Industrial Engineering
& Management Systems
Vol 15, No 3, September 2016, pp.206-214
ISSN 1598-7248 │ EISSN 2234-6473 │

http://dx.doi.org/10.7232/iems.2016.15.3.206
© 2016 KIIE

A High Quality Solution Constructive Heuristic for No-Wait Flow Shop Scheduling Problem

Marcelo Seido Nagano*, Hugo Hissashi Miyata
School of Engineering of São Carlos, University of São Paulo, São Carlos-SP, Brazil

(Received: February 6, 2016 / Revised: July 1, 2016 / Accepted: September 9, 2016)

ABSTRACT

This paper deals with the no-wait flow shop scheduling problem in order to minimize the total time to complete the schedule or makespan. It is introduced a constructive heuristic which builds the production schedule from job partial sequences by using an appropriate mechanism of insertion. An extensive computational experiment has been performed to evaluate the performance of proposed heuristic. Experimental results have clearly shown that the presented heuristic provides better solutions than those from the best heuristics existing.

Keywords: Production Scheduling, No-Wait Flow Shop, Heuristics, Makespan

* Corresponding Author, E-mail: drnagano@usp.br, drnagano@sc.usp.br

1. INTRODUCTION

This paper deals with the n-job m-machine no-wait flow shop scheduling problem. The traditional flow shop problem have been studied since 1950 (Johnson, 1954). A flow shop scheduling problem is a production problem where a set of n jobs have to be processed on m machines with identical machine routing. The traditional problem model considers that job processing times are known, fixed, and included 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, then, 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 $\pi = J_{[1]}, J_{[2]}, \cdots, J_{[n]}$ be a job sequence, that is, a possible permutation schedule, where $J_{[i]}$ denotes the job in the ith position of π. The processing time of job $J_{[i]}$ on machine k ($k = 1, 2, \cdots, m$) 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

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

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

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

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 to 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 $C_{[i](k-1)}$ or $C_{[i-1]k}$, unless $C_{[i](k-1)} = C_{[i-1]k}$.

Suppose that $C_{[i](k-1)} \neq C_{[i-1]k}$. If $E_{[i]k} = C_{[i](k-1)}$, 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, food processing industries and printed circuit board manufacturing (Adiri and Pohoryles, 1982; Hall and Sriskandarajah, 1996; Nagano *et al*., 2012; Laha and Sapkal, 2014; Nagano *et al*., 2015). In addition, modern manufacturing environments where robots and industrial machines provide a highly coordinated process, can frequently be modeled as a no-wait scheduling problem (Bertolossi, 2000). 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 machine between successive jobs.

Consider a feasible schedule for the no-wait problem given by an arbitrary job sequence $\pi = \{J_{[1]}, J_{[2]}, \cdots, J_{[n]}\}$, and let $I_{[i-1][i]}$ (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 $I_{[i-1][i]}$ reach their minimum values in order to keep the schedule feasibility. These minimum $I_{[i-1][i]}$, denoted by $I_{[i-1][i]\min}$, are calculated as a function of the processing times of jobs $J_{[i-1]}$, and $J_{[i]}$. Wismer (1972) presents the procedure to obtain $I_{[i-1][i]\min}$ for i = 2, 3, ⋯, n, and for any job pair from the set of n jobs.

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

$$C_{[n]m} = \sum_{i=1}^{n} p_{[i]1} + \sum_{i=2}^{n} p_{[i-1][i]\min} + \sum_{k=1}^{n} p_{[n]k} \tag{3}$$

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). Minimizing makespan meansimprove the utilization of the productive resources (Baker, 1974).

An early survey of the basic no-wait flow shop scheduling can be found in Hall and Sriskandarajah (1996). Nagano and Miyata (2015) reviewed and classified the constructive heuristics for the m-machine case with makespan and total flow time minimization. Recently, Allahverdi (2016) published a comprehensive survey of the no-wait in process scheduling problems. The no-wait flow shop scheduling problems with more than two machines belong to the class of NP-hard (Papadimitriou and Kanelakis, 1980; Rock, 1984; Van Der Veen *et al*., 1991; Fink and Voβ, 2003; Laha and Sapkal, 2014). Because of exponential computational effort requirements, heuristic methods have been proposed for the problem with m-machines in order to provide near-optimal solution in small computational effort (Laha *et al*., 2013).

For makespan minimization, some of the early researchesare 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 constructive heuristics which perform better than the heuristics presented by Bonney and Gundry (1976) and King and Spachis (1980). Recently, some constructive heuristics are reported by Grabowski and Pempera (2005), Li *et al*. (2008), Li and Wu (2008) and Laha and Chakraborty (2009). Regarding metaheuristics, applications by Genetic Algorithms can be seen in Aldowaisan and Allahverdi (2003), Chaudhry and Khan (2012), Samarghandy and Elmekkawy (2012) and Ying *et al*. (2012). The application of Partice Swarm Optimization can be seen in Liu *et al*. (2007), Pan *et al*. (2008) and Samarghandy and Elmekkawy (2014). Grabowski and Pempera (2005) proposed three different versions of the Tabu Search. França *et al*. (2006) proposed a memetic algorithm. Aydilek and Allahverdi (2012) proposed four different versions of Simulated Annealing. Nagano, Almeida da Silva and Lorena (2014) developed an Evolutionary Clustering Search for the problem explicitly adding the dependent setup times. Recently, Ding *et al*. (2015) proposed an Improved Iterated Greedy with a Tabu-based reconstruction strategy. For the problem with two machines and the single server constraint, Samarghandy (2015) proposed two versions of Genetic Algorithm for the problem with side-server and dependent sequence setup times constraints.

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 C_{max} is obtained by the best matching for all possible pairs of adjacent jobs, which is related to the minimum idles times $I_{[i-1]\,[i]\min}$. 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.

Aldowaisan and Allahverdi (2003) have proposed meta-heuristics by using Genetic Algorithm, and Simulated Annealing. As it is well-known, meta-heuristics generally yield 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.* (2008) 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 phase consists of a solution improvement, as follows: If the C_{max} 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.

Li and Wu (2008) have proposed a new composite heuristic called Fast Composite Heuristic with three phases. The first phase employed the Nawaz *et al.* (1983) heuristic, e.g., the jobs are arranged according to non-decreasing order of their total processing time and the jobs are inserted according to the construction procedure of NEH. In the second phase, FCH uses a procedure similar to the solution construction procedure of the existing RZ heuristic proposed by Rajendran and Ziegler (1997) for the permutation flow shop scheduling problem with total flow time minimization. The procedure is repeated until no more bettersolutions are found or until 10 iterations. The last phase employs the Backward Swap and Continue (BSC) local search until no more better solutions are found or until 20 iterations.

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.* (1983) 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 lowest C_{max} 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 outperformed GRH-2 (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 RAJ heuristic (Rajendran, 1994), NEHDS+M (Grabowski and Pempera, 2005), LWW heuristic (Li *et al.*, 2008), FCH (Li and Wu, 2008) and LC heuristic (Laha and Chakraborty, 2009).

In this paper, we propose a new constructive heuristic for minimizing makespan in *m*-machine 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

The new heuristic introduced in this paper is composed by three phases: Initial arrangement for the jobs, solution construction and improvement of the solution. In the first phase, the constructive heuristic the uses index developed by Framinan and Nagano (2008) to create an initial sequence of jobs. The second phase employs the construction mechanism by Gao *et al.* (2012). In order to obtain better solutions, the local search of insertion by Nagano *et al.* (2015) and backward swap and continue by Li and Wu (2008) are employed as an improvement local search.

The pseudo code of the heuristic, denoted by NM_{LS}, can be stated as follows:

Given a set $J = \{J_1, J_2, J_3, \cdots, J_n\}$ of *n* jobs, let π be the set of unscheduled jobs, π' be the initial sequence, π'' be the initial constructed sequence, π''' be the constructed sequence, σ be the best sequence found in the improvement phase, p_{ij} be the processing time of job *i* on machine *j* ($1 \le j \le m$), P_i be the total processing time of the job and K_i be the auxiliary index of job *i*.

{Stage I: Generation of the Initial Sequence}

```
Step 1: Compute the total processing time of each job $\left(P_i = \sum_{j=1}^{m} p_{ij}\right)$ from $\pi$.
Step 2: For each pair of jobs $(J_i, J_k) \in \pi$, compute the matrix $C_{\pi(Ji,\, Jk)m}$ that corresponds to the completion time of job k on the last machine according to Eq. (3);
Step 3: Select the job with the shortest processing time $P_i$ as the first job $J_{[1]}$ of the initial sequence $\pi'$. Select the second job such that $C_{max\,\pi'(J_{[1]}, J_{[2]}),m}$ is the lowest, being $J_{[2]}$ any job from the sequence assigned to $\pi$. The pair of jobs are assigned, respectively, on the first and last positions of the initial sequence $\pi'$.
Step 4: Set $\pi \leftarrow \pi - J_i - J_k$ being $J_i$ and $J_k$ the pair of jobs from $\pi$ that were inserted in $\pi'$ and $i \leftarrow n -2$;
Step 5: While $\pi \ne \phi$ do
    Step 5.1: For $k \leftarrow 1$ to $i$ do
        Step 5.1.1: Insert $J_{[k]} \in \pi$ on the second and third positions of $\pi'$ and according to the matrix generated in Step 2, evaluate $C_{\pi'(J_{[1]}, J_{[2]}),m}$ and $C_{\pi'(J_{[2]}, J_{[3]}),m}$, respectively. Assign the lowest value to $J_{[k]}$ as an index $K_i$;
    Step 5.2: Set $\pi \leftarrow \pi - J_{[k]}$ and $i \leftarrow i - 1$. If $\pi \leftarrow \phi$, go to Step 6, otherwise, go to Step 5.
Step 6: Order all jobs from $\pi$ in non-ascending order of $K_i$. Set $k \leftarrow 3$;
Step 7: While $\pi \ne \phi$ do
    Step 7.1: For $j \leftarrow 2$ to $(k - 1)$ do
        Step 7.1.1: Insert $J \in \pi$ in the jth position of $\pi'$ without modify the relative position of the jobs already scheduled. Compute the cost C = $C_{\pi'(J_{[j-1]}, J_{[j]}),m} + C_{\pi'(J_{[j]}, J_{[j+1]}),m} - C_{\pi'(J_{[j-1]}, J_{[j+1]}),m}$. The partial sequence that generates the lowest C is adopted as $\pi$ for the next iteration.
    Step 7.2: Set $k \leftarrow k+1$ and $\pi \leftarrow \pi - J$. If $\pi \leftarrow \phi$, go to Step 8, otherwise, go to Step 7.
```

{Stage II: Construction of the Solution}

```
Step 8: Pick the first two jobs from $\pi$ and evaluate the two possible partial sequences. The best partial sequence is chosen as the partial initial constructed sequence $\pi''$;
Step 9: While $\pi \ne \phi$ do
    Step 9.1: Pick the first two jobs from $\pi$ as a block and insert them in each of the j possible positions of $\pi''$; Select the best partial sequence as $\pi''$.
    Step 9.2: Set $\pi' \leftarrow \pi' - J_{[1]} - J_{[2]}$. If in the last iteration, $\pi$ remain with only one job, consider the single job as a block. If $\pi' \leftarrow \phi$ go to Step 10, otherwise, go to Step 9;
Step 10: Pick the first two jobs from $\pi''$ and evaluate the two possible partial sequences. The best partial sequence is chosen as the current sequence $\pi'''$;
Step 11: While $\pi'' \ne \phi$ do
    Step 11.1: Pick the first job from $\pi''$ and find the best partial sequence by placing it in all possible positions of $\pi'''$. Then, the best partial sequence is adopted as the current sequence $\pi'''$;
    Step 11.2: Pick $\pi'''$ generated in step 11.1 and insert each job $J \in \pi'''$ in all possible positions of $\pi'''$. Select the sequence that minimizes $C_{max}$. If $C_{max}$ of the candidate sequence is lower than $C_{max}$ given by $\pi'''$ set the candidate solution as $\pi'''$.
    Step 11.3: Set $\pi'' \leftarrow \pi'' - J_{[1]}$. If $\pi'' \leftarrow \phi$, go to Step 12, otherwise, go to Step 11.
```

{Stage III: Improvement of the Solution}

```
Step 12: Repeat Step 12.1 until better solution is not found
    Step 12.1: Insert each job from the sequence $\pi'''$ at the (n-1) possible positions. From the $(n-1)^2$ generated sequences, choose the sequence $\sigma$ that obtains the lowest $C_{max\,\sigma}$ and if $C_{max\,\sigma} < C_{max\,\pi'''}$, set $\pi''' \leftarrow \sigma$.
Step 13: Repeat Step 13.1 until better solution is not found.
    Step 13.1: Exchange each possible pair of jobs from the sequence $\pi'''$, apply the Backward Swap and Continue generating (n-1)/2 sequences for each job. From the (n(n-1))/2 generated sequences, choose the sequence $\sigma$ that obtains the lowest $C_{max\,\sigma}$ and if $C_{max\,\sigma} < C_{max\,\pi'''}$, set $\pi''' \leftarrow \sigma$.
```

3. COMPUTATIONAL RESULTS

The new constructive heuristic has been compared with the mainly existing algorithms: GRH-2 (Gangadharan and Rajendran, 1993), RAJ heuristic (Rajendran, 1994), NEHDS+M (Grabowski and Pempera, 2005), LWW heuristic (Li *et al.*, 2008), Fast Composite Heuristic (FCH)

(Li and Wu, 2008), and LC heuristic (LCH) (Laha and Chakraborty, 2009).

For the computational experiments, the heuristics were coded in Delphi and have been run on a micro-computeri5, 2410M, 2.3Ghz with 4Gb of RAM.The heuristics were tested by the means of the benchmark instance problems by Vallada *et al.* (2015).The benchmark consists of a set of problems of small and medium size, being each combination of n = {10, 20, 30, 40 50, 60} and *m* = {5, 10, 15, 20} and a set of problems of big size, being each combination of *n* = {100, 200, 300, 400, 500, 600, 700, 800} and *m* = {20, 40, 60}. Each combination between jobs and machines has ten problems, totalizing 480 problems. The processing times were randomly generated by using the integer uniform distribution over the interval [1, 99].

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 C_{max} (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 C_{max} of the best sequence obtained by the heuristic *h*, and M_* the best C_{max} 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 it can be seen from Tables 1 and 2, the proposed NM_{LS} heuristic clearly outperforms in solution quality all others compared heuristics, mainly for large size problems.

Taking into account the percentages of success (PS), it is observed that for the small and medium instances the average PS is 59.6%, better than FCH and growing up to 94% for the large instances. The relative deviations (RD), given by average percentage, substantiate the results concerning the percentages of success.

The results presented in Tables 1 and 2 show that NM_{LS} outperforms the heuristics of the literature with mean average relative percentage deviation (ARPD) equal to 0.204%. Regarding the heuristics of the literature, FCH performs better than the other heuristics of the literature, with mean ARPD equal to 0.728% followed by LWW (1.27%), NEHDS+M (2.72%), RAJ (4.71%) and GRH-2 (9%). NM_{LS} has the best performances in the small problems set (10 to 60 jobs) in general with Mean ARPD equal to 0.39%, followed by FCH (0.68%), LWW (1.09%), NEHDS+M (2.76%), RAJ (4.53%) and GRH-2 (5.31%). However, the proposed heuristic performs much better in the large problems set (100 to 800 jobs), once the solutions found, in most of the instances, are the best in comparison with other methods evaluated.

Concerning the computation times, the fastest heuristic are the RAJ and GRH-2 heuristics followed by the proposed NM_{LS}, and then by the LWW heuristic. However, the NM_{LS} heuristic has taken one of the longest computation times for solving the largest instances. Of course, such a computational effort is not a constrained factor.

4. FINAL REMARKS

As it is known, desired features of heuristic methodsare simplicity, easy implementation, computational efficiency, and effectiveness, in order to yield near-optimal solutions. Having this in mind, this paper has introduced a new constructive heuristic for the basic no-wait Flow Shop Sequencing with the objective of minimizing makespan. Regarding solution quality, results from computation results have shown that the proposed heuristic NM_{LS} outperforms the ones found in the literature. NM_{LS} in general obtained one of the best results for small problems set and provides the best results in large problems set in most of the instances evaluated. Although the proposed heuristic spend more computation time in comparison with most heuristics of the literature, but the computational effort is not a constrained factor.

ACKNOWLEDGEMENT

This work was supported by National Council for Scientific and Technological Development (CNPq), Brazil-Processes: 308047/2014-1 and 448161/2014-1.

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 and Operations Research*, **30**, 1219-1231.

Allahverdi, A. (In Press), A Survey of scheduling problems with no-wait in process, *European Journal of Operational Research*, doi: 10.1016/j.ejor.2016.05.036.

Aydilek, H. and Allahverdi, A. (2012), Heuristics for no-wait flowshops with makespan subject to mean completion time, *Applied Mathematics and Computation*, **219**, 351-359.

Baker, K. R. (1974), *Introduction to sequencing and scheduling*, Wiley.

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

Problem		GRH-2			RAJ			NEHDS+M			LWW			FCH			LCH			NM_{LS}		
n	m	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)
10	5	20	1.65	0.000	20	1.63	0.000	70	0.27	0.000	50	0.62	0.000	70	0.16	0.000	50	1.35	0.000	60	0.25	0.002
	10	30	2.05	0.000	30	2.77	0.000	60	1.07	0.000	70	0.49	0.002	60	0.59	0.000	40	1.96	0.002	60	0.76	0.006
	15	30	1.31	0.000	10	3.74	0.000	20	2.36	0.000	40	0.51	0.000	40	0.75	0.002	20	1.67	0.000	60	0.59	0.003
	20	40	2.37	0.000	20	2.37	0.000	30	1.85	0.002	70	0.11	0.002	70	0.53	0.000	30	1.73	0.000	60	0.45	0.000
Average		30	1.84	0.000	20	2.63	0.000	45	1.39	0.000	58	0.43	0.001	60	0.51	0.000	35	1.68	0.000	60	0.51	0.003
20	5	10	3.42	0.000	0	4.20	0.000	0	2.44	0.000	0	1.52	0.002	30	0.60	0.000	0	2.76	0.000	60	0.53	0.003
	10	0	4.11	0.000	10	3.76	0.000	20	1.68	0.002	40	1.09	0.003	30	0.98	0.000	0	3.02	0.002	20	0.56	0.005
	15	0	4.63	0.000	0	3.70	0.000	20	2.33	0.002	10	1.32	0.002	10	0.89	0.002	0	3.15	0.000	60	0.28	0.003
	20	0	3.89	0.000	0	3.85	0.000	0	2.92	0.002	10	1.11	0.003	30	0.72	0.002	10	3.21	0.002	60	0.39	0.005
Average		3	4.01	0.000	3	3.88	0.000	10	2.34	0.001	15	1.26	0.002	25	0.80	0.001	3	3.03	0.001	50	0.44	0.004
30	5	0	5.88	0.002	0	5.03	0.000	0	3.29	0.002	10	1.25	0.009	30	0.61	0.005	0	3.60	0.000	60	0.25	0.011
	10	0	4.44	0.000	0	4.34	0.002	0	2.82	0.000	10	1.24	0.008	50	0.28	0.003	0	3.67	0.000	50	0.55	0.006
	15	0	4.32	0.000	0	4.10	0.000	10	2.76	0.003	20	0.96	0.009	60	0.66	0.003	0	3.51	0.002	20	0.66	0.011
	20	0	6.46	0.002	0	4.06	0.002	0	2.69	0.002	0	1.12	0.011	70	0.26	0.005	0	3.39	0.000	40	0.97	0.014
Average		0	5.27	0.001	0	4.38	0.001	3	2.89	0.002	10	1.14	0.009	53	0.45	0.004	0	3.54	0.000	43	0.61	0.011
40	5	0	5.71	0.000	0	4.89	0.000	0	3.58	0.001	20	1.17	0.007	10	0.76	0.003	0	3.62	0.001	70	0.10	0.008
	10	0	5.73	0.002	0	4.79	0.000	0	3.71	0.002	10	1.26	0.023	40	0.81	0.005	0	3.87	0.002	50	0.57	0.017
	15	0	6.95	0.000	0	4.94	0.000	10	3.00	0.001	0	1.79	0.011	30	0.79	0.004	0	3.73	0.001	60	0.38	0.012
	20	0	5.51	0.002	0	4.64	0.002	0	3.13	0.005	10	0.87	0.025	10	0.89	0.011	0	3.10	0.003	80	0.25	0.025
Average		0	5.97	0.001	0	4.81	0.001	3	3.36	0.002	10	1.27	0.016	23	0.81	0.005	0	3.58	0.002	65	0.33	0.016
50	5	0	6.23	0.003	0	6.42	0.003	0	4.12	0.003	0	1.40	0.039	40	0.87	0.011	0	3.41	0.003	60	0.39	0.056
	10	0	6.54	0.001	0	5.06	0.001	0	3.49	0.002	10	1.39	0.019	40	0.82	0.006	0	3.99	0.002	50	0.50	0.022
	15	0	6.45	0.001	0	5.08	0.001	0	2.85	0.002	10	0.85	0.023	30	0.35	0.007	0	2.75	0.002	60	0.27	0.025
	20	0	6.64	0.003	0	4.44	0.002	0	3.19	0.005	0	1.11	0.042	40	0.55	0.016	0	4.04	0.003	60	0.13	0.047
Average		0	6.46	0.002	0	5.25	0.002	0	3.41	0.003	5	1.19	0.031	38	0.64	0.010	0	3.55	0.003	58	0.32	0.038
60	5	0	5.37	0.002	0	6.20	0.000	0	3.71	0.006	10	1.25	0.073	10	0.52	0.014	0	3.09	0.005	90	0.09	0.072
	10	0	7.69	0.003	0	4.71	0.002	0	3.44	0.003	0	1.61	0.062	20	0.98	0.016	0	3.98	0.003	80	0.11	0.062
	15	0	7.10	0.003	0	4.86	0.003	0	2.71	0.005	0	1.21	0.075	20	0.83	0.019	0	3.40	0.006	80	0.15	0.062
	20	0	7.17	0.003	0	4.90	0.003	0	2.76	0.008	0	1.04	0.067	20	1.08	0.022	0	4.06	0.010	80	0.11	0.083
Average		0	6.83	0.003	0	5.17	0.002	0	3.15	0.005	3	1.28	0.069	18	0.85	0.018	0	3.63	0.006	83	0.11	0.070

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

Problem		GRH-2			RAJ			NEHDS+M			LWW			FCH			LCH			NM_{LS}		
n	m	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)	PS (%)	ARPD (%)	CPU Time (s)
100	20	0	8.59	0.011	0	5.73	0.008	0	3.06	0.014	0	1.33	0.365	10	0.74	0.059	0	3.47	0.027	90	0.03	0.259
	40	0	9.08	0.008	0	4.80	0.009	0	2.76	0.014	10	1.11	0.384	30	0.65	0.147	0	3.70	0.019	60	0.15	0.248
	60	0	8.46	0.005	0	4.38	0.011	0	3.33	0.016	0	0.99	0.421	30	0.72	0.273	0	3.09	0.025	70	0.17	0.246
Average		0	8.71	0.008	0	4.97	0.009	0	3.05	0.015	3	1.14	0.390	23	0.70	0.160	0	3.42	0.023	73	0.12	0.251
200	20	0	10.47	0.041	0	5.33	0.033	0	2.71	0.059	0	1.50	2.354	10	0.71	0.276	0	3.23	0.058	90	0.03	1.749
	40	0	11.34	0.042	0	5.67	0.047	0	2.77	0.067	0	1.26	2.623	20	0.46	0.587	0	3.28	0.075	80	0.11	1.841
	60	0	10.97	0.049	0	4.99	0.051	0	2.91	0.066	0	1.38	2.820	0	0.54	1.161	0	3.50	0.091	100	0.00	1.797
Average		0	10.93	0.044	0	5.33	0.044	0	2.80	0.064	0	1.38	2.599	10	0.57	0.674	0	3.34	0.074	90	0.05	1.796
300	20	0	11.50	0.088	0	4.84	0.055	0	2.47	0.117	0	1.33	9.346	10	0.85	0.677	0	2.98	0.133	90	0.00	5.819
	40	0	12.53	0.094	0	5.42	0.072	0	2.90	0.126	0	1.58	7.658	0	0.83	1.414	0	3.61	0.167	100	0.00	5.830
	60	0	12.49	0.106	0	5.10	0.080	0	2.81	0.145	0	1.58	8.806	0	0.83	2.560	0	3.47	0.172	100	0.00	5.856
Average		0	12.17	0.096	0	5.12	0.069	0	2.72	0.129	0	1.50	8.603	3	0.84	1.550	0	3.35	0.157	97	0.00	5.835
400	20	0	12.62	0.162	0	4.95	0.150	0	2.55	0.208	0	1.61	19.684	0	0.82	1.200	0	3.25	0.262	100	0.00	13.436
	40	0	13.42	0.189	0	5.31	0.183	0	2.57	0.236	0	1.50	20.785	0	0.79	2.540	0	3.56	0.283	100	0.00	13.438
	60	0	13.88	0.208	0	5.11	0.204	0	2.65	0.261	0	1.45	22.913	10	0.64	4.594	0	3.42	0.307	90	0.01	13.633
Average		0	13.31	0.186	0	5.12	0.179	0	2.59	0.235	0	1.52	21.128	3	0.75	2.778	0	3.41	0.284	97	0.00	13.502
500	20	0	13.10	0.231	0	5.35	0.159	0	2.53	0.331	0	1.57	37.303	0	0.85	2.030	0	3.09	0.427	100	0.00	26.266
	40	0	14.43	0.267	0	5.05	0.200	0	2.53	0.365	0	1.42	40.855	10	0.72	3.967	0	3.37	0.465	90	0.02	26.320
	60	0	14.54	0.306	0	4.92	0.239	0	2.59	0.406	0	1.45	43.474	0	0.84	7.441	0	3.45	0.490	100	0.00	26.359
Average		0	14.02	0.268	0	5.11	0.199	0	2.55	0.367	0	1.48	40.544	3	0.80	4.479	0	3.30	0.461	97	0.01	26.315
600	20	0	13.42	0.338	0	4.67	0.404	0	2.60	0.488	0	1.43	72.272	0	0.94	3.086	0	3.07	0.651	100	0.00	45.847
	40	0	14.91	0.390	0	5.20	0.451	0	2.57	0.538	0	1.43	75.635	0	0.79	5.773	0	3.31	0.732	100	0.00	45.785
	60	0	14.90	0.437	0	5.25	0.516	0	2.78	0.593	0	1.44	69.397	0	0.84	10.560	0	3.40	0.786	100	0.00	46.015
Average		0	14.41	0.388	0	5.04	0.457	0	2.65	0.540	0	1.43	72.435	0	0.85	6.473	0	3.26	0.723	100	0.00	45.882
700	20	0	13.56	0.466	0	4.83	0.325	0	2.47	0.786	0	1.43	124.477	0	0.88	4.307	0	3.15	1.008	100	0.00	73.434
	40	0	15.05	0.535	0	4.90	0.412	0	2.55	0.864	0	1.49	113.147	0	0.79	8.296	0	3.39	1.092	100	0.00	73.860
	60	0	15.82	0.604	0	5.12	0.479	0	2.60	0.950	0	1.46	127.402	0	0.77	14.627	0	3.45	1.162	100	0.00	73.574
Average		0	14.81	0.535	0	4.95	0.405	0	2.54	0.867	0	1.46	121.675	0	0.81	9.077	0	3.33	1.087	100	0.00	73.623
800	20	0	14.19	0.683	0	4.86	0.739	0	2.39	1.059	0	1.51	171.118	0	0.79	6.326	0	3.18	1.387	100	0.00	111.602
	40	0	15.63	0.774	0	5.08	0.852	0	2.59	1.172	0	1.75	177.957	0	0.95	11.374	0	3.62	1.493	100	0.00	111.581
	60	0	15.50	0.871	0	4.69	0.925	0	2.60	1.276	0	1.46	186.183	0	0.91	19.391	0	3.49	1.557	100	0.00	111.768
Average		0	15.11	0.776	0	4.88	0.839	0	2.53	1.169	0	1.58	178.419	0	0.88	12.364	0	3.43	1.479	100	0.00	111.650

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.

Chaudhry, I. A. and Khan, A. M. (2012), Minimizing makespan for a no-wait flowshop using genetic algorithm, *Sãdhanã*, **37**(6), 695-707.

Ding, J.-Y., Song, S., Gupta J. N. D., Zhang, R., Chiong, R., and Wu, C. (2015), *An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem*, **30**, 604-613.

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. and Nagano, M. S. (2008), Evaluating the performance for makespan minimisation in no-wait flowshop sequencing, *Journal of Materials Processing Technology*, **97**, 1-9.

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

Gangadharan, R. and Rajendran, C. (1993), Heuristic algorithms for scheduling in the no-wait flowshop, *International Journal of Production Economics*, **32**, 285-290.

Grabowski, J. and Pempera, J. L. (2005), Some local search algorithms for no-wait flow-shop problem with makespan criterion, *Computers and Operations Research*, **32**, 2197-2212.

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.

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.

Laha, D., Gupta, J., and Sapkal, S. (2013), A penalty-shift-insertion-based algorithm to minimize total flow time in no-wait flow shops, *Journal of the Operational Research Society*, 1-13.

Laha, D. and Sapkal, S. (2014), An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop, *Computers and Industrial Engineering*, **67**, 36-43.

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.

Li, X. and Wu, C. (2008), Heuristic for no-wait flow shops with makespan minimization based on total idle-time increments, *Science in China Series F-Information Science*, **51**(7), 896-909.

Liu, B., Wang, L., and Jin, Y. H. (2007), An effective hybrid particle swarm optimization for no-wait flow shop scheduling, *International Journal of Advanced Manufacturing Technology*, **31**(9/10), 1001-1011.

Nagano, M. S. and Miyata, H. H. (*In press*), Review and classification of constructive heuristics for no-wait flow shop problem, *International Journal of Advanced Manufacturing Technology*, DOI: 10.1007/s00170-015-8209-5.

Nagano, M. S., Silva, A. A., and Lorena, L. A. N. (2012), A new evolutionary clustering search for a no-wait flow shop problem with set-up times, *Engineering Applications of Artificial Intelligence*, **25**, 1114-1120.

Nagano, M. S., Silva, A. A., and Lorena, L. A. N. (2014), An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times, *Expert Systems with Applications*, **41**, 3628-3633.

Nagano, M. S., Miyata, H. H., and Araújo, D. C. (2015), A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times, *Journal of Manufacturing Systems*, **36**, 224-230.

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.

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 and Operations Research*, **35**, 2807-2839.

Pan, Q.-K., Wang, L., and Zhao, B.-H. (2012), An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion, *International Journal of Advanced Manufacturing Technology*, **38**,778-786.

Papadimitriou, C. and Kanellakis, P. C. (1980), Flowshop scheduling with limited temporary storage, *Journal of the Association for Computing Machinery*, **27**, 533-549.

Rajendran, C. (1994), A no-wait flowshop scheduling heuristic to minimize makespan, *Journal of the Operational Research Society*, **45**, 472-478.

Reddi, S. S. and Ramamoorthy, C. V. (1972), On the flowshop sequencing problems with no wait in process, *Operational Research Quarterly*, **23**, 323-331.

Rock, H. (1984), Some new results in flow shop scheduling, *Zeitschriflfiir Operations Research*, **28**, 1-16.

Samarghandy, H. and Elmekkawty, T. Y. (2012), A genetic algorithm and particle swarm optimization for no-wait flow shop problem with separable setup times and makespan criterion, *International of Advanced Manufacturing Technology*, **61**, 1101-114.

Samarghandy, H. and Elmekkawty, T. Y. (2014), Solving the no-wait flow-shop problem with sequence-dependent set-up times, *International Journal of Computer Integrated Manufacturing*, **27**(3), 213-228.

Samarghandy, H. (2015), Studying the effect of server side-constraints on the makespan of the no-wait flow shop problem with sequence-dependent set-up times, *International Journal of Production Research*, **53** (9), 2652-2673.

Vallada, E., Ruiz, R., and Framinanan, J. V. (2015), New Hard Benchmark for Flowshop Scheduling Problems Minimising makespan, *European Journal of Operational Research*, **240**(3), 666-677.

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.

Ying, K.-C., Lee, Z.-J., Lu, C.-C., and Lin, S.-W. (2012), Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups, *The International Journal of Advanced Manufacturing Technology*, **58**(5~8), 671-682.