Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem

Daniella Castro Araújo[a] and Marcelo Seido Nagano[a*]

[a]*Department of Industrial Engineering, School of São Carlos, University of São Paulo, Brazil*

A R T I C L E I N F O

Article history:
Received 15 June 2010
Received in revised form
31 August 2010
Accepted 1 September 2010
Available online
1 September 2010

Keywords:
Scheduling
Heuristic
No-wait flowshop
Sequence-dependent setup
Makespan

A B S T R A C T

In this paper, we address the problem of scheduling jobs in a no-wait flowshop problem with sequence-dependent setup times with the objective of minimizing makespan. This problem is well-known for being nondeterministic polynomial-time hard, and small contribution to the problem has been made. We propose a new constructive heuristic named GAPH based on a structural property. The effectiveness of the structural property is crucial given that it is responsible for 100% of the success rate of the total problems tested. The computational results demonstrate that the proposed approach is superior than three of the best-know methods in the literature such as the twos by Bianco, Dell'Olmo and Giordani (*INFOR Journal*, 37 (1), 3-19, 1999) and TRIPS heuristic adapted for sequence-dependent setup times objective by Brown, Mcgarvey and Ventura (*Journal of the Operational Research Society*, 55 (6), 614-621, 2004) in terms of the solution quality and that it requires less computational effort.

1. Introduction

The first systematic approach to scheduling problems was undertaken in the mid-1950s. Since then, thousands of papers on different scheduling problems have appeared in the literature. The majority of these papers assumed that the setup time is negligible or part of the job processing time. Treating setup times separately from processing times allows operations to be performed simultaneously and hence improves resource utilization. This is, in particular, important in modern production management systems such as just-in-time (JIT), optimized production technology (OPT), group technology (GT), cellular manufacturing (CM), and time-based competition (ALLAHVERDI et al., 2008). Another important area in scheduling arises in no-wait flowshop problems (NWFSP), where jobs have to be processed without interruption between consecutive machines. There are several industries where the no-wait flowshop problem applies including the metal, plastic, and chemical industries. As noted by Hall and Sriskandarajah (1996), the first of two main reasons for the occurrence of a no-wait or blocking production environment lies in the production technology itself. In some processes, for example, the temperature or other characteristics (such as viscosity) of the material require that each operation follow the previous one immediately. According to Bianco et al.

* Corresponding author. Tel: +55 16 3373-9428, Fax: +55 16 3373-9425
E-mail addresses: drnagano@usp.br (M. S. Nagano),

doi: 10.5267/j.ijiec.2010.05.003

(1999), flowshop no-wait scheduling problems are also motivated by concepts such as JIT and zero inventory in modern manufacturing systems.

A survey on NWFSP has been conducted by Hall and Sriskandarajah (1996), where several practical applications are shown. Allahverdi et al. (1999, 2008) provided a comprehensive review of the literature on scheduling problems with setup times. The NWFSP with sequence dependent setup times and with the objective of minimizing makespan was first proposed by Bianco et al. (1999). They showed how to reduce this problem to the asymmetric travelling salesman problem (ATSP) and presented two lower bounds and two heuristics, named BAH and BIH. The computational results showed that BIH outperformed BAH in the solutions quality. Kumar et al. (2000) considered a NWFSP that used lot-streaming to improve productivity. They developed a TSP formulation for the multi-product and continuous-sized case and proposed a heuristic to obtain an optimal sequence for integer-sized sublots. Allahverdi and Aldowaisan (2000) found optimal solutions for the $F3/ ST_{si}, no-wait/\sum C_j$ problem, where the setup and processing times satisfy certain conditions, and presented five heuristics for the general problem. Later, Allahverdi and Aldowaisan (2001) considered the $F2/ST_{sd}, no-wait/\sum C_j$ problem and presented five heuristics that used a repeated insertion technique. Stafford and Tseng (1990, 2002) proposed two mixed-integer linear programming (MILP) models to solve the *m*-machine NWFSP with sequence dependent setup times in order to minimize the makespan. Aldowaisan and Allahverdi (2003) proposed six heuristics based on simulated annealing and genetic algorithms techniques for the $F_m/no-wait/C_{max}$ problem. The simulated annealing based heuristics performed better than the others. Fink and Voβ (2003) proposed three constructive heuristics and several meta-heuristics for the NWFSP with total flowtime as the criteria. Shyu et al. (2004) presented an ant colony optimization algorithm for the $F2/ST_{si}, no-wait/\sum C_j$ problem, and showed that their algorithm outperformed earlier heuristics. Brown et al. (2004) presented a non-polynomial time solution method and a heuristic named TRIPS for the NWFSP with sequence independent setup times, considering for the performance measures both the total flowtime and makespan. Ruiz et al. (2005) addressed the $F_m/ST_{sd}/C_{max}$ problem and proposed two genetic algorithms, named GA and HGA. França et al. (2006) considered the same problem as Bianco et al. (1999) and solved it by an evolutionary approach. Their genetic algorithm outperformed BIH. Ruiz and Allahverdi (2007a) presented a domination relation for the $F4/ST_{si}, no-wait/\sum C_j$ problem and proposed an iterated local search method and five heuristics for the same problem with *m*-machines. The results showed that three of their heuristics outperformed TRIPS and the ant colony algorithm of Shyu et al. (2004). Ruiz and Allahverdi (2007b) proposed seven heuristics and four genetic algorithms for the NWFSP with sequence independent setup times in order to minimize the maximum lateness. Their genetic algorithms outperformed the heuristics of Ruiz and Allahverdi (2007a). Grabowski and Pempera (2007) developed and compared five heuristcs for the $F_m/no-wait/C_{max}$ problem. In order to decrease the computational effort, they used multimoves. Ruiz and Stüzle (2008) presented two simple local search based iterated greedy algorithms for both $F_m/ST_{sd}/\sum w_j T_j$ and $F_m/ST_{sd}/C_{max}$ problems, and showed that their algorithms performed better than GA and HGA. Framinan and Nagano (2008) studied the $F_m/no-wait/C_{max}$ problem and proposed a heuristic based on an analogy between the problem under consideration and the travelling salesman problem (TSP). Pan et al. (2008) presented a discrete particle swarm optimization (DPSO) to solve the NWFSP with both makespan and total flowtime criteria. The results showed that the algorithm outperformed the heuristics of Grabowski and Pempera (2007) and Fink and Voβ (2003). Yaurima et al. (2009) proposed a genetic algorithm for the hybrid flowshop problem with unrelated machines, sequence dependent setup times, availability constraints and limited buffer, and introduced a crossover operator and stopping criterion to improve the solution quality. Eren (2010) proposed an integer programming model to solve the flowshop problem with sequence dependent setup times. The objective function was the weighted sum of total completion time and the makespan. The model could solve problems up to 6 machines and 18 jobs. Wang et al. (2010) considered the NWFSP with maximum lateness criterion and developed properties to reduce the time to evaluate a candidate in a

tabu search approach. Framinan et al. (2010) addressed the $F_m/no-wait/\sum C_j$ problem and proposed a constructive heuristic based on an analogy with the two-machine problem. The computational results showed that the heuristic outperformed existing ones regarding the solution quality.

In this paper, we consider the problem of scheduling a no-wait flowshop problem with sequence dependent setup times ($F_m/ST_{sd}, no-wait/C_{max}$), which consists of a set J $= \{j_1, j_2, j_3, \dots, j_n\}$ of n jobs to be processed on a set $M = \{m_1, m_2, m_3, \dots, m_m\}$ of m dedicated machines, each one being able to process only one job at a time. Job j_i consists of m operations $op_{1i}, \dots, op_{ki}, op_{k+1i}, \dots, op_{mi}$, to be executed in this order, where operation op_{ki} must be executed on machine k, with p_{ki} processing time, immediately before operation op_{k+1i}. There is a sequence dependent setup time s_{ij}^k between operations op_{ki} and op_{kj} in machine k. In addition, we propose a new heuristic method for the problem, which outperforms the existing heuristics. The new heuristic is based on a property of the scheduling problem that provides the time break between the beginning of job $j_{[i+1]}$ and the beginning of job $j_{[i]}$ at machine k, where, $j_{[i]}$ is the job of J occupying position i in σ. This paper is organized as follows. In Sections 2 and 3, we describe the set of constructive heuristics available for the problem and present a property concerning this scheduling, which is used for the development of the new heuristic proposed. In Section 4, we test the new heuristic effectiveness. Finally, conclusions and final considerations are given in Section 5.

2. Existing constructive heuristics for the problem

In this section, we review the main contributions to the problem regarding constructive methods. More specifically, we explain in detail the constructive heuristics BAH and BIH, from Bianco et al. (1999), and TRIPS, from Brown et al. (2004).

2.1. BAH

BAH algorithm finds a feasible sequence in n iterations. At each iteration, given a partial sequence of the scheduled jobs computed in the previous iteration, the algorithm examines a set of candidates of the unscheduled jobs, and appends a candidate job to a partial sequence minimizing the time when the shop is ready to process an unscheduled job.

The pseudo-code of the heuristic is as follows:

Given a set J $= \{j_1, j_2, j_3, \dots, j_n\}$ of n jobs, let σ be the set of programmed jobs and U be the set of non-programmed jobs.

Step 1: $U \leftarrow J$; $\sigma \leftarrow \emptyset$;

Step 2: While $U \neq \emptyset$, do:

Step 2.1: Choose the job $j_i \in U$ to be added at the end of the sequence σ , such that the makespan is minimum;

Step 2.2: Add job j_i to the end of the sequence σ;

Step 2.3: $U \leftarrow U\text{-}j_i$.

2.2. BIH

The BIH algorithm also finds a sequence of n jobs on n iterations. But in this algorithm, at each iteration it considers a sequence of a subset of jobs, and finds the best sequence obtained inserting an unscheduled job in any position of the given sequence.

A more detailed description of the heuristic is as follows:

Given a set J = $\{j_1, j_2, j_3, \ldots, j_n\}$ of *n* jobs, let σ be the set of programmed jobs, *U* be the set of non-programmed jobs and *h* the relative insertion position.

Step 1: $U \leftarrow J$; $\sigma \leftarrow \emptyset$;

Step 2: While $U \neq \emptyset$, do:

Step 2.1: Choose the job $j_i \in U$ which can be inserted in the sequence σ, such that the makespan is minimum. Let *h* be the relative insertion position;

Step 2.2: Insert job j_i at position *h* in the sequence σ;

Step 2.3: $U \leftarrow U\text{-}j_i$.

2.3. TRIPS

TRIPS heuristic was developed for the no-wait flowshop with sequence-independent setup times, for minimizing total flowtime ($F_m/ST_{si}/\sum C_j$) or makespan ($F_m/ST_{si}/C_{max}$). In this paper, because there are only BIH and BAH constructive heuristics for the $F_m/ST_{sd}/C_{max}$ problem, we will adapt it to this problem.

TRIPS examines all possible three-job combinations from the set of unscheduled jobs *U* and chooses the sequence $\{j_w, j_x, j_y\}$ that minimizes the three-job objective. Then, assigns job j_w to the last empty position in the sequence σ and removes j_w from *U*. The heuristic repeats the process, assigning one more job to σ for each set of triplets examined until only three jobs are left. Then, it selects the optimal sequence for these jobs and places them in the final positions of heuristic sequence σ.

The pseudo-code of the heuristic is as follows:

Given a set J = $\{j_1, j_2, j_3, \ldots, j_n\}$ of *n* jobs, let σ be the set of programmed jobs and *U* be the set of non-programmed jobs.

Step 1: $U \leftarrow J$; $\sigma \leftarrow \emptyset$; $h \leftarrow 0$;

Step 2: While $h<n-2$, do:

Step 2.1: Given that the first *h* jobs are assigned in sequence σ, compare all ordered triplets of jobs from *U*;

Step 2.2: Choose the tripplet $\{j_w, j_x, j_y\}$ such that the performance measure (either makespan or flowtime) is minimized for jobs $\{j_w, j_x, j_y\}$ in positions *h+1*, *h+2*, *h+3*, respectively, of sequence σ.

Step 2.3: Place j_w in position *h+1* of σ;

Step 2.4: $h \leftarrow h+1$; $U \leftarrow U\text{-}j_w$;

Step 3: Assign j_x and j_y to the last two positions, respectively, of σ.

3. A useful structural property for the new heuristic

Given a sequence σ of J, $j_{[i]}$ is the job of J occupying position *i* in σ. The time break between the beginning of job $j_{[i+1]}$ and the beginning of job $j_{[i]}$ at machine *k* is $\Delta t^k_{[i][i+1]}$, calculated as follows:

$$\Delta t^{1}_{[i][i+1]} = \max_{1\le k\le m}[s^{k}_{[i][i+1]} + \sum_{h=1}^{k}(p_{h[i]} - p_{h[i+1]}) + p_{k[i+1]}, \quad (1)$$

$$\Delta t^{k}_{[i][i+1]} = \Delta t^{1}_{[i][i+1]} + \sum_{h=1}^{k-1}(p_{h[i+1]} - p_{h[i]}). \quad (2)$$

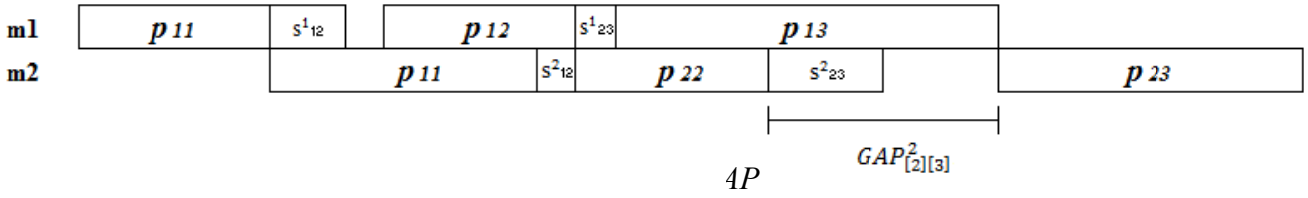
Defining $GAP^{k}_{[i][i+1]}$ as the time break between the end of job $j_{[i]}$ and the beginning of job $j_{[i+1]}$ at machine k, it can be calculated as follows:

$$GAP^{k}_{[i][i+1]} = \Delta t^{k}_{[i][i+1]} - p_{k[i]}. \quad (3)$$

The *GAP* of the first job in the sequence on machine k is defined by as follows,

$$GAP^{k}_{[0][1]} = \sum_{h=1}^{k-1} p_{h[1]}. \quad (4)$$

Fig. 1 shows the time break between the end of job j_2 and the beginning of job j_3 on machine 2 ($GAP^{2}_{[2][3]}$).



3.1. The new heuristic

The new heuristic proposed in this paper will be called GAPH – Gap Heuristic. The pseudo-code of the algorithm is given next:

Given a set $J = \{j_1, j_2, j_3, \dots, j_n\}$ of n jobs, let U be the set of non-programmed jobs and $\sigma_x = (j_{[1]}, j_{[2]}, \dots, j_{[n]})$ be the sequence of n jobs scheduled, where $x = \{1,2,3,4\}$. Calculate the $GAP^{k}_{[i][j]}$ of each job $i = 1, ..., n$ to each job $j = 1, ..., n$ at all m machines

Step 1: $U \leftarrow J$; $\sigma_1 \leftarrow \emptyset$;

Step 2: While $U \neq \emptyset$, do:

Step 2.1: Calculate the *total cost** on the last machine for all possible insertions of each job $j_i \in U$ in the sequence σ_1. Let h be the relative insertion position;

Step 2.2: Choose the job j_i that gives the lower *total cost* at position h;

Step 2.3: Insert job j_i at position h of the sequence σ_1;

Step 2.4: $U \leftarrow U\text{-}j_i$;

Step 3: $U \leftarrow J$; $\sigma_2 \leftarrow \emptyset$

Step 4: While $U \neq \emptyset$, do:

Step 4.1: Calculate the *total GAP*** for all possible insertions of each job $j_i \in U$ in the sequence σ_2. Let *h* be the relative insertion position;

Step 4.2: Choose the job j_i that gives the lower *total GAP* at position *h*;

Step 4.3: Insert job j_i at position *h* of the sequence σ_2;

Step 4.4: $U \leftarrow U\text{-}j_i$;

Step 5: $U \leftarrow J$; $\sigma_3 \leftarrow \emptyset$;

Step 6: While $U \neq \emptyset$, do:

Step 6.1: Calculate the sum of the *GAPs* on the last machine for all possible insertions of each job $j_i \in U$ in the sequence σ_3. Let *h* be the relative insertion position;

Step 6.2: Choose the job j_i that gives the lower sum of the *GAPs* on the last machine at position *h*;

Step 6.3: Insert job j_i at position *h* of the sequence σ_3;

Step 6.4: $U \leftarrow U\text{-}j_i$;

Step 7: $U \leftarrow J$; $\sigma_4 \leftarrow \emptyset$;

Step 8: While $U \neq \emptyset$, do:

Step 8.1: Calculate, for all possible insertions of each job $j_i \in U$ in the sequence σ_4, the sum of the *GAPs* with the processing time of the job on the last machine, Let *h* be the relative insertion position;

Step 8.2: Choose the job j_i that gives the lower sum of the *GAPs* with the processing time of the job on the last machine at position *h;*

Step 8.3: Insert job j_i at position *h* of the sequence σ_4

Step 8.4: $U \leftarrow U\text{-}j_i$;

Step 9: Choose, among the sequences $\{\sigma_1,\sigma_2,\sigma_3,\sigma_4\}$, the one with the lower makespan

*The *total cost* on a *k* machine is defined as the scheduling total time on this machine. Thus, the *total cost* encompasses the sum of the *GAP*s on machine *k* with the scheduled operations processing times on that machine. Note that the *total cost* on the last machine is equivalent to the makespan (see Fig. 2).

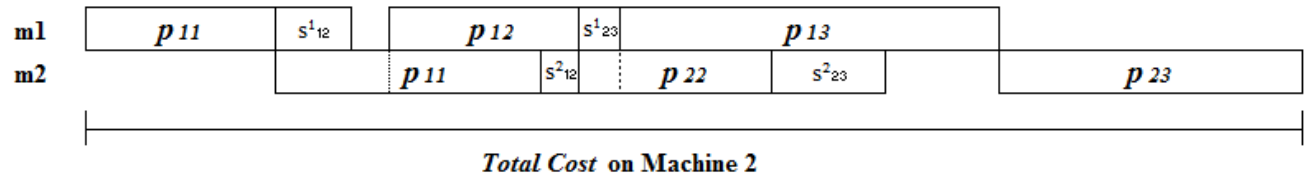


Fig. 2. Example of the total cost

**The *total GAP* is the sum of all *GAP*s in all machines.

In Figure 3, the *total GAP* is: $GAP^1_{[1][2]} + GAP^1_{[2][3]} + GAP^2_{[0][1]} + GAP^2_{[1][2]} + GAP^2_{[2][3]}$.

The sum of the *GAPs* on the last machine is: $GAP^2_{[0][1]} + GAP^2_{[1][2]} + GAP^2_{[2][3]}$.

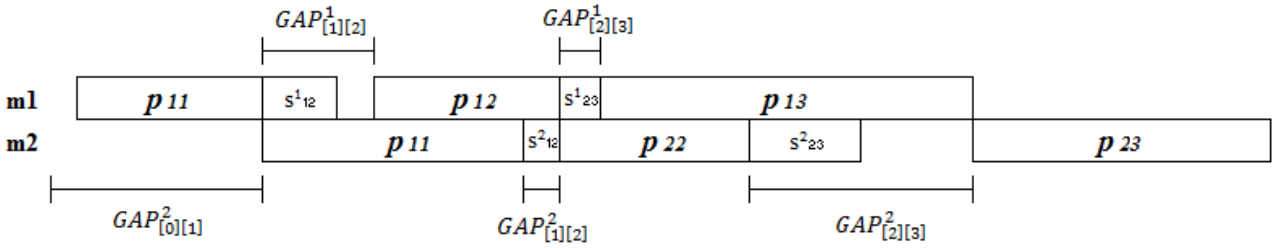


Fig. 3. Example of all the *GAPs* in the scheduling

The sum of the *GAPs* with the processing time of the job to be inserted on the last machine is: $GAP^2_{[0][1]} + GAP^2_{[1][2]} + GAP^2_{[2][3]} + p_{23}$. Fig. 4 shows how the method works for each pair of steps (1-2; 3-4; 5-6; 7-8). In the example, jobs j_1 and j_2 were already scheduled ($\sigma = (j_1, j_2)$) and job j_3 is scheduled on each possible position of the sequence (h). For example, if the method were on the second step, the sequence chosen would be the third one, that gives the lower *total cost* on the last machine.

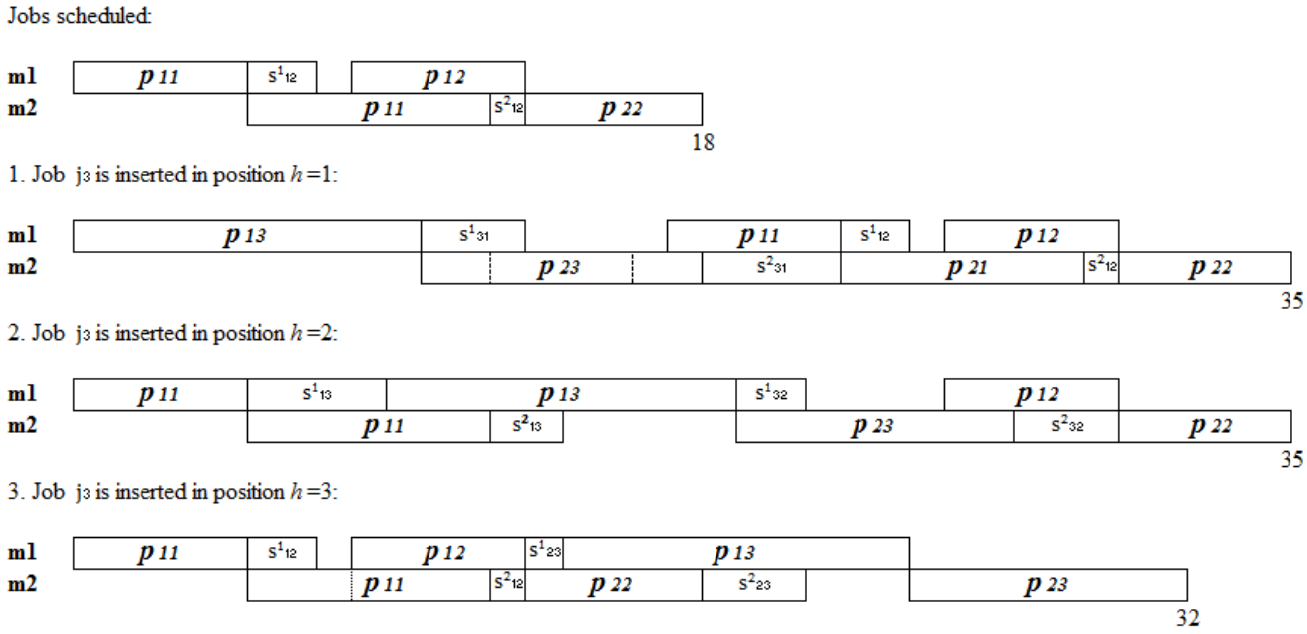


Fig. 4. Numerical example of how the method works

4. Computational experience

We carried out an extensive computational experiment in order to test GAPH, as well as BIH, BAH (BIANCO et al., 1999) and TRIPS (BROWN et al., 2004) heuristics. The heuristics were tested in the well-known testbed of Taillard (1993). This testbed contains twelve sets for a given combination of jobs and machines, i.e., n {20,50,100,200,500} and m {5,10,20}. We performed four experiments, one for each of the four different sequence-dependent Taillard-based instance sets from Ruiz et al. (2005). The tests contain four different processing times to sequence-dependent setup times ratios. For example, the instance set SSD-10 is composed of 120 instances where the processing times are those of Taillard's benchmark and where the sequence-dependent setup times are 10% of the processing times. In the instance set SSD-50, the setup times are 50% of the processing times and the instance sets SSD-100 and SSD-125 have setup times that are 100% and 125% of the processing times, respectively.

Table 1
Comparison of results in Taillard´s testbed SSD-10 and SSD-50

	SSD-10				SSD-50			
n x m	BAH	BIH	TRIPS	GAPH	BAH	BIH	TRIPS	GAPH
20x5	0.00*	60	0	100	0	60	0	100
	15.54**	1.83	9.27	0	13.13	1.27	7.23	0
	0.03***	0.11	0.34	0.27	0.03	0.11	0.34	0.27
20x10	0	50	0	100	0	70	0	100
	15.87	2.48	8.23	0	14.21	1.59	8.09	0
	0.06	0.15	0.37	0.31	0.06	0.15	0.36	0.31
20x20	0	60	0	100	0	70	0	100
	13.42	0.42	8.82	0	13.21	0.88	8.37	0
	0.18	0.26	0.43	0.43	0.17	0.26	0.42	0.43
Average	0	56.67	0	100	0	66.67	0	100
	14.94	1.58	8.77	0	13.51	1.25	7.9	0
	0.09	0.17	0.38	0.34	0.09	0.17	0.37	0.34
50x5	0	70	0	100	0	50	0	100
	11.74	0.41	5.77	0	9.8	0.98	5.44	0
	0.17	2.87	14.3	7.92	0.16	2.89	14.11	7.96
50x10	0	90	0	100	0	80	0	100
	13.42	0.08	6.67	0	11.31	0.08	5.57	0
	0.4	3.09	14.45	8.3	0.4	3.14	14.22	8.35
50x20	0	80	0	100	0	90	0	100
	13.92	0.1	8.41	0	12.22	0.03	7.6	0
	1.16	3.84	14.85	9.1	1.17	3.88	14.66	9.15
Average	0	80	0	100	0	73.33	0	100
	13.03	0.2	6.95	0	11.11	0.37	6.2	0
	0.58	3.27	14.53	8.44	0.57	3.3	14.33	8.49
100x5	0	70	0	100	0	60	0	100
	10.11	0.73	6.08	0	8.27	0.36	5.32	0
	0.83	40.7	226.25	119.23	0.67	40.75	225.78	121.01
100x10	0	40	0	100	0	30	0	100
	11.16	0.85	7.07	0	8.68	0.43	6.68	0
	1.67	39.97	226.09	118.98	1.69	40.17	226.26	118.94
100x20	0	60	0	100	0	80	0	100
	11.42	0.26	7.54	0	9.81	0.14	6.74	0
	5.6	43.73	225.7	122.87	5.07	43.82	226.89	123.65
Average	0	56.67	0	100	0	56.67	0	100
	10.89	0.61	6.9	0	8.92	0.31	6.25	0
	2.7	41.47	226.01	120.36	2.48	41.58	226.31	121.2
200x10	0	50	0	100	0	80	0	100
	8.11	0.33	6.08	0	6.68	0.12	4.58	0
	6.88	611.5	3799.28	1837.43	6.81	612.89	3759.58	1839.95
200x20	0	70	0	100	0	80	0	100
	8.68	0.11	5.52	0	7.43	0.02	4.4	0
	21.63	626.84	3681.3	1834.39	20.81	629.81	3653.72	1843.07
Average	0	60.00	0.00	100.00	0.00	80.00	80.00	100.00
	8.39	0.22	5.80	0.00	7.05	0.07	0.07	0.00
	14,25	614.02	3740.29	1835.91	13.81	613.57	613.57	1841.45

* Success Rate (%);** ARPD (%);*** Average CPU time (second).

So for example, if the processing times in Taillard's instances are generated from a uniform distribution in the range [1; 99], in the SSD-10 instance set the setup times are uniformly distributed in the range [1; 9] ([1; 49], [1; 99] and [1; 124] for the instance sets SSD-50, SSD-100 and SSD-125 respectively). Thus, we have four problem sets and a total of 480 different instances. The 500 job instance was rather large and we chose to solve only the first 110 instances (up to 200 jobs and 20 machines). The instances in the testbed have been solved by the selected heuristics (coded in Python) in a computer with a Pentium IV 3.00GHz processor and 512MB RAM. Tables 1 and 2 summarize the result obtained for the different heuristics in terms of the success percentage, the average relative percentage deviation (ARPD), and the average CPU time. The success rate is defined by the ratio between the number of problems for which a particular method was the best solution and the total number of problems solved. Therefore, when two methods get the best solution for the same problem, their percentages of success are both improved. The ARPD consists of averaging the RPD over a number of instances with the same number of jobs. We have grouped the results for a given number of jobs and different machines, as the number of machines had almost no influence in the results. For a given objective function f, the RPD obtained by a heuristic H on a given instance is computed as follows:

$$RPD\ (H) = \frac{f(H) - f^*}{f^*} \cdot 100, \tag{5}$$

where $f(H)$ is the makespan computed by method h and f^* is the best makespan computed by the methods. According to Tables 1 and 2, the proposed heuristic obtains better results than the rest of the constructive heuristics. Over all configurations, the maximal ARPD from the best solution found was 0.14% for GAPH (when TRIPS found the best solution) and 2.48% for BIH. The maximal ARPD for BAH was 15.87% and 9.27% for TRIPS, respectively. All success rates for BAH were zero, and TRIPS only got the best solution once which means that BAH and TRIPS are not competitive with the other heuristics tested. Another point is that the methods seem to be unaffected by distribution of processing or setup times, i.e., there are no better methods depending on the specific distribution of processing or setup times. Comparing GAPH with BIH, we observe that GAPH always gets equal or better results than BIH. The minimum success rates of GAPH and BIH were 90% and 20%, respectively.

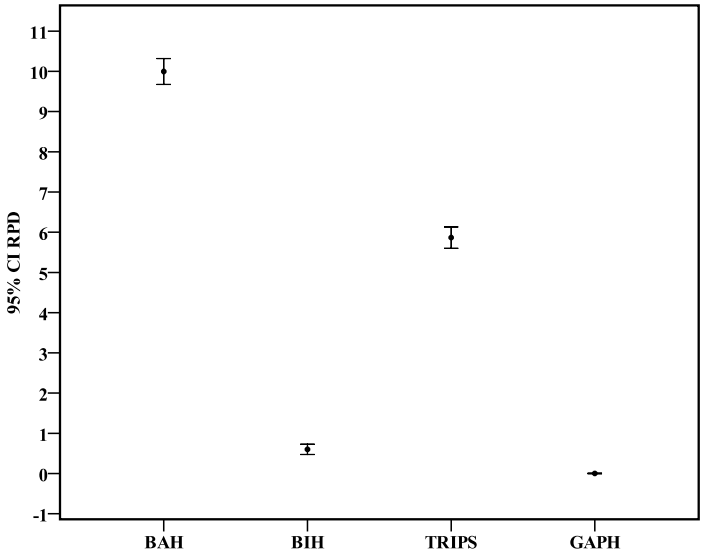


Fig. 5. Means and 95% confidence intervals for the different algorithms

Table 2
Comparison of results in Taillard´s testbed SSD-100 and SSD-125

	SSD-100				SSD-125			
n x m	BAH	BIH	TRIPS	GAPH	BAH	BIH	TRIPS	GAPH
20x5	0	20	0	100	0	60	10	90
	9.95	1.08	5.71	0	10.54	0.85	5.08	0.14
	0.02	0.11	0.34	0.27	0.02	0.11	0.33	0.27
20x10	0	50	0	100	0	50	0	100
	12.2	1.09	6.82	0	11.46	1.03	5.62	0
	0.07	0.14	0.36	0.31	0.06	0.15	0.36	0.31
20x20	0	60	0	100	0	50	0	100
	12.15	0.35	7.57	0	11.85	0.93	6.63	0
	0.18	0.26	0.42	0.43	0.18	0.26	0.42	0.44
Average	0	43.33	0	100	0	53.33	3.33	96.67
	11.44	0.84	6.7	0	11.28	0.94	5.78	0.05
	0.09	0.17	0.37	0.34	0.09	0.17	0.37	0.34
50x5	0	50	0	100	0	30	0	100
	9.51	0.8	4.38	0	9.21	1.32	5.7	0
	0.17	2.88	14.11	8.01	0.17	2.88	14.11	8.02
50x10	0	70	0	100	0	60	0	100
	8.98	0.13	4.01	0	9.97	0.93	4.99	0
	0.39	3.12	14.27	8.37	0.4	3.11	14.3	8.34
50x20	0	80	0	100	0	90	0	100
	9.95	0.04	5.53	0	9.07	0.01	5.44	0
	1.19	3.85	14.71	9.13	1.16	3.85	14.74	9.13
Average	0	66.67	0	100	0	60	0	100
	9.48	0.32	4.64	0	9.41	0.75	5.38	0
	0.58	3.28	14.36	8.5	0.58	3.28	14.38	8.5
100x5	0	30	0	100	0	20	0	100
	7.32	0.71	4.4	0	7.14	1.02	4.44	0
	0.68	41.16	223.3	122.54	0.67	40.93	223.37	118.93
100x10	0	60	0	100	0	70	0	100
	7.38	0.17	4.88	0	7.05	0.24	3.81	0
	1.75	40.67	226.42	119.35	2.03	40.36	227.38	119.32
100x20	0	70	0	100	0	90	0	100
	8.05	0.37	5.37	0	6.82	0.08	4.47	0
	5.18	43.87	228.77	123.35	5.19	44.09	229.54	123.4
Average	0	53.33	0	100	0	60	0	100
	7.58	0.42	4.89	0	7	0.44	4.24	0
	2.54	41.9	226.16	121.75	2.63	41.8	226.76	120.55
200x10	0	40	0	100	0	30	0	100
	6.37	0.59	4.21	0	5.59	0.56	3.44	0
	6.87	613.21	3780.1	1853.66	7.03	618	3793.94	1836.08
200x20	0	60	0	100	0	60	0	100
	6.55	0.09	3.45	0	5.7	0.09	3.15	0
	21.19	632.35	3657.28	1846.33	21.13	633.6	3649.61	1844.22
Average	0.00	50.00	0.00	100.00	0.00	45.00	0.00	100.00
	6.46	0,34	3.83	0.00	5.64	0.32	3.29	0.00
	14.03	616.47	3718.69	1850.0	14.08	615.14	3721.78	1840.15

To observe the statistical significance of the differences between the heuristics, we plotted the means of each heuristic and the corresponding 95% confidence intervals in Fig. 5. A Tukey honestly significant difference test was conducted to determine which means differ (see Table 3).

Table 3
Results of Tukey honestly significant difference tests

Heuristic (I)	Heuristic (J)	Mean Difference (I-J)	Std. Error	Sig.
BAH	BIH	9.39593*	0.15777	0.000
	TRIPS	4.12916*	0.15777	0.000
	GAPH	9.99338*	0.15777	0.000
BIH	BAH	-9.39593*	0.15777	0.000
	TRIPS	-5.26678*	0.15777	0.000
	GAPH	0.59745*	0.15777	0.001
TRIPS	BAH	-4.12916*	0.15777	0.000
	BIH	5.26678*	0.15777	0.000
	GAPH	5.86422*	0.15777	0.000
GAPH	BAH	-9.99338*	0.15777	0.000
	BIH	-0.59745*	0.15777	0.001
	TRIPS	-5.86422*	0.15777	0.000

*Mean difference is significant at the 0.05 level (95%)

The results indicate that the differences between the proposed heuristic (GAPH) and the rest are statistically significant. The results of GAPH are significantly better than those of BAH, BIH and TRIPS with respect to the quality of the solutions. With respect to the CPU time, TRIPS require much more computational effort than GAPH and BIH. As it can be observed in Tables 1 and 2, for the biggest problem analyzed (200x20), the average CPU time of TRIPS was nearly 3660s, while TRIPS required nearly 630s and GAPH required nearly 1850s. Finally, our proposal heuristic is statistically better than the rest of the heuristics, although it is more time consuming than BIH. GAPH always gets the better result, and is more efficient than TRIPS.

5. Conclusion

In this paper, we dealt with the problem of scheduling a no-wait flowshop with sequence-dependent setup times with a makespan objective by means of constructive heuristics. We presented a new heuristic, named GAPH and carried out an extensive computational experiment. The results showed that GAPH gets better results than the other constructive heuristics tested. Henceforth, it can be concluded that the proposed heuristic obtains high solution quality comparing to the existing constructive heuristics for the problem, in acceptable computational times.

Acknowledgment

The authors wish to thank the referees for their comments on an earlier version of the paper. The research of the first author is partially supported by The State of São Paulo Research Foundation (FAPESP) under grant number 09/06832-2. The research of the second author is partially supported by a grant number 473654/2009-1 from the National Council for Scientific and Technological Development (CNPq), Brazil.

References

Aldowaisan, T. (2001). A new heuristic and dominance relations for no-wait flowshops with setups, *Computers & Operations Research, 28 (6), 563-584.*

Aldowaisan, T. & Allahverdi, A. (2003). New heuristics for no-wait flowshops to minimize make span, *Computers & Operations Research*, 30 (8), 1219-1231.

Allahverdi, A. & Aldowaisan, T. (2000). No-wait and separate setup three-machine flowshop with total completion time criterion, *International Transactions in Operational Research*,7 (3), 245-64.

Allahverdi, A. & Aldowaisan, T. (2001). Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. *Journal of the Operational Research Society*, 52 (4), 449-462.

Allahverdi, A., Gupta, J.N.D. & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega. The international Journal of Management Science*, 27 (2), 219-239.

Allahverdi, A., Ng, C.T., Cheng, T.C.E. & Kovalyov, M.Y. (2008). A survey of scheduling problems with setups times or costs. *European Journal of Operational Research*, 187 (3), 985-1032.

Allahverdi, A. & Soroush, H.M. (2008). The significance of reducing setup times/setup costs. *European Journal of Operational Research*, 187 (3), 978-984.

Bianco, L., Dell'Olmo, P. & Giordani, S. (1999). Flow shop no-wait scheduling with sequence-dependent setup times and release dates. *INFOR Journal*, 37 (1), 3-19.

Brown, S.I., Mcgarvey, R. & Ventura, J. A. (2004). Total flowtime and makespan for a no-wait m-machine flowshop with set-up times separated. *Journal of the Operational Research Society*, 55 (6), 614-621.

Eren, T. (2010). A bicriteria m-machine flowshop scheduling with sequence-dependent setup times. *Applied Mathematical Modelling*, 34 (2), 284-293.

Fink, A. & Voβ, S. (2003). Solving the continuos flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research,* 151 (2), 400-414.

Framinan, J. M. & Nagano, M. S. (2008). Evaluating the performance for makespan minimisation in no-wait flowshop sequencing, *Journal of Materials Processing Technology*, 197 (1-3), 1-9.

Framinan, J.M., Nagano, M.S. & Moccellin, J.V. (2010). An efficient heuristic for total flowtime minimization in no-wait flowshops. *International Journal of Advanced Manufacturing Technology*, 46 (9-12), 1049-1057.

França, P. M., Tin Jr, G. & Buriol, L. S. (2006). Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions. *International Journal of Production Research*, 44 (5), 939-957.

Grabowski, J., Pempera, J. (2007). The permutation flowshop problem with blocking. A tabu search approach, Omega*. The International Journal of Management Science*, 35 (3), 302-311.

Gupta, J. N. D. (1986). Flowshop schedules with sequence-dependent setup times. *Journal of Operations Research Society of Japan*, 29 (3), 206-219.

Hall, N. G. & Sriskandarajah, C. (1996). A survey of machine scheduling problems with block-ing and no-wait in process. *Operations Research*, 44 (3), 510-525.

Kumar, S., Bagchi, T. P. & Sriskandarajah, C. (2000). Lot streaming and scheduling heuristics for m-machine no-wait flowshops. *Computers & Industrial Engineering*, 38 (1), 149-172.

Pan, Q.-K., Tasgetiren, M.F. & Liang, Y.-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research,* 35 (9), 2807-2839.

Ruiz, R. & Allahverdi, A. (2007a). Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. *Annals of Operations Research*, 156 (1), 143-171.

Ruiz, R. & Allahverdi, A. (2007b). A. No-wait flowshop with separate setup times to minimize maximum lateness. *The International Journal of Advanced Manufacturing Technology*, 35 (5-6), 551-565.

Ruiz, R., Maroto, C. & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence-dependent setup times using advanced metaheuristics*. European Journal of Operational Research*, 165 (1), 34-54.

Ruiz, R. & Stützle, T. (2008). An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187 (3), 1143-1159.

Shyu, S. J., Lin, B. M. T. & Yin, P. Y. (2004). Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. *Computers & Industrial Engineering*, 47 (2-3), 181-193.

Stafford Jr, E. F. & Tseng, F. T. (1990). On the Srikar-Ghosh MILP model for the NxM SDST flowshop problem*. International Journal of Production Research*, 28 (10), 1817-1830.

Stafford Jr, E. F. & Tseng, F. T. (2002). Two models for a family of flowshop sequencing problems. *European Journal of Operational Research*, 142 (2), 282-293.

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

Wang, C., Li, X. & Wang, Q. (2010). Accelerated tabu search for no-wait flowshop scheduling problem with maximum lateness criterion. *European Journal of Operational Research*, 206 (1), 64-72.

Yaurima, V., Burtseva, L. & Tchernykh, A. (2009). Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. *Computers & Industrial Engineering*, 56 (4), 1452-1463.