Effective hybrid algorithms for no-wait flowshop scheduling problem

Habib Elyasi[1]

[1]*Department of Management, Islamic Azad University, Marand Branch, Marand, Iran*

***

Abstract - *No-wait flowshop scheduling problems with minimizing the total makespan is one of the significant problems in production scheduling. In no-wait flowshop problems, when the process of any job is initiated on any of the machines, that job must pass through all the processes and the machines without waiting until it leaves the last machine. In other words, the process of a job on the first machine would be put off until it is made sure it would not face any delay throughout the process. A hybrid genetic algorithm is proposed for solving such a problem that not only will avoid local optimization, but it is also proven that this algorithm is outstanding while facing NP-hard questions. The results of the numerical example prove that the hybrid genetic algorithm outperforms other algorithms in the literature as well as the genetic algorithm.*

Key Words: **Flowshop, No-wait, Genetic Algorithm, Hybrid Genetic Algorithm, Scheduling.**

1. INTRODUCTION

Distributed multisite production has been placed a premium on because of the competitive nature of globalization. Because of their nature, optimization problems have always been considered in the literature (see for example: [1], [2], [3], [4]). Numerous variants of the flowshop scheduling problem (FSP) have been studied [5] due to the nature of distinct industrial processes. One of the variants of FSPs is called no-wait FSPs (NWFSPs) that are applied in numerous industries such as plastics, metals and electronics. A no-wait flowshop problem occurs when the operations of a job have to be processed continuously from start to end without interruptions either on or between machines. Hence, if needed, the start of a job on a given machine is delayed in order that the operation's completion coincides with the start of the next operation on the subsequent machine. Allahverdi [6] reviewed more than 300 papers in scheduling problem with no-wait constraints and classifies the problems based on shop environment, performance measures, and other factors. Two commonly used performance measures in the scheduling literature are makespan and mean completion time. Minimizing makespan is important in situations where a simultaneously received batch of jobs is required to be completed as soon as possible.

Callahan [7] was one of the first people who studied no-wait flow shop problems. His study was focused on steel industry and he studied no-wait problems with respect to dependent parameters such as temperature. No-wait flow shop problem with minimizing the total makespan is proved to be an NP-hard problem by Röck [8]. Therefore, instead of looking for exact methods, many heuristics have been developed in order to generate solutions with high calculation qualities (see for example: [9], [10], [11], [12], [13]). Aldowaisan and Allahverdi [14] also suggested some heuristics and proved that their proposed algorithms outperform the previous ones. Grabowski and Pempera [15] proposed several heuristics and compared the results with the local search algorithm proposed by Schuster and Framinan [16] and the algorithms proposed by Rajendran [13]. A new heuristic which is based on analogy between the machines was proposed by Framinan and Nagano [17]. Lin and Ying [18] proposed a mixed integer programming mathematical model and an iterated cocktail greedy algorithm as the solution method and they were proved to be more effective than the known algorithms in the literature. Nagano et al. [19] introduced a constructive heuristic named QUARTS and they achieved approximation solutions in a short CPU time.

Metaheuristic algorithms have also been proposed for solving no-wait flowshop problem with the minimization of the total makespan as the objective [see for example; [20], [21]). Tseng and Lin [22] proposed a hybrid genetic algorithm utilizing the base genetic algorithm and a heuristic proposed by them. Zhu et al. [23] studied the problem by implementation of a local search algorithm. Aldowaisan and Allahverdi [14] studied six different meta-heuristics consisting of three (simulated annealing (SA) algorithms and three genetic algorithms (GA) to solve the problem. The results revealed that the two of the algorithms outperformed the algorithms proposed by Gangadharan and Rajendran [12] and Rajendran [13] but required more processing time. Schuster and Framinan [16] proposed a hybrid algorithm that used both SA and GA for solving the problem. Their studies proved that their hybrid algorithm outperform the algorithm provided by Rajendran [13]. Framinan and Schuster [24] improved the algorithm proposed in Schuster and Framinan [16] using an algorithm called complete local search with memory. Grabowski and Pempera [15] developed some Tabu search based algorithms that outperform the previous findings including the one proposed by Rajendran [13]. Schuster [25] introduced a fast Tabu search algorithm for solving the no-wait jobshop scheduling problem and found their algorithm compared well to the algorithm proposed by Schuster and Framinan [16]. Ying et al. [26] proposed self-adaptive and ruin-and-recreate algorithm which improved best known solutions in more than half of the benchmarks. Lin and Ying [27] proposed a meta-heuristic algorithm and that could handle very hard and large problems. Zhao et al. [28] proposed a discrete water wave optimization (DWWO) algorithm and a revised greedy algorithm for the problem and proved that their algorithm outperforms the algorithms in the literature.

The reminder of the paper is organized as follows. Section 2 formally defines the problem. Section 3 describes the solution methods and proposed algorithms in details. In section 4 computational results are proposed. Finally, in section 5, final remarks as well as the discussions and conclusions are provided.

2. PROBLEM FORMULATION

In this section no-wait flowshop problem (F_m |nwt|C_{max}) is defined. The following assumptions and notation are considered for the problem.

2.1. Assumptions

i) No machine is allowed to have more than one job at the moment and if any job is initiated it cannot be interrupted until the procedure is finished completely.

ii) Because of no-wait assumption in the problem, no operation can be paused and no other operation would be done while pausing the other operation.

iii) Machines can become idle during the process.

iv) None of the machines can face break down and there is not maintenance time allowed.

v) There is only a machine of a kind and the jobs do not have the option of choosing between the machines.

vi) All jobs are considered to be known beforehand.

vii) The carrying time between the machines is neglected and that time is considered as the process time of the previous machine.

viii) The setup time is not considered in this problem.

ix) All the data in this problem are considered to be deterministic.

2.2. Notations

n	Number of the jobs that are supposed to be scheduled
m	Number of the machines
$t_{i,j}$	Process time of the ith job on the jth machine
$d_{i,k}$	Minimum delay on the first machine between the start of the ith job and kth job
$[i]$	The job which is processed in ith place
$C_{[i]}$	The total time of the processing of the job in the current place
TFT	The total time of the process of all jobs

Minimum delay on the first machine between the start of the ith job and kth job and the total time of the processing of the job in the current place can be calculated by utilizing equations (1), (2) and (3) respectively:

$$d_{i,k} = t_{i,1} + \max_{2\le j\le m} \left(\sum_{p=2}^{j} t_{ip} - \sum_{p=1}^{j-1} t_{kp}, 0\right) \quad (1)$$

$$C_{[1]} = \sum_{j=1}^{m} t_{[1],j} \quad (2)$$

$$C_{[i]} = \sum_{j=1}^{m} t_{[i],j} + \sum_{k=2}^{i} d_{[k-1],[k]} \quad i = 1,2,3,\dots,n \quad (3)$$

Assuming that all the jobs are available in the beginning of scheduling horizon, total process time can be calculated as follows:

$$\begin{aligned} TFT &= \sum_{i=2}^{n}\left(\sum_{k=2}^{i} d_{[k-1],[k]} + \sum_{j=1}^{m} t_{[i],j}\right) + \sum_{j=1}^{m} t_{[1],j} \\ TFT &= \sum_{i=2}^{n}\sum_{k=2}^{i} d_{[k-1],[k]} + \sum_{i=1}^{n}\sum_{j=1}^{m} t_{[i],j} \\ TFT &= \sum_{i=2}^{n} (n+1-i) d_{[i-1],[i]} + \sum_{i=1}^{n}\sum_{j=1}^{m} t_{i,j} \end{aligned} \quad (4)$$

It is important to note that $\sum_{i=1}^{n}\sum_{j=1}^{m} t_{i,j}$ equals the total sum of all of the jobs in all of the machines and it has a constant value. Also, the value of $d_{i,k}$ depends on the process time of the sequence of jobs (i,k). Therefore, the sequence of two jobs that have the minimum value of $d_{i,k}$ should be chosen as the first two jobs in the process.

3. PROPOSED ALGORITHMS

In this paper, a genetic algorithm and 2 hybrid algorithms that utilize the local search algorithms proposed in Aldowaisan and Allahverdi [14], and Grabowski and Pempera [15] a hybrid algorithm utilizing the both local search methods are proposed.

3.1. Population Size

The first population size (P_s) is calculated according to equation (5):

$$P_s = int[100 \times (1 - e^{-0.01n})] + 20 \quad (5)$$

3.2. Selection

The selection operation used here is a Tournament selection. Tournament selection is a method of selecting an individual from a population of individuals in a genetic algorithm. Tournament selection involves running several tournaments among a few chromosomes chosen at random from the population. The winner of each tournament (the one with the best fitness) is chosen.

3.3. Crossover

In this algorithm, a simplified partially matched crossover (PMX) operator is used. The major merits of this operator would be revealed in problems of a larger size. This is a double-point crossover. Table 1 represents the PMX crossover.

Table -1: PMX crossover

Parent 1	i	h	d	e	f	g	a	c	b	j
Parent 2	h	g	a	b	c	j	i	e	d	f
Offspring 1	i	h	d	b	c	j	a	c	b	j
Offspring 2	h	g	a	e	f	g	i	e	d	f

As it is presented in Table 1, j, c, b are repeated twice in first offspring and the same problem has happened in second offspring for g, f, e and this problem need to be dealt with. After modification of the offsprings, the new offsprings will be as presented Table 2.

Table -2: Offsprings generated from PMX crossover

Offspring 1	i	h	d	b	c	j	a	f	e	g
Offspring 2	f	g	i	b	d	c	h	j	a	e

3.4. Mutation

The mutation function used in this research, chooses two random genes in one parent and replaces them and generates a new offspring.

3.5. Termination Criteria

While dealing with the termination of an algorithm, two opposing options are available:

1) Termination according to a number of iterations
2) Termination according to convergence

Therefore, logical termination criteria should be defined for defined. Two termination criteria are defined here.

i. If the number of the chromosomes with minimum TFT is more than 60% of the population, the iteration is terminated.
ii. If the generation of the population is more than 100, the iteration is terminated.

3.6 Algorithms

3.6.1 Hybrid algorithm with Grabowski and Pempera [15] (Hybrid 1)

1. Set the values of n and m.
2. Calculate population size.
3. m number of the population is calculated according to local search presented by Grabowski and Pempera [15].
4. The rest of the population is calculated randomly.
5. Calculated the value of TFT for each chromosome.
6. Calculated the fitness function as
$$fitness = \max_i TFT - TFT(i)$$
7. Choose a pair of parents according to tournament selection.
8. Utilize crossover and mutation operators over the selected parents.
9. if termination criteria are satisfied, stop. Else, move to step 3.

3.6.2 Hybrid algorithm with Aldowaisan and Allahverdi [14] (Hybrid 2)

1. Set the values of n and m.
2. Calculate population size.
3. a single member of the population is calculated according to local search presented by Aldowaisan and Allahverdi [14].
4. The rest of the population is calculated randomly.
5. Calculated the value of TFT for each chromosome.
6. Calculated the fitness function as
$$fitness = \max_i TFT - TFT(i)$$
7. Choose a pair of parents according to tournament selection.
8. Utilize crossover and mutation operators over the selected parents.
9. if termination criteria are satisfied, stop. Else, move to step 3.

3.6.3 Hybrid algorithm with both algorithms (Hybrid 3)

1. Set the values of n and m.
2. Calculate population size.
3. m+1 number of the population is calculated according to local search presented by Aldowaisan and Allahverdi [14] and Grabowski and Pempera [15].
4. The rest of the population is calculated randomly.
5. Calculated the value of TFT for each chromosome.
6. Calculated the fitness function as
$$fitness = \max_i TFT - TFT(i)$$
7. Choose a pair of parents according to tournament selection.
8. Utilize crossover and mutation operators over the selected parents.

9. if termination criteria are satisfied, stop. Else, move to step 3.

4. COMPUTATIONAL RESULTS

In order to evaluate the efficiency of the proposed algorithms the benchmarks introduced by Taillard [32] are used in this study. The process time of the jobs on the machines is chosen from a set of data uniformly distributed from 0 to 100. Utilizing the genetic algorithm, and the hybrid algorithm of the genetic algorithm with the algorithm proposed by Aldowaisan and Allahverdi [14] and Grabowski and Pempera [15] as well as a hybrid genetic algorithm with both of the algorithms proposed by Aldowaisan and Allahverdi [14] and Grabowski and Pempera [15], the results are presented and compared. This comparison is carried out in between 12 different problems with 20, 30 and 50 jobs on 5, 10, 15 and 20 machines. Each problem is solved for 10 times and in sum, 480 different instances are studied.

4.1. The results of minimum value of TFT

Table 3 represents the minimum value of TFT calculated in all 4 algorithms. The lower the value, the better the algorithm. As it is presented, the minimum values belong to the hybrid genetic algorithm with Grabowski and Pempera [15] and the hybrid algorithm with both of the local search algorithms. Furthermore, as the size of the problem increases the hybrid algorithms outperform the basic genetic algorithm.

Table -3: Minimum value of TFT in 4 algorithms

Jobs	Machines	Genetic	Hybrid 1	Hybrid 2	Hybrid 3
20	5	10676	10690	10909	10592
20	10	15593	15396	15466	15659
20	15	17671	17352	17356	17057
20	20	19281	19010	19282	19010
30	5	28283	26386	27681	26517
30	10	36566	33670	36333	33603
30	15	42019	38742	39565	38569
30	20	47342	44266	46315	44113
50	5	82897	75218	81962	75333
50	10	109154	100026	107356	104622
50	15	131974	119702	127467	123748
50	20	149203	135528	145781	135717

4.2. Average Relative Error

Average relative error represents the error between the best-found answer and the calculated one. The percentage of the average relative error can be calculated according to equation (6):

$$ARE = \frac{C_{avg} - C^*}{C^*} \times 100 \qquad (6)$$

Where C^* represents the minimum value of TFT in each sample and C_{avg} represents the mean of the TFT in each sample for 10 instances.

Table -4: Average Relative Error in 4 algorithms

Jobs	Machines	Genetic	Hybrid 1	Hybrid 2	Hybrid 3
20	5	6.7248	2.4074	5.3274	2.2290
20	10	4.5713	4.3706	4.8272	3.5035
20	15	6.1810	4.9223	5.4183	4.6145
20	20	7.4476	1.0231	5.4345	2.8963
30	5	10.1804	3.7644	7.6105	2.8015
30	10	10.9508	2.9092	10.4963	4.3710
30	15	12.1833	2.2671	7.6539	3.4226
30	20	11.5891	2.2093	7.6539	2.1765
50	5	13.4037	2.0737	11.6032	4.9577
50	10	13.3941	2.7663	10.1364	7.3764
50	15	13.0233	1.8111	9.7860	7.6779
50	20	13.7189	1.8846	9.4726	5.7290

Table 4 represents the average relative error in all 4 algorithms. It is made clear that the hybrid algorithm with Grabowski and Pempera [15] and the one with Aldowaisan and Allahverdi [14] have less average relative error in comparison with two other algorithms. Moreover, if the size of the problem is small, hybrid one with Aldowaisan and Allahverdi [14] has less average relative error than the one with Grabowski and Pempera [15] but when the size of the problem increases hybrid 2 shows more efficiency. Figure 1 represents the information in Table4 in comparative way.

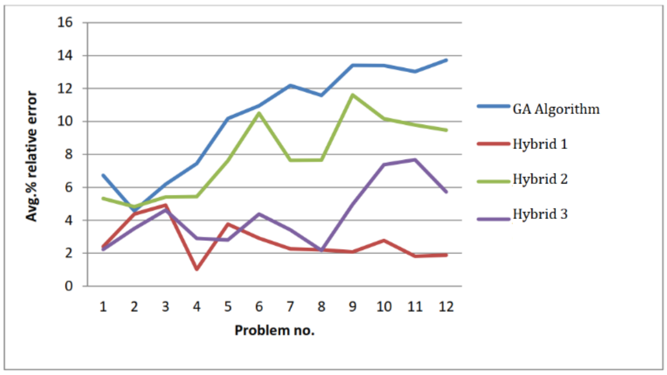

Fig -1: Representation of average relative error

5. DISCUSSION AND CALCULATION

In this paper a no-wait flowshop problem was defined and formulalized. Following that, different algorithms were proposed as the solution concept and their performance in dealing with a benchmark problem was compared. As initial population plays a pivotal role and since generating the first population randomly has been prove to be not that effective, different methods are being utilize in generating first population. In this paper, local search algorithms that are based on the best responding algorithms in the literature are utilize for generating the first population. The results prove that hybridizing the genetic algorithm with different heuristics for generating the first population results in better outcomes in most of the cases. Furthermore, the following suggestions are made for future studies:

1. Modeling the problem in cells of machines and considering more than one machine in each cell.
2. Utilizing different genetic algorithm operators.
3. Utilizing different meta-heuristic algorithms.
4. Doing parameter design for the algorithm according to statistical approaches.
5. Considering the breakdown of the machines in the problem.
6. Considering the capacity of each machine cell.

ACKNOWLEDGEMENT

The author is grateful of the chief editor and anonymous reviewers of the paper for their attention and consideration.

REFERENCES

[1] N. Moradinasab, M. R. Amin-Naseri, T. J. Behbahani, H. Jafarzadeh. "Competition and cooperation between supply chains in multi-objective petroleum green supply chain: A game theoretic approach". Journal of Cleaner Production, vol. 170, 2018, pp.818-841, doi: 10.1016/j.jclepro.2017.08.114.

[2] M. Elyasi, H. Jafarzadeh, F. Khoshalhan. An economical order quantity model for items with imperfect quality: a non-cooperative dynamic game theoretical model. 3rd International Logistics and Supply Chain Conference; 2012; Tehran, Iran.

[3] M. Elyasi, H. Jafarzadeh, F. Khoshalhan. A non-cooperative game theoretic model for EOQ model for items with imperfect quality Proceedings of the 5th International Conference of the Iranian Society of Operations Research, Tabriz, Iran; 2012.

[4] M. Elyasi, F. Khoshalhan, M. Khanmirzaee. "Modified economic order quantity (EOQ) model for items with imperfect quality: Game-theoretical approaches". International Journal of Industrial Engineering Computations, vol. 5(2), 2014, pp. 211-222. doi: 10.5267/j.ijiec.2014.1.003.

[5] M. M. Yenisey, B. Yagmahan. "Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends". Omega, vol. 45, 2014, pp. 119-135, doi: https://doi.org/10.1016/j.omega.2013.07.004.

[6] A. Allahverdi. "A survey of scheduling problems with no-wait in process". European Journal of Operational Research, vol. 255(3), 2016, pp. 665-686,doi: https://doi.org/10.1016/j.ejor.2016.05.036.

[7] J. R. Callahan. "The Nothing Hot Delay Problem in the Production of Steel [microform]: Thesis (Ph.D.)--University of Toronto. (1971).

[8] H. Röck. "The Three-Machine No-Wait Flow Shop is NP-Complete". Journal of the ACM, vol. 31(2), 1984, pp. 336-345, doi: 10.1145/62.65.

[9] H. Jafarzadeh, N. Moradinasab, A. Gerami. "Solving no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines and rework time by the adjusted discrete Multi Objective Invasive Weed Optimization and fuzzy dominance approach". Journal of Industrial Engineering and Management, vol. 10(5), 2017, pp. 887, doi: 10.3926/jiem.2348.

[10] M. C. Bonney, S. W. Gundry. "Solutions to the Constrained Flowshop Sequencing Problem". Operational Research Quarterly (1970-1977), vol. 27(4), 1976, pp. 869-883, doi: 10.2307/3009170.

[11] J. R. King, A. S. Spachis. "Heuristics for flow-shop scheduling". International Journal of Production Research, vol. 18(3), 1980, pp. 345-357, doi: 10.1080/00207548008919673.

[12] R. Gangadharan, C. Rajendran. "Heuristic algorithms for scheduling in the no-wait flowshop". International Journal of Production Economics, vol. 32(3), 1993, pp. 285-290, doi: https://doi.org/10.1016/0925-5273(93)90042-J.

[13] C. Rajendran. "A No-Wait Flowshop Scheduling Heuristic to Minimize Makespan". The Journal of the Operational Research Society, vol. 45(4), 1994, pp. 472-478, doi: 10.2307/2584218.

[14] T. Aldowaisan, A. Allahverdi. "New heuristics for no-wait flowshops to minimize makespan". Computers & Operations Research, vol. 30(8), 2003, pp. 1219-1231, doi: https://doi.org/10.1016/S0305-0548(02)00068-0.

[15] J. Grabowski, J. Pempera. "Some local search algorithms for no-wait flow-shop problem with makespan criterion". Computers & Operations Research, vol. 32(8), 2005, pp. 2197-2212, doi: https://doi.org/10.1016/j.cor.2004.02.009.

[16] C. J. Schuster, J. M. Framinan. “Approximative procedures for no-wait job shop scheduling". Operations Research Letters, vol. 31(4), 2003, pp. 308-318, doi: https://doi.org/10.1016/S0167-6377(03)00005-1.

[17] J. M. Framinan, M. S. Nagano. “Evaluating the performance for makespan minimisation in no-wait flowshop sequencing". Journal of Materials Processing Technology, vol. 197(1), 2008, pp. 1-9, doi: https://doi.org/10.1016/j.jmatprotec.2007.07.039.

[18] S.-W. Lin, K.-C. Ying. “Minimizing makespan for solving the distributed no-wait flowshop scheduling problem". Computers & Industrial Engineering, vol. 99, 2016, pp. 202-209, doi: https://doi.org/10.1016/j.cie.2016.07.027.

[19] M. S. Nagano, H. H. Miyata, D. C. Araújo. “A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times". Journal of Manufacturing Systems, vol. 36, 2015, pp. 224-230,doi: https://doi.org/10.1016/j.jmsy.2014.06.007.

[20] M. Elyasi, H. Jafarzadeh, F. Khoshalhan. Hybrid Ant Colony Optimization (ACO) algorithm for no-wait flow-shop makespan. Proceedings of the 5th International Conference of the Iranian Society of Operations Research, Tabriz, Iran; 2012.

[21] H. Jafarzadeh, N. Moradinasab, M. Elyasi. "An Enhanced Genetic Algorithm for the Generalized Traveling Salesman Problem". Engineering, Technology & Applied Science Research, vol. 7(6), 2017, pp. 2260-2265.

[22] L.-Y. Tseng, Y.-T. Lin. “A hybrid genetic algorithm for no-wait flowshop scheduling problem". International Journal of Production Economics, vol.128(1),2010,pp.144-152,doi: https://doi.org/10.1016/j.ijpe.2010.06.006.

[23] J. Zhu, X. Li, Q. Wang. “Complete local search with limited memory algorithm for no-wait job shops to minimize makespan". European Journal of Operational Research, vol. 198(2), 2009, pp. 378-386,doi: https://doi.org/10.1016/j.ejor.2008.09.015.

[24] J. M. Framinan, C. Schuster. “An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach". Computers & Operations Research, vol. 33(5), 2006,pp.1200-1213,doi: https://doi.org/10.1016/j.cor.2004.09.009.

[25] C. J. Schuster. “No-wait Job Shop Scheduling: Tabu Search and Complexity of Subproblems". Mathematical Methods of Operations Research, vol.63(3),2006,pp.473-491, doi: 10.1007/s00186-005-0056-y.

[26] K.-C. Ying, S.-W. Lin, W.-J. Wu. “Self-adaptive ruin-and-recreate algorithm for minimizing total flow time in no-wait flowshops". Computers & Industrial Engineering, vol. 101, 2016, pp. 167-176,doi: https://doi.org/10.1016/j.cie.2016.08.014.

[27] S.-W. Lin, K.-C. Ying. “Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics". Omega, vol. 64, 2016, pp. 115-125,doi: https://doi.org/10.1016/j.omega.2015.12.002.

[28] F. Zhao, H. Liu, Y. Zhang, W. Ma, C. Zhang. “A discrete Water Wave Optimization algorithm for no-wait flow shop scheduling problem". Expert Systems with Applications, vol. 91, 2018, pp. 347-363,doi: https://doi.org/10.1016/j.eswa.2017.09.028.

[29] E. Taillard. “Benchmarks for basic scheduling problems”. European Journal of Operational Research, vol. 64, 1993, pp. 278-285.