On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem *

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

May 21, 2015

Abstract

The most efficient approximate procedures so far for the flowshop scheduling problem with makespan objective –i.e. the NEH heuristic and the Iterated Greedy Algorithm– are based on constructing a sequence by iteratively inserting, one by one, the non-scheduled jobs into all positions of an existing subsequence, and then, among the so obtained subsequences, selecting the one yielding the lowest (partial) makespan. This procedure usually causes a high number of ties (different subsequences with the same best partial makespan) that must be broken via a tie-breaking mechanism. The tie-breaking mechanism employed is known to have a great influence in the performance of the NEH, therefore different procedures have been proposed in the literature. However, to the best of our knowledge, no tie-breaking mechanism has been proposed for the Iterated Greedy. In our paper, we present a new tie-breaking mechanism based on an estimation of the idle times of the different subsequences in order to pick the one with the lowest value of the estimation. The computational experiments carried out show that this mechanism outperforms existing ones both for the NEH and the Iterated Greedy for different CPU times. Furthermore, embedding the proposed tie-breaking mechanism into the Iterated Greedy provides the most efficient heuristic for the problem so far.

Keywords: Scheduling, Flowshop, Heuristics, NEH.

*Preprint submitted to Computers & Operations Research. http://dx.doi.org/10.1016/j.cor.2013.12.012

†Corresponding author. Tel.: +34-954487214; fax: +34-954487329.

1 Introduction

The permutation flowshop scheduling problem with makespan objective (also known as problem $Fm|prmu|C_{max}$, in the notation by Graham et al., 1979) involves the determination of the order of processing of n jobs on m machines while all jobs have the same machine sequence. This problem is, without doubt, one of the most studied problems in Operations Research (see in this regard the reviews by Framinan et al., 2004; Reza Hejazi and Saghafian, 2005; Ruiz and Maroto, 2005). There are several reasons for this fact: On the one hand, the flow shop layout is the common configuration in many manufacturing scenarios, as it presents a number of advantages over the more general job shop configuration (see e.g. Vakharia and Wemmerlov, 1990; Krajewski et al., 1987), and, in addition, many job shops are a flow shop for most of the jobs (Storer et al., 1992). On the other hand, makespan minimisation leads to the minimisation of the total production run, therefore maximising machine utilisation and thus minimising fixed unit costs.

Aside to the practical relevance of the problem, since the early work by Johnson (1954), contributions on the $Fm|prmu|C_{max}$ problem have pioneered the research in scheduling with different objectives and layouts, and many models and solution procedures for the latter problems have their origins in the flowshop scheduling problem with makespan objective.

Since Rinnooy Kan (1976) proved the $Fm|prmu|C_{max}$ problem to be NP-complete if the number of machines is higher than two, most of the contributions have focused on providing approximate methods yielding good (but nor necessarily optimal) solutions in reasonable time. In the race for designing efficient heuristics for the problem, a breakthrough was obtained by Nawaz et al. (1983) when they proposed the NEH heuristic. This heuristic consists of two phases: In the first phase, jobs are arranged with respect to the descending sums of their processing times. Within the second phase (denoted by insertion phase in this paper), a job sequence is constructed by evaluating the partial schedules originating from the initial order given by the first phase: Assuming a sequence already determined for the first $k-1$ jobs, k candidate (sub)sequences are obtained by inserting job in position 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 relative (sub)sequence for these first k jobs given by phase one. Then, job in position $k+1$ from the first phase is considered analogously, and so on until all n jobs have been sequenced.

Note that the complexity of this heuristic is $O(n^3m)$, as the evaluation of an m-machine makespan can be accomplished in $O(nm)$ and the evaluation of the k subsequences resulting in step k can be completed in $O(n^2m)$. However, Taillard (1990) proposed a mechanism –named *Taillard's acceleration* in the following– so the evaluation of the k subsequences can be done in $O(nm)$ thus reducing the overall complexity of the heuristic to $O(n^2m)$.

If we consider the NEH heuristic as a particular case of a family of heuristics, there are several elements (options) within this family. These are:

- Starting order, i.e. how to obtain an initial order in which the jobs are arranged in the first phase.
- Sequence generation, i.e. how the candidate (sub)sequences are generated from the initial starting order.
- Tie-breaking mechanism, i.e. how ties are treated in the evaluation of the candidate (sub)sequences.

The starting order determines which job is to be picked for insertion in the current (sub)sequence. The original proposal by Nawaz et al. (1983) is to arrange the jobs in descending order of the sum of their processing times. Framinan et al. (2003) conducted an extensive study with different initial orders and showed that there were significant differences among them and that, the original order remained the best for the makespan objective. These results were later confirmed by Kalczynski and Kamburowski (2007). Nagano and Moccellin (2002) proposed a different starting order based on an estimation of an idle time of the jobs. Although the authors claimed that their proposal outperforms the original NEH, an extensive simulation study carried out by Kalczynski and Kamburowski (2008) showed that this seems to be true only for $m < 6$ and that the resulting differences were not statistically significant. The latter authors also proposed an initial starting order which they claim to outperform the original one. Dong et al. (2008) proposed a modification of the NEH heuristic in which a specific mechanism for tie-breaking is applied in addition to a starting order based on the mean and the variance of the processing times of the jobs and finally, Kalczynski and Kamburowski (2009) proposed a new modification of the classical. Although this last modification outperforms the original NEH (and the modification of the NEH proposed by Kalczynski and Kamburowski, 2008 and Dong et al., 2008, see Kalczynski and Kamburowski, 2011) in an extended test bed proposed by Kalczynski and Kamburowski (2009), it was not proved that the proposed starting order was superior to the original in the benchmark set of Taillard (1993) where the starting order proposed by Dong et al. (2008) presents the best results.

With respect to sequence generation, the original proposal is to insert the job in the k possible slots of the current sequence. However, it is clear that different strategies could be adopted, either by reducing the number of candidates (by e.g. evaluating just a fraction of the k possible slots), or by exploring more candidates. With respect to the former strategies, Rajendran (1993) limited the insertion to positions $\lfloor k/2 \rfloor$ to k with good results, while different strategies have been proposed for exploring more candidate solutions by Rad et al. (2009) (note that other strategies have been explored for the total flowtime by Woo and Yim, 1998 and by Framinan and Leisten, 2003, but there is no proof that they are efficient with respect to makespan). In all these contributions, the gains (losses) in the quality of the solutions are compensated by the increase (decrease) in CPU time requirements.

Finally, with respect to the tie-breaking mechanism, modifications with respect to the original tie-breaking mechanism have been suggested by several authors. Note that, in general, tie-breaking mechanisms may refer either to the starting order (i.e. how to rank jobs with the same indicator value in the initial ordering sequence), or to the sequence generation phase (i.e. how to choose among different subsequences with the same best partial makespan). In this paper we focus on the second type –labelled insertion tie-breaking in the following– so existing contributions will be discussed in detail in Section 2.

With or without the aforementioned modifications, NEH has turned out to be the most efficient heuristic found for the problem, and nowadays it remains the cornerstone of subsequent heuristics that have been proposed for the problem and that can be seen as refinements and/or enhancements of NEH. The reason for this efficiency probably lies in the procedure employed for inserting and evaluating –using Taillard's acceleration– the non-scheduled jobs, a mechanism also present in the Iterated Greedy Algorithm (denoted as IG_RS_{LS} in the following) proposed by Ruiz and Stützle (2007) and considered among the best heuristics for the problem (see Ruiz and Stützle, 2007; Pan et al., 2008). IG_RS_{LS} starts with an initial sequence π provided by the NEH heuristic, and then applies an iterative improvement scheme consisting of the following phases:

- Destruction phase. Here, d jobs are randomly removed from the sequence, π, without repetition. The sequence without these jobs is denoted by π_D and is formed by $n-d$ jobs.
- Construction phase. The d destructed jobs are inserted one by one in the sequence π_D following the same procedure as in the insertion phase of the NEH heuristic but only for the last d jobs (using again Taillard's acceleration). The final sequence is denoted by π'.
- Local search. The solution generated in the construction phase is improved by a local search phase. This phase successively removes a job from the sequence π' and inserting this job in the best possible position (using Taillard's acceleration).
- Acceptance criterion. A simple simulated annealing-like acceptance criterion with a constant temperature, T, is considered.

These four phases are repeated until the stopping criterion (usually a maximum CPU-time) is reached. More recently, the local search phase of IG_RS_{LS} has been improved by Pan et al. (2008) by means of a so-called referenced insertion scheme (RIS), which is denoted as IG_{RIS} in the following. Nevertheless, the insertion scheme remains the same, so IG_RS_{LS} could be considered as an extremely efficient heuristic for the problem.

Both in NEH and the Iterated Greedy algorithm, ties among (sub)sequences yielding the lowest makespan may occur. In the original proposals, no specific mention on ties is given, so it is usually assumed that the first slot for which the minimum makespan is achieved when inserting job in position k is kept as the best (sub)sequence. However, the mechanism employed to break these ties has a great influence on the performance of these algorithms, as Kalczynski and Kamburowski (2007) first attested for the NEH. To the best of our knowledge, there is no proposal of integrating tie-breaking mechanisms in the Iterated Greedy algorithm. In this paper, we propose a new tie-breaking mechanism that outperforms existing ones both in the NEH and in the Iterated Greedy.

The paper is organized as follows: in Section 2, the problem under consideration is formally stated and the existing tie-breaking mechanisms are presented. Section 3 is devoted to explain the proposed tie-breaking mechanism. In Section 4, our proposal is compared against existing tie-breaking mechanisms when embedded in the NEH, and in the IG_RS_{LS} and IG_{RIS} algorithms. Finally, in Section 5, the main conclusions are discussed.

2 Problem statement. Tie-breaking mechanisms

The problem under consideration can be stated as follows: There are n jobs to be scheduled in a flowshop consisting of m machines. On each machine i, each job j has a processing time denoted as t_{ij}. Given a sequence of jobs $\pi := (\pi_1, \dots \pi_n)$, let us denote $p_{ij}(\pi)$ the processing time of job π_j on machine i, i.e. $p_{ij}(\pi) = t_{i\pi_j}$. Whenever it does not lead to confusion, this notation is abbreviated to p_{ij}. Analogously, $C_{ij}(\pi)$ (abbreviated to C_{ij} whenever it does not lead to confusion) denotes the completion time of job π_j on machine i. C_{ij} can be calculated in the following recursive manner:

$$C_{ij} = max\{C_{i-1,j}, C_{i,j-1}\} + p_{ij} \tag{1}$$

where $C_{0j} = C_{i0} = 0$.

Then, the makespan or maximum completion time is C_{mn}.

Let us assume that a partial schedule of $k-1$ jobs has been constructed. An unscheduled job r (whose processing time in machine i is denoted by t_{ir}) is to be inserted in position l ($l = 1 \dots k$), thus obtaining k partial sequences of k jobs denoting $\pi(l)$ the sequence when the unscheduled job is inserted in position l. Additionally, let us denote e_{ij} the earliest completion time of job π_j in machine i before inserting the unscheduled job. e_{ij} can be calculated as follows:

$$e_{ij} = max\{e_{i,j-1}, e_{i-1,j}\} + p_{ij}, i = 1 \dots m, j = 1 \dots k-1 \tag{2}$$

with $e_{0j} = 0$, and $e_{i0} = 0$. Similarly, q_{ij} the duration between the starting time of job π_j on machine i (before inserting the unscheduled job) and the end of all operations can be calculated according to the following expression:

$$q_{ij} = max\{q_{i+1,j}, q_{i,j+1}\} + p_{ij}, i = m \dots 1, j = k-1 \dots 1 \tag{3}$$

with $q_{m+1,j} = 0$, and $q_{i,k} = 0$.

One possibility to calculate the makespan of each of these k sequences is to use equation (1) for each sequence, which results in a complexity $O(n^2m)$. Taillard (1990) proposed a mechanism based on equations (2) and (3) to reduce this complexity to $O(nm)$: Since the earliest completion times of the jobs in π prior to position l have not changed, then f_{il} the completion time on machine i of job inserted in position l can be computed in the following manner:

$$f_{il} = \max\{e_{i,l-1}, f_{i-1,l}\} + t_{ir}, i = 1 \dots m \tag{4}$$

with $f_{0l} = 0$. Therefore, $C_{max}(l)$ the completion time of sequence $\pi(l)$ is:

$$C_{max}(l) = \max_{i=1 \dots m} \{f_{il} + q_{il}\} \tag{5}$$

Given the ability of this mechanism to quickly evaluate the makespans resulting of inserting a job into all positions of a partial schedule, a greedy procedure which selects the best out of these makespans has been incorporated by the most remarkable heuristics for the problem (including NEH and IG_RS_{LS}). Quite often, such procedure originates different schedules with the same (best) makespan. In the literature, mechanisms for breaking these ties have been proposed by Kalczynski and Kamburowski (2007, 2008), by Dong et al. (2008), and by Ribas et al. (2010). In Kalczynski and Kamburowski (2011), an analysis of the three first tie-breaking mechanisms in the NEH was performed, being the best the one by Dong et al. (2008). Nevertheless, these mechanisms were not tested for the IG.

All of the aforementioned mechanisms (with the exception of that by Ribas et al., 2010) can be implemented in time $O(n \cdot m)$ using Taillard's acceleration, therefore not altering the original complexity of NEH and IG_RS_{LS}. The tie-breaking mechanism of Ribas et al. (2010) is either $O(n^2 \cdot m)$ or $O(n \cdot m^2)$, depending on the use or not of Taillard's acceleration. Therefore, its implementation in the NEH heuristic results in a complexity of the algorithm $O(n^2m^2)$ or $O(n^3m)$. In addition, since the NEH with this tie-breaking mechanism does not improve that of the NEH with Dong's tie-breaking mechanism (see Ribas et al., 2010) we exclude Ribas et al. tie-breaking mechanism from the analysis. The rest of tie-breaking mechanisms are discussed in the next subsections.

2.1 Dong's Tie-breaking mechanism

Dong et al. (2008) presents a tie-breaking mechanism (TB_D) to decide the position l where an unscheduled job r is to be inserted in case of ties. This tie-breaking mechanism is based on:

- $EC_{i,l-1}$ the earliest possible completion time of the job in position $l-1$ where l corresponds to a position which minimises the makespan;
- $S_{i,l+1}$ the latest possible start time of the job in position $l+1$; and
- t_{ir} the processing time of the unscheduled job r in machine i.

In order to determine the position where job r is inserted, the position $l(l = 1, \dots, k)$ minimising D_l as defined in equation (6) is chosen.

$$D_l = \sum_{i=1}^{m} \left(\frac{t_{ir}}{S_{i,l+1} - EC_{i,l-1}} - E_l \right)^2 \tag{6}$$

where:

$$E_l = \frac{1}{m} \sum_{i=1}^{m} \frac{t_{ir}}{S_{i,l+1} - EC_{i,l-1}} \tag{7}$$

2.2 Kalczynski & Kamburowski's Tie-breaking mechanism I

This tie-breaking mechanism is due to Kalczynski and Kamburowski (2007), and it is denoted by TB_{KK1} onwards. It chooses the first index for which the minimum value is achieved for the sequence ρ if $min(a_\rho, b_r) \geq min(a_r, b_\rho)$. Otherwise, the last index for which the minimum is achieved is chosen. Parameters a_ρ, a_r, b_ρ and b_r are defined according to the expressions (8), (9),

(10) and (11).

$$a_r = \sum_{i=1}^{m} t_{ir} - t_{mr} \tag{8}$$

$$b_r = \sum_{i=1}^{m} t_{ir} - t_{1r} \tag{9}$$

$$a_\rho = C_{max}(\rho) - \sum_{j\in\rho} p_{mj} \tag{10}$$

$$b_\rho = C_{max}(\rho) - \sum_{j\in\rho} p_{1j} \tag{11}$$

2.3 Kalczynski & Kamburowski's Tie-breaking mechanism II

To improve the tie-breaking mechanism described in Section 2.2, Kalczynski and Kamburowski (2008) propose replacing the aforementioned parameters a_ρ, a_r, b_ρ and b_r by two new parameters $\hat{a}_r$ and $\hat{b}_r$, as defined by expressions (12) and (13). Thereby, the first index (for which the minimum is reached) is chosen in case of ties when $\hat{a}_r \geq \hat{b}_r$ and otherwise, it is chosen the last index. In this paper, TB_{KK2} is employed to denote this tie-breaking mechanism.

$$\hat{a}_r = \sum_{i=1}^{m} \left[(m-1)\frac{(m-2)}{2} + m - i \right] \cdot t_{ir} \tag{12}$$

$$\hat{b}_r = \sum_{i=1}^{m} \left[(m-1)\frac{(m-2)}{2} + i - 1 \right] \cdot t_{ir} \tag{13}$$

3 The proposed tie-breaking mechanism

The tie-breaking mechanism presented in this paper (denoted by TB_{FF} in the following) is related to the minimisation of total idle times. According to Framinan et al. (2003), the definition of machine idletime is not unambiguous and at least three different ways have been used:

- The idletime considering front delays (time before first job) and back delays (time after the last job on the machine).
- Excluding front and back delays.
- Including front delays and excluding back delays.

In this paper, we assume the third definition of the idle time. Therefore, it_i the idle time of machine i can be calculated according to the expression $it_i = C_{in} - \sum_{j=1}^{n} p_{ij}$ and the total idle time by $it = \sum_{i=1}^{m} it_i$. If we denote by Δ_{ij} the idle time in machine i induced between the completion of job j and the beginning of job $j+1$, then Δ_{ij} can be written in terms of equations (2) as follows:

$$\Delta_{ij} = (e_{i,j+1} - p_{i,j+1}) - e_{ij} \tag{14}$$

The first two terms in the right side of the equation indicate the starting time of job $j+1$, therefore subtracting the completion time of job j yields the idle time between jobs j and $j+1$ in machine i. Clearly, $it_i = \sum_{j=0}^{n-1} \Delta_{ij}$, and therefore $it = \sum_{i=1}^{m} \sum_{j=0}^{n-1} \Delta_{ij}$

In order to explain the tie-breaking mechanism, let us assume that we have a subsequence of $k-1$ jobs (see Figure 1). Then, an unscheduled job r is going to be inserted in all positions in the subsequence in order to select the position yielding the minimum makespan. If ties occur, then the position whose insertion yields the minimum total idle time is to be selected. Note that, if the unscheduled job is to be inserted in position l, then $g_{i,l-1}$ the cumulative idle times on machine i induced by jobs prior to position $l-1$ is:

$$g_{i,l-1} = \sum_{j=0}^{l-2} \Delta_{ij} = \sum_{j=1}^{l-1} [(e_{ij} - p_{ij}) - e_{i,j-1}] \tag{15}$$

Analogously, h_{il} the cumulative idle times on machine i induced by jobs after position l is:

$$h_{il} = \sum_{j=l}^{k-2} \Delta_{ij} = \sum_{j=l}^{k-2} [(e_{i,j+1} - p_{i,j+1}) - e_{ij}] \tag{16}$$

It is clear that, for each machine i, when an unscheduled job r (with t_{ir} its processing time on machine i) is inserted in position l (see example in Figure 2), $g_{i,l-1}$ remains the same. However, this does not happen for h_{il}, which would have to be recomputed. Unfortunately, doing so would substantially increase the computation time since Taillard's acceleration cannot be employed to calculate the new idle times. As a consequence, we suggest using an estimation of the idle time as tie-breaking indicator, based on the assumption that the new Δ_{ij} values for jobs in positions

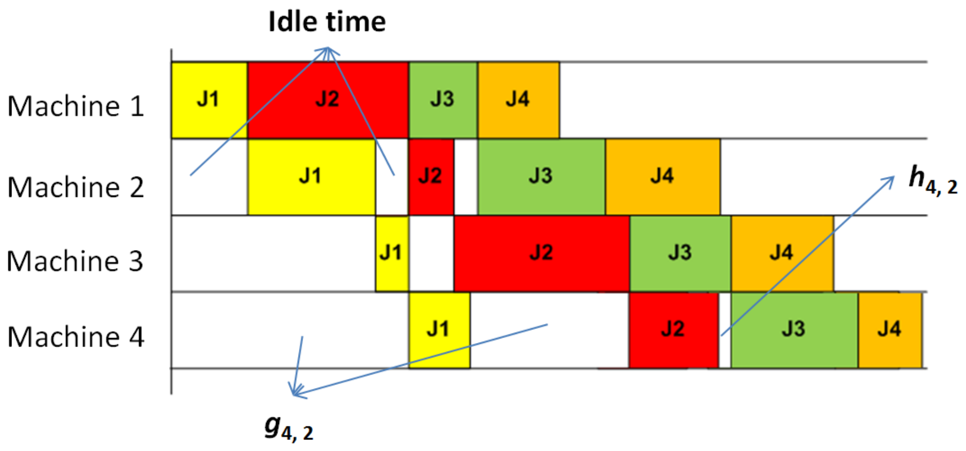

Figure 1: Sequence of jobs before inserting the new job in position l

$l + 1$ to k are not very different from the old ones. Therefore, when inserting an unscheduled job, e_{ij}, q_{ij} and f_{il} are used according to equations (2-4) in order to obtain the makespans for each position. Then, the position yielding the minimum makespan is selected. In case of ties, we calculate an estimation of the new idle time denoted by $it^{'}(l)$ for each position l for which the tie occurs, and selects the position l yielding the minimum makespan for which $it^{'}(l)$ is minimum:

$$it'(l) = \sum_{i=1}^{m} \left(g_{i,l-1} + h_{il} + \Delta^{'}_{i,l-1} + \Delta^{'}_{il}\right) \tag{17}$$

The first term in Equation (17) denotes the idle time in machine i caused by the jobs prior to position $l-1$. This value has been already obtained, as it has not been modified by the insertion of the job. The second term is the idle time in machine i caused by jobs in (old) positions l to $k-1$ (now positions $l+1$ to k once the job is inserted). As stated before, after the insertion of job l, this is not anymore the idle time of the new sequence, but we will assume that they are the same (hence the estimation). Finally, the insertion of the job in position l induces a new idle time between the job in position $l-1$ and the new job (denoted by $\Delta^{'}_{i,l-1}$), and between the new job and the job in the old position l ($l+1$ after the insertion), denoted by $\Delta^{'}_{i,l}$. Both terms can be easily calculated from the data obtained from Taillard's acceleration:

$$\Delta^{'}_{i,l-1} = (f_{il} - t_{ir}) - e_{i,l-1} \tag{18}$$

and

$$\Delta'_{i,l} = \max\{f'_{i-1,l} - f_{il}, 0\} \tag{19}$$

where $f'_{i-1,l}$ is the completion time on machine i of the job which before was in position l (after inserting the new job, it corresponds to the job placed in position $l+1$) and can be computed as follows:

$$f'_{il} = \max\{f_{il}, f'_{i-1,l}\} + p_{il}, i = 1 \ldots m \tag{20}$$

and $f'_{0l} = 0$ being $p_{i,l}$ the processing time of the job that before was in position l.

Note that Equation (17) can be simplified by means of the idle time, $\Delta_{i,l-1}$, between the job in position $l-1$ and l:

$$it'(l) = \sum_{i=1}^{m} \left(g_{i,l-1} + h_{il} + \Delta_{i,l-1} - \Delta_{i,l-1} + \Delta'_{i,l-1} + \Delta'_{il}\right) \tag{21}$$

$$it'(l) = \sum_{i=1}^{m} (g_{i,l-1} + h_{il} + \Delta_{i,l-1}) + \sum_{i=1}^{m} \left(\Delta'_{i,l-1} + \Delta'_{il} - \Delta_{i,l-1}\right) \tag{22}$$

Equation (22) can be decomposed into two terms, i.e.:

$$it'(l) = C + it''(l) \tag{23}$$

where $C = \sum_{i=1}^{m} (g_{i,l-1} + h_{il} + \Delta_{i,l-1})$ is a constant that does not depend on the tie-breaking l, and $it''(l)$ is:

$$it''(l) = \sum_{i=1}^{m} \left(\Delta'_{i,l-1} + \Delta'_{il} - \Delta_{i,l-1}\right) = \sum_{i=1}^{m} \left(f_{il} - e_{il} + p_{il} - t_{ir} + \max\{f'_{i-1,l} - f_{il}, 0\}\right) \tag{24}$$

where it has been used that $\Delta_{i,l-1} = (e_{il} - p_{il}) - e_{i,l-1}$, see Equation (14). Additionally, since $\sum_{i=1}^{m} t_{ir}$ is the same regardless the position l where the job is inserted, we can define $it'''(l)$ a more concise indicator equivalent to $it''(l)$ as follows:

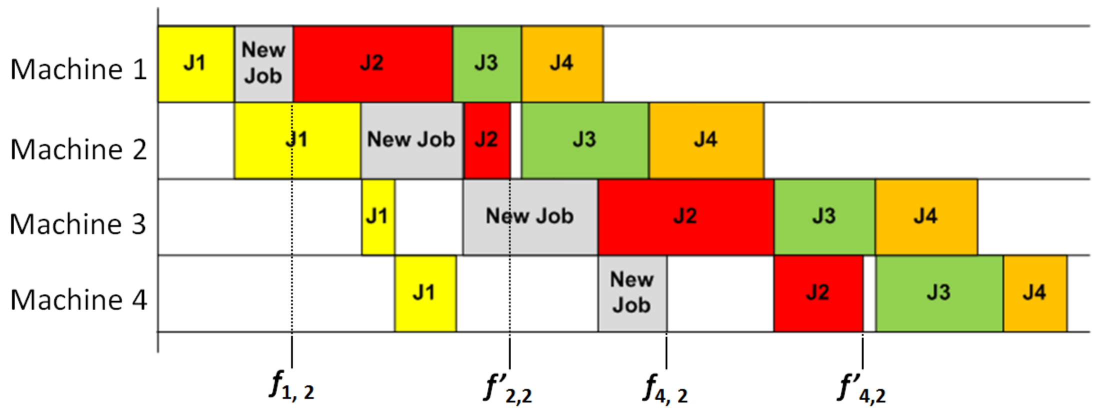

Figure 2: Sequence of jobs after inserting the job in position l

$$it^{'''}(l) = \sum_{i=1}^{m} (f_{il} - e_{il} + p_{il} + \max\{f^{'}_{i-1,l} - f_{il}, 0\}) \tag{25}$$

Therefore, the proposal breaks the ties according to the minimisation of $it^{''}(l)$ –or, equivalently, to the minimisation of $it^{'''}(l)$. The pseudo code of this tie-breaking mechanism for the NEH is shown in Figure 3. Note that the idle time $it^{''}(l)$ is forced to be zero for the last job to be inserted, i.e. no tie-breaking mechanism is considered for the last job to be inserted. It can be easily checked that the insertion of tie-breaking mechanism does not alter the complexity of the algorithm, i.e. it remains $O(n^2 \cdot m)$. Analogously, this mechanism can be easily incorporated in the constructive and in the local search phase of IG_RS_{LS} Ruiz and Stützle (2007), and in the IG_{RIS} by Pan et al. (2008).

4 Computational experience

The tie-breaking mechanisms described in the previous section have been coded in C$\#$ and embedded into the NEH and the two versions of the Iterated Greedy. As for initial ordering in the NEH, the non-increasing order of the sum of the processing times has been adopted. This is the initial order of the original NEH and it has been chosen because, on one hand, it is the most widely-employed mechanism and the results are easier to compare with the rest of the literature. On the other hand, this allows focusing exclusively on insertion tie-breaking mechanisms and removes the possible influence of more elaborated initial ordering rules such as the ones discussed

```
$\pi^{'} \longleftarrow$ Sort in decreasing order of sum of processing times $p_{ij}$;
$\pi \longleftarrow \pi^{'}_1$;
for $k = 2$ to $n$ do
    $r \longleftarrow {\pi_k}'$;
    Determine the values of $e_{ij}$, $q_{ij}$ and $f_{il}$ from Taillard's acceleration (see equations 2, 3,
    and 4);
    Determine minimal makespan resulted of testing the job $r$ in all possible positions of
    $\pi$;
    $bp \longleftarrow$First position where the makespan is minimal;
    $tb \longleftarrow$ Number of positions with minimal makespan (i.e. number of ties);
    $ptb \longleftarrow$ Array (of length $tb$) with the positions where the makespan is minimal;
    $it_{bp}$ is the idletime corresponding to the $bp$ and set to a very large number;
    if $tb > 1$ and $k < n$ then
        for $l = 1$ to $tb$ do
            $it^{''} \longleftarrow 0$;
            if $ptb[l] = k$ then
                for $i = 2$ to $m$ do
                    $it^{''} \longleftarrow it^{''} + f_{i,k} - e_{i,k-1} - t_{i,r}$;
                end
            else
                $f^{'}_{1,ptb[l]} \longleftarrow f_{1,ptb[l]} + p_{1,ptb[l]}$;
                for $i = 2$ to $m$ do
                    $it'' \longleftarrow it^{''} + f_{i,ptb[l]} - e_{i,ptb[l]} + p_{i,ptb[l]} - t_{i,r} + \max\{0, f^{'}_{i-1,ptb[l]} - f_{i,ptb[l]}\}$;
                    $f^{'}_{i,ptb[l]} \longleftarrow \max\{f^{'}_{i-1,ptb[l]}, f_{i,ptb[l]}\} + p_{i,ptb[l]}$;
                end
            end
            if $it_{bp} > it^{''}$ then
                $bp \longleftarrow ptb[l]$;
                $it_{bp} \longleftarrow it^{''}$;
            end
        end
    end
    $\pi \longleftarrow$ Array obtained by inserting job $r$ in position $bp$ of $\pi$;
end
```

Figure 3: Our Tie-Breaking Method

in Section 1. Nevertheless, we also provide the results using these more advanced initial orderings.

The computational experiments are carried out on an Intel Core i7-930, 2.8GHz, 16GB RAM under Windows 7. This section is divided into two parts depending on which heuristic (NEH or Iterated Greedy Algorithm) the tie breaks are implemented.

4.1 Comparison of Tie-breaking mechanisms for the NEH

The performance of the NEH with the tie-breaking mechanisms by Dong, Kalczynski&Kamburowski and our proposal, as well as with the original tie-breaking mechanism of the NEH (labelled TB_{FS} in the following) are compared using the benchmark set of Taillard (1993) with 120 instances. Note that, although in Kalczynski and Kamburowski (2011) it was established that Dong's tie-breaking mechanism outperformed the two suggested by Kalczynski&Kamburowski for the NEH, we nevertheless include them to test them against the proposal and to make the comparison homogeneous with that of the IG (for which none of the mechanisms' performance was tested).

For each instance, the Relative Percentage Deviation (RPD) is computed with respect to the best known solution according to expression (26), where NEH_j is the solution obtained for the instance by the NEH algorithm using the j tie-breaking mechanism while $Best$ is the best known solution or the lowest known upper bound value for the instance. The Average RPD ($ARPD$) values are obtained by averaging RPD for each instance size or for the whole testbed. The results in Table 1 show that the $ARPD$ found by the original NEH is 3.325 while each other tie break yields better $ARPD$, being the value of 3.034 the best one, obtained by our tie-breaking proposal.

$$RPD = \left(\frac{NEH_j - Best}{Best}\right) \cdot 100 \tag{26}$$

Since we use the same test bed for all tie-breaking mechanisms, being each one a version of the same algorithm, the random variables ($ARPD$) are related and the hypothesis of independence can be rejected. Therefore, a paired samples t-test (shown in Table 2) can be used to compare the results. Note that paired samples t-test is a usual test to establish the statistical significance of the differences in the performance of algorithms for flowshop scheduling problems in Taillard's testbed (see e.g. Hamed Hendizadeh et al., 2008; Tan et al., 2000; Dong et al.,

Table 1: Average relative percentage deviation of NEH implemented with tie-breaking mechanisms

Instance	TB_{FS}	TB_D	TB_{KK1}	TB_{KK2}	TB_{FF}
20 x 5	3.300	2.655	2.638	2.729	2.293
20 x 10	4.601	4.661	4.488	4.312	4.152
20 x 20	3.731	3.443	3.683	3.407	3.305
50 x 5	0.727	0.497	0.586	0.588	0.922
50 x 10	5.073	5.082	5.022	4.875	5.150
50 x 20	6.648	6.091	6.274	6.412	6.207
100 x 5	0.527	0.459	0.354	0.397	0.378
100 x 10	2.215	2.065	1.829	1.771	2.182
100 x 20	5.345	5.235	5.417	5.284	5.021
200 x 10	1.258	1.182	1.179	1.166	0.984
200 x 20	4.408	3.901	4.243	4.232	4.037
500 x 20	2.066	1.779	2.080	2.020	1.776
Average	3.325	3.088	3.149	3.099	3.034

2013). In view of the values of the significance levels, it can be stated that each tie-breaking mechanism is statistically significant with respect to TB_{FS}. However, no statistical significance among the rest of the tie-breaking mechanisms can be found due to the small size of the benchmark, a fact also noted by Kalczynski and Kamburowski (2008) when proposing their tie-breaking mechanisms. Therefore, in line with these authors, an extended test-bed of 400 instances with $n \in \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$, and $m \in \{5, 10, 15, 20\}$, with 10 replications for each combinations of n and m is generated with the processing times uniformly distributed in the interval $[1, 99]$. A paired-samples t-test (shown in Table 3) was performed, indicating that our proposal is statistically significant with respect to the other tie-breaking mechanisms, being 0.01 the maximum p-value found.

Results of the NEH algorithm with the proposed tie-breaking mechanism using different initial orders are shown in Table 4 for the Taillard's testbed. As explained in Section 1, three different initial orders outperforming the original non-ascending order of the sum of their processing times have been proposed in the literature by Kalczynski and Kamburowski (2008) (denoted as $KK1-Init$), by Dong et al. (2008) (denoted as $AvgDev - Init$); and by Kalczynski and Kamburowski

Table 2: Paired samples t-test for NEH using Taillard's benchmark

Algorithm	Mean	SEM	IC - Lower	IC - Upper	t	Significance
TB_{FS} - TB_{FF}	0.291	0.806	0.146	0.437	3.958	0.000
TB_{FS} - TB_{D}	0.237	0.852	0.084	0.391	3.054	0.003
TB_{FS} - TB_{KK1}	0.176	0.736	0.043	0.308	2.626	0.010
TB_{FS} - TB_{KK2}	0.226	0.711	0.098	0.354	3.490	0.001
TB_{D} - TB_{FF}	0.054	0.770	-0.086	0.193	0.764	0.446
TB_{KK1} - TB_{FF}	0.115	0.842	-0.037	0.268	1.502	0.136
TB_{KK2} - TB_{FF}	0.066	0.877	-0.093	0.224	0.819	0.415

Table 3: Paired samples t test for NEH using the extended benchmark

Algorithm	Mean	SEM	IC - Lower	IC - Upper	t	Significance
TB_{FS} - TB_{FF}	0.226	0.496	0.177	0.274	9.117	0.000
TB_{FS} - TB_{D}	0.169	0.486	0.121	0.216	6.943	0.000
TB_{FS} - TB_{KK1}	0.102	0.466	0.056	0.148	4.375	0.000
TB_{FS} - TB_{KK2}	0.126	0.472	0.080	0.173	5.351	0.000
TB_{D} - TB_{FF}	0.057	0.450	0.013	0.101	2.545	0.011
TB_{KK1} - TB_{FF}	0.124	0.490	0.076	0.172	5.057	0.000
TB_{KK2} - TB_{FF}	0.099	0.410	0.059	0.140	4.849	0.000

Table 4: Average relative percentage deviation of NEH implemented with our tie-breaking mechanisms and different initial order

Instance	$KK1 - Init$	$KK2 - Init$	$AvgDev - Init$	$Original$
20 x 5	2.484	2.372	2.559	2.293
20 x 10	4.919	4.453	3.543	4.152
20 x 20	3.265	3.509	3.331	3.305
50 x 5	0.555	0.791	0.749	0.922
50 x 10	4.865	4.861	4.905	5.150
50 x 20	6.139	7.026	5.812	6.207
100 x 5	0.379	0.321	0.412	0.378
100 x 10	1.961	2.057	1.719	2.182
100 x 20	5.284	5.114	5.147	5.021
200 x 10	1.030	0.899	0.987	0.984
200 x 20	3.712	3.895	3.885	4.037
500 x 20	1.726	1.650	1.713	1.776
Average	3.027	3.079	2.897	3.034

(2009) (denoted as $KK2 - Init$). All three were implemented in order to obtain the best initial order for the NEH using our tie-breaking mechanism. The $ARPD$ using the initial order $AvgDev$ was 2.897 being the best initial order for Taillard's testbed, a result in line with those obtained by Kalczynski and Kamburowski (2011).

4.2 Comparison of the tie-breaking mechanisms in the Iterated Greedy

As explained before, the iterated greedy has two parameters (T, d) to be set. Ruiz and Stützle (2007) conducted a full factorial design to determine both parameters, resulting $d = 4$ and $T = 0.4$ as the best combination. Therefore, these values are used in our implementation. Two versions of the iterated greedy are analysed: IG_RS_{LS} as in Ruiz and Stützle (2007) and IG_{RIS} as proposed by Pan et al. (2008). The tie-breaking mechanism analysed in Section 2 was integrated in these Iterated Greedy Algorithms, together with our proposal. In order to compare them, the same test bed as in Ruiz and Stützle (2007) was employed, i.e. Taillard's benchmark using 5 replicates for each instance to increase the power of the analysis.

The termination criterion considered for both versions of the Iterated Greedy is the CPU

Table 5: Average relative percentage deviation of iterated greedy algorithms implemented with tie breaks and $n \cdot (m/2) \cdot 30$ milliseconds as stopping criterion

	IG_RS_{LS}					IG_{RIS}				
Instance	TB_{FS}	TB_{D}	TB_{KK1}	TB_{KK2}	TB_{FF}	TB_{FS}	TB_{D}	TB_{KK1}	TB_{KK2}	TB_{FF}
20 x 5	0.045	0.066	0.076	0.049	0.076	0.037	0.041	0.076	0.053	0.041
20 x 10	0.055	0.052	0.087	0.080	0.104	0.080	0.096	0.099	0.064	0.057
20 x 20	0.092	0.095	0.085	0.066	0.114	0.081	0.093	0.098	0.090	0.092
50 x 5	0.007	0.039	0.021	0.003	0.017	0.006	0.020	0.024	0.007	0.006
50 x 10	0.724	0.754	0.842	0.707	0.566	0.683	0.651	0.787	0.666	0.621
50 x 20	1.199	1.188	1.228	1.191	1.134	1.160	1.066	1.195	1.149	1.173
100 x 5	0.005	0.066	0.030	0.013	0.014	0.005	0.067	0.018	0.013	0.013
100 x 10	0.274	0.383	0.415	0.215	0.226	0.258	0.301	0.336	0.202	0.219
100 x 20	1.624	1.446	1.789	1.624	1.346	1.547	1.365	1.770	1.542	1.387
200 x 10	0.317	0.477	0.284	0.140	0.155	0.267	0.361	0.263	0.161	0.148
200 x 20	1.656	1.401	1.925	1.466	1.239	1.549	1.287	1.898	1.478	1.248
500 x 20	0.767	0.724	1.033	0.668	0.542	0.728	0.621	0.987	0.626	0.530
Average	0.564	0.558	0.651	0.518	0.461	0.534	0.497	0.629	0.504	0.461

Table 6: Average relative percentage deviation of iterated greedy algorithms implemented with tie breaks and $n \cdot (m/2) \cdot 60$ milliseconds as stopping criterion

	IG_RS_{LS}					IG_{RIS}				
Instance	TB_{FS}	TB_{D}	TB_{KK1}	TB_{KK2}	TB_{FF}	TB_{FS}	TB_{D}	TB_{KK1}	TB_{KK2}	TB_{FF}
20 x 5	0.024	0.039	0.041	0.041	0.032	0.032	0.041	0.041	0.032	0.041
20 x 10	0.043	0.031	0.049	0.057	0.059	0.038	0.042	0.064	0.032	0.046
20 x 20	0.067	0.042	0.047	0.049	0.057	0.052	0.070	0.066	0.060	0.071
50 x 5	0.004	0.009	0.016	0.003	0.007	0.000	0.026	0.010	0.004	0.001
50 x 10	0.529	0.615	0.692	0.595	0.441	0.549	0.524	0.626	0.584	0.478
50 x 20	1.044	1.005	1.047	0.978	1.048	1.011	0.940	1.060	1.008	1.012
100 x 5	0.008	0.056	0.011	0.006	0.006	0.006	0.028	0.006	0.006	0.009
100 x 10	0.218	0.228	0.310	0.170	0.149	0.173	0.214	0.223	0.184	0.111
100 x 20	1.423	1.317	1.643	1.449	1.118	1.394	1.145	1.589	1.402	1.245
200 x 10	0.250	0.397	0.217	0.092	0.093	0.174	0.271	0.188	0.113	0.101
200 x 20	1.407	1.217	1.819	1.313	1.049	1.407	1.125	1.754	1.401	1.036
500 x 20	0.720	0.627	0.992	0.602	0.453	0.650	0.519	0.958	0.573	0.473
Average	0.478	0.465	0.573	0.446	0.376	0.457	0.412	0.549	0.450	0.385

Table 7: Average relative percentage deviation of iterated greedy algorithms implemented with tie breaks and $n \cdot (m/2) \cdot 90$ milliseconds as stopping criterion

	IG_RS_{LS}					IG_{RIS}				
Instance	TB_{FS}	TB_D	TB_{KK1}	TB_{KK2}	TB_{FF}	TB_{FS}	TB_D	TB_{KK1}	TB_{KK2}	TB_{FF}
20 x 5	0.015	0.024	0.041	0.041	0.041	0.041	0.041	0.039	0.041	0.041
20 x 10	0.035	0.022	0.072	0.026	0.024	0.025	0.040	0.048	0.057	0.024
20 x 20	0.038	0.030	0.049	0.048	0.035	0.028	0.054	0.028	0.041	0.051
50 x 5	0.000	0.007	0.007	0.002	0.004	0.003	0.007	0.011	0.001	0.003
50 x 10	0.517	0.567	0.649	0.532	0.438	0.493	0.526	0.583	0.555	0.453
50 x 20	0.918	0.953	0.978	0.874	0.858	0.902	0.837	0.912	0.925	0.935
100 x 5	0.006	0.053	0.008	0.008	0.001	0.006	0.037	0.008	0.004	0.003
100 x 10	0.213	0.253	0.293	0.183	0.169	0.199	0.187	0.239	0.139	0.155
100 x 20	1.261	1.193	1.485	1.388	1.096	1.274	1.048	1.516	1.350	1.106
200 x 10	0.169	0.388	0.180	0.080	0.078	0.155	0.241	0.171	0.069	0.061
200 x 20	1.337	1.184	1.706	1.276	1.026	1.278	1.049	1.704	1.344	0.987
500 x 20	0.674	0.611	0.933	0.558	0.428	0.605	0.488	0.920	0.543	0.412
Average	0.432	0.441	0.533	0.418	0.350	0.417	0.380	0.515	0.422	0.353

time. In line with most papers, this time t depends on the amount of jobs and machines, i.e. $t = n \cdot (m/2) \cdot 30$, $t = n \cdot (m/2) \cdot 60$ and $t = n \cdot (m/2) \cdot 90$ milliseconds (see e.g. Ruiz and Stützle, 2007, or Tzeng and Chen, 2012). $ARPD$ results for each version of the iterated greedy algorithm and for each tie-breaking mechanism are shown in Tables 5, 6 and 7 for each stopping time, respectively. The results show that the $ARPD$ for IG_RS_{LS} with our tie-breaking mechanism is the best for every stopping time, being the average results 0.461, 0.376 and 0.350 respectively. Kalczynski & Kamburowski's tie-breaking mechanism II also yields good $ARPD$ results: 0.518, 0.446 and 0.418 respectively. Both mechanisms performs better than the original iterated greedy algorithm. Nevertheless, it is to note that Dong's tie-breaking mechanism and Kalczynski & Kamburowski's tie-breaking mechanism I give worse results when included in IG_RS_{LS}.

Furthermore, a paired-samples t- test was carried out in order to analyse the different mechanisms (see Table 8). Our tie-breaking mechanism was found to be statistically significant with respect to every other tie-breaking mechanism for every value of t considered, being 0.017 the highest p-value. Regarding the rest of the tie-breaking mechanisms, Kalczynski & Kamburowski's tie-breaking mechanism II was found to be statistically significant with respect to the original IG_RS_{LS} for $t = n \cdot (m/2) \cdot 30$ and $t = n \cdot (m/2) \cdot 60$ but not for $t = n \cdot (m/2) \cdot 90$, being 0.118 the

p-value for this case. On the other hand, Kalczynski & Kamburowski's tie-breaking mechanism I was found statistically worse, with p-value of 0.000. Finally, no statistically significance was found for any t between Dong's tie-breaking mechanism and the original IG_RS_{LS}. Regarding IG_{RIS}, very similar results were found (results are shown in Tables 5, 6 and 7 for the different values of t). On the one hand, the proposed tie-breaking mechanism yields again the best $ARPD$, being 0.461 for $t = n \cdot (m/2) \cdot 30$ milliseconds, 0.385 for $t = n \cdot (m/2) \cdot 60$ milliseconds, and 0.353 $t = n \cdot (m/2) \cdot 90$ milliseconds. On the other hand, Kalczynski & Kamburowski's tie-breaking mechanism I is again the one with the worst results.

It is worth to highlight that the proposed tie-breaking mechanism performs better than existing mechanisms when embedded in the iterated greedy than when integrated in the NEH. Note that the fact that a tie-breaking mechanism performs efficiently for the NEH does not imply the same for the iterated greedy. This is due to the fact that, in the NEH, the insertion is performed in all steps (i.e. from a one-job sequence until the n jobs have been scheduled), while the construction phase of the iterated greedy is performed only for the last d steps (beginning with a sequence of $N - d$ jobs). Therefore, a tie-breaking mechanism should have a good performance in the last steps of the insertion phase in order to be efficient when embedded in the iterated greedy algorithm.

5 Conclusions

In this paper, we have presented a new tie-breaking mechanism based on an estimation of the idle times of the different subsequences in order to pick the one with the lowest value of the estimation. This tie-breaking mechanism can be incorporated into the most efficient approximate procedures for the flowshop scheduling problem with makespan objective, resulting in statistically significant better results than existing tie-breaking mechanisms.

The rationale of our proposed tie-breaking mechanism is relatively simple, as it seems intuitive that lower values of the total idle time would mean less delays in the processing of the jobs, which would eventually lead to a better utilization of the machines and to a shortest makespan value once all jobs have been positioned. The challenge is to calculate these idle times in an efficient manner,

Table 8: Paired samples t test for IG_RS_{LS} using the benchmark of Taillard

CPU time	Algorithm	Mean	SEM	IC - Lower	IC - Upper	t	Significance
$n \cdot (m/2) \cdot 30$	TB_{FS} - TB_{FF}	0.103	0.277	0.080	0.125	9.078	0.000
	TB_{D} - TB_{FF}	0.097	0.283	0.074	0.119	8.361	0.000
	TB_{KK1} - TB_{FF}	0.190	0.335	0.163	0.217	13.889	0.000
	TB_{KK2}- TB_{FF}	0.057	0.235	0.038	0.076	5.961	0.000
	TB_{FS} - TB_{D}	0.006	0.286	-0.017	0.029	0.522	0.602
	TB_{FS} - TB_{KK1}	-0.087	0.259	-0.108	-0.067	-8.266	0.000
	TB_{FS} - TB_{KK2}	0.045	0.222	0.028	0.063	5.002	0.000
$n \cdot (m/2) \cdot 60$	TB_{FS} - TB_{FF}	0.102	0.266	0.081	0.124	9.402	0.000
	TB_{D} - TB_{FF}	0.089	0.272	0.067	0.111	8.034	0.000
	TB_{KK1} - TB_{FF}	0.197	0.361	0.169	0.226	13.397	0.000
	TB_{KK2}- TB_{FF}	0.070	0.250	0.050	0.090	6.880	0.000
	TB_{FS} - TB_{D}	0.013	0.263	-0.008	0.034	1.205	0.229
	TB_{FS} - TB_{KK1}	-0.095	0.267	-0.117	-0.074	-8.746	0.000
	TB_{FS} - TB_{KK2}	0.032	0.251	0.012	0.052	3.132	0.002
$n \cdot (m/2) \cdot 90$	TB_{FS} - TB_{FF}	0.082	0.243	0.063	0.102	8.280	0.000
	TB_{D} - TB_{FF}	0.091	0.254	0.070	0.111	8.753	0.000
	TB_{KK1} - TB_{FF}	0.184	0.311	0.159	0.209	14.455	0.000
	TB_{KK2}- TB_{FF}	0.068	0.221	0.050	0.086	7.554	0.000
	TB_{FS} - TB_{D}	-0.009	0.252	-0.029	0.012	-0.846	0.398
	TB_{FS} - TB_{KK1}	-0.102	0.249	-0.121	-0.082	-10.006	0.000
	TB_{FS} - TB_{KK2}	0.014	0.219	-0.004	0.031	1.564	0.118

particularly taking into account that Taillard's acceleration provides a very fast mechanism to evaluate the subsequences which is at the core of the excellent performance of NEH and Iterated Greedy. Our proposal is to use an ersatz of the idle times that can be calculated in parallel to the evaluation of the makespan of the subsequences and thus not adding computational complexity to the algorithms.

Note that Dong's tie-breaking mechanism can be also analysed under the light of idle time minimisation. Dong's tie-breaking mechanism seeks a balanced usage of the machines at the time slot of the insertion job, so idle time is 'locally' reduced. A drawback of this mechanism is that it may generates fragmented time slots, which cannot be easily used by other jobs at a later stage. This may explain the different relative performance of this tie-breaking mechanism, depending on whether it is embedded in the NEH (a single pass construction), or in IG (iterative).

The contribution of the paper can be summarised as follows: The main tie-breaking mechanisms proposed for the NEH in the literature are also embedded in the Iterated Greedy Algorithm for the first time, and their impact is analysed. Additionally, a new tie-breaking mechanism is proposed, which is shown to be statistically better than other tie-breaking mechanisms both for the NEH and for the Iterated Greedy. Furthermore, it has been shown that $IG_{RIS}(TB_{FF})$ outperforms the original IG_{RIS}. Given the fact that IG_{RIS} is a state-of-the-art heuristic for the problem under consideration (see e.g. Pan et al., 2008), it turns out that $IG_{RIS}(TB_{FF})$ becomes the best heuristic for the permutation flowshop problem with makespan objective for different allowed CPU times.

Acknowledgements

The authors are sincerely grateful to the anonymous referees, who provide very valuable comments as well as a hint to speed up the tie-breaking mechanism proposed in an earlier version of the manuscript. This research has been funded by the Spanish Ministry of Science and Innovation, under the project “SCORE” with reference DPI2010-15573/DPI.

References

Dong, X., Chen, P., Huang, H., and Nowak, M. (2013). A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time. *Computers and Operations Research*, 40(2):627–632.

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.

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. (2003). An efficient constructive heuristic for flowtime minimisation in permutation flowshops. *OMEGA, The International Journal of Management Science*, 31:311–317.

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.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, 5:287–326.

Hamed Hendizadeh, S., Faramarzi, H., Mansouri, S., Gupta, J., and Y ElMekkawy, T. A. (2008). Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. *International Journal of Production Economics*, 111(2):593–605.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1:61–68.

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.

Krajewski, L., King, B., Ritzman, L., and Wong, D. (1987). Kanban, MRP, and shaping the manufacturing environment. *Management Science*, 33:39–57.

Nagano, M. and Moccellin, J. (2002). A high quality solution constructive heuristic for flow shop sequencing. *Journal of the Operational Research Society*, 53(12):1374–1379.

Nawaz, M., Enscore, Jr, E. 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., Tasgetiren, M., and Liang, Y.-C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers and Industrial Engineering*, 55(4):795–816.

Rad, S. F., Ruiz, R., and Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *OMEGA, The International Journal of Management Science*, 37(2):331–345.

Rajendran, C. (1993). Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *International Journal of Production Economics*, 29(1):65–73.

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

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.

Rinnooy Kan, A. H. G. (1976). *Machine Scheduling Problems: Classification, Complexity and Computations*. Martinus Nijhoff, The Hague.

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

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

Storer, R., Wu, S., and Vaccari, R. (1992). New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38:1495–1509.

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

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

Tan, K.-C., Narasimhan, R., Rubin, P., and Ragatz, G. (2000). A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times. *OMEGA, The International Journal of Management Science*, 28(3):313–326.

Tzeng, Y.-R. and Chen, C.-L. (2012). A hybrid eda with acs for solving permutation flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, 60(9-12):1139–1147.

Vakharia, A. and Wemmerlov, U. (1990). Designing a cellular manufacturing system: a materials flow approach based on operation sequences. *IIE Transactions*, 22:84–97.

Woo, H.-S. and Yim, D.-S. (1998). A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers & Operations Research*, 25(3):175–182.