A HIGH QUALITY SOLUTION HEURISTIC FOR NO-WAIT FLOW SHOP SCHEDULING

Marcelo Seido Nagano
School of Engineering of São Carlos, University of São Paulo
Av. Trabalhador São-Carlense, 400, 13566-590 São Carlos-SP
drnagano@usp.br

Fábio José Ceron Branco
School of Engineering of São Carlos, University of São Paulo
Av. Trabalhador São-Carlense, 400, 13566-590 São Carlos-SP
fbranco@hotmail.com

João Vitor Moccellin
School of Engineering of São Carlos, University of São Paulo
Av. Trabalhador São-Carlense, 400, 13566-590 São Carlos-SP
jvmoccel@sc.usp.br

ABSTRACT
This paper deals with the basic no-wait Flow Shop Sequencing in order to minimize the total time to complete the schedule. It is introduced a constructive heuristic which builds the production schedule from job partial sequences by using an appropriate combination of both shift and interchange job-sequence neighborhoods. An extensive computational experiment has been performed for the performance evaluation of the proposed heuristic. Experimental results have clearly shown that the presented heuristic provides better solutions than those from the best three existing ones.

KEYWORDS: Production scheduling. No-wait flow shop. Heuristics.

1. Introduction

This paper deals with the basic *n*-job *m*-machine no-wait flow shop scheduling problem.

A flow shop scheduling problem is a production problem where a set of *n* jobs have to be processed on *m* different machines with identical machine routing. The traditional problem model considers that job processing times are known, fixed, and include machine setup times. Moreover, job operations on the machines may not be preempted. Usually, the jobs have the same sequencing on all machines. This processing environment is known as permutation flow shop. If job passing is not allowed, and all jobs have equals release dates the number of possible schedules is *n*!. Therefore, the scheduling problem consists of finding a job sequence that optimizes an appropriate schedule performance measure. In this paper, such a performance measure is the makespan, that is, the total time to complete the schedule. This traditional *n*-job, *m*-machine permutation flow shop scheduling problem can be mathematically defined as follows.

Let $\sigma = J_{[1]}\, J_{[2]} \ldots J_{[n]}$ be a job sequence, that is a possible permutation schedule, where $J_{[i]}$ denotes the job in the *i*th position of σ. The processing time of job $J_{[i]}$ on machine k ($k = 1, 2, ..., m$) is given by $p_{[i]\,k}$, which includes the machine setup time.

According to the general assumptions for the traditional flow shop scheduling problem, the processing start time of job $J_{[i]}$ on machine *k* is given by

$$E_{[i]\,k} = \max\left[C_{[i]\,(k\text{-}1)}\,,\, C_{[i\text{-}1]\,k}\right] \qquad (1)$$

where $C_{[i]\,(k\text{-}1)}$ is the completion time of job $J_{[i]}$ on machine (k-1) , and $C_{[i\text{-}1]\,k}$ the completion time of job $J_{[i-1]}$ on machine *k*. Therefore, the completion time of job $J_{[i]}$ on machine *k* is obtained by $C_{[i]\,k} = E_{[i]\,k} + p_{[i]\,k}$, that is

$$C_{[i]\,k} = \max\left[C_{[i]\,(k\text{-}1)}\,,\, C_{[i\text{-}1]\,k}\right] + p_{[i]\,k} \qquad (2)$$

Where $C_{[i]\,0} = 0$ and $C_{[0]\,k} = 0$.

If all job release dates are equals, which are adopted to be zero, then the makespan equals the maximum job completion time $C_{\max} = C_{[n]\,m}$.

As aforementioned, the scheduling problem consists of finding a sequence for the jobs that minimizes the makespan $C_{[n]\,m}$.

According to expression (1) the processing start time $E_{[i]\,k}$ of job $J_{[i]}$ on machine *k* is given by either $C_{[i]\,(k\text{-}1)}$ or $C_{[i\text{-}1]\,k}$, unless $C_{[i]\,(k\text{-}1)} = C_{[i\text{-}1]\,k}$.

Suppose that $C_{[i]\,(k\text{-}1)} \neq C_{[i\text{-}1]\,k}$. If $E_{[i]\,k} = C_{[i]\,(k\text{-}1)}$, then there will be an idle time on machine *k* between the end of job $J_{[i-1]}$ and the start of job $J_{[i]}$. Otherwise, the operation of job $J_{[i]}$ on machine *k* must wait for the end of job $J_{[i-1]}$ on the same machine, resulting in a waiting time between the successive operations of job $J_{[i]}$ on machines (*k*-1) and *k*. Therefore, it is usual to have in a feasible schedule both idle times on machines and waiting times between successive operations of a job. It is worth noting that in this traditional flow shop scheduling problem there is no idle times between successive jobs on the first machine.

The Flow Shop environment is common in a number of production systems. For some of them the processing of a job cannot be interrupted once started. Some examples of such production systems are chemical, metal, and food processing industries. For these production

environments the traditional flow shop scheduling model is not an appropriate one. However, the traditional scheduling model can easily be adapted to those production systems. Assuming that the general assumptions for the traditional flow shop scheduling problem can be accepted, it is sufficient the addition of a constraint concerning the waiting times between successive operations of the jobs, that is, these waiting times must always be zero. This is the basic no-wait flow shop scheduling problem that is treated in this paper.

Due to the constraint regarding the waiting times between successive operations of the jobs, it is expected the occurrence of idle times on the first machines between successive jobs.

Consider a feasible schedule for the no-wait problem given by an arbitrary job sequence $\sigma = J_{[1]}\, J_{[2]} \ldots J_{[n]}$, and let $I_{[i\text{-}1]\,[i]}$ (i = 2, 3, …, n) be the idle time on the first machine between the successive jobs $J_{[i-1]}$ and $J_{[i]}$. According to the no-wait scheduling structure there are as many feasible schedules as it is desired with the same job sequence σ. Of course, the σ-schedule with the minimum makespan is the best. Such a schedule is given when the idle times $I_{[i\text{-}1]\,[i]}$ reach their minimum values in order to keep the schedule feasibility. These minimum $I_{[i\text{-}1]\,[i]}$, denoted by $I_{[i\text{-}1]\,[i]\,\min}$, are calculated as a function of the processing times of jobs $J_{[i-1]}$, and $J_{[i]}$. Wismer (1972) presents the procedure to obtain $I_{[i\text{-}1]\,[i]\,\min}$ for i = 2, 3, ….., n, and for any job pair from the set of n jobs.

Taking into account the idles times $I_{[i\text{-}1]\,[i]\,\min}$ on the first machine, the makespan is calculated by the following expression:

$$C_{[n]\,m} = \sum_{i=1}^{n} p_{[i]\,1} + \sum_{i=2}^{n} I_{[i-1]\,[i]\,\min} + \sum_{k=2}^{m} p_{[n]\,k} \qquad (3)$$

Therefore, the basic n-job m-machine no-wait flow shop scheduling problem considered in this paper consists of finding a sequence for the jobs that minimizes the makespan $C_{[n]\,m}$ given by expression (3).

Another usual production scheduling performance measure is the mean flow time, which is the same as total flow time. Concerning this performance measure, Van Deman and Baker (1974) were the first to deal with the basic no-wait flow shop scheduling problem. After them, Adiri and Pohoryles (1982), Rajendran and Chaudhuri (1990), Van Der Veen and Van Dal (1991), Chen et al. (1996), Bertolissi (2000), Fink and Voβ (2003), Aldowaisan and Allahverdi (2004), Kumar et al. (2006), Pan et al. (2008), and recently Framinan et al (2010).

For makespan minimization, some of the early researches are reported by Reddi and Ramamoorthy (1972), Wismer (1972), Bonney and Gundry (1976), and King and Spachis (1980). Gangadharan and Rajendran (1993), and Rajendran (1994) have developed heuristics which perform better than the heuristics presented by Bonney and Gundry (1976), and King and Spachis (1980).

The heuristic presented by Rajendran (1994) is a constructive heuristic that yields good solutions with small computational effort. The author has observed in expression (3) that the minimum makespan is obtained by the best matching for all possible pairs of adjacent jobs, which is related to the minimum idles times $I_{[i\text{-}1]\,[i]\,\min}$. In addition, the total processing time of the last job $J_{[n]}$ should be reduced. Moreover, it was also used the concept of Johnson's algorithm (1954) concerning both increasing and decreasing trend in job processing times.

An early survey of the basic no-wait flow shop scheduling can be found in Hall and Sriskandarajah (1996).

Aldowaisan and Allahverdi (2003) have proposed meta-heuristics by using Genetic Algorithm, and Simulated Annealing. As it is well-known, meta-heuristics generally yields high

quality solutions but they are inefficient regarding computation times. As expected, the best two of the proposed meta-heuristics, denoted by SA2 and GEN2, obtain schedules with smaller makespan than Rajendran's method (1994).

Li et al. (2008) have presented a composite heuristic for the basic no-wait flow shop scheduling. A heuristic is named as a composite one when it involves one or more another existing heuristics. The composite heuristic by Li et al. has three phases. In the first phase the jobs are arranged according to non-descending order of their total processing time. Phase two constructs a *n*-job sequence by using a procedure similar to the solution construction procedure of the existing FL heuristic proposed by Framinan and Leisten (2003) for the traditional flow shop scheduling problem with the objective of minimizing mean flow time. At the end of the second heuristic phase, there are two complete sequences. One of them is that given by the initial job arrangement from the first phase and the second one is that obtained in the phase two. The last heuristic phase consists of a solution improvement, as follows: If the makespan for the sequence from phase two is smaller than that concerning the initial job arrangement, then phase two is performed again assuming as a new initial job arrangement the complete sequence that has been constructed in phase two. Otherwise, the heuristic stop criterion is reached. This iterative procedure may be done at most three times. Experimental results show that the heuristic presented by Li et al. (2008) outperforms in solution quality the existing algorithms SA2 (Aldowaisan and Allahverdi, 2003), RAJ (Rajendran, 1994), and GR (Gangadharan and Rajendran, 1993). Moreover, its CPU time was the least among the computation times required by the compared algorithms.

Recently, Laha and Chakraborty (2009) introduced a new constructive heuristic which is similar to the well-known NEH heuristic (Nawaz et al., 1983) which was originally developed for the traditional flow shop scheduling problem with the objective of minimizing makespan. The heuristic by Nawaz et al. has two basic phases. In the first phase an initial job arrangement is obtained by sequencing the jobs according to non-ascending order of their total processing time. The second phase consists of an iterative job insertion procedure, which starts with a partial 2-job sequence, and according to the job ordering from phase 1 the remaining jobs are one at a time successively scheduled. In the heuristic proposed by Laha and Chakraborty, the initial arrangement for the jobs is obtained by two steps. In the first step the jobs are arranged according to non-ascending order of their total processing time, as it is made in the NEH heuristic. Then, the second step generates 2(n-1) shift neighbors of the job ordering from step 1. The *n*-job neighbor sequence with the minimum makespan is selected as the initial job arrangement. By using this initial job arrangement, and starting from a partial sequence with the first pair of jobs, an iterative 2-job insertion procedure is performed up to a complete job sequence is constructed. Results from computational experience show that the proposed heuristic is superior to four of the best-known methods that have been reported in the literature, that is: GR (Gangadharan and Rajendran, 1993), RAJ (Rajendran, 1994), an insertion heuristic presented by Aldowaisan and Allahverdi (2003), and the Simulated Annealing meta-heuristic by Osman and Potts (1989) which was originally proposed for the traditional flow shop sequencing.

According to the literature examination for the basic *n*-job *m*-machine no-wait flow shop scheduling problem, the best heuristics with an appropriate trade-off between solution quality (minimum makespan) and computational effort are the RAJ heuristic (Rajendran, 1994), LWW heuristic (Li et al., 2008), and LC heuristic (Laha and Chakraborty, 2009).

In this paper, it is proposed a new constructive heuristic for minimizing makespan in basic no-wait flow shop scheduling problems. The remainder of this paper is organized as follows: Section 2 presents the proposed heuristic method. Computational results are presented in Section 3, where comparisons with the performance of three of the best-known existing heuristics are provided. Finally, conclusions are presented in Section 4.

2. The new heuristic

Similarly to existing heuristics as for instance the aforementioned heuristic by Laha and Chakraborty (2009), the new heuristic introduced in this paper is also related to the NEH one (Nawaz et al., 1983), having two stages. The first one is the same as the initial stage from NEH heuristic. The second stage uses local search procedures based on both shift and interchange neighborhoods of successive partial sequences, in order to obtain a complete job sequence.

The new heuristic, which is denoted by NBM-NWFS, can be stated as follows:

{*Stage I – Initial arrangement for the jobs*}
Step 1: For each job j calculate the total processing time on all the machines, given by $P_j = \sum_{k=1}^{m} p_{jk}$ ($j = 1, 2, \ldots, n$).
Step 2: Arrange the n jobs according to non- ascending order of P_j .

{Stage II – Solution construction}
Step 3: Select the two jobs from the first and second position of the arrangement for the jobs of Step 2, and find the best sequence for these two jobs by calculating the makespan for the two possible partial sequences.
Step 4: For $l = 3$ to n do
Select the job in the l-th position of the list generated in Step 2, and insert it at the last position of the current best partial sequence.
Denote this l-job sequence by S .
Find the best sequence from the entire shift neighborhood of sequence S . If the makespan of the best neighbor is better than that of S assign it to S .
Next, find the best sequence from the entire interchange neighborhood of sequence S . If the makespan of the best neighbor is better than that of S assign it to S .
The best n-job sequence S obtained by Step 4 is the solution sequence.

3. Computational results

The new constructive heuristic has been compared with three of the best-known existing algorithms, that is, RAJ heuristic (Rajendran, 1994), LWW heuristic (Li et al., 2008), and LC heuristic (Laha and Chakraborty, 2009).

In the computational tests, the heuristics were coded in Delphi and have been run on a microcomputer Intel Core 2 Quad, 2.4 GHz, 2 Gb RAM.

The computational experience was performed on two instance groups. The first group concerns small and medium size problems having 10, 20, 30, 40, 50, 60, and 70 jobs with 5, 10, 15, 20, 25, and 30 machines. The second group consists of large size problems having 80, 90, 100, 110, 120, 130, 140, and 150 jobs, with 5, 10, 15, 20, 25, and 30 machines. Each of the m x n combinations was replicated 100 times. The operation processing times were randomly generated from the discrete uniform distribution over the interval [1, 99]. Therefore, a total of 9000 problem instances were solved, where 4200 have had small/medium sizes, and 4800 large size test problems.

In the computational experience, two traditional statistics are used in order to evaluate the heuristic performances: percentage of success (in finding the best solution), and relative deviation (between the heuristics).

The percentage of success PS is given by the number of times the heuristic obtains the best makespan (alone or in conjunction with other) divided by the number of solved instances.

The relative deviation RD is given by:

$$RD_h = \frac{(M_h - M_*)}{M_*}$$

Where M_h is the makespan of the best sequence obtained by the heuristic *h*, and M_* the best makespan obtained by the heuristics, for a given test problem.

Table 1 shows the experimental results for small and medium size problems, while Table 2 presents the results related to large size problems.

As can be noted from Tables 1 and 2, the proposed NBM-NWFS heuristic clearly outperforms in solution quality all others compared heuristics.

Taking into account the percentages of success (PS), it is observed that for the smallest instances (number of jobs n = 10) the average PS is 73.16% growing up to 100 % for n ≥ 60 jobs. The relative deviations (RD), given by average percentage, substantiate the results concerning the percentages of success.

The results presented in Tables 1 and 2, with reference to solution quality (PS combined with RD), show that the compared heuristics could be arranged as follows: NBM-NWFS, LC, LWW, and RAJ.

Concerning the computation times, the fastest are the RAJ and LC heuristics followed by the proposed NBM-NWFS, and then by the LWW heuristic. However, the NBM-NWFS heuristic has taken on average 1.4467 seconds for solving the largest instances with n = 150 jobs. Of course, such a computational effort is not a constrained factor.

Table 1 – RD, PS and CPU times of the compared heuristics for small/medium size problems

Problem		RAJ			LWW			LC			NBM-NWFS		
n	M	RD (%)	PS (%)	CPU time (seconds)	RD (%)	PS (%)	CPU time (seconds)	RD (%)	PS (%)	CPU time (seconds)	RD (%)	PS (%)	CPU time (seconds)
10	5	2.343	21	0.0003	2.278	15	0.0024	2.166	22	0.0002	0.282	71	0.0006
	10	2.816	9	0.0003	1.922	20	0.0019	1.821	25	0.0001	0.436	69	0.0005
	15	3.084	10	0.0006	1.918	19	0.0020	1.804	29	0.0003	0.182	82	0.0006
	20	3.169	13	0.0002	1.810	22	0.0019	1.993	23	0.0005	0.372	74	0.0003
	25	2.982	11	0.0006	1.857	21	0.0020	1.975	27	0.0001	0.352	73	0.0006
	30	2.779	10	0.0003	1.634	21	0.0024	1.707	28	0.0002	0.298	70	0.0008
Average		2.862	12.33	0.0004	1.903	19.66	0.0021	1.911	25.66	0.0002	0.321	73.16	0.0006
20	5	3.791	0	0.0005	2.697	5	0.0148	2.748	6	0.0001	0.092	89	0.0016
	10	3.447	4	0.0003	3.451	7	0.0147	2.971	5	0.0001	0.187	84	0.0014
	15	3.559	3	0.0003	3.281	4	0.0144	2.599	7	0.0003	0.107	86	0.0014
	20	3.377	1	0.0002	3.505	3	0.0145	2.456	6	0.0001	0.070	91	0.0044
	25	3.489	2	0.0006	3.324	6	0.0148	2.372	12	0.0003	0.102	83	0.0019
	30	3.399	6	0.0008	3.149	5	0.0146	2.420	8	0.0005	0.167	83	0.0020
Average		3.511	2.66	0.0005	3.234	5	0.0146	2.594	7.33	0.0002	0.121	86	0.0021
30	5	4.885	0	0.0005	3.007	1	0.0491	3.309	5	0.0006	0.010	95	0.0045
	10	4.221	2	0.0005	3.998	1	0.0489	3.419	1	0.0006	0.036	96	0.0042
	15	4.107	0	0.0005	4.154	0	0.0492	3.009	3	0.0008	0.015	98	0.0050
	20	4.244	1	0.0006	3.884	0	0.0491	3.013	3	0.0008	0.031	96	0.0047
	25	4.109	1	0.0006	3.667	2	0.0491	3.111	4	0.0005	0.074	93	0.0055
	30	3.985	3	0.0008	4.066	1	0.0499	3.091	2	0.0006	0.049	94	0.0055
Average		4.258	1.16	0.0006	3.796	0.83	0.0492	3.158	3	0.0007	0.036	95.33	0.0049
40	5	5.741	0	0.0003	3.019	2	0.1170	3.236	0	0.0009	0.007	98	0.0109
	10	4.411	0	0.0008	4.088	0	0.1174	3.234	1	0.0011	0.004	99	0.0119
	15	4.134	0	0.0006	4.363	0	0.1172	3.778	0	0.0008	0.000	100	0.0117
	20	3.834	1	0.0009	4.280	1	0.1183	3.363	1	0.0011	0.019	97	0.0125
	25	3.894	0	0.0009	4.392	1	0.1178	3.285	1	0.0013	0.021	98	0.0130
	30	4.043	2	0.0014	4.526	1	0.1181	3.122	2	0.0017	0.031	95	0.0138
Average		4.343	0.50	0.0008	4.111	0.83	0.1176	3.336	0.83	0.0012	0.014	97.83	0.0123

50	5	6.202	0	0.0006	3.015	1	0.2322	3.667	0	0.0006	0.007	99	0.0238
	10	4.343	0	0.0011	4.137	0	0.2325	3.514	0	0.0013	0.000	100	0.0250
	15	4.753	0	0.0014	4.297	1	0.2331	3.810	0	0.0014	0.012	99	0.0264
	20	4.369	0	0.0016	4.632	0	0.2344	3.495	0	0.0017	0.000	100	0.0272
	25	4.482	0	0.0016	4.632	0	0.2335	3.483	1	0.0019	0.006	99	0.0280
	30	4.063	1	0.0019	4.415	0	0.2342	3.515	0	0.0017	0.001	99	0.0289
Average		4.702	0.16	0.0014	4.188	0.33	0.2333	3.581	0.16	0.0014	0.004	99.33	0.0266
60	5	6.717	0	0.0011	3.115	0	0.4063	3.695	0	0.0017	0.000	100	0.0488
	10	4.459	0	0.0012	4.060	0	0.4067	3.588	0	0.0020	0.000	100	0.0475
	15	4.718	0	0.0020	4.774	0	0.4077	3.701	0	0.0017	0.000	100	0.0492
	20	4.522	0	0.0019	4.840	0	0.4075	3.728	0	0.0022	0.000	100	0.0513
	25	4.133	0	0.0022	4.714	0	0.4091	3.525	0	0.0025	0.000	100	0.0517
	30	4.145	0	0.0028	4.499	0	0.4088	3.616	0	0.0028	0.000	100	0.0534
Average		4.782	0	0.0019	4.334	0	0.4077	3.642	0	0.0022	0.000	100	0.0503
70	5	7.312	0	0.0016	3.079	0	0.6564	3.851	0	0.0025	0.000	100	0.0805
	10	4.643	0	0.0020	3.827	0	0.6580	3.537	0	0.0028	0.000	100	0.0822
	15	4.853	0	0.0022	4.502	0	0.6584	3.842	0	0.0025	0.000	100	0.0869
	20	4.537	0	0.0025	4.683	0	0.6594	3.668	0	0.0033	0.000	100	0.1044
	25	4.571	0	0.0028	4.746	0	0.6588	3.666	0	0.0036	0.000	100	0.1386
	30	4.551	0	0.0036	4.700	0	0.6605	3.410	0	0.0041	0.000	100	0.0931
Average		5.077	0	0.0025	4.256	0	0.6586	3.662	0	0.0031	0.000	100	0.0976

Table 2 – RD, PS and CPU times of the compared heuristics for large size problems

Problem		RAJ			LWW			LC			NBM-NWFS		
n	m	RD (%)	PS (%)	CPU time (seconds)	RD (%)	PS (%)	CPU time (seconds)	RD (%)	PS (%)	CPU time (seconds)	RD (%)	PS (%)	CPU time (seconds)
80	5	7.485	0	0.0023	2.988	0	1.0920	3.913	0	0.0027	0.000	100	0.1305
	10	5.065	0	0.0025	4.510	0	1.0249	3.852	0	0.0033	0.000	100	0.1331
	15	4.602	0	0.0030	4.811	0	1.0020	3.631	0	0.0036	0.000	100	0.1436
	20	4.613	0	0.0034	4.882	0	1.0028	3.819	0	0.0045	0.000	100	0.1478
	25	4.449	0	0.0041	5.039	0	1.0042	3.685	0	0.0047	0.000	100	0.1458
	30	4.537	1	0.0044	5.141	0	1.0208	3.874	0	0.0053	0.001	99	0.1500
Average		5.125	0.16	0.0033	4.562	0	1.0245	3.796	0	0.0040	0.000	99.83	0.1418
90	5	7.741	0	0.0027	3.184	0	1.4633	4.018	0	0.0038	0.000	100	0.2058
	10	4.866	0	0.0033	4.291	0	1.4533	3.803	0	0.0041	0.000	100	0.2105
	15	4.529	0	0.0041	4.593	0	1.4653	3.734	0	0.0048	0.000	100	0.2252
	20	4.550	0	0.0044	5.008	0	1.4755	3.723	0	0.0053	0.000	100	0.2286
	25	4.410	0	0.0052	5.073	0	1.4556	3.692	0	0.0058	0.000	100	0.2252
	30	4.481	0	0.0056	5.099	0	1.4553	3.838	0	0.0067	0.000	100	0.2309
Average		5.096	0	0.0042	4.541	0	1.4614	3.801	0	0.0051	0.000	100	0.2210
100	5	7.891	0	0.0034	3.019	0	2.0233	4.028	0	0.0047	0.000	100	0.3047
	10	4.809	0	0.0042	4.167	0	2.0241	3.867	0	0.0053	0.000	100	0.3023
	15	4.513	0	0.0050	4.615	0	2.0259	3.765	0	0.0061	0.000	100	0.3288
	20	4.596	0	0.0056	4.873	0	2.0505	3.664	0	0.0066	0.000	100	0.3374
	25	4.597	0	0.0063	5.233	0	2.0748	3.919	0	0.0073	0.000	100	0.3285
	30	4.615	0	0.0070	5.102	0	2.0392	3.897	0	0.0078	0.000	100	0.3339
Average		5.170	0	0.0053	4.501	0	2.0396	3.856	0	0.0063	0.000	100	0.3226
110	5	8.306	0	0.0042	2.938	0	2.7366	3.900	0	0.0055	0.000	100	0.4316
	10	4.715	0	0.0053	4.086	0	2.7381	3.671	0	0.0066	0.000	100	0.4327
	15	4.505	0	0.0064	4.824	0	2.7392	3.770	0	0.0075	0.000	100	0.4641
	20	4.712	0	0.0072	5.005	0	2.7402	3.852	0	0.0080	0.000	100	0.4720
	25	4.609	0	0.0078	5.015	0	2.7655	3.972	0	0.0088	0.000	100	0.5214
	30	4.440	0	0.0089	5.291	0	2.7405	4.026	0	0.0094	0.000	100	0.4739
Average		5.214	0	0.0066	4.526	0	2.7434	3.865	0	0.0076	0.000	100	0.4660

120	5	8.481	0	0.0055	2.914	0	3.5944	4.043	0	0.0063	0.000	100	0.6031
	10	4.876	0	0.0066	4.136	0	3.5945	3.539	0	0.0072	0.000	100	0.5994
	15	4.468	0	0.0072	4.638	0	3.5978	3.619	0	0.0092	0.000	100	0.6333
	20	4.680	0	0.0085	5.154	0	3.5978	3.884	0	0.0097	0.000	100	0.7017
	25	4.438	0	0.0094	5.139	0	3.5988	3.926	0	0.0105	0.000	100	0.6492
	30	4.418	0	0.0105	5.140	0	3.5983	3.878	0	0.0117	0.000	100	0.6586
Average		5.227	0	0.0080	4.521	0	3.5969	3.815	0	0.0091	0.000	100	0.6409
130	5	8.428	0	0.0067	2.766	0	4.6239	3.952	0	0.0080	0.000	100	0.8156
	10	4.834	0	0.0081	3.991	0	4.6478	3.621	0	0.0094	0.000	100	0.8850
	15	4.594	0	0.0092	4.774	0	4.6353	3.860	0	0.0103	0.000	100	0.8545
	20	4.604	0	0.0106	5.000	0	4.6380	3.774	0	0.0117	0.000	100	0.8649
	25	4.289	0	0.0114	5.286	0	4.6394	3.927	0	0.0125	0.000	100	0.9191
	30	4.347	0	0.0123	5.297	0	4.6453	3.833	0	0.0141	0.000	100	0.8830
Average		5.1827	0	0.0097	4.519	0	4.6383	3.8278	0	0.0110	0.000	100	0.8704
140	5	8.775	0	0.0081	2.858	0	5.8372	4.049	0	0.0094	0.000	100	1.0820
	10	5.063	0	0.0095	4.077	0	5.8255	3.672	0	0.0108	0.000	100	1.0664
	15	4.505	0	0.0108	4.752	0	5.8306	3.958	0	0.0125	0.000	100	1.1739
	20	4.440	0	0.0119	4.990	0	5.8331	3.682	0	0.0134	0.000	100	1.1364
	25	4.352	0	0.0134	5.227	0	5.8364	3.982	0	0.0147	0.000	100	1.1375
	30	4.619	0	0.0147	5.414	0	5.8514	3.811	0	0.0159	0.000	100	1.2161
Average		5.2923	0	0.0114	4.553	0	5.8357	3.859	0	0.0128	0.000	100	1.1354
150	5	8.967	0	0.0097	2.885	0	7.2222	3.997	0	0.0111	0.000	100	1.4011
	10	4.723	0	0.0114	3.998	0	7.2294	3.475	0	0.0127	0.000	100	1.3769
	15	4.426	0	0.0133	4.915	0	7.2313	3.859	0	0.0144	0.000	100	1.4773
	20	4.418	0	0.0142	5.210	0	7.2534	3.913	0	0.0158	0.000	100	1.4628
	25	4.448	0	0.0161	5.337	0	7.2377	3.919	0	0.0165	0.000	100	1.4720
	30	4.540	0	0.0175	5.448	0	7.2409	3.983	0	0.0186	0.000	100	1.4900
Average		5.254	0	0.0137	4.632	0	7.2358	3.857	0	0.0149	0.000	100	1.4467

4. Final remarks

As it is well-known, desired features of heuristic methods are: simplicity, easy implementation, computational efficiency, and effectiveness, in order to yield near-optimal solutions. Having this in mind, this paper has introduced a new simple heuristic for the basic no-wait Flow Shop Sequencing with the objective of minimizing makespan.
Regarding solution quality, results from computational experience have shown that the proposed heuristic performs better than the best three ones that have been presented in the literature. Moreover, the computational effort is not significant to be worth considering.

Acknowledgements – The research reported in this paper is partially supported by a grant from the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, Brasil.

References

Adiri, I. and Pohoryles, D. (1982), Flow shop/no-idle or no-wait scheduling to minimize the sum of completion times, *Naval Research Logistics*, 29, 495-504.
Aldowaisan, T. and Allahverdi, A. (2003), New heuristics for no-wait flow shops to minimize makespan, *Computers & Operations Research*, 30, 1219-1231.
Aldowaisan, T. and Allahverdi, A. (2004), A new heuristic for m-machine no-wait flow shop to minimize total completion time, *Omega*, 32, 345-352.
Bertolissi, E. (2000), Heuristic algorithm for scheduling in the no-wait flow-shop, *Journal of Materials Processing Technology*, 107, 459-465.
Bonney, M. C. and Gundry, S. W. (1976), Solutions to the constrained flowshop sequencing problem, *Operations Research Quarterly*, 24, 869-883.
Chen, C-L., Neppalli, R. V. and Aljaber, N. (1996), Genetic algorithms applied to the continuous flow shop problem, *Computers & Industrial Engineering*, 30, 919–929.
Fink, A. and Voß, S. (2003), Solving the continuous flow-shop scheduling problem by metaheuristics, *European Journal of Operational Research*, 151, 400–414.
Framinan, J. M., Nagano, M. S. and Moccellin, J. V. (2010), An efficient heuristic for total flowtime minimisation in no-wait flowshops, *International Journal of Advanced Manufacturing Technology*, 46, 1049-1057.
Framinan, J. M. and Leisten, R. (2003), An efficient constructive heuristic for flowtime minimisation in permutation flow shops, *Omega*, 31, 311-317.
Gangadharan, R. and Rajendran, C. (1993), Heuristic algorithms for scheduling in the no-wait flowshop, *International Journal of Production Economics*, 32, 285-290.
Hall, N.G. and Sriskandarajah C. (1996), A survey of machine scheduling problems with blocking and no-wait in process, *Operations Research*, 44, 510-525.
Johnson, S.M. (1954), Optimal two-and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, 1, 61-68.
King, J. R. and Spachis, A. S. (1980), Heuristics for flowshop scheduling, *International Journal of Production Research*, 18, 343-357.
Kumar, A., Prakash, A., Shankar, R. and Tiwari, M. K. (2006), Psychoclonal algorithm based approach to solve continuous flow shop scheduling problem, *Expert Systems with Applications*, 31, 504-514.
Laha, D. and Chakraborty, U. K. (2009), A constructive heuristic for minimizing makespan in no-wait flow shop scheduling, *International Journal of Advanced Manufacturing Technology*, 41, 97-109.
Li, X., Wang Q. and Wu, C. (2008), Heuristic for no-wait flow shops with makespan minimization, *International Journal of Production Research*, 46, 2519-2530.
Nawaz, M., Enscore, E. Jr. and Ham, I. (1983), A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, 1, 91–95.

Osman, I.H. and Potts, C.N. (1989), Simulated annealing for permutation flow-shop scheduling, *Omega*, 17, 551–557.
Pan, Q-K., Tasgetiren, M. F. and Liang, Y-C. (2008), A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, *Computers & Operations Research*, 35, 2807-2839.
Rajendran, C. (1994), A no-wait flowshop scheduling heuristic to minimize makespan, *Journal of the Operational Research Society*, 45, 472-478.
Rajendran, C. and Chaudhuri, D. (1990), Heuristic algorithms for continuous flow shop problem, *Naval Research Logistics*, 37, 695-705.
Reddi, S. S. and Ramamoorthy, C. V. (1972), On the flowshop sequencing problems with no wait in process, *Operational Research Quarterly*, 23, 323-331.
Van Deman, J. M. and Baker, K. R. (1974), Minimizing flowtime in the flow shop with no intermediate queues, *IIE Transactions*, 6, 28-34.
Van Der Veen, J. A. A. and Van Dal, R. (1991), Solvable cases of the no-wait flowshop scheduling problem, *Journal of the Operational Research Society*, 42, 971-980.
Wismer, D. A. (1972), Solution of the flowshop sequencing problem with no intermediate queues, *Operations Research*, 20, 689-697.