Journal of Project Management 9 (2024) 17–26

Contents lists available at GrowingScience

Journal of Project Management

homepage: www.GrowingScience.com

Evaluating procedures in the NEH heuristic for the PFSP - SIST

Clarissa Tararam de Laurentys[a*] and Marcelo Seido Nagano[a*]

[a]*Department of Production Engineering, University of São Paulo, 400, Avenue Trabalhador São Carlense, São Carlos, São Paulo, Brazil*

C H R O N I C L E

Article history:
Received: February 9, 2023
Received in revised format: April 28, 2023
Accepted: September 14, 2023
Available online: September 14, 2023

Keywords:
Flow shop scheduling
Sequence-independent setup time
NEH heuristic
Makespan

A B S T R A C T

The development and assessment of 48 heuristics for the sequence-independent setup time permutation flow shop problem (PFSP-SIST) are presented in this article. This contribution combines four tie-breaking solutions with twelve priority rules for the NEH heuristic fourth and first stage, respectively. Heuristics are evaluated on Ruiz and Allahverdi (2007) benchmark problem instances, that covers small, medium and large-size problems. The popular accelerations of Taillard were used in all tests, which were adapted to the sequence-independent setup time constraint. The aim is to reduce the longest job completion time, which is also referred to as makespan. Computation results show that using different tie-breaking strategies has a greater impact on performance than using different priority rules. The heuristics that presented the best results in relatively low computation time are those that use the FFs tie-breaking strategy procedure to the sequence-independent setup time problem.

1. Introduction

A scheduling problem consists of finding a sequence of jobs (goods and services) that will be processed in a set of machines (equipment, employees, machines themselves) with the objective of minimizing a certain objective function (Baker 1974; Graham et al. 1979). One of the most studied production environments since the 1950s is the Permutation Flow Shop Problem, also known as PFSP (Johnson 1954; Maccarthy & Liu 1993; Reza Hejazi & Saghafian 2005). This problem consists of sequencing *n* jobs that must be processed in m available machines, going through the same processing order and, on each machine, obeying the First In, First Out (FIFO) rule.

The makespan (Cmax) has been one of the most adopted performance measures in flow shop environments (Hejazi & Saghafian 2005). The attention given by researchers to this measure expresses its importance in the scheduling decision process. Several heuristics have been developed to minimize makespan in permutation flow shop problems. The NEH (Nawaz, Enscore, & Ham 1983) heuristic is considered the most effective algorithm for solving PFSP (Liu, Jin & Price 2016).

A more realistic situation for the PFSP is the incorporation of setup time. Setup time or preparation time comprises the time necessary to prepare the machine to process a given task. This includes getting tools, placing materials between machines, returning tools, cleaning, adjusting tools, and materials inspection.

The incorporation of setup time in scheduling problems has been studied since the 1960s. In practice, setup incorporation directly impacts on better resource utilization. The benefits of reducing setup time include cost reduction, increased production speed, reduced lead times, greater agility in the process of changing tools and delivering the product to the customer, increasing their satisfaction (Allahverdi et al. 2008). As Belabid, Aqil and Allali (2020) point out: “setup time may be the subject of several cases such as assembly and disassembly of parts or tools in the machine, cleaning, evacuation of production means, and so on. These activities can be carried out by robots or manipulators independent of the processed jobs in the machine”.

* Corresponding author.
E-mail address: clarissalaurentys@gmail.com (C. T. Laurentys) and drnagano@usp.br (M. S. Nagano)

doi: 10.5267/j.jpm.2023.11.001

When the duration of setup time depends solely on the current job being processed, it is considered sequence-independent. Conversely, if the setup time duration is contingent on the immediately preceding job as well, it is referred to as sequence-dependent setup time. (Allahverdi et al. 2008). Research on sequence-dependent setup time is more present in the literature when compared to sequence-independent setup time. Therefore, the development of research related to the latter is necessary to reduce this gap and increase the understanding of the different scheduling conditions.

In this perspective, this article presents the study of the PFSP with the SIST constraint, a crucial production scenario in the context of emerging industrial technologies. (Belabid, Aqil & Allali 2020). The objective is to propose new heuristics, obtained through adaptations of the NEH to include the sequence-independent setup time constraint. The performance measure adopted is the makespan.

The article is organized into 5 sections. Section 2 presents the materials and methods with all proposed approaches for solving the PFSP-SIST problem. The results are presented in section 3 and section 4 presents the conclusion.

2. Materials and Methods

One of the key heuristics for resolving the PFSP with n jobs and m machines is the NEH heuristic (Belabid, Aqil & Allali 2020). The NEH mainly consists of five steps. First is calculating the priority rule for all list of jobs and second is sorting jobs in descending order. Third is to calculate makespan for the two potential sequences to determine which order is best for the first and second jobs on the list. The fourth step involves determining the optimal arrangement for the job at the i-th index within the list, maintaining the relative positions of the previously designated jobs, by placing it at each potential i position in the partial sequence. Step 5 is a stop criterion, if $n = i$ the process stops, if not set $i = i + 1$ and start at the fourth step.

Taillard (1990) proposed an alternative method for computing partial makespan, which reduced the NEH algorithm's complexity from $O(n^3m)$ to $O(n^2m)$. This well-known procedure is adapted to the PFSP-SIST problem by including the setup time in the calculation of the following parameters, being the setup and process time of job j in machine i respectively represented by $s_{i,j}$ and $p_{i,j}$:

- Job j earliest completion time on machine *i*, Eq. (1);
- Tail, Equation (2);
- Job k on position *j* earliest relative completion time on machine *i*, Eq. (3);
- Partial makespan, when job *k* is added at position *j*, Eq. (4).

$$e_{i,j} = \max\left(e_{i,j-1} + s_{i,j},\ e_{i-1\,j}\right) + p_{i,j}\ ,$$
$$e_{0\,j} = e_{i0} = 0;\ i = 1,\ldots,m;\ j = 1,\ \ldots,\ k-1 \quad (1)$$

$$q_{i,j} = \max\left(q_{i,j+1} + s_{i,j+1},\ q_{i+1,j}\right) + p_{i,j}\ ,$$
$$q_{i,k} = q_{m+1,j} = 0;\ i = m,\ldots,1;\ j = k-1,\ \ldots,\ 1 \quad (2)$$

$$f_{i,j} = \max\left(e_{i,j-1} + s_{i,k},\ f_{i-1,j}\right) + p_{k,j}\ ,$$
$$f_{i,0} = 0;\ i = 1,\ldots,m;\ j = 1,\ \ldots,\ k \quad (3)$$

$$M_j = max_i\left(f_{i,j} + q_{i,j} + s_{i,j}\right)$$
$$i = 1,\ldots,m;\ j = 1,\ \ldots,\ k \quad (4)$$

A large number of authors developed adapted versions of the NEH heuristic to follow the constraints of their specific problem. In the following sub-sections, the selected adapted versions of the NEH to this study are explained.

2.1 Priority Rule (PR)

Job priority rules have been developed by sorting methods so that they can be independently sorted based on their characteristics (Zhang et al., 2023). Originally, in the NEH's first step, all jobs are sequenced based on the priority rule of total process time.

To this article, which aims to propose new heuristics through adaptations of the NEH heuristic, various priority rules were assessed. The research carried out by Dong et al. (2008) and Liu et al. (2017) indicated that other priority rules exhibited enhanced performance.

Dong, et al. (2008) presented three different priority rules. The first is denoted as Avg and it refers to the method of arranging jobs based on each job's average process time. This method yields the same outcome as the NEH's. The second method, Dev, denotes the approach of ordering jobs based on the standard deviation of process times. The last method, AvgDev, implies arranging jobs based on the combined sum of the average and standard deviation of processing times. Based on the obtained results, the Dev priority rule performed less favorably compared to the other two cases. Furthermore, there was no statistically significant difference in the use of the AvgDev and Avg priority rules, even though the overall performance of the former was superior to that of the latter.

Therefore, in this article, the Avg and AvgDev rules were adapted to the PFSP-SIST problem by adding the original metrics to those calculated for the setup time. The Avg priority rule is evaluated as the representation of the NEH original rule and the AvgDev is evaluated to verify if its best performance is achieved even with the addition of the setup constraint.

Liu et al. (2017) introduced an additional priority rule, SKE, where jobs are ordered by the non-increasing sum of AVG, STD, and job j's absolute value of the skewness, abs(SKEj), which measures the asymmetry of a probability distribution. This new addition allowed for the resolution of ties that were common when sorting, since various jobs could have identical averages and standard deviations. The results obtained from this newly proposed SKE priority rule demonstrated superior performance compared to the AvgDev rule proposed by Dong et al. (2008). To assess its performance in solving the PFSP-SIST problem, the SKE rule was adapted by including the original metric with the abs(SKEj) calculated for the setup time.

In order to test the behaviour of these metrics in the PFSP-SIST problem, all possible combinations between them were tested as priority rules. For example, ordering activities by the sum of Avg process time and AvgDev setup time. Furthermore, a new metric for setup time was tested, the maximum setup time for job j. This new metric can help prioritize jobs with larger setup values in case of ties. Hence, for this article, a total of 12 priority rules were evaluated. These priority rules are defined in Table 1.

2.2 Tie-breaking Strategy (TB)

The insertion of a new job in different positions at step 4 can generate the same partial makespan. Thus, several tie-breaking strategies were developed and compared with the original (TB_{NEH}), in which, in the event of a tie, the first partial sequence, that is, the initial insertion position, is preserved.

For the purpose of this article, three different tie-breaking strategies were evaluated that presented superior performance than TB_{NEH}. These tie-breaking strategies were proposed in the studies conducted by Dong, et al. (2008), Fernandez-Viagas and Framinan (2014) and Ribas et al. (2010).

Dong et al. (2008) presented a strategy based on choosing the location that would most likely achieve balance in the utilization of all machines, TB_D. For this, the measures $E_j(x)$ and $D_j(x)$ are calculated and, in case of ties, the partial sequence that presents the minimum D is chosen. The results indicate that the newly implemented tie-breaking strategy is, in fact, more efficient than the initial NEH approach. For the PFSP-SIST problem, these measures were adapted according to Eq. (5) and Eq. (6), being $L_{i,j}$ job j latest feasible start time on machine *i*.

$$E_j = \frac{1}{m} \times \sum_{i=1}^{m} \left(\frac{p_{i,j} + s_{i,j}}{L_{i,j} - e_{i,j-1}} \right) . \; j = 1.,,,.n \tag{5}$$

$$D_j = \sum_{i=1}^{m} \left(\frac{p_{i,j} + s_{i,j}}{L_{i,j} - e_{i,j-1}} - E_j \right)^2 . \; j = 1.,,,.n \tag{6}$$

Fernandez-Viagas et al. (2017) proposed a strategy substantiated on minimizing machine idle time, TB_{FF}. This proposal unties the partial sequences by choosing the position that presents the minimum idle time, $IT(l)$. The experimental results show TB_{FF} to be better than TB_D and TB_{NEH} mechanisms. Hence, it is adapted to PFSP-SIST problem according to Eq. (7), being $e_{i,j}$ and $f_{i,j}$ calculated according to Eq. (1) and Eq. (3), respectively, and $f'_{i,l}$ the job's completion time on machine i, prior to being inserted at position l, calculated as per Eq. (8).

$$IT(l) = \sum_{i=1}^{m} (f_{i,l} - e_{i,l} + p_{i,l} + s_{i,l} + \max\{f'_{i,l} - f_{i,l}, 0\}) \; , For \; l = k, IT(l) = \sum_{i=1}^{m} (f_{i,l} - e_{i,l-1}) \tag{7}$$

$$f'_{i,l} = max\{f_{i,l} + s_{i,l}, f'_{i-1,l}\} + p_{i,l} \; , i = 1.,,,.m; \; f'_{0.l} = 0 \tag{8}$$

Ribas et al. (2010) proposed another tie-breaking strategy, TB_{RTC}, based on two tie-breaking methods. Better results are achieved using TB_D when exclusively applied to the direct instance, but still TB_{RTC} obtained better results than the original tie-breaking strategy, TB_{NEH}. The primary tie-breaking method from TB_{RTC} is designed to minimize the overall machine idle time, adapted to the PFSP-SIST problem in the Eq. (9) and Eq. (10). In case of remaining ties after the first strategy, a second one is applied, proposed by Kalczynski and Kamburowski (2008). The job is inserted at the position closest to the start of the partial sequence if $a_j \leq b_j$, otherwise it is the one closest to the last position. These metrics, a_j and b_j, were adapted to the PFSP-SIST problem according to Eq. (11) and Eq. (12).

$$IT_i = f_{i,n} - e_{i,1} - \sum_{j=1}^{n} p_{i,j} - \sum_{i=1}^{m} s_{i,j} \tag{9}$$

$$f_{i,j} = e_{i,j} + p_{i,j} + s_{i,j} \; , i = 1, \ldots, m; \; j = 1, \ldots, n; \; f_{0,j} = f_{i,0} = 0 \tag{10}$$

$$a_j = \sum_{i=1}^{m} \left(\left((m-1) \times \frac{m-2}{2} + m - i \right) \times (p_{i,j} + s_{i,j}) \right) \quad (11)$$

$$b_j = \sum_{i=1}^{m} \left(\left((m-1) \times \frac{m-2}{2} + i - 1 \right) \times (p_{i,j} + s_{i,j}) \right) \quad (12)$$

In summary, four tie-breaking strategies were selected and adapted to evaluate their impact on the PFSP-SIST problem: TB_{NEHs}, TB_{Ds}, TB_{FFs} and TB_{RTCs}, the suffix "s" was added to represent the setup constraint.

2.3 Performance metrics

Employing Taillard's Acceleration adaptation, the 48 heuristics were tested, which derived from combining 12 priority rules and 4 tie-breaking strategies for the NEH heuristic first and fourth stage, respectfully. Ruiz and Allahverdi (2007) benchmark were used to compare the heuristics because it is more exhaustive symmetric and, therefore, possesses greater discriminant power than Taillard's benchmark (1993). It includes a total of 5,400 instances from two sets. The "small" set contains 3,000 instances, and the "large" set 2,400. The "small" one comprises all combinations of problem instances where n = {15, 20, 25, 30} and m = {2, 3, 4, 5, 6} and for the "large" one n = {50, 100, 150, 200} and m = {10, 20, 30, 40}. As Ruiz and Allahverdi (2007) points out:

"For each combination of n and m, there are six different combinations of distributions of processing and setup times. Processing times are uniformly distributed in the range [1,10] or [1,100]. It is well known in the scheduling literature that uniformly distributed processing times result in instances harder to solve and that the range in which the processing times are distributed has an influence over solution methods (see Watson et al. 2002 for a complete study). Setup times are uniformly distributed so that the maximum setup is 50%, 100% and 150% of the maximum processing time, respectively. Zero duration setup times are allowed. This results in setup times being uniformly distributed in [0, 5], [0, 10], and [0, 15] in the case where processing times are distributed in the range [1, 10], and in [0, 50], [0, 100], and [0, 150] in the case processing times are distributed in the range [1, 100]. As with the distributions of the processing times, having small or large setup times influences algorithm performance as was shown in Ruiz et al. (2005)".

Also, 25 replicates were obtained for each situation which results in the total of 5,400 instances tested for each of the 48 heuristics. The performance was evaluated using the ARPD and ARPT criteria. The ARPD represents the average of Relative Percentage Deviation, denoted as RPD, calculated using Equation (13). It is evident that a lower RPD value indicates better heuristic performance, signifying that its solutions closely align with the best result among all the compared methods (Rossi & Nagano, 2019).

$$RPD\left(C_{max}(H_i)\right) = 100 \times \left(\frac{C_{max}(H_i) - C^*_{max}}{C^*_{max}} \right); i = \{1,2,\dots,48\} \quad (13)$$

where $C_{max}(H_i)$ refers to heuristic H_i's job sequence's makespan and C^*_{max} represents the optimal solution from all the heuristics compared. ARPT is the average Relative Percentage Computational Time, called RPT, calculated according to Eq. (14) and Eq. (15). The ARPT measures the computational effort of the methods, thus, the lower its value, smaller the computational effort required to execute the method.

$$ACT_t = \frac{\sum_{h=1}^{H}(T_{th})}{H};\ t = \{1,\dots,T\};\ h = \{1,\dots,H\} \quad (14)$$

$$RPT_{th} = \frac{T_{th} - ACT_t}{ACT_t} + 1;\ t = \{1,\dots,T\};\ h = \{1,\dots,H\}, \quad (15)$$

Being:

T_{th}: heuristic h's CPU time on instance t;

ACT_t: average CPU time from all heuristics for instance t;

H: total of evaluated heuristics;

T: total number of instances.

3. Results

The results indicate that the impact on the heuristic's performance is mainly a result of the choice of the tie-breaking strategy. Table 2 presents a summary of the results from the 48 heuristics. Independently of the priority rule used in the first step, a pattern is clear. Heuristics that used TB_{NEHs} showed the worst ARPD results, followed by those that used TB_{Ds}. The best results are either obtained by TB_{FFs} or by TB_{RTCs}, depending on the chosen priority rule and the process and setup times' distributions. TB_{FFs} heuristics showed the best ARPD at 150% setup distribution and average ARPD results for almost all

cases, except when combined with PR_6, PR_{10}, PR_{11} and PR_{12}. In contrast, TB_{RTCs} heuristics showed the best ARPD results in most cases at 50% and 100% setup distribution.

Furthermore, considering only the average CPU time and ARPT results, another pattern is clear. In all cases, TB_{NEHs} showed the best results, followed by TB_{FFs}. TB_{RTCs} had high average CPU time and ARPT due to its complexity, but in all cases, TB_{Ds} showed the worst results. Therefore, although TB_{RTCs} and TB_{FFs} showed very similar ARPD results, TB_{FFs} were notable for using significantly less CPU time.

The results summarized in Table 2 were plotted in the scatter chart presented in Figure 1. The scatter chart also shows that the impact on the heuristic's performance is mainly a result of the choice of the tie-breaking strategy. Four zones are evidently formed when comparing the ARPD and ARPT values. Each zone is composed by combining the 12 priority rules with one of the tie-breaking strategies.

The first zone is composed by the heuristics that used the TB_{NEHs}, represented in detail in Fig. 2. These heuristics presented the lowest ARPT values, between 0.472 and 0.479, and the highest ARPD values, between 0.412 and 0.435. The second zone is composed by the heuristics that used the TB_{Ds}, represented in detail in Fig. 3. These heuristics presented the highest ARPT values, between 1.696 and 1.779, and second highest ARPD values, between 0.365 and 0.381.

The third zone is composed by the heuristics that used the TB_{RTCs}, represented in detail in Fig. 4. These heuristics presented the second highest ARPT values, between 1.206 and 1.264, and low ARPD results of ARPD, between 0.257 and 0.271. The fourth zone is composed by the heuristics that used the TB_{FFs}, represented in detail in Fig. 5. These heuristics presented the second lowest ARPT values, between 0.538 and 0.546, and low results of ARPD, between 0.253 and 0.272. Therefore, the zones ordered by ARPD values from highest to lowest are: $TB_{NEHs} > TB_{Ds} > TB_{RTCs} >= TB_{FFs}$. And the order by ARPT values from highest to lowest is: $TB_{Ds} > TB_{RTCs} > TB_{FFs} > TB_{NEHs}$.

Although it was not possible to identify a clear pattern between the use of priority rules and the performance of the heuristic, some conclusions can be drawn by comparing the impact of each priority rule within the same tiebreaker strategy. These impacts can be seen in Figures 2, 3-5. The priority rules that impacted negatively the ARPD and ARPT are PR_9 and PR_4, respectively. PR_{10} impacted negatively on both. All heuristics that used PR_9 had the worst ARPD results, except when combined with TB_{FFs}. In this case, it presented the third worst result, 0.269, very close to the worst one 0.272. PR_4, when combined with TB_{Ds} and TB_{RTCs}, had the worst ARPT results and, when combined with TB_{FFs} and TB_{NEHs}, had the second worst ARPT results, differing from the worst by only 0.001.

PR_{10} also showed high results of ARPD when combined with TB_{NEHs}, TBD_s and TB_{FFs}. And high results of ARPT when combined with TB_{NEHs}, TB_{Ds} and TB_{RTCs}. In contrast, the priority rules that positively impacted the ARPD and ARPT are PR_2 and PR_5. PR_2 combined with TB_{NEHs} and TB_{Ds} had the best ARPD results and PR_5 combined with TB_{Ds}, TB_{FFs} and TB_{RTCs} had the best ARPT results.

Table 1
Adapted and developed priority rules for the PFSP-SIST.

Priority Rule	Definition
PR_1	$AvgDev_{proc} + AvgDev_{setup}$
PR_2	$AvgDev_{proc} + Avg_{setup}$
PR_3	$Avg_{proc} + AvgDev_{setup}$
PR_4	$AvgDev_{proc} + max_{setup}$
PR_5	$AVG_{proc} + AVG_{setup}$
PR_6	$Avg_{proc} + max_{setup}$
PR_7	$AvgDev_{proc} + AvgDev_{setup} + SKE_{proc} + SKE_{setup}$
PR_8	$AvgDev_{proc} + Avg_{setup} + SKE_{proc} + SKE_{setup}$
PR_9	$Avg_{proc} + AvgDev_{setup} + SKE_{proc} + SKE_{setup}$
PR_{10}	$AvgDev_{proc} + max_{setup} + SKE_{proc} + SKE_{setup}$
PR_{11}	$Avg_{proc} + Avg_{setup} + SKE_{proc} + SKE_{setup}$
PR_{12}	$Avg_{proc} + max_{setup} + SKE_{proc} + SKE_{setup}$

Table 2
Summary of the results obtained by each of the 48 heuristics.

Algorithm			ARPD			Average ARPD	Average CPU time (ms)	ARPT
			50%	100%	150%			
Alg_1	PR_1	TB_{NEHs}	0,434	0,394	0,424	0,417	**3000**	**0,475**
Alg_2	PR_1	TB_{Ds}	0,371	0,388	0,378	0,379	13364	1,76
Alg_3	PR_1	TB_{FFs}	**0,265**	**0,263**	**0,262**	**0,263**	3125	0,54
Alg_4	PR_1	TB_{RTCs}	**0,265**	0,269	0,270	0,268	9782	1,254
Alg_5	PR_2	TB_{NEHs}	0,421	0,416	0,398	0,412	**3001**	**0,478**
Alg_6	PR_2	TB_{Ds}	0,369	0,368	0,357	0,365	13179	1,742
Alg_7	PR_2	TB_{FFs}	0,271	**0,262**	**0,263**	**0,266**	3123	0,54
Alg_8	PR_2	TB_{RTCs}	**0,249**	0,263	0,285	**0,266**	9796	1,246
Alg_9	PR_3	TB_{NEHs}	0,43	0,401	0,414	0,415	**3001**	**0,478**
Alg_{10}	PR_3	TB_{Ds}	0,389	0,374	0,363	0,375	13091	1,732
Alg_{11}	PR_3	TB_{FFs}	**0,253**	0,267	**0,271**	**0,264**	3125	0,54
Alg_{12}	PR_3	TB_{RTCs}	0,267	**0,265**	**0,271**	0,268	9695	1,229
Alg_{13}	PR_4	TB_{NEHs}	0,438	0,405	0,428	0,424	**3000**	**0,478**
Alg_{14}	PR_4	TB_{Ds}	0,363	0,38	0,375	0,373	13506	1,779
Alg_{15}	PR_4	TB_{FFs}	0,257	0,271	**0,251**	**0,26**	3126	0,545
Alg_{16}	PR_4	TB_{RTCs}	**0,253**	**0,254**	0,283	0,263	9933	1,264
Alg_{17}	PR_5	TB_{NEHs}	0,438	0,413	0,394	0,415	**2999**	**0,477**
Alg_{18}	PR_5	TB_{Ds}	0,383	0,381	0,345	0,37	12920	1,696
Alg_{19}	PR_5	TB_{FFs}	0,282	0,274	**0,246**	**0,267**	3120	0,538
Alg_{20}	PR_5	TB_{RTCs}	**0,262**	**0,266**	0,273	**0,267**	9508	1,206
Alg_{21}	PR_6	TB_{NEHs}	0,442	0,391	0,416	0,417	**2999**	**0,477**
Alg_{22}	PR_6	TB_{Ds}	0,369	0,379	0,377	0,375	13434	1,743
Alg_{23}	PR_6	TB_{FFs}	**0,253**	0,275	0,288	0,272	3126	0,541
Alg_{24}	PR_6	TB_{RTCs}	0,256	**0,242**	**0,272**	**0,257**	9760	1,236
Alg_{25}	PR_7	TB_{NEHs}	0,446	0,406	0,432	0,428	**3000**	**0,475**
Alg_{26}	PR_7	TB_{Ds}	0,381	0,355	0,368	0,368	13399	1,765
Alg_{27}	PR_7	TB_{FFs}	0,266	**0,25**	**0,244**	**0,253**	3127	0,546
Alg_{28}	PR_7	TB_{RTCs}	**0,261**	0,261	0,276	0,266	9777	1,251
Alg_{29}	PR_8	TB_{NEHs}	0,447	0,435	0,414	0,432	**2999**	**0,477**
Alg_{30}	PR_8	TB_{Ds}	0,377	0,375	0,375	0,376	13242	1,746
Alg_{31}	PR_8	TB_{FFs}	**0,256**	0,273	**0,242**	**0,257**	3124	0,543
Alg_{32}	PR_8	TB_{RTCs}	0,263	**0,253**	0,275	0,264	9702	1,243
Alg_{33}	PR_9	TB_{NEHs}	0,462	0,416	0,425	0,435	**2998**	**0,472**
Alg_{34}	PR_9	TB_{Ds}	0,399	0,376	0,369	0,381	13153	1,729
Alg_{35}	PR_9	TB_{FFs}	**0,264**	0,284	**0,258**	**0,269**	3123	0,54
Alg_{36}	PR_9	TB_{RTCs}	0,281	**0,26**	0,272	0,271	9764	1,226
Alg_{37}	PR_{10}	TB_{NEHs}	0,441	0,421	0,426	0,43	**3000**	**0,479**
Alg_{38}	PR_{10}	TB_{Ds}	0,382	0,378	0,375	0,378	13479	1,778
Alg_{39}	PR_{10}	TB_{FFs}	**0,254**	0,271	0,276	0,267	3128	0,54
Alg_{40}	PR_{10}	TB_{RTCs}	0,26	**0,263**	**0,268**	**0,264**	9923	1,261
Alg_{41}	PR_{11}	TB_{NEHs}	0,444	0,407	0,409	0,42	**3001**	**0,477**
Alg_{42}	PR_{11}	TB_{Ds}	0,379	0,376	0,374	0,376	13130	1,71
Alg_{43}	PR_{11}	TB_{FFs}	0,277	**0,254**	0,263	0,265	3123	0,54
Alg_{44}	PR_{11}	TB_{RTCs}	**0,257**	0,275	**0,257**	**0,263**	9561	1,213
Alg_{45}	PR_{12}	TB_{NEHs}	0,442	0,419	0,409	0,424	**3000**	**0,474**
Alg_{46}	PR_{12}	TB_{Ds}	0,384	0,376	0,368	0,376	13218	1,744
Alg_{47}	PR_{12}	TB_{FFs}	**0,259**	0,282	0,267	0,27	3127	0,538
Alg_{48}	PR_{12}	TB_{RTCs}	0,263	**0,243**	**0,263**	**0,257**	9742	1,24

The best results in each column for the combination with the same priority rule are in bold and the worst in red.

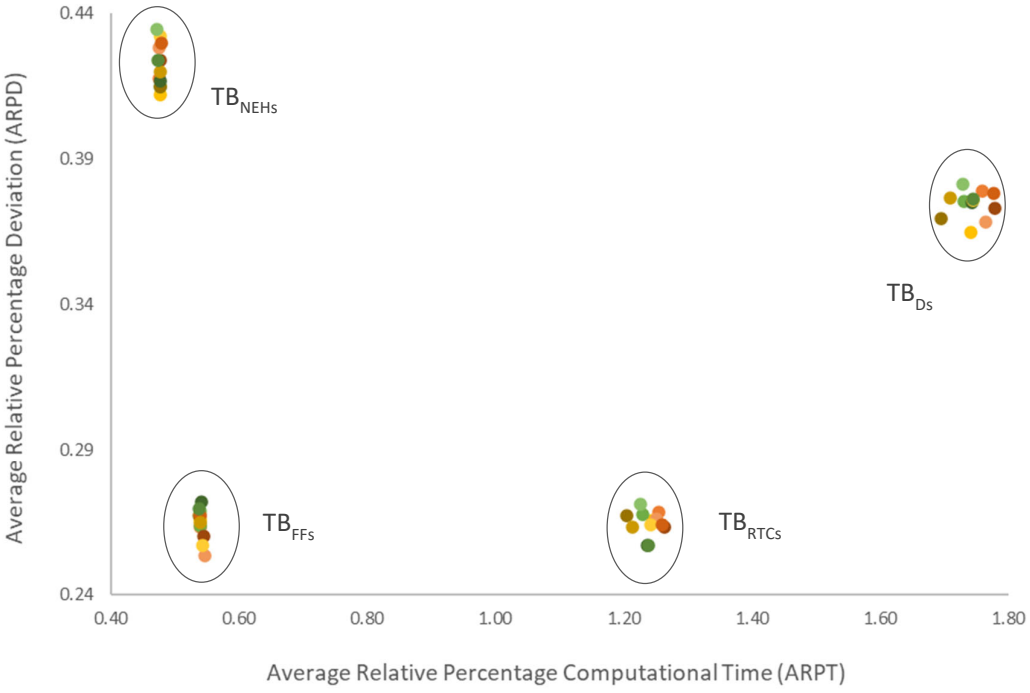

Fig. 1. ARPD and ARPT values in relation to the number of jobs for the 48 compared methods. Each combination of 12 priority rules with a tie-breaking strategy is circled with the name of the tie-breaking strategy.

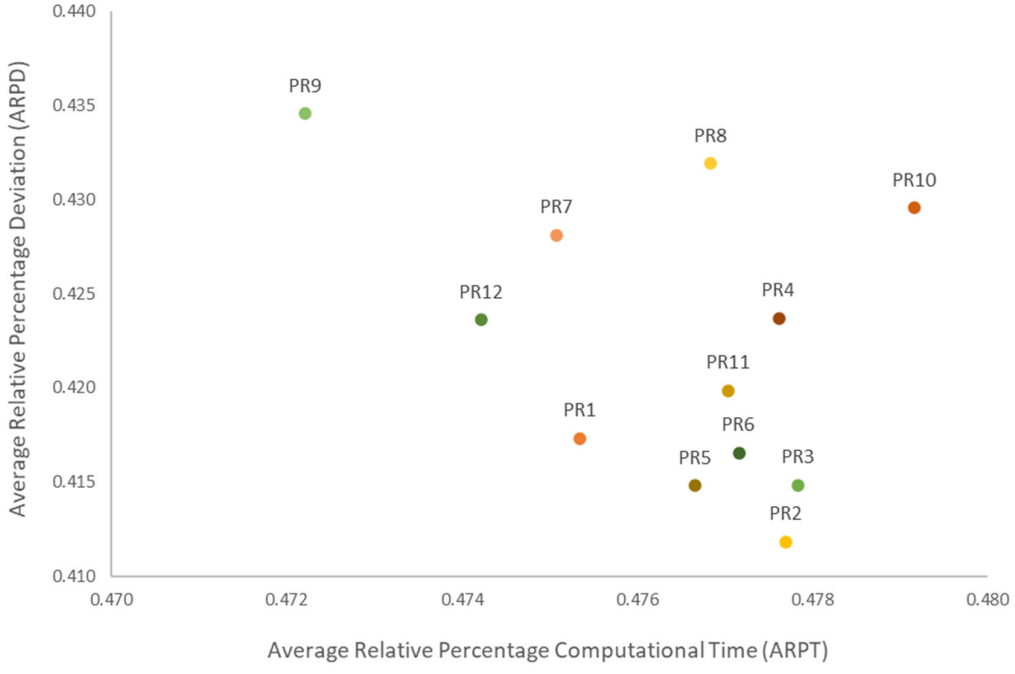

Fig. 2. ARPD and ARPT values for methods using the NEHs tie-breaking strategy combined with each of the 12 priority rules.

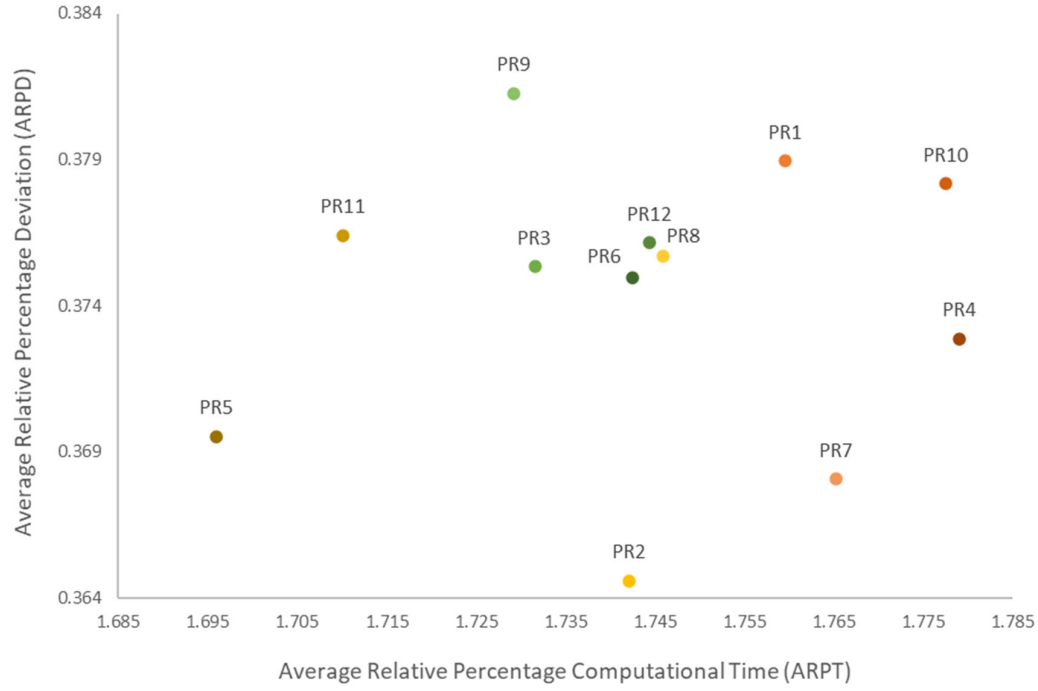

Fig. 3. ARPD and ARPT values for methods using the Ds tie-breaking strategy combined with each of the 12 priority rules.

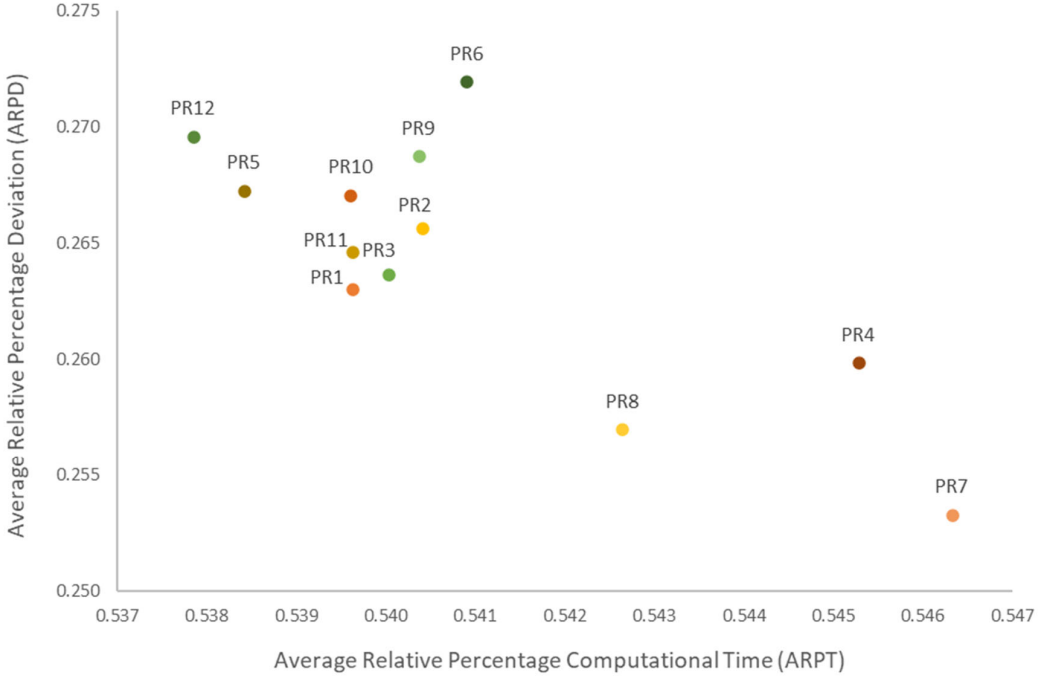

Fig. 4. ARPD and ARPT values for methods using the FFs tie-breaking strategy combined with each of the 12 priority rules.

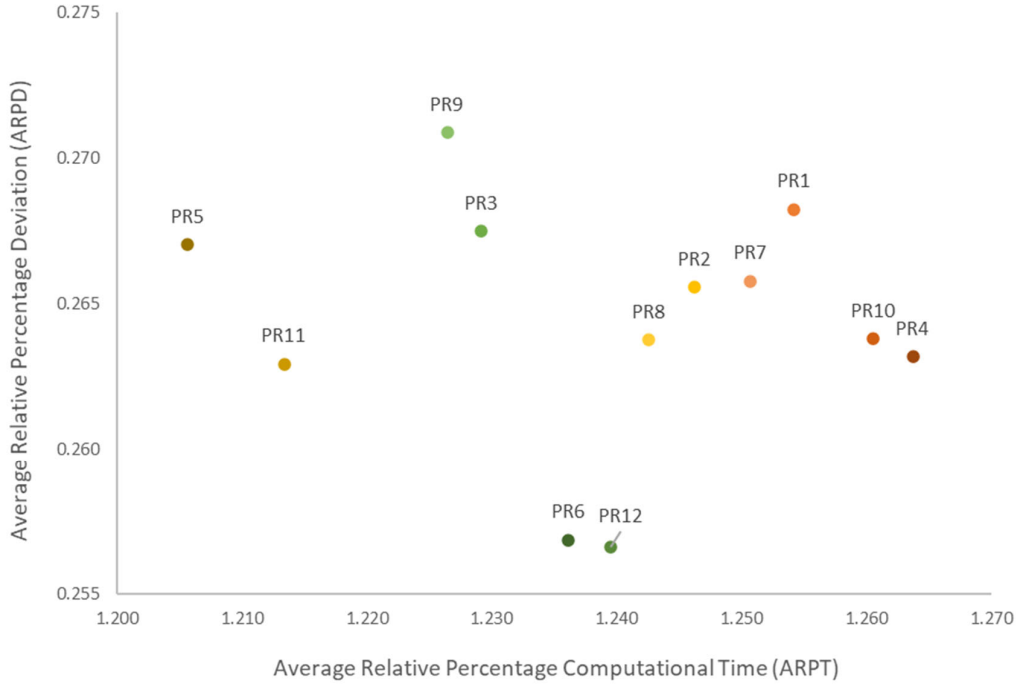

Fig. 5. ARPD and ARPT values for methods using the RTCs tie-breaking strategy combined with each of the 12 priority rules.

4. Conclusion

In this article, the PFSP-SIST was addressed with the objective of finding the sequence among the n! possibilities that minimize the makespan. The heuristic under analysis is the NEH, for which 48 heuristics were developed by combining 12 priority rules and 4 tie-breaking strategies for first and fourth stage, respectively.

The heuristics were evaluated using Taillard's acceleration adaptation to the PFSP-SIST problem, which allowed the execution of the tests with $O(n^2m)$ complexity. The results can be summarized in four main conclusions:

(1) Using tie-breaking strategies has a great impact on the final results;
(2) The impact of priority rules on final results varies according to the tie-breaking strategy used;
(3) All tie-breaking strategies showed better ARPD results when compared to the original strategy, NEH;
(4) The strategies denoted as FFs and RTCs showed the best ARPD results, but the FFs was notable for using significantly less CPU time.

Thus, the superiority of the heuristics that use the FFs tiebreaker strategy is evident and, despite the variation between the priority rules not generating significant differences, in terms of ARPD, its combination with PR_7 yields the best results. But in terms of ARPT, its combination with PR_5 and PR_{12} yield the best results.

In future research, it would be interesting to analyze different flow shop problems that are normally solved with algorithms that consider only the processing time to adapt them to the constraint of independent setup times. For instance, in the case of the blocking flow-shop scheduling problem (BFSP), reducing the work in process is vital for preserving capital investment, with buffers set to zero between machines, where Wu, Gao, Liu and Cheng (2023) proposed an improved NEH-based heuristic. Furthermore, another possibility to be explored is implementing the methods presented here in different databases, which present different variations in jobs and machines' quantity.

Acknowledgment

This research was partially supported by grants 312585/2021-7 and 404819/2023-0 from the National Council for Scientific and Technological Development (CNPq), Brazil.

References

Allahverdi, A., Ng, C. T., Cheng, T. E. & Kovalyov, M. Y. (2008). A Survey Of Scheduling Problems With Setup Times Or Costs. *European Journal Of Operational Research, 187*(3), 985-1032. https://doi.org/10.1016/j.ejor.2006.06.060.
Baker, K. R. (1974). *Introduction To Scheduling And Sequencing*. Nova York: Wiley.

Belabid, J., Aqil, S. & Karam, A. (2020). Solving Permutation Flow Shop Scheduling Problem with Sequence-Independent Setup Time. *Journal of Applied Mathematics, 2020*(1), Article ID 7132469. https://doi.org/10.1155/2020/7132469.

Dong, X., Huang, H. & Chen, P. (2008). An Improved NEH-Based Heuristic For The Permutation Flowshop Problem. *Computers & Operations Research, 35*(12), 3962-3968. https://doi.org/10.1016/j.cor.2007.05.005.

Fernandez-Viagas, V. & Framinan, J. M. (2014). On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers & Operations Research, 45*(1), 60-67. https://doi.org/10.1016/j.cor.2013.12.012.

Fernandez-Viagas, V., Ruiz, R. & Framinan, J. M. (2017). 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, 257*(3), 707-721. https://doi.org/10.1016/j.ejor.2016.09.055.

Graham, R. L., Lawler, E. L., Lenstra, J. K. & Kan, A. R. (1979). Optimization And Approximation In Deterministic Sequencing And Scheduling: A Survey. *Annals Of Discrete Mathematics, 5*(1), 287-326. https://doi.org/10.1016/S0167-5060(08)70356-X.

Johnson, S. M. (1954). Optimal Two- And Three-Stage Production Schedules With Setup Times Included. *Naval Research Logistics Quarterly, 1*(1), 61-68. https://doi.org/10.1002/nav.3800010110.

Kalczynski, P. J. & Kamburowski, J. (2008). An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers & Operations Research, 35*(9), 3001-3008. https://doi.org/10.1016/j.cor.2007.01.020.

Liu, W., Jin, Y. & Price, M. (2016). A new Nawaz–Enscore–Ham-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time. *Engineering Optimization, 48*(10), 1808-1822. https://doi.org/10.1080/0305215X.2016.1141202.

Liu, W., Jin, Y. & Price, M. (2017). A new improved NEH heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics, 193*(1), 21-30. https://doi.org/10.1016/j.ijpe.2017.06.026.

Maccarthy, B. L. & Liu, J. (1993). Addressing The Gap In Scheduling Research: A Review Of Optimization And Heuristic Methods In Production Scheduling. *International Journal Of Production Research, 31*(1), 59-79. https://doi.org/10.1080/00207549308956713.

Nawaz, M., Ensore Jr., E. & Ham, I. (1982). A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem. *Omega, 11*(1), 91-95. https://doi.org/10.1016/0305-0483(83)90088-9.

Reza Hejazi, S. & Saghafian, S. (2005). Flowshop-Scheduling Problems With Makespan Criterion: A Review. *International Journal Of Production Research, 43*(14), 2895-2929. https://doi.org/10.1080/00207540500564l7.

Ribas, I., Companys, R. & Tort-Martorell, X. (2010). Comparing three-step heuristics for the permutation flow shop problem. *Computers & Operations Research, 37*(12), 2062-2070. https://doi.org/10.1016/j.cor.2010.02.006.

Rossi, F. L. & Nagano, M. S. (2019). Heuristics for the mixed no-idle flowshop with sequence-dependent setup times. *Expert Systems with Applications, 125*(1), 40-54. https://doi.org/10.1016/j.eswa.2019.01.057.

Ruiz, R. & Allahverdi, A. (2007). Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. *Annals of Operations Research, 156*(1), 143-171. RePEc:spr:annopr:v:156:y:2007:i:1:p:143-171:10.1007/s10479-007-0227-8.

Ruiz, R. & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*(2), 479-494. https://doi.org/10.1016/j.ejor.2004.04.017.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research, 47*(1), 65-74. https://doi.org/10.1016/0377-2217(90)90090-X.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*(2), 278-285. https://doi.org/10.1016/0377-2217(93)90182-M.

Watson, J. P., Barbulescu, R., Whitley, L. D. & Howe, A. E. (2002). Contrasting structured and random permutation flowshop scheduling problems: search-space topology and algorithm performance. *INFORMS Journal on Computing, 14*(2), 98-123. https://doi.org/10.1287/ijoc.14.2.98.120.

Wu, Q., Gao, Q., Liu, W. & Cheng, S. (2023). Improved NEH-based heuristic for the blocking flow-shop problem with bicriteria of the makespan and machine utilization. *Engineering Optimization, 55*(3), 399-415. https://doi.org/10.1080/0305215X.2021.2010727.

Zhang, J., Dao, S. D., Zhang, W., Goh, M., Yu, G., Jin, Y. & Liu, W.. (2023). A new job priority rule for the NEH-based heuristic to minimize makespan in permutation flowshops. *Engineering Optimization*. https://doi.org/10.1080/0305215X.2022.2085259.