EasyChair Preprint

№ 440

A New Tiebreaker in the NEH heuristic for the Permutation Flow Shop Scheduling Problem

Alexander J. Benavides

September 14, 2018

A New Tiebreaker in the NEH heuristic for the Permutation Flow Shop Scheduling Problem

Alexander J. Benavides
Universidad Católica San Pablo, Arequipa, Perú.
Universidad Nacional de San Agustín de Arequipa, Perú.
ajbenavides@ucsp.edu.pe, ajbenavides@unsa.edu.pe

Abstract

The most efficient constructive heuristic so far for the Permutation Flow shop Scheduling Problem (PFSSP) with makespan minimization criterion is the NEH heuristic. It iteratively inserts a non-scheduled job into the position of the partial schedule that reduces the makespan. It iterates until a complete schedule is produced. It usually produces a high number of ties when selecting the best position, and the recent literature is proposing tiebreakers to improve the results. In this paper we propose a new tiebreaker that is based on the estimation of the variation of idle times produced with the insertion of the new job, and that takes into account the reversibility property of the PFSSP. Computational results show that this tiebreaker outperforms the state-of-the-art heuristics.

Keywords: Scheduling; Flow shop; Heuristics; NEH; Tiebreaker.

1 Introduction

The flow shop scheduling problem (FSSP) consists in finding a processing order for n jobs on m sequential machines such that certain criterion is optimized. When all the machines are constrained to the same processing order of jobs, the problem called the Permutation Flow Shop Scheduling Problem (PFSSP). The PFSSP with the makespan minimization criterion is NP-Complete is one of the most studied problems in the field of Operations Research (see [5, 6, 10, 13]). The constructive heuristic NEH, proposed by Nawaz, Enscore, and Ham [9], is considered the most efficient even nowadays. Many recent research publications are variants of this heuristic (e.g. [2, 3, 4, 7, 8, 11]).

In this paper, we propose a new tie-breaking mechanism for the NEH heuristic, and compare its results to state-of-the-art heuristics. The rest of the paper is organized as follows: In the next section we review the theoretical concepts of the PFSSP and we study the NEH heuristic and relevant variants from the literature. Section 3 describes our new tiebreaker. Section 4 shows computational results. Finally, Section 5 presents our concluding remarks and hints our next research steps.

The general Flow Shop Scheduling Problem does not impose the same processing order to all machines. Even when the makespan may be reduced when considering different processing orders on some machines [1, 12], this paper only considers the permutation version of this problem.

2 Problem Statement

This section starts reviewing the theoretical concepts of the problem. Section 2.2 explains the NEH heuristic; and Sections 2.3 and 2.4 explain variants of the NEH heuristic from the literature.

2.1 The Permutation Flow Shop Scheduling Problem (PFSSP)

The PFSSP can be stated as follows: There are n jobs that must be processed on m machines, and a processing order of the jobs must be found, such that the maximum completion time (or makespan, or $C_{\max}$) is minimized. Such order is called a schedule, because it allocates the jobs to the machines over time. The following conditions are assumed for the PFSSP: Each job can be processed on at most one machine at a time, and no machine can process more than one job simultaneously. All the jobs must follow the same machine sequence, and all the machines must process the jobs in the same order. Job pre-emption is not allowed. Release times of jobs are 0. Set-up times are either insignificant or considered within the processing time. In-process inventory is considered unlimited. All machines are available for the scheduling time window.

To give a mathematical definition, let us assume that each operation ij of job $j \in [n]$ on machine $i \in [m]$ has a given processing time t_{ij}. Let the variable x_{ij} be the starting time of operation ij. Also let the variable $y_{jj'}$ indicate the precedence order of jobs j and j'. Then, an integer linear program for the PFSSP is

$$\textbf{min.} \quad C_{\max}, \tag{1}$$

$$\textbf{s.t.} \quad x_{mj} + t_{mj} \leq C_{\max}, \qquad \forall j \in [n], \tag{2}$$

$$x_{ij} + t_{ij} \leq x_{(i+1)j}, \qquad \forall i \in [m-1], j \in [n], \tag{3}$$

$$x_{ij} + t_{ij} \leq x_{ij'} + M(1 - y_{jj'}), \qquad \forall i \in [m], j \neq j' \in [n], \tag{4}$$

$$y_{jj'} + y_{j'j} = 1, \qquad \forall j \neq j' \in [n], \tag{5}$$

$$x_{ij} \geq 0, \qquad \forall i \in [m], j \in [n], \tag{6}$$

$$y_{jj'} \in \{0, 1\}, \qquad \forall j \neq j' \in [n]. \tag{7}$$

Constraints (2) define the makespan. Constraints (3) require a job to finish its process on a machine before starting on the following one. Constraints (4) require two jobs j and j' to be processed in the order defined by the variable $y_{jj'}$. Finally, Constraints (5) enforce a linear ordering of the jobs.

The PFSSP has a reversibility property [11]. To explain this, let the bijection $\pi : [n] \to [n]$ represent a permutation of the jobs, i.e. a partial or a complete solution for the PFSSP. The reversibility property states that the makespan of a schedule $\pi = (\pi(1), \ldots, \pi(n))$ for the direct instance I (formed by the processing times t_{ij} for $j \in [n]$ jobs and $i \in [m]$ machines) is the same as the makespan of the reverse permutation $\pi' = (\pi(n), \ldots, \pi(1))$ for the reverse instance I' (formed by the processing times $t'_{ij} = t_{m-i+1,j}$ for $j \in [n]$ jobs and $i \in [m]$ machines). Thus, it is equivalent to approach the PFSSP of either the direct instance I or the reverse instance I'.

2.2 NEH Heuristic with Taillard's Acceleration

The NEH Heuristic was proposed by Nawaz, Enscore, and Ham [9], and it is still considered the best performing constructive heuristic for the PFSSP [4, 5, 8]. The NEH Heuristic has two main steps: First, determine an initial priority order π_0 of the jobs and set the first job as the initial partial schedule ($\pi = \{\pi_0(1)\}$). Second, insert the next job $\pi_0(k)$ into the partial schedule at the position that minimizes the makespan. The second step is repeated for $k \in [2, n]$ until all jobs are scheduled.

The variants of the NEH heuristic in the literature differ on the selected initial priority order within the first step and on the tie-breaking mechanisms during the insertion step. Those variants are explained in Sections 2.3 and 2.4 respectively.

Let $p_{ij} = t_{i,\pi(j)}$ represent the processing time of the scheduled job $\pi(j)$ on machine i. To evaluate a partial schedule $\pi = (\pi(1), \ldots, \pi(k))$, we must calculate the earliest completion time of

each job j on each machine i as

$$e_{ij} = \max\{e_{i,j-1}, e_{i-1,j}\} + p_{ij}, \qquad i \in [m], j \in [k], \tag{8}$$

with $e_{i,0} = e_{0,j} = 0$. The evaluation of a partial solution has a time complexity of $O(km)$, and we must evaluate k insertion positions to insert the k-th job. As $k \in O(n)$, the time complexity of the original NEH heuristic is $O(n^3m)$.

Taillard [15] proposed a method that reduces this time complexity to $O(n^2m)$ by calculating the makespan of all insertion positions for the next job in one sweep. Let us assume that job l shall be inserted into the partial schedule $\pi = (\pi(1), \ldots, \pi(k))$, thus there are $[k+1]$ insertion positions. Taillard's method first calculates the earliest completion times (or heads) for the partial solution before the insertion with Equation (8). Analogously, it also calculates the time difference q_{ij} between the makespan and the latest starting time (or tail) of each job j on each machine i as

$$q_{ij} = \max\{q_{i,j+1}, q_{i+1,j}\} + p_{ij}, \qquad i \in [m], j \in [k], \tag{9}$$

with $q_{m+1,j} = q_{i,k+1} = 0$. When job l is inserted after job $\pi(j-1)$, the head times $e_{i,j-1}$ will not change. Thus, they can be used to calculate the relative earliest completion time e'_{ij} of job l if it is inserted at position j on each machine i as

$$e'_{ij} = \max\{e'_{i-1,j}, e_{i,j-1}\} + t_{il}, \qquad i \in [m], j \in [k+1], \tag{10}$$

with $e'_{0j} = 0$. Finally, the makespan MC_j of the schedule produced by the insertion of job l into position $j \in [k+1]$ is

$$MC_j = \max_{i \in [m]}\{e'_{ij} + q_{ij}\}. \tag{11}$$

The calculation of all the values of equations (8) to (11) takes $O(km)$, and we must calculate those values to insert each of the $k \in [2, n]$ remaining jobs. As $k \in O(n)$, the NEH heuristic with Taillard's acceleration has a time complexity of $O(n^2m)$.

2.3 Priority Rules for the Initial Order

First, let us recall three statistical measures on the processing times of job j: the average processing time is $AVG_j = \frac{1}{m}\sum_{i=1}^{m} t_{ij}$, their standard deviation is $STD_j = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(t_{ij} - AVG_j)^2}$, and their skewness is $SKE_j = \left(\frac{1}{m}\sum_{i=1}^{m}(t_{ij} - AVG_j)^3\right) \Big/ \left(\sqrt{\frac{1}{m}\sum_{i=1}^{m}(t_{ij} - AVG_j)^2}\right)^3$.

The original NEH heuristic [9] ordered the jobs by their non-increasing total completion times. This is equivalent to order them by their non-increasing AVG_j. Dong, Huang, and Chen [2] ordered the jobs by their non-increasing $AVG_j + STD_j$. Liu, Jin, and Price [8] ordered the jobs by their non-increasing $AVG_j + STD_j + \mathrm{abs}(SKE_j)$, where $\mathrm{abs}(SKE_j)$ is the absolute value of the skewness. Those different priority orders presented improvements within their publications, as we show along our computational results.

2.4 Insertion Tiebreakers

When there are multiple insertion positions that produce the best possible makespan during the second step of NEH, the original NEH heuristic [9] just selects the first one.

Ribas, Companys, and Tort-Martorell [11] applied NEH variants to the direct instance and the corresponding reverse instance, retaining the best between the two produced schedules. Applying a NEH variant to the corresponding reverse instance only changes the selected insertion position in the case of ties. This approach reaches better results, but it also duplicates the computational time.

In the case of ties, Fernandez-Viagas and Framinan [4] minimize an approximation of the variation in the the idle time as a tie-breaking criterion. To explain their approximation, let us assume that job l shall be inserted into a partial schedule $\pi = (\pi(1), \ldots, \pi(k))$, and that the relative earliest completion time of job $\pi(j)$ after the insertion of job l in position $j \in [k+1]$ is

$$e''_{ij} = \max\{e''_{i-1,j}, e'_{ij}\} + p_{ij}, \qquad i \in [m], \tag{12}$$

with $e''_{0j} = 0$ and $e''_{i,k+1} = e'_{i,k+1}$, because there is no job in the partial schedule π at position $k+1$. Their approximation of the variation in the idle time produced by the insertion of job l in position $j \in [k+1]$ is

$$it(j) = \sum_{i=1}^{m} \Delta'_{i,j-1} + \Delta'_{i,j} - \Delta_{i,j-1}, \tag{13}$$

where the idle time before job l after it is inserted in position j is $\Delta'_{i,j-1} = \left(e'_{ij} - t_{il}\right) - e_{i,j-1}$, the idle time after job l after it is inserted in position j is $\Delta'_{i,j} = \left(e''_{ij} - p_{ij}\right) - e'_{ij}$, and the idle time between jobs in positions j and $j-1$ before job l is inserted is $\Delta_{i,j-1} = (e_{ij} - p_{ij}) - e_{i,j-1}$. Thus, their approximation is equivalent to

$$it(j) = \sum_{i=1}^{m} e''_{ij} - t_{il} - e_{ij}. \tag{14}$$

Considering that $\sum_{i=1}^{m} t_{il}$ is constant for any insertion position, it is also equivalent to

$$it'(j) = \sum_{i=1}^{m} e''_{ij} - e_{ij}. \tag{15}$$

Finally, Fernandez-Viagas and Framinan [4] use Equations (12) and (15) to calculate their approximation as

$$it''(j) = \sum_{i=1}^{m} e'_{ij} - e_{ij} + p_{ij} + \max\{e''_{i-1,j} - e'_{ij}, 0\}. \tag{16}$$

In the case of ties, Liu, Jin, and Price [8] minimize the sum of weighted job completion times and the variation over the gaps as a tie-breaking criterion. For the insertion position $j \in [k+1]$, their metric is

$$LJP_j = \sum_{i=1}^{m} w_i e'_{ij} + \alpha \sum_{i=1}^{m} \mathrm{abs}(g_{ij} - \bar{g}), \tag{17}$$

where w_i is the weight assigned to machine i, the parameter α is the weight for the variation of gaps, $g_{ij} = C_{\max} - e'_{ij} - q_{ij}$ is the gap after job l on machine i if it is inserted at position j, the average of the gaps is $\bar{g}_j = \frac{1}{m}\sum_{i=1}^{m} g_{ij}$, and q_{ij} and e'_{ij} are defined in Equations (9) and (10).

Liu, Jin, and Price [8] use Equation (17) to compare two tied insertion positions j and j' ($j < j'$), but they propose to compare the same number of jobs, thus they calculate the relative completion times for all the jobs after the insertion in position j until position j'. This duplicates their calculations without increasing the computational complexity. Besides, they determine the machine weights w_i according to a decreasing order of the gaps at each insertion position. Their matlab implementation performs a call to the function *sort* each time it compares two tied insertion positions. This do increases the computational complexity of the NEH heuristic to $O(n^2 m^2 \log(m))$ unwittingly.

3 Proposed Tiebreaker

The reversibility property of the PFSSP inspired us to design a new tiebreaker based on the tiebreaker proposed by Fernandez-Viagas and Framinan [4]. We propose to use the minimization of a weighted approximation of the iddle time variation of both direct and reverse instances as a tiebreaker criterion. An unweighted approximation of the iddle time variation of both direct and reverse instances using the Equation (14) is

$$DR'(j) = \sum_{i=1}^{m} e''_{ij} + q''_{ij} - 2t_{il} - e_{ij} - q_{ij}. \tag{18}$$

For this approximation, we must calculate two relative tail times produced by the insertion of job l at position $j \in [k+1]$. The relative tail time q'_{ij} of job l when inserted at position j for each machine i is

$$q'_{ij} = \max\{q'_{i+1,j}, q_{i,j}\} + t_{il}, \qquad i \in [m], \tag{19}$$

with $q'_{m+1,j} = 0$; and the relative tail time of job $\pi(j-1)$ after the insertion of job l in position j is

$$q''_{ij} = \max\{q''_{i+1,j}, q'_{i,j}\} + p_{i,j-1}, \qquad i \in [m], \tag{20}$$

with $q''_{m+1,j} = 0$ and $q''_{i0} = q'_{i0}$, because there is no job in the partial schedule π at position 0.

We change the term $2t_{il}$ in Equation (18) for $(p_{ij} + p_{i,j-1})$ considering that $\sum_{i=1}^{m} t_{il}$ is constant for any insertion position and that the idle times of the approximation might be influenced by the times of the adjacent jobs. Thus, the proposed weighted approximation for the insertion of job l into the partial schedule π in position $j \in [2,k]$ is

$$DR(j) = \sum_{i=1}^{m} w_i \left(\alpha(e''_{ij} + q''_{ij}) - \beta(e_{ij} + q_{i,j-1}) - \gamma(p_{ij} + p_{i,j-1})\right), \tag{21}$$

where w_i is a weight in the range $[1,m]$ assigned to machine i, and the three parameters α, β, and γ are used to distinguish the influence of each term. Section 4.3 shows the calibration of parameters α, β, and γ. The weight assigned to machine i is $w_i = \left\lfloor \frac{(m-1)(tm_i - tm_{\min})^2}{(tm_{\max} - tm_{\min})^2} \right\rfloor + 1$, where $tm_i = \sum_{j=1}^{n} t_{ij}$ is the total processing time on machine i, and $tm_{\min} = \min_{i\in[m]}\{tm_i\}$ and $tm_{\max} = \max_{i\in[m]}\{tm_i\}$ are the minimum and the maximum of those times among the machines.

Neither $DR(0)$ nor $DR(k+1)$ can be calculated because there is no job in positions 0 or $k+1$. Thus, we set $DR(0) = DR(k+1) = \infty$, discarding the first and the last insertion positions, to avoid the selections made by the original NEH heuristic for the direct and the reverse instances.

4 Computational Results

4.1 Experimental methodology

The constructive heuristics were implemented in C++17, compiled with the GNU C++ compiler version 7.2.0 with optimization level 2, and run on a PC with an AMD Opteron 6238 processor running at 2.9 GHz, and with 64 GB of main memory, using only one core in each execution.

We have tested the constructive heuristics on the 120 instances proposed by Taillard [16] and the 480 instances proposed by Vallada, Ruiz, and Framinan [18], divided in 240 VRF-small and 240 VRF-large. These are the standard benchmarks in the literature. As the evaluated constructive heuristics are deterministic, we perform each experiment once for each instance and NEH variant.

NEH variants differ in priority rules and tiebreakers. We refer to each priority rule by the last statistical measure used to calculate it: PR_{AVG} by Nawaz, Enscore, and Ham [9], PR_{STD} by

Dong, Huang, and Chen [2], and PR_{SKE} by Liu, Jin, and Price [8]. We refer to each tiebreaker by the initials of the authors that proposed them: T_{NEH} by Nawaz, Enscore, and Ham [9], T_{FF} by Fernandez-Viagas and Framinan [4], T_{LJP} by Liu, Jin, and Price [8], and T_B for our proposal.

We present the quality of the results as the relative deviation RD $= (C_{\max} - C_{\max}{}^*)/C_{\max}{}^*$ from the best known value $C_{\max}{}^*$, and as the average relative deviation (ARD) for groups of instances. The best known values are those reported by Taillard [17] and by Vallada, Ruiz, and Framinan [18]. We assess the performance of a heuristic h as the average relative time $\text{ART}_h = \left(\sum_{\forall i} \frac{CPU_{h,i} - ACT_i}{ACT_i} \Big/ |I|\right) + 1$, over all instances $i \in I$, where $CPU_{h,i}$ is the computation time of heuristic h performed on instance i, and $ACT_i = \sum_{\forall h} CPU_{h,i}/|H|$ is the average CPU time for instance i over all heuristics $h \in H$. We also report the average time $AT_h = \sum_{\forall i} CPU_{h,i}/|I|$ for each heuristic h.

4.2 Parameter setting for the tiebreaker

We performed the calibration experiments of the T_B tiebreaker with the best priority rule (PR_{SKE}) on the benchmark of Taillard. The term with the largest value in Equation (21) is $(e''_{ij} + q''_{ij})$. The other two terms are substracted from it. Thus, to calibrate the parameters α, β, and γ, we set $\alpha = 100$, and we test $\beta \in [0, 100]$ and $\gamma \in [0, 100]$ (10201 combined levels). Thirteen combinations reached a percentual ARD below 2.75, and three of them below 2.74. The lowest ARD of 2.724 is achieved with $\alpha = 100$, $\beta = 88$, and $\gamma = 25$; thus we set these values for the rest of the experiments.

4.3 Comparison of tiebreakers

Table 1 shows ARDs for twelve NEH variants (with different combinations of priority rules and tiebreakers) on the twelve size groups of Taillard's benchmark. The best ARD for each group is highlighted in gray. Among the priority rules, PR_{SKE} achieves the best results for seven size groups, PR_{STD} for four, and PR_{AVG} for one. The best average results are achieved using the PR_{SKE} priority rule, confirming the results of Liu, Jin, and Price [8]. Among the tiebreakers, both T_{LJP} and T_B achieve the best results for five groups each, and both T_{NEH} and T_{FF} for one group each. The combination PR_{SKE} and T_B produces the best results for four size groups, and its ARDs are less than 0.3% above the best results for other groups. The closest best combination is PR_{SKE} and T_{LJP}, whose ARDs are less than 0.38% above the best results for each group. Results on VRF benchmarks are similar: the combinations PR_{SKE} with T_{LJP} and PR_{SKE} with T_B closely produce the best average results, and no combination has the best results for a majority of size groups.

Table 2 shows ARDs for twelve NEH variants on each benchmark. In general, PR_{SKE} priority rule produces the best average results for each tiebreaker. The combination PR_{SKE} with T_B produces the best overall ARD of 3.021%, followed by the combination PR_{SKE} with T_{LJP} with an overall ARD of 3.025%. PR_{SKE} with T_B produces the best ARD on Taillard's benchmark, followed by PR_{SKE} with T_{LJP} with less than 0.027% of difference. PR_{SKE} with T_{LJP} produces the best ARD on VRF benchmarks, followed by PR_{SKE} with T_B with less than 0.008% of difference. The results of both combinations PR_{SKE} with T_B and PR_{SKE} with T_{LJP} are closely the best.

Table 2 also shows average times (AT) for each benchmark. Priority rules are applied in the first step before the iterative insertion phase, thus they show no strong effect on the runtime. There is a small variation in the runtime among T_{NEH}, T_{FF}, and T_B. This is because they all have the same time complexity of $O(n^2m)$. Otherwise, T_{LJP} doubles the time of other tiebreakers on VRF-large benchmark, and it triples the time on the other benchmarks.

Figure 1 plots for each benchmark the ARDs vs. the ARTs presented in Table 2. The PR_{SKE} priority rule produces the best average results for each tiebreaker. Among the tiebreakers, T_{LJP}

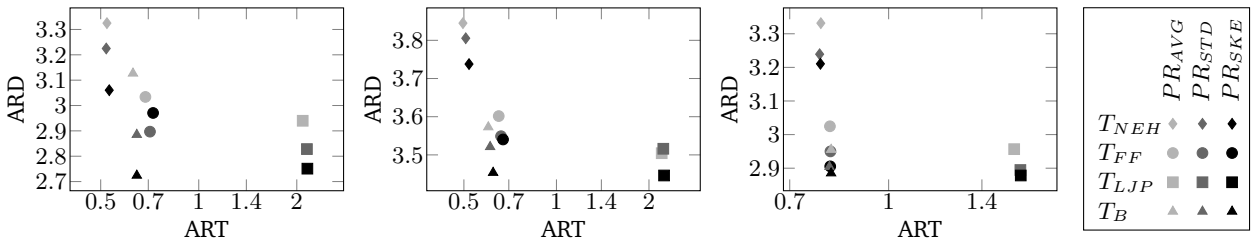
Table 1: Percentual ARDs for different priority rules and tiebreakers on Taillard's benchmark (by size).

		PR_{AVG}				PR_{STD}				PR_{SKE}			
		T_{NEH}	T_{FF}	T_{LJP}	T_B	T_{NEH}	T_{FF}	T_{LJP}	T_B	T_{NEH}	T_{FF}	T_{LJP}	T_B
20	5	3.300	2.293	2.365	2.978	2.703	2.559	2.193	2.401	2.708	2.359	2.164	2.382
20	10	4.601	4.152	4.726	4.866	4.084	3.543	3.794	3.854	3.684	3.563	3.679	3.550
20	20	3.731	3.305	3.337	3.318	3.816	3.331	3.531	3.151	2.914	3.156	3.061	2.931
50	5	0.727	0.922	0.562	0.801	0.893	0.749	0.634	0.952	0.879	0.848	0.641	0.746
50	10	5.073	5.150	4.688	5.442	4.904	4.905	4.641	4.763	4.844	5.174	4.246	4.094
50	20	6.648	6.207	6.111	5.961	6.121	5.812	5.780	6.230	6.419	6.485	6.154	5.982
100	5	0.527	0.378	0.360	0.450	0.411	0.412	0.345	0.393	0.538	0.464	0.364	0.344
100	10	2.215	2.182	1.620	2.011	2.156	1.719	1.455	1.516	2.241	1.889	1.721	1.748
100	20	5.345	5.021	5.085	5.129	5.653	5.147	4.998	4.990	4.988	5.100	4.813	4.757
200	10	1.258	0.984	0.928	0.998	1.270	0.987	0.988	1.007	1.243	1.022	0.894	0.950
200	20	4.408	4.037	3.785	3.858	4.569	3.885	3.861	3.755	4.145	3.810	3.649	3.592
500	20	2.066	1.776	1.711	1.708	2.121	1.713	1.718	1.607	2.123	1.777	1.624	1.612

Table 2: Percentual ARD, ART and AT (in microseconds) for different priority rules and tiebreakers on all benchmarks.

		PR_{AVG}				PR_{STD}				PR_{SKE}			
		T_{NEH}	T_{FF}	T_{LJP}	T_B	T_{NEH}	T_{FF}	T_{LJP}	T_B	T_{NEH}	T_{FF}	T_{LJP}	T_B
ARD	Taillard	3.325	3.034	2.940	3.127	3.225	2.897	2.828	2.885	3.060	2.971	2.751	2.724
	VRF-small	3.845	3.602	3.504	3.573	3.805	3.549	3.516	3.521	3.738	3.540	3.446	3.453
	VRF-large	3.332	3.025	2.957	2.955	3.239	2.950	2.894	2.904	3.210	2.905	2.878	2.885
AT	Taillard	2.54	2.93	8.42	2.89	2.52	2.96	8.30	2.87	2.54	3.00	8.35	2.87
	VRF-small	0.07	0.09	0.31	0.09	0.07	0.10	0.31	0.09	0.08	0.10	0.31	0.09
	VRF-large	43.91	45.30	84.87	45.17	43.89	45.29	86.25	45.13	43.92	45.34	86.24	45.23
ART	Taillard	0.523	0.686	2.085	0.628	0.520	0.708	2.148	0.645	0.531	0.724	2.155	0.645
	VRF-small	0.496	0.649	2.200	0.600	0.507	0.660	2.226	0.609	0.520	0.670	2.239	0.622
	VRF-large	0.783	0.809	1.573	0.813	0.780	0.811	1.608	0.808	0.782	0.810	1.611	0.813

is clearly the least efficient. The combination PR_{SKE} with T_B is faster than PR_{SKE} with T_{LJP}, producing similar results. The combination PR_{SKE} with T_B also reaches better results than any priority rule with either T_{NEH} or T_{FF}, using a comparable running time. The combination PR_{SKE} with T_B shows the best trade-off between efficiency and quality of results.



a. Taillard's benchmark. b. VRF-small benchmark. c. VRF-large benchmark. d. Legend.

Figure 1: Computational efficiency for each NEH variant on each benchmark.

5 Conclusions

This paper presented a new tiebreaker (T_B) for the NEH heuristic. It is based on the estimation of the variation of idle times produced with the insertion of a new job, taking into account the reversibility property of the PFSSP. The T_B tiebreaker, when combined with PR_{SKE} priority rule, shows the best trade-off between efficiency and quality of results, outperforming other heuristics.

Our next research includes two aspects: the proposal of better priority rules based on deconstructing known best schedules; and the evaluation of the proposed tiebreaker within state-of-the-art metaheuristics such as the Iterated Greedy Algorithm proposed by Ruiz and Stützle [14].

References

[1] Alexander J. Benavides and Marcus Ritt. Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops. *Computers & Operations Research*, 66:160–169, 2016.

[2] Xingye Dong, Houkuan Huang, and Ping Chen. An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research*, 35(12):3962–3968, 2008.

[3] Shahriar Farahmand Rad, Rubén Ruiz, and Naser Boroojerdian. New high performing heuristics for minimizing makespan in permutation flowshops. *Omega*, 37(2):331–345, 2009.

[4] Victor Fernandez-Viagas and Jose M Framinan. On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers & Operations Research*, 45:60–67, 2014.

[5] Victor Fernandez-Viagas, Rubén Ruiz, and Jose M Framinan. A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation. *European Journal of Operational Research*, 257(3):707–721, 2017.

[6] Jose M Framinan, Jatinder ND Gupta, and Rainer Leisten. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55(12):1243–1255, 2004.

[7] Pawel J Kalczynski and Jerzy Kamburowski. An empirical analysis of the optimality rate of flow shop heuristics. *European Journal of Operational Research*, 198(1):93–101, 2009.

[8] Weibo Liu, Yan Jin, and Mark Price. A new improved neh heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics*, 193:21–30, 2017.

[9] Muhammad Nawaz, E Emory Enscore, and Inyong Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

[10] S Reza Hejazi and S Saghafian. Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, 43(14):2895–2929, 2005.

[11] Imma Ribas, Ramon Companys, and Xavier Tort-Martorell. Comparing three-step heuristics for the permutation flow shop problem. *Computers & Operations Research*, 37(12):2062–2070, 2010.

[12] Daniel Alejandro Rossit, Fernando Tohmé, and Mariano Frutos. The non-permutation flow-shop scheduling problem: a literature review. *Omega*, 2017.

[13] Rubén Ruiz and Concepción Maroto. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494, 2005.

[14] Rubén Ruiz and Thomas Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

[15] Eric Taillard. Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74, 1990.

[16] Eric Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285, 1993.

[17] Eric Taillard, 2004. URL: `http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/flowshop.dir/best_lb_up.txt`. Best known lower and upper bounds of the PFSSP for Taillard's instances.

[18] Eva Vallada, Rubén Ruiz, and Jose M Framinan. New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3):666–677, 2015.