International Journal of Computational Intelligence Systems

An adaptive learning approach for no-wait flowshop scheduling problems to minimize makespan

Orhan Engin *
Department of Industrial Engineering,
Faculty of Engineering and Architecture,
University of Selçuk, Konya, 42079 Turkey
E-mail:orhanengin@yahoo.com

Cengiz Günaydın
Department of Industrial Engineering,
Faculty of Engineering and Architecture,
University of Selçuk, Konya, 42079 Turkey
www.selcuk.edu.tr

Received: 10 / 03 / 2011
Accepted: 02 / 05 / 2011

Abstract

No-wait flowshop scheduling problem (NW-FSSP) with the objective to minimize the makespan is an important sequencing problem in the production plans and applications of no-wait flowshops can be found in several industries. In a NW-FSSP, jobs are not allowed to wait between two successive machines. The NW-FSSPs are addressed to minimize makespan and the NW-FSSP is known as a NP- Hard problem. In this study, Agarwal et al.'s[1] adaptive learning approach (ALA) is improvement for NW-FSSPs. Improvements in adaptive learning approach is similar to neural-network training. The improvement adaptive learning approach (IALA) is applied to all of the 192 problems. The proposed IALA method for NW-FSSP is compared with Aldowaisan and Allahverdi's[2] results by using Genetic heuristic. The results of computational experiments on randomly generated NW-FSSPs are show that the proposed adaptive learning approach performs quite well.

Keywords: No-wait flowshop; Adaptive learning approach; Genetic algorithm; Makespan

1. Introduction

In the permutation flowshop, *n* different jobs have to be processed on *m* machines. Each job has one operation on each machine and all jobs have the same ordering sequencing on each machine. At any time, each machine can process at most one job. Preemption is not allowed[3]. An important kind of flowshop scheduling problem is characterized by a no-wait environment[4]. In a NW-FSSP, jobs are not allowed to wait between two successive machines. This implies that the starting time of a job at the first machine has to be delayed to ensure that the job can go through the flow shop without having to wait for any machine[5]. The NW-FSSP has important applications in many industries. Examples include the metal, plastic and chemical industries, food and pharmaceutical industries. Additional applications can be found in advanced manufacturing environments,

* Corresponding Author: Orhan Engin
E-mail: orhanengin@yahoo.com, Tel: +90-332-2232039, Fax:+90-332-2410635

such as just-in-time and flexible manufacturing systems[6].

The NW-FSSP has attracted the attention of many researchers. Reddi and Ramamoorthy[7] and Wismer[8] were the first to address the NW-FSSP with the single criterion of makespan. Bonney and Gundry[9] and King and Spachis[10] have later developed heuristics. Gangadharan and Rajendran[11] and Rajendran[12] have developed additional heuristics, and showed that their heuristics outperform those of Bonney and Gundry[9] and King and Spachis[10]. Bertolissi[13] presented a heuristic algorithm for NW-FSSPs to minimize the sum of the total flowtimes and at the same time minimizes the average processing time. Aldowaisan[14] developed a new heuristic method for two-machine no-wait flowshop problem with separate setup times from processing times and sequence independent. Aldowaisan and Allahverdi[2] presented two new heuristics that are based on simulated annealing and genetic algorithm techniques for no-wait flowshops to minimize makespan by incorporating a modified Nawaz-Enscore-Ham (NEH) heuristic (see Nawaz et al.[15]), which were shown to outperform those of Gangadharan and Rajendran[11] and Rajendran[12]. Later, Aldowaisan and Allahverdi[6] presented several new heuristics for the *m*-machine no-wait flowshop with total completion time as the criterion. Shyu et al.[16] developed an ant colony optimization for NW-FSSP to minimize the total completion time. Wang and Cheng[4] proposed a heuristic approach for two-machine no-wait flowshop scheduling with due dates and class setups. Lin et al.[17] developed a heuristic genetic algorithm for NW-FSSP. The authors compared the performance of the developed heuristic genetic algorithm with the heuristic proposal by Aldowaisan and Allahverdi[6]. Rahimi-Vahed[18] considered a multi-objective scatter search for a bi-criteria NW-FSSP in which weighted mean completion time and weighted mean tardiness are to be minimized simultaneously. Pan et al.[19] proposed an iterated greedy algorithm for NW-FSSP with the objective to minimize the makespan and also Pan et al.[20] presented a discrete particle swarm optimization for the NW-FSSP with both makespan and total flowtime criteria. Tavakkoli-Moghaddam et al.[21] proposed an immune algorithm for a multi-objective NW-FSSP by minimizing the weighted mean completion time and weighted mean tardiness simultaneously. Laha and Chakraborty[22] presented a new constructive heuristic, based on the principle of job insertion, for minimizing makespan in NW-FSSPs. Qian et al.[23] proposed a memetic algorithm based on differential evolution for the multi-objective NW-FSSPs. Later, Qian et al.[24] proposed an effective hybrid differential evolution for the NW-FSSP with the makespan criterion.

The aim of this paper is to suggest an adaptive learning approach for NW-FSSP with makespan criteria. The performance of the proposed approach is tested on a set of 192 instances. The comparison is made with the Aldowaisan and Allahverdi's[2] results by using Genetic heuristic. The rest of this article is organized as follows. Section 2 provides adaptive learning approach. Genetic heuristic approach is given in Section 3. Section 4 presents computational results. Finally, the conclusions about adaptive learning approach are presented in Section 5.

2. Adaptive Learning Approach

Adaptive learning approach is similar to neural-network training. Numerous neural network models have been developed for solving optimization problem in the past two decades. The most popular seminal network model for optimization problem is Hopfield and Tank[25] model. During the past decade, numerous neural network models have been developed for solving the scheduling problems. Park et al.[26] proposed an approach for solving a parallel-machine scheduling problem applying a known heuristic rule combining it with a neural network technique. Lee and Shaw[27] presented a neural-net approach for real time flow-shop sequencing. They apply the neural-net approach to construct a sequence for a set of jobs that arrive in different job combinations over time. Fonseca and Navaresse[28] proposed an artificial neural network for the traditional job-shop simulation. Solimanpur et al.[29] proposed a neural networks-based tabu search method, for the flow shop scheduling problem. Agarwal et al.[1] propose an improvement-heuristic approach for the general flow-shop problem based on the idea of adaptive learning. Also, Agarwal et al.[30] proposed new heuristics along with an augmented-neural-network formulation for solving the makespan minimization task-scheduling problem for the non-identical machine environment. Akyol and Bayhan[31] presented a review on evolution of production scheduling with neural networks.

In this study, Agarwal et al.'s[1] adaptive learning approach (ALA) is improvement for no-wait flowshop scheduling problems. The general idea is that Agarwal et al.[1] define a weight factor associated with each operation. They use weighted processing times instead of regular processing times and also weights are modified using a certain strategy.

3. Genetic Heuristic Approach

Genetic algorithm (GA) is a stochastic optimization technique for solving the scheduling problem. GAs are based on natural selection and genetics. They were first developed by John Holland[32] in 1975. GAs use a

collection of solutions called population. Each individual in the population is called a chromosome and a chromosome represents a solution to the problem. The chromosomes can be produced through successive iterations, called generations also the chromosomes are evaluated using the value of fitness function during each generation[33].

In this study the improvement learning approach for NW-FSSP is compared with Aldowaisan and Allahverdi's[2] results by using Genetic heuristic (GEN-2). A brief outline of the Aldowaisan and Allahverdi's[2] Genetic heuristic is given below, where the notation *P(t)* denotes the population at the *t*th generation, $s_i(t)$ represents the *i*th job sequence in *P(t)* and $f(s_i(t))$ is the fitness value of $s_i(t)$.

Specify GAs parameters
Population size POPSIZE,
Number of generations NGEN,
Probability of crossover PCROSS,
Probability of mutation PMUTE,
Sequence fit tolerance FITTOLER,
Population fitted percent parameter POPFIT,
End
For initial generation
Generated by Gangadharan and Rajendran[11] heuristic
Generated by Dannenbring[34] heuristic
Generated by randomly
Next:
Calculate the fitness value $f(s_i(t))$
Calculate the selection probability $p_i(t)$
Do
Select two jobs sequences for reproduction according to their selection probability
Add it initial population
Do
Choose the parents for PCROSS
Crossover
Calculate the fitness value $f(s_i(t))$
Do
Choose the parents for PMUTE
Mutation
Calculate the fitness value $f(s_i(t))$
Loop until the new generation is full
Saved the best sequence obtained at any generation
Loop until reach NGEN or fitted the new generation reaches POPFIT
While stopping criteria= false.

Table 1 lists the parameters used in Aldowaisan and Allahverdi's[2] best Genetic heuristic(GEN-2).

Table 1. Best GEN-2[2] Parameters List

Parameter	**Value**
Population size	95
Number of generations	60
Fitted tolerance	1E-10
Population fitted percent	60
Probability of crossover	0.725
Probability of mutation	0.009

The Aldowaisan and Allahverdi's[2] best GEN-2 heuristic is summarized in Table 2.

Table 2. Best GEN-2 heuristic method[2]

Heuristic	**Initial solution**	**New solution generation**
GEN-2	Gangadharan and Rajendran[11] Dannenbring[34] Campbell Dudek Smith[35] Random	Roulette wheel selection, Partially Mapped Crossover-PMX Swapping

4. Computational Results

In this study, to evaluate the performance of the improvement adaptive learning approach for no-wait flowshop scheduling, 192 problems are generated. The processing times on each machine were randomly generated from a discrete uniform distribution from the different intervals for, *a* type problems [1, 10], *b* type problems [1, 50] and *c* type problems [1, 100] which is commonly used in the literature e.g[16, 2, 12]. Also the values of setup time, s, on each job were randomly generated from a discrete uniform distribution the interval [1, 10] which is commonly used in the literature.

The number of jobs (*n*) considered is *8, 10, 12, 50, 100, 150, 200, 250*, and the number of machines (*m*) is 2, 3, 5, 8, 10, 15, 20, and 25 which is commonly used in the literature. The improvement Agarwal et al.'s[1] IALA for no-wait flowshop scheduling problems notation and algorithm is given below.

Notation

n	Set of *n* jobs
m	Set of *m* machines
O_{ij}	Operation of *i*th job on *j*th machine
T_{ij}	Processing time of operation O_{ij}
W_{ij}	Weighted associated time of operation O_{ij}
WT_{ij}	Weighted processing time of operation O_{ij}
k	Iteration number

K_{max}	Max number of iteration
MS_k	Makespan in the kth iteration
s_i	Set up time of ith on each machine
RF	Reinforcement factor
$TINI$	Tolerate iterations with no improvement
α	Learning rate
BMS	Best makespan
BW_{ij}	Best weights
RND	Random number

The proposed IALA method can be summarized as follows;

Set IALAs value:
Learning rate:
Reinforcement factor:
Tolerate iterations with no improvement:
Set initialize weight
Set generation size:
Set CPU time:
End
Calculate $WT_{ij} = W_{ij} * T_{ij}$
Generated by CDS
Generated by Palmer heuristic (see Palmer[36])
Generated by NEH heuristic (see Nawaz et al.[15])

Do
Apply learning strategy
Improvement occurs, Modify weights
$(W_{ij})_k = (W_{ij})_k + RF * ((W_{ij})_k - (W_{ij})_{k-1})$.
No improvement occurs, set $W_{ij} = BW_{ij}$
Modify weights for RND > 0.5,
$(W_{ij})_{k+1} = (Wij)_k + RND * \alpha * Tij$ *or*
for RND ≤ 0.5, $(W_{ij})_{k+1} = (W_{ij})_k - RND * \alpha * T_{ij}$
Saved the best sequence obtained at any generation
Loop until reach generation size
While stopping criteria= false.

The initialize weight is set 1 for all jobs. In the IALA method, three learning parameters, namely, the learning rate (α), the reinforcement factor (*RF*) and the tolerate iterations with no improvement (TINI) are used. These parameters were explained by Agarwal et al[1].
It is well known that the IALAs' efficiency depends to a high degree upon to the selection of the learning parameters. The determination of the suitable settings for the learning parameters of IALA is very difficult task. In general, there are a few control mechanisms for these learning parameters, and the full factorial design of experiments is used in this study for efficiency of IALA to solve the NW-FSSP. The application involved three learning parameters, each having possible different values.
These parameters are given as follows

Learning rate: 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.100

Reinforcement factor: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Tolerate iterations with no improvement: 10, 20, 30, 40, and 50

The number of replicates is 25. In total, 192 problems are classified into 8 groups depending on the number of machines and an instance problem is taken from each group. The parameter optimization is implemented and the best parameter set is determined for the instance. The best parameter set determined for an instance is generalized and used for the other problems of the same group. Therefore, the parameter optimization is implemented for 64 instances. Consequently, 10 learning rate levels, 10 reinforcement factor levels and 5 tolerate iterations with no improvement levels are implemented among the 64 problems with the 25 replicated.
A total number of 10*10*5*25= 12,500 runs are made among the 64 problems. Table 3 shows the parameter setting in each of the replicated runs.
There are 192 instances and *nxm-a*, *nxm-b* and *nxm-c* problems defined as having *n*-jobs (*n=8, 10, 12, 50, 100, 150, 200, 250*), *m*-machine (*m*= 2, 3, 5, 8, 10, 15, 20, 25) and *a,b* and *c* type problems. The processing times of *a* type problems are [1, 10] interval, *b* type problems [1, 50] interval and *c* type problems [1, 100] interval.
Our computational study aims to seek a good heuristic solution for the NW-FSSPs. The results are compared with the Aldowaisan and Allahverdi's (2003) results by using Genetic heuristic (GEN-2). The algorithm is implemented in Borland Delphi and is executed with a PC with a Pentium IV with 3.0 GHz processor and 1 GB memory.
The results obtained with the IALA and GEN-2 and the comparisons of these two heuristic are presented in Table 4. Percentage Deviation (PD) is defined as follows.
For the *IALA*

$$PD=((IALA(C_{max})-Best(C_{max}))/Best(C_{max}))*100 \quad (1)$$

For the *GEN-2*

$$PD=((GEN\text{-}2(C_{max})-Best(C_{max}))/Best(C_{max}))*100 \quad (2)$$

Table 3. Best parameter sets

Machine (m)	***Job (n)***	***RF***	***TINI***	***α***
2	8 10 12	3	30	0.02
	50 100 150	7	50	0.05
	200 250	5	40	0.06
3	8 10 12	2	20	0.07
	50 100 150	5	40	0.04
	200 250	5	40	0.06
5	8 10 12	2	10	0.02
	50 100 150	7	50	0.06
	200 250	8	50	0.07
8	8 10 12	2	50	0.02
	50 100 150	4	30	0.08
	200 250	10	40	0.09
10	8 10 12	4	40	0.03
	50 100 150	4	20	0.01
	200 250	6	40	0.06
15	8 10 12	3	40	0.07
	50 100 150	6	30	0.08
	200 250	9	40	0.06
20	8 10 12	4	30	0.08
	50 100 150	6	20	0.06
	200 250	8	30	0.01
25	8 10 12	6	50	0.02
	50 100 150	5	20	0.07
	200 250	9	20	0.05

Table 4. Heuristics comparison

Problem	PD		Problem	PD		Problem	PD	
	GEN-2	IALA		GEN-2	IALA		GEN-2	IALA
8x2-a	0.000	0.000	8x3-a	0.000	0.000	8x5-a	0.794	0.000
8x2-b	0.000	0.000	8x3-b	0.000	4.828	8x5-b	0.000	0.505
8x2-c	0.000	0.645	8x3-c	0.534	0.000	8x5-c	1.075	0.000
10x2-a	0.000	0.000	10x3-a	0.000	0.840	10x5-a	1.429	0.000
10x2-b	0.000	1.775	10x3-b	3.270	0.000	10x5-b	0.000	3.846
10x2-c	1.541	0.000	10x3-c	0.727	0.000	10x5-c	0.885	0.000
12x2-a	0.000	0.000	12x3-a	0.000	0.000	12x5-a	0.000	1.149
12x2-b	0.000	0.000	12x3-b	1.790	0.000	12x5-b	1.034	0.000
12x2-c	1.387	0.000	12x3-c	0.000	1.952	12x5-c	0.000	1.473
50x2-a	1.054	0.000	50x3-a	0.000	1.124	50x5-a	0.419	0.000
50x2-b	0.000	1.461	50x3-b	1.710	0.000	50x5-b	0.665	0.000
50x2-c	1.158	0.000	50x3-c	0.292	0.000	50x5-c	0.000	3.213
100x2-a	0.719	0.000	100x3-a	0.000	0.722	100x5-a	0.756	0.000
100x2-b	0.000	0.167	100x3-b	0.547	0.000	100x5-b	0.538	0.000
100x2-c	0.000	0.240	100x2-c	0.573	0.000	100x5-c	0.571	0.000
150x2-a	0.475	0.000	150x3-a	0.154	0.000	150x5-a	0.000	0.275
150x2-b	0.000	0.477	150x3-b	0.113	0.000	150x5-b	0.198	0.000
150x2-c	0.102	0.000	150x3-c	0.371	0.000	150x5-c	1.031	0.000
200x2-a	0.482	0.000	200x3-a	0.157	0.000	200x5-a	0.343	0.000
200x2-b	0.000	0.473	200x3-b	0.000	0.368	200x5-b	0.159	0.000
200x2-c	0.483	0.000	200x3-c	1.303	0.000	200x5-c	0.000	0.503
250x2-a	0.000	0.821	250x3-a	0.000	0.124	250x5-a	0.273	0.000
250x2-b	0.531	0.000	250x3-b	0.235	0.000	250x5-b	0.325	0.000
250x2-c	0.035	0.000	250x3-c	0.766	0.000	250x5-c	0.332	0.000
8x8-a	0.000	0.000	8x10-a	0.658	0.000	8x15-a	0.000	0.000
8x8-b	0.000	0.000	8x10-b	0.000	0.000	8x15-b	0.000	0.000
8x8-c	0.000	0.000	8x10-c	0.000	0.602	8x15-c	0.000	0.000
10x8-a	0.000	0.000	10x10-a	0.000	1.099	10x15-a	0.905	0.000
10x8-b	0.000	0.809	10x10-b	0.000	3.823	10x15-b	0.910	0.000
10x8-c	0.000	0.000	10x10-c	5.049	0.000	10x15-c	0.000	1.426
12x8-a	0.995	0.000	12x10-a	0.943	0.000	12x15-a	0.760	0.000
12x8-b	1.108	0.000	12x10-b	0.000	2.522	12x15-b	0.484	0.000
12x8-c	4.808	0.000	12x10-c	2.291	0.000	12x15-c	1.403	0.000
50x8-a	2.261	0.000	50x10-a	0.231	0.000	50x15-a	0.000	1.117
50x8-b	1.364	0.000	50x10-b	0.500	0.000	50x15-b	0.158	0.000
50x8-c	0.089	0.000	50x10-c	0.276	0.000	50x15-c	0.000	0.920
100x8-a	1.094	0.000	100x10-a	0.228	0.000	100x15-a	0.866	0.000
100x8-b	0.000	0.446	100x10-b	0.000	0.339	100x15-b	0.000	0.761
100x8-c	0.575	0.000	100x10-c	0.000	0.080	100x15-c	1.401	0.000
150x8-a	0.493	0.000	150x10-a	0.190	0.000	150x15-a	0.136	0.000
150x8-b	0.023	0.000	150x10-b	0.898	0.000	150x15-b	0.992	0.000
150x8-c	0.000	0.665	150x10-c	0.890	0.000	150x15-c	0.773	0.000
200x8-a	0.152	0.000	200x10-a	0.000	0.086	200x15-a	0.903	0.000
200x8-b	0.271	0.000	200x10-b	1.159	0.000	200x15-b	0.020	0.000
200x8-c	0.813	0.000	200x10-c	0.000	0.036	200x15-c	0.000	0.302
250x8-a	0.000	0.340	250x10-a	0.135	0.000	250x15-a	0.162	0.000
250x8-b	0.000	0.080	250x10-b	0.979	0.000	250x15-b	1.022	0.000
250x8-c	0.069	0.000	250x10-c	0.309	0.000	250x15-c	0.000	0.035

Table 4. (Continued)

Problem	PD		Problem	PD	
	GEN-2	IALA		GEN-2	IALA
8x20-a	0.463	0.000	8x25-a	0.000	0.000
8x20-b	0.000	0.000	8x25-b	0.842	0.000
8x20-c	0.000	0.000	8x25-c	2.487	0.000
10x20-a	0.763	0.000	10x25-a	0.352	0.000
10x20-b	0.833	0.000	10x25-b	0.160	0.000
10x20-c	0.858	0.000	10x25-c	1.632	0.000
12x20-a	0.000	3.136	12x25-a	0.597	0.000
12x20-b	0.000	0.000	12x25-b	0.349	0.000
12x20-c	2.285	0.000	12x25-c	0.000	0.177
50x20-a	0.189	0.000	50x25-a	0.086	0.000
50x20-b	0.568	0.000	50x25-b	0.534	0.000
50x20-c	0.000	0.096	50x25-c	0.729	0.000
100x20-a	0.235	0.000	100x25-a	0.000	0.214
100x20-b	0.523	0.000	100x25-b	0.158	0.000
100x20-c	1.630	0.000	100x25-c	0.560	0.000
150x20-a	0.063	0.000	150x25-a	0.116	0.000
150x20-b	0.000	0.110	150x25-b	0.924	0.000
150x20-c	0.498	0.000	150x25-c	0.000	0.673
200x20-a	0.000	0.379	200x25-a	0.131	0.000
200x20-b	0.365	0.000	200x25-b	1.598	0.000
200x20-c	0.000	0.450	200x25-c	0.055	0.000
250x20-a	0.261	0.000	250x25-a	0.295	0.000
250x20-b	0.442	0.000	250x25-b	0.000	0.809
250x20-c	0.434	0.000	250x25-c	0.616	0.000

As it seen in Table 4, for 2-machines problems the proposed IALA found best-C_{max} values for 16 problems over 24 while GEN-2 found only13 best-C_{max} values. When all problems are considered, the proposed IALA found best-C_{max} values for 142 problems over 192 while GEN-2 found only 70 best-C_{max} values. These results are summarized in Table 5.

Table 5. The performance of IALA and GEN-2

Problems For- machines	Best-C_{max}	
	GEN-2	Proposed IALA
2	13	16
3	9	17
5	7	17
8	10	19
10	9	16
15	9	18
20	8	19
25	5	20
Total	**70**	**142**

Average Percentage Deviation (APD) of proposed IALA is also compared with GEN-2. The APD of proposed IALA and GEN-2 are defined as follows;

$$APD = \frac{\sum_{L=1}^{I} PD(L)}{I} \tag{3}$$

The results are presented in Table 6.

Table 6 The APD computational results of IALA and GEN-2

Problems For machines	APD	
	GEN-2	Proposed IALA
2	0.332	0.252
3	0.523	0.415
5	0.451	0.457
8	0.588	0.097
10	0.614	0.358
15	0.454	0.237
20	0.434	0.174
25	0.509	0.078
Total Average	**0.4881**	**0.2585**

As it seen in Table 6 for all problems average APD for proposed IALA and GEN-2 is 0.2585 and 0.4881 respectively. Proposed IALA has found the smaller average APD than GEN-2.
The CPU times of proposed IALA and GEN-2 are compared in Table 7.

Table 7 The maximum CPU times of the IALA and GEN-2

Methods	Configuration of the computer	Maximum CPU times (sec)
Proposed IALA	PC Pentium IV-3.0 GHz- 1 GB memory	110
GEN-2	PC Pentium IV-3.0 GHz- 1 GB memory	156

The computational results indicated that the proposed IALA method is effective on average in terms of a reduced makespan for the NW-FSSP.

5. Conclusions

In this paper, an adaptive learning approach is suggested for NW-FSSP with the makespan criteria. The considered problem is a NP-Hard problem. Most of the studies to solve that problem are approximate methods rather than an exact method. The test problem are generated according to the literature e.g., Shyu et al.[16], Aldowaisan and Allahverdi[2], Gangadharan and Rajendran[11] and Rajendran[12]. The percentage deviations from Aldowaisan and Allahverdi's[2] results by using Genetic heuristic (GEN-2) are calculated. The better solutions are resulted with the suggested adaptive learning approach. The proposed adaptive learning approach is a good problem solving technique for NW-FSSP with the makespan criteria on average.
For further research, some changes may be done in the learning strategy or the algorithm may be hibritted with another method to improve solution quality.

References

1. A. Agarwal, S. Colak, E. Eryarsoy, Improvement heuristic for the flow-shop scheduling problem: An adaptive-learning approach, *European Journal of Operational Research* 169 (2006) 801–815.
2. T. Aldowaisan, and A. Allahverdi, New heuristics for no-wait flowshops to minimize makespan, *Computers & Operations Research* 30 (8) (2003) 1219–1231.
3. D.E. Akyol, Application of neural networks to heuristic scheduling algorithms, *Computers & Industrial Engineering* 46 (2004) 679- 696.
4. X. Wang, and T.C.E. Cheng, A heuristic approach for two-machine no-wait flowshop scheduling with due dates and class setups, *Computers & Operations Research* 33 (2006) 1326–1344.
5. M.L. Pinedo, *Scheduling, Theory, Algorithm, and Systems*, Third Edition (Springer, New York, 2008).
6. T. Aldowaisan, and A. Allahverdi, New heuristics for *m*-machine no-wait flowshop to minimize total completion time, *Omega* 32 (5) (2004) 345–352.
7. S.S. Reddi, C.V. Ramamoorthy, On the flowshop sequencing problem with no-wait in process, *Operational Research Quarterly*, 23 (3) (1972) 323–331.
8. D.A. Wismer, Solution of the flowshop scheduling problem with no intermediate queues, *Operations Research*, 20 (3) (1972) 689–697.
9. M.C. Bonney, and S.W. Gundry, Solutions to the constrained flowshop sequencing problem, *Operational Research Quarterly*, 27 (4) (1976) 869–883.
10. J.R. King, and A.S. Spachis, Heuristics for flowshop scheduling. *International Journal of Production Research*, 18 (3) (1980) 345–357.
11. R. Gangadharan, C. Rajendran, Heuristic algorithms for scheduling in the no-wait fowshop, *International Journal of Production Economics* 32 (3) (1993) 285- 290.
12. C. Rajendran, A no-wait flowshop scheduling heuristic to minimize makespan, *Journal of the Operational Research Society*, 45 (4) (1994) 472-478.
13. E. Bertolissi, Heuristic algorithm for scheduling in the no-wait flow-shop, *Journal of Materials Processing Technology*, 107 (2000) 459- 465.
14. T. Aldowaisan, A new heuristic and dominance relations for no-wait flowshops with setups, *Computers & Operations Research* 28 (2001) 563- 584.
15. M. Nawaz, E. E. Enscore, and I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, 11 (1983) 91–95.
16. S.J. Shyu, B.M.T. Lin, P.Y. Yin, Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time, *Computers & Industrial Engineering* 47 (2004) 181–193.
17. C. J. Lin, G. Dun-wei, M. Xiao-ping, A heuristic genetic algorithm for the no wait flowshop scheduling problem, *Journal of China University of Mining and Technology,* 17 (4) (2007) 582- 586.
18. A.R. Rahimi-Vahed, B. Javadi, M. Rabbani, R. Tavakkoli-Moghaddama, A multi-objective scatter search for a bi-criteria no-wait flow shop scheduling problem, *Engineering Optimization,* 40 (4) (2008) 331–346.
19. Q. K. Pan, L. Wang, B, H. Zhao, An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion, *Int J Adv Manuf Technol*, 38 (2008) 778–786.
20. Q. K. Pan, M. F. Tasgetiren, Y. C. Liang, A discrete particle swarm optimization algorithm for the no wait flowshop scheduling problem, *Computers and Operations Research,* 35 (9) (2008) 2807- 2839.
21. R. Tavakkoli-Moghaddam , A. R. Rahimi-Vahed, A.H. Mirzae, Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm, *Int J Adv Manuf Technol,* 36 (2008) 969–981.
22. D. Laha, U.K. Chakraborty, A constructive heuristic for minimizing makespan in no-wait flow shop scheduling, *Int J Adv Manuf Technol* 41 (2009) 97–109.
23. B. Qian, L. Wang, D. X., Huang, X. Wang, Multi-objective no-wait flow-shop scheduling with a memetic algorithm based on differential evolution, *Soft Comput*, 13 (2009) 847–869
24. B. Qian, L. Wang, R. Hu, D.X. Huang, X. Wang, A DE-based approach to no-wait flow-shop scheduling, *Computers & Industrial Engineering,* 57 (2009) 787–805.
25. J. J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, *Biological Cybernetics* 52 (3) (1985) 141- 152.

26. Y. Park, S. Kim, Y. H. Lee, Scheduling jobs on parallel machines applying neural network and heuristic rules, *Computers & Industrial Engineering,* 38 (2000) 189-202.
27. I. Lee, M. J. Shaw, A neural-net approach to real time flow-shop sequencing, *Computers & Industrial Engineering,* 38 (2000) 125- 147.
28. D. J. Fonseca, and D. Navaresse, Artificial neural networks for job shop simulation, *Advanced Engineering Informatics* 16 (2002) 241–246.
29. M. Solimanpur, P. Vrat, R. Shankar, A neuro-tabu search heuristic for the flow shop scheduling problem, *Computers & Operations Research* 31 (2004) 2151–2164.
30. A. Agarwal, S. Colak, V. S. Jacob, H. Pirkul, Heuristics and augmented neural networks for task scheduling with non-identical machines, *European Journal of Operational Research,* 175 (2006) 296–317.
31. D. E. Akyol, and G. M. Bayhan, A review on evolution of production scheduling with neural Networks, *Computers & Industrial Engineering* 53 (2007) 95–122.
32. J., Holland, *Adaptation in Natural and Artificial Systems* (*University of Michigan Press, 1975, Ann Arbor*).
33. C. Kahraman, O. Engin, I. Kaya, M.K. Yılmaz, An application of effective genetic algorithms for solving hybrid flow shop scheduling problems, *International Journal of Computational Intelligence Systems*, 1(2) (2008)134- 147.
34. DG. Dannenbring, An evaluation of fowshop sequencing heuristics. *Management Science,* 23 (1977) 174–82.
35. H. G, Campbell, R.A. Dudek, M.L. Smith, A heuristic algorithm for the n-job, m-machine sequencing problem. *Management Science*;16:B (1970) 630–637.
36. D.S. Palmer, Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near-optimum, *Operational Research Quarterly*, 16 (1) (1965) 101–107.