optimum and the less influenced by the instances having the best solution close to the optimum. Making each instance to have comparable impact on the final evaluation of an algorithm, allows to compare different algorithms in terms of the stability of their results in relation to the dynamically determined value, which in this case is the result of NEH. The result of NEH can therefore be considered as a kind of assessment of the difficulty of a given instance. The stability of an algorithm should be understood here as a possibility to obtain better results than NEH for as many instances as possible. Let us note that the value of ARID, similarly as the value of ARD.NEH, should be maximized. The next section presents the experiments that aim to show the usefulness of the proposed measure of the algorithms quality.

III. Computational experiment

The measures proposed in Section II were used to assess the results of various algorithms for Taillard benchmark [10] and VRF Large benchmark instances [12]. Best solutions provided by the authors of the benchmarks are updated with the recent results presented in [11] (Taillard) and [13] (VRF Large).

Tables I and II show the values of the ARPD, ARD.NEH and ARID measures obtained for, respectively, Taillard and VRF Large benchmarks by using selected deterministic algorithms for solving PFSP (cf., [14], [15], [16], [17], [18], [7], [8], [19], [20], [21], [22], [9]). As can be seen from the tables, only two algorithms (RAER and RAER-di) achieved negative values of ARD.NEH measure, which means that their average results were worse than the average result of NEH. It can be seen as well that only FRB and N-list technique-based algorithms (the latter will be further referred to as N-algorithms) achieved the results that are better than NEH results by more than 1 percent, for both benchmarks. As for the ARID measure, only FRB algorithms and N-algorithms achieved the values greater than 15%. Moreover, only 3 algorithms (for Taillard benchmark) and 2 algorithms (for VRF Large benchmark instances) achieved the results greater than 50%, which means that only 3 algorithms were able to improve the results of NEH by, on average, more than a half distance between the best solution produced by NEH and the best solution known so far for a given instance.

TABLE I
ARPD, ARD.NEH and ARID values for Taillard benchmark

Algorithm	ARPD	ARD.NEH	ARID
RAER	3.94	-0.56	-56.99
RAER-di	3.57	-0.20	-40.97
NEH	3.37	0.00	0.00
NEMR	3.21	0.15	-6.91
NEHKK1-di	3.20	0.17	-0.16
KKER	3.19	0.17	-3.25
NEH1-di	3.15	0.21	4.73
NEHKK2	3.14	0.22	7.54
NEHR	3.10	0.26	2.52
NEH-di	3.08	0.28	9.37
vN-NEH+(2)	3.02	0.34	9.69
NEMR-di	3.01	0.34	4.81
N-NEH+(2)	2.99	0.36	10.27
NEHFF	2.95	0.41	1.41
KKER-di	2.91	0.44	13.87
NEHR-di	2.90	0.46	13.48
NEHD-di	2.88	0.47	6.00
vN-NEH+(3)	2.82	0.52	15.83
N-NEH+(3)	2.74	0.60	18.64
SP+(0.3)N+(2)	2.70	0.64	18.25
vN-NEH+(4)	2.67	0.67	20.42
N-NEH+(4)	2.60	0.74	22.09
$FRB4_2$	2.37	0.95	31.04
N-NEH+(8)	2.36	0.96	29.48
vN-NEH+(8)	2.28	1.04	32.77
SP+(0.3)N+(4)	2.27	1.05	32.85
SMα+(8)N+(2)	2.26	1.06	35.14
N-NEH+(16)	2.24	1.08	33.15
$FRB4_4$	2.17	1.15	34.93
vN-NEH+(16)	2.07	1.25	42.25
SP+(0.3)N+(8)	2.03	1.29	40.02
SMα+(8)N+(4)	2.01	1.31	43.51
$FRB4_8$	1.99	1.32	40.22
FRB2	1.98	1.33	32.81
$FRB4_6$	1.96	1.35	40.88
$FRB4_{10}$	1.92	1.39	42.49
SP+(0.3)N+(16)	1.89	1.41	44.51
SMα+(8)N+(8)	1.86	1.44	48.47
$FRB4_{12}$	1.84	1.46	45.01
SMα+(8)N+(16)	1.75	1.55	51.70
FRB3	1.66	1.64	50.06
FRB5	1.53	1.77	55.79

Figures 1 and 2 show the rank (y-axis) of each algorithm with respect to the specific quality measure. As we can see, the ranks with respect to ARD.NEH and ARPD coincide for all algorithms. This means that the ARPD measure can be successfully replaced with the ARD.NEH measure. If we take a look at the ARID measure, we can see that this measure ranks the algorithms in a different manner than the other two measures. Those algorithms that are ranked below the line determined by the ARPD and ARD.NEH measures can be considered as more stable. The results produced by these algorithms are less due to the fact of significant improvements of NEH results for single instances, and more due to improvements of NEH results for more instances. Due to the design of the measure, large improvements for single instances are less promoted than frequent but less significant improvements. Hence the deterioration of the results of individual algorithms,

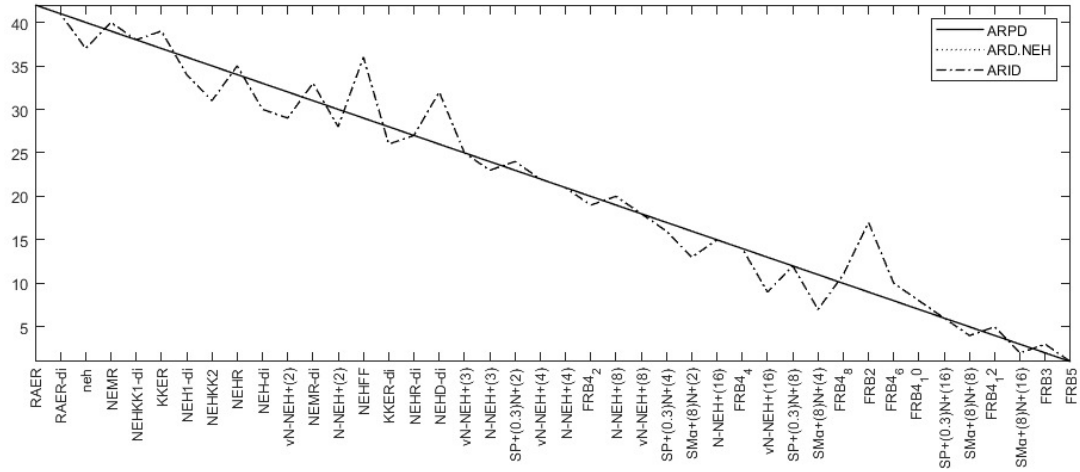

Fig. 1. Ranking of algorithms based on ARPD, ARD.NEH and ARID values for Taillard benchmark

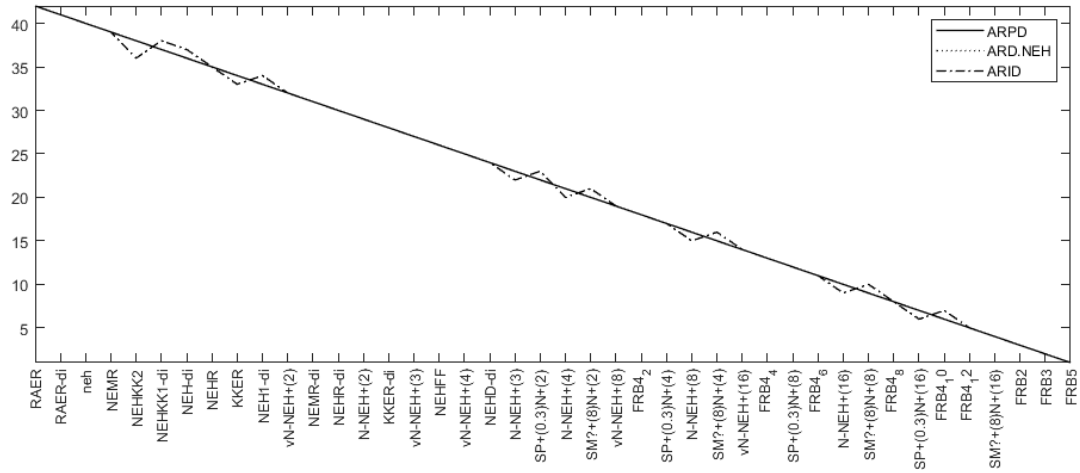

Fig. 2. Ranking of algorithms based on ARPD, ARD.NEH and ARID values for VRF Large instances

for which the position for the ARID measure deviates upwards from the line ARPD/ART.NEH.

IV. Conclusion

This work proposes two new measures for assessing the quality of results produced by algorithms for solving permutation flow-shop problems with the makespan criterion. The first ARD.NEH measure has the very useful feature of elimination of the dependency of the quality assessment from the best known results which, as shown by the performed analysis, change over time, and therefore the comparison of new results with the older one might be cumbersome. The second ARID measure is to our best knowledge the first interval-based measure. It is worth to underline that the ARID measure with properly selected intervals is equivalent to ARPD or ARD.NEH measures. The proposed new measure have been tested on 42 selected deterministic algorithms for solving PFSP run on Taillard and VRF Large benchmarks. Based on the obtained results it can be concluded that ARPD and ARD.NEH measures coincide, i.e., they rank the algorithms in a very similar manner. The ARID measure, in turn, is useful in assessing the stability of the algorithms, i.e., it indicates whether a good (average) quality of results stems from good results for a few instances of from good results for most instances. The numerical experiments show that the proposed measures are very useful for more reliable comparison of algorithms for solving PSFP with the makespan criterion.

References

[1] R. Graham, E. Lawler, J. Lenstra, and A. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," in *Discrete Optimization II*, ser. Annals of Discrete Mathematics, P. Hammer, E. Johnson, and B. Korte, Eds. Elsevier, 1979, vol. 5, pp. 287–326. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016750600870356X

TABLE II
ARPD, ARD.NEH AND ARID VALUES FOR VRF LARGE INSTANCES

Algorithm	**ARPD**	**ARD.NEH**	**ARID**
RAER	3.54	-0.13	-5.29
RAER-di	3.41	-0.01	-1.37
NEH	3.41	0.00	0.00
NEMR	3.30	0.10	3.00
NEHKK2	3.29	0.12	4.07
NEHKK1-di	3.27	0.13	3.70
NEH-di	3.27	0.14	3.78
NEHR	3.24	0.16	4.86
KKER	3.23	0.17	5.44
NEH1-di	3.23	0.17	5.09
vN-NEH+(2)	3.21	0.19	5.65
NEMR-di	3.14	0.26	8.00
NEHR-di	3.10	0.29	9.15
N-NEH+(2)	3.08	0.32	9.80
KKER-di	3.08	0.32	9.92
vN-NEH+(3)	3.07	0.33	9.93
NEHFF	3.03	0.37	12.33
vN-NEH+(4)	2.96	0.44	12.82
NEHD-di	2.94	0.45	14.88
N-NEH+(3)	2.88	0.51	15.79
SP+(0.3)N+(2)	2.88	0.52	15.59
N-NEH+(4)	2.75	0.64	19.82
SMα+(8)N+(2)	2.74	0.65	19.43
vN-NEH+(8)	2.68	0.71	21.50
$FRB4_2$	2.65	0.73	22.35
SP+(0.3)N+(4)	2.57	0.82	25.03
N-NEH+(8)	2.47	0.91	28.54
SMα+(8)N+(4)	2.47	0.91	27.92
vN-NEH+(16)	2.40	0.98	29.99
$FRB4_4$	2.39	0.98	30.27
SP+(0.3)N+(8)	2.30	1.07	33.42
$FRB4_6$	2.25	1.12	34.45
N-NEH+(16)	2.22	1.15	36.09
SMα+(8)N+(8)	2.22	1.15	35.76
$FRB4_8$	2.15	1.22	37.65
SP+(0.3)N+(16)	2.06	1.31	40.77
$FRB4_{10}$	2.05	1.31	40.28
$FRB4_{12}$	2.02	1.34	41.15
SMα+(8)N+(16)	2.02	1.34	42.08
FRB2	1.82	1.53	47.38
FRB3	1.40	1.94	59.77
FRB5	1.12	2.21	68.37

[2] M. Garey and D. Johnson, *Computers and intractability*. San Francisco: W.H. Freeman, 1979, vol. 174.

[3] M. Nawaz, 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0305048383900889

[4] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research*, vol. 165, no. 2, pp. 479–494, 2005.

[5] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Computers & Operations Research*, vol. 37, pp. 754–768, 04 2010.

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

[7] V. Fernandez-Viagas and J. Framinan, "On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem," *Computers & Operations Research*, vol. 45, pp. 60–67, 2014.

[8] V. Fernandez-Viagas, R. Ruiz, and J. Framinan, "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*, vol. 257, no. 3, pp. 707–721, 2017.

[9] R. Puka, I. Skalna, J. Duda, and A. Stawowy, "*vN*-neh+ algorithm with modified n-list technique to solve the permutation flow shop problem with makespan criterion," 2022, http://dx.doi.org/10.2139/ssrn.4239708.

[10] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993, project Management anf Scheduling. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037722179390182M

[11] J. Gmys, "Exactly solving hard permutation flowshop scheduling problems on peta-scale gpu-accelerated supercomputers," *INFORMS Journal on Computing*, vol. 34, no. 5, pp. 2502–2522, 2022.

[12] E. Vallada, R. Ruiz, and J. Framinan, "New hard benchmark for flowshop scheduling problems minimising makespan," *European Journal of Operational Research*, vol. 240, no. 3, pp. 666–677, 2015.

[13] J.Gmys, M. Mezmaz, N. Melab, and D. Tuyttens, "A computationally efficient branch-and-bound algorithm for the permutation flow-shop scheduling problem," *European Journal of Operational Research*, vol. 284, no. 3, pp. 814–833, 2020.

[14] 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, part Special Issue: Telecommunications Network Engineering.

[15] P. Kalczynski and J. Kamburowski, "An improved NEH heuristic to minimize makespan in permutation flow shops," *Computers & Operations Research*, vol. 35, no. 9, pp. 3001–3008, 2008, part Special Issue: Bio-inspired Methods in Combinatorial Optimization.

[16] ——, "An empirical analysis of the optimality rate of flow shop heuristics," *European Journal of Operational Research*, vol. 198, no. 1, pp. 93–101, 2009.

[17] S. Rad, R. Ruiz, and N. Boroojerdian, "New high performing heuristics for minimizing makespan in permutation flowshops," *Omega*, vol. 37, no. 2, pp. 331–345, 2009.

[18] I. Ribas, R. Companys, and X. Tort-Martorell, "Comparing three-step heuristics for the permutation flow shop problem," *Computers & Operations Research*, vol. 37, no. 12, pp. 2062–2070, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030505481000050X

[19] K.-C. Ying and S.-W. Lin, "A high-performing constructive heuristic for minimizing makespan in permutation flowshops," *Journal of Industrial and Production Engineering*, vol. 30, no. 6, pp. 355–362, 2013. [Online]. Available: https://doi.org/10.1080/21681015.2013.843597

[20] R. Puka, J. Duda, A. Stawowy, and I. Skalna, "N-NEH+ algorithm for solving permutation flow shop problem," *Computers & Operations Research*, vol. 132, p. 105296, 2021.

[21] R. Puka, B. Łamasz, and I. Skalna, "Improving n-neh+ algorithm by using starting point method," in *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2022, pp. 357–361.

[22] R. Puka, I. Skalna, and B. Łamasz, "Swap method to improve n-neh+ algorithm," in *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, 2022, pp. 1–6.