International Journal of Research in Advent Technology, Vol.6, No.9, September 2018
E-ISSN: 2321-9637
Available online at www.ijrat.org

Minimizing Makespan in Flow Shop under Continuous Machine Operation

Kewal Krishan Nailwal[1], Deepak Gupta[2], Kawal Jeet[3]
Department of Mathematics, Apeejay College of Fine Arts, Jalandhar, Punjab,India[1]
Department of Mathematics and Humanities, M.M. University Mullana, Ambala, Haryana, India[2]
Department of Computer Science, D.A.V. College, Jalandhar, Punjab, India[3]
kk_nailwal@yahoo.co.in[1], guptadeepak@yahoo.co.in[2], kawaljeet80@gmail.com[3]

Abstract- The role of production scheduling is indispensible in the environment of manufacturing and service industries. Continuous machine operation or No-idle time on machines refers to the environment of production scheduling in which the machines once started work continuously to process all the jobs from first to last. The classical problem in flow shop scheduling is the minimization of makespan under different environments. The problem of makespan minimization in flow shop scheduling under continuous machine operation or no-idle situation is NP-hard. The heuristic proposed in this paper works on the principle of basic insertion to solve the flow shop problem for processing *n*-jobs on *m*-machines with continuous machine operation. The proposed constructive heuristic is compared with important heuristics solving no-idle problem that exists in flow shop literature. The computational result on Taillard's benchmark problems shows the superiority of the heuristic proposed over other heuristics.

Keywords- Flow shop; scheduling; constructive heuristic; no-idle.

1. INTRODUCTION

Production scheduling plays a vital role in manufacturing and service industries. For improving the production efficiency of the system, one must use effective and efficient solutions for manufacturing and scheduling technologies for the rapidly changing market demands having great competitive environment. According to the different industrial setups in flow shop, there prevail different constraints either on the jobs or on the machines to complete the task. The two important constraints in flow shop scheduling problem are No-wait of jobs and No-idle or Zero-idle time on machines. The No-wait flow shop scheduling problem (NWFSP) is an important branch of scheduling problems which related to 'zero buffer'. It arises due to the characteristics of jobs processing producing certain products, where the processing of jobs demands the processing to be continuous. The continuous flow shop originates in the scheduling theory because of the production environment in industry. In many flow shops, the production environment is such that the delay in job processing between the subsequent machines is not allowed i.e. the assumption of infinite storage capacity between the machines in flow shop is no longer valid. For maintaining the continuous flow of job, the processing of jobs is delayed on the first machine so that the jobs do not wait in the subsequent processing on machines. Various applications can be found in the industry related to this type of flow shop. For example, the process of making iron sheets in industry involves the no-wait situation as the sequence in which the jobs are processed after the heating of iron is to be continuous so that the temperature of heated iron falls within the permissible interval specified. This constraint is necessary for the defect free production of iron sheets and making the good quality product. Also, in food processing industry, the food is canned immediately after the food is prepared so that the food quality is maintained. However, to maintain freshness in the food the continuous flow in the sequence of jobs processing is maintained throughout the process. In contrast to no-wait, the no-idle situation in some production scheduling occurs if the operation of the machines is not permitted to interrupt once started as their interruption causes increase in cost or decreases the benefits. The reason for this is the expensive parts used in the operation of machines with cost depending upon the actual time of operating or the environment of the manufacturing system does not allow. Such type of scheduling problem is known as no-idle flow shop scheduling problem. The problem of no-idle time of machines in the scheduling literature is also referred to Continuous Machine Operation (CMO). To maintain the no-idle environment in flow shop, the machines needs to be delayed for processing the first job so that idle time on machines for job processing is reduced to zero from start of processing to completion of jobs. Clear examples are the steppers used in the production

of integrated circuits by means of photolithography. Other examples come from sectors where less expensive machinery is used but where machines cannot be easily stopped and restarted. Ceramic roller kilns, for example, consume a large quantity of natural gas when in operation. Idling is not an option because it takes several days to stop and to restart. Other such example includes the production of glass fiber. The temperature inside the furnace for molten made from glass is maintained at high temperature, the furnace once stopped halts the production as it takes time in days to retain that temperature [7].

Adiri and Pohoryles [1] was first to study the *m*-machine no-idle situation in flow shop and showed that the problem $F_2 / prmu / C_{\max}$ can be referred as the problem $F_2 / prmu, no-idle / C_{\max}$ i.e. these two problems are equivalent. The solution of this can be obtained by applying Johnson [6] algorithm. For details relating to permutation (*prmu*) flow shop, one may refer to Pinedo [17]. Narain and Bagga [12] also developed an algorithm based on branch and bound method for two stage flow shop scheduling problem with objective as total flow time under no-idle condition. The problem related to $F_3 / prmu, no-idle / C_{\max}$ have been studied extensively some researchers. Baptiste and Hguny [3] gave a integer linear programming model for solving $F_m / prmu, no-idle / C_{\max}$, and a corresponding branch and bound algorithm; also see Saadani et al. [21]. Saadani et al. [19] explains the environment of no-idle in a three-stage production of engine blocks in a foundry which includes the casting of sand moulds and sand cores. Narain and Bagga [2] formulated the three stage flow shop problem minimizing the total elapsed time with zero idle time on machines. He illustrated that it is not necessary that the sequence having minimum total elapsed time also possesses the same under no-idle constraint. They discussed the concept of sequencing in rental situations under specific rental policy. The idle time on the first machine is always zero as first machine is required only for the time equal to the sum of the processing time of all the jobs on it. Under the circumstances of scarcity of funds, the machines have to be taken on rent or hired in order to complete the assignments. Hiring of machines enable saving working capital, give option for having the equipment, and allow the up-gradation to new technology. Minimization of the total hiring cost of machines would be the criterion in these types of situations. In this direction, they did some work and explored the area for solving this problem [14].

Kamburowski [9] identified the simple network representation of total elapsed time which efficiently solves the three machine flow shop problem under no-idle situation. Besides this, the results of three machine problem are extended to *m*-machine no-idle flow shop problem. Woollam [24] studied the problem of no-idle time of machines using various benchmark heuristics like NEH heuristic [15] when modified to no-idle problem and other heuristics known for makespan problems. Narain and Bagga [13] gave four algorithms using branch and bound technique to solve *n*-job, *m*-machine flow shop problem under no-idle situation with objective of minimizing total elapsed time in which processing time of jobs follow certain relationship. Corresponding to these algorithms, theorems have been proved to show when one should hire the machine on rent to minimize total rental cost of machines. Saadani et al. [20] presented the model of $F_m / prmu / C_{\max}$ problem with travelling salesman approach by using the nearest insertion technique and implemented with the help of the solver. Kalczynski and Kamburowski [7] proposed a improved heuristic based on the Johnson algorithm which outperforms the heuristic of Saadani et al. [21] and the modified NEH for no-idle constraint. Apart from this heuristic, Kalczynski and Kamburowski [8] have discussed no-wait and no-idle flow shops scheduling with makespan criterion and found that makepan with no-idle constraint is not the increasing function of processing times of jobs. Goncharov and Sevastyanov [5] have given review and approximation on the flow shop problem with no-idle constraints. Ruiz et al. [18] studied the variant of the classical permutation flow shop problem with makespan criterion with no-idle constraint. Baraz and Mosheiov [4] proposed the improved greedy (IG) heuristic with an improvement step involving pairwise exchange of jobs. Makuchowski [11] shows the significant correlation between the two variants of NEH algorithm for makespan with no-idle and without no-idle constraint i.e. the algorithm used for solving problem without no-idle can be effectively used for no-idle problems. Recently, Xiaoxia, and Yunhong [25] proposed a probability based model for solving mixed no-idle permutation flowshop scheduling problem.

Since, with advent of time the problem $F_m / prmu, no-idle / C_{\max}$ is also solved by using metaheuristics for getting high quality results. Various

papers such as Pan and Wang [16], Seido Nagano and Januário, [22], Zhou and Chen [27], Li et al. [10] and Yazdani and Naderi [26] used metaheuristics for solving $F_m / prmu, no-idle / C_{\max}$. This describes the significance of proposing improved constructive heuristic algorithm for providing the good initial solution to various metaheuristics. This is in turn implies that results of the metaheuristics can be improve further by the proposed constructive heuristic. In this work, a constructive heuristic is proposed and is compared with two important heuristics namely the improved greedy (IG) and Kalczynski-Kamburowski (KK) heuristic on various Taillard [23] instances. The computational result on Taillard's instances shows the superiority of the proposed heuristic (PH) over both the heuristics.

The rest of paper is organized as follows: the section two deals with problem formulation. Section three explains the proposed heuristic algorithm. In section four, the working of the proposed heuristic algorithm is explained. Section five reports the results along with analysis of the results obtained. At the end, we conclude the paper in section six followed by references.

2. PROBLEM FORMULATION

Let some job $i\,(1 \le i \le n)$ is to be scheduled on machine $j(1 \le j \le m)$ in the same technological order with criteria to be optimized as minimization of makespan under no-idle constraint i.e. $C_{\max}$. The general assumptions under which the *n*-jobs are processed through *m*-machines are: Pre-emption of jobs is not permitted. All jobs and machines are available at the beginning of processing. The machines are available throughout the processing and never breakdown. Each job is processed through each of the machine once and only once. Each machine can perform only one task at a time. The setup times of jobs on machines are negligibly small and hence can be ignored or are included in the processing time of jobs.

Let $t_{i,j}$ be the time of processing of the job i on the machine j, $T_i = \sum_{j=1}^{m} t_{i,j}$ be the sum total of processing times corresponding to job i on m machines. For calculating the makespan of a schedule of *n*-jobs in a no-idle situation concerning flow shop, we have to first find the time of initiation of machines so that the machines start processing jobs without any idle time on them. Following this, we denote this initiation of machines by L_j, j = {1,2,3,…..,*m*}. Clearly, $L_1 = 0$. We calculate the initiation of machines denoted by L_j as follows:

$$L_j = L_{j-1} + \max_{1 \le k \le n} \left\{ \sum_{i=1}^{k} t_{i,(j-1)} - \sum_{i=1}^{k-1} t_{i,j} \right\}, j = \{2,3,....,m\}$$

After the calculation of values of L_j , the makespan $C_{\max}$ can obtained from the following equation as:

$$C_{\max} = L_m + \sum_{i=1}^{n} t_{i,m} .$$

3. PROPOSED ALGORITHM

The steps formulated for algorithm concerning flow shop scheduling problem, processing *n*-jobs through *m*-machines with objective of minimizing makespan under no-idle constraint ($C_{\max}$) are given below.

Step1. Find the sum total of the processing time T_i of every job i (i=1,2,3,…,n) on the given *m*-machines by the expression: $T_i = \sum_{j=1}^{m} t_{i,j}$.

Step2. Exhibit the job list according to the decreasing values of T_i so obtained in step 1.

Step3. Take the first two jobs from the job list. Find the two-job partial sequence with minimum $C_{\max}$ out of the possible two job sequences. Select it as the current partial sequence. If there is a tie for $C_{\max}$, select that two job sequence for further construction of jobs in which the first job is with lower index to break the tie.

Step4. Take the next job from the job list and insert in all possible positions of the partial sequence obtained in step 3. Find the partial sequence with minimum $C_{\max}$. This is the current sequence for further construction of final sequence of jobs. Break the tie if any as per step 3.

Step5. Check if there is more than one job in the job list. If so, consider the next two jobs else go to step 4. Find the possible two-job partial schedule from these. Select the two-job partial sequence with minimum $C_{\max}$. Generate all the sequences by inserting the two-job partial sequence at all possible locations of the partial sequence so obtained in step 4. Select the sequence with minimum $C_{\max}$ as the current sequence. Next the first job of the latest pair of jobs is inserted in the all the possible locations of the current sequence to generate the possible sequences. If the any of these sequences have better result, then record that sequence as the current sequence. Break the tie if any as per step 3.

Step6. Repeat the step 4 and 5, if more jobs are present in the job list otherwise stop. The steps are repeated until all the jobs are scheduled. The sequence

so obtained is the final best sequence with minimum C_{max} .

4. NUMERICAL ILLUSTRATION

Consider the following flow shop instance with number of jobs, n = 5:

Table 1. Flow shop instance

Job *i*	**Machine *j***		
	Machine 1	**Machine 2**	**Machine 3**
1	94	47	95
2	43	10	31
3	90	69	77
4	47	22	31
5	76	89	2

As per the step 1, the sum total of the processing times for every job on the three machines is T_1 = 236, T_2 = 84, T_3 = 236, T_4 = 100, T_5 = 167. Following step 1, arrange all the 5 jobs in the order of decreasing values of T_i in the job list resulting in {1, 3, 5, 4, 2}. According to step 3, pick the first two jobs from the job list namely {1, 3}. The possible two job sequences resulting from {1, 3} are 1-3 and 3-1 with respective C_{max} = 356 and 356. Since, there is a tie for the selection of two job sequence for further construction of jobs. Therefore, picking the sequence 1-3 as per the step proposed to break the tie. This is the best current sequence. Now picking the next job 5 using step 4 and inserting it at all the possible locations of the current sequence generating the sequences 5-1-3, 1-5-3 and 1-3-5 having C_{max} = 432, 406 and 365 respectively. The best current sequence of jobs having minimum C_{max} = 365 is 1-3-5. Now, pick next two unscheduled jobs {4, 2} from the job list. Find the minimum C_{max} for the possible arrangement of {4, 2}. The C_{max} = 152 for the partial sequences 4-2 and 2-4. Therefore, inserting the partial two job sequence 4-2 as block in the current best partial sequence so obtained in the last step generating the sequences 4-2-1-3-5, 1-4-2-3-5, 1-3-4-2-5, 1-3-5-4-2 with C_{max} = 460, 485, 485, 433 respectively. The sequence 1-3-5-4-2 becomes the current best sequence with minimum makespan. Now, insert the first job {4} of the last job-block in the last best sequence to generate 4-1-3-5-2, 1-4-3-5-2, 1-3-4-5-2, 1-3-5-4-2 and 1-3-5-2-4 with C_{max} = 443, 452, 452, 433, 427. The job sequence 1-3-5-4-2 retain as the best sequence. Further the second job 2 of the last pair is selected for inserting at all the possible locations of the current sequence generating the sequences 2-1-3-5-4, 1-2-3-5-4, 1-3-2-5-4, 1-3-5-2-4 and 1-3-5-4-2 with C_{max} = 445, 460, 460, 427 and 433 respectively. The sequence 1-3-5-2-4 is the final best sequence with minimum C_{max} = 427.

5. COMPUTATIONAL RESULTS

IG (improved greedy) is a heuristic method put forward by Baraz and Mosheiov [4] for solving NFSP. Their numerical tests showed that the IG heuristic method performed better. KK is a heuristic developed by Kalczynski and Kamburowski [7] showed that the KK heuristic performed better than NEH when modified to no-idle (NEH_M) problem. For the performance evaluation, the proposed heuristic (PH) is tested against the IG, KK heuristic and the results are compared to the solution produced by modified NEH for no-idle (NEH_M) on 100-Taillard instances. These 100-Taillard's instances contain 10-subsets having 10-problems each.

Table 2. Makespan values for 20, 50-job Taillard instances

Problem Instances	**Makespan**		**Problem Instances**	**Makespan**	
20x5	**NEH_M**	**PH**	**50x5**	**NEH_M**	**PH**
Ta001	1413	1408	Ta031	3028	3038
Ta002	1456	**1432**	Ta032	3172	**3160**
Ta003	1278	1294	Ta033	2876	**2841**
Ta004	**1396**	1402	Ta034	3107	**3057**
Ta005	**1454**	1463	Ta035	3158	3174
Ta006	**1463**	1469	Ta036	3061	**3047**
Ta007	**1270**	1295	Ta037	3120	**3051**
Ta008	1341	**1301**	Ta038	3040	**3030**
Ta009	1467	**1417**	Ta039	2700	2701
Ta010	1262	**1230**	Ta040	2885	**2884**
Avg	1380	**1371.1**	Avg	3014.7	**2998.3**
20x10			**50x10**		
Ta011	2266	**2247**	Ta041	3652	**3528**
Ta012	2198	**2170**	Ta042	3643	**3639**
Ta013	2007	2029	Ta043	3676	**3556**
Ta014	2048	**1929**	Ta044	3640	3755
Ta015	**1799**	1867	Ta045	3507	**3507**
Ta016	**1835**	1915	Ta046	4021	4023
Ta017	1988	**1881**	Ta047	3753	**3702**
Ta018	1931	**1883**	Ta048	4092	**4082**
Ta019	2054	**2022**	Ta049	3640	**3585**
Ta020	2011	**1956**	Ta050	3901	**3867**
Avg	2013.7	**1989**	Avg	3752.5	**3724.4**

		.9			
20x20			**50x20**		
Ta021	3482	3599	Ta051	5774	5898
Ta022	3329	**3234**	Ta052	6367	6378
Ta023	3578	3589	Ta053	5676	**5671**
Ta024	**3227**	3313	Ta054	5968	5995
Ta025	**3433**	3531	Ta055	**5477**	5503
Ta026	3677	**3615**	Ta056	6300	**6146**
Ta027	3584	**3476**	Ta057	5979	**5969**
Ta028	3372	**3335**	Ta058	5785	**5766**
Ta029	3480	3510	Ta059	5639	**5628**
Ta030	3855	3943	Ta060	5800	**5859**
Avg	**3501.7**	3514 .5	Avg	**5876.5**	5881.3

Table 3. Makespan values for 100, 200-job Taillard instances

Problem Instances	**Makespan**		**Problem Instances**	**Makespan**	
100x5	**NEH_M**	**PH**	**100x20**	**NEH_M**	**PH**
Ta061	**5848**	5893	Ta081	9694	**9680**
Ta062	**5418**	**5418**	Ta082	8820	**8744**
Ta063	5355	5403	Ta083	9308	**9237**
Ta064	5321	5333	Ta084	**9619**	9664
Ta065	**5836**	**5836**	Ta085	9869	**9777**
Ta066	5399	**5383**	Ta086	**9049**	9219
Ta067	5735	5743	Ta087	9111	**9007**
Ta068	5195	5204	Ta088	**9005**	9027
Ta069	5588	5621	Ta089	9498	**9455**
Ta070	5511	5477	Ta090	**8825**	8810
Avg	**5520.6**	5531 .1	Avg	9279.8	**9262**
100x10			**200x10**		
Ta071	6853	**6849**	Ta091	**11840**	11860
Ta072	6258	**6189**	Ta092	**12560**	12641
Ta073	6817	**6756**	Ta093	**13341**	13386
Ta074	6783	**6629**	Ta094	11844	**11813**
Ta075	**6484**	6527	Ta095	12197	**12154**
Ta076	**6644**	6658	Ta096	**12015**	12019
Ta077	**6942**	6956	Ta097	11735	**11664**
Ta078	**7172**	7176	Ta098	**12397**	12419
Ta079	7371	**7232**	Ta099	11191	**11044**
Ta080	**7393**	7429	Ta100	**12624**	12649
Avg	6871.7	**6840 .1**	Avg	12174. 4	12164 .9

Both proposed heuristic and modified NEH for no-idle (NEH_M) are implemented in MATLAB-R2008a. The average relative percent deviation (ARPD) of the proposed heuristic (PH), IG and KK heuristic is calculated as the statistics for the performance measures.

Relative Percentage Deviation calculated as:

$$\text{Relative Percentage Deviation (RPD)} = \frac{Makespan_{heuristic} - Makespan_{NEH}}{Makespan_{NEH}} \times 100 ,$$

Where, $Makespan_{heuristic}$ is the value of the makespan obtained by the heuristic for a particular set of problems and $Makespan_{NEH}$ is the value of the makespan obtained from modified NEH for no-idle(NEH_M).

The ARPD results of proposed heuristic, IG [4] and KK heuristic [7] are compared and are shown in table 4. It can be observed that the proposed heuristic (PH) outperforms IG heuristic in all the 100-Taillard's instances considered. The proposed heuristic (PH) performs well than KK heuristic for job instances from 20x5 to 50x5 and for size 100x10 but KK features well for Taillard [23] problem instances of size 50x10 and larger except 100x10. For the overall average, it can be seen from table 4 that ARPD obtained by proposed heuristic (PH) algorithm is better than both IG and KK. That means that the makespan obtained by proposed heuristic (PH) algorithm is smaller than those by IG and KK.

Table 4. Comparison of IG, KK and Proposed Heuristic (PH)

Taillard problem instances	**Average Relative Percentage Deviation**		
	IG	**KK**	**PH**
20X5	9.19	0.87	**-0.63**
20X10	8.37	2	**-1.18**
20X20	5.4	1.29	**0.37**
50X5	11.53	-0.03	**-0.54**
50X10	12.34	**-1.79**	-0.75
50X20	12.44	**-0.55**	0.1
100X5	16.4	**-0.24**	0.19
100X10	13.98	0.08	**-0.46**
100X20	14.78	**-2.57**	-0.19
200X10	17.18	**-0.55**	-0.1
Overall Avg	12.16	-0.15	**-0.32**

6. CONCLUSION

In this paper, a constructive heuristic algorithm is proposed for solving no-idle flow shop scheduling problems. Under no-idle constraint the two more objective such as total flow time of jobs and mean flow time are also achieved as these problems

are equivalent under the no-idle constraint. The operational cost of the machines is also minimized as the machines are delayed in their operation and also they work continuously without waiting for any job. The proposed heuristic (PH) algorithm outperforms the two well known IG and KK heuristics existing in the literature on the 100-Taillard problem instances. The ARPD of the proposed heuristic (PH) is -0.32 as compared to the IG and KK heuristic with 12.16 and -0.15 respectively. The proposed heuristic (PH) algorithm therefore can be the taken as the good initial solution to the various metaheuristics, thereby improving upon the solutions. The main aim is to provide a good initial solution to various algorithms based on the metaheuristics so that the results of these metaheuristics can be improved further.

REFERENCES

[1] Adiri, I. and Pohoryles, D. "Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times". Naval Research Logistics Quarterly, 29(3), 495-504, 1982.
[2] Bagga, P.C. "Minimizing total elapsed time subject to zero total idle time of machines in n X 3 flowshop problem". Indian J. pure appl. Math, 34(2), 219-228, 2003.
[3] Baptiste, P. and Hguny, L. K. A branch and bound algorithm for the F/no− idle/Cmax. In Proceedings of the international conference on industrial engineering and production management, pages, 429-438. 1997.
[4] Baraz, D. and Mosheiov, G. "A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time". European Journal of Operational Research, 184(2), 810-813, (2008).
[5] Goncharov, Y. and Sevastyanov, S. "The flow shop problem with no-idle constraints: A review and approximation". European Journal of Operational Research, 196(2), 450-456, 2009.
[6] Johnson, S.M. "Optimal two-and three-stage production schedules with setup times included". Naval research logistics quarterly, 1(1), 61-68, 1954.
[7] Kalczynski, P.J. and Kamburowski, J. "A heuristic for minimizing the makespan in no-idle permutation flow shops". Computers & Industrial Engineering, 49(1), 146-154, 2005.
[8] Kalczynski, P.J. and Kamburowski, J. "On no-wait and no-idle flow shops with makespan criterion". European journal of Operational research, 178(3), 677-685, 2007.
[9] Kamburowski, J. "More on three-machine no-idle flow shops". Computers & Industrial Engineering, 46(3), 461-466, 2004.
[10] Li, L., Wu, X. and Wang, Z. "Research of no-idle flow shop scheduling based on improved bacteria foraging optimization algorithm". Computer Engineering and Applications, 17, 48, 2015.
[11] Makuchowski, M. "Permutation, no-wait, no-idle flow shop problems". Archives of Control Sciences, 25(2), 189-199, 2015.
[12] Narain, L. and Bagga, P.C. "Flowshop/no-idle scheduling to minimise the mean flowtime". The ANZIAM Journal, 47(2), 265-275, 2005.
[13] Narain, L. and Bagga, P.C. "Flowshop/no-idle scheduling to minimize total elapsed time". Journal of Global Optimization, 33(3), 349-367, 2005.
[14] Narian, L. and Bagga, P.C. "Scheduling problems in rental situation". Bulletin of Pure and Applied Sciences: Section E, 24, 2005.
[15] Nawaz, M., Enscore Jr, E.E. and Ham, I. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". Omega, 11(1), 91-95, 1983.
[16] Pan, Q.K. and Wang, L. " No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm". The International Journal of Advanced Manufacturing Technology, 39(7-8), 796-807, 2008.
[17] Pinedo, M.L. "Scheduling: theory, algorithms, and systems". Springer, 2016.
[18] Ruiz, R., Vallada, E. and Fernández-Martínez, C. "Scheduling in flowshops with no-idle machines. In Computational intelligence in flow shop and job shop scheduling" pages 21-51, Springer, Berlin, Heidelberg, 2009.
[19] Saadani, N.E.H. , Guinet, A. and Moalla, M. "Three stage no-idle flow-shops". Computers & industrial engineering, 44(3), 425-434, 2003.
[20] Saadani, N.E.H., Guinet, A. and Moalla, M. "A travelling salesman approach to solve the F/no-idle/Cmax problem". European Journal of Operational Research, 161(1), 11-20, 2005.
[21] Saadani, N.E.H., Baptiste, P. and Moalla, M. "The simple F2//C max with forbidden tasks in first or last position: A problem more complex that it seems". European journal of operational research, 161(1), 21-31, 2005.
[22] Seido Nagano, M. and Soriano Sampaio Januário, J.C. "Evolutionary heuristic for makespan minimization in no-idle flow shop production systems". Acta Scientiarum. Technology, 35(2), 271-278, 2013.
[23] Taillard, E. "Benchmarks for basic scheduling problems". European Journal of Operational Research, 64(2), 278-285, 1993.
[24] Woollam, C.R. "Flowshop with no idle machine time allowed". Computers & industrial engineering, 10(1), 69-76, 1986.
[25] Xiaoxia, Z. and Yunhong, L. "A tabu estimation of distribution algorithm to solve the mixed no-idle permutation flowshop scheduling problem". Computer Applications and Software, 1, 049, 2017.

[26] Yazdani, M. and Naderi, B. "Modeling and scheduling no-idle hybrid flow shop problems". Journal of Optimization in Industrial Engineering, 10(21), 59-66, 2016.
[27] Zhou, Y., Chen, H. and Zhou, G. "Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem". Neurocomputing, vol. 137, 285-292, 2014.