Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Evaluating the performance of constructive heuristics for the blocking flow shop scheduling problem with setup times

Mauricio Iwama Takano[a] and Marcelo Seido Nagano[b*]

[a]*Federal Technological University - Paraná, Av. Alberto Carazzai, 1640, 86300-000, Cornélio Procópio, Paraná, Brazil*
[b]*University of São Paulo, São Carlos School of Engineering, Av. do Trabalhador São-Carlense, 400, 13566-590, São Carlos, SP, Brazil*

C H R O N I C L E

Article history:
Received January 30 2018
Received in Revised Format
February 18 2018
Accepted May 4 2018
Available online
May 5 2018

Keywords:
Flow shop
Blocking
Zero buffer
Setup times
Makespan
Heuristics

A B S T R A C T

This paper addresses the minimization of makespan for the permutation flow shop scheduling problem with blocking and sequence and machine dependent setup times, a problem not yet studied in previous studies. The 14 best known heuristics for the permutation flow shop problem with blocking and no setup times are pre-sented and then adapted to the problem in two different ways; resulting in 28 differ-ent heuristics. The heuristics are then compared using the Taillard database. As there is no other work that addresses the problem with blocking and sequence and ma-chine dependent setup times, a database for the setup times was created. The setup time value was uniformly distributed between 1% and 10%, 50%, 100% and 125% of the processing time value. Computational tests are then presented for each of the 28 heuristics, comparing the mean relative deviation of the makespan, the computa-tional time and the percentage of successes of each method. Results show that the heuristics were capable of providing interesting results.

1. Introduction

This paper addresses the scheduling problem in a permutation flow shop with zero buffer and a sequence and machine dependent setup environment, an NP-Complete problem (Gupta & Darrow, 1986) which has not yet been explored in the literature. In the problem, there are *n* jobs that must be processed in *m* machines and all jobs must be processed in all machines in the same flow. The permutation constraint indicates that the process sequence of the jobs must be the same for all the machines.

Many papers considered the problem with unlimited buffers (Nawaz et al., 1983) or considering the setup time embedded in the processing time of the job (Mccormick et al., 1989; Pan & Wang, 2012; Ronconi, 2004). In this paper, the setup time is separated from the process time, allowing a machine to be prepared to initiate the process of a job before the previous machine has finished processing this job. This ensures better flexibility for scheduling, allowing for a better use of time, hence a possible reduction in the makespan. Furthermore, the setup time depends on the sequence of jobs, as well as the machine. In other words, there is a different setup time for each pair of jobs in each machine.

* Corresponding author
E-mail: drnagano@usp.br (M. S. Nagano)

2019 Growing Science Ltd.
doi: 10.5267/j.ijiec.2018.5.002

In an environment with blocking, there are limited buffers between the machines. In this paper, a zero buffer constraint is considered, i.e., if machine k finishes the process of job j and machine k+1 is not able to receive the job (because it is still processing job j-1 or is still being set up), the job remains in machine k, blocking it. In this case machine k is unable to receive the next job of the sequence. The flow shop problem with zero buffer can be used to model any flow shop problem with limited buffer because a unit capacity buffer can be represented by a machine with zero processing time for all jobs (Mccormick et al., 1989). Fig. 1 shows an example of the problem.

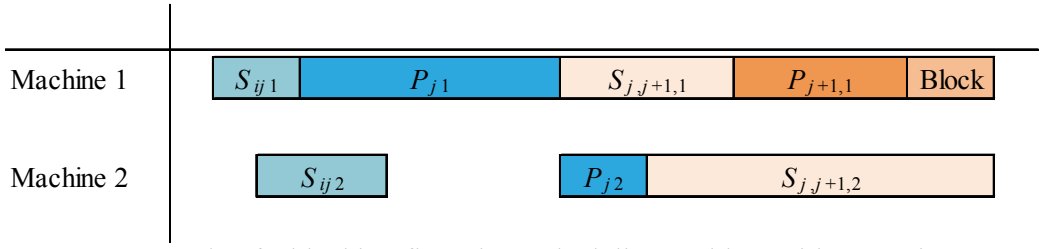

Fig. 1. Example of a blocking flow shop scheduling problem with setup times

Mccormick et al. (1989) presented a heuristic method called Profile Fitting (PF) for the flow shop problem with zero buffer, in which the jobs are scheduled in such a way that the idle and blocking times of the machines are minimized. Ronconi (2004) addressed the zero buffer problem with the objective of minimizing the makespan. A constructive heuristic method called MM (MinMax) that uses specific characteristics of the problem is presented and compared to the PF and LPT (Longest Processing Time) heuristics. These heuristics were also used as the initial solution for an adapted version of the constructive heuristic Nawaz-Enscore-Ham (NEH). These proposed heuristics were called MME and PFE, and were compared to the original NEH. Both comparisons were made using the database provided by Taillard (1993). In the first comparison, the PF heuristic obtained better results and in the second comparison the PFE heuristic outperformed the other heuristics regarding the number of better solutions.

Pan and Wang (2012) addressed the flow shop with blocking problem aiming at minimizing the makespan. Initially, the authors presented two constructive heuristic methods called wPF and PW, both based on the PF procedure. Then they proposed five II phase heuristics (according to the classification by Framinan et al., 2004) called wPFE, PWE, PF-NEH(x), wPF-NEH(x) and PW-NEH(x) by combining the previous methods with the classic NEH. Finally, using a local search method based on job insertion, three III phase heuristics were developed, called PF-NEH_{ls}(x), wPF-NEH_{ls}(x) and PW-NEH_{ls}(x). The proposed methods were evaluated and compared with other existing methods using the database provided by Taillard (1993). The experiments showed that the proposed methods outperformed all other methods previously presented in the literature. Furthermore, the III phase methods were able to significantly improve the results obtained by the constructive methods. The computational tests provided better solutions for 17 large problems of the database.

Pan et al. (2013) proposed a memetic algorithm (MA) using a combination of the PF and the NEH heuristics to provide an initial solution for the minimization of the makespan in a permutation flow shop problem with zero buffer. The proposed meta-heuristic was compared to many previously reported algorithms. The proposed MA algorithm not only outperformed the other algorithms, but it also improved 75 out of the 120 problems from Taillard´s (1993) database best known makespan values.

Han et al. (2015) presented a novel discrete artificial bee colony (DABC) incorporated with differential evolution (DE) to minimize the makespan in a permutation flow shop problem with zero buffer. The algorithm is tested with the well-known Taillard (1993) database and compared to many previously reported algorithms. The results demonstrate the superiority of the proposed algorithm called DE-ABC in terms of the mean relative deviation of the makespan when compared to the other existing algorithms.

Ribas and Companys (2015) developed heuristics based on PF and NEH for the $F_m \mid block \mid \sum_i C_i$ problem. Experimental results show that all the heuristics present similar performances and HPF2 and NHPF2 are the best heuristics for the problem. Fernandez-Viagas et al. (2016) proposed a Beam-Search-Based constructive heuristic for the $F_m \mid block \mid \sum_i C_i$ problem. The heuristic adopts a beam-search-based strategy that combines the diversification of the based-population algorithms and the efficiency of constructive heuristics. Some versions of the BS(x) outperform the main heuristics for the problem. Tasgetiren et al. (2016) developed the tPF, tPF_NEH(x), PFT and PFT_NEH(*x*) heuristics based on PF_NEH(x) by Pan and Wang (2012) for the $F_m \mid block \mid \sum_i C_i$ problem. Computational results show that tPF_NEH(x) outperforms HPF2 and NHPF2 by Ribas and Companys (2015). Shao et al. (2017) proposed seven improved versions of FPDNEH with different tie-break mechanisms to the partial sequence during the insertion phase of the heuristic. Experimental results show that the proposed heuristics outperformed tPF_NEH(x) by Tasgetiren et al. (2016) and $FPDNEH_{IT1}$ was the best heuristic evaluated.

Tasgetiren et al. (2017) proposed PFT_NEH(x) for the $F_m \mid block \mid \sum_i C_i$ problem. PFT_NEH(x) uses an index to generate the initial sequence based on the processing times, the front delay of each job and a different cost function. Experimental results show that the proposed heuristic, in general, outperforms the heuristics by Pan and Wang (2012). Sanches et al. (2017) evaluated heuristic methods to obtain an initial solution for a branch-and-bound algorithm to minimize the makespan in a flow shop problem with blocking. The best heuristic methods found in the literature were used to provide an initial solution to the problem, and then each of them was evaluated in terms of computational time and the mean relative deviation of the number of nodes and of the makespan. The heuristic methods used were: MM (Ronconi, 2004); PF (Mccormick et al., 1989); PW and wPF (Pan & Wang, 2012). Results show that the use of an initial solution for the branch-and-bound algorithm enhances its performance.

Nagano et al. (2017) proposed an evolutionary clustering search (ECS) for the total tardiness blocking flow shop problem. The proposed ECS uses NEH-based procedure to generate an initial solution, the genetic algorithm to generate solutions and a variable neighborhood search (VNS) to improve the solutions. The proposed method was compared to the iterated local search (ILS) (Ribas et al., 2013). Computational tests show superiority of the new method for the set of problems evaluated. Also 67 values of the best-know values of the total tardiness criteria table presented by Ribas, Companys and Tort-Martorell (2013) were updated. The first study to address the flow shop problem with limited buffers and sequence and machine dependent setup found in the literature is Norman (1999). The evaluation criterion used in this study is the minimum makespan. A Tabu search and two adapted constructive heuristic methods (NEH and PF) were presented to solve the problem. A greedy improvement procedure was added to the constructive heuristics. Nine hundred problems were generated to evaluate the proposed methods varying setup times, buffer sizes and number of jobs and machines.

Maleki-Darounkolaei et al. (2012) developed a MILP model and a Simulated Annealing (SA) approach for the flow shop problem with three work stations, sequence dependent setup time only in the first stage and blocking time between each stage with two objectives (to minimize the makespan and the flow time). Problems with more than nine jobs were not solved due to the high computational time. Takano and Nagano (2017) presented a branch-and-bound method to minimize the makespan in a permutation flow shop with blocking and setup times. The selection of the node to be branched was used by Ronconi (2005) and is made by the depth first rule, where the node with the highest number of jobs in the partial sequence is selected. The structural property of the problem is presented, and four lower bounds for the problem, which explore this property, are proposed. The four lower bounds are then compared to each other. Next, a MILP model is presented and is compared to the best lower bound. Results show the consistency of the proposed lower bounds.

It is important to note that although three papers discuss the blocking with sequence dependent setup times problem, no studies were found in the literature that use heuristic methods to solve the zero buffer problem together with the sequence and machine dependent setup time in all machines. Therefore, there are no heuristic methods to solve this problem in the literature. In this paper, 11 adapted models for this problem are presented. The models were originally formulated by Ronconi (2004) and Pan and Wang (2012) for the permutation flow shop problem with blocking and no setup constraints.

This paper is structured as follows. The calculus used to calculate the makespan is presented in Section 2. The heuristic methods and their adaptations are explained in Section 3. The computational tests for the models and their results are presented in Section 4. Finally, the conclusions are presented in Section 5.

2. Material and methods

This section describes how to calculate the makespan in a permutation flow shop environment with blocking and sequence and machine dependent setup time and presents the adapted constructive heuristics for the problem. The heuristics are divided according to the classification method proposed by Framinan et al. (2004).

2.1. Makespan calculus

Let σ={1, 2, …, *i*, *j*, …, *n*} be an arbitrary sequence of jobs, *k*={1, 2, …, *m*} be the sequence of available machines, *i* be the job that directly precedes job *j* in the sequence, P_{jk} be the processing time of the *j*-th job in the sequence in machine *k*, S_{ijk} be the setup time of machine *k* between the *i*-th and the *j*-th job in the sequence, S_{01k} be the setup time of machine *k* before processing the first job in the sequence, R_{jk} be the completion time of the setup of machine *k* to the *j*-th job in the sequence and C_{jk} be the departure time of the *j*-th job in the sequence in machine *k*. The makespan is then calculated by the following:

$$R_{1k} = S_{01k} \qquad 1 \le k \le m \tag{1}$$

$$C_{j1} = \max\left(R_{j2}, R_{j1} + P_{j1}\right) \qquad 1 \le j \le n \tag{2}$$

$$C_{jk} = \max\left(R_{j,k+1}, C_{j,k-1} + P_{jk}\right) \qquad 1 \le j \le n; 2 \le k \le m-1 \tag{3}$$

$$C_{jm} = C_{j,m-1} + P_{jm} \qquad 1 \le j \le n \tag{4}$$

$$R_{jk} = C_{ik} + S_{ijk} \qquad 2 \le j \le n; 1 \le k \le m \tag{5}$$

Initially, the setup completion times of the machines for the first job in the sequence are calculated by Eq. (1). Next, the departure times of the first job in all machines are calculated by Eqs. (2-4). Then, Eq. (5) is used to calculate the setup completion time of all machines for the following job. Eq. 2, 3, and 4 are used again to calculate the departure times of the following job in all machines. The makespan (*Cmax*) is equal to the departure time of the last job in the sequence in the last machine. In other words, *Cmax*=*Cnm*.

3. The Adapted Constructive Heuristics

The best constructive heuristics for the flow shop problem with blocking found in the literature were adapted for the problem studied. The methods were divided into classes according to the framework proposed by Framinan et al. (2004) to make it easier to evaluate.

According to Framinan et al. (2004), the development of a heuristic may consist of three phases: Phase I – Index development; Phase II - Solution construction; Phase III – Solution improvement.

Each heuristic may consist of one or more phases, however a heuristic that consists of more than one of these phases must perform the phases according to the order presented. The heuristic is classified according to the last phase that is performed.

3.1. Phase I heuristics

According to Framinan et al. (2004), in this phase the jobs are sequenced according to a certain property based on the data of the problem. In this paper, the phase I heuristic models are: MM proposed by Ronconi (2004); PF proposed by McCormick et al. (1989); and wPF and PW proposed by Pan and Wang (2012). The MinMax (MM) method from Ronconi (2004) tries to schedule the jobs based on the makespan properties of the problem. Initially, the MM heuristic sets the first and last jobs in the sequence according to Johnson's rule (Johnson, 1954). In other words, the first job in the sequence is the one with the shortest processing time in the first machine and the last job in the sequence is the one with the shortest processing time in the last machine. The next job in the sequence, starting from the second position, is the job with the smallest sums of maximum values between the processing times of consecutive machines and by the sum of m processing times. Therefore, the consecutive job to the job (i) already set in the sequence is the job (j) with the smallest value of the following expression:

$$\alpha \sum_{l=1}^{m-1} \left| P_{jl} - P_{i,l+1} \right| + (1-\alpha) \sum_{k=1}^{m} P_{jk} \tag{6}$$

where α is a parameter that weighs the two terms of the expression. The same method was applied to the permutation flow shop problem with blocking and sequence and machine dependent setup times. The first and the last jobs were chosen as the ones with the smallest processing time in the first and in the last machine, respectively. Therefore, Eqn. 6 was used to select the position of the other jobs in the sequence. Parameter α was set to 0.6, as the method presents its best performance for this value of α (Ronconi, 2004). Also another adaptation of the method is proposed. As the setup time is also a variant in this problem, it was also considered in the calculus. This adapted method was called MM1. The first and last jobs of the sequence are the jobs with the smallest sum of the processing time and the setup time in the first and in the last machine, respectively. The second job in the sequence is defined as the job (c) that obtains the smallest value in the following expression:

$$\alpha \sum_{l=1}^{m-1} \left| \left(P_{jl} + S_{ijl} \right) - \left(P_{i,l+1} + S_{0jl} \right) \right| + (1-\alpha) \sum_{k=1}^{m} \left(P_{jk} + S_{ijk} \right), \tag{7}$$

where S_{0il} is the setup time of the first job in the sequence in machine l. The next job in the sequence, starting from the third position, is the job with the smallest value in the following expression:

$$\alpha \sum_{l=1}^{m-1} \left| \left(P_{jl} + S_{ijl} \right) - \left(P_{i,l+1} + S_{hjl} \right) \right| + (1-\alpha) \sum_{k=1}^{m} \left(P_{jk} + S_{ijk} \right), \tag{8}$$

where h is the preceding job of job i in the sequence. As both adaptations provide different results from each other, they are called MM and MM1. Both adapted methods are compared in Section 4. In the PF heuristic, the first job in the sequence is the one with the smallest sum of processing times. The next position in the sequence (c) belongs to the job (j) that obtains the smallest δ_j, which is obtained by the following expression:

$$\delta_j = \sum_{k=1}^{m} \left(C_{[c]k} - C_{[i]k} - P_{jk} \right). \tag{9}$$

To adapt the method to the problem, δ_j is calculated by the following expression:

$$\delta_j = \sum_{k=1}^{m} \left(C_{[c]k} - C_{[i]k} - P_{jk} - S_{ijk} \right), \tag{10}$$

where S_{ijk} is the setup time between the last job in the partial sequence and job (j). The first job in the sequence is still the one with the smallest sum of processing times. Another way of adapting the model is by considering the setup time in the calculus. As the setup time also depends on the sequence, the first job in the sequence is the one with the smallest sum of the processing times and setup times in all the machines. The next job in the sequence, starting with the second position, is the one that obtains the smallest sum of the idle, blocking and setup times. In other words, the next job in the sequence is the one with the smallest δ_j, given by Eq. (9). As both adaptations provide different results from each other, they are called PF and PF1. Both adapted methods are compared in Section 4.

The wPF heuristic from Pan and Wang (2012) works the same way as the PF heuristic. However, it considers a different weight for different machines and different positions in the sequence. The first job in the sequence is the job with the smallest sum of processing times, the next position in the sequence (c) belongs to the job (j) with the smallest δ_j, which is given by:

$$\delta_j = \sum_{k=1}^{m} w_k \left(C_{[c]k} - C_{[i]k} - P_{jk} \right), \tag{11}$$

where w_k is the weight of the machine and the position in the sequence, and is given by:

$$w_k = \frac{m}{k + \frac{(c-1)(m-k)}{n-1}}, \tag{12}$$

where c is the position in the sequence in which the job is being assigned. In order to adapt the method for the problem, δ_j was changed so it would calculate the weighted sum of idle and blocking time in all the machines:

$$\delta_j = \sum_{k=1}^{m} w_k \left(C_{[c]k} - C_{[i]k} - P_{jk} - S_{ijk} \right). \tag{13}$$

The first job in the sequence is the one with the smallest sum of processing time. Another way of adapting the method is by considering the setup time in the calculus as it is a variable that depends on the sequence and on the machine. Therefore, the first job in the sequence is the one with the smallest sum of processing and setup times in all machines. The next position in the sequence (c) belongs to the job (j) with the smallest weighted sum of setup, idle and blocking times, i.e., the job with the smallest δ_j, given by Eqn. 11. As both adaptations provide different results from each other, they are called wPF and wPF1. Both adapted methods are compared in Section 4. The PW heuristic from Pan and Wang (2012) also tries to provide the schedule that minimizes the sum of the idle and blocking times. However, in PW the job to be appended in a position of the sequence is determined by a function which has two parts: 1. calculating the weighted sum of the idle and blocking times of the job that are being tested; and 2. calculating the effects of the remaining jobs (those that are not part of the partial sequence) on the idle and blocking times. The first part of the method is similar to the wPF heuristic. For the second part, an artificial job (v) is created. This artificial job is then appended to position (c+1) in the sequence. The processing time of job v is the average processing time of all jobs that are not yet scheduled (NS), that is:

$$P_{vk} = \sum_{\substack{q \in NS \\ q \neq j}} \frac{P_{qk}}{n-c}. \tag{14}$$

The second part of the method uses P_{vk} to calculate $C_{[c+1],k}$. The sum of the idle and blocking times caused by the jobs that are not yet scheduled (X_j) is then calculated by:

$$X_j = \sum_{k=1}^{m} w_k \left(C_{[c+1]k} - C_{[c]k} - P_{vk} \right). \tag{15}$$

By combining the sum of the idle and blocking times caused by appending job j to position (c) of the sequence and the sum of the idle and blocking times caused by the jobs that are not yet scheduled (v), f_j is calculated:

$$f_j = (n-c-1)\delta_j + X_j\,, \tag{16}$$

where (n-c-1) is a weight used to balance the effects caused by the idle and blocking times in later jobs. As more jobs are appended to the partial sequence, the more efficient X_j will be, and therefore for earlier jobs, X_j has less effect on f_j and its effect gradually improves for later jobs. Thus, the job (j) that obtains the smallest f_j is appended to position (c) in the sequence. This index f_j is also used to determine the first job in the sequence, however, δ_j for the first position in the sequence is:

$$\delta_j = \sum_{k=1}^{m} w_k \left(C_{[c]k} - P_{jk} \right). \tag{17}$$

The first adaptation of this method assumes δ_j and X_j as the weighted sum of the idle and blocking times of job j and v, respectively. Therefore, δ_j was changed to the following expression:

$$\delta_j = \sum_{k=1}^{m} w_k \left(C_{[c]k} - C_{[i]k} - P_{jk} - S_{ijk} \right). \tag{18}$$

and X_j was adapted to the following:

$$X_j = \sum_{k=1}^{m} w_k \left(C_{[c+1]k} - C_{[c]k} - P_{vk} - S_{jvk} \right), \tag{19}$$

where S_{jvk} is the average setup time of all jobs that are not yet scheduled (NS), defined by:

$$S_{jvk} = \sum_{\substack{q \in NS \\ q \neq j}} \frac{S_{jqk}}{n-c}\,. \tag{20}$$

The second adaptation method used was to consider the setup time in the calculus for both δ_j and X_j. Therefore, for the second adaptation method, δ_j is calculated by Eqn. 17 and X_j is calculated using Eqn. 15. As both adaptations provide different results from each other, they are called PW and PW1. Both adapted methods are compared in Section 4.

3.2. Phase II heuristics

According to Framinan et al. (2004), in this phase the solution is constructed in a recursive manner trying one or more unscheduled jobs to be inserted in one or more positions of a partial schedule until the schedule is complete. In this paper, phase II heuristic models are: MME and PFE proposed by Ronconi (2004); wPFE, PWE, PF-NEH(x), wPF-NEH(x) and PW-NEH(x), proposed by Pan and Wang (2012).

MME, PFE, wPFE and PWE heuristics are a modification of the NEH (Nawaz, Enscore, and Ham; 1983) heuristic. While the NEH heuristic uses the LPT dispatching rule to provide an initial solution, the heuristics MME, PFE, wPFE, and PWE use the MM, PF, wPF, and PW heuristics, respectively, to provide an initial solution. The heuristics first obtain an initial solution using one of the phase I heuristics. Then, the first job in sequence π is set as the first job in the initial solution. The remaining jobs are set by an insertion heuristic. This procedure is carried out until all jobs are scheduled in sequence π.

All adapted phase I heuristics were used to provide an initial solution for these phase II heuristics. The objective of the NEH heuristic is the makespan for the problem, calculated with Eqns. 1 to 5. The adapted heuristics are called MME, MM1E, PFE, PF1E, wPFE, wPF1E, PWE and PW1E.

Similar to PFE, wPFE and PWE, the PF-NEH(x), wPF-NEH(x) and PW-NEH(x) heuristics obtain an initial solution and then improve the obtained result. However, the initial solution is obtained first by the LPT dispatching rule. Then, the first job of the initial solution is set as the first job of sequence β and the remaining jobs are scheduled using the PF, wPF or PW heuristics. Sequence β is then used as the initial solution for a modified NEH heuristic. The modified NEH heuristic only applies the insertion heuristic for the last λ jobs in the sequence, generating sequence π^1. This procedure is repeated x times, each time setting the first job of sequence β as the x-th job of the initial solution provided by the LPT dispatching rule, thereby generating x different sequences π (π^1, π^2, π^3, …, πx). Sequence π resulting in the lowest objective value is then the best result of the heuristic. All adapted phase I heuristics were used to provide an initial solution for these phase II heuristics. The objective of the NEH heuristic is the makespan for the problem, calculated with Eqns. 1 to 5. The adapted heuristics are called PF-NEH(x), PF1-NEH(x), wPF-NEH(x), wPF1-NEH(x), PW-NEH(x) and PW1-NEH(x).

3.3. Phase III heuristics

According to Framinan et al. (2004), in this phase an initial solution is improved by using a procedure, always resulting in a better quality solution than the initial solution provided. In this paper, the phase III heuristic models are: PF-NEH_{ls}(x), wPF-NEH_{ls}(x) and PW-NEH_{ls}(x) proposed by Pan and Wang (2012).

PF-NEH(x), wPF-NEH(x) and PW-NEH(x) heuristics are used to provide an initial solution to the PF-NEH_{ls}(x), wPF-NEH_{ls}(x) and PW-NEH_{ls}(x) heuristics, respectively. Therefore, a local search is used as an improvement procedure. The local search used is the referenced local search (RLS) presented by Pan and Wang (2012). The RLS procedure starts from the solution generated by the constructive heuristic and then iteratively moves in the neighbour solutions until a local optimum is found. All adapted phase II heuristics were used to provide an initial solution for these phase III heuristics. The adapted heuristics are called PF-NEH_{ls}(x), PF1-NEH_{ls}(x), wPF-NEH_{ls}(x), wPF1-NEH_{ls}(x), PW-NEH_{ls}(x) and PW1-NEH_{ls}(x).

4. Results and discussion

All 28 heuristics were used to solve the 120 problems proposed by Taillard (1993). These problems vary in the number of jobs and machines with 20, 50, 100, 200 and 500 jobs and 5, 10 and 20 machines. The setup times were uniformly distributed between 1 and 10%, 50%, 100% and 125% of the processing time maximum value. All heuristic algorithms were coded in C++. The experiments were performed in an Intel® core i7 3610QM with 2.3 GHz, 8 Gb DDR3 RAM and operational system Windows 7. The x value for PF-NEH(x), wPF-NEH(x), PW-NEH(x), PF1-NEH(x), wPF1-NEH(x), PW1-NEH(x), PF-NEH_{ls}(x), wPF-NEH_{ls}(x), PW-NEH_{ls}(x), PF1-NEH_{ls}(x), wPF1-NEH_{ls}(x) and PW1-NEH_{ls}(x) heuristics was set to 5 as it is the value that, according to Pan and Wang (2012), provides the best results for the heuristics.

As the number of results obtained is very large, only the average results of the computational time (CPU time) of the mean relative deviation of the makespan (MRD Makespan) and the percentage of success of each heuristic method (PS) for all the 120 problems for each setup time are presented. This percentage is defined as the total number of times the heuristic obtains the best makespan divided by the number of generated problems. Obviously, when two or more heuristics obtain the best makespan for the same problems, all of them achieve success, and consequently their percentage of success rates are improved simultaneously. The results are then presented divided into three classes according to the classification method proposed by Framinan et al. (2004). First, the obtained results for the adapted phase I heuristics are presented in Table 1, then the results for the adapted phase II heuristics are presented in Table 2. Finally, the results for the adapted phase III heuristics are presented in Table 3. Fig. 1, 2 and 3 depict the MRD of the makespan of phase I, II and III heuristics, respectively.

Table 1
Performance of phase I heuristics

Setup time	10%	50%	100%	125%	Mean
MM	0.83%*	0%	0%	0%	0.21%
	0.08593**	0.10224	0.08434	0.08535	0.08946
	6.84%***	10.96%	17.93%	20.53%	14.06%
MM1	2.50%	1.67%	0%	0%	1.04%
	0.26127	0.08303	0.26388	0.26361	0.21795
	6.65%	8.30%	11.10%	12.29%	9.59%
PF	10%	3.33%	0%	0%	3.33%
	0.03191	0.04014	0.03239	0.03274	0.03430
	2.63%	4.01%	6.82%	8.37%	5.46%
PF1	10.83%	40.83%	62.50%	65%	44.79%
	0.03442	0.01479	0.03486	0.03448	0.02964
	2.55%	1.48%	1.00%	0.94%	1.49%
wPF	10.83%	0.83%	1.67%	0%	3.33%
	0.03618	0.03984	0.03654	0.03639	0.03724
	2.16%	3.98%	7.12%	8.12%	5.35%
wPF1	11.67%	12.50%	15%	15%	13.54%
	0.03868	0.01881	0.03794	0.03833	0.03344
	2.14%	1.88%	1.99%	2.21%	2.06%
PW	32.50%	5.83%	1.67%	0.83%	10.21%
	0.23095	0.02948	0.23236	0.23183	0.18116
	0.90%	2.95%	6.09%	7.39%	4.33%
PW1	28.33%	35.83%	22.50%	19.17%	26.46%
	0.23425	0.00891	0.24754	0.24793	0.18466
	0.94%	0.89%	1.37%	1.75%	1.24%

*PS; **CPU time (sec); ***MRD Makespan

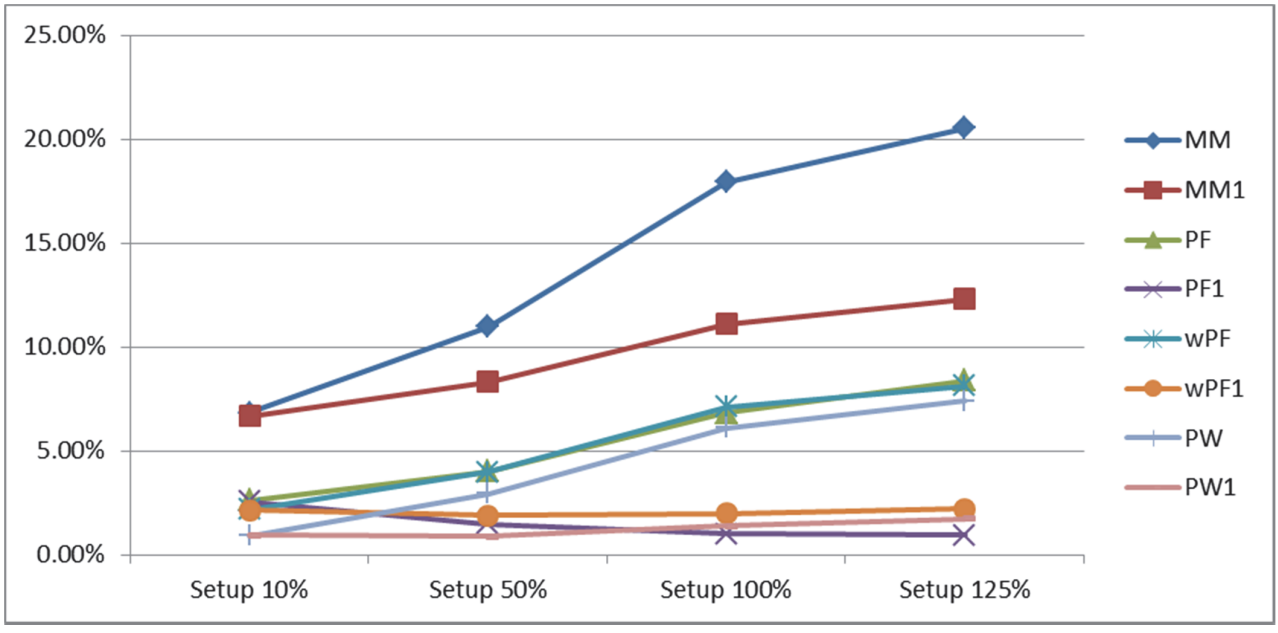

Fig. 2. MRD of the makespan of phase I heuristics

Table 2
Performance of phase II heuristics

Setup time	10%	50%	100%	125%	Mean
MME	4.17%*	4.17%	0%	0%	2.08%
	30.520**	30.533	30.579	30.365	30.499
	2.89%***	2.77%	3.98%	4.63%	3.57%
MM1E	3.33%	2.5%	2.5%	2.5%	2.71%
	30.678	30.608	30.461	30.444	30.548
	2.70%	2.71%	3.67%	4.28%	3.34%
PFE	1.67%	0%	0.83%	0.83%	0.83%
	29.271	29.202	29.205	29.256	29.234
	3.06%	2.92%	3.83%	4.30%	3.53%
PF1E	1.67%	3.33%	0%	1.67%	1.67%
	29.325	29.322	29.344	29.338	29.332
	3.07%	2.72%	3.25%	3.46%	3.13%
wPFE	3.33%	1.67%	0.83%	1.67%	1.88%
	28.788	28.804	28.982	29.006	28.895
	3.16%	3.09%	3.85%	4.39%	3.62%
wPF1E	2.5%	1.67%	3.33%	1.67%	2.29%
	30.013	30.052	30.075	30.094	30.059
	3.33%	2.88%	3.42%	3.85%	3.37%
PWE	0.83%	0%	0%	0%	0.21%
	29.124	29.172	29.098	29.211	29.151
	3.20%	3.02%	3.88%	4.30%	3.60%
PW1E	0.83%	1.67%	1.67%	1.67%	1.46%
	29.255	29.291	29.319	29.334	29.300
	3.01%	2.66%	3.21%	3.75%	3.16%
PF-NEH (x)	17.5%	3.33%	5%	3.33%	7.29%
	23.941	23.924	23.952	23.913	23.932
	0.98%	2.01%	3.78%	4.51%	2.82%
PF1-NEH (x)	21.67%	45%	50.83%	60%	44.38%
	24.924	24.881	24.969	24.962	24.934
	0.91%	0.64%	0.57%	0.42%	0.64%
wPF-NEH (x)	12.5%	6.67%	2.5%	1.67%	5.83%
	24.740	24.710	24.749	24.729	24.732
	0.72%	1.95%	3.68%	4.58%	2.74%
wPF1-NEH (x)	15%	14.17%	21.67%	15.83%	16.67%
	24.843	24.803	24.778	24.781	24.801
	0.78%	0.75%	0.93%	1.01%	0.87%
PW-NEH (x)	11.67%	5%	4.17%	4.17%	6.25%
	25.759	25.801	25.743	25.757	25.765
	0.72%	1.89%	3.68%	4.45%	2.68%
PW1-NEH (x)	20%	19.17%	19.17%	11.67%	17.5%
	25.946	25.860	25.840	25.839	25.871
	0.68%	0.69%	0.83%	1.08%	0.82%

*PS; **CPU time (sec); ***MRD Makespan

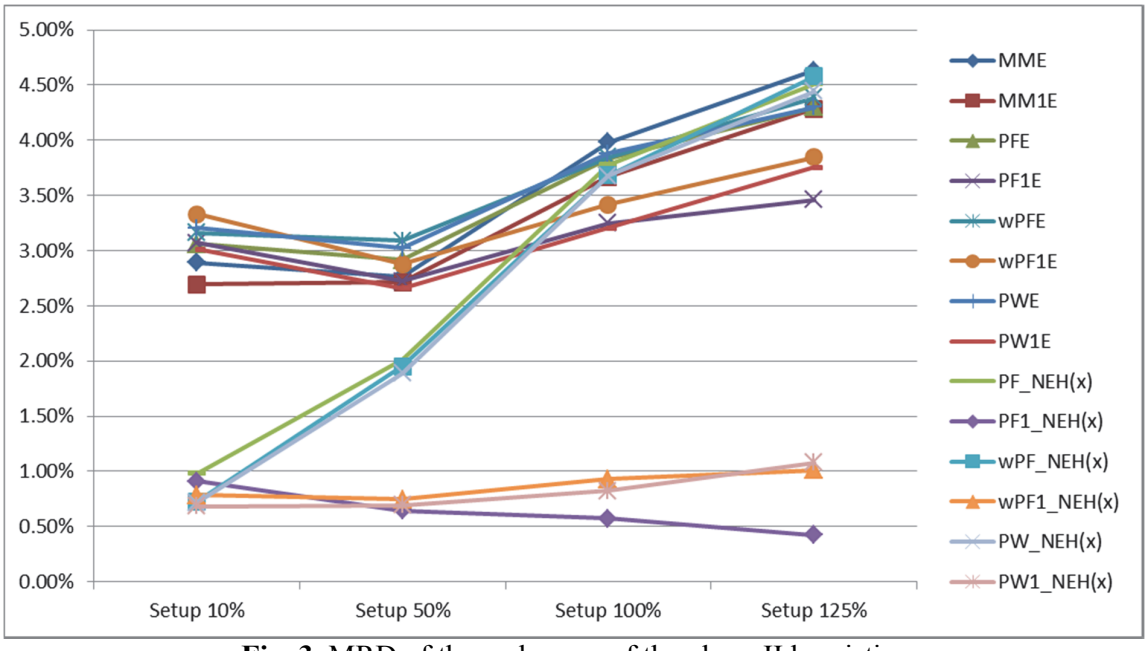

Fig. 3. MRD of the makespan of the phase II heuristics

Table 3
Performance of phase III heuristics

Setup time	10%	50%	100%	125%	Mean
PF-NEH$_{ls}$ (x)	22.5%*	10%	7.5%	5%	11.25%
	2295.5**	2746.7	3012.3	3336.2	2847.7
	0.44%***	1.19%	2.46%	2.90%	1.75%
PF1-NEH$_{ls}$ (x)	34.17%	53.33%	54.17%	62.5%	51.04%
	2206.9	2073.8	1706.0	1586.2	1893.2
	0.38%	0.36%	0.42%	0.35%	0.38%
wPF-NEH$_{ls}$ (x)	12.5%	8.33%	7.5%	4.17%	8.13%
	2112.3	2520.4	3144.4	3373.0	2787.5
	0.57%	1.41%	2.43%	3.01%	1.85%
wPF1-NEH$_{ls}$ (x)	14.17%	15%	17.5%	10%	14.17%
	2147.7	2133.1	2073.6	2163.7	2129.5
	0.52%	0.64%	0.72%	1.02%	0.73%
PW-NEH$_{ls}$ (x)	19.17%	6.67%	3.33%	8.33%	9.38%
	2239.6	2619.3	3406.1	3419.2	2921.0
	0.51%	1.36%	2.52%	2.91%	1.83%
PW1-NEH$_{ls}$ (x)	10%	15%	17.5%	14.17%	14.17%
	2280.2	2159.2	2096.9	1989.5	2131.5
	0.58%	0.63%	0.73%	1.00%	0.73%

*PS; **CPU time (sec); ***MRD Makespan

It can be observed in Tables 1, 2 and 3 that all adaptation methods that considered the setup times (those methods whose names are followed by the number 1) outperformed those methods that did not consider them. As can be seen from Table 1, PF1 obtained the best results in most of the problems among all I Phase heuristics, however PW1 achieved the best mean relative deviation of the makespan with an overall mean MRD for the makespan equal to 1.24% compared to 1.49% obtained by the second best heuristic PF1. PF1 obtained the best mean computational time with an overall mean computational time of 0.029637 seconds compared to the overall mean computational time of 0.184657 seconds obtained by PW1. PW1 obtained an average of 0.25% better results in the MRD of the makespan than PF1; however

PF1 was on average 83.851% faster at solving the problems and achieved the best result for 18.33% more problems than PW1. Fig. 2 shows that PF1 provides better results as the range of the setup time increases.

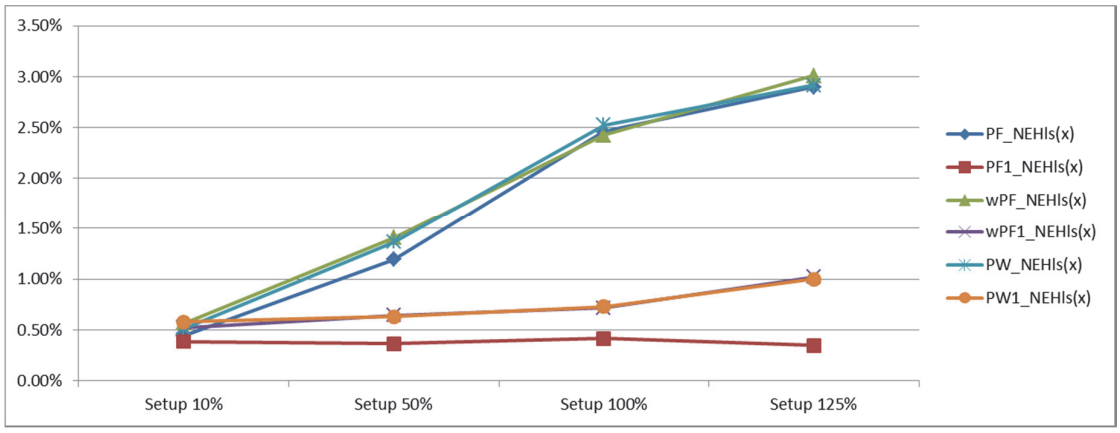

Fig. 4. MRD of the makespan of phase III heuristics

PF1-NEH (x) and PF-NEH(x) obtained the best mean relative deviation of the makespan and the minimum computational time among all II phase heuristics, respectively (Table 2). PF-NEH(x) was on average 4.02% faster than PF1-NEH (x). Meanwhile PF1-NEH (x) had an average of 2.18% better results in the MRD of the makespan and obtained the best result for 37.09% more problems than PF-NEH(x). Fig. 3 shows that PF1-NEH (x) provides better results as the range of the setup time increases.

PF1-NEH$_{ls}$(x) obtained the best results for more problems and also obtained the best mean relative deviation of the makespan and the minimum computational time among all III phase heuristics, observed in Table 3. Fig. 4 shows that the difference between the MRD of the makespan of PF1-NEH$_{ls}$(x) and the other heuristics increases as the range of the setup time increases.

5. Conclusion

This paper considered a permutation flow shop problem with blocking and sequence and machine dependent setup times in all machines. Only two studies have considered the blocking and sequence dependent setup times, however neither of them considers the sequence and machine dependent setup time and the zero buffer constraints in the same problem, therefore there are no former methods to solve this problem. The 14 best known heuristics for the permutation flow shop problem with blocking and no setup time were adapted in two different ways for the problem, each providing different results. Each of the adapted heuristics was compared using a 480-problem database that varied in the number of jobs and machines, and in the setup value. All heuristics provided unique results that were all tested with respect to their factibility.

The computational time to solve the problems does not vary very much from one method to another in the same phase. Therefore, the computational time was not used to compare the performance of the methods. The results presented by previous studies show that the PW algorithm outperforms the PF heuristic in most of the cases. However, by the results presented in this paper, the constructive heuristic PF1, combined or not with other methods, can be considered the heuristic with the best results for the problems. Possibly the variation of the setup time depending on the sequence on each machine makes it too difficult to predict the effects of the remaining jobs on the idle and blocking times. This can be noted by the increasing difference between the results from PF1 and PW1 based heuristics as the range of the setup time value increases. By analysing the percentage of successes of the methods, it is clear that PF1 becomes more advantageous as the range of the setup time value increases.

Concerning future work, we propose to develop a meta-heuristic goal for the problem using one of the constructive heuristics presented to provide an initial solution. In addition, a comparison of the constructive heuristics presented with an exact method, such as an MILP model or a branch-and-bound algorithm, can be performed to estimate the efficiency of the heuristics presented, since there are no previous results for problems to compare with the results obtained by the proposed heuristics.

Acknowledgements

The authors acknowledge the partial research support from the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq) – Brazil (projects 448161/2014-1, 308047/2014-1, 306075/2017-2).

References

Fernandez-Viagas, V., Leisten, R., & Framinan, J. M. (2016). A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. *Expert Systems with Applications, 61*, 290-301.

Framinan, J. M., Gupta, J. N. D., & 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.

Gupta, J. N. D., & Darrow, W. P. (1986). The two-machine sequence dependent flowshop scheduling problem. *European Journal of Operational Research*, *24*(3), 439-446.

Han, Y.-Y., Gong, D., & Sun, X. (2015). A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. *Engineering Optimization*, *47*(7), 927-946.

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

Maleki-Darounkolaei, A., Modiri, M., Tavakkoli-Moghaddam, R., & Seyyedi, I. (2012). A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *Journal of Industrial Engineering International*, *8*(1), 1-7.

Mccormick, S. T., Pinedo, M. L., Shenker, S., & Wolf, B. (1989). Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. *Operations Research*, *37*(6), 925 - 935.

Nagano, M. S., Komesu, A. S., & Miyata, H. H. (2017). An evolutionary clustering search for the total tardiness blocking flow shop problem. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-017-1358-7.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

Norman, B. A. (1999). Scheduling flowshops with finite buffers and sequence-dependent setup times. *Computer & Industrial Engineering*, *16*(1), 163-177.

Pan, Q. -K., & Wang, L. (2012). Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. *Omega*, *40*(2), 218-229.

Pan, Q. -K., Wang, L., Sang, H. -Y, Li, J. -Q., & Liu, M. (2013). A High Performing Memetic Algorithm for the Flowshop Scheduling Problem With Blocking. *IEEE Transactions on Automation Science and Engineering*, *10*(3), 741-756.

Ribas, I., & Companys, R. (2015). Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. *Computers & Industrial Engineering, 87*, 30-39.

Ribas, I., Companys, R., & Tort-Martorell, X. (2013). An efficient iterated local search algorithm for the total tardiness blocking flow shop problem. *International Journal of Production Research*, *51*(17), 5238–5252.

Ronconi, D. P. (2004). A note on constructive heuristics for the flowshop problem with blocking. *International Journal of Production Economics*, *87*(1), 39-48.

Sanches, F. B., Takano, M. I., & Nagano, M. S. (2017). Evaluation of heuristics for a branch and bound algorithm to minimize the makespan in a flowshop with blocking. *Acta Scientiarum Technology*, *38*(3), 321-326.

Shao, Z., Pi, D., & Shao, W. (2017). Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. *Computers & Industrial Engineering, 111*, 331-351.

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

Takano, M. I., & Nagano, M. S. (2017). A branch-and-bound method to minimize the makespan in a permutation flow shop with blocking and setup times. *Congent Engineering*, *4*(1), 1389638.

Tasgetiren, M. F., Pan, Q. K., Kizilay, D., & Gao, K. (2016). A variable block insertion heuristic for the blocking flowshop scheduling problem with total flowtime criterion. *Algorithms, 9*(4), 71.

Tasgetiren, M. F., Kizilay, D., Pan, Q. K., & Suganthan, P. N. (2017). Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. *Computers & Operations Research, 77*, 111-126.