Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

An algorithm for a no-wait flowshop scheduling problem for minimizing total tardiness with a constraint on total completion time

Ali Allahverdi[a*], Harun Aydilek[b] and Asiye Aydilek[c]

[a]*Industrial & Management Systems Eng. Dept., Kuwait University, Kuwait*
[b]*Department of Mathematics & Natural Sciences, Gulf University for Science and Technology, Kuwait*
[c]*Department of Economics & Finance, Gulf University for Science and Technology, Kuwait*

C H R O N I C L E

Article history:
Received February 17 2021
Received in Revised Format
May 23 2021
Accepted July 28 2021
Available online
August, 4 2021

Keywords:
Algorithm
Scheduling
Statistical analysis
No-wait

A B S T R A C T

We consider a no-wait m-machine flowshop scheduling problem which is common in different manufacturing industries such as steel, pharmaceutical, and chemical. The objective is to minimize total tardiness since it minimizes penalty costs and loss of customer goodwill. We also consider the performance measure of total completion time which is significant in environments where reducing holding cost is important. We consider both performance measures with the objective of minimizing total tardiness subject to the constraint that total completion time is bounded. Given that the problem is NP-hard, we propose an algorithm. We conduct extensive computational experiments to compare the performance of the proposed algorithm with those of three well performing benchmark algorithms in the literature. Computational results indicate that the proposed algorithm reduces the error of the best existing benchmark algorithm by 88% under the same CPU times. The results are confirmed by extensive statistical analysis. Specifically, ANOVA analysis is conducted to justify the difference between the performances of the algorithms, and a test of hypothesis is performed to justify that the proposed algorithm is significantly better than the best existing benchmark algorithm with a significance level of 0.01.

1. Introduction

Jobs should be processed uninterruptedly on successive machines in a no-wait m-machine flowshop manufacturing setting. In other words, each job is processed successively from machine 1 to machine m. For example, steel, chemical, and plastic industries require no-wait in process. Hall and Sriskandarajah (1996) and Allahverdi (2016) summarize the research that has been conducted on flowshop scheduling with the no-wait constraint. Hall and Sriskandarajah (1996) reviewed more than 100 papers while Allahverdi (2016) surveyed about 300 papers. Petroleum refineries (Arabameri & Salmasi, 2013), bakery production (Hecker et al., 2014), and surgery scheduling (Wang et al., 2015) are some recent applications of the no-wait flowshop scheduling problem. The objective is minimizing total tardiness (TT) such that total completion time (TCT) is bounded above. Customer satisfaction is vital in today's economics which face harsh international competition. Customer satisfaction is achieved by minimizing total tardiness as loss of customer goodwill is related to unfulfilled due dates, Ding et al. (2015). Therefore, minimizing TT is essential. As reported by Arabameri and Salmasi (2013), no-wait flowshop scheduling applications with the minimization of TT include refineries, as chemical activities necessitate processes to be completed without delays as much as possible. This is due to penalties which result from tardiness in production. Tardiness of productions may have various penalties. Moreover, for some manufacturing environments, jobs are needed as soon as possible, which indicates that minimizing total completion time is important. Minimizing total completion time is predominantly significant when reducing holding cost is of main concern. Many researchers have discussed the importance of minimizing total cost of inventory or holding cost.

* Corresponding author
E-mail: ali.allahverdi@ku.edu.kw (A. Allahverdi)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.8.003

The m-machine no-wait flowshop scheduling problem to minimize TT (*Fm/no-wait/ΣTj)* was addressed by Ding et al. (2015), Aldowaisan and Allahverdi (2012), and Liu et al. (2013). Aldowaisan and Allahverdi (2012) proposed algorithms including simulated annealing (SA) and genetic algorithm (GA). A new version of the algorithm NEH was presented by Liu et al. (2013). Ding et al. (2015) presented an accelerated NEH algorithm and also presented iterated greedy algorithms. They showed that their algorithm outperforms the others. Many researchers addressed the m-machine no-wait flowshop scheduling problem with the objective of minimizing total completion time, e.g., Rajendran and Chaudhuri (1990), Chen et al. (1996), Fink and Voß (2003), Pan et al. (2008a). Moreover, Aldowaisan and Allahverdi (2004) proposed different heuristics and showed that their heuristics perform better than the algorithms of Rajendran and Chaudhuri (1990) and those of Chen et al. (1996). An Ant Colony Optimization heuristic was proposed by Shue et al. (2004) for the problem while a constructive heuristic was presented by Framinan et al. (2010), where they showed that their constructive heuristic performs better than the earlier ones. Nagano et al. (2012) addressed the same problem but considered setup times separately. A single criterion was addressed in the aforementioned research. However, there are some other scheduling environments with more than a single criterion, which is called multi-criteria scheduling. Multi-criteria flowshop scheduling problems are addressed by utilizing different approaches. For example, Jenabi et al. (2010), Ruiz and Allahverdi (2009), Allahverdi and Aldowaisan (2004) used a weighted linear combination of all the criteria as their objective function. On the other hand, Pan et al. (2008b, 2009) and Qian et al. (2009) used The Pareto analysis. Additionally, another approach for multi-criteria scheduling problems is constraint optimization, e.g., Allahverdi and Aydilek (2013), Aydilek and Allahverdi (2012, 2013), Allahverdi et al. (2020). In constraint optimization, using the notation of T'Kindt and Billaut (2002), the problem can be characterized by *ε(PM-1/ PM-2),* where the objective is to minimize *PM-1* such that *PM-2* is bounded by a predetermined value. Allahverdi and Aydilek (2014) addressed the *Fm/no-wait, STsi/∈(ΣCj/Cmax)* problem. They presented several algorithms such SA, GA and a Differential Evolution (DE) and showed that SA outperformed the others. Allahverdi et al. (2018) presented different algorithms to the *Fm/no-wait/ε(ΣTj/Cmax)* problem and presented an algorithm showing that their proposed algorithm significantly outperforms the other algorithms. Allahverdi et al. (2020) addressed the Fm*/no-wait/∈ (∑T_j/C_{maj})* problem and presented an algorithm. In the current paper, we consider the no-wait flowshop scheduling problem on an *m*-machine to minimize TT with an upper bound on TCT, i.e., the Fm*/no-wait/∈ (∑T_j/∑C_j)* problem. We propose a new algorithm for the problem and compare it with the three benchmark algorithms in the literature.

The problem is defined in the next section while the proposed algorithm is presented in Section 3. Sections 4, 5, and 6 present the evaluation of the proposed algorithm, statistical analysis, and concluding remarks, respectively.

2. Problem description

We investigate the problem of *Fm/no-wait/ε(∑T_j /∑C_j),* which can be represented as follows:

min $\sum T_j$

subject to

$\sum C_j \leq UTCT$

where *UTCT* is an upper bound on the total completion time which is practically specified by the scheduler. The investigated problem is NP-hard since it is known that a special case of our problem, the two-machine no-wait flowshop problem of minimizing L_{max}, is *NP*-hard. Hence, we propose different algorithms in the following section. Let $t_{j,k}$ denote the processing time of job j (j= 1, …, n) on machine k (k=1, …, m), and d_j represent the due date of job j. Moreover, let C_j and T_j denote the completion time and tardiness of job j, respectively, where $T_j = \max\{0, C_j - d_j\}$. Then, the total tardiness and total completion time can be computed as:

$$TT = \sum_{j=1}^{n} T_j \quad \text{and} \quad TCT = \sum_{j=1}^{n} C_j.$$

3. The proposed method

As stated earlier, the m-machine no-wait scheduling problem of minimizing total tardiness subject to a constraint on the total completion time is NP-hard. In general, two methodologies are commonly utilized in the scheduling literature for NP-hard problems. One of the methodologies is to use implicit enumeration techniques, such as dynamic programming, while another methodology is to use approximate algorithms. In this research, the second methodology will be taken as problems of any reasonable size can be solved in a short time if algorithms are utilized while problems of a limited size can be solved if implicit enumeration techniques are utilized. In this paper, we propose an algorithm for the *Fm/no-wait/ε(∑T_j /∑C_j)* problem. Specifically, we propose an algorithm which is merged with an iterated insertion and a constructive algorithms. No algorithm exists for our problem in the literature since the investigated problem has not been addressed earlier. In order to evaluate our proposed algorithm's performance, we also adapted a few algorithms, which have been presented for closely related scheduling problems and shown to be performing very well, to our problem as benchmarks. Our proposed algorithm along with the adapted algorithms are described in the next subsection.

3.1. Adapted Benchmark Algorithms

The unconstrained problem of *Fm/no-wait/*$\sum T_j$ is a special case of our constrained problem of *Fm/no-wait/*$\varepsilon(\sum T_j/\sum C_j)$. In other words, our problem *Fm/no-wait/*$\varepsilon(\sum T_j/\sum C_j)$ reduces to the problem *Fm/no-wait/TT* when the upper bound *UTCT* is set to a large value. Even though our problem *Fm/no-wait/*$\varepsilon(\sum T_j/\sum C_j)$ has not been addressed earlier, the problem *Fm/no-wait/*$\sum T_j$ has been investigated and several algorithms were proposed. Therefore, we consider a few existing well performing algorithms designed for the unconstrained problem *Fm/no-wait/*$\sum T_j$ to compare our proposed algorithm. It is known that the algorithm *MNEH* performs the best among six algorithms that Liu et al. (2013) considered for the unconstrained problem of *Fm/no-wait/*$\sum T_j$. Therefore, we adapted their algorithm *MNEH* to our problem and we represented it by Adp-L. Moreover, Ding et al. (2015) also provided three algorithms for the unconstrained problem and showed that one of their algorithms, AIG*1*, outperformed the other two. Hence, we adapted their algorithm *AIG1* to our problem of *Fm/no-wait/*$\varepsilon(\sum T_j/\sum C_j)$. We denote the adapted algorithm with Adp-D. Allahverdi et al. (2018) investigated the *Fm/no-wait /*$\varepsilon(\sum T_j/C_{\max})$ problem, which is also related to our problem since they consider minimizing TT with a constraint on the makespan. IIn this paper we consider the same objective but with a constraint on TCT. We also adapt their algorithm to our problem, which is denoted by Adp-A.

3.2. The Proposed Constructive Algorithm (PCA)

The proposed constructive algorithm consists of three stages: constructing an initial sequence based on the performance measure of TT, applying an insertion to the initial sequence to satisfy the constrain on TCT, and utilizing an iterated search to improve TT such that the constraint is not violated. The PCA algorithm requires a parameter I which denotes the number of iterations, used in Step 3 of the algorithm, which is obtained based on a fair computational time of CPU with the benchmark algorithms.

Steps of PCA

Step 1: Constructing the initial sequence

Select the job for the first position of the initial sequence as the job with the smallest due date. For the remaining position of the sequence, choose the job that minimizes TT. Next, insert the last selected job in all the previous positions, and select the sequence that yields the minimum TT. Continue this step for all the n jobs. The obtained sequence at the end is denoted by π_1.

Step 2: Checking the constraint on TCT

Check if the sequence π_1 satisfies the constraint on TCT. If so, then go to Step 3. Otherwise, conduct a pairwise exchange of the jobs in the sequence π_1 until the constraint on the TCT is satisfied. The resulting sequence is denoted by π_2.

Step 3: Improving TT

Conduct an iterated search on the sequence π_2 to improve TT such that the TCT constraint is not violated. The search is continued until the parameter I is reached. The final sequence is denoted by π_3.

4. Evaluation of the PCA

The performance of the proposed algorithm PCA is evaluated by comparing its performance with those of the adapted three benchmark algorithms stated earlier, which are Adp-L, Adp-D, and Adp-A. The earlier stated upper bound *UTCT*, which is a constraint on TCT, is required to perform computational experiments. In practice, the value of the parameter *UTCT* is determined by the manager of the manufacturing unit. For computational experiments, we set the *UTCT* value to the average value of *TCT* of randomly selected ten sequences. Obtaining such a value for *UTCT* results in neither a large nor a small *TCT*. This is essential since a large value of *UTCT* results in a problem without a constraint. On the other hand, a small value of *UTCT* results in an infeasible solution. A uniform distribution with a large variance is recommended for computational purposes. Therefore, we generate processing times of jobs from a uniform distribution with range of 1 to 100, which has a large variance. Generating data sets using this approach is common in the literature, e.g., Ruiz-Torres et al. (2017) and Liu et al. (2013). Moreover, due dates of jobs are needed to be generated as well. The uniform distribution with the range of *LB(1-T-R/2)* and *LB(1-T+R/2)* is usually used in the literature to generate job due dates. The parameter LB is a lower bound on makespan. The following LB is used.

$$LB = \max_{1\le h\le m}\left\{\sum_{i=1}^{n} t_{i,h} + \sum_{\mathrm{r=h+1}}^{\mathrm{m}} \min_{1\le i\le n}\{t_{i,r}\}\right\}$$

where, $\sum_{\mathrm{r=m+1}}^{\mathrm{m}} \min_{1\le i\le n}\{t_{i,r}\} = 0$.

The parameters T and R denote the tardiness factor and due date range, respectively. Both parameters are chosen between 0 and 1, and usually the values 0.25, 0.50, and 0.75 are used in the literature. Generating due dates utilizing this approach is common in the literature, e.g., Lee and Kim (2017), and Shao et al. (2017). The parameters of *m, n, T,* and *R*, which are used

for computational experiments are given in Table 1. There are a total of 252 (7×4×3×3) combinations of the parameters of *n, m, R*, and *T*. For each combination, we conduct 40 replications, which results in 10080 problems.

Table 1
Parameter values

Parameter	Description	Parameter values
n	Number of jobs	20, 30, 40, 50, 60, 70, 80
m	Number of machines	2, 5, 10, 12
R	Range factor	0.25, 0.50, 0.75
T	Tardiness factor	0.25, 0.50, 0.75

The error of an algorithm is defined as $RE = 100 \times \frac{TT - TT_{best}}{TT_{best}}$ where TT is the total tardiness of the algorithm and TT_{best} is the smallest total tardiness among the considered algorithms. It should be noted that we use the average of the 40 replicates and present the results as a percentage. The computational values of RE for the algorithms Adp-A, PCA, Adp-D, and Adp-L are given in Tables 2-5 for m = 2, 5, 10, and 12, respectively.

Table 2
Computational results of the algorithms for *m*=2

						T				
			0.25			0.5			0.75	
			R			***R***			***R***	
n	*Alg.*	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	Adp-A	1.10	1.16	1.02	1.30	0.97	1.19	1.29	1.08	1.42
	PCA	0.11	0.15	0.20	0.07	0.35	0.30	0.13	0.19	0.10
20	*Adp-D*	5.18	4.51	3.79	5.15	3.93	4.59	3.85	4.53	4.15
	Adp-L	10.18	7.32	6.84	6.26	5.81	6.81	5.93	6.39	6.17
	Adp-A	1.57	1.42	1.18	1.34	1.31	1.11	1.28	1.56	1.40
	PCA	0.19	0.23	0.19	0.20	0.14	0.19	0.18	0.10	0.09
30	*Adp-D*	4.09	4.76	4.27	4.06	5.09	3.75	4.27	4.37	4.78
	Adp-L	7.06	7.29	6.14	4.69	7.66	6.11	6.81	6.51	5.79
	Adp-A	1.57	1.97	1.44	0.97	0.76	1.18	1.39	1.54	1.56
	PCA	0.18	0.12	0.11	0.25	0.28	0.21	0.18	0.13	0.14
40	*Adp-D*	4.05	4.27	4.16	4.17	4.26	4.05	4.26	3.67	3.87
	Adp-L	4.63	6.11	6.72	5.58	5.51	6.24	6.29	5.47	4.78
	Adp-A	1.42	0.92	1.24	1.45	1.63	1.29	1.37	1.25	1.45
	PCA	0.26	0.26	0.17	0.16	0.13	0.14	0.18	0.17	0.16
50	*Adp-D*	4.23	4.33	4.09	3.79	4.00	4.24	3.66	3.55	3.94
	Adp-L	7.40	5.23	7.28	5.04	5.21	6.32	5.08	5.45	7.25
	Adp-A	1.63	1.51	1.52	1.85	1.62	1.44	1.18	1.25	1.46
	PCA	0.13	0.18	0.10	0.09	0.19	0.02	0.19	0.16	0.23
60	*Adp-D*	3.63	3.98	3.65	3.84	3.76	3.66	3.46	3.83	3.83
	Adp-L	5.61	4.47	6.17	5.34	6.30	4.34	6.31	5.97	5.72
	Adp-A	1.31	2.35	1.52	1.84	1.90	1.68	1.80	1.64	1.10
	PCA	0.30	0.07	0.12	0.08	0.05	0.09	0.11	0.18	0.38
70	*Adp-D*	3.57	3.61	3.56	3.41	3.42	3.65	3.49	3.76	3.33
	Adp-L	4.02	5.83	6.86	5.25	3.52	3.52	5.33	6.68	4.53
	Adp-A	1.84	1.73	1.35	1.83	1.56	2.08	1.46	1.22	2.29
	PCA	0.17	0.15	0.29	0.15	0.26	0.08	0.06	0.15	0.06
80	*Adp-D*	3.66	3.61	3.08	3.43	3.47	3.30	3.38	3.13	3.36
	Adp-L	4.09	4.29	3.24	4.78	6.67	5.32	4.30	4.83	5.32

The computational times and CPU times of all the algorithms were maintained to be the same for a fair comparison. In the tables, the first column indicates the number of jobs, the second column shows the algorithm, the next three columns present RE values for R = 0.25, 0.5, and 0,75, respectively, for T = 0.25. Similarly, the next three columns are RE values of T = 0.5, and the final three columns represent the RE values for T = 0.75. It should be noted that each value of RE represents the average value of 40 replications. The values of RE in the tables are summarized in Figs. 1-5. Fig. 1 shows the Error versus the number of jobs. The values on the graphs show the average values over the variables R, T, and m with 40 replications. Among the benchmark algorithms, Adp-A performs better than the benchmark algorithms Adp-D and Adp-L. Moreover, the proposed algorithm PCA performs much better than the benchmark algorithm Adp-A. Fig. 2 shows the same results by excluding the two benchmark algorithms Adp-D and Adp-L. It is clear from Fig. 2 that the performance of the benchmark algorithm Adp-A deteriorates, while that of the proposed algorithm PCA improves slightly as the number of jobs increases. Fig. 3 indicates the results with respect to the number of machines for the average errors over R, T, n, and 40 replicates. As in Figs. 1-2, the benchmark algorithm Adp-A is the best among the benchmark algorithms, and the proposed algorithm PCA performs better than the algorithm Adp-A. Fig. 4 illustrates the errors with respect to R values by taking the averages over the parameters T, n, m, and 40 replications. Similarly, Fig. 5 gives the errors for T values by taking the averages over the parameters R, n, m, and 40 replications. The figures yield the same results of the benchmark algorithm Adp-A being the best

benchmark algorithm, and of the algorithm PCA performing much better than the benchmark algorithm Adp-A. The average error of all the algorithms do not seem to be dependent to either R or T values.

Table 3
Computational results of the algorithms for m=5

						T				
			0.25			0.5			0.75	
			R			***R***			***R***	
n	*Alg.*	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	Adp-A	1.32	1.34	1.56	1.30	1.04	1.58	1.68	1.01	0.98
	PCA	0.26	0.22	0.18	0.24	0.31	0.21	0.22	0.16	0.28
20	*Adp-D*	3.25	4.08	3.84	3.59	3.67	3.89	3.47	3.62	3.27
	Adp-L	3.97	5.12	6.34	5.16	6.67	6.52	5.73	4.84	5.30
	Adp-A	1.55	1.95	1.24	1.76	0.92	1.30	1.17	1.11	1.13
	PCA	0.22	0.12	0.19	0.13	0.34	0.25	0.07	0.18	0.19
30	*Adp-D*	4.07	3.82	3.59	3.55	3.76	3.38	3.65	3.60	3.41
	Adp-L	7.43	5.42	4.97	6.72	6.26	4.72	6.59	4.19	4.22
	Adp-A	1.15	1.38	1.91	1.22	1.51	1.35	1.46	0.79	1.14
	PCA	0.13	0.29	0.20	0.19	0.32	0.23	0.14	0.25	0.33
40	*Adp-D*	3.67	3.61	3.67	3.85	3.45	3.61	3.26	3.36	3.49
	Adp-L	5.01	4.78	4.99	4.60	5.16	5.45	5.56	4.54	4.77
	Adp-A	1.78	1.53	1.92	1.34	1.73	1.40	1.90	1.67	1.77
	PCA	0.17	0.18	0.19	0.13	0.10	0.16	0.15	0.17	0.08
50	*Adp-D*	3.27	3.26	3.19	3.28	3.22	3.39	3.17	3.27	3.22
	Adp-L	3.86	5.43	5.19	4.20	4.45	6.34	3.50	5.32	4.05
	Adp-A	2.26	2.27	1.40	1.23	2.07	1.21	1.38	1.93	1.89
	PCA	0.04	0.09	0.30	0.31	0.17	0.19	0.21	0.11	0.14
60	*Adp-D*	3.04	2.95	3.15	3.26	3.33	2.94	3.04	3.05	3.11
	Adp-L	4.80	5.39	5.81	4.52	4.33	4.12	5.55	4.47	4.49
	Adp-A	2.03	1.23	1.92	1.78	2.15	1.37	1.97	1.77	1.53
	PCA	0.11	0.18	0.12	0.12	0.09	0.17	0.14	0.25	0.18
70	*Adp-D*	2.85	3.06	3.05	2.81	2.97	3.05	2.69	2.74	3.10
	Adp-L	4.16	4.51	4.74	4.14	4.13	4.24	3.31	3.41	4.81
	Adp-A	1.79	1.75	2.00	2.09	1.64	1.75	1.38	1.68	1.86
	PCA	0.16	0.08	0.16	0.10	0.26	0.13	0.27	0.09	0.12
80	*Adp-D*	2.77	2.76	2.97	2.97	2.61	2.68	2.53	2.92	2.78
	Adp-L	3.53	3.94	5.30	5.82	3.26	4.56	3.55	3.74	5.03

Table 4
Computational results of the algorithms for m=10

						T				
			0.25			0.5			0.75	
			R			***R***			***R***	
n	*Alg.*	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	Adp-A	1.06	0.81	1.31	1.38	1.36	0.90	1.14	1.16	1.14
	PCA	0.32	0.34	0.32	0.19	0.24	0.30	0.26	0.15	0.18
20	*Adp-D*	2.85	3.49	3.32	3.26	3.10	3.16	2.92	2.79	3.28
	Adp-L	4.58	6.27	5.07	4.81	5.51	3.80	4.62	3.82	4.24
	Adp-A	1.07	0.99	1.82	0.99	1.30	1.67	1.76	1.73	1.70
	PCA	0.18	0.28	0.20	0.20	0.27	0.21	0.20	0.22	0.18
30	*Adp-D*	3.18	3.29	3.31	3.09	3.02	3.27	3.21	3.21	3.22
	Adp-L	4.77	5.03	4.84	4.69	3.38	3.97	5.56	5.22	4.37
	Adp-A	1.77	1.79	1.80	1.40	1.49	1.75	1.43	1.42	1.44
	PCA	0.25	0.20	0.11	0.23	0.19	0.23	0.22	0.21	0.24
40	*Adp-D*	3.22	2.91	2.89	3.18	2.95	3.22	3.13	2.92	2.97
	Adp-L	4.68	4.76	4.82	5.53	4.47	4.27	5.67	4.09	5.48
	Adp-A	1.55	1.73	2.59	1.62	1.45	1.15	1.70	0.96	1.52
	PCA	0.23	0.19	0.11	0.14	0.13	0.13	0.23	0.28	0.14
50	*Adp-D*	2.91	3.01	2.79	2.82	2.71	2.83	2.94	2.62	2.80
	Adp-L	4.95	4.98	4.28	4.76	4.44	3.40	4.05	3.61	3.33
	Adp-A	1.08	1.33	1.49	2.18	2.11	1.82	1.91	1.59	1.85
	PCA	0.34	0.32	0.30	0.23	0.13	0.27	0.16	0.15	0.24
60	*Adp-D*	2.58	2.78	2.95	2.62	2.64	2.71	2.57	2.63	2.47
	Adp-L	4.39	4.94	4.93	2.49	4.76	3.09	3.91	2.53	3.01
	Adp-A	2.15	1.72	1.94	2.50	1.90	1.92	1.42	1.01	2.05
	PCA	0.18	0.11	0.17	0.02	0.18	0.15	0.22	0.20	0.18
70	*Adp-D*	2.44	2.56	2.48	2.66	2.52	2.36	2.37	2.42	2.52
	Adp-L	3.77	3.27	3.73	3.38	3.45	3.43	4.37	4.44	3.66
	Adp-A	2.35	1.70	1.88	2.04	2.00	1.97	2.39	2.23	2.08
	PCA	0.13	0.08	0.08	0.13	0.15	0.26	0.06	0.08	0.07
80	*Adp-D*	2.11	2.31	2.28	2.26	2.22	2.40	2.22	2.19	2.23
	Adp-L	2.88	3.61	3.31	4.22	2.94	3.42	3.10	3.63	3.45

Table 5
Computational results of the algorithms for m=12

						T				
			0.25			0.5			0.75	
			R			***R***			***R***	
n	*Alg.*	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	Adp-A	0.72	1.44	1.40	0.56	1.59	1.17	1.34	0.99	0.89
	PCA	0.27	0.14	0.31	0.43	0.05	0.34	0.21	0.15	0.26
20	*Adp-D*	3.08	2.86	2.97	2.92	2.79	2.88	2.75	3.07	2.77
	Adp-L	4.72	3.02	4.79	4.89	3.74	3.89	4.48	3.74	4.51
	Adp-A	2.13	1.61	1.82	0.77	2.01	1.22	0.99	0.71	1.46
	PCA	0.10	0.26	0.05	0.32	0.23	0.33	0.42	0.34	0.22
30	*Adp-D*	2.94	3.08	2.86	3.09	2.88	3.04	3.17	3.20	2.88
	Adp-L	4.56	4.12	4.42	5.43	4.41	3.92	4.21	5.52	3.46
	Adp-A	1.46	1.39	1.66	1.52	1.80	0.80	0.98	1.53	1.48
	PCA	0.31	0.37	0.18	0.26	0.24	0.36	0.18	0.28	0.25
40	*Adp-D*	3.14	3.12	2.90	2.93	3.13	2.84	2.91	3.03	2.76
	Adp-L	3.56	4.77	4.31	3.99	5.44	3.63	5.39	5.01	3.57
	Adp-A	1.55	1.96	1.72	1.91	1.48	1.50	1.26	1.79	2.06
	PCA	0.20	0.07	0.06	0.24	0.17	0.14	0.12	0.12	0.17
50	*Adp-D*	2.84	2.68	2.64	2.61	2.82	2.45	2.69	2.71	2.51
	Adp-L	3.60	4.23	4.38	3.74	4.21	3.28	4.42	4.05	4.16
	Adp-A	1.69	1.70	1.66	2.33	1.30	2.08	2.49	1.59	1.29
	PCA	0.17	0.24	0.10	0.08	0.23	0.18	0.09	0.14	0.11
60	*Adp-D*	2.63	2.28	2.44	2.41	2.53	2.45	2.48	2.48	2.44
	Adp-L	3.69	3.90	3.93	3.93	4.38	3.03	4.04	4.41	4.00
	Adp-A	1.99	1.65	1.59	2.25	1.89	1.57	1.83	1.72	2.24
	PCA	0.10	0.15	0.14	0.10	0.19	0.13	0.18	0.21	0.12
70	*Adp-D*	2.27	2.14	2.45	2.30	2.52	2.53	2.31	2.34	2.38
	Adp-L	2.83	3.79	3.50	4.51	4.24	3.09	3.04	3.46	3.73
	Adp-A	1.37	1.77	1.81	1.94	2.13	2.13	1.90	1.82	1.84
	PCA	0.22	0.12	0.18	0.17	0.15	0.15	0.08	0.09	0.07
80	*Adp-D*	2.38	2.11	2.28	2.33	2.21	2.07	2.12	2.13	2.13
	Adp-L	2.96	3.36	3.48	3.46	2.77	2.10	3.77	3.28	3.11

The overall average errors of the algorithms Adp-A, PCA, Adp-D, and Adp-L are 1.56, 0.18, 3.16, and 4.70, respectively. The error of Adp-A is 50% and 67% smaller than the errors of Adp-D, and Adp-L, respectively. Moreover, the error of the proposed algorithm PCA is 88% smaller than the error of the best benchmark algorithm Adp-A.

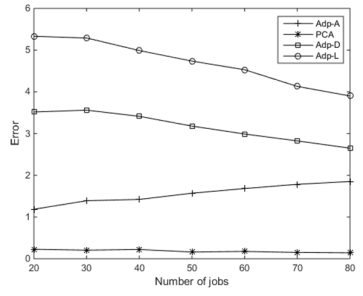

Fig. 1. Error versus number of jobs, n

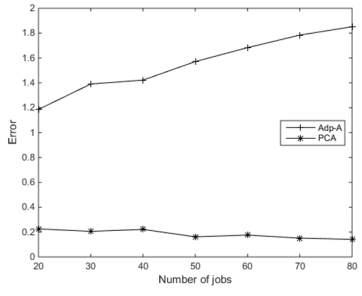

Fig. 2. Error versus number of jobs, n, (best two algorithms)

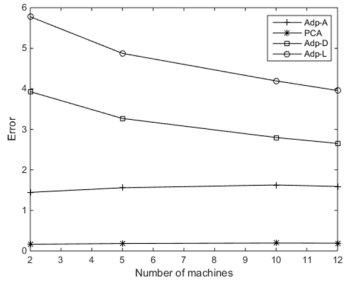

Fig. 3. Error versus number of machines, m

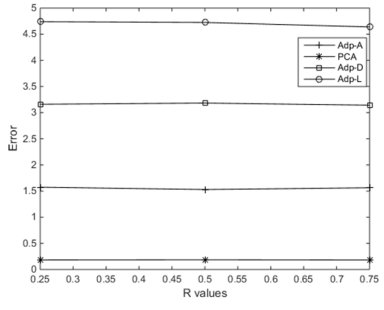

Fig. 4. Error versus range factor, R

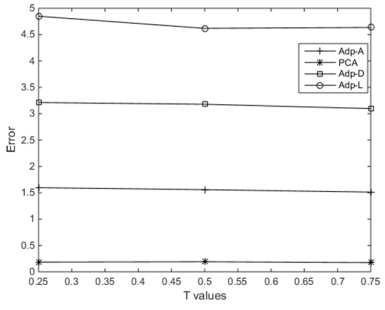

Fig. 5. Error versus tardiness factor, T

5. Statistical Analysis

An ANOVA analysis is conducted to statistically compare the performances of the proposed algorithm PCA and the three benchmark algorithms Adp-A, Adp-D, and Adp-L. Specifically the following null and alternative hypotheses are performed at a significance level of 0.01.

H_0: $\mu(PCA) = \mu(Adp\text{-}A) = \mu(Adp\text{-}L) = \mu(Adp\text{-}D)$

H_1: At least one is different

where $\mu(.)$ shows the average error of the stated algorithm. The null hypothesis is rejected (at a significance level of 0.01) indicating that there is a significant difference between the performances of the algorithm. In order to justify the given results in the previous section, we further statistically compare the performance of the proposed algorithm PCA and the best benchmark algorithm Adp-A by using a test of hypothesis. A two-sample *t*-test is performed to statistically verify that the proposed algorithm PCA is better than the best existing benchmark algorithm Adp-A. The following hypotheses testing were performed.

H_0: $\mu(PCA) = \mu(Adp\text{-}A)$

H_1: $\mu(PCA) < \mu(Adp\text{-}A)$

The null hypothesis (H_0) is rejected for the significance level of 0.01. Therefore, the proposed algorithm PCA statistically outperforms the best benchmark algorithm Adp-A.

6. Concluding Remarks

We addressed the problem of an m-machine no-wait flowshop scheduling with the objective of minimizing total tardiness such that total completion time is bounded above. We proposed an algorithm (PCA) for the problem and compared its performance with three well performing benchmark algorithms (Adp-A, Adp-D, Adp-L) from the literature. We conducted extensive computational experiments. On the average, the error of Adp-A is 50% smaller than that of Adp-D. Moreover, on average, the error of Adp-A is 67% smaller than the error of Adp-L. In other words, Adp-A performs the best among the benchmark algorithms. The computational experiments also reveal that the proposed algorithm PCA reduces the error of the best benchmark algorithm Adp-A by 88% on average under the same CPU times. Statistical analysis (ANOVA and test of hypothesis) reveals that the proposed algorithm PCA outperforms the best benchmark algorithm at a significance level of 0.01. Setup times are assumed to be zero in the current paper. Even though this assumption is valid for some manufacturing environments, it may not be valid for some other manufacturing environments, e.g., Allahverdi (2015). Therefore, an extension to the problem addressed in this paper is to consider non-zero setup times.

Acknowledgements

This research was supported and funded by Kuwait University Research Grant No. EI02/18.

References

Aldowaisan, T., & Allahverdi, A. (2004). New heuristics for m-machine no-wait flowshop to minimize total completion time. *Omega*, *32*(5), 345-352.

Allahverdi, A. (2012). Minimizing total tardiness in no-wait flowshops. *Foundations of Computing and Decision Sciences*, *37*(3), 149-162.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, *246*(2), 345-378.

Allahverdi, A. (2016). A survey of scheduling problems with no-wait in process. *European Journal of Operational Research*, *255*(3), 665-686.

Allahverdi, A., & Aldowaisan, T. (2004). No-wait flowshops with bicriteria of makespan and maximum lateness. *European Journal of Operational Research*, *152*(1), 132-147.

Allahverdi, A., & Aydilek, H. (2013). Algorithms for no-wait flowshops with total completion time subject to makespan. *The International Journal of Advanced Manufacturing Technology*, *68*(9-12), 2237-2251.

Allahverdi, A., & Aydilek, H. (2014). Total completion time with makespan constraint in no-wait flowshops with setup times. *European Journal of Operational Research*, *238*(3), 724-734.

Allahverdi, A., Aydilek, H., & Aydilek, A. (2018). No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. *European Journal of Operational Research*, *269*(2), 590-601.

Allahverdi, A., Aydilek, H., & Aydilek, A. (2020). No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan. *Applied Mathematics and Computation*, *365*, 124688.

Arabameri, S., & Salmasi, N. (2013). Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. *Computers & Industrial Engineering*, *64*(4), 902-916.

Aydilek, H., & Allahverdi, A. (2012). Heuristics for no-wait flowshops with makespan subject to mean completion time. *Applied Mathematics and Computation*, *219*(1), 351-359.

Aydilek, H., & Allahverdi, A. (2013). New heuristics for no-wait flowshops with performance measures of makespan and mean completion time. *International Journal of Operations Research*, *10*, 29-37.

Chen, C. L., Neppalli, R. V., & Aljaber, N. (1996). Genetic algorithms applied to the continuous flow shop problem. *Computers & Industrial Engineering*, *30*(4), 919-929.

Ding, J., Song, S., Zhang, R., Gupta, J. N., & Wu, C. (2015). Accelerated methods for total tardiness minimisation in no-wait flowshops. *International Journal of Production Research*, *53*(4), 1002-1018.

Fink, A., & Voß, S. (2003). Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research*, *151*(2), 400-414.

Framinan, J. M., Nagano, M. S., & Moccellin, J. V. (2010). An efficient heuristic for total flowtime minimisation in no-wait flowshops. *The International Journal of Advanced Manufacturing Technology*, *46*(9-12), 1049-1057.

Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations research*, *44*(3), 510-525.

Hecker, F. T., Stanke, M., Becker, T., & Hitzmann, B. (2014). Application of a modified GA, ACO and a random search procedure to solve the production scheduling of a case study bakery. *Expert systems with applications*, *41*(13), 5882-5891.

Jenabi, M., Naderi, B., & Ghomi, S. F. (2010, September). A bi-objective case of no-wait flowshops. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)* (pp. 1048-1056). IEEE.

Lee, J. Y., & Kim, Y. D. (2017). Minimizing total tardiness in a two-machine flowshop scheduling problem with availability constraint on the first machine. *Computers & Industrial Engineering*, *114*, 22-30.

Liu, G., Song, S., & Wu, C. (2013). Some heuristics for no-wait flowshops with total tardiness criterion. *Computers & operations research*, *40*(2), 521-525.

Nagano, M. S., Da Silva, A. A., & Lorena, L. A. N. (2012). A new evolutionary clustering search for a no-wait flow shop problem with set-up times. *Engineering Applications of Artificial Intelligence*, *25*(6), 1114-1120.

Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, *35*(9), 2807-2839.

Pan, Q. K., & Wang, L. (2008). A novel multi-objective particle swarm optimization algorithm for no-wait flow shop scheduling problems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *222*(4), 519-539.

Pan, Q. K., Wang, L., & Qian, B. (2009). A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. *Computers & Operations Research*, *36*(8), 2498-2511.

Qian, B., Wang, L., Huang, D. X., & Wang, X. (2009). Multi-objective no-wait flow-shop scheduling with a memetic algorithm based on differential evolution. *Soft Computing*, *13*(8), 847-869.

Rajendran, C., & Chaudhuri, D. (1990). Heuristic algorithms for continuous flow-shop problem. *Naval Research Logistics (NRL)*, *37*(5), 695-705.

Ruiz, R., & Allahverdi, A. (2009). New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness. *International Journal of Production Research*, *47*(20), 5717-5738.

Ruiz-Torres, A. J., Paletta, G., & M'Hallah, R. (2017). Makespan minimisation with sequence-dependent machine deterioration and maintenance events. *International Journal of Production Research*, *55*(2), 462-479.

Shao, Z., Pi, D., & Shao, W. (2017). Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. *Computers & Industrial Engineering*, *111*, 331-351.

Shyu, S. J., Lin, B. M., & Yin, P. Y. (2004). Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. *Computers & industrial engineering*, *47*(2-3), 181-193.

T'kindt, V., & Billaut, J. C. (2006). *Multicriteria scheduling: theory, models and algorithms*. Springer Science & Business Media.

Wang, Y., Tang, J., Pan, Z., & Yan, C. (2015). Particle swarm optimization-based planning and scheduling for a laminar-flow operating room with downstream resources. *Soft Computing*, *19*(10), 2913-2926.