IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 11 April 2024, accepted 5 May 2024, date of publication 14 May 2024, date of current version 22 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3400900

RESEARCH ARTICLE

Deterministic Method for Input Sequence Modification in NEH-Based Algorithms

RADOSŁAW PUKA, IWONA SKALNA, (Member, IEEE), BARTOSZ ŁAMASZ, JERZY DUDA, AND ADAM STAWOWY
Faculty of Management, AGH University of Krakow, 30-059 Kraków, Poland

Corresponding author: Radosław Puka (rpuka@agh.edu.pl)

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

ABSTRACT Scheduling of production jobs falls into the area of planning, which, according to Henri Fayol's conception, is one of the basic functions of management. The permutation flow-shop scheduling problem (PFSP) with makespan criterion is one of the most studied scheduling problems in the area of scheduling theory and applications. The most-known polynomial complexity method for solving this complex problem is the Nawaz-Enscore-Ham (NEH) deterministic constructive algorithm. The subject literature shows that the results of NEH strongly depend on the input sequence of jobs. In this paper, we propose a new method to build the input sequence of jobs for NEH-based heuristics. The proposed Turn-off-Machine (ToM) method and its generalized version ToM+ (which has the feature to produce a set of input sequences that can be used in population-based optimization methods) compute the total processing time of jobs by virtually "turning off" one machine. The ToM+ method is one of a few deterministic methods for modifying the input sequence, and is the first one that modifies the input sequence based on individual machine processing times. Extensive numerical experiments on standard Taillard and VRF benchmarks show the good efficiency of the proposed method in solving PSFP with makespan criterion. The method improved the performance (measured using ARPD) of the NEH-based algorithms by up to nearly 35%. Moreover, by combining ToM+ method, SMαP+, N-list, and vN-list technique, it was possible to improve the results of the original NEH algorithm by up to nearly 50% (the method outperformed most of the NEH-based methods). This confirms that creating an adequate input sequence is of great importance for the performance of NEH-based algorithms.

INDEX TERMS Heuristics, NEH, ToM method, N-list technique, permutation flow-shop scheduling problem, makespan.

I. INTRODUCTION

The permutation flow-shop scheduling problem (PFSP) is one of the most studied scheduling problems in the area of theory and applications. The PFSP can be described as a production decision problem in which n jobs $J_1, \ldots, J_n$ are processed on m machines $M_1, \ldots, M_m$ in a fixed order, and the goal is to find the sequence $\pi = (\pi(1), \ldots, \pi(n)) \in S_n$ of jobs that minimizes an objective function (e.g., makespan, total earliness and tardiness, total weighted completion time, total flow time). Each job to be scheduled is identified by a set of tasks (operations) that form the path to be followed in the production process, and each task has specified processing time on each machine. The tasks are not allowed to overtake each other [15], and, obviously, no machine can perform more than one operation simultaneously. Many modifications of PFSP have also been developed, such as distributed permutation flow shop scheduling problem (DPFSP) [19], blocking permutation flow-shop scheduling problems (BPFSP) [36], distributed permutation flowshop scheduling problem with blocking constraints (DBHFSP) [40], sequence-independent setup time permutation flow shop problem (PFSP-SIST) [20]. The PFSP and its variants have many real-world applications; it is used, for example, in the chemical industry, the waste treatment industry, the aeronautics parts fabrication industry [24], in the

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Gaggero.

TABLE 1. Notation.

n	number of jobs to be processed,
m	number of available machines,
J_j	job j,
M_i	machine i,
$p_{i,j}$	processing time of job j on machine i,
L	list of jobs to be processed,
L_P	partial sequence of scheduled jobs,
$L_P(m')$	sequence of scheduled jobs obtained with machine m' "turned off",
L_N	N-list (list of candidate jobs),
TPT_j	total processing time of job j,
$\mathrm{TPT}'(m')_j$	total processing time of job j with machine m' virtually "turned off",
$C_{\max}$	makespan (maximum completion time of all jobs on all machines),
$C_{\max}(m')$	makespan obtained by virtually "turning off" machine m'.

manufacturing of printed circuit boards [2], in iron and steel industry, shipbuilding [36], in the production of cider [32], and in the production of apparel and garments [4].

A. COMPLEXITY OF PFSP

As mentioned above, different optimization criteria (objective functions) are considered in the subject literature, however, the PFSP with the makespan criterion, often denoted as $Fm|prmu|C_{\max}$ [14], is one of the most thoroughly studied production scheduling problems; it is also of the main concern in this paper. The high interest in $Fm|prmu|C_{\max}$ may be due to the fact that the problem has been proven to be NP-hard if $m > 2$ [11], so solving it effectively is a big challenge. To tackle the complexity of $Fm|prmu|C_{\max}$, various deterministic, stochastic, constructive, and metaheuristic algorithms have been developed over the past years, and currently different modifications of these methods emerge in the literature. In this work, we focus on constructive deterministic algorithms for the PFSP.

B. DETERMINISTIC CONSTRUCTIVE ALGORITHMS

One of the best-known deterministic constructive algorithms for solving $Fm|prmu|C_{\max}$, as evidenced by the subject literature, is the Nawaz-Ham-Enscore (NEH) heuristic [23]. The undoubted advantages of NEH (its simplicity and good efficiency) made it an inspiration for many research aimed at improving its operation. The improvements presented in the literature include, among others, the insertion tie-breaking rules [17], [18] and modifications of the input sequence. An important improvement based on the N-list technique was proposed in [25]. Other improvements can be found, for example, in [16] and [33]. A more detailed overview of the literature on NEH improvements is presented in the section devoted to NEH-based algorithms.

C. PROPOSED METHOD FOR SOLVING PFSP

In this paper, we propose a new method to modify the input sequence in NEH-based algorithms. Our motivation to develop this method was that input sequence modifications, especially within the context of deterministic methods, are still not well studied in the literature. Most of the works on the subject consider various input sequence ordering methods (priority rules). The main objective of our work is to show that the existing priority rules can be enhanced with the proposed ToM method and that this enhancement can bring significant improvement of the results of NEH-based algorithms.

The results of numerical experiments on standard Taillard and VRF benchmarks show that the proposed modification can significantly improve the results of the NEH-based algorithms. The results were analyzed using the latest best-known solutions for both benchmarks [12], [13]. In addition to the classical ARPD measure, the relative measure ARD.NEH [27] was also used for error analysis.

Another advantage of one of the proposed methods is that they partially solve the problem of generating input for population-based scheduling optimization algorithms. This is due to the fact that these methods are able to produce a set of good quality solutions to $Fm|prmu|C_{\max}$ in a single run.

D. PAPER ORGANIZATION

The rest of the paper is organized as follow. First, we present selected NEH-based algorithms. Then, we describe the proposed method for creating the input sequence. Next, we provide the results of extensive numerical experiments on the Taillard and VRF benchmark problems. The paper ends with a discussion and concluding remarks.

II. NEH-BASED ALGORITHMS

The NEH algorithm, shown in Algorithm 1, is a constructive heuristic with polynomial time complexity $\mathcal{O}(n^3m)$ which can be reduced to $\mathcal{O}(n^2m)$ by using Taillard acceleration [37]. NEH consists of the initial phase and the insertion phase. In the initial phase, jobs are sorted in nonincreasing order of their total processing times. The total processing time (TPT) of job j is defined as

$$\mathrm{TPT}_j = \sum_{i=1}^{m} p_{i,j},\ j = 1, \dots, n, \tag{1}$$

where $p_{i,j}$ is the processing time of job j on machine i. Thus, the resulting input sequence $J_{\pi(1)}, \dots, J_{\pi(n)}$, $\pi \in S_n$, fulfills the condition:

$$\mathrm{TPT}_{\pi(1)} \geqslant \dots \geqslant \mathrm{TPT}_{\pi(n)}. \tag{2}$$

The impact of the input sequence (obtained in the initial phase) on NEH results has been studied, for example, in [10], [26], and [30]. Framinan et al. [10] showed that the initial order used in the original NEH algorithm is the best single criterion among the 177 starting sequences evaluated. Ruiz and Maroto [34] examined 25 different heuristics with different starting sequences and showed that the NEH algorithm gives the best results among all heuristics examined, moreover, in a much shorter time. Many other multi-element rules have been later proposed. Dong et al. [3] presented a special

Algorithm 1 NEH Algorithm for Solving $Fm|prmu|C_{\max}$

Initial phase
Sort n jobs in non-increasing order of their total processing times and put them into the initial list of jobs $L = \{1, \ldots, n\}$.
Insertion phase
Initialize the partial sequence L_P of jobs with the first job from L ($L_P = L_P \cup \{1\}$) and remove this job from the list of jobs ($L = L \setminus \{1\}$).
for $j = 2, \ldots, n$ **do**
  Insert the first job from L in L_P in the place (among k possible) that minimizes the partial makespan.
  $L = L \setminus \{j\}$.
end for
return L_P and $C_{\max}$.

Algorithm 2 N-NEH Algorithm for Solving $Fm|prmu|C_{\max}$

Initial phase
Sort n jobs in non-increasing order of their total processing times and put them into the initial list of jobs $L = \{1, \ldots, n\}$.
Insertion phase
Initialize the partial sequence L_P of jobs with the first job from L and $L = L \setminus \{1\}$
Initialize the N-list, L_N, with the jobs $\{2, \ldots, \min\{N+1, n\}\}$ and remove these jobs from L.
for $j = 2, \ldots, n$ **do**
  Evaluate each job in L_N, put the best job in the respective place in L_P and remove this job from L_N
  if ($L \neq \emptyset$) **then**
    Append the first job from L to L_N and remove this job from L.
  end if
end for
return L_P and $C_{\max}$.

Algorithm 3 vN-NEH Algorithm for Solving $Fm|prmu|C_{\max}$

Initial phase
Sort n jobs in non-increasing order of their total processing times and put them into the initial list of jobs $L = \{1, \ldots, n\}$.
Insertion phase
Initialize the partial sequence L_P of jobs with the first job from L and $L = L \setminus \{1\}$
Initialize the vN-list, L_N, with the jobs $\{2, \ldots, \min\{N+1, n\}\}$ and remove these jobs from L
for $j = 2, \ldots, n$ **do**
  Evaluate each job in L_N, put the best job in the respective place in L_P and remove this job from L_N
  if ($L_N = \emptyset$ and $L \neq \emptyset$) **then**
    Refill L_N with $\min\{N, |L|\}$ jobs from L and remove these jobs from L.
  end if
end for
return L_P and $C_{\max}$.

priority rule assigning higher priority to jobs with a larger variation of the processing times on each machine. The authors used the sum of the average processing times and the standard deviation of the processing times for the first phase of their NEH-D algorithm. NEH-D is able to provide better solutions than the original NEH algorithm. At the same time, Kalczynski and Kamburowski [17]) proposed the NEHKK1 heuristic, where (different) weights are assigned to job processing times according to their position in the flow line. Liu et al. [21] proposed to incorporate the third and fourth statistical moments into the Dong et al. [3] priority rule in the initial phase of NEH. The authors demonstrated the effectiveness of the priority rule based on skewness and the ineffectiveness of the kurtosis-based rule. Recently, Zhang et al. [43] introduced the self-attention mechanism, and job similarities (characterized by the dot-product of processing time matrices) were used as job priorities. The computational results with the Taillard and VRF benchmarks demonstrate that the new priority rule dominates the existing ones at a nominal cost of computation time. A method to modify the input sequence, based on a selected factor (e.g., TPT), was proposed in [29]. Modifications of the second (insertion) phase of NEH (where the jobs from the input sequence are placed in the partial sequence to obtain the smallest makespan $C_{\max}$) have been proposed, e.g., in [25] and [28]. The algorithms N-NEH (Algorithm 2) and vN-NEH (3), proposed therein, are based on the N-list technique. The so-called candidate jobs from the N-list (additional to the main list of jobs), are analyzed in each step of the insertion phase to give the smallest makespan. Therefore, in contrast to classical NEH, more than one job can be analyzed at each step of the insertion phase.

The literature review shows that modifications of constructive algorithms proposed in the literature are mainly based on new priority rules and tie-breaking strategies. There is only one study on a deterministic method to modify the input sequence [29]. The results of this method indicate that modifications of the input sequence should be analyzed more closely, since they can significantly improve the results of NEH and NEH-based algorithms.

The next section presents a new modification of the input sequence in NEH-based algorithms. The operation of the proposed approach will be verified in Section IV using numerical experiments on standard Taillard and VRF benchmarks.

III. PROPOSED METHOD

The *Turning-off-Machine* (ToM) modifies the way the input sequence is created in NEH-based algorithms. Let us recall that in the original NEH algorithm, jobs are sorted in non-increasing order of their total processing times, so the jobs in the input sequence fulfill the condition (2).

TABLE 2. Processing times of jobs for $Fm|prmu|C_{\max}$ with four jobs ($n = 4$) and four machines ($m = 4$).

Job	Machine			
	1	2	3	4
1	26	72	79	95
2	49	89	81	67
3	52	73	93	77
4	70	55	46	88

TABLE 3. Input sequences produced by turning of one (respectively M_1, M_2, M_3, M_4) machine and resulting input sequences.

Job	Machine				TPT'(1)		Input sequence
	1	2	3	4			
1		72	79	95	246		1
2		89	81	67	237	⇒	3
3		73	93	77	243		2
4		55	46	88	189		4

Job	Machine				TPT'(2)		Input sequence
	1	2	3	4			
1	26		79	95	200		3
2	49		81	67	197	⇒	4
3	52		93	77	222		1
4	70		46	88	204		2

Job	Machine				TPT'(3)		Input sequence
	1	2	3	4			
1	26	72		95	193		4
2	49	89		67	205	⇒	2
3	52	73		77	202		3
4	70	55		88	213		1

Job	Machine				TPT'(4)		Input sequence
	1	2	3	4			
1	26	72	79		177		2
2	49	89	81		219	⇒	3
3	52	73	93		218		1
4	70	55	46		171		4

The proposed here ToM method relies on excluding ("turning off") the m' (m' is a parameter of the ToM method) machine from computing the TPT of jobs. Thus, the resulting input sequence $J_{\pi(1)}, \ldots, J_{\pi(n)}$ fulfills the condition:

$$\mathrm{TPT}'(m')_{\pi(1)} \geqslant \ldots \geqslant \mathrm{TPT}'(m')_{\pi(n)}, \tag{3}$$

where $\mathrm{TPT}'(m')$ is computed as follows

$$\mathrm{TPT}'(m')_j = \sum_{\substack{i=1 \\ i \neq m'}}^{m} p_{i,j}, \; j = 1, \ldots, n. \tag{4}$$

Remark. It should be underlined that the ToM method can be used to create other (than TPT) sorting criteria, it is enough to exclude ("turn off") the selected machine from the computation in the initial phase.

Example 1: The operation of the ToM method is illustrated using a simple example of $Fm|prmu|C_{\max}$ with 4 machines and 4 jobs. Input data for $Fm|prmu|C_{\max}$ is given

TABLE 4. TPT and TPT′(1)-(4) values for data in Table 2 and respective input sequences.

	Job	TPT	TPT′(1)	TPT′(2)	TPT′(3)	TPT′(4)
	1	272	246	200	193	177
	2	286	237	197	205	219
	3	295	243	222	202	218
	4	259	189	204	213	171
		⇓	⇓	⇓	⇓	⇓
Inp. seq.		3	1	3	4	2
		2	3	4	2	3
		1	2	1	3	1
		4	4	2	1	4

in Table 2, the operation of ToM is illustrated in Table 3, and the summary of total processing times and the respective input sequences are summarized in Table 4. As can be seen from Table 4, the ToM method has a significant impact on the input sequence, i.e., the obtained input sequences differ greatly from each other. If $m' \notin \{1, \ldots, m\}$, then ToM gives the same input sequence as the one obtained by using TPT.

ToM can be run together with the insertion phase (e.g., from NEH) for all $m' \in \{0, 1, \ldots, m\}$. The resulting ToM+ method (see Algorithm 4) can be considered as a new method for solving $Fm|prmu|C_{\max}$. As can be seen, ToM+ produces

Algorithm 4 ToM+ Algorithm for Solving $Fm|prmu|C_{\max}$

for $m' = 0, 1, \ldots, m$ **do**
  Initial phase
  Sort n jobs in non-increasing order of $\mathrm{TPT}'(m')_j$ and put them into the initial list of jobs $L(m') = \{1, \ldots, n\}$.
  Insertion phase
  Insert jobs into partial sequence $L_P(m')$ according to an insertion strategy.
end for
return $L_P(m''), C_{\max}(m'')$
(where $m'' = \arg\min\{C_{\max}(m') : m' = 0, 1, \ldots, m\}$).

the population of $m + 1$ solutions from among which the best, with respect to the considered criterion (here makespan), solution is selected as a final result. The asymptotic time complexity of ToM+ is m-times greater than the complexity of the respective (underlying) scheduling algorithm. For example, the asymptotic time complexity of NEH algorithm with Taillard acceleration and combined with the ToM+ method is $\mathcal{O}(m^2n^2)$.

The quality of the solutions produced by the ToM-based methods and other methods considered in this study will be evaluated by using two quality measures. The first measure is the average relative percentage deviation (ARPD) (used, e.g., in [22], [34]) defined as:

$$\mathrm{ARPD} = \frac{1}{I} \sum_{i=1}^{I} \frac{S_i - S_{i,best}}{S_{i,best}}, \tag{5}$$

TABLE 5. ARPD [%] for *N*-NEH, ToM+*N*-NEH (ToM+*N*), *vN*-NEH and ToM+*vN*-NEH (ToM+*vN*) algorithms on Taillard and VRF benchmarks.

Benchmark	Method	Length of list of candidate jobs							
		1	2	3	4	5	6	7	8
Taillard	N-NEH	3.37	3.23	3.14	3.27	3.38	3.28	3.25	3.26
	ToM+N	2.45	2.35	2.34	2.27	2.26	2.3	2.29	2.33
	vN-NEH	3.37	3.27	3.15	3.04	3	3.01	2.93	3.04
	ToM+vN	2.45	2.35	2.28	2.21	2.24	2.2	2.17	2.19
VRF S	N-NEH	3.88	3.79	3.89	4.04	4.01	3.97	3.98	3.94
	ToM+N	2.66	2.64	2.62	2.65	2.66	2.75	2.78	2.84
	vN-NEH	3.88	3.76	3.73	3.6	3.6	3.52	3.52	3.45
	ToM+vN	2.66	2.61	2.55	2.5	2.46	2.46	2.42	2.46
VRF L	N-NEH	3.41	3.14	2.98	2.88	2.81	2.78	2.73	2.69
	ToM+N	2.87	2.62	2.46	2.38	2.33	2.26	2.23	2.19
	vN-NEH	3.41	3.32	3.19	3.12	3.07	3	2.95	2.88
	ToM+vN	2.87	2.76	2.69	2.6	2.54	2.51	2.45	2.42

TABLE 6. ARPD [%] for *N*-NEH+, ToM+*N*+ and ToM+*vN*+ algorithms on Taillard and VRF benchmarks.

Benchmark	Method	Length of list of candidate jobs			
		1	2	4	8
Taillard	N-NEH+	3.37	2.99	2.60	2.36
	ToM+N+	2.45	2.24	2.02	1.83
	vN-NEH+	3.37	3.02	2.67	2.28
	ToM+vN+	2.45	2.25	2.02	1.81
VRF S	N-NEH+	3.88	3.36	2.99	2.68
	ToM+N+	2.66	2.40	2.19	2.01
	vN-NEH+	3.88	3.40	2.99	2.50
	ToM+vN+	2.66	2.43	2.17	1.91
VRF L	N-NEH+	3.41	3.08	2.75	2.47
	ToM+N+	2.87	2.61	2.33	2.10
	vN-NEH+	3.41	3.21	2.96	2.68
	ToM+vN+	2.87	2.73	2.55	2.34

TABLE 7. Non-parametric Wilcoxon signed-rank test for pairwise comparison of ToM+ method variants (with various lengths of the *vN*-list).

Benchmark	Compared algorithms	p-value
Taillard	ToM+N+(2) vs ToM+vN+(2)	0.349528
	ToM+N+(4) vs ToM+vN+(4)	0.516214
	ToM+N+(8) vs ToM+vN+(8)	0.852944
	ToM+NEH vs N+(4)	0.057228
	ToM+N+(2) vs N+(8)	0.101547
	ToM+N+(2) vs vN+(8)	0.659323
	ToM+vN+(2) vs N+(8)	0.126739
	ToM+vN+(2) vs vN+(8)	0.728701
VRF Small	ToM+N+(2) vs ToM+vN+(2)	0.469475
	ToM+N+(4) vs ToM+vN+(4)	0.472051
	ToM+NEH vs N+(8)	0.406127
	ToM+vN+(2) vs vN+(8)	0.060619
VRF Large	ToM+N+(4) vs ToM+vN+(8)	0.440911
	ToM+vN+(2) vs N+(4)	0.723265

where I is the number of problem instances, S_i is the solution of the evaluated algorithm on the instance $i \in I$, and $S_{i,best}$ is the best solution known so far for this instance. The ARPD measure depends on the best known solution so its value can change over time. In contrast, the value of the second quality measure ARD.NEH (Average Relative Deviation over NEH) measure, proposed in [27], does not change over time since it refers to the respective NEH results. The ARD.NEH measure is computed as follows:

$$\text{ARD.NEH} = \frac{1}{I} \sum_{i=1}^{I} \frac{\text{NEH}_i - S_i}{\text{NEH}_i}, \tag{6}$$

where NEH_i is the solution obtained using the NEH algorithm for instance $i \in I$.

The running time of the algorithms will be evaluated by using the average CPU (ACPU) usage [37] defined as:

$$\text{ACPU} = \frac{1}{I} \sum_{i=1}^{I} \text{CPU}_i, \tag{7}$$

where CPU_i is the CPU time of an algorithm on the instance $i \in I$. We will also use the ART.NEH (the Average Relative Time over NEH) relative measure recently proposed in [28]:

$$\text{ART.NEH} = \frac{\sum_{i=1}^{I} \frac{\text{CPU}_i}{\text{CPU}_{i,\text{NEH}}}}{I}, \tag{8}$$

where CPU_i is the CPU time of an algorithm on the instance $i \in I$, and $\text{CPU}_{i,\text{NEH}}$ is the CPU time of NEH on that instance. As can be seen from the formula (8), 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 algorithm. Moreover, ART.NEH, in contrary to ACPU, is software and hardware independent, and therefore, it is much more reliable and does not require reimplementation of algorithms from the literature.

Next section presents the results of the *N*-NEH, *N*-NEH+, *vN*-NEH, and *vN*-NEH+ algorithms with and without the usage of the proposed ToM+ method. We would like to underaline that all these algorithms employ Taillard's acceleration. For the purposes of this paper the algorithms are denoted as follows: ToM+*N* will stand for *N*-NEH combined with ToM+, ToM+*N*+ will stand for *N*-NEH+ combined with ToM+, ToM+*vN* will stand for *vN*-NEH combined with

TABLE 8. ART.NEH for ToM+*N* and ToM+*vN* algorithms on Taillard and VRF benchmarks.

Benchmark	Algorithm	Length of list of candidate jobs							
		1	2	3	4	5	6	7	8
Taillard	*N*-NEH	1	1.4	1.7	2.1	2.5	2.8	3.1	3.4
	ToM+*N*	13.5	18.8	24.2	29.1	34.5	38.9	43.5	48.3
	vN-NEH	1	1.2	1.3	1.5	1.7	1.9	2.1	2.2
	ToM+*vN*	13.5	16.1	18.8	21.4	23.9	26.7	29	31.1
VRF S	*N*-NEH	1	1.4	1.7	2	2.4	2.7	2.9	3.2
	ToM+*N*	13.3	18.5	23.3	27.8	32.1	35.8	39.3	42.5
	vN-NEH	1	1.2	1.4	1.6	1.7	1.8	2	2.2
	ToM+*vN*	13.3	15.7	18.4	20.7	23.1	25	27	29
VRF L	*N*-NEH	1	1.4	1.8	2.2	2.6	3	3.4	3.8
	ToM+*N*	41.2	57.8	74.5	90.9	107.4	123.3	140.9	157.1
	vN-NEH	1	1.2	1.4	1.6	1.8	2	2.2	2.4
	ToM+*vN*	41.2	49.6	57.8	66.1	74.4	82.4	90.8	99.6

ToM+, and ToM+*vN*+ will stand for *vN*-NEH+ combined with ToM+. Let us point out that if the length of the list of candidate jobs is equal to 1 then the above algorithms correspond, respectively, to NEH and NEH combined with ToM+.

IV. COMPUTATIONAL EXPERIMENTS

The scheduling algorithms mentioned in the previous section were implemented in C# and run on a computer with two Intel Xeon E5-2660 v4 CPUs (14 cores, each with 2.0 GHz base clock speed). Although some of these algorithms can be run on multiple cores, all of them were run on one core to make the comparison reliable.

We evaluate the efficiency of the considered algorithms by using the Taillard [38] and VRF [39] benchmark problems:

- Taillard benchmark includes 120 instances:
  $n \in \{20, 50, 100, 200, 500\}$,
  $m \in \{5, 10, 20\}$.
- VRF benchmark includes 480 instances divided into Small and Large subsets (each with 240 instances):
  - Small instances:
    $n \in \{10, 20, 30, 40, 50, 60\}$,
    $m \in \{5, 10, 15, 20\}$,
  - Large instances:
    $n \in \{100, 200, 300, 400, 500, 600, 700, 800\}$
    $m \in \{20, 40, 60\}$.

Best solutions, provided by the authors of these benchmarks, are updated in the present paper with recent results for Taillard and VRF benchmarks from, respectively, [12] and [13].

A. QUALITY OF RESULTS (ARPD)

Table 5 presents the ARPD values for the *N*-NEH+, *vN*-NEH+, ToM+*N*-NEH and ToM+*vN*-NEH algorithms (these result are provided for illustrative purposes only), and Table 6 presents the ARPD values for the ToM+*N*-NEH+, and ToM+*vN*-NEH+ algorithms. The lengths of list of candidate jobs for extended versions of algorithms (marked with a plus sign), have been selected based on [25].

The tables show that thanks to the use of the ToM+ method it is possible to significantly improve the results of the *N*-NEH, *N*-NEH+, *vN*-NEH, and *vN*-NEH+ algorithms. The improvement depends on the benchmark problem:

- for the VRF Large instances, the improvement does not exceed 20%, and the average improvement over all VRF Large instances is about 17%,
- for the VRF Small and Taillard benchmark problems, the improvement is greater than 25% for each length of the *N*-list, and the average improvement over all instances in both sets is nearly 30%.

Moreover, ToM+ improved the results of classical NEH (corresponding to the results with *N*-list/*vN*-list of length 1) on average by 25%.

To verify if the results of ToM+*N*-NEH, ToM+*vN*-NEH, *N*-NEH, and *vN*-NEH statistically differ significantly from each other, we used the non-parametric Wilcoxon signed-rank test (a paired difference test). Tables 7 shows only these pairs of compared methods for which no significant difference (at the significance level $\alpha = 0.05$) between their results has been observed. It is seen that for Taillard benchmark there is no statistically significant difference between the results of compared methods. In contrary, for the VRF Small instances the results of most of the compared methods statistically differ significantly, and for VRF Large instances there are only two pairs of methods whose results are not statistically different from each other. Based on the results of the Wilcoxon test we can conclude that for the majority of benchmark instances algorithms employing ToM+ methods produce results that are statistically different from each other.

B. COMPUTATIONAL TIME (ACPU, ART.NEH)

As already mentioned, the asymptotic time complexity of the algorithms that employ ToM+ ((ToM+)-based algorithms) increases $m + 1$ times, where m is the number of machines. Therefore, the biggest difference in computational times between the basic versions of the algorithms and their (ToM+)-based counterparts can be observed for VRF

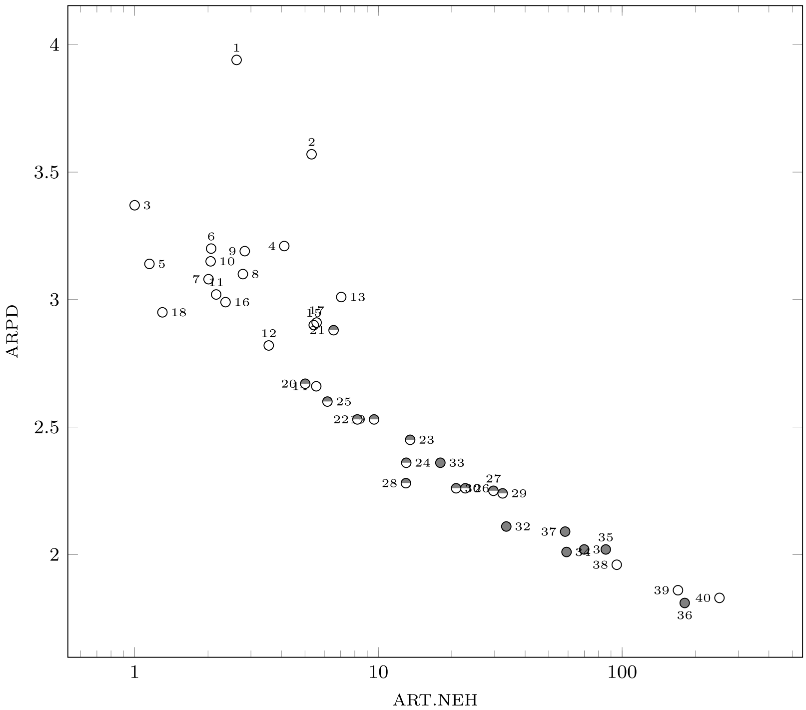


FIGURE 1. ARPD vs. ART.NEH of selected heuristics in logarithmic scale on Taillard's instances; circles mark state-of-the-art algorithms and bullets mark algorithms.

TABLE 9. ART.NEH for ToM+N+ and ToM+vN+ algorithms on Taillard and VRF benchmarks.

Benchmark	Algorithm	Length of list of candidate jobs			
		1	2	4	8
Taillard	*N*-NEH+	1	2.4	6.2	18
	ToM+*N*+	13.5	32.3	85.7	250.8
	vN-NEH+	1	2.2	5	13
	ToM+*vN*+	13.5	29.6	69.9	180.6
VRF S	*N*-NEH+	1	2.4	6.1	17.2
	ToM+*N*+	13.3	31.8	82.9	232.7
	vN-NEH+	1	2.2	5.1	12.8
	ToM+*vN*+	13.3	29	68.2	172.3
VRF L	*N*-NEH+	1	2.4	6.4	19.3
	ToM+*N*+	41.2	99.1	264.5	793.2
	vN-NEH+	1	2.2	5.2	13.6
	ToM+*vN*+	41.2	90.8	214.7	561.9

Large instances (where m ranges from 20 to 60). From Tables 15 and 16, we can see that the computational time of the (ToM+)-based algorithms increased about 15 times for VRF S and 45 times for VRF L instances, and about 18 times for Taillard benchmark.

Tables 8 and 9 show the ART.NEH values obtained for the (ToM+)-based algorithms. As can be seen from the tables, ART.NEH for VRF benchmark are close to the values obtained from the division of ACPU of an algorithm by ACPU of NEH:

$$\mathrm{ART.NEH}_a \approx \mathrm{ACPU}_a/\mathrm{ACPU}_{\mathrm{NEH}}, \tag{9}$$

where a denotes the evaluated algorithm. This follows from the specific construction of the VRF benchmark, where for each number of jobs, there is a constant set of machines ($m \in \{5, 10, 15, 20\}$ for VRF S and $m \in \{20, 40, 60\}$ for VRF L). In the case of Taillard benchmark, the CPU for the largest instance (with $n = 500$, $m = 20$) has a strong impact on the value of $\mathrm{ACPU}_a/\mathrm{ACPU}_{\mathrm{NEH}}$. Therefore, for the Taillard benchmark, there is a large difference between $\mathrm{ACPU}_a/\mathrm{ACPU}_{\mathrm{NEH}}$, which is on average over 18, and the $\mathrm{ART.NEH}_a$, which on average does not exceed 14. The ART.NEH is software, hardware, and instance independent, therefore it is more reliable than ACPU. Hence, based on ART.NEH, the usage of the ToM+ method for the Taillard benchmark increases the computation time by about 14 times (which is as much as 1/4 lower than it would appear from the ACPU index).

C. COMPARISON OF RESULTS

Figs. 1 and 2 show the comparison of (ToM+)-based algorithms with the most known deterministic constructive algorithms (listed in Table 10) for solving $Fm|prmu|C_{\max}$ ([3], [5], [6], [17], [18], [31], [33], [42]).

The figures show that the effectiveness of the ToM+ method varies depending on the benchmark analyzed. For the Taillard's benchmark, the proposed method definitely stands out in terms of efficiency of operation. For the VRF Large instance, one can also see the high efficiency of the ToM+ method, which, however, appears to be slightly less efficient than the SMα+ method. To verify if the results of

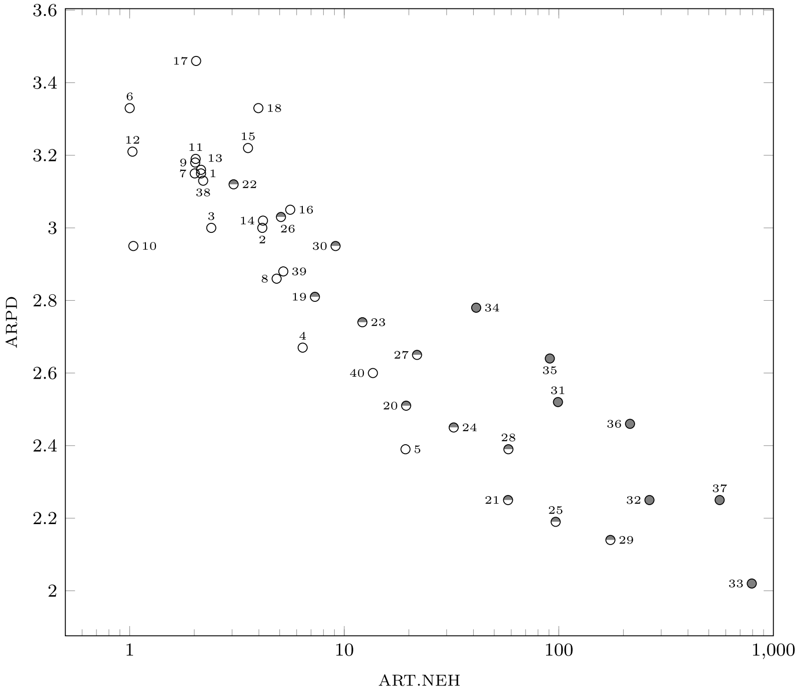


FIGURE 2. ARPD vs. ART.NEH (in logarithmic scale) of selected heuristics on VRF L instances; circles mark state-of-the-art algorithms and bullets mark algorithms.

TABLE 10. (ToM+)-based and other known deterministic constructive methods for solving $Fm|prmu|C_{max}$.

No.	Algorithm	No.	Algorithm
1	KKER	21	SMα+(2)N-NEH+(8)
2	KKER-di	22	SMα+(2)NEH
3	N-NEH+(2)	23	SMα+(4)N-NEH+(2)
4	N-NEH+(4)	24	SMα+(4)N-NEH+(4)
5	N-NEH+(8)	25	SMα+(4)N-NEH+(8)
6	NEH	26	SMα+(4)NEH
7	NEH1-di	27	SMα+(8)N-NEH+(2)
8	NEHD-di	28	SMα+(8)N-NEH+(4)
9	NEH-di	29	SMα+(8)N-NEH+(8)
10	NEHFF	30	SMα+(8)NEH
11	NEHKK1-di	31	ToM+N-NEH+(2)
12	NEHKK2	32	ToM+N-NEH+(4)
13	NEHR	33	ToM+N-NEH+(8)
14	NEHR-di	34	ToM+NEH
15	NEMR	35	ToM+vN-NEH+(2)
16	NEMR-di	36	ToM+vN-NEH+(4)
17	RAER	37	ToM+vN-NEH+(8)
18	RAER-di	38	vN-NEH+(2)
19	SMα+(2)N-NEH+(2)	39	vN-NEH+(4)
20	SMα+(2)N-NEH+(4)	40	vN-NEH+(8)

TABLE 11. Results of non-parametric Wilcoxon signed-rank test.

Benchmark	Compared algorithms	*p*-value
Taillard	ToM+N-NEH & SMα+(2)N-NEH+(2)	0.263956
	ToM+*N*-NEH+(2) & SMα+(2)N-NEH+(4)	0.520901
	ToM+*N*-NEH+(2) & SMα+(8)N-NEH+(2)	0.547370
	ToM+*N*-NEH+(4) & SMα+(2)N-NEH+(8)	0.166506
	ToM+*N*-NEH+(4) & SMα+(8)N-NEH+(4)	0.518320
	ToM+*N*-NEH+(8) & SMα+(8)N-NEH+(8)	0.284563
	ToM+*vN*-NEH & SMα+(2)N-NEH+(2)	0.263956
	ToM+*vN*-NEH+(2) & SMα+(2)N-NEH+(4)	0.924866
	ToM+*vN*-NEH+(2) & SMα+(8)N-NEH+(2)	0.740821
	ToM+*vN*-NEH+(4) & SMα+(2)N-NEH+(8)	0.815055
	ToM+*vN*-NEH+(4) & SMα+(4)N-NEH+(4)	0.214941
	ToM+*vN*-NEH+(4) & SMα+(8)N-NEH+(4)	0.261867
	ToM+*vN*-NEH+(8) & SMα+(8)N-NEH+(8)	0.404533
VRF L	ToM+vN-NEH & NEHFF	0.076991
	ToM+*vN*-NEH & NEHD-di	0.533140
	ToM+N-NEH & SMα+(2)N-NEH+(2)	0.893035
	ToM+N-NEH+(4) & SMα+(2)N-NEH+(8)	0.073970
	ToM+vN-NEH & SMα+(2)N-NEH+(2)	0.893035
	ToM+vN-NEH+(2) & SMα+(8)N-NEH+(2)	0.385093
	ToM+vN-NEH+(4) & SMα+(2)N-NEH+(4)	0.591575

N-NEH with ToM+ and N-NEH with SMα+ are statistically different from each other, we again use the non-parametric Wilcoxon signed-rank test. The results are presented in Table 11. The results of the test indicate that in general there is no statistically significant difference between the results of the compared methods. It is worth noting that the input sequence modification methods (ToM+ and SMα+) allow obtaining by far the best results among the analyzed algorithms. This observation confirms great impact of the input sequence on the results of NEH. Therefore, we decided to combine ToM+ with SMα+ and N-list technique, and investigate the potentials of these combinations. The obtained result are described in the next section.

TABLE 12. APRD values for ToM+SMα+N+ and ToM+SMα+vN+ algorithms.

Benchmark	Swap α	ToM+N+				ToM+vN+			
		1	2	4	8	1	2	4	8
Taillard	0	2.45	2.24	2.02	1.83	2.45	2.25	2.02	1.81
	2	2.15	2	1.81	1.66	2.15	2	1.83	1.65
	4	2.05	1.89	1.71	1.58	2.05	1.92	1.74	1.57
	8	1.97	1.78	1.61	**1.51**	1.97	1.82	1.67	**1.5**
VRF S	0	2.66	2.4	2.19	2.01	2.66	2.43	2.17	1.91
	2	2.3	2.1	1.92	1.8	2.3	2.09	1.87	1.66
	4	2.13	1.96	1.81	1.72	2.13	1.94	1.74	1.57
	8	2.02	1.83	1.72	**1.63**	2.02	1.83	1.65	**1.49**
VRF L	0	2.87	2.61	2.33	2.1	2.87	2.73	2.55	2.34
	2	2.77	2.51	2.25	2.03	2.77	2.63	2.45	2.25
	4	2.72	2.47	2.21	2	2.72	2.59	2.42	2.21
	8	2.66	2.42	2.18	**1.96**	2.66	2.55	2.38	**2.17**

TABLE 13. ARD.NEH [%] for N-NEH+, ToM+N+ and ToM+vN+ algorithms on Taillard and VRF benchmarks.

Benchmark	Method	Length of list of candidate jobs							
		1	2	3	4	5	6	7	8
Taillard	N-NEH	0	0.13	0.22	0.09	-0.02	0.08	0.1	0.1
	ToM+N	0.88	0.98	0.98	1.05	1.06	1.02	1.03	0.99
	vN-NEH	0	0.09	0.2	0.31	0.35	0.34	0.42	0.31
	ToM+vN	0.88	0.97	1.05	1.11	1.08	1.12	1.15	1.13
VRF S	N-NEH	0	0.08	-0.02	-0.16	-0.14	-0.1	-0.1	-0.07
	ToM+N	1.16	1.18	1.2	1.17	1.16	1.06	1.04	0.99
	vN-NEH	0	0.11	0.14	0.26	0.26	0.34	0.34	0.41
	ToM+vN	1.16	1.22	1.27	1.31	1.35	1.35	1.39	1.35
VRF L	N-NEH	0	0.26	0.41	0.51	0.58	0.61	0.66	0.7
	ToM+N	0.52	0.76	0.92	1	1.04	1.12	1.14	1.18
	vN-NEH	0	0.09	0.22	0.28	0.33	0.4	0.44	0.52
	ToM+vN	0.52	0.63	0.7	0.78	0.84	0.87	0.93	0.95

TABLE 14. ARD.NEH [%] for N-NEH+, ToM+N+ and ToM+vN+ algorithms on Taillard and VRF benchmarks.

Benchmark	Method	Length of list of candidate jobs			
		1	2	4	8
Taillard	N-NEH+	0.00	0.36	0.74	0.96
	ToM+N+	0.88	1.08	1.29	1.47
	vN-NEH+	0.00	0.34	0.67	1.04
	ToM+vN+	0.88	1.07	1.29	1.49
VRF S	N-NEH+	0.00	0.49	0.85	1.14
	ToM+N+	1.16	1.41	1.61	1.78
	vN-NEH+	0.00	0.46	0.85	1.31
	ToM+vN+	1.16	1.39	1.63	1.88
VRF L	N-NEH+	0.00	0.32	0.64	0.91
	ToM+N+	0.52	0.77	1.04	1.27
	vN-NEH+	0.00	0.19	0.44	0.71
	ToM+vN+	0.52	0.65	0.84	1.04

D. ToM+SMα+N+ AND ToM+SMα+vN+ ALGORITHMS

This section presents algorithms that extend the N-NEH algorithm by using the ToM+ and SMα+ methods, respectively. An undoubted advantage of the ToM+ and SMα+ methods (in addition to the ability of being combined with each other) is the possibility to use them with the original NEH algorithm. In addition, NEH-based algorithm employing these two input sequence modification methods preserve their deterministic and constructive character (let us recall that these features are an important factor in selecting algorithms considered in this paper). Table 12 shows the results of the proposed algorithms. For a given parameter k, SMα+, ToM+ and ToM+vN+ must be run $k+1$ times, hence their computational times is respectively greater. However, the presented ARPD values confirm the efficiency of the method resulting from the combination of the ToM+SMα+ method with the N-list technique. The ToM+SMα+vN+ algorithm gave better results for smaller instances, whereas ToM+SMα+N+ gave better results for larger instances. For Taillard instances, both algorithms produced similar results and they both outperformed the FRB5 algorithm (which achieved the smallest ARPD $=$ 1.53 [5] known so far for deterministic algorithms). FRB5 is known to be one of the most effective algorithms for solving $Fm|prmu|C_{\max}$ (it belongs to the FRB deterministic algorithms, which, however, are not constructive algorithms, therefore, they are not included in the presented summary of the results).

For the Taillard benchmark, the ToM+SMα+ improved ARPD result of the NEH algorithm from 3.37 to 1.97. This means that among deterministic construction algorithms only N-list-based methods allowed to obtain results better than

TABLE 15. ACPU [s] for ToM+*N* and ToM+*vN* algorithms on Taillard and VRF benchmarks.

Benchmark	Algorithm	Length of list of candidate jobs							
		1	2	3	4	5	6	7	8
Taillard	*N*-NEH	64.3	88.2	95.2	128.9	151.0	165.9	190.2	233.7
	ToM+*N*	1155.6	1627.5	2082.7	2528.7	3019.1	3441.1	3880.9	4469.4
	vN-NEH	64.3	74.4	80.8	86.7	98.8	116.9	128.0	145.5
	ToM+*vN*	1155.6	1378.6	1648.8	1842.9	2070.2	2337.2	2532.0	2757.3
VRF S	*N*-NEH	1.7	2.3	3.0	3.6	4.2	4.9	5.4	5.9
	ToM+*N*	26.7	37.4	47.8	57.8	67.4	76.4	85.1	93.5
	vN-NEH	1.7	2.0	2.3	2.7	3.0	3.3	3.6	3.9
	ToM+*vN*	26.7	32.0	37.1	42.2	47.3	52.2	56.9	61.0
VRF L	*N*-NEH	970.0	1329.4	1724.1	2115.1	2481.3	2854.1	3258.2	3696.0
	ToM+*N*	44377.5	62031.5	80050.1	97536.3	115522.9	132293.1	152445.2	170653.9
	vN-NEH	970.0	1135.3	1334.0	1521.3	1695.0	1903.9	2109.6	2316.1
	ToM+*vN*	44377.5	53317.3	61983.1	71018.0	79719.4	88366.7	97481.6	107578.7

TABLE 16. ACPU for ToM+*vN*-NEH+ and ToM+*vN*-NEH+ algorithms on Taillard and VRF benchmarks.

Benchmark	Algorithm	Length of list of candidate jobs			
		1	2	4	8
Taillard	*N*-NEH+	64.3	152.6	376.7	1117.4
	ToM+*N*+	1155.6	2783.2	7394.6	22205.2
	vN-NEH+	64.3	138.7	306.2	795.3
	ToM+*vN*+	1155.6	2534.2	6026	15722.7
VRF S	*N*-NEH+	1.7	4.1	10.7	31.2
	ToM+*N*+	26.7	64.1	169.8	492.3
	vN-NEH+	1.7	3.8	8.8	22.6
	ToM+*vN*+	26.7	58.7	138.1	355.5
VRF L	*N*-NEH+	970	2299.4	6138.6	18428.1
	ToM+*N*+	44377.5	106409	283995.4	854910.5
	vN-NEH+	970	2105.4	4960.6	12985.2
	ToM+*vN*+	44377.5	97694.8	230695.9	603842.2

TABLE 17. ARD.NEH values for ToM+SMα+N+ and ToM+SMα+vN+ algorithms.

Benchmark	Swap α	ToM+N+				ToM+vN+			
		1	2	4	8	1	2	4	8
Taillard	0	0.88	1.08	1.29	1.47	0.88	1.07	1.29	1.49
	2	1.16	1.31	1.49	1.64	1.16	1.31	1.48	1.65
	4	1.26	1.41	1.59	1.71	1.26	1.39	1.56	1.72
	8	1.34	1.52	1.68	**1.78**	1.34	1.48	1.63	**1.79**
VRF S	0	1.16	1.41	1.61	1.78	1.16	1.39	1.63	1.88
	2	1.5	1.7	1.87	1.99	1.5	1.7	1.92	2.12
	4	1.67	1.83	1.97	2.06	1.67	1.85	2.04	2.2
	8	1.78	1.95	2.06	**2.15**	1.78	1.96	2.13	**2.28**
VRF L	0	0.52	0.77	1.04	1.27	0.52	0.65	0.84	1.04
	2	0.62	0.87	1.12	1.33	0.62	0.75	0.93	1.12
	4	0.67	0.91	1.16	1.37	0.67	0.8	0.96	1.16
	8	0.72	0.96	1.19	**1.4**	0.72	0.83	0.99	**1.2**

the existing results. For VRF Small benchmark, ARPD has changed from 3.84 to 2.02, which gives an improvement of about 50%. The smallest improvement was achieved for the VRF Large benchmark, where ARPD has changed from 3.41 to 2.66.

Both described methods (ToM+ and SMα+) modify the input sequence (the first step of the NEH-based scheduling algorithm). As shows the obtained results, the modification of the input sequence can significantly improve NEH results. Even though this issue has been studied in many works so far, still there is no sorting method that would provide significantly better results than sorting tasks based on their total production time. This work shows, however, that the sorting problem is very important for NEH-based algorithms and it is still an open issue.

TABLE 18. ARPD [%] for *N*-NEH, ToM+*N*-NEH (ToM+*N*), *vN*-NEH and ToM+*vN*-NEH (ToM+*vN*) algorithms on Taillard and VRF benchmarks.

Bench.	Algorithm	Length of list of candidate jobs							
		1	2	3	4	5	6	7	8
Taillard	*N*-NEH	3.33	3.18	3.10	3.22	3.34	3.23	3.21	3.22
	ToM+*N*	2.40	2.30	2.29	2.22	2.21	2.26	2.25	2.29
	vN-NEH	3.33	3.23	3.11	3.00	2.95	2.96	2.88	2.99
	ToM+*vN*	2.40	2.31	2.23	2.16	2.19	2.15	2.12	2.14
VRF S	*N*-NEH	3.84	3.75	3.85	4.00	3.97	3.93	3.94	3.90
	ToM+*N*	2.62	2.60	2.58	2.61	2.62	2.72	2.75	2.80
	vN-NEH	3.84	3.72	3.69	3.56	3.56	3.48	3.48	3.41
	ToM+*vN*	2.62	2.57	2.51	2.46	2.42	2.43	2.39	2.42
VRF L	*N*-NEH	3.33	3.06	2.90	2.80	2.73	2.70	2.65	2.61
	ToM+*N*	2.78	2.53	2.38	2.29	2.24	2.17	2.14	2.11
	vN-NEH	3.33	3.24	3.11	3.04	2.99	2.92	2.87	2.80
	ToM+*vN*	2.78	2.67	2.60	2.51	2.45	2.42	2.37	2.34

TABLE 19. ARPD [%] for *N*-NEH+, ToM+*N*+ and ToM+*vN*+ algorithms on Taillard and VRF benchmarks.

Benchmark	Algorithm	Length of list of candidate jobs			
		1	2	4	8
Taillard	*N*-NEH+	3.33	2.95	2.55	2.32
	ToM+*N*+	2.40	2.19	1.98	1.78
	vN-NEH+	3.33	2.97	2.62	2.23
	ToM+*vN*+	2.40	2.20	1.97	1.77
VRF S	*N*-NEH+	3.84	3.32	2.95	2.64
	ToM+*N*+	2.62	2.36	2.15	1.97
	vN-NEH+	3.84	3.36	2.95	2.47
	ToM+*vN*+	2.62	2.39	2.13	1.88
VRF L	*N*-NEH+	3.33	3.00	2.67	2.39
	ToM+*N*+	2.78	2.52	2.25	2.02
	vN-NEH+	3.33	3.13	2.88	2.60
	ToM+*vN*+	2.78	2.64	2.46	2.26

E. PRACTICAL AND RESEARCH IMPLICATIONS

The proposed approach has important practical and research implications, some of them are listed below:

- it enables to shorten production time – as shows the analysis of representative benchmarks, the proposed method has reduced the deviation of the jobs completion time (makespan) from the optimal solution by up to 50%,
- it enables to realize larger number of orders and thus increase the company's revenues – the ART.NEH measure indicates that it is possible to increase production by over 1% compared to scheduling with the NEH algorithm,
- it can be easily implemented into existing NEH-based algorithms (NEH-di, NEHD, NEHKK, NEHFF, etc.), also those used in real production environment,
- it can be useful for other researchers investigating different priority rules – here, we have only analyzed the TPT priority rule, but the method can be easily used with other priority rules (such as sum of average times and standard deviation used in NEHFF),
- it can be used to modify NEH-based algorithms for solving PFSP with other optimization criteria, e.g., minimization of total tardiness [7], minimization of total flow time [8], minimization of core waiting time and core idle time [1],
- it can be used to solve such problems as: blocking flow-shop scheduling problem (BFSP) [41], distributed heterogeneous hybrid flow shop lot-streaming scheduling problem (DHHFLSP) [35], permutation flow shop scheduling problem with multiple servers (PFSMS) [9],
- it can be used as an element of a hybrid approach where NEH-based algorithms are used to generate an initial solution, and then this initial solution is improved using, for example, local optimization algorithms (such as LS, ALNS, VNS),
- it can also be used to generate an initial population for population-based metaheuristics – the number of generated solutions is $m \cdot N \cdot S$, where m is the number of machines for a given instance of the problem, N is the length of the list of candidate jobs, and S is the parameter of the Swap method.

V. CONCLUSION

In this paper, we propose a new ToM+ method for modifying the input sequence in the NEH-based algorithms. The ToM+ method consists in excluding successive machines and not taking into account the operation time on a given machine when calculating the index used to sort the jobs in the initial phase in NEH-based algorithms.

A. RESULTS OF ToM+

The proposed method has been implemented for NEH, *N*-NEH and *vN*-NEH algorithms. Tests of the ToM+ method were carried out using the most popular benchmarks for the PFSP problem: the Taillard benchmark and the VRF benchmark. The proposed method enabled the performance (ARPD) of the analyzed algorithms to be improved, the obtained improvement ranges from 13.5% to nearly 35%. The amount of improvement achieved was strongly dependent on the benchmark analyzed. The ToM+ method increases the computational complexity of the algorithms by m times, where m is the number of machines for a given instance.

TABLE 20. Detailed results for Figure 1.

No.	Method	ARPD	ARD.NEH	ART.NEH	No.	Method	ARPD	ARD.NEH	ART.NEH
1	RAER	3.94	-0.56	2.62	21	NEHD-di	2.88	0.47	6.55
2	RAER-di	3.57	-0.2	5.32	22	SMα+(2)N+(2)	2.53	0.81	8.2
3	NEH	3.37	0	1	23	ToM+NEH	2.45	0.88	13.5
4	NEMR	3.21	0.15	4.11	24	SMα+(4)N+(2)	2.36	0.96	13.01
5	NEHKK2	3.14	0.22	1.15	25	N+(4)	2.6	0.74	6.18
6	NEHKK1-di	3.2	0.17	2.06	26	SMα+(8)N+(2)	2.26	1.06	22.73
7	NEH-di	3.08	0.28	2.01	27	ToM+vN+(2)	2.25	1.07	29.65
8	NEHR	3.1	0.26	2.78	28	vN+(8)	2.28	1.04	12.97
9	KKER	3.19	0.17	2.83	29	ToM+N+(2)	2.24	1.08	32.34
10	NEH1-di	3.15	0.21	2.05	30	SMα+(2)N+(4)	2.26	1.06	20.84
11	vN+(2)	3.02	0.34	2.16	31	ToM+vN+(4)	2.02	1.29	69.89
12	SMα+(2)NEH	2.82	0.52	3.55	32	SMα+(4)N+(4)	2.11	1.21	33.48
13	NEMR-di	3.01	0.34	7.04	33	N+(8)	2.36	0.96	17.96
14	SMα+(4)NEH	2.66	0.68	5.56	34	SMα+(8)N+(4)	2.01	1.31	59.14
15	NEHR-di	2.9	0.46	5.43	35	ToM+N+(4)	2.02	1.29	85.7
16	N+(2)	2.99	0.36	2.36	36	ToM+vN+(8)	1.81	1.49	180.61
17	KKER-di	2.91	0.44	5.58	37	SMα+(2)N+(8)	2.09	1.23	58.38
18	NEHFF	2.95	0.41	1.3	38	SMα+(4)N+(8)	1.96	1.35	95.03
19	SMα+(8)NEH	2.53	0.8	9.6	39	SMα+(8)N+(8)	1.86	1.44	169.5
20	vN+(4)	2.67	0.67	5.01	40	ToM+N+(8)	1.83	1.47	250.8

TABLE 21. Detailed results for Figure 2.

No.	Method	ARPD	ARD.NEH	ART.NEH	No.	Method	ARPD	ARD.NEH	ART.NEH
1	RAER	3.54	-0.13	2.04	21	NEHD-di	2.94	0.45	4.84
2	RAER-di	3.41	-0.01	3.98	22	SMα+(2)N+(2)	2.89	0.5	7.3
3	NEH	3.41	0	1	23	ToM+NEH	2.87	0.52	41.24
4	NEMR	3.3	0.1	3.56	24	SMα+(4)N+(2)	2.82	0.57	12.15
5	NEHKK2	3.29	0.12	1.03	25	N+(4)	2.75	0.64	6.41
6	NEHKK1-di	3.27	0.13	2.03	26	SMα+(8)N+(2)	2.74	0.65	21.84
7	NEH-di	3.27	0.14	2.02	27	ToM+vN+(2)	2.73	0.65	90.82
8	NEHR	3.24	0.16	2.15	28	vN+(8)	2.68	0.71	13.58
9	KKER	3.23	0.17	2.15	29	ToM+N+(2)	2.61	0.77	99.07
10	NEH1-di	3.23	0.17	2.01	30	SMα+(2)N+(4)	2.59	0.79	19.43
11	vN+(2)	3.21	0.19	2.19	31	ToM+vN+(4)	2.55	0.84	214.7
12	SMα+(2)NEH	3.2	0.2	3.05	32	SMα+(4)N+(4)	2.53	0.85	32.37
13	NEMR-di	3.14	0.26	5.6	33	N+(8)	2.47	0.91	19.28
14	SMα+(4)NEH	3.11	0.29	5.07	34	SMα+(8)N+(4)	2.47	0.91	58.22
15	NEHR-di	3.1	0.29	4.18	35	ToM+vN+(8)	2.34	1.04	561.92
16	N+(2)	3.08	0.32	2.39	36	ToM+N+(4)	2.33	1.04	264.48
17	KKER-di	3.08	0.32	4.15	37	SMα+(2)N+(8)	2.33	1.04	57.99
18	NEHFF	3.03	0.37	1.04	38	SMα+(4)N+(8)	2.27	1.1	96.67
19	SMα+(8)NEH	3.03	0.37	9.11	39	SMα+(8)N+(8)	2.22	1.15	173.98
20	vN+(4)	2.96	0.44	5.17	40	ToM+N+(8)	2.1	1.27	793.24

The proposed ToM+ method was combined with the SMα+ method. We want to underline that, according to our best knowledge, both methods are the first deterministic methods that modify the input sequence. It is also worth noting that it is possible to combine these two methods in the frames of one algorithm.

B. RESULTS OF ToM+ AND SMαP+

The proposed algorithms, combining ToM+ method, SMαP+, N-list, and vN-list technique, produced the best results among the deterministic constructive algorithms. Moreover, it is worth to underline that for the Taillard benchmark the proposed algorithm produced the better average relative percentage deviation (ARPD) than the FRB5, which is the best algorithm among FRB algorithms. The obtained results confirm the importance of the problem of creating input sequence. Thanks to the proposed modifications, it is possible to improve the results by up to nearly 50%. We believe that this confirms the importance of this problem for NEH-based algorithms.

C. FUTURE WORKS

An undoubted advantage of the ToM+ and SMα+ methods is that they produce the results (a family of good quality solutions) that can be used as an initial population, for example, in genetic algorithms. It is also worth noting the possibility of simple parallelization of the calculations performed by the ToM+ method, i.e. running a given algorithm for different input sequences, which resulted from the ToM+ method. Ease of implementation, the ability to parallelize calculations and the population-based nature of the resulting solution are undoubted advantages of the ToM+ method. We plan to analyze the effectiveness of using the created populations of solutions (described in the previous step) for genetic algorithms as an initial population. We also plan to implement methods that employ local optimization of the partial sequence. Our last but not least goal is to analyze the possibilities to improve the results of NEH but various other modifications of the input sequence.

APPENDIX

Tables 13-14 show the values of ARD.NEH measure. Tables 15-16 include ACPU values of considered methods. Tables 18 and 19 show the ARPD values of the considered methods (computed based on the results published by Eric Taillard on his homepage and the results from [39]). Tables 20 and 21 provide the detailed results corresponding to Figures 1 and 2.

REFERENCES

[1] A. Alfieri, M. Garraffa, E. Pastore, and F. Salassa, "Permutation flowshop problems minimizing core waiting time and core idle time," *Comput. Ind. Eng.*, vol. 176, Feb. 2023, Art. no. 108983.

[2] C. E. Dodu and M. Ancău, "A tabu search for time minimization in printed circuit board assembly," *Academic J. Manuf. Eng.*, vol. 19, no. 2, pp. 5–10, 2021.

[3] X. Dong, H. Huang, and P. Chen, "An improved NEH-based heuristic for the permutation flowshop problem," *Comput. Oper. Res.*, vol. 35, no. 12, pp. 3962–3968, Dec. 2008.

[4] G. Erseven, G. Akgün, A. Karakaş, G. Yarıkcan, Ö. Yücel, and A. Öner, "An application of permutation flowshop scheduling problem in quality control processes," in *Proc. Int. Symp. Prod. Res.*, N. M. Durakbasa and M. G. Gencyilmaz, Eds. Cham, Switzerland: Springer, 2018, pp. 849–860.

[5] V. Fernandez-Viagas and J. M. Framinan, "On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 45, pp. 60–67, May 2014.

[6] V. Fernandez-Viagas, R. Ruiz, and J. M. Framinan, "A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation," *Eur. J. Oper. Res.*, vol. 257, no. 3, pp. 707–721, Mar. 2017.

[7] V. Fernandez-Viagas and J. M. Framinan, "NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness," *Comput. Oper. Res.*, vol. 60, pp. 27–36, Aug. 2015.

[8] V. Fernandez-Viagas, P. Perez-Gonzalez, and J. M. Framinan, "The distributed permutation flow shop to minimise the total flowtime," *Comput. Ind. Eng.*, vol. 118, pp. 464–477, Apr. 2018.

[9] V. Fernandez-Viagas, L. Sanchez-Mediano, A. Angulo-Cortes, D. Gomez-Medina, and J. M. Molina-Pariente, "The permutation flow shop scheduling problem with human resources: MILP models, decoding procedures, NEH-based heuristics, and an iterated greedy algorithm," *Mathematics*, vol. 10, no. 19, p. 3446, Sep. 2022.

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

[11] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.

[12] J. Gmys, "Exactly solving hard permutation flowshop scheduling problems on peta-scale GPU-accelerated supercomputers," *Informs J. Comput.*, vol. 34, no. 5, pp. 2502–2522, Sep. 2022.

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

[14] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," in *Discrete Optimization II* (Annals of Discrete Mathematics), vol. 5, P. L. Hammer, E. L. Johnson, and B. H. Korte, Eds. Amsterdam, The Netherlands: Elsevier, 1979, pp. 287–326.

[15] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Nav. Res. Logistics Quart.*, vol. 1, no. 1, pp. 61–68, Mar. 1954.

[16] P. Kalczynski and J. Kamburowski, "On the NEH heuristic for minimizing the makespan in permutation flow shops," *Omega*, vol. 35, no. 1, pp. 53–60, Feb. 2007.

[17] P. J. Kalczynski and J. Kamburowski, "An improved NEH heuristic to minimize makespan in permutation flow shops," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 3001–3008, Sep. 2008.

[18] P. J. Kalczynski and J. Kamburowski, "An empirical analysis of the optimality rate of flow shop heuristics," *Eur. J. Oper. Res.*, vol. 198, no. 1, pp. 93–101, Oct. 2009.

[19] A. Khare and S. Agrawal, "Effective heuristics and metaheuristics to minimise total tardiness for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 59, no. 23, pp. 7266–7282, Dec. 2021.

[20] C. T. D. Laurentys and M. S. Nagano, "Evaluating procedures in the NEH heuristic for the PFSP-SIST," *J. Project Manage.*, vol. 9, no. 1, pp. 17–26, 2024.

[21] W. Liu, Y. Jin, and M. Price, "A new improved NEH heuristic for permutation flowshop scheduling problems," *Int. J. Prod. Econ.*, vol. 193, pp. 21–30, Nov. 2017.

[22] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 754–768, Apr. 2010.

[23] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, Jan. 1983.

[24] M. A. H. Newton, V. Riahi, K. Su, and A. Sattar, "Scheduling blocking flowshops with setup times via constraint guided and accelerated local search," *Comput. Oper. Res.*, vol. 109, pp. 64–76, Sep. 2019.

[25] R. Puka, J. Duda, A. Stawowy, and I. Skalna, "N-NEH+ algorithm for solving permutation flow shop problems," *Comput. Oper. Res.*, vol. 132, Aug. 2021, Art. no. 105296.

[26] R. Puka, J. Duda, and A. Stawowy, "Input sequence of jobs on NEH algorithm for permutation flowshop scheduling problem," *Manage. Prod. Eng. Rev.*, vol. 1, no. 13, pp. 32–43, Mar. 2022.

[27] R. Puka, I. Skalna, and T. Derlecki, "New measures of algorithms quality for permutation flow-shop scheduling problem," in *Proc. Ann. Comput. Sci. Inf. Syst.*, vol. 35, Sep. 2023, pp. 1107–1111.

[28] R. Puka, I. Skalna, J. Duda, and A. Stawowy, "Deterministic constructive vN-NEH+ algorithm to solve permutation flow shop scheduling problem with makespan criterion," *Comput. Oper. Res.*, vol. 162, Feb. 2024, Art. no. 106473.

[29] R. Puka, I. Skalna, and B. Lamasz, "Swap method to improve N-NEH+ algorithm," in *Proc. Int. Conf. Electr., Comput. Energy Technol. (ICECET)*, Jul. 2022, pp. 1–6.

[30] R. Puka and B. Łamasz, "Impact of input sequence on N-NEH+ algorithm," in *Proc. 10th Carpathian Logistics Congr. (CLC)*, 2022, pp. 191–196.

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

[32] V. Riahi, M. Khorramizadeh, M. A. Hakim Newton, and A. Sattar, "Scatter search for mixed blocking flowshop scheduling," *Expert Syst. Appl.*, vol. 79, pp. 20–32, Aug. 2017.

[33] I. Ribas, R. Companys, and X. Tort-Martorell, "Comparing three-step heuristics for the permutation flow shop problem," *Comput. Oper. Res.*, vol. 37, no. 12, pp. 2062–2070, Dec. 2010.

[34] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *Eur. J. Oper. Res.*, vol. 165, no. 2, pp. 479–494, Sep. 2005.

[35] W. Shao, Z. Shao, and D. Pi, "Modelling and optimization of distributed heterogeneous hybrid flow shop lot-streaming scheduling problem," *Expert Syst. Appl.*, vol. 214, Mar. 2023, Art. no. 119151.

[36] Z. Shao, W. Shao, and D. Pi, "Effective constructive heuristic and iterated greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem," *Knowl.-Based Syst.*, vol. 221, Jun. 2021, Art. no. 106959.

[37] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *Eur. J. Oper. Res.*, vol. 47, no. 1, pp. 65–74, Jul. 1990.

[38] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, Jan. 1993.

[39] E. Vallada, R. Ruiz, and J. M. Framinan, "New hard benchmark for flowshop scheduling problems minimising makespan," *Eur. J. Oper. Res.*, vol. 240, no. 3, pp. 666–677, Feb. 2015.

[40] Y. Wang, Y. Wang, Y. Han, J. Li, K. Gao, and Y. Nojima, "Intelligent optimization under multiple factories: Hybrid flow shop scheduling problem with blocking constraints using an advanced iterated greedy algorithm," *Complex Syst. Model. Simul.*, vol. 3, no. 4, pp. 282–306, Dec. 2023.

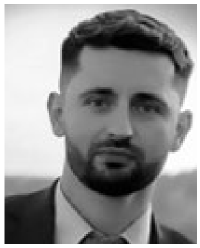
[41] Q. Wu, Q. Gao, W. Liu, and S. Cheng, "Improved NEH-based heuristic for the blocking flow-shop problem with bicriteria of the makespan and machine utilization," *Eng. Optim.*, vol. 55, no. 3, pp. 399–415, Mar. 2023.

[42] K.-C. Ying and S.-W. Lin, "A high-performing constructive heuristic for minimizing makespan in permutation flowshops," *J. Ind. Prod. Eng.*, vol. 30, no. 6, pp. 355–362, Sep. 2013.

[43] J. Zhang, S. D. Dao, W. Zhang, M. Goh, G. Yu, Y. Jin, and W. Liu, "A new job priority rule for the NEH-based heuristic to minimize makespan in permutation flowshops," *Eng. Optim.*, vol. 55, no. 8, pp. 1296–1315, Aug. 2023.

RADOSŁAW PUKA received the Ph.D. degree in industrial engineering from the AGH University of Science and Technology, Poland, in 2019. He has been involved in the development of novel algorithms for a number of manufacturing companies. His research interests include design and improvement of algorithms, and use of AI to support decision making.

IWONA SKALNA (Member, IEEE) is currently a Researcher and a Lecturer with the Department of Applied Computer Science, AGH University of Krakow, Poland. She is also a Professor of computer science. She is the author of more than 100 scientific publications, including several monographs and articles in prestigious international journals. Her research interests include modeling of uncertainty, scheduling optimization, and artificial intelligence. She is a member of the Polish Information Processing Society and GDR MACS.



BARTOSZ ŁAMASZ received the Ph.D. degree in management science from the AGH University of Science and Technology, Poland, in 2017. He is currently an Assistant Professor with the Department of Management, AGH University of Krakow. His research interests include price risk management and commodity options market. He also works on issues related to applications of decision rules and machine learning to decision-making in financial markets.

JERZY DUDA is currently a Professor with the Department of Business Informatics and Managerial Engineering, Faculty of Management, AGH University of Krakow. He was involved in many projects on information technology and production management. His research interests include computational intelligence in production management, heuristic algorithms, single and multi-criteria optimization, advanced planning and scheduling systems, and uncertainty modeling.

ADAM STAWOWY received the M.S. degree in management science and the Ph.D. degree in economics from the AGH University of Science and Technology (UST), Cracow, Poland. He is currently a Professor with the Department of Business Informatics and Management Engineering, AGH University of Krakow. His current research interests include the application of heuristic algorithms and meta-heuristic to solve scheduling and classification problems.