Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management
Istanbul, Turkey, July 3 – 6, 2012

Solving No-wait Flow shop by Heuristic Algorithm

GH. M. Komaki
Department of Mechanical, Industrial and Manufacturing Engineering
University of Toledo
2801 West Bancroft Street, Toledo, OH 43606, USA

Vahid Kayvanfar
Research Institute of Food Science and Technology
Mashhad 91735-139, Iran

Abstract

In this paper, we consider no-wait flow shop with makespan criterion and propose an effective heuristic algorithm to solve the problems. In order to verify effectiveness of this algorithm, three other heuristic algorithms have been compared with our proposed one. Computational results illustrate efficiency of our algorithm with respect to other ones.

Keywords
No-wait, flow shop, makespan, heuristic algorithm.

1. Introduction
This study deals with the basic *n*-job *m*-machine no-wait flow shop scheduling problem. In this problem, there are *n* jobs in which each job has *m* operations and must be processed on a set of series machines continually. Once a job is started on the first machine, it has to be processed continuously and without any interruption through completion at last machine. So, to satisfy the no-wait requirement the start of a job on first machine must be delayed when it is necessary. The No-wait flow shop can be found in real industrial environments including chemical processing (Rajendran 1994), food processing (Hall and Sriskandarayah 1996), steel production, nylon production, and pharmaceutical industry. For example, in steel production case, the heated metals must continuously go through the sequence of operations before getting cold and preventing the defects in the composition of final products.

In this study, we consider the makespan as optimization criterion. This problem is denoted as $F|prmu, no\text{-}wait|C_{max}$ and has been proved NP-hard by Garey and Johnson (1979). Therefore, many researchers instead finding the optimal solution have focused on finding the near optimal solution by applying heuristic and meta-heuristic algorithms which can find the near optimal solution in reasonable computational time.

The no-wait flow shop with makespan criterion has been studied by many researchers such as Rajendran (1994), Reddi and Ramamoorthy (1972), Laha and Chakraborty (2009). Rajendran (1994) proposed a constructive heuristic algorithm based on index development and then job inserting to construct the solution that generates good solutions within small computational time. Framinan and Nagano (2008) considered the no-wait with makespan and proposed heuristic algorithm based on the analogy between no-wait flow shop and Travelling Salesman (TSP). Laha and Chakraborty (2009) for this problem suggested a constructive heuristic algorithm. This algorithm is based on the NEH's (Nawaz et al. 1983) heuristic that shift neighborhood mechanism proposed by Osman and Potts (1989) has been embedded in it. Pan et al. (2008) investigated the same problem and proposed iterated greedy algorithm (IIGA). In this algorithm, they have applied a speed-up method for inserting the job in the available neighborhoods and then improved NEH heuristic algorithm to construct solutions in initial steps, then a simple local search algorithm incorporated into the iterated greedy algorithm to do exploitation. The IIGA has been compared with Rajendran (1994) and meta-heuristic algorithms such as Tabu Search (TS) and Discrete Particle Swarm Optimization (PSO). The comprehensive review of related researches about no-wait flow shop has been presented in Hall and Sriskandarayah (1996).

The rest of paper is organized as below. In the next section the considered problem has been described. Section 3 describes the proposed heuristic algorithm and in the Section 4 the computational results have been illustrated.

2. No-wait Flow shop Problem

In this section, the no-wait flow shop problem is explained. Used notations throughout the paper are as follows:

- n number of jobs to be processed
- m number of machines
- t_{ij} processing time of job J_j (j = 1, 2,…, n) on machine M_i (i = 1, 2, …, m).
- C_{max} completion time of last job on machine m
- d_{ij} minimum delay on the first machine between job J_j and J_i
- π, σ schedule (permutation) of the given jobs
- [j] job found in the j-th position of a sequence

The no-wait flow shop is a constraint flow shop that n jobs $\{J_1,J_2,\ldots,J_n\}$ must be processed on m machines$\{M_1,M_2,\ldots,M_m\}$. Each job J_j has m operations $\{O_{1j},O_{2j},\ldots,O_{mj}\}$ that these operations must be processed sequentially on machines without delay between adjacent machines. As mentioned earlier, the start of a job when is necessary to satisfy the no-wait constraint must be delayed on the first machine. This time can be calculated as below (Fink and Voß 2003):

$$d_{i,k} = \max_{1\le j\le m}\left\{\sum_{h=1}^{j} t_{i,h} - \sum_{h=2}^{j} t_{k,h-1}\right\} \quad (1)$$

Where t_{ij} represents the processing time of job i on machine j, $1 \le i \le n$, $1 \le j \le m$.
Having sequence of jobs, $\pi = \{J_{[1]}, J_{[2]},..., J_{[n]}\}$ where $J_{[i]}$ represents the job in the i-th position of the sequence, makespan can be computed as follow:

$$C_{\max} = \sum_{i=1}^{n-1} d_{[i][i+1]} + \sum_{h=1}^{m} t_{[n],h} \quad (2)$$

3. Heuristic Algorithm

Inspection of (2) reveals that the processing time of the job in the last position of the sequence and delays between adjacent jobs affect the makespan of any sequence of jobs. Therefore, the last job must have the minimum summation of processing time and jobs must have the minimum delays. According to this idea, an algorithm is proposed, named Heuristic algorithm Based on Delay (HABD). HABD has two phases, the first phase generates initial sequence of jobs and the second phase improves the quality of obtained sequence in the first phase by applying the change in the jobs positions to reduce the delay between adjacent jobs.

In the first phase, initially pair of jobs that have the least $C_{max}(i, j)$ is selected and considered as partial sequence σ.

$$C_{max}(i,j) = d_{ij} + \sum_{k=1}^{m} t_{kj} \quad (3)$$

Thereafter among the remained jobs, pair of jobs that have least increment in objective function, $\delta(i, j)$, are located before the jobs in σ as first job of the sequence.

$$\delta(i,j) = d_{ij} + d_{j[\sigma([1])]} \quad (4)$$

This procedure continues until no outstanding job is found, i.e., all jobs have been scheduled. The main idea of improvement phase is based on the reducing the delay between consecutive jobs. In this phase, the job that has the maximum delay with adjacent jobs, max {D (i)| $2 \le i \le n-1$} is selected and inserted in the position that has the least delay.

$$D(i) = \{d_{I_{[i-1]}I_{[i]}} + d_{I_{[i]}I_{[i+1]}}) | 2 \le i \le n-1\} \quad (5)$$

To insert the selected job, three cases may happen:

- Inserting the job into the first position of σ.
- Inserting the job into the last position of σ.
- Inserting the job into position $\{i| 2 \le i \le n-1\}$.

To determine which position is the best position to insert the selected job, it needs to compute the change in the objective function. To do this, consider that π is the current schedule with $C_{max}(\pi)$ and the selected job is J^* with *max* {D(i)} and is located in the *i*-th position of π; $\{2 \le i \le n-1\}$. Removing the J^* from π results to σ.

$$\pi = J_{[1]}, J_{[2]}, \dots, J_{[i-1]}, J_{[i]}, J_{[i+1]}, \dots, J_{[j-1]}, J_{[j]}, J_{[j+1]}, \dots, J_{[n]} \quad (6)$$

$$\sigma = \{J_{[1]}, J_{[2]}, \dots, J_{[i-1]}, \overline{J_{[i+1]}}, \dots, \overline{J_{[j-1]}}, \overline{J_{[j]}}, \dots, \overline{J_{[n]}}\} \quad (7)$$

Where jobs that are in position $J < [i]$ in σ and π, are the same but other jobs are different and $\overline{J_{[j+1]}} \equiv J_{[j]} | j > i$. Also it is obvious that π has *n* jobs and σ has $n-1$ jobs. By removing the J^* from π, we then have:

$$C_{max}(\sigma) = C_{max}(\pi) + d_{J_{[i-1]}\overline{J_{[i+1]}}} - (d_{J_{[i-1]}J^*} + d_{J^*J_{[i+1]}}) \quad (8)$$

As mentioned earlier, to insert job J^* in σ there are three options:

I. Inserting the job in the first position:

If the job J^* is inserted in the first position, only the delay between job J^* and the first job, $d_{J^*J_{[1]}}$, will be changed in the makespan, i.e.:

$$\pi^{New} = \{J^*, \sigma\} = \{J^*, J_{[1]}, J_{[2]}, \dots, J_{[i-1]}, \overline{J_{[i+1]}}, \dots, \overline{J_{[j-1]}}, \overline{J_{[j]}}, \dots, \overline{J_{[n]}}\} \quad (9)$$

$$C_{max}(\pi^{New}) = C_{max}(\sigma) + d_{J^*J_{[1]}} \quad (10)$$

II. Inserting the job in the last position

$$\pi^{New} = \{\sigma, J^*\} = \{J_{[1]}, J_{[2]}, \dots, J_{[i-1]}, \overline{J_{[i+1]}}, \dots, \overline{J_{[j-1]}}, \overline{J_{[j]}}, \dots, \overline{J_{[n]}}, J^*\} \quad (11)$$

If the job J^* is inserted in the last position, see (11), the change in makespan will be the delay between job J^* and the last job of π^{New}, $d_{\overline{J_{[n]}}J^*}$, and $\sum_{k=1}^{m} t_{kJ^*} - \sum_{k=1}^{m} t_{k\overline{J_{[n]}}}$, i.e.:

$$C_{max}(\pi^{New}) = C_{max}(\sigma) + d_{\overline{J_{[n]}}J^*} + \sum_{k=1}^{m}(t_{kJ^*} - t_{k\overline{J_{[n]}}}) \quad (12)$$

III. Inserting the job in position *i* where $2 \le i \le n-1$

In this case, the job J^* must be inserted between jobs that have the least increment in the objective function. To find the best possible position $\min_{1 \le i \le n-1}\{d_{[i]J^*} + d_{J^*[i+1]}\}$ must be found. By finding the position and inserting the job into position *i*, we have:

$$\pi^{New} = \{\sigma, J^*\} = \{J_{[1]}, J_{[2]}, \dots, J_{[i-1]}, \overline{J_{[i+1]}}, \dots, \overline{J_{[j-1]}}, \overline{J_{[j]}}, \dots, \overline{J_{[n]}}, J^*\} \quad (13)$$

$$C_{max}(\pi^{New}) = C_{max}(\sigma) + \min_{1 \le i \le n-1}\{d_{[i]J^*} + d_{J^*[i+1]}\} \quad (14)$$

Now, among these cases; a, b and c, the π^{New} that has the least $C_{max}(\pi^{New})$ must be selected. If this $C_{max}(\pi^{New})$ is less than $C_{max}(\pi)$, π^{New} it will be accepted as new sequence and then D(i) will be updated, otherwise the next job that have the least D(i) must be selected.

These procedures continue until the stop criteria is met, i.e. the number of iteration with-out improvement reaches to $K = n-1$.

The outline of proposed algorithm is shown as follows:

Phase 1: Initial sequence

1) Set $\sigma = \emptyset, U = \{1,2,\dots,n\}, K = 0$,

2) Compute the delay matrix according to Eq. (2).

3) For each job pairs (i,j) compute the partial objective functions $C_{max}(i,j), C_{max}(j,i)$, $i,j \in U$ where $C_{max}(i,j)$ means that job Ji processes before job Jj on all machines, as below:

$$C_{max}(i,j) = d_{[i][j]} + \sum_{k=1}^{m} t_{kj} \quad (15)$$

4) Select the minimum partial objective function, $(x,y) = min_{i,j \in U}\{C_{max}(i,j)\}$.

5) $K \leftarrow K + 2$

6) Add the selected pair of jobs to the first positions of σ and drop those jobs from U.

7) If K = n go to Step 11, otherwise go to Step 8.

8) If n−K =1 add the remained job in the U to the first position of σ, σ(1), go to Step 11.

9) For each pair jobs (i,j) ∈ U compute increment of objective function as below

$$\delta(i,j) = d_{[i][j]} + d_{[i][\sigma(K)]} \quad (16)$$

10) Select the pair of jobs (i,j) with the minimum increment $(x,y) = min_{i,j\in U}\{\delta(i,j)\}$. Go to Step 5.
11) Stop. The σ is the final sequence.

Phase II: Improvement mechanism

1) $K = 1, U = \emptyset, \pi^c = \sigma, \sigma^{best} = \sigma, C_{max}^{best} = C_{max}(\sigma)$

2) For jobs in π^c compute $D(i) = \{d(J_{[i-1][i]} + d(J_{[i][i+1]} | 2 \le i \le n-1\}$.
3) Sort D(i) nonascendingly and store them in U.
4) While K < n −1:
4.1) Select the *K*-th job in U. Remove it from π^c and insert it into the best possible position in the π^{New}. Now, compute these values:

$$A = d_{J[i]J} + d_{JJ[i+1]} \quad (17)$$

$$B = \min\left\{\begin{matrix} d_{JJ[1]} \\ d_{J[n-1]J} + \sum_{k=1}^{m}(t_{kJ} - t_{k[n-1]}) \\ min_{1\le i\le n-1}\{d_{J[i]J} + d_{JJ[i+1]}\} \end{matrix}\right\} \quad (18)$$

4.2) If $B + d_{J[i]J[i+1]} > A$ go to Step 4.3; othewise

- If B=$d_{JJ[1]}$, the best position for job *J* is the first poisition, move the job *J* to the first position.
- If $B = d_{J[n-1]J} + \sum_{k=1}^{m}(t_{kJ} - t_{kJ[n-1]})$; the best position for job *J* is last position, move the job *J* to the last position.
- If $B = min_{1\le i\le n-1}\{d_{J[i]J} + d_{JJ[i+1]}\}$, the best poisition for job *J* is between job *J*[*i*] and *J*[*i*+1]. Insert the job *J* after job *J*[*i*].

Update σ^{best}, C_{max}^{best}, *K*=1 and go to Step 2.

4.3) $K \leftarrow K+1$

5) The σ^{best} is the final sequence and C_{max}^{best} is the final objective.

To clarify the HABD, an example has been solved. Consider a problem with 6 jobs and 4 machines. The processing time has been shown in Table 1.

Phase I.

Table 1: Jobs processing times on machines

job	Machine			
	1	2	3	4
1	62	90	61	43
2	29	39	24	62
3	74	17	98	62
4	87	13	5	92
5	67	63	37	12
6	79	48	71	86

Using (1), we compute the delay matrix as shown in the below.

Delay Matrix of Jobs

$$d = \begin{pmatrix} - & 164 & 122 & 151 & 89 & 86 \\ 29 & - & 29 & 49 & 29 & 29 \\ 74 & 159 & - & 146 & 84 & 74 \\ 87 & 105 & 87 & - & 87 & 87 \\ 68 & 101 & 76 & 74 & - & 67 \\ 79 & 192 & 107 & 179 & 117 & - \end{pmatrix}$$

The makespan of partial sequence of (i, j) is shown in Table 2 When job 6 is the first job of the partial sequence and job 4 comes after that, the makespan is equal to 376, i.e., $C_{max}(6,4) = 376$. The minimum makespan belongs to the partial sequence of (2,5). So the first job is 2 and the second one is 5, i.e., $\sigma = \{2,5\}$.

Table 2: Makespan of the partial sequence of jobs

	1	**2**	**3**	**4**	**5**	**6**
1	-	318	373	348	268	370
2	285	-	280	246	208	313
3	330	313	-	343	263	358
4	343	259	338	-	266	371
5	324	255	327	271	-	351
6	335	346	358	376	296	-

Among the remained jobs, we compute the $\delta(i,j)$ and based on this calculation, the next jobs are (6,1) and the new partial sequence is $\sigma = \{2,5,6,1\}$. In the second iteration, the partial sequence is $\sigma = \{2,5,6,1,4,3\}$ and the makespan is 606.

Phase II.

Using (5) for obtained sequence in Phase I, we have $D = \{96,146,230,238\}$. The maximum D is 238 which means the job 4 is candidate to change its position. Removing the job 4 and inserting it into all possible positions, the best sequence is $\sigma = \{2,5,6,1,3,4\}$ and makespan is 582.

Now, set K=1 and go to Step 2, update D.

$D = \{96,146,201,268\}$ which means job 3 is the best candidate to change its position. The new sequence is $\sigma = \{2,3,5,6,1,4\}$ and makespan is 568. By repeating these steps the final sequence will be $\sigma = \{2,3,6,1,5,4\}$ with makespan 542.

4. Experimental Results

In this section, we evaluate the performance of HABD. To do so, a test-bed is built based on the common setting of number of jobs and machine (see e.g. Rajendran (1994)). The number of jobs is $n = \{10,20,30,40,50,60,70,80\}$ and the number of machines is $m = \{5,10,15,20\}$. Also the possessing time of jobs uniformly distributed between 1 and 99. For each problem size 100 instances are generated. To evaluate the performance of HABD, the same problems have been solved to compare with Rajendran's heuristic (1994) (named RAJ), Framinan and Nagano (2008) (named FN) and heuristic algorithm proposed by Laha and Chakraborty (2009) (called LC). All algorithms are coded in Matlab 2010a and run on a PC with a 2.66 GHz Intel Core 2 Duo processor and 4 GB RAM memory. The computational results are shown in Table 3.

Table 3: Computational results

n	m	HABD		Raj		FN		LC	
		MOF	***MCT***	***MOF***	***MCT***	***MOF***	***MCT***	***MOF***	***MCT***
10	5	819.5	0.002	864.4	0.007	838.0	0.010	831.1	0.019
	10	1294.4	0.002	1370.7	0.007	1312.6	0.003	1335.4	0.005
	15	1677.1	0.002	1787.6	0.008	1739.9	0.003	1787.4	0.005
	20	1968.8	0.003	2107.4	0.008	1980.2	0.004	1977	0.005
Average		**1439.9**	**0.002**	**1532.5**	**0.007**	**1467.7**	**0.005**	**1482.7**	**0.008**
20	5	1510.5	0.007	1543.7	0.036	1524.7	0.011	1540.9	0.016
	10	2121.4	0.003	2252	0.023	2166.3	0.005	2195.7	0.008
	15	2657.1	0.003	2882.8	0.018	2748.1	0.005	2720.6	0.007
	20	3142.8	0.002	3231.9	0.021	3187.2	0.004	3168.9	0.006
Average		**2357.9**	**0.004**	**2477.6**	**0.024**	**2406.6**	**0.006**	**2406.5**	**0.009**
30	5	2267.3	0.010	2336.3	0.055	2272.9	0.012	2245.9	0.016
	10	3037	0.006	3211.6	0.035	3106.3	0.013	3027.4	0.022
	15	3652.1	0.007	3862.5	0.043	3748.3	0.012	3692.9	0.018
	20	4214.2	0.006	4357.3	0.043	4313.2	0.011	4336.1	0.017
Average		**3292.6**	**0.007**	**3441.9**	**0.044**	**3360.2**	**0.012**	**3325.6**	**0.018**
40	5	2838.1	0.008	2884	0.109	2886.4	0.016	2894.3	0.028
	10	3817.2	0.008	3881.5	0.091	3803.9	0.016	3911.2	0.028
	15	4652.2	0.009	4708.9	0.085	4785.7	0.014	4623.2	0.031
	20	5208.8	0.010	5392.6	0.080	5339.7	0.019	5400.5	0.031
Average		**4129.1**	**0.009**	**4216.7**	**0.091**	**4203.9**	**0.016**	**4207.3**	**0.029**
50	5	3496.1	0.014	3641.4	0.140	3538.7	0.027	3657.3	0.048
	10	4563.2	0.013	4712.8	0.124	4647.1	0.030	4695.6	0.049
	15	5462.2	0.014	5702.3	0.119	5588.9	0.029	5680.6	0.050
	20	6338	0.016	6480	0.135	6413.6	0.027	6420.3	0.050
Average		**4964.9**	**0.014**	**5134.1**	**0.129**	**5047.1**	**0.028**	**5113.4**	**0.049**
60	5	4113.2	0.022	4310.9	0.210	4219.8	0.023	4174.9	0.032
	10	5339.8	0.023	8096.1	0.169	5456.3	0.022	5520.8	0.031
	15	6392.2	0.024	6563.7	0.189	6536.2	0.025	6541	0.033
	20	7230.9	0.026	7598.9	0.161	7356	0.028	7432.6	0.034
Average		**5769.0**	**0.024**	**6642.4**	**0.182**	**5892.1**	**0.024**	**5917.3**	**0.032**
70	5	4746.6	0.030	4819	0.263	4782.5	0.035	4780.4	0.043
	10	6118.3	0.032	6333.3	0.244	6217.1	0.027	6278	0.046
	15	7362.3	0.034	7563.5	0.253	7489.1	0.033	7552.7	0.047
	20	8232.6	0.035	8687.1	0.268	8413.7	0.057	8557.1	0.094

Average		**6614.9**	**0.033**	**6850.7**	**0.257**	**6725.6**	**0.038**	**6792.0**	**0.057**
80	5	5381.6	0.042	5530.3	0.383	5474	0.103	5487.4	0.174
	10	6901.9	0.047	7012.9	0.378	6934.2	0.107	7086.4	0.179
	15	8120.5	0.048	8612.2	0.372	8138.2	0.107	8393.1	0.178
	20	9248.1	0.052	9559.8	0.393	9514.3	0.109	9516.9	0.178
Average		**7413.0**	**0.047**	**7678.8**	**0.381**	**7515.2**	**0.106**	**7620.1**	**0.177**
Total Average		**4497.7**	**0.017**	**4746.8**	**0.139**	**4577.3**	**0.029**	**4608.1**	**0.047**

Where MOF is mean of objective function and MCT is mean of computational time. As can be seen in Table III, our HABD outperforms all other methods with respect to MOF in average. Also HABD spends less time as compared to other techniques in average. According to the computational results, one can claim that our proposed HABD is a good and efficient algorithm for such a no-wait flow shop problem.

References

Fink, A., and Voß, S., Solving the continuous flow-shop scheduling problem by metaheuristics, *European Journal of Operational Research*, vol. 151, pp. 400–414, 2003.

Framinan, J. M., and Nagano, M. S., Evaluating the performance for makespan minimisation in no-wait flowshop sequencing, *Journal of materials processing technology*, vol. 197, pp. 1-9, 2008.

Garey, M.R., and Johnson, D.S., Computers and intractability, aguide to the theory of NP-completeness. Freeman, San Francisco,1979.

Hall, N.G., and Sriskandarayah, C., A survey of machine scheduling problems with blocking and no-wait in process, *Operations Research*, vol. 44, pp. 510–525, 1996.

Laha, D., and Chakraborty, U.K., A constructive heuristic for minimizing makespan in no-wait flow shop scheduling, *International Journal of Advanced Manufacturing Technology*, vol. 41, pp. 97–109, 2009.

Nawaz, M., Enscore, E., and Ham, I., A heuristic algorithm for the m-machine, n-job flowshop sequencing problem, *OMEGA International Journal of Management Science*, vol. 11, pp. 91–95,1983.

Osman, I.H., and Potts, C.N., Simulated annealing for permutation flow-shop scheduling, *Omega*, vol. 17, pp. 551-557,1989.

Pan, Q.-K., Wang, L., and Zhao, B.-H., An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion, *International Journal of Advanced Manufacturing Technology*, vol. 38, pp. 778–786, 2008.

Rajendran, C., A no-wait flowshop scheduling heuristic to minimize makespan, *Journal of Operations Research Society*, vol. 45, pp. 472–478, 1994.

Reddi, S.S., and Ramamoorthy, C.V., On the flowshop sequencing problems with no wait in process, *Operations Research*, Q. 23, pp. 323–331, 1972.