A NEW HEURISTIC METHOD FOR MINIMIZING THE MAKESPAN IN A NO-IDLE PERMUTATION FLOWSHOP

Marcelo Seido Nagano
EESC/USP
Avenida Trabalhador São-Carlense, 400, São Carlos/SP
drnagano@usp.br

Fábio José Ceron Branco
EESC/USP
fbranco@hotmail.com

ABSTRACT

This paper deals with the no-idle permutation flowshop sequencing in order to minimize the total time of completing the schedule (makespan). A heuristic denoted by NB is introduced, which is compared with the best heuristics reported in the literature. An extensive computational experiment was carried out for the performance evaluation of the proposed heuristic. Experimental results clearly show that the presented heuristic provides better solutions than those from the three best existing ones.

KEYWORDS: Production scheduling. No-idle flowshop. Heuristics.

Main area: AD & GP - OR in Administration & Production Management.

1. Introduction

The general flowshop scheduling problem is a production problem where a set of n jobs has to be processed in 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 in the machines may not be preempted. The jobs usually have the same sequencing on all machines. This processing environment is known as permutation flowshop. If job passing is not allowed, and all jobs have equal 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 paper is basically concerned with solving a variant of the permutation flowshop scheduling problem where no-idle times are allowed in machines. The no-idle constraint refers to an important practical situation in the production environment, where expensive machinery is used. There are wide applications of no-idle permutation flowshop scheduling (NIPFS) problems, especially in the fiberglass processing (Kalczynski and Kamburowski, 2005) and foundry operations (Saadani, Guinet and Moalla, 2003). Numerous practical examples of job scheduling in on-idle environments are shown in the article by Tasgetiren et al. (2011).

In the no-idle permutation flowshop, there must be no idle intervals between the processing of any consecutive operations in each machine. That is, each machine must process jobs without any interruption from the start of processing the first job to the completion of processing the last job. Therefore, when needed, the start of processing the first job on a given machine must be delayed in order to meet the no-idle requirement. Here we denote it with the well-known three fold notation of $F_m/prmu, no-idle/C_{max}$. The computational complexity of the $F_m/prmu, no-idle/C_{max}$ problem is briefly commented on in Taneav et al. (1994). The $NP-Hardness$ of the $\mathrm{F_3/prmu, no-idle/C_{max}}$ problem was proved by Baptiste and Hguny (1997) and Saadani et al. (2003).

Cepek et al. (2000) reported some mistakes in the paper by Adiri and Pohoryles (1982). Narain and Bagga (2003) studied the $\mathrm{F_3/prmu, no-idle/C_{max}}$ problem. The same problem with three machines is studied by Saadani et al. (2003), where a lower bound and an efficient heuristic are presented. This new heuristic favours the earlier method of the authors (Saadani et al. 2001). They published this work later on in Saadani et al. (2005). Kamburowski (2004) further enhanced the idea in Saadani et al. (2003) by proposing a network representation. Saadani et al. (2005) proposed a heuristic for the $\mathrm{F_3/prmu, no-idle/C_{max}}$ problem based on the traveling salesman problem (TSP). Narain and Bagga (2005a, 2005b) studied the $\mathrm{F_2/prmu, no-idle/C_{max}}$ and $\mathrm{F_m/prmu, no-idle/C_{max}}$ problems in two similar papers, respectively. Kalczynski and Kamburowski (2005) developed a heuristic, named KK heuristic, for the $\mathrm{F_m/prmu, no-idle/C_{max}}$ problem with a time complexity of $\mathrm{O(n^2m)}$. The authors also presented an adaptation of the NEH heuristic (Nawaz et al. 1983) for the NIPFS problem. In addition to the above, Kalczynski and Kamburowski (2007) studied the interactions between the no-idle and no-wait flowshops. Baraz and Mosheiov (2008) introduced an improved greedy algorithm consisting of a simple greedy heuristic and an improvement step.

As well-known in recent years, meta-heuristics have attracted increasing attention to solve scheduling problems owing to the fact that they are able to provide high quality solutions with reasonable computational effort. In addition to the above literature, in two similar papers, Pan and Wang (2008a and 2008b) proposed a discrete differential evolution (DDE) and discrete particle swarm optimization (DPSO) algorithms for the same problem. In both papers, a speed-up scheme for the insertion neighbourhood is proposed, which reduces the computational complexity of a single insertion neighbourhood scan from $\mathrm{O(n^3m)}$ to $\mathrm{O(n^2m)}$ when the insertion is carried out in order. The speed-up they proposed is based on the very well-known accelerations presented by Taillard (1990) for the insertion neighbourhood for the PFSP. In fact, both in DDE and DPSO, an advanced local search form, which is an IG algorithm proposed by Ruiz and Stutzle (2007), is used as a local search. Both DDE and DPSO used the well known benchmark

suite of Taillard (1993) by treating them as the NIPFS instances in order to test the results. In both papers, the authors tested the proposed methods against the heuristics of Baraz and Mosheiov (2008) and Kalczynski and Kamburowski (2005). More recently, Ruiz et al. (2009) presented an IG algorithm for the NIPFS problem with the makespan criterion. They used their own benchmark standard and examined the performance of IG in detail compared to the existing heuristics and meta-heuristics from the literature. From the heuristics tested, the authors highlight two heuristics adapted for the NIPFS problem: the FRB3 from Rad et al. (2009) and GH_BM2 with accelerations, based on the two phases of GH_BM from Baraz and Mosheiov (2008).

Goncharov and Sevastyanov (2009) proposed several polynomial time heuristics based on a geometrical approach for the $\mathrm{Fm/no-idle/C_{max}}$, however, neither computational experiments nor comparisons were provided.

Tasgetiren et al. (2011) presented the use of a continuous algorithm for the no-idle permutation flowshop scheduling (NIPFS) problem with tardiness criterion. A differential evolution algorithm with a variable parameter search (vpsDE) is developed to be compared to a well-known random key genetic algorithm (RKGA) from the literature. The research presented the following contributions. First of all, a continuous optimisation algorithm is used to solve a combinatorial optimisation problem, where some efficient methods of converting a continuous vector to a discrete job permutation and vice versa are presented. Secondly, a variable parameter search is introduced for the differential evolution algorithm which significantly accelerates the search process for global optimization and enhances the solution quality. Thirdly, some novel ways of calculating the total tardiness from makespan are introduced for the NIPFS problem. The computational results show its highly competitive performance when compared to RKGA. It is shown in this paper that the vpsDE performs better than the RKGA, thus providing an alternative solution approach to the literature in which the RKGA can be well suited.

Deng and Gu (2011) proposed a hybrid discrete differential evolution (HDDE) algorithm for $\mathrm{Fm/no-idle/C_{max}}$ and a novel speed-up method based on network representation is proposed to evaluate the whole insert neighborhood of a job permutation and employed in HDDE. Moreover, an insert neighborhood local search is modified effectively in HDDE to balance global exploration and local exploitation. Experimental results show that HDDE is better than the existing state-of-art algorithms.

The main objective of this paper is to introduce a new heuristic for minimizing makespan in a no-idle permutation flowshop production environment. In the following section, we present a property concerning this scheduling problem, which is used for the development of the new heuristic. A computational experiment was carried out in order to compare the proposed heuristic with best methods found in the literature. Finally, we end the paper with some conclusions.

2. Useful property of the no-idle permutation flowshop sequencing problem

For any n-job sequence σ, the makespan $M(\sigma)$ (Figure 1) can be expressed by:

$$M(\sigma) = \sum_{j=1}^{n} p_{1j} + \sum_{k=2}^{m} p_{kn} + \sum_{k=1}^{m-1} y_n^k \qquad (1)$$

Where:

p_{kj}: processing time on machine k of the job in the jth position of σ;

y_j^k: waiting time for the job in the jth position of σ, between the end of the operation on machine k and the beginning of the operation on machine $(k+1)$.

Of course, we do not know the waiting times y_j^k in advance unless we have considered a particular job sequence. However, given an arbitrary pair (J_u, J_v) of adjacent jobs (J_u immediately precedes J_v), we can calculate a lower bound on the waiting time for job J_v between the end of its operation on machine k and the beginning of machine $(k+1)$, regardless of the position of these two jobs.

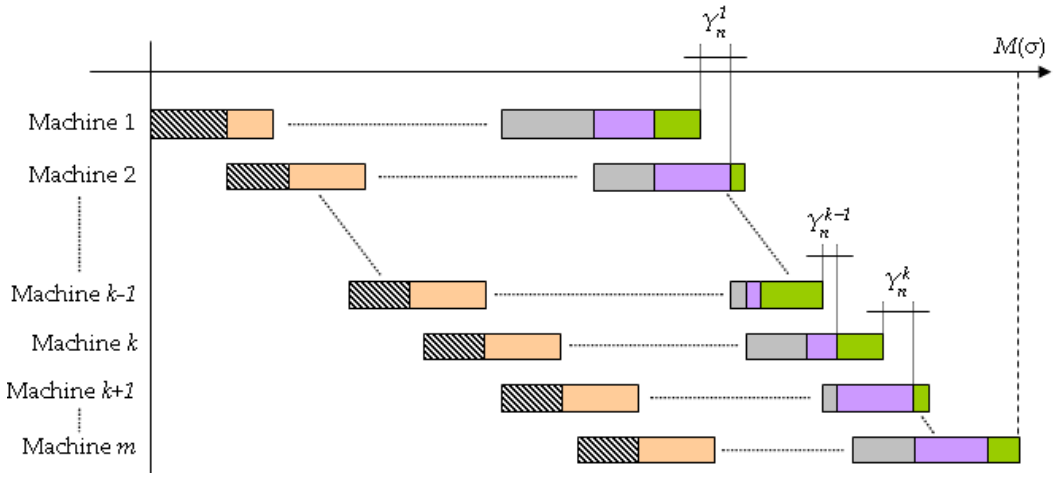


Figure 1 – Total time to complete a general permutation no-idle flowshop scheduling (makespan $M(\sigma)$).

Based on an investigation made by Nagano and Moccellin (2002), the following can be stated:

-Let u and v be any two jobs from the set of n jobs. For an arbitrary sequence σ with jobs u and v respectively scheduled in positions j and $(j+1)$, $j = 1, 2, \dots, n-1$, one has that:

$$LBY_{uv}^k = max[0, (p_{k+1\ u} - p_{k\ v}) - UBX_{u\ v}^k] \qquad (2)$$

Where:
LBY_{uv}^k: a Lower Bound for Y_{uv}^k;
Y_{uv}^k: waiting time for job v between the end of its operation in machine k and the beginning of the operation in machine $(k+1)$, when job u immediately precedes job v;
$p_{k+1\ u}$: processing time of job u on machine $(k+1)$;
$p_{k\ v}$: processing time of job v on machine processing time of job u on machine k;
$UBX_{u\ v}^k$: an Upper Bound for X_{uv}^k (Moccellin, 1995) given by $UBX_{u\ v}^k = max[0, UBX_{u\ v}^{k-1} + (p_{k-1\ v} - p_{k\ u})]$, where $UBX_{u\ v}^1 = 0$, and X_{uv}^k=idle time of machine k between the end of job u and start of job v.

-Let u and v be two arbitrary jobs from the set of n jobs. For any sequence σ with jobs u and v respectively scheduling in positions j and $(j+1)$, $j = 1, 2, \dots, n-1$. If we consider $UBX_{u\ v}^k = 0$, one has that:

$$LBY_{uv}^k = max[0, (p_{k+1\ u} - p_{k\ v})] \qquad (3)$$

Expression 3 is a Lower Bound for Y_{uv}^k for the no-idle permutation flowshop.

3. The proposed heuristic

The heuristic method we introduce has two stages. In the first one for each job u calculate I_u and arrange the jobs in non-descending order of I_u. The second stage uses the same iterative job insertion method of the NEH heuristic (Nawaz et al., 1983) with improved by using local search procedures based on both the Shift and Interchange Neighbourhoods of partial sequences.

The heuristic method is introduced in this paper, which is denoted NB. For the NB heuristic these steps are:

{Stage 1 – Initial arrangement for the jobs}

Step 1: For each job u calculate $I_u = \sum_{v=1}^{n}\left(\sum_{k=1}^{m-1} LBY_{uv}^{k}\right)$

where:

LBY_{uv}^{k}: the lower bound for Y_{uv}^{k}, given by expression (3).

Step 2: Arrange the jobs in non-descending order of I_u.

{Stage 2 – Construction & Improvement procedure}

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 $j = 3$ to n do

Step 4.1: Select the job in the j-th position of the list generated in Step 2 and find the best sequence by placing it in all j possible positions in the current best partial sequence. Denote this j-job sequence by S.

Step 4.2: Apply the insertion neighbourhood in sequence S. For each neighbour that is obtained by removing a determined job u and its insertion in a determined position of the sequence ($S[k]$, $k = 1, 2, \ldots, j$), verify if the condition $\sum_{k=1}^{m-1} LBY_{uv}^{k} \leq \sum_{k=1}^{m-1} LBY_{vu}^{k}$ is true, where $v \leftarrow S[k]$. If the condition is true and the makespan of the neighbor sequence is better than that of S assign it to S.

Step 4.3: Find the best sequence from the entire interchange neighbourhood 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.

The complexity of the heuristic is clearly determined by Stage 2, which can be executed in $O(n^3 m)$. Therefore, the complexity of the heuristic is $O(n^3 m)$.

4. Computational experience

The new heuristic developed in Section 3 has been compared with three of the best-known existing algorithms, that is, KK heuristic (Kalczynski and Kamburowski, 2005), GH heuristic (Baraz and Mosheiov, 2008), and FRB3 heuristic (Ruiz et al., 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 heuristics are tested in the well-known testbed by Taillard (1993). This testbed contains ten instances for a given combination of jobs and machines, i.e. n $\{20, 50, 100, 200, 500\}$ and $\{5, 10, 20\}$. This testbed has been consistently used for the problem (see Kalczynski and Kamburowski, 2005; Pan and Wang, 2008a; 2008b), which provides accurate upper bounds in order to test the performance of different approximate methods.

In the computational experiment, three traditional statistics are used in order to evaluate the heuristic performances: percentage of success (in finding the best solution), average relative percentage deviation (between the heuristics), and Average CPU time.

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 average relative percentage deviation $ARPD$ 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 M, the RPD obtained by a heuristic h on a given instance is computed as follows:

$$RPD_h = \frac{(M_h - M_*)}{M_*} \cdot 100 \qquad (4)$$

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

Table 2 – Comparison of results in Taillard testbed for the different heuristics in terms of PS, ARPD and Average CPU time (seconds)

	Heuristic			
Number of jobs n	GH	KK	FRB3	NB
20	[a]0.000	3.333	40.000	63.333
	[b]20.048	3.614	1.101	0.424
	[c]0.097	0.004	0.007	0.004
50	0.000	6.667	33.333	66.667
	20.949	1.658	0.698	0.252
	0.160	0.066	0.135	0.130
100	0.000	10.000	30.000	70.000
	21.242	1.096	0.292	0.166
	0.362	0.932	1.952	2.020
200	0.000	0.000	30.000	70.000
	20.879	1.034	0.389	0.067
	0.815	29.771	38.585	39.762
500	0.000	10.000	10.000	80.000
	21.962	0.459	0.312	0.006
	5.014	2164.908	1999.442	2019.725
Average	0.000	5.833	31.667	68.333
	20.870	1.803	0.614	0.222
	0.708	185.621	173.575	175.476

[a]Percentage of success
[b]Average relative percentage deviation
[c]CPU times (seconds)

Table 2 shows the comparative evaluation of the proposed heuristic NB, the GH heuristic, KK heuristic, and the FRB3 based on the PS, ARPD and Average CPU time.

As can be seen from the results in Table 2, the heuristic NB obtains better results than the rest of the heuristics.

It can be observed that GH gives rather poor results when compared to the other methods, but it is many times faster, on average. The performance of the NB heuristic with respect to the PS has been found to be better than the existing methods.

KK gives good results, better than GH, but yields rather poor results when compared to o FRB3.

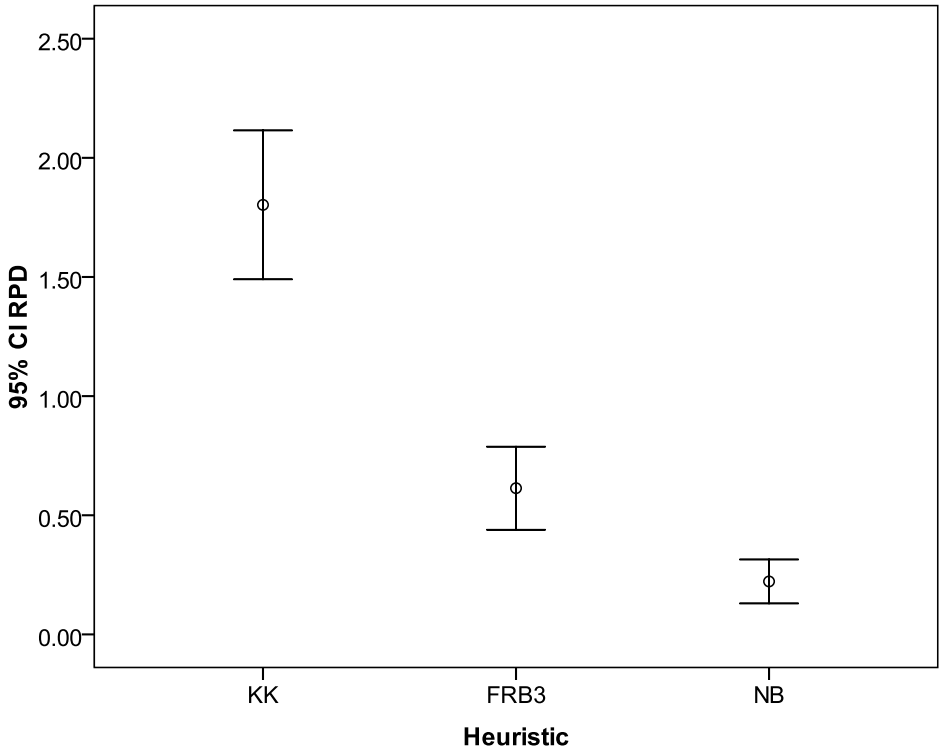
The last three methods (KK, FRB3 and NB) in the comparison show good results. NB, for example, gives the lowest ARPD among all the tested heuristics. The CPU times needed, however, are the third highest after GH and FRB3, but lower than the KK.

The average CPU times presented in Table 2 shows an obvious result. The NB computation time should be greater than the CPU time for FRB3 as a consequence of the arithmetic process required for obtaining the job indexes Step 1, and the solution process using local search procedures based on both the Shift and Interchange Neighborhoods of partial

sequences Step 4. However, the differences between average CPU times have not reached 2 seconds.

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 Figure 2, not considering the GH method.

Figure 2 – Means and 95% confidence intervals for the KK, FRB3 and NB algorithms



A Tukey honestly significant difference test was conducted to determine which means differ (see Table 3). The results indicate that the differences between the NB, KK and FBR3 are statistically significant, with the exception of GH which was not considered in the analysis.

Table 3 – Results of Tukey honestly significant difference tests

Heuristic φ	Heuristic ψ	Mean difference (φ- ψ)	Std. Error	Significance
KK	FRB3	1.18897*	0.15237	0.000
	NB	1.58038*	0.15237	0.000
FRB3	KK	-1.18897*	0.15237	0.000
	NB	0.39141*	0.15237	0.029
NB	KK	-1.58038*	0.15237	0.000
	FRB3	-0.39141*	0.15237	0.029

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

The results in Table 3 also indicate that three types of heuristics may be identified. These three heuristics have no similarity. For all the methods which were evaluated except for the GH method, the CPU times were not significantly different. The results show that the NB heuristic takes less computational time than that of the KK heuristic and comparable CPU time than that of the FRB3 heuristic. Finally, our NB heuristic is statistically better than the rest of the heuristics, although it is more time consuming.

5. Final remarks

In this paper, we dealt with the problem of scheduling a no-idle permutation flowshop with a makespan objective by means of heuristics methods. However, the search for near optimal solutions by using efficient and simple heuristics still remains as future research, taking into account that the problem is NP-hard. We propose a new heuristic for the problem, which is much better (in terms of quality of the solutions) than the existing ones and whose performance is comparable to that of a successful local search method for the problem while requiring much less CPU time. Our results have shown to be statistically significantly better than those produced by the best deterministic methods known to date, while maintaining the competitive algorithmic complexity. Henceforth, it can be concluded that the proposed heuristic is more efficient than existing heuristics for the problem.

References

Adiri I., Pohoryles D. (1982). Flowshop no-idle or no-wait scheduling to minimize the sum of completion times. Naval Research Logistics 29, 495–504.

Baptiste P., Hguny, L. K. (1997). A branch and bound algorithm for the $F/no-idle/C_{max}$. Proceedings of the international conference on industrial engineering and production management (IEPM97), Lyon, France, 429–438.

Baraz D., Mosheiov G. (2008). A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time. European Journal of Operational Research, 184, 810–813.

Cepek O., Okada M., Vlach M. (2000). Note: On the two-machine no-idle flowshop problem. Naval Research Logistics 47, 353–358.

Deng G, Gu X. (2011). A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion. Computers & Operations Research. doi:10.1016/j.cor.2011.10.024.

Goncharov Y., Sevastyanov S. (2009). The flow shop problem with no-idle constraints: A review and approximation. European Journal of Operational Research 196, 450-456.

Kalczynski P. J., Kamburowski J. (2005). A heuristic for minimizing the makespan in no-idle permutation flowshop. Computers and Industrial Engineering 49, 146-154.

Kalczynski P. J., Kamburowski J. (2005). A heuristic for minimizing the makespan in no-idle permutation flow shops. Computers & Industrial Engineering 49, 146–154.

Kalczynski P. J., Kamburowski J. (2007). On no-wait and no-idle flow shops with makespan criterion. European Journal of Operational Research 178, 677–685.

Kamburowski J. (2004). More on three-machine no-idle flow shops. Computers & Industrial Engineering 46, 461–466.

Moccellin J. V. (1995). A new heuristic method for the permutation flow shop scheduling problem. Journal of the Operational Research Society 46, 883-886.

Nagano M. S., Moccellin J. V. (2002). A high quality solution constructive heuristic for flow shop sequencing. Journal of the Operational Research Society 53, 1374-1379.

Narain L., Bagga P. C. (2003). Minimizing total elapsed time subject to zero total idle time of machines in n × 3 flowshop problem. Indian Journal of Pure & Applied Mathematics 34, 219–228.

Narain L., Bagga P. C. (2005a). Flowshop/no-idle scheduling to minimise the mean flowtime. Anziam Journal 47, 265–275.

Narain L., Bagga P. C. (2005b). Flowshop/no-idle scheduling to minimize total elapsed time. Journal of Global Optimization 3, 349–367.
Nawaz M., Enscore Jr E. E., Ham I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA, The International Journal of Management Science 11, 91–95.
Pan Q.-K., Wang L. (2008a). No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. International Journal of Advanced Manufacturing Technology 39, 796-807.
Pan Q.-K., Wang L. (2008b). A novel differential evolution algorithm for noidle permutation flow-shop scheduling problems. European Journal of Industrial Engineering 2, 279–297.
Rad S. F., Ruiz R., Boroojerdian N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. OMEGA, the International Journal of Management Science 37, 331–345.
Ruiz R., Vallada E., Fernandez-Martinez C. (2009). Scheduling in flowshops with no-idle machines. Computational intelligence in flowshop and job shop scheduling. Berlin, Heidelberg, Springer-Verlag, 21-51.
Saadani H., Guinet A., Moalla M. (2003). Three stage no-idle flowshops. Computers and Industrial Engineering 44, 425–434.
Saadani N. E. H., Guinet A., Moalla M. (2001). A travelling salesman approach to solve the F/no−idle/Cmax problem. In Proceedings of the International Conference on Industrial Engineering and Production Management, IEPM'01, volume 2, pages 880–888, Quebec, Canada.
Saadani N. E. H., Guinet A., Moalla M. (2005). A travelling salesman approach to solve the F/no−idle/Cmax problem. European Journal of Operational Research 161, 11–20.
Saadani N. H., Guinet A., Moalla M. (2003). Three stage no-idle flow-shops, Computers and Industrial Engineering 44, 425–434.
Taillard E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. European Journal of Operational Research 47, 67–74.
Taillard E. (1993). Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285.
Tanaev V. S., Sotskov Y. N., Strusevich V. A. (1994). Scheduling theory. Multi-stage systems. Dordrecht: Kluwer Academic Publishers.
Tasgetiren M. F., Pan Q-K., Suganthan P. N., Chua T. J. (2011). A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. International Journal of Production Research 49, 5033-5050.