Computers & Operations Research 37 (2010) 2062–2070

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Comparing three-step heuristics for the permutation flow shop problem

Imma Ribas [a], Ramon Companys [a,*], Xavier Tort-Martorell [b]

[a] Laboratori d'Organització Industrial, DOE – ETSEIB - Universitat Politècnica de Catalunya, Avda. Diagonal, 647, 7th Floor, 08028 Barcelona, Spain
[b] Departament de Estadística E investigación Operativa- ETSEIB - Universitat Politècnica de Catalunya, Avda. Diagonal, 647, 6th Floor, 08028 Barcelona, Spain

ARTICLE INFO

Available online 3 March 2010

Keywords:
Permutation flow shop
Scheduling
Heuristics

ABSTRACT

In this paper a three steps heuristic for the permutation flow shop problem is proposed. The objective is to minimize the maximum time for completing the jobs, or the makespan. The first two steps are inspired by the NEH heuristic, to which a new tie breaking strategy has been incorporated in the insertion phase. Furthermore, the reversibility property of the problem dealt with is taken as a tool for improving the obtained solution. The third step consists of an iterated local search procedure with an embedded local search which is a variant of the non exhaustive descent algorithm. The statistical analysis of the results shows the effectiveness of the proposed procedures.

1. Introduction

The permutation flow shop scheduling problem (PFSP) consists of finding a sequence of n jobs which must be processed in m machines, always in the same order, with the objective of minimizing one or various efficiency measures. The most commonly studied efficiency measure, which is also applied here, is the minimization of the maximum completion time. This is the makespan. This problem is expressed following the notation proposed by Graham et al. [1], Fm|prmu|C_{max}. Hejazi and Saghafian [2] presented a complete revision of the flow shop scheduling problems with makespan criterion. Since the publication of Johnson's paper [3], one of the most investigated problems has been the PFSP. Gupta and Stafford [4] give a summarized version of the evolution of this research over the last five decades. Garey and Johnson [5] showed that for $m \geq 3$ the problem is NP-hard. The difficulty of exactly resolving the problem has led to the development and use of heuristic procedures which give high quality solutions to large size problems in short times.

Heuristic procedures can be classified either as constructive or improvement heuristics. The former prepare an initial solution starting from zero, step by step. The latter group starts out from an initial solution which they aim to improve by exploring its neighbourhood. Ruiz and Maroto [6] evaluate and compare both procedure types for the permutation flow shop problem with makespan criterion.

The first constructive heuristic was the one proposed by Giglio and Wagner [7] for $m=3$ based on Johnson's procedure. Other heuristics, such as those proposed by Palmer [8] and Gupta [9], assign a weight or index to each job in order to determine the scheduled sequence for the jobs in accordance with certain rules. Campbell et al. [10] developed the CDS procedure as an extension of the Johnson's algorithm for $m>2$. The procedures proposed by Companys [11] and Dannenbring [12], known respectively as Trapeziums (TR) and Rapid Access (RA), can be interpreted as variants of the CDS procedure. Nawaz et al. [13] proposed a NEH heuristic which is considered one of the best heuristics for the permutation flow shop problem. The NEH procedure can be divided into two steps. The first step is the generation of an initial order for the jobs applying the largest processing time (LPT) rule. The second step is the iterative insertion of jobs in a partial sequence in accordance with the initial order obtained in the first step. Given its efficiency this procedure has been widely studied and different variants of it have been proposed in the literature. The first type of variant is focused on the priority order. Nagano and Moccellin [14] proposed a different initial ordering. Framiñan et al. [15] examine 176 rules used to obtain the initial sequence and they show that the ordering proposed initially in the NEH heuristic is the one which obtains the best results when the objective is to minimize the makespan. However, Nagano's and Moccelin's procedure and Kalczynski and Kamburowski's rule [16] are not included among the examined rules. Rad et al. [17] propose procedures based on the NEH heuristic in which the initial order is based on the operations' processing times.

The second type of variants deals with the tie-breaking criterion to be used when two different positions give the same makespan in the insertion phase. Recently in [16] and in [18] new strategies for tie breaking to be used in step 2 of the NEH heuristic are proposed. Finally, we can classify Taillard's [19] procedure to accelerate the NEH heuristic as a third type of variant.

Two of the improvement heuristics, based on the *hill climbing* type of procedure, were proposed by Dannenbring [12] in 1977,

* Corresponding author. Tel.: +34 93 401 65 71; fax: +34 93 401 60 54.
E-mail address: ramon.companys@upc.edu (R. Companys).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.02.006

who observed that by changing two adjacent jobs in a sequence obtained from the RA heuristic it was possible to achieve better solutions and he proposed two improvement procedures, rapid access with close order search (RACS) and rapid access with extensive search (RAES). The RACS procedure generates $n-1$ neighbouring solutions, for a given solution, interchanging two jobs, one after the other in a sequence, and retaining the best solution obtained. In contrast, RAES explores a potentially much wider neighbourhood as it proposes applying RACS to the initial sequence, and if the retained sequence is different it continues with the same procedure until the $C_{\max}$ value does not diminish any more. Suliman [20] proposes improving the sequence generated by the CDS heuristic by interchanging jobs. The main problem of this kind of local search procedure is how to escape from a local optima found or how to escape from the extensive plateaus existing in the domain of the solutions defined by the adopted neighbourhood. One way is to randomly explore the neighbourhood of the incumbent solution and to deal with ties (apply tie-breaking), i.e. solutions with the same makespan. Another way is to try to extend the solution space using a diversification technique. One simple technique, which has turned out to be very effective, is to perturb the best sequence found and to apply the local search on this new solution, as is done in the iterative local search algorithm [21]. Stützle [22] proposes an iterated local search (ILS) for the PFSP with makespan criterion with a local search based on the insertion neighbourhood. Ruiz and Stützle [23] propose a very effective iterated greedy algorithm (IGA), for this problem, which is quite similar to ILS.

In this paper we propose a new heuristic procedure made up of three steps with the objective of minimizing the maximum completion time, or the makespan. The first two steps of the procedure are variants of the NEH heuristic. Step 1 consists of a sequencing order and step 2 consists of the NEH's insertion procedure to which a new tie breaking strategy has been incorporated. Furthermore, we propose taking advantage of the reversibility property of the problem dealt with as a means by which we can improve the solution. This new tie breaking outperforms those proposed by Kalczynski and Kamburowski [16] and by Dong et al. [18] when the reversibility property is used, as can be seen in Section 5. Since the combination of step one and two builds a variant of the NEH heuristic, with our improvements proposed, we are contributing to increasing the performance of this procedure which has been considered one of the most efficient so far. In step 3, an iterated local search algorithm was implemented with an embedded local search based on swap-moves, similar to the RAES, which incorporates two tools to escape from the extensive plateaus existing in the domain of the solutions defined by the neighbourhood adopted: the first one, which we call revolver, allows the neighbourhood to be explored randomly, and the second one is a procedure to deal with ties between neighbouring solutions, i.e. solutions with the same makespan value. The computational experiment shows the efficiency of the procedure proposed.

After this introduction, we describe the problem dealt with in Section 2. Section 3 details the proposed procedure. In Section 4 the variants of the proposed procedure are set out. Section 5 shows the computational results on 120 Taillard's instances and 2500 randomly generated instances. Finally, Section 6 includes the main conclusions.

2. Stating the problem

At instant zero there are n jobs which must be processed, in the same order, in m machines. Each job goes from machine 1 to machine m. The processing time for each operation is $p_{j,i}$, being $p_{j,i}>0$, where $j\in\{1,2,\ldots,m\}$ indicates the machine and $i\in\{1,2,\ldots,n\}$ the job. The set up times are considered to be independent of the sequence and are included in the processing times. The objective function considered is the minimization of the makespan, which is equivalent to maximizing the use of the machines.

Given a permutation, $\mathbf{P}$, of the n jobs, $[k]$ indicates the job in position k in the sequence. Given a feasible schedule associated to a permutation, $e_{j,k}$ is defined as the initial instant in which the job that occupies position k starts to be processed on the machine j and $f_{j,k}$ as the instant when the job that occupies position k in machine j is finalized. The Fm|prmu|$C_{\max}$ problem can be expressed by the following formulation:

$$e_{j,k}+p_{j,[k]}\le f_{j,k} \quad j=1,2,\ldots,m \quad k=1,2,\ldots,n \tag{1}$$

$$e_{j,k}\ge f_{j,k-1} \quad j=1,2,\ldots,m \quad k=1,2,\ldots,n \tag{2}$$

$$e_{j,k}\ge f_{j-1,k} \quad j=1,2,\ldots,m \quad k=1,2,\ldots,n \tag{3}$$

$$C_{\max}=f_{m,n} \tag{4}$$

being, $f_{j,0}=0\ \forall j$, $f_{0,k}=0\ \forall k$, the initial conditions.

The schedule is semi-active if the Eq. (1) is written as $e_{j,k}+p_{j,[k]}=f_{j,k}$ and the Eqs. (2) and (3) are summarized as $e_{jk}=\max\{f_{j,k-1};\,f_{j-1,k}\}$.

2.1. Reversibility of the PFSP

Given an instance I, which we will call the direct instance, with processing times $p_{j,i}$, another instance I' can be determined, where the processing times $p_{j,i}'$ are calculated according to (5):

$$p'_{j,i}=p_{m-j+1,i} \quad j=1,2,\ldots,m \quad i=1,2,\ldots,n \tag{5}$$

For a permutation π, the value of $C_{\max}$ in I is the same as that obtained in I' for the reverse permutation π'. Therefore, the $C_{\max}$ minimum is the same for the permutation I and I', and the permutations associated are the inverse one of the other. As a consequence, it is indifferent whether the instance I or the I' is solved.

Some authors, such as Brown and Lomnicki [24] and McMahon and Burton [25], deduced from the results obtained in their tests that the inverse instance is sometimes solved more efficiently than the direct instance in Branch and Bound (B&B) procedures. In addition, we have observed that in some cases applying the B&B procedure to the direct instance is more efficient for generating solutions, while the application to the inverse instance is more efficient for the bounds.

3. Heuristic procedure

The procedure designed is divided into three steps (see Fig. 1). The two first steps, based on the NEH heuristic, construct an initial solution which is after improved through an iterated local search procedure, in step 3. We have called this "Soft Simulated Annealing" (SSA).

In the following section each of the three steps are described.

3.1. The first step

We have considered 4 initial sequencing rules for the jobs. Each rule defines a variant of the proposed procedure. The rules considered are: the secuence proposed by [16] (KK), the NEH proposal [13] (LPT), the Nagano and Moccellin procedure [14] (NM) and the ordering obtained randomly (RA).

In the section below we describe the ordering procedure (step 1) in its 4 variants:

- KK: For each job i calculate $a_i=\sum_{j=1}^{m}((m-1)\cdot(m-2)/2+m-j)\cdot p_{j,i}$ and $b_i=\sum_{j=1}^{m}((m-1)\cdot(m-2)/2+j-1)\cdot p_{j,i}$. Sequence the jobs according to the non-increasing order of $c_i=\min(a_i;b_i)$.
- LPT: Order the n jobs in non-increasing order of $P_i=\sum_{j=1}^{m}p_{ji}$.
- NM: For each job i calculate $\overline{P}_i=P_i-\max_h\{BT_{hi}\}$, being BT_{hi} the lower bound for the waiting time for job i, from the completion time of its operations in each machine to the beginning of the operation in the following machine, when job h immediately proceeds job i (and only jobs h and i are being considered). Order the n jobs in non-increasing order of $\overline{P}_i$.
- RA: An initial sequence is generated randomly.

3.2. The second step

In step 2 we have implemented a new strategy which consists of two tie-breaking methods for when two different positions give the same makespan. The first method aims at minimizing the total idle time of machines (TIT), (see Fig. 2), and the second method is the one proposed in Kalczynski and Kamburowski [16] (KK1).

- Method TIT: The total idle time $\sum_{j=1}^{m}IT(j)$ is calculated for each possible inserting position, where $IT(j)=f_{j,n}-e_{j,1}-\sum_{j=1}^{n}p_{j,i}$.
  If there is a tie between two positions, the job is inserted in the position which has less total idle time associated.
- Method KK1: Let i be the job to be inserted, if there is a tie between two positions the position chosen is the one nearest to the first position, if $a_i\le b_i$, and the nearest to the last one if $a_i>b_i$. Where a_i and b_i are calculated as is indicated in the KK rule of step 1.

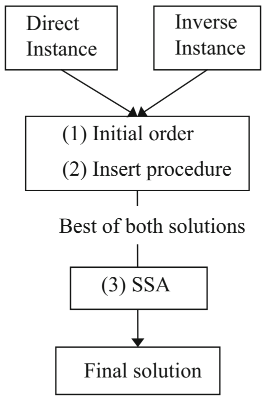

Fig. 1. Three-step procedure.

As a consequence, step 2 is as follows:

- Step 2: in accordance with the order established in step 1, take the first two jobs and schedule them in such a way that they minimize the partial makespan, considering an instance with only two jobs. Then, for $k=3$ up to n, insert the k-th job into one of the possible k positions of the partial sequence. The objective is to minimize the $C_{\max}$ of the $Fm|prmu|C_{\max}$ problem with k jobs. To break ties, choose the sequence with the lowest idle time for the machines (method TIT). If there is still a tie, use the procedure defined in [16] for NEHKK1 (method KK1). If the tie persists, the first position considered is chosen.

The complexity of this version of the NEH heuristic, with the new tie breaking strategy, is O (m^2n^2).

3.3. The third step

The improvement phase consists of an iterated local search on the initial solution with a local search, which is a variant of the non exhaustive descent algorithm (NEDA). The name given to this procedure is soft simulated annealing (SSA).

NEDA tries to improve the solution (π) by swapping any two positions in the sequence. Potentially this procedure can generate $n\cdot(n-1)/2$ neighbours. If during the process a new permutation improves the value of the objective function $(C_{\max}(\pi))$, it becomes the new current solution and the process continues until all the positions have been permuted and improvement is no longer taking place. In this procedure the neighbourhood is always explored in the same order. However, the local search embedded in the SSA uses an auxiliary array, which allows the neighbourhood to be explored randomly. The revolver is a pointer array whose components are initialized with the different positions that a job can have in the sequence. Next, the components are randomly mixed (Procedure Shuffle) and used to codify the searching positions in the solution's neighbourhood. Given two pointers to positions i,j in the job sequence, their equivalent i_{rev} and j_{rev} are searched for in the revolver array, rev, being $i_{rev}=rev(i)$ and $j_{rev}=rev(j)$. These new positions are used when the non-exhaustive descent search is applied. During the procedure, when all the neighbourhood of the current solution has been explored without improving the solution, the process restarts again. It accepts solutions with the same makespan (ties) with a certain probability, α. The improvement phase finishes when the number of ties reaches a predefined number β or there is no change in the current solution. Then, the current solution π goes to the deconstruction and construction phase before returning to the beginning of the local search. In the deconstruction phase d jobs are chosen, randomly, and are extracted from π and stored in π'' one after another. Then, in the construction phase, these jobs are returned to the solution, one at a time, using the insertion procedure of step 2, as is done in [23]. The new solution π^* is the

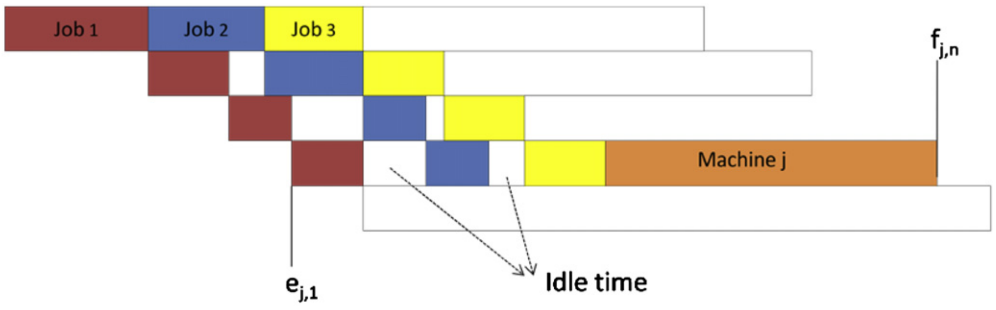

Fig. 2. Components of machine idle time in machine j.

incumbent solution for the local search. Its outline is given in Figs. 3 and 4. The process continues until the stopping criterion is reached. In our implementation the algorithm finishes when the time reached is $n \cdot m \cdot 50$ ms.

```
Procedure SSA
  $C_{best}$ :=$C_{solini}$; $\pi_{best}$ :=$\pi_{solini}$; $\pi^*$ := $\pi_{solini}$;
  while time < timelimit
    $\pi$ :=Local Search ($\pi^*$);
    $\pi''$ := Ø;
    for i :=1 to d do
      remove one job of $\pi$ randomly and insert it in position i of $\pi''$;
    endfor
    for i :=1 to d do
      insert jobs of $\pi''$ in $\pi$ according to the insertion procedure used in step 2;
    endfor
    $\pi^*$ := $\pi$; $C^*$ :=$C_{max}(\pi)$;
    if $C^* < C_{best}$ then
      $C_{best}$ :=$C^*$; $\pi_{best}$ := $\pi^*$;
    endif
  endwhile
  return $\pi_{best}$
end
Figure 3. SSA Algorithm.
```

Fig. 3. SSA algorithm.

```
procedure Local search ($\pi$)
    ties := 0; a :=1; C :=$C_{max}$($\pi$) ;
      for i:=1 to n do
        rev(i) :=i;
      endfor
        rev :=Shuffle (rev);
        b :=0; $Ind_{rev}$ :=0;
      while a :=1 and ties <$\beta$
          a :=0;
          for i := 1 to n-1 do
            $i_{rev}$ :=rev(i);
            for j :=i+1 to n do
              $i_{rev}$:=rev(j);
              SWAP positions ($i_{rev}$ ,$j_{rev}$) in $\pi$ to obtain $\pi'$;
              evaluate $\pi'$;
             if $C_{max}(\pi')$ < C then
              $\pi$ := $\pi'$; C := C($\pi'$); ties := 0; a :=1; b :=0; $Ind_{rev}$ :=0;
             else if C($\pi'$) :=C and b :=1 and random<$\alpha$ then
              $\pi$ := $\pi'$; a :=1; ties := ties + 1;
             endif
            endfor
          endfor
        If $Ind_{rev}$ :=1 then
          rev :=Shuffle (rev);
        endif
        If a :=0 and b :=0 then
          a :=1; b :=1; $Ind_{rev}$ :=1;
        endif
      endwhile
    if $C_{max}(\pi)$ < $C_{best}$ then
      $C_{best}$ := $C_{max}(\pi)$; $\pi_{best}$ := $\pi$;
    endif
    return $\pi$
end

procedure Shuffle (rev)
  for i :=1 to n do
    j := 1+Int (n · random);
    SWAP (rev (i), rev (j));
  endfor
  return rev
end
```

Fig. 4. Local search procedure.

4. Variants on the procedure proposed

Each one of the 4 procedures considered to create the initial sequence (step 1) defines a variant of the algorithm proposed. In each procedure, at the end of each one of the three steps, there is a sequence which can be evaluated in terms of makespan. In order to distinguish between these sequences and evaluations we have defined the following notation (Table 1): As a general rule, in step 1 we have used the name of the priority rule and in step 2 we have added to this name 'ER' to indicate the NEH's insertion procedure with our tie-breaking strategy. The only exception to this codification is in the NEH heuristic where we have only added 'R' to the name to indicate the addition of our tie-breaking strategy. It has to be noted that the solution obtained after step 2 is the best between the solution obtained when the procedure is applied on the direct instance and on the inverse instance. Finally, in step 3, we have added '+' to the name given in step 2.

Comparing the values associated to the sequences, obtained after steps 2 and 3, makes it possible to appreciate the improvement obtained thanks to applying the SSA procedure.

5. Experimental evaluation

In this section the performances of both the proposed tools to improve solutions obtained by NEH-based heuristic and the proposed SSA algorithm are evaluated. Statistical analysis is also carried out in order to adjust the SSA algorithm parameters.

The algorithms have been coded in the same programming language (Quick basic) and are tested on the same computer; an Intel Core 2 Duo E8400 CPU, 3 GHz and 2 GB of RAM memory. To analyze the experimental results obtained we have used the index I_{hs} calculated as in (6):

$$I_{hs} = \frac{Heur_{hs} - Best_s}{Best_s} \times 100 \qquad (6)$$

where $Heur_{hs}$ is the average of the makespan values obtained by the heuristic h and $Best_s$ is the lowest makespan known for the instance s.

5.1. Experimental parameter setting of the SSA algorithm

The proposed SSA algorithm has only 3 parameters that we have tuned using the design of experiments (DOE) approach. We have used the three parameters as factors at the following levels:

- α, 2 levels: 0.5, 0.75;
- β, 2 levels: $n \cdot (n-1)/2$, $n \cdot (n-1)$;
- d, 2 levels: 4, 6.

This yields eight different combinations to be tested. For the experiments we have generated 28 sets of instances, five instances per set for every combination of m and n where n={20, 50, 80, 110, 140, 170, 200} and m={5, 10, 15, 20}. The instances have been generated according to Taillard's [26] indications. Therefore, 140 instances have been processed for each of the 8 combinations of parameters with a computation time limit fixed at $n \cdot m \cdot 50$ ms. Due to the randomness of the improvement procedure we have done 5 replicates for each instance and combination of parameters.

The results have been analyzed by means of a multi-factor analysis of variance (ANOVA) where n and m are non-controllable factors. We have checked the three main hypothesis of ANOVA: normality, homoscedasticy and independence of residuals. The ANOVA results can be seen in Table 1 where it can be observed that the only factors which are significant are n and m, the size of

Table 1
Implemented algorithms.

Step 1	KK	LPT	NM	RA
Step 2	KKER	NEHR	NMER	RAER
Step 3	KKER+	NEHR+	NMER+	RAER+

Table 2
ANOVA table for the experiment on tuning the parameters of SSA.

Source	DF	SS	MS	F	P
Main effects					
n	6	0.000417868	0.000069645	15.74	0.000
m	3	0.001477890	0.000492630	111.33	0.000
α	1	0.000008049	0.000008049	1.82	0.178
β	1	0.000010056	0.000010056	2.27	0.132
d	1	0.000009788	0.000009788	2.21	0.137
Interactions					
n*m	18	0.000397221	0.000022068	4.99	0.000
n*α	6	0.000033291	0.000005549	1.25	0.276
n*β	6	0.000026449	0.000004408	1.00	0.426
n*d	6	0.000014854	0.000002476	0.56	0.763
m*α	3	0.000013737	0.000004579	1.03	0.376
m*β	3	0.000007268	0.000002423	0.55	0.650
m*d	3	0.000007393	.000002464	0.56	0.644
α*β	1	0.000000000	0.000000000	0.00	0.997
α*d	1	0.000001581	0.000001581	0.36	0.550
β*d	1	0.000000004	0.000000004	0.00	0.977
Error	1059	0.004686137	0.000004425		
Total	1119	0.007111585			

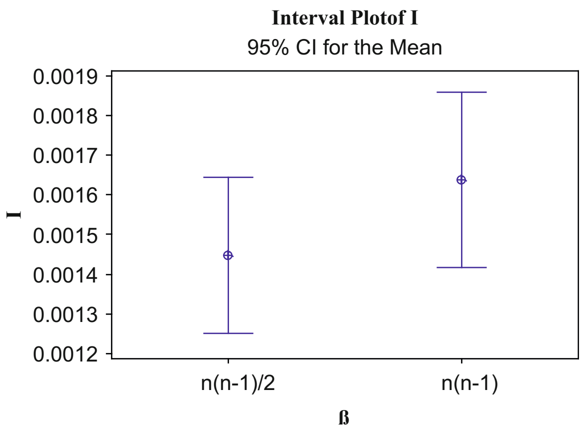

Fig. 5. Mean plot for index I_{hs} and the corresponding confidence interval for the two levels of β.

the instance. The p-value of the remaining factors is bigger than 0.05 which means that neither these factors nor their interactions are significant (Table 2).

A 95% confidence interval plot for parameters β, α and d can be seen in Figs. 5–7 respectively. As was commented previously, there is no statistically significant difference between the levels of these factors. However, since they are close to being significant at the 90% level for these factors we have set the levels giving the smallest index. They are: α=0.75, $\beta=n\cdot(n-1)/2$ and d=4, as can be seen in Figs. 5–7.

5.2. Experimental results

In this stage we continue with the analysis of the behaviour of the proposed procedure. Two tests were carried out. For the first test we used Taillard's instances [26] which combine 20, 50, 100,

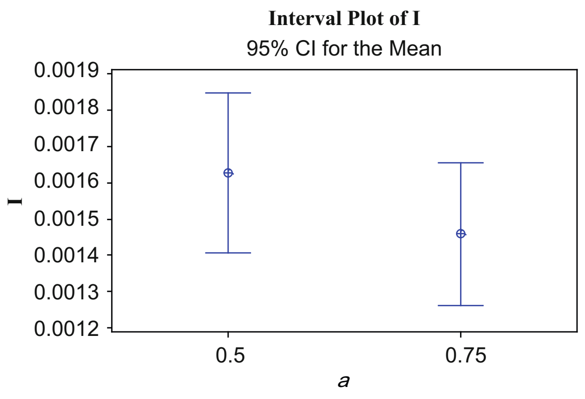

Fig. 6. Mean plot for index I_{hs} and the corresponding confidence interval for the two levels of α.

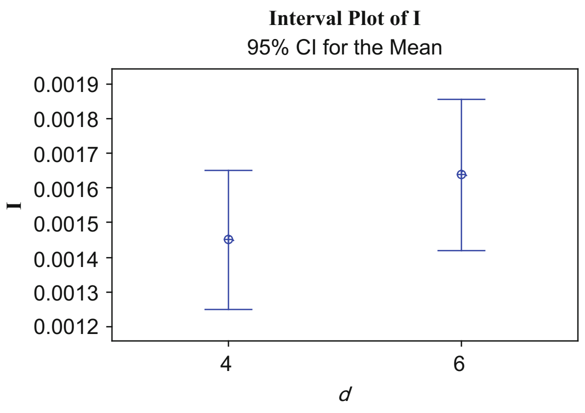

Fig. 7. Mean plot for index I_{hs} and the corresponding confidence interval for the two levels of d.

200 and 500 jobs with 5, 10 and 20 machines. For the second test, 25 set of instances were generated in accordance with the Taillard indications. These 25 sets can be separated into small instances with n=14, 15, 16 and 17 jobs combined with m=3, 4, 5 and 6 machines (the optimal solutions of which have been obtained with the LOMPEN algorithm proposed in [27]), and large instances with n=50, 100 and 150 jobs combined with m=5, 10 and 15 machines. For every combination 100 problem instances were generated. Therefore, 2500 problem instances were considered. The quality of the solution obtained by each heuristic h for each instance s has been measured using the I_{hs} index calculated as in (6).

5.2.1. Results of the first test

The first test has two main objectives: first, to analyze the performance of the proposed tie-breaking strategy and second, to analyze the improvement obtained with the application of each procedure on the direct and inverse instances retaining the best of both solutions (steps 1 and 2).

To achieve the first objective we have implemented the NEH algorithm with the different tie-breaking criterion proposed in the literature. We have named NEH0 the original implementation, NEH1 the implementation with the tie-breaking strategy proposed by Kalczynski and Kamburowski [16] in their NEH1 algorithm, NEH2 the implementation with the tie-breaking used in the NEHKK1 algorithm proposed by [16], NEH3 the implementation with the strategy proposed by Dong et al. [18] and NEHR the implementation with our strategy. Note that NEH0,

Table 3
Average of index I_{hs} on Taillard's instances.

	d-NEH0	NEH0	d-NEH1	NEH1	d-NEH2	NEH2	d-NEH3	NEH3	d-NEHR	NEHR
20 × 5	3.30	2.56	2.69	2.65	2.73	2.57	**2.48**	2.48	2.52	2.33
20 × 10	4.60	4.26	4.35	4.35	4.31	4.31	**4.13**	4.13	4.32	3.87
20 × 20	3.73	3.36	3.68	3.68	**3.41**	3.38	3.70	3.70	3.54	3.29
50 × 5	0.73	0.58	0.87	0.74	0.59	0.59	0.73	0.60	0.60	0.47
50 × 10	5.07	4.90	5.08	5.03	4.87	4.60	**4.80**	4.68	4.83	4.35
50 × 20	6.66	6.01	6.51	6.33	6.42	6.18	6.25	5.96	**5.77**	5.56
100 × 5	0.53	0.40	0.48	0.41	0.40	0.40	0.49	0.41	**0.35**	0.34
100 × 10	2.21	1.97	2.10	1.92	1.77	1.64	**1.96**	1.57	2.08	1.68
100 × 20	5.34	5.19	5.28	5.05	5.28	5.05	**5.01**	4.67	5.44	5.09
200 × 10	1.26	1.14	1.19	1.08	1.17	1.10	**1.01**	0.90	1.02	0.94
200 × 20	4.41	4.31	4.41	4.21	4.23	4.03	**3.88**	3.69	4.17	4.04
500 × 20	2.06	2.01	1.98	1.93	2.03	1.88	**1.71**	1.62	1.96	1.81
Average	3.33	3.06	3.22	3.11	3.10	2.98	3.01	2.87	3.05	**2.81**

Table 4
Average of index I_{hs} obtained by the heuristics on Taillard's instances.

	d-KKER	KKER	d-NEHR	NEHR	d-NMER	NMER	d-RAER	RAER
20 × 5	2.46	**2.32**	2.52	2.33	2.71	2.37	3.12	2.57
20 × 10	4.97	4.11	4.32	3.87	4.13	**3.15**	5.94	4.18
20 × 20	3.47	**3.25**	3.54	3.29	3.98	3.42	4.57	3.61
50 × 5	0.74	0.49	0.60	**0.47**	0.88	0.71	1.67	1.22
50 × 10	5.08	**4.35**	4.83	**4.35**	5.03	4.63	5.59	5.59
50 × 20	5.98	5.69	5.77	**5.56**	5.93	5.68	6.68	6.14
100 × 5	0.36	0.35	0.35	0.34	0.47	**0.32**	0.92	0.55
100 × 10	1.82	1.78	2.08	**1.68**	2.14	1.73	2.65	2.24
100 × 20	5.37	**4.70**	5.44	5.09	5.41	5.00	5.71	5.32
200 × 10	1.11	0.97	1.02	0.94	1.07	**0.91**	1.69	1.56
200 × 20	4.18	3.93	4.17	4.04	3.99	**3.86**	4.57	4.25
500 × 20	1.87	**1.77**	1.96	1.81	1.92	1.82	2.41	2.21
Average	3.12	2.81	3.05	2.81	3.14	**2.80**	3.79	3.29

Table 5
Average of index I_{hs} obtained by the heuristics on Taillard's benchmark.

	IGA	KKER+	NEHR+	NMER+	RAER+
20 × 5	0.073	**0.041**	0.059	**0.041**	**0.041**
20 × 10	**0.030**	0.119	0.119	0.079	0.120
20 × 20	**0.067**	0.102	0.174	0.094	0.167
50 × 5	**0.016**	0.025	0.048	0.031	0.035
50 × 10	**0.755**	1.148	0.938	1.152	0.969
50 × 20	**1.367**	2.428	2.178	2.376	2.006
100 × 5	0.064	0.041	0.052	**0.040**	0.051
100 × 10	**0.417**	0.461	0.470	0.446	0.473
100 × 20	**1.802**	2.279	2.087	2.343	2.117
200 × 10	0.462	**0.351**	0.408	0.351	0.437
200 × 20	2.222	1.913	1.983	**1.912**	1.935
500 × 20	1.273	1.115	1.114	**1.020**	1.192
Average	**0.712**	0.835	0.802	0.823	0.795

NEH1, NEH2, and NEH3 have the complexity of $O(mn^2)$ while NEHR requires $O(m^2n^2)$ time.

Table 3 shows the average of index I_{hs} for the 10 instances of each set and procedure. Columns with a "d-" before the name of the procedure indicate that it is applied only on the direct instance whereas the other columns indicate that the procedures are applied on the direct and inverse instances retaining the best of both solutions found.

In Table 3 it can be noted that when the heuristics are applied on the direct instance the best results are obtained with the tie breaking proposed by Dong et al. [18], column d-NEH3. However, when the procedures are applied on the direct and inverse instances retaining the best of both solutions, on average, the best results are obtained using the new tie-breaking proposed here.

To achieve our second objective, we have compared the results obtained by applying the 4 variants on the direct instance with those obtained when they are applied on the direct and inverse instances retaining the best of both solutions found. The results are shown in Table 4 where, as before, a column with a "d-" before the name of the procedure indicates that it is applied only on the direct instance. We can observe that the best results are obtained with NMER, NEHR and KKER but from the average value obtained it is not possible to say if one is better than the other. It can be also noted that, on average, the obtained improvement when procedures are applied on the direct and inverse instances is between 8% and 13%, as can be seen in Table 4. Therefore, one can conclude, that both the tie breaking method proposed and the execution of the algorithm on the direct and inverse instances are advisable for improving the results obtained with the original NEH heuristic.

Next, in order to analyze the improvement brought about by the improvement procedure (step 3), the KKER+, NEHR+, NMER+ and RAER+ algorithms were executed on these same instances.

Due to the stochastic nature of the SSA algorithm, 3 runs for each instance are carried out and the results are averaged to calculate the I_{hs} index. We have compared the results obtained with these procedures with those obtained with the IGA heuristic proposed by Ruiz and Stützle [23]. This is worthwhile because it has been shown that theirs is one of the most efficient algorithms for this problem. We have implemented this algorithm according to the indications found in [23]. We have used as stopping criteria, for all algorithms, the computational time which has been set to $n \cdot m \cdot 50$ ms. Table 5 shows the average value of the I_{hs} index of the 10 instances of each set, for each of the proposed procedures.

In Table 5 it can be noted that the best average is obtained with the IGA algorithm but, according to the 95% confidence intervals of Fig. 8, this difference is not significant. In the second test, with a bigger number of instances, we will try to reach a conclusion as to whether the performance of one procedure is better than the others.

To finish with the first test we are going to analyze the improvement made to the solution when step 3 is applied. It can be observed, comparing the values of Tables 4 and 5, that when the improvement procedure is applied, the discrepancy in the solutions obtained with respect to the optimal value, or the best-known solution, diminishes, approximately, to a third. It is worth pointing out that when step 3 is applied the discrepancies between the procedures fall. For example, if we focus on the average values of the I_{hs} index in Table 4, RAER is about 17% worse than NMER, which is totally predictable as RAER constructs the initial solution randomly. In contrast, if we observe these same values in Table 5 we see that, contrary to what one could think, RAER+ is, on average, better than NMER+. It will be a point to confirm in the second test presented in the next section.

5.2.2. *Results of the second test*

The second test performed used an increased number of instances which allows us to carry out a detailed statistical analysis of whether there exist significant differences between these procedures. For each algorithm, 3 independent runs were performed.

First we analyze the results obtained on the small size instances which are combinations of $n=\{14, 15, 16, 17\}$ jobs with $m=\{3, 4, 5, 6\}$. For every combination 100 problem instances were generated. Therefore, 1600 problem instances were considered. The results obtained are analyzed with I_{hs} calculated according to (6). The optimal solutions are obtained with the Lompen algorithm proposed in [27]. The results were compared using a three way completely randomized ANOVA with interactions [28] to test if there are significant differences between algorithms. We have considered as factors the algorithm, the number of jobs (n) and the number of machines (m). The interactions allow to study if the algorithms' behaviour is constant for all the cases, or, on the contrary, produce better results in some cases than in others. Although the data obtained do not comply with the normality hypothesis, due to the high number of zeros, this hypothesis is known not to be critical [29], especially with the large amount of data available in the second case. The critical hypothesis of independence is satisfied (this is obvious from the way the test was designed) and to a great extent the homocedasticity hypothesis is also satisfied. In any case the violations of the hypothesis are too small to affect the results and conclusions obtained. Furthermore, the same results have been obtained using non-parametric techniques, although logically with lower signification levels. The ANOVA results are shown in Table 6.

The results of the analysis show that there are highly significant differences among the procedures and among the instances (both p-values for number of jobs (n) and number of machines (m) are 0.00). It can be observed that the interaction between n and m and between m and *Algorithm* are significant.

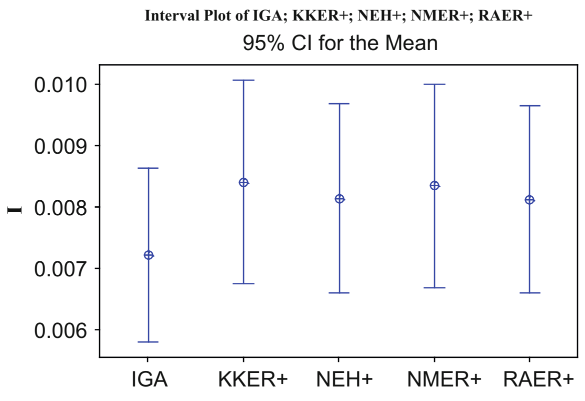

Fig. 8. Mean plot and confidence interval for the tested heuristics.

Table 6
Three way ANOVA table for *Algorithm*, n and m.

Source	DF	SS	MS	F	P
Algorithm	4	13.7092	3.4273	29.51	0.000
n	3	5.4989	1.8330	15.78	0.000
m	3	171.5620	57.1873	492.48	0.000
Algorithm $*$ n	12	0.6144	0.0512	0.44	0.947
Algorithm $*$ m	12	12.9249	1.0771	9.28	0.000
$n*m$	9	9.3521	1.0391	8.95	0.000
Error	7956	923.8591	0.1161		
Total	7999	1137.5206			

Table 7
Average of index I_{hs} for each algorithm stratified by n and m.

Machines	Jobs	IGA	KKER+	NEHR+	NMER+	RAER+
5	50	**0.0052**	0.0076	0.0084	0.0073	0.0090
	100	**0.2432**	0.3921	0.3202	0.3911	0.3186
	150	**0.3823**	1.2855	1.0190	1.2949	1.0199
10	50	0.0037	0.0017	0.0023	**0.0015**	0.0040
	100	0.1531	**0.1300**	0.14869	0.1362	0.1700
	150	**0.4644**	0.7788	0.7811	0.7948	0.8342
20	50	0.0049	**0.0038**	0.0059	0.0039	0.0060
	100	0.1225	0.0844	0.0948	**0.0758**	0.1087
	150	0.7851	**0.5221**	0.5661	0.5435	0.5503

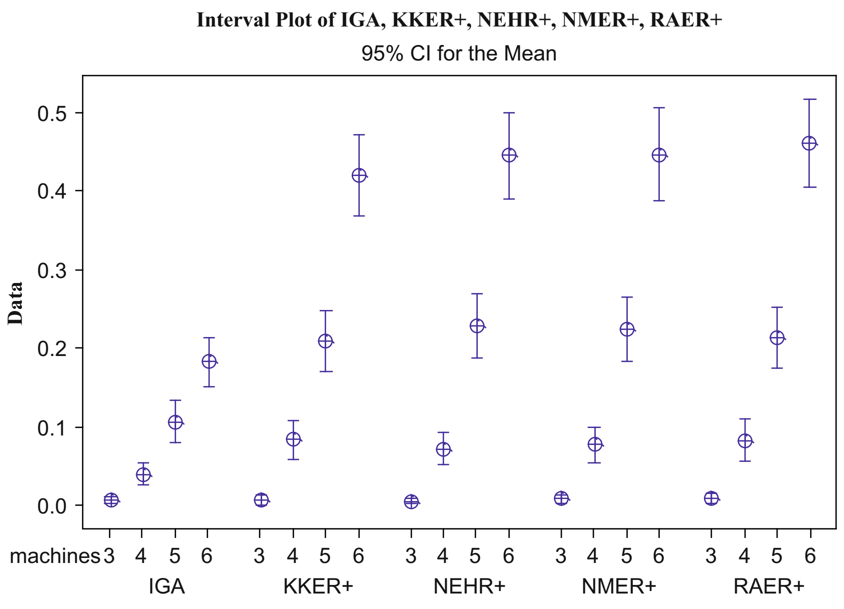

Fig. 9. Interval plot for the average index I_{hs} for each procedure stratified by the number of machines.

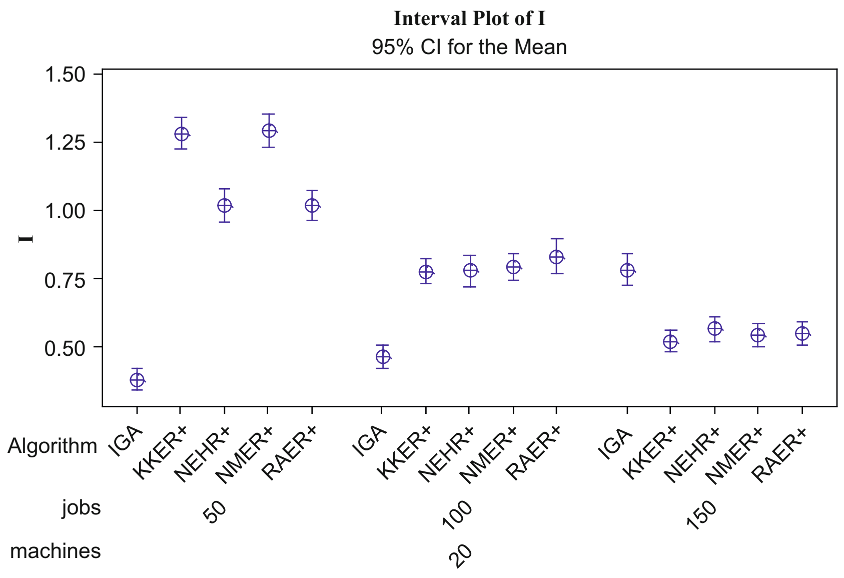

Fig. 10. Interval plot of the average of index I_{hs} for procedure stratified by jobs when $m=20$.

By contrast, the p-value for the interaction of the *Algorithm* and n is 0.947 which means that the procedures' behaviour does not depend of the number of jobs. Fig. 9 shows confidence intervals for the average index I_{hs}, for each algorithm stratified by number of machines.

In Fig. 9, we can observe that the IGA procedure is significantly better than the other procedures. It can be also observed that when the number of machines increases the results obtained are significantly worse in all procedures.

The next step is to analyze the obtained results on the large size instances which combine $n=\{50, 100, 150\}$ with $m=\{5, 10, 20\}$. For every combination, 100 problem instances were generated. The obtained results were analyzed as before by a three ways completely randomized ANOVA with interactions and with the I_{hs} calculated as (6).

Table 7 shows the average value of index I_{hs} for each combination of n and m. It can be noted that the IGA procedure is the best when the number of machines is 5 but when m increases its performance diminishes. This very clear effect is responsible for the *Algorithm* $\times$ *number of machines* interaction and it can be seen, in the interval plot of Fig. 10, that the difference is clearly significant.

It can also be noted that the other procedures behave in a similar way even when the initial solution is generated randomly. This enables us to conclude that the improvement procedure implemented, though straightforward, is very powerful when it comes to improving the initial solutions.

6. Conclusions

This paper presents a competitive heuristic procedure for the resolution of the permutation flow shop problem. The aim is to minimize the maximum completion time of jobs or makespan. The procedure consists of 3 steps. The first two steps are conceived based on the NEH heuristic and use the reversibility property as a means of improving the procedure's performance. The last step consists of an iterated local search improvement procedure with a local search based on swap-move with random path and tie breaking procedure.

Firstly, we have put forward two tools to improve the performance of the NEH heuristic. The NEH heuristic was used as it is one of the best constructive procedures for the problem dealt with. The first tool is a new tie breaking strategy, based on the machines' loads, to be used in the insertion phase. The second tool takes advantage of the reversibility of the problem, and proposes the application of the procedure to the direct and inverse instance retaining the best of both solutions found.

Two tests were carried out to analyze the efficiency of the procedure designed. The objective of the first test, which used the well-known instances of Taillard, was to analyze the impact of each one of the measures adopted in each step. Comparing the results obtained with the different variants implemented in steps 1 and 2 with respect to the NEH heuristic, it was shown that there is an increase in the quality of the solution obtained through the combination of the reversibility property and the tie breaking criterion, (step 2). Therefore, we recommend that this method be used in future applications. Step 3 shows that when the local improvement is applied the quality of the solution increases significantly.

The reduced number of instances considered in the first test makes a detailed statistical analysis of the results worthless. Therefore, a second test was performed on 2500 instances created according to the Taillard indications. These instances have been separated into two groups: small size instances, for which the optimal solutions were calculated by the exact algorithm proposed in [27], and large size instances. In this second test, an ANOVA analysis was carried out to determine if there were significant differences between the different variants implemented. The analysis carried out with small size instances shows the superiority of the IGA procedure but with large size instances it has been shown that this superiority diminishes when the number of machines considered increases.

Finally, we have shown the efficiency of the improvement procedure because it is the RAER+ variant which, on average, obtains similar results than the other procedures which use good initial solutions. We believe that the proposed procedure can be also efficient for solving the blocking flow shop problem with makespan criterion.

Acknowledgements

The authors wish to thank the referees for the insightful comments on an earlier version of the paper. The research conducted by the authors has been partly funded by the Spanish Ministry of Science and Innovation under Grant DPI2007-61371.

References

[1] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 1979;5:287–326.

[2] Hejazi RS, Saghafian S. Flowshop-scheduling problems with makespan criterion: a review. International Journal of Production Research 2005;43(14):2895–929.

[3] Johnson SM. Optimal two- and three-stage production schedules with set up times included. Naval Research Logistics Quarterly 1954;1:61–8.

[4] Gupta JND, Stafford J, Edward F. Flowshop scheduling research after five decades. European Journal of Operational Research 2006;169(3):699–711.

[5] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: Freeman; 1979.

[6] Ruiz R, Maroto C. A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research 2005;165(2): 479–94.

[7] Giglio RJ, Wagner HM. Approximate solutions to the three-machine scheduling problem. Operations Research 1964;12(2):305–24.

[8] Palmer DS. Sequencing jobs through a multi-stage process in the minimum total time- a quick method of obtaining a near optimum. Operations Research Q 1965;16:101–7.

[9] Gupta JND. A functional heuristic algorithm for the flow shop scheduling problem. Operations Research Q 1971;22(1):39–47.

[10] Campbell HG, Dudek RA, Smith ML. An heuristic algorithm for the n job m machine sequencing problem. Management Science 1970;16:630–7.

[11] Companys R. Métodos heurísticos en la resolución del problema del taller mecánic. Estudios Empresariales 1966;5(2):7–18.

[12] Dannenbring DG. An evaluation of flow shop sequencing heuristics. Management Science 1977;23(1):1174–82.

[13] Nawaz M, Enscore Jr EE, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 1983;11(1):91–5.

[14] Nagano MS, Moccellin JV. A high quality constructive heuristic for flow shop sequencing'. Journal of the Operational Research Society 2002;53:1374–9.

[15] Framinan JM, Leisten R, Ramamoorthy B. 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 2003;41(1):121–48.

[16] Kalczynski PJ, Kamburowski J. An improved NEH heuristic to minimize makespan in permutation flow shops. Computers & Operations Research 2008;35(9):3001–8.

[17] Rad SF, Ruiz R, Boroojerdian N. New high performing heuristics for minimizing makespan in permutation flowshops. Omega 2009;37(2): 331–45.

[18] Dong X, Huang H, Chen P. An improved NEH-based heuristic for the permutation flowshop problem. Computers & Operations Research 2008;35(12):3962–8.

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

[20] Suliman S. A two-phase heuristic approach to the permutation flow-shop scheduling problem. International Journal of Production Economics 2000;64:143–52.

[21] Baum EB. Towards practical 'Neural' computation for combinatorial optimization problems. Neural networks for computing, AIP conference proceedings, 1986. p. 53–64.

[22] Stützle T. Applying iterated local search to the permutation flowshop problem. Technical report, AIDA-98-04, FG Intellektik, FB Informatik, TU Darmstadt, 1998. p. 1–16.

[23] Ruiz R, Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 2007;177(3):2033–49.

[24] Brown APG, Lomnicki ZA. Some applications of the 'branch-and-bound' algorithm to the machine scheduling problem. Operations Research Q 1966;7(4):173–86.

[25] McMahon GB, Burton PG. Flow-shop scheduling with the branch-and-bound method. Operations Research 1967;15(3):473–81.

[26] Taillard E. Benchmarks for basic scheduling problems. European Journal of Operational Research 1993;64(2):278–85.

[27] Companys R, Mateo M. Different behaviour of a double branch-and-bound algorithm on $Fm|prmu|C_{max}$ and $Fm|block|C_{max}$ problems. Computers & Operations Research 2007;34(4):938–53.

[28] Jeff Wu CF, Hamada M. Experiments: planning, analysis, and parameter design optimization. Wiley series in probability and statistics. New York: Wiley-Interscience; 2000.

[29] Box GEP, Andersen SL. Permutation theory in derivation of robust criteria and the study of departures from assumptions. Journal of the Royal Statistical Society 1955;series B(17):1–26.