ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/273398783

NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness

Article *in* Computers & Operations Research · August 2015

DOI: 10.1016/j.cor.2015.02.002

CITATIONS
33

READS
422

2 authors:

Victor Fernandez-Viagas
Universidad de Sevilla
34 PUBLICATIONS **401** CITATIONS
SEE PROFILE

Jose M. Framinan
Universidad de Sevilla
188 PUBLICATIONS **3,125** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Collaboration strategies in decentralized supply chains with partial information sharing View project

Models and algorithms for the order scheduling problems considering setup times View project

All content following this page was uploaded by Victor Fernandez-Viagas on 06 November 2017.

The user has requested enhancement of the downloaded file.

NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness*

Victor Fernandez-Viagas[1]†, Jose M. Framinan[1]

[1] Industrial Management, School of Engineering, University of Seville,

Ave. Descubrimientos s/n, E41092 Seville, Spain, {vfernandezviagas,framinan}@us.es

November 6, 2017

Abstract

Since Johnson's seminal paper in 1954, scheduling jobs in a permutation flowshop has been receiving the attention of hundreds of practitioners and researchers, being one of the most studied topics in the Operations Research literature. Among the different objectives that can be considered, minimising the total tardiness (i.e. the sum of the surplus of the completion time of each job over its due date) is regarded as a key objective for manufacturing companies, as it entails the fulfilment of the due dates committed to customers. Since this problem is known to be NP-hard, most research has focused on proposing approximate procedures to solve it in reasonable computation times. Particularly, several constructive heuristics have been proposed, with NEHedd being the most efficient one, serving also to provide an initial solution for more elaborate approximate procedures. In this paper, we first analyse in detail the decision problem depending on the generation of the due dates of the jobs, and discuss the similarities with different related decision problems. In addition, for the most characteristic tardiness scenario, the analysis shows that a huge number of ties appear during the construction of the solutions done by the NEHedd heuristic, and that wisely breaking the ties greatly influences the quality of the final solution. Since no tie-breaking mechanism has been designed for this heuristic up to now, we propose several mechanisms that are exhaustively tested. The results show that some of them outperform the original NEHedd by about 25% while keeping the same computational requirements.

Keywords: Scheduling, Flowshop, Heuristics, NEH, tie-breaking mechanism,tardiness, PFSP.

*Preprint submitted to Computers and Operations Research, http://dx.doi.org/10.1016/j.cor.2015.02.002.

†Corresponding author

1 Introduction

The Permutation Flowshop Scheduling Problem (denoted as PFSP) is one of the most studied problems in the Operations Research literature (see the reviews by Framinan et al., 2004; Ruiz and Maroto, 2005). This decision problem deals with scheduling n jobs that have to be processed on m machines in the same order and with the same job sequence on every machine. Different criteria have been considered in the literature for this decision problem (see e.g. the reviews by Framinan et al., 2005; Vallada et al., 2008; Pan and Ruiz, 2013), such as the maximum completion time among the jobs or makespan, the total flowtime (sum of completion times of all jobs), and the total tardiness (sum of the tardiness of each job). Makespan and total completion time are related to the fast processing of the products and to a balanced use of resources, both issues being of great importance in make-to-stock manufacturing scenarios. In contrast, total tardiness focuses on the satisfaction of customers and it is therefore better suited for make-to-order manufacturing scenarios as due dates play a key role (Panwalkar et al., 1982; Kim et al., 2001). Thereby, among the objectives considered for the PFSP, the total tardiness highlights a critical concern for manufacturing systems (see e.g. Raman, 1995; Panwalkar et al., 1973), since delays may lead to an increase in costs such as penalty clauses in a contract, loss of customers and/or bad reputation for other customers (Sen and Gupta, 1984). The PFSP with total tardiness minimisation objective is denoted as $Fm|prmu|\sum T_j$ (see e.g. Pinedo, 1995).

Since the problem is known to be NP-hard, most researchers have focused on developing solution procedures (i.e. heuristics) that do not guarantee the optimality of the solution, but that can provide a (hopefully) good solution in a reasonable time interval. More specifically, several heuristics and metaheuristics have been proposed in the literature for the $Fm|prmu|\sum T_j$ problem, such as those by e.g. Gelders and Sambandam (1978); Kim et al. (1996); Rajendran and Ziegler (2003); Framinan and Leisten (2008); Vallada and Ruiz (2010). Among them, the NEHedd heuristic (Kim, 1993) stands out since, as we will discuss later, many works employ it to obtain an initial solution. The NEHedd is an adaptation for the tardiness objective of the well-known NEH heuristic by Nawaz et al. (1983) for makespan minimisation. In the NEH heuristic, jobs are initially arranged in non ascending order of their processing times. Then, a

job sequence is constructed by evaluating the partial schedules originating from the initial order: Assuming a sequence already set for the first $k-1$ jobs, k candidate (sub)sequences are obtained by inserting job k in the k possible slots of the current sequence. Out of these k (sub)sequences, the one yielding the minimum makespan is kept as the relative (sub)sequence for these first k jobs. Then, job $k+1$ from the initial order is considered analogously, and so on until all n jobs have been sequenced. In order to reduce the computational burden of the NEH heuristic, Taillard (1990) proposed a mechanism (known as Taillard's acceleration) to reduce the complexity of the NEH heuristic from $n^3 \cdot m$ to $n^2 \cdot m$. The excellent performance of the NEH heuristic and its easy adaptation to similar problems have led to its application to other scheduling decisions, such as the PFSP with total completion time minimisation (see e.g. Framinan et al., 2003), denoted as $Fm|prmu|\sum C_j$, or the hybrid flowshop scheduling problem (see e.g. Brah and Loo, 1999). For these problems, Taillard's acceleration cannot be applied, but Li et al. (2009) present a mechanism that saves between 30-50% of CPU time for the $Fm|prmu|\sum C_j$ problem, however without reducing its complexity.

The NEHedd heuristic differs from the NEH heuristic in the starting order (jobs are arranged now according to the Earliest Due Date or EDD rule), and in the evaluation of the partial sequences (as the one with lowest total tardiness is selected). Taillard's acceleration cannot be applied to the NEHedd, and, although Vallada and Ruiz (2010) propose a mechanism similar to that by Li et al. (2009), the complexity of the NEHedd remains $O(n^3 \cdot m)$.

The extensive computational evaluation of heuristics for the $Fm|prmu|\sum T_j$ problem carried out by Vallada et al. (2008) shows that NEHedd is a key constructive heuristic for the problem since, aside to being very efficient, the rest of efficient heuristics in the literature with more average CPU time employ NEHedd as an initial solution. More specifically, more than half of the state-of-the-art improvement heuristics or metaheuristics for the problem use NEHedd as a starting solution. This fact can be also seen in more recent works, such as Vallada and Ruiz (2010), or Schaller (2012).

Despite the excellent performance of the NEHedd heuristic, we believe that additional improvements could be gained by further analysis of the problem under consideration. First, the tardiness minimisation problem could resemble different scheduling problems depending on the

due dates of the jobs for each specific instance: Intuitively, it is clear that, for an instance with due dates much greater than the sum of the processing times of its jobs, almost every schedule may yield zero total tardiness, thus turning the problem into a trivial one. Analogously, unachievable due dates for each job results in an instance for which almost every sequence yields tardiness for every job and therefore the problem resembles that of minimising flowtime. By conducting an analysis of these possible scenarios, further insights on the problem can be obtained, so the performance of the NEHedd procedure can be enhanced. More specifically, we will show in this paper that such an analysis reveals the importance of adequately addressing the high number of ties appearing in the constructive phase of the NEHedd. In order to handle these ties in an efficient way, we propose several tie-breaking mechanisms for the problem and conduct an extensive computational experiment to test their performance. The results show that one of these mechanisms (based on machine idle time) improves the original results obtained by NEHedd by roughly 25% while requiring the same CPU time. Another one (based on Taillard's acceleration for makespan) outperforms the NEHedd by 15% while employing less CPU time. Furthermore, when using the idle time- based version of the NEHedd as starting solution for the metaheuristic by Vallada and Ruiz (2010) (which is the state-of-the-art metaheuristic for the problem), the metaheuristic improves its result for different stopping criteria.

The remainder of the paper is organized as follows: in Section 2 the problem under consideration is described and analysed to derive some properties of the problem that may serve to identify the different scenarios and related decision problems. As a result, it is shown that there is a high number of ties for the instances in the most tardiness-specific scenario (i.e. that one not leading to trivial problems or to flowtime minimisation). Therefore, in Section 3, eight tie-breaking mechanisms are proposed. An extensive comparison among them and with the original NEHedd procedure are performed in Section 4. Finally, conclusions and future research lines are discussed in Section 5.

2 Analysis of the Problem

The problem under study can be stated as follows: n jobs have to be scheduled in a flowshop composed of m machines. Each job j has a processing time, t_{ij}, on each machine i. Considering a sequence $\pi := (\pi_1, \ldots \pi_n)$, the processing time of the job in position j, i.e. job π_j, is denoted as p_{ij}, where $p_{ij} = t_{i\pi_j}$. C_{ij} denotes the completion time of job π_j on machine i. The completion time of the last job of the sequence in the last machine, $C_{m,n} = C_{max}$, is defined as the maximum completion time or makespan of the sequence. Let d_j be the due date of job j, and $p_j = \sum_{i=1}^{m} p_{ij}$ the sum of the processing times of job j across all machines. The tardiness (earliness) of job j is defined as $T_j = max\{C_{mj} - d_j, 0\}$ ($E_j = max\{d_j - C_{mj}, 0\}$). Finally, total tardiness is defined as $\sum T_j = \sum_{\forall j} max\{C_{mj} - d_j, 0\}$.

As discussed in Section 1, our problem is highly influenced by the due dates of the jobs in the specific instance. In this Section, we make an effort to gain a better understanding of the problem so the performance of existing solution procedures (most notably the NEHedd) can be enhanced. To do so, we first state two simple properties of the problem under consideration:

Property 2.1. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|\sum T_j$ problem, and WM be the maximum (worst) makespan that can be obtained for $\mathcal{I}$. If $d_j \geq WM, \forall j$, then each feasible sequence π is an optimal solution for $\mathcal{I}$. That is, $\mathcal{I}$ has $n!$ optimal solutions.*

Proof. The proof of this property is obvious: since WM is the worst makespan of the problem (i.e. $WM \geq C_{m,j}, \forall j$) and each due date is greater than or equal to WM (i.e. $d_j \geq WM \geq C_{m,j}$, $\forall j$), then minimising $\sum_{\forall j} max\{C_{m,j} - d_j, 0\}$ is equal than minimising $\sum_{\forall j} max\{-P, 0\} = 0$, where P is a non-negative number, and hence each feasible solution is an optimal solution of the problem. □

Property 2.2. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|\sum T_j$ problem with $d_j \leq \sum_{i=1}^{m} p_{ij}$, $\forall j$. Then, an optimal solution for $\mathcal{I}$ can be obtained by solving the corresponding $Fm|prmu|\sum C_j$ problem for $\mathcal{I}$.*

Proof. p_j is a lower bound of the makespan of the job j. When $d_j \leq p_j$, then each completion time C_{mj} is greater than or equal to its due date, d_j, and, hence $\sum_{\forall j} max\{C_{m,j} - d_j, 0\} = \sum_{\forall j}(C_{m,j} - d_j) = \sum_{\forall j} C_{m,j} - \sum_{\forall j} d_j = \sum_{\forall j} C_{m,j} + const.$ □

These two properties formalise the interdependence between the due dates and processing times of an instance, and the type of optimisation problem. If the due dates are extremely tight, the problem is similar to that of flowtime minimisation according to Property 2.2 whereas extremely loose due dates lead to a trivial problem according to Property 2.1. Therefore, a problem instance can be classified along these two extreme cases. To do so, we first define for each job j the following indicator v_j:

$$v_j = \frac{d_j - p_j}{WM - p_j} \tag{1}$$

Clearly, $v_j \leq 0$ indicates that the due date cannot be met for job j, regardless of the position where it is scheduled. Similarly, $v_j \geq 1$ corresponds to the case where the completion time of job j is lower than its due date. By adequately truncating v_j, we can obtain a normalised indicator for job j, i.e.: $\min\{1; \max\{0; v_j\}\} \in [0, 1]$.

Then, the indicator v can be defined as:

$$v = \sum_{j=1}^{n} \frac{\min\{1; \max\{0; v_j\}\}}{n} = \sum_{j=1}^{n} \frac{\min\{WM - p_j; \max\{0; d_j - p_j\}\}}{n \cdot (WM - p_j)} \tag{2}$$

It can be shown that $v \in [0, 1]$, and that if, for a given instance, $v = 0$ (tight due dates), then minimising the total tardiness is equivalent to minimising the total flowtime. On the other extreme, if $v = 1$ (loose due dates), then any sequence is optimal.

In addition to how tight/loose the due dates are, the variability of the due dates among jobs also plays an important role in the optimization problem, which is formalised using the following property:

Property 2.3. *The sequence* $\pi^{edd} := \left(\pi_1^{edd}, \cdots, \pi_n^{edd}\right)$ *obtained by the EDD rule, is an optimal solution of the* $Fm|prmu|\sum_j T_j$ *problem if* $d_{\pi_j^{edd}} \geq d_{\pi_{j-1}^{edd}} + \sum_{i=1}^{m} p_{i\pi_j^{edd}}$ *(or, equivalently,* $d_{\pi_j^{edd}} \geq \sum_{k=1}^{j} \sum_{i=1}^{m} p_{i\pi_k^{edd}}$*),* $\forall j > 1$*, and* $d_{\pi_1^{edd}} \geq \sum_{i=1}^{m} p_{i\pi_1^{edd}}$*.*

Proof. Taking into account that $C_{m,\pi_{j-1}^{edd}} + \sum_{i=1}^{m} p_{i\pi_j^{edd}}$ is an upper bound of $C_{m,\pi_j^{edd}}$, i.e. $C_{m,\pi_{j-1}^{edd}} + \sum_{i=1}^{m} p_{i\pi_j^{edd}} \geq C_{m,\pi_j^{edd}}$, the property can be easily proved recursively, as follows: Beginning with the first job of the sequence, π_1^{edd}, and assuming that $d_{\pi_1^{edd}} \geq \sum_{i=1}^{m} p_{i\pi_1^{edd}}$, then $C_{m,\pi_1^{edd}} - d_{\pi_1^{edd}} \leq$

$C_{m,\pi_1^{edd}} - \sum_{i=1}^{m} p_{i\pi_1^{edd}} = \sum_{i=1}^{m} p_{i\pi_1^{edd}} - \sum_{i=1}^{m} p_{i\pi_1^{edd}} = 0$, where it has been used that the completion time of the first job is equal to the sum of processing times, i.e. $C_{m,\pi_1^{edd}} = \sum_{i=1}^{m} p_{i\pi_1^{edd}}$. Hence, the first term of the objective function is zero, i.e. $C_{m,\pi_1^{edd}} - d_{\pi_1^{edd}} \leq 0 \longrightarrow max(C_{m,\pi_1^{edd}} - d_{\pi_1^{edd}}, 0) = 0$.

Following with the job in second position and assuming that $d_{\pi_2^{edd}} \geq d_{\pi_1^{edd}} + \sum_{i=1}^{m} p_{i\pi_2^{edd}}$, where $C_{m,\pi_1^{edd}} \leq d_{\pi_1^{edd}}$ by means of the job in the first position. Then $d_{\pi_2^{edd}} \geq C_{m,\pi_1^{edd}} + \sum_{i=1}^{m} p_{i\pi_2^{edd}}$. Note that $C_{m,\pi_1^{edd}} + \sum_{i=1}^{m} p_{i\pi_2^{edd}}$ is an upper bound of $C_{m,\pi_2^{edd}}$ and, hence $d_{\pi_2^{edd}} \geq C_{m,\pi_1^{edd}} + \sum_{i=1}^{m} p_{i\pi_2^{edd}} \geq C_{m,\pi_2^{edd}}$ which implies that the completion time of the job in second position is again lower than its due date and that the second term of the objective function is again zero, i.e. $C_{m,\pi_2^{edd}} - d_{\pi_2^{edd}} \leq 0 \longrightarrow max(C_{m,\pi_2^{edd}} - d_{\pi_2^{edd}}, 0) = 0$.

For the job in a position j, we assume $d_{\pi_j^{edd}} \geq d_{\pi_{j-1}^{edd}} + \sum_{i=1}^{m} p_{i\pi_j^{edd}}$. As $C_{m,\pi_{j-1}^{edd}} \leq d_{\pi_{j-1}^{edd}}$ from the previous job and $C_{m,\pi_{j-1}^{edd}} + \sum_{i=1}^{m} p_{i\pi_j^{edd}} \leq C_{m,\pi_j^{edd}}$, then the completion time of the job in position j is lower than its due date as well as the jth term of the objective function is zero, i.e. $C_{m,\pi_j^{edd}} - d_{\pi_j^{edd}} \leq 0 \longrightarrow max(C_{m,\pi_j^{edd}} - d_{\pi_j^{edd}}, 0) = 0$.

Taking into account the last expression, the minimisation of total tardiness can be written as $max \sum max\{C_{m,j} - d_j, 0\} = max(0)$ and, hence, the EDD rule is optimal. □

Property 2.3 suggests that, for instances with high values of indicator v (i.e. loose due dates) and a high variability in the due dates of the jobs, the EDD rule may have a good performance, as the due dates would have a greater influence in the objective function than the completion times of the jobs. Clearly, for such instances, employing more sophisticated algorithms might not pay off.

The three simple properties stated above determine three extreme cases of the total tardiness problem where good/optimal solutions can be found by any algorithm (i.e. $v \approx 1$), or by algorithms designed for different problems (i.e. by algorithms for flowtime minimisation if $v \approx 0$, or for tardiness minimisation in a single-machine if v is high and there is a high variability in the due dates). Obviously, the interest lies in finding efficient algorithms for instances in between these extreme cases. Therefore it is useful to review the different sets of instances that have been generated in the literature to check whether they adequately cover the specific tardiness minimisation case, or not.

To the best of our knowledge, testbeds for the $Fm|prmu|\sum T_j$ problem have been built

employing three different methods to generate due dates:

- Gelders and Sambandam (1978) generate the due dates according to a uniform distribution drawn between the sum of processing time of the job and this sum plus an upper bound. This method for generating due dates is labelled in the following as GS.

- Potts and Van Wassenhove (1982) generate the due dates using two parameters, T and R, related to the mean and variance of the due dates, respectively, according to an uniform distribution between $P \cdot (1 - T - R/2)$ and $P \cdot (1 - T + R/2)$, where P is a lower bound for the makespan. This method is labelled in the following as PV.

- In Hasija and Rajendran (2004), due dates are generated according to $(1+3 \cdot U[0,1]) \sum p_{ij}$. This method is denoted as HR in the following.

Clearly, these methods produce instances with different values of the indicator v and, in the case of the PV method, parameter R controls the variability of the due dates among jobs. To analyse the range of instances generated by each method, three different benchmarks have been built in the following manner: we consider the data regarding number of jobs, machines, and processing times as in the testbed by Vallada et al. (2008) (i.e. $n = \{50, 150, 250, 350\}$, $m = \{10, 30, 50\}$ and processing times drawn from a uniform $[1, 99]$ distribution), and generate three testbeds:

- The first testbed is that by Vallada et al. (2008), which was generated using the PV procedure with parameters $T = \{0.2, 0.4, 0.6\}$ and $R = \{0.2, 0.6, 1.0\}$, and produced 5 replicates for each combination of m, n, T, and R. In total, 540 instances were obtained.

- The second testbed is generated using the GS procedure. To have the same number of instances than in the previous testbed, 45 replicates are generated for each combination of m and n.

- The third testbed is generated in an analogous manner to the previous one (with 45 replicates for each combination of m and n), but using the HR procedure for due date generation.

Figure 1: Distribution of the percentage of instances depending on v for different generation of due dates (In the left, the percentage of instances of the testbeds in each interval of v is shown, while the right figure shows the cumulative percentage of instances).

For each instance in the three benchmarks, the indicator v has been calculated according to expression (2), where the worst makespan, WM has been approximated using a modified version of the NEH to maximise makespan. The amount of instances for different intervals of v is shown in Figure 1 for the three benchmarks. In the figure in the left side, the percentage of instances is classified according to the parameter v whereas the figure in the right shows the cumulative percentage of instances. As can be seen, HR and specially GS produce many instances with very low values of v for which the problem is similar to minimising the total flowtime. For HR, 65% instances have a v lower than 0.15 while with GS all the instances have v lower than 0.20. Hence, in this paper, we focus in the generation of due dates according to the PV method, which is more likely to generate instances in the range of interest of the $Fm|prmu|\sum_j T_j$ problem.

To further analyse the similarities between the $Fm|prmu|\sum T_j$ and to minimise total flowtime for low values of the parameter v, we solve all instances in the testbed with the PV due date generation method using the NEHedd heuristic and the NEH heuristic for flowtime minimisation (denoted as NEH_FT). In addition, we obtain the solution given for each instance by the EDD rule in order to test the influence of higher values of v and R. Note that there are only two differences between NEHedd and NEH_FT:

1. The starting order of NEHedd is the EDD rule whereas in NEH_FT the starting order is the ascending order of the sum of the processing times, and

2. When iteratively constructing the solution, NEHedd selects the best partial sequence with

Figure 2: $CRDI_{edd}$ and $CRDI_{NEH\_FT}$ for different values of v in each instance of the Benchmark of Vallada et al. (2008).

lowest total tardiness, while NEH_FT selects the one with lowest flowtime.

The usual indicator of the quality of heuristic i with respect to tardiness when applied to a given instance j is the so-called Relative Deviation Index (RDI), which is defined as follows:

$$RDI_{ij} = \frac{sumT_{ij} - Best_j}{Worst_j - Best_j} \cdot 100$$

where $sumT_{ij}$ is the total tardiness obtained by heuristic i when applied to instance j. $Worst_j$ and $Best_j$ are the worst and best known total tardiness for instance j. RDI is usually employed for tardiness instead of the average relative deviation (most used indicator for makespan or flowtime objectives) since tardiness may yield 0 for some instances and therefore the value of the average relative deviation would be distorted (see Vallada et al., 2008; Kim, 1993; Kim et al., 1996).

To better compare the performance obtained by the different heuristics that are to be tested in this paper and those by the NEHedd (which is the reference heuristic for the problem under consideration), we build the Compared Relative Deviation Index ($CRDI$), which is simply the difference between the RDI of the heuristic i and that of the NEHedd when both heuristics are applied to instance j, i.e.:

$$RDI_{ij} - RDI_{NEHedd,j} = CRDI_{ij} = \frac{sumT_{ij} - sumT_{NEHedd,j}}{Worst_j - Best_j} \cdot 100 \quad (3)$$

Figure 3: $CRDI_{edd}$ and $CRDI_{NEH\_FT}$ in each instance of the Benchmark of Vallada et al. (2008) for different values of parameters v and R.

Clearly, $CRDI \in [-100, 100]$. In the subsequent experiments, $Worst$ and $Best$ are taken from the best and worst known total tardiness for the instances recorded in http://soa.iti.es/problem-instances. The values of $CRDI_{edd}$ and $CRDI_{NEH\_FT}$ are shown in Figure 2 with respect to indicator v for each instance of the benchmark, while Figure 3 groups the results for different values of R. The following conclusions can be obtained according to those results:

- As predicted by Property 2.2, the performance of NEH_FT and NEHedd procedure is very similar for low values of v. $CRDI_{NEH\_FT}$ is on average 0.79 for instances with $v < 0.1$ and 2.91 for instances with $v < 0.15$.

- NEH_FT outperforms NEHedd when the variance of the due dates is low, i.e. $R = 0.2$, even for high values of v. The average $CRDI_{NEH\_FT}$ for $R = 0.2$ is -2.71. This fact can be explained if we analyse the objective function when the variance of the due dates is zero (common due dates). Then, minimising $\sum_j max\{C_{m,j} - d_j, 0\} = \sum_{j \in late}(C_{m,j} - d_j) = \sum_{j \in late} C_{m,j} - \sum_{j \in late} d_j = \sum_{j \in late} C_{m,j} - L \cdot const$, where L is the number of jobs late. The first term is directly included in the minimisation of total flowtime, while the second term decreases when minimising total flowtime.

- In general, the performance of NEH_FT deteriorates as v increases until it reaches medium-high values (this is particularly clear for the combination of parameters $R = 0.6$ and $R = 1.0$), i.e. NEH_FT procedure only performs better when the problem can be reduced to either a flowtime minimisation problem (low v) or to a trivial one (high v).

- The performance of the EDD rule improves as v increases.

- For high values of v and a high variance of the due dates ($R = 1.0$), the EDD rule performs roughly as good as the NEHedd procedure, i.e. $CRDI_{edd} \simeq 0$. This could be predicted as a consequence of Property 2.3, since if the variance of the due dates of an instance is high enough to verify the conditions of Property 2.3, then EDD is optimal.

According to the previous analysis and conclusions, the problem of minimising tardiness on an instance is bounded by three different problems depending on v and on the variance of the due dates of the jobs, as shown in Figure 4. Roughly speaking, high values of the mean and variance of

Figure 4: Location of the problem based on the mean and variance of the due dates.

the due dates correspond to a problem where the EDD rule is optimal (see Region 3 of Figure 4). Low values of the mean and variance determine a problem similar to $Fm|prmu|\sum C_j$ (see Region 1 of Figure 4). Finally, high values of the mean of the due dates combined with a low variance correspond to a trivial problem where each sequence is optimal (Region 2). The interesting region to be analysed for the $Fm|prmu|\sum T_j$ problem is the region between 1, 2 and 3, since otherwise we would be solving a different decision problem.

The analysis carried out also serves to explain the excellent performance of the NEHedd procedure and to identify possible improvements. The NEHedd heuristic performs well in the three regions since it minimises also flowtime in Region 1, and, since it includes the EDD rule as a sorting order, it guarantees good performance in Region 3. However, it can be seen that its performance decreases for medium/high values v as compared to that for low/medium values of v. An explanation of this rather surprising fact could lie in the high number of ties that would have to broken when, for each iteration of the NEHedd procedure, several partial sequences may have the same total tardiness. This situation could be rather common, as in the first iterations of the algorithm the total tardiness of the partial sequence is zero, thus leading to a high number of ties. In addition, since these ties appear in the first iterations, the mechanism chosen to solve

Figure 5: Average number of ties in each instance grouped by the parameter v.

them can greatly influence the final sequence obtained.

To confirm this fact, the number of ties on the well-known benchmark of instances proposed by Vallada et al. (2008) has been studied. Results are shown in Figure 5 for different values of v, and yield an average of 10.1 ties per iteration, where 210 is the maximum number of ties found in an iteration. The number of ties increases with v and is close to zero for low values of v, which is consistent with the fact that the problem is similar to that of flowtime minimisation. The analysis also shows that, for some instances with a high value of v, an average of around a 40% of the positions where the job is to be inserted has the same total tardiness in each iteration, which represents a huge amount of ties.

In view of the results of the experiments, it can be concluded that the existence of a mechanism to break ties is extremely important for the NEHedd procedure in the $Fm|prmu|\sum T_j$ problem. However, a tie-breaking mechanism is not considered either in the NEHedd procedure, or in the original NEH algorithm for makespan minimisation. In the next section, we propose different tie-breaking mechanisms so the performance of NEHedd procedure can be improved in the most interesting region of the $Fm|prmu|\sum T_j$.

3 Proposed tie-breaking mechanisms

As mentioned in the previous section, no specific tie-breaking mechanism is mentioned in the original NEH heuristic for makespan minimisation. Indeed, it is cited (Nawaz et al., 1983) that '... Next, the job with the third highest total process time is selected and the three partial sequences are tested in which this job is placed at the beginning, middle and end of the partial sequence...', which seems to indicate that the first position where a tie is found is selected. In the following, we will denote this tie-breaking mechanism as FT (First-Tie). Later, in the race for improving the NEH heuristic, Kalczynski and Kamburowski (2007) established the importance of breaking ties in the NEH heuristic and proposed a tie-breaking mechanism to improve the results obtained by the NEH heuristic. Since then, this aspect has been extensively analysed in the literature and several tie-breaking mechanisms have been proposed for the PFSP to minimise makespan (see Kalczynski and Kamburowski, 2008, Dong et al., 2008, Kalczynski and Kamburowski, 2009, Ribas et al., 2010, Kalczynski and Kamburowski, 2011, or Fernandez-Viagas and Framinan, 2014).

To the best of our knowledge, there are no tie-breaking mechanisms proposed for the NEHedd procedure, which adopts the first-tie mechanism as in the original NEH heuristic. However, it has to be noted that, since the EDD rule sorts the jobs according to non-decreasing due dates, in case of ties in the first iterations of NEHedd, the jobs would be finally ordered according to non-increasing due dates, which would probably lead to a worse final sequence than using a different mechanism.

In this section, several tie-breaking mechanisms are proposed to improve the traditional tie-breaking mechanism of the NEHedd procedure. The pseudo-code for the NEHedd algorithm including a generic tie-breaking mechanism is shown in Figure 6.

The proposed tie-breaking mechanisms involve using a secondary indicator related to the performance of the partial sequence. The goal is to pick, among those slots with the same tardiness, the slot yielding the best value of the secondary indicator for the unscheduled jobs. Thereby, total idle time ($IT1$ or $IT2$, see below), total flowtime (CT)), total earliness (ET) and makespan (MS) are chosen as potential secondary indicators. Note that, since these indicators have to be computed for every slot where the job is to be inserted in each iteration of the

Procedure *NEHedd(TB_X)*
  α := Jobs ordered by non-decreasing due dates where $\alpha = \{\alpha_1, ..., \alpha_i, ..., \alpha_n\}$;
  $\pi := \{\alpha_1\}$;
  for $k = 2$ **to** n **do**
    Test job α_k in any possible position of π.
    π := permutation obtained by inserting α_k in the position of π with less total tardiness breaking ties according to an specific mechanism;
  end
end

Figure 6: NEHedd with different tie-breaking mechanisms

algorithm, they have to be carefully chosen so that the additional computational effort pays off.

More specifically, the tie-breaking mechanisms analysed in this paper are:

- **First tie,** $NEHedd(TB_{FT})$. Original tie-breaking mechanism of the NEHedd algorithm proposed in Nawaz et al. (1983) where, in case of ties, the first tie is chosen.

- **Last tie,** $NEHedd(TB_{LT})$. This tie-breaking mechanism simply consists in selecting the last tie as reference for the next iteration. This tie-breaking mechanism tries to solve the problem of TB_{FT} where jobs are sorted according to the reverse EDD rule.

- **Total idle time,** $NEHedd(TB_{IT1})$ **and** $NEHedd(TB_{IT2})$. Denoting *front delay* of a machine as the time until it starts processing the first job, and *back delay* of a machine as the time between completing the processing of the last job and the completion of all jobs in any machine, machine idle time can be ambiguously defined by means of at least three different ways (Framinan et al., 2003), i.e.: idle time including front delays and excluding back delays (denoted as IT1); idle time excluding front and back delays (denoted as IT2); and idle time considering front delays and back delays.

  If we adopt the first definition of idle time, then the idle time of machine i can be calculated as $IT1_i = C_{in} - \sum_{j=1}^{n} p_{ij}$. Consequently, the total idle time is $IT1 = \sum_{i=1}^{m} IT1_i$. Minimising $IT1$ looks for a more compacted schedule of the inserted jobs and it is equivalent to the minimisation of the sum of the completion times of each job in each machine. On the other hand, the second definition of idle time (excluding both delays) can be calculated as $IT2 = \sum_{j=2}^{n} \sum_{i=2}^{m} max\{C_{i-1,j} - C_{i,j-1}, 0\}$. The heuristics resulting from the

use of these tie-breaking mechanisms in NEHedd are denoted as $NEHedd(TB_{IT1})$ and $NEHedd(TB_{IT2})$ respectively. Finally, note that the minimisation of the third definition of idle time is analogous to the minimisation of makespan and, therefore, it is considered below when discussing breaking ties according to the makespan.

- **Total completion time,** $NEHedd(TB_{CT})$. Total completion time can be defined as follows: $ct = \sum_{j=1}^{j} C_{m,j}$. As with idle time, this tie-breaking mechanism tries to balance the use of resources, and the resulting NEHedd heuristic is denoted by $NEHedd(TB_{CT})$.

- **Total earliness,** $NEHedd(TB_{ET})$. If a job finishes before its due date, its earliness indicates the time between the due date and the completion time of the job. Given several sequences with the same total tardiness, sequences with a high value of the total earliness indicate that, on average, the completion times of the jobs are far from their due dates. Thus, breaking ties by maximising earliness looks for sequences with a greater buffer against the due date of each job, which tries to improve the objective function when the following jobs are inserted in any position of the sequence. $NEHedd(TB_{ET})$ is denoted when earliness maximisation is used in the NEHedd algorithm to break ties.

- **Makespan,** $NEHedd(TB_{MS})$. Similarly to the first two tie-breaking mechanisms, the minimisation of the makespan tries to compress the sequence for the subsequent iterations. The NEHedd heuristic using the minimisation of the makespan as tie-breaking mechanism is denoted as $NEHedd(TB_{MS})$.

- **Makespan using Taillard's acceleration,** $NEHedd(TB_{MS-Taillard,IT1})$. As explained in Section 1, Taillard's acceleration represents a huge reduction of the computation time of the NEH algorithm and it is one of the main reasons for its efficiency. However, it cannot be applied to total tardiness minimisation since the completion time of each job in the last machine is needed. To reduce the computation time of the NEHedd algorithm for the tardiness goal, this tie-breaking mechanism applies the NEH algorithm to minimise the makespan, using Taillard's acceleration as long as the tardiness of the (partial) sequence is zero, i.e. in the first iterations of the algorithm when applied. Once the (partial) tardiness is greater than zero, the proposed algorithm minimises the total tardiness (without Taillard's

```
Procedure NEHedd(TB_{MS−Taillard,IT})
    α := Jobs ordered by non-decreasing due dates where α = {α_1, ..., α_i, ..., α_n};
    π := {α_1};
    flag := true;
    for k = 2 to n do
        if flag then
            π_1 := π;
            Test job α_k in any possible position of π_1 (using Taillard's acceleration).
            π_1 := permutation obtained by inserting α_k in the position of π_1 with less
            makespan;
            TT := total tardiness of the sequence π_1;
            if TT > 0 then
                flag := false;
            else
                π := π_1;
            end
        end
        if flag ≠ true then
            Insert job α_k in the position of π which minimises the total tardiness breaking
            ties according to the total idle time IT1 of the sequence.
        end
    end
end
```

Figure 7: $NEHedd(TB_{MS-Taillard,IT})$

acceleration) breaking ties according to the total idle time, $IT1$. The pseudo code of this method is shown in Figure 7.

- **Random,** $NEHedd(TB_{rand})$. A random tie-breaking mechanism is proposed as a baseline for comparisons with the other mechanisms.

4 Computational Experience

Each proposed tie-breaking mechanism has been compared under the same computer conditions, which means the same computer and the same programming language (C#). Algorithms were tested using the set of instances of the benchmark of Vallada et al. (2008) described in Section 2. The different tie-breaking mechanisms were compared by means of the RDI (described in Section 2) as an indicator of the quality of the solution.

Instance	TB_{FT}	TB_{LT}	TB_{rand}	TB_{IT1}	TB_{IT2}	TB_{CT}	TB_{MK}	TB_{ET}	$TB_{MS-Taillard,IT1}$
50x10	17.46	17.46	17.25	13.72	15.22	15.21	14.47	15.21	14.53
50x30	19.79	20.31	19.55	18.61	18.69	18.80	18.74	18.80	18.68
50x50	18.17	17.94	17.88	17.57	18.12	17.88	17.98	17.88	17.97
150x10	13.80	13.60	14.45	9.91	10.91	11.11	10.61	11.11	10.69
150x30	20.70	20.35	20.68	15.81	16.47	18.32	17.83	18.32	17.02
150x50	22.04	21.26	21.70	18.57	19.64	19.96	20.14	19.96	19.64
250x10	10.06	9.46	10.02	6.70	7.31	7.45	7.47	7.45	7.26
250x30	17.81	17.03	17.93	11.62	12.19	14.58	13.82	14.58	13.29
250x50	20.21	19.52	20.13	13.96	14.73	17.49	16.87	17.49	15.90
350x10	9.01	8.59	8.86	6.14	6.30	6.47	6.63	6.47	6.65
350x30	15.74	15.43	15.95	9.84	10.41	12.21	11.88	12.21	11.40
350x50	17.38	16.87	17.11	11.10	11.63	14.01	13.74	14.01	13.11
Average	16.85	16.48	16.79	12.80	13.47	14.46	14.18	14.46	13.84

Table 1: Relative deviation index (RDI) for the NEHedd heuristic using different tie-breaking mechanisms

The results of the heuristics are shown in Table 1 in terms of their values of RDI. The best overall results are found using $IT1$ as tie-breaking mechanism with an average RDI (denoted as $ARDI$) of 12.80, roughly about a 25% less than in the original FT. Note that each tie-breaking mechanism (also including the random mechanism) outperforms on average the original mechanism of the NEHedd algorithm, $NEHedd(TB_{FT})$, which has an $ARDI$ of 16.85. Although the difference between this original tie-breaking mechanism and the $NEHedd(TB_{LT})$ or the $NEHedd(TB_{rand})$ is less than 0.37, for the rest of tie-breaking mechanisms the $ARDI$ is at least a 2.39 lower than that obtained by $NEHedd(TB_{FT})$, which represents an increase in the quality of the solution without increasing the complexity of the algorithm. The CPU times of each algorithm for each combination of n and m are shown in Table 2. The differences between CPU times are negligible with the exception of the $NEHedd(TB_{MS-Taillard,IT1})$, which requires a bit less computational effort and has an $ARDI$ of 5.63, 3.22 lower than that ofFT.

Given that each tie-breaking mechanism is a version of the original NEHedd algorithm and that the same test bed for all tie-breaking mechanisms is used, it is clear that the random variables (RDI) are related and the hypothesis of independence can be rejected (see Table 3 for each comparison). However, the hypothesis of normality is not fulfilled, so a paired samples t-test cannot be used. Two non-parametric statistical hypothesis tests (Wilcoxon signed-rank

Instance	TB_{FT}	TB_{LT}	TB_{rand}	TB_{IT1}	TB_{IT2}	TB_{CT}	TB_{MK}	TB_{ET}	$TB_{MS-Taillard,IT1}$
50x10	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
50x30	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
50x50	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
150x10	0.43	0.42	0.43	0.42	0.42	0.44	0.42	0.43	0.40
150x30	1.29	1.29	1.29	1.30	1.30	1.30	1.29	1.30	1.27
150x50	2.16	2.16	2.16	2.18	2.18	2.17	2.15	2.16	2.15
250x10	1.92	1.90	1.91	1.89	1.89	1.96	1.90	1.95	1.77
250x30	5.86	5.84	5.86	5.85	5.84	5.94	5.85	5.92	5.67
250x50	9.92	9.91	9.95	9.96	9.97	9.99	9.89	9.94	9.78
350x10	5.15	5.14	5.15	5.08	5.07	5.26	5.12	5.24	4.73
350x30	15.86	15.85	15.89	15.80	15.76	16.10	15.85	16.03	15.23
350x50	26.99	26.98	27.06	27.01	27.01	27.26	27.05	27.18	26.35
Average	5.81	5.80	5.82	5.80	5.80	5.88	5.81	5.86	5.63

Table 2: Average CPU times for the NEHedd heuristic with different tie-breaking mechanisms

Comparison	N	Correlation	Sig.
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{IT1})$	540	0.707	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{IT2})$	540	0.760	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{MS-Taillard,IT1})$	540	0.816	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{CT})$	540	0.876	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{MK})$	540	0.826	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{ET})$	540	0.876	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{LT})$	540	0.949	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{rand})$	540	0.962	0.000

Table 3: Analysis of dependence of samples

test and sign test) are carried out then with a confidence level of 99% to compare the statistical significance between the mean and the median of the samples, respectively. Results of the tests are shown in Table 4. For both tests, each tie-breaking mechanism statistically outperforms the original one with the exception of the random tie-breaking mechanism, for which no statistical difference was found (p-values of 0.794 and 0.727 for the Wilcoxon signed-rank test and sign test respectively). Regarding the significance of the different tie-breaking mechanisms, the highest p-value found was 0.004 when comparing TB_{LT} and TB_{FT}, which indicates the relatively bad performance of the original tie-breaking mechanism of the NEHedd procedure. The rest of the p-values are 0.000.

ARDI is shown in Table 5 grouped by the values of the different parameters in the testbed:

Comparison	Wilcoxon signed-rank test		Sign test	
	Z	Sig.	Z	Sig.
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{IT1})$	-14.498	0.000	-12.363	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{IT2})$	-13.665	0.000	-11.446	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{MS-Taillard,IT1})$	-13.829	0.000	-12.020	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{CT})$	-13.616	0.000	-13.207	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{MK})$	-13.246	0.000	-11.810	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{ET})$	-13.616	0.000	-13.207	0.000
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{LT})$	-3.865	0.000	-2.904	0.004
$NEHedd(TB_{FT})$ vs $NEHedd(TB_{rand})$	-0.262	0.794	-0.349	0.727

Table 4: Wilcoxon signed-rank test and sign test

The first and second columns correspond to the value of each parameter in each row according to the values of T, R, n and m of the testbed. The third and fourth columns represent the average number of ties per iteration and the maximum number of ties in an iteration, respectively. The rest of the columns show the $ARDI$ values for each tie-breaking mechanism. $ARDI$ values for each tie-breaking mechanism are always lower than the $ARDI$ of TB_{FT} regardless of the value of the parameters, with the exception of TB_{rand} and TB_{LT}. Although $NEHedd(TB_{LT})$ statistically outperforms $NEHedd(TB_{FT})$ in the whole testbed, this does not happen when grouping by parameters. The minimum difference between the original tie-breaking mechanism and the rest is found for $T = 0.6$ and $R = 1.0$, which corresponds to tighter due dates with high variance. Obviously, the performance of the tie-breaking mechanism is related to the average number of ties. Thereby, note that the average and maximum number of ties decreases as m, T, or R increase, or as n decreases, reaching the maximum value of ties for the following combination of parameters: $T = 0.2$, $R = 0.2$, $n = 350$ and $m = 10$. Regarding the behaviour with respect to indicator v, $CRDI_{NEH\_TB(IT1)}$ the difference between the best tie-breaking mechanism TB_{IT1} as compared to the original one is shown in Figure 8. Most points are below zero in the y-axis, which highlights the improvement achieved by the heuristic when using $IT1$ as a tie-breaking mechanism, especially for $v > 0.15$ where the problem is far from being of the type $Fm|prmu|\sum C_j$.

Figure 8: $CRDI_{NEH\_TB(IT1)}$ in each instance of the Benchmark of Vallada et al. (2008).

Parameter		Ties		Tie-breaking mechanisms								
		Mean	Max.	FT	LT	rand	IT1	IT2	CT	MK	ET	MS-Taillard, IT1
T	0.2	22.5	210	14.57	13.91	14.47	7.18	8.27	10.45	9.71	10.45	8.90
T	0.4	6.5	144	18.14	17.92	18.08	14.15	14.95	15.84	15.67	15.84	15.45
T	0.6	1.1	81	17.83	17.62	17.83	17.06	17.18	17.09	17.17	17.09	17.19
R	0.2	17.1	210	20.86	19.72	20.49	13.14	14.27	15.65	15.02	15.65	14.96
R	0.6	8.3	146	16.83	16.70	16.84	12.88	13.60	15.00	14.84	15.00	14.07
R	1	4.7	112	12.86	12.98	12.95	12.41	12.51	12.69	12.77	12.69	12.57
n	50	0.5	23	18.25	18.28	17.85	16.81	17.32	17.18	17.20	17.18	17.15
n	150	3.8	81	18.85	18.40	18.94	14.76	15.67	16.47	16.19	16.47	15.78
n	250	8.9	153	16.03	15.33	16.02	10.76	11.41	13.17	12.72	13.17	12.15
n	350	14.9	210	14.11	13.70	14.04	9.13	9.55	10.98	10.84	10.98	10.47
m	10	16.0	210	12.37	12.02	12.32	9.13	9.80	9.90	9.82	9.90	9.75
m	30	8.7	182	18.25	18.06	18.38	13.82	14.26	15.79	15.43	15.79	14.97
m	50	5.4	162	19.30	18.78	19.04	15.16	15.86	17.20	17.05	17.20	16.52

Table 5: Average number of ties for iteration, maximum number of ties in an iteration and $ARDI$ for each tie-breaking mechanism.

Stopping Criterion	*ARDI*-GAPR		Wilcoxon signed-rank test	Sign test
	$NEHedd(TB_{FT})$	$NEHedd(TB_{IT1})$	p-value	p-value
$t = 0.5$	14.66	11.01	0.000	0.000
$t = 1$	12.65	9.72	0.000	0.000
$t = 2$	10.61	8.42	0.000	0.000
$t = 5$	7.57	6.65	0.000	0.000
$t = 10$	6.25	5.63	0.000	0.000
$t = 20$	5.09	4.71	0.000	0.000

Table 6: ARDI, Wilcoxon signed-rank test and sign test for the GAPR algorithm when it is initialized with $NEHedd(TB_{IT1})$ and $NEHedd(TB_{FT})$

4.1 Influence of the proposed tie-breaking mechanisms on iterative improvement algorithms

In this section, we evaluate the influence of the proposed NEH-based heuristics when they are incorporated as seed sequences in iterative improvement algorithms. For this comparison, we use the genetic algorithm, GAPR, proposed by Vallada and Ruiz (2010). Three types of genetic algorithms were proposed. Each one was shown to be statistically more efficient than other iterative improvement algorithms in the literature for three different stopping criteria. The GAPR algorithm uses a fast selection mechanism denoted as n-tournament as well as the path relinking as crossover mechanism. As initial solution, the algorithm uses 28 random sequences and two individuals provided by the original NEHedd algorithm and by the EDD despatching rule. To analyse the influence of the chosen tie-breaking mechanism, we substitute the NEHedd seed sequence by the best proposed NEHedd-based constructive heuristic, i.e. $NEHedd(TB_{IT1})$, and we compare them using the same benchmark as in the previous Section. Average computational results in terms of *ARDI* are shown in Table 6 and in Figure 9 for six different stopping criteria to observe the evolution of the performance for different CPU times, $t \cdot n \cdot (m/2)$ with $t \in [0.5, 1, 2, 5, 10, 20]$ expressed in milliseconds.

Obviously, one might expect that the influence of the initial solution on a well-designed metaheuristic such as the GAPR would decrease with the CPU time. Still, for the range of CPU times employed (which represents around 3 minutes of CPU times per instance for the biggest sizes), the results show that our proposal positively impacts on the quality of the solution. In fact,

Figure 9: Evolution of the GAPR algorithm with different initial solutions for six different stopping criteria.

the positive contribution of the tie-breaking mechanism is found to be statistically significant for every stopping criteria, for both non-parametric statistical hypothesis tests (Wilcoxon signed-rank test and sign test). The highest found p-value was 0.000 (see Table 6).

5 Conclusions

In this paper, several tie-breaking mechanisms for the NEH heuristics have been proposed to solve the $Fm|prmu|\sum T_j$ problem. It is clear that, depending on the due dates, the decision problem to be solved is different. Extremely tight due dates induce to a $Fm|prmu|\sum C_j$ problem, whereas very loose due dates lead to a trivial problem. Thereby, the problem has been first analysed in detail, depicting the limits between the tardiness problem and other problems. As a conclusion, it was obtained that several testbeds generate instances for a problem more similar to $Fm|prmu|\sum C_j$. Additionally, it has been found that the number of ties in each iteration of the NEHedd heuristic is very high outside these limits (i.e. the most interesting setting regarding tardiness minimisation), and that the original tie-breaking mechanism of NEHedd would result in worse sequences as it orders the jobs in non-increasing due dates in the very likely case of ties in the first iterations. To address this problem and to enhance the performance of the

NEHedd procedure, a set of eight tie-breaking mechanism have been proposed. These are tested against the original one in an extensive computational evaluation, and the results show that some of these mechanisms improve the performance of the NEHedd procedure by more than 25% while requiring similar computation time. Additionally, when embedding this mechanism as seed sequence in a state-of-the-art iterative improvement algorithm, the performance of the resulting algorithm significantly improves that of the original one.

Regarding future research lines, although the due date generation mechanism by Potts and Van Wassenhove (1982) has been chosen to build the testbed, further analysis could be conducted to develop more extensive testbeds, including bigger intervals for indicator v.

Acknowledgements

The authors are sincerely grateful to the anonymous referees, who provide very valuable comments on the earlier version of the paper. This research has been funded by the Spanish Ministry of Science and Innovation, under projects "SCORE" with reference DPI2010-15573/DPI, and "ADDRESS" with reference DPI2013-44461-P/DPI.

References

Brah, S. and Loo, L. (1999). Heuristics for scheduling in a flow shop with multiple processors. *European Journal of Operational Research*, 113(1):113–122.

Dong, X., Huang, H., and Chen, P. (2008). An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research*, 35(12):3962–3968.

Fernandez-Viagas, V. and Framinan, J. M. (2014). On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers and Operations Research*, 45(0):60 – 67.

Framinan, J., Gupta, J., and Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55(12):1243–1255.

Framinan, J. and Leisten, R. (2008). Total tardiness minimization in permutation flow shops: A simple approach based on a variable greedy algorithm. *International Journal of Production Research*, 46(22):6479–6498.

Framinan, J., Leisten, R., and Ruiz-Usano, R. (2005). Comparison of heuristics for flowtime minimisation in permutation flowshops. *Computers and Operations Research*, 32(5):1237–1254.

Framinan, J. M., Leisten, R., and Rajendran, C. (2003). Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the

static permutation flowshop sequencing problem. *International Journal of Production Research*, 41(1):121–148.

Gelders, L. F. and Sambandam, N. (1978). Four simple heuristics for scheduling a flow-shop. *International Journal of Production Research*, 16(3):221–231.

Hasija, S. and Rajendran, C. (2004). Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 42(11):2289–2301.

Kalczynski, P. J. and Kamburowski, J. (2007). On the NEH heuristic for minimizing the makespan in permutation flow shops. *OMEGA, The International Journal of Management Science*, 35(1):53–60.

Kalczynski, P. J. and Kamburowski, J. (2008). An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers & Operations Research*, 35(9):3001–3008.

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

Kalczynski, P. J. and Kamburowski, J. (2011). On recent modifications and extensions of the NEH heuristic for flow shop sequencing. *Foundations of Computing and Decision Sciences*, 36(1):17–34.

Kim, Y.-D. (1993). Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of the Operational Research Society*, 44(1):19–28.

Kim, Y.-D., Kim, J.-G., Choi, B., and Kim, H.-U. (2001). Production scheduling in a semiconductor wafer fabrication facility producing multiple product types with distinct due dates. *IEEE Transactions on Robotics and Automation*, 17(5):589–598.

Kim, Y.-D., Lim, H.-G., and Park, M.-W. (1996). Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research*, 91(1):124–143.

Li, X., Wang, Q., and Wu, C. (2009). Efficient composite heuristics for total flowtime minimization in permutation flow shops. *Omega*, 37(1):155–164.

Nawaz, M., Enscore Jr., E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA, The International Journal of Management Science*, 11(1):91–95.

Pan, Q.-K. and Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers and Operations Research*, 40(1):117–128.

Panwalkar, S., Dudek, R., and Smith, M. (1973). Sequencing research and the industrial problem. In *Symposium on the Theory of Scheduling*. Springer, Berlin.

Panwalkar, S., Smith, M., and Seidmann, A. (1982). Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operations Research*, 30(2):391–399.

Pinedo, M. (1995). *Scheduling: Theory, Algorithms and Systems*. Prentice Hall.

Potts, C. and Van Wassenhove, L. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, 1(5):177–181.

Rajendran, C. and Ziegler, H. (2003). Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. *European Journal of Operational Research*, 149(3):513–522.

Raman, N. (1995). Minimum tardiness scheduling in flow shops: Construction and evaluation of

alternative solution approaches. *Journal of Operations Management*, 12(2):131–151.

Ribas, I., Companys, R., and Tort-Martorell, X. (2010). Comparing three-step heuristics for the permutation flow shop problem. *Computers & Operations Research*, 37(12):2062–2070.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494.

Schaller, J. (2012). Scheduling a permutation flow shop with family setups to minimise total tardiness. *International Journal of Production Research*, 50(8):2204–2217.

Sen, T. and Gupta, S. (1984). A state-of-art survey of static scheduling research involving due dates. *Omega*, 12(1):63–76.

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

Vallada, E. and Ruiz, R. (2010). Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, 38(1-2):57–67.

Vallada, E., Ruiz, R., and Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers and Operations Research*, 35(4):1350–1373.

View publication stats