An Optimization Algorithm for Optimal Problem of Permutation Flow Shop Scheduling

Abdel Nasser H. Zaied
Dean, Faculty of Computer and Informatics, Zagazig University, Zagazig, Egypt

Mahmoud M. Ismail
Decision Support System, Zagazig University, Zagazig, Egypt

Shimaa S. Mohamed*
Decision Support System, Zagazig University, Zagazig, Egypt

ABSTRACT
Nowadays the permutation flow shop scheduling problems become one of the most important problems in scheduling field. In this paper whale optimization algorithm was modified for solving PFSP. WOA is new meta-heuristic was proposed by Sayedali and Andrew in 2016 that was inspired from the nature of humpback whales movements in hunting prey. The modification is depending on two stages: firstly; WOA algorithm is converted to discrete algorithm to deal with PFSP; secondly; the mutation permutation strategy was used to improve the results of WOA. The modified algorithm is implemented on MATLAB workspace. The modified algorithm is tested with various benchmark datasets available for flow shop scheduling. The statistical results prove that the modified algorithm (MWOA) is competent and efficient for solving flow shop problems.

Keywords
Permutation flow shop scheduling problem, Whale Optimization Algorithm, meta-heuristic algorithm, MWOA.

1. INTRODUCTION
Permutation flow shop scheduling problem is known as combinatorial optimization problem [1]. Due to its importance in manufacturing, production and mathematical branched and PFSP was become one of the most important problem in scheduling field, nowadays this made it an area of interest for researchers [2]. There are three methods for solving the PFSP. First method; exact algorithm, this kind of algorithms is implemented for solving small scale problems and not suitable for complex problem that have more than few jobs and/or machines because of increasing in computational time. Second method; heuristic algorithm deal with large scale problems the quality of the solution is often not high .third method; meta-heuristic algorithm also deal with large scale but it is better than heuristic algorithm and it is developed by mimic nature of certain phenomena and processes, which is generally from a solution as the starting point, it will continue to search the search space until approximate optimal solution [1], [3].

Nowadays, researchers proposed different meta-heuristics to solve the flow shop scheduling problems by obtaining the minimum makespan. M. F. Tasgetiren et al. [4] solved the flow shop scheduling problems using a particle swarm optimization algorithm (PSO). Hongcheng et al. [5] proposed a hybrid particle swarm optimization with estimation of distribution algorithm (PSO-EDA) for solving permutation flow shop scheduling problem. A discrete African wild DOG algorithm also was used for solving flow shop scheduling problem by authors in [2]. Vanita et al. [6] improved differential evolutionary (DE) algorithm for solving permutation flow shop scheduling problem. Also Simon el al. [7] solved the permutation flow shop problem with Firefly Algorithm (FA). A. Baskar [8] minimized the makespan in Permutation flow shop scheduling problems using simulation. Hong-Qing et al. [1] modified cuckoo search (MCS) algorithm for solving permutation flow shop problem. Also Kannan et al. [9] proposed a hybrid approach for solving traditional flow shop scheduling problems to minimize the makespan (total completion time). They solved scheduling problems by using a combination of Decision Tree (DT) and Scatter Search (SS) algorithms. They also evaluated the performance of the different proposed meta-heuristics so in this paper is used new meta-heuristic algorithm to improve quality of solutions of flowshop scheduling problem.

Recently, Sayedali Andrew [10] proposed new meta-heuristic algorithm called whale optimization algorithm (WOA) that was inspired from the nature of humpback whales whale movements in hunting prey. WOA algorithm could well balance of exploration and exploitation phase that assists this algorithm to find the global optimum. WOA had the potential to be very effective in solving real problems with unknown search spaces as well. Also high exploration and exploitation ability of WOA leads this algorithm towards the local optima [10]. So in this paper WOA algorithm was used for solving PFSP, then WOA algorithm was modified to improve results. This paper is organized as following: in section 2, formulation of permutation flow shop scheduling problem is described. In section 3, whale optimization algorithm is described. In section 4, modified whale optimization algorithm with inertia weight (MWOA) is explained. In section 5, experimental results are discussed. Finally, in section 6, conclusion and future work is described.

2. FORMULATION OF PERMUTATION FLOWSHOP SCHEDULING PEOBLEM
The objective of the permutation flowshop scheduling problem is to determine the best order for processing all jobs on all machines to minimize total completion time of jobs (makespan (Cmax)).

2.1 Mathematical Model of PFSP
The permutation flowshop scheduling problem with n jobs and m machines [1], [4]:

- n is number of jobs unlimited j=1,2,……………,n
- m is number of machines is unlimited k=1,2,……………,m
- processing time of each job on machine k is given (tj,k)
- the sequence of jobs in the machine is the same of all machines

- each job is processed on only one machine, at the same time, each machine could process only one machine
- π (π1, π2, ………., πn) is the job mutation
- C(πj,k) represent the completion time of job πj on machine m
- tπj,k is the processing time of job πj on machine k

The formulation of n-job m-machine problem is described as following:

$$C(\pi_{1,1}) = t_{\pi_{1,1}}$$

$$C(\pi_{j,1}) = C(\pi_{j-1,1}) + t_{\pi_{j,1}} \qquad j=2,\ldots\ldots\ldots,n$$

$$C(\pi_{1,k}) = C(\pi_{1,k-1}) + t_{\pi_{1,k}} \qquad k=2,\ldots\ldots\ldots,m$$

$$C(\pi_{j,k}) = \max(C(\pi_{j-1,k}), C(\pi_{j,k-1})) + t_{\pi_{j,k}}$$
j=2,………..,n k=2,…………,m

So, makespan can be described as:

$$C_{max}(\pi) = C(\pi_{n,m})$$

The objective of scheduling is to find a best permutation π*, from all permutation ∏

$$C_{max}(\pi^*) = (\pi_{n,m}) \qquad \pi \in \prod$$

For example, assume that there are 2 jobs and 3 machines and the sequence of jobs was [2 1] with processing time of each job in machine shown in table (1), calculate completion of jobs based the given sequence [7].

Table 1. Simple example 1

	Job1	Job2
Mac1	2	4
Mac2	3	2
Mac3	4	5

The following Gantt chart illustrate the completion time of given sequence Cmax(π) = 15 where π=(2,1) as shown in figure(1).

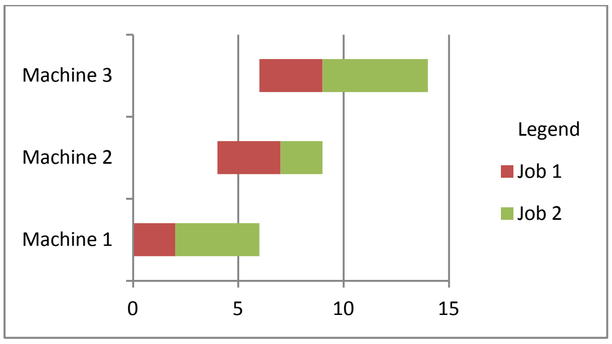


Fig 1: Gantt Chart

3. WHALE OPTIMIZATION ALGORITHM

Sayedali and Andrew [10] suggested a novel whale optimization meta-heuristic algorithm that was inspired from the bubble-net hunting technique of humpback whales. The WOA simulate the special pursuing behavior of humpback whales, in which the whales try to encircle the prey near the surface of the water while creating bubbles that are in the shape of a circle. In the bubble-net hunting mechanism, the humpback whales dive almost 12 meters down and then start to make bubbles in a spiral shape around the prey and swim toward the surface.

3.1 Encircling prey

Humpback whales can notice the prey location and surrounded them. For the unknown position of the optimal design in the search space, the current best candidate solution is the target prey or is close to the optimal in the WOA algorithm. After selecting the best search agent, other search agents will try to update their positions to the best search agent. From this behavior the following equations are proposed:

$$\vec{D} = |\vec{C}.\vec{X}^*(t) - \vec{X}(t)| \qquad (1)$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A}.\vec{D} \qquad (2)$$

Where, t: represent the current iteration, A and C are coefficient vectors, X* is the position vector of the best solution obtained so far, X is the position vector, | | is the absolute value, and · is an element-by-element multiplication.

The vectors A and C are calculated as follows:

$$\vec{A} = 2\,\vec{a}\,.\vec{r} - \vec{a} \qquad (3)$$

$$\vec{C} = 2\,.\vec{r} \qquad (4)$$

Where $\vec{a}$ is linearly decreased from 2 to 0 over the course of iterations (in both exploration and exploitation phases) and $\vec{r}$ is a random vector in [0, 1].

3.2 Bubble-net attacking method (exploitation phase)

To build the mathematical model for the bubble-net behavior of humpback whales, two approaches are modeled as follows:

1. Shrinking encircling approach: This approach is achieved by decreasing the value of $\vec{a}$ in the Eq. (3). Note that the fluctuation range of $\vec{A}$ is also decreased by $\vec{a}$. In other words $\vec{A}$ is a random value in the interval [−a, a] where a is decreased from 2 to 0 over the course of iterations. Setting random values for $\vec{A}$ in [−1, 1], the new position of a search agent can be defined anywhere in between the original position of the agent and the position of the current best agent.

2. Spiral updating position: after searching humpback whales for the prey then a spiral equation is then created between the position of whale and prey to update the position of humpback whales as follows:

$$\vec{X}(t+1) = \vec{D}'.e^{bl}.\cos(2\pi l) - \vec{X}^*(t) \qquad (5)$$

Where $\vec{D}' = |\vec{C}.\vec{X}^*(t) - \vec{X}(t)|$ and indicates the distance of the ith whale to the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, l is a random number in [−1, 1], and . is an element-by-element multiplication.

Note that humpback whales swim around the prey within a shrinking circle and along a spiral-shaped path at one time. To model this synchronous behavior, assume that there is a 50% probability of choose either the shrinking encircling technique or the spiral model to update the whales position during optimization. The mathematical model is as follows:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A}.\vec{D}, & \text{if } p < 0.5 \\ \vec{D}\,.\,e^{bl}.\cos(2\pi l) - \vec{X}^*(t)\,, & if\ p \geq 0.5 \end{cases} \quad (6)$$

Where p is a random number in [0, 1], In addition to the bubble-net strategy, the humpback whales have another behavior of searching for prey randomly. This behavior represented as following:

3.3 Search for prey (exploration phase)

The same mechanism based on differentiate of the A vector can be used to look for prey as exploration phase. In fact, humpback whales look for randomly based on the position of each other. Therefore, A used with the random values more than 1 or less than −1 to force look for agent to move far away from a reference whale. In contrast to the exploitation phase, the position of a search agent was updated in the exploration phase randomly based on selected search agent rather than the best search agent appeared so far.| A | > 1 in this approach is used to confirm exploration and allow the WOA algorithm to execute a global search, as showed in the following equations:

$$\vec{D} = \vec{C}.\ \overrightarrow{X_{rand}} - \vec{X} \quad (7)$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}.\vec{D} \quad (8)$$

Where $\overrightarrow{X_{rand}}$ is a random position vector selected from the current population;

Researchers in [11] improved WOA algorithm by changing in mechanism of updating whale position in each iteration based on added new parameter an inertia weight ω∈ [0, 1] is random number between 0 1nd 1 but the mechanism of search for the best solution was as the basic algorithm (A new control parameter, inertia weight, is introduced). Researchers introduced the inertia weight into WOA to obtain an improve whale optimization algorithm (IWOA) for high dimensional continuous function optimization problems and to tune the influence on the current best solution. This change was in two phase as following:

- In Encircling prey, the updated method is represented by the following equations:

$$\vec{D} = \left|\vec{C}.\ \omega\vec{X}^*(t) - \vec{X}(t)\right| \quad (9)$$

$$\vec{X}(t+1) = \omega\vec{X}^*(t) - \vec{A}.\vec{D} \quad (10)$$

- In Spiral updating position, the updated method is represented by the following equations:

$$\vec{X}(t+1) = \vec{D}\,.\,e^{bl}.\cos(2\pi l) - \omega\,\vec{X}^*(t) \quad (11)$$

$$\vec{X}(t+1) = \begin{cases} \omega\vec{X}^*(t) - \vec{A}.\vec{D}, & \text{if } p < 0.5 \\ \vec{D}\,.\,e^{bl}.\cos(2\pi l) - \omega\vec{X}^*(t) & if\ p \geq 0.5 \end{cases} \quad (12)$$

Improved Whale optimization algorithm (WOA) can be summarized in the pseudo code shown in Algorithm 1 in figure (2) [10], [11].

```
------------------------------------------------------------------------
Algorithm 1 IWOA Algorithm
------------------------------------------------------------------------
Initialize the whales population Xi (i = 1; 2;……..; n)
  Calculate the fitness of each search agent
  X* = the best search agent
  while (t < maxi iteration)
    for each search agent
    Update a, A, C, l, p, and ω
    if1 (p < 0.5)
    if2 (|A| < 1)
    Update the position of the current search agent by Eq. (10)
    else if2 (|A| ≥ 1)
    Select a random search agent (Xrand)
    Update the position of the current search agent by Eq. (8)
    end if2
    else if1 (p ≥ 0.5)
    Update the position of the current search by Eq. (11)
    end if1
    end for
    Check if any search agent goes beyond the search space and amend it
    Calculate the fitness of each search agent
    Update X* if there is a better solution
    t = t+1
    end while
    return X*
------------------------------------------------------------------------
```

Fig 2: The pseudo code of IWOA

4. MODIFIED WHALE OPTIMIZATION ALGORITHM WITH INTERIA WEIGHT (MWOA)

The MWOA algorithm begins with a set of random solutions based on number of search agent, and then evaluates all search agents to determine the initial best solution. At each iteration, search agents update their positions with respect to either a randomly selected search agent or the best solution obtained so far based on the previous equations.

4.1 Representation of Solution

Firstly: WOA is proposed for solving continuous problems, but PFSP problem is discrete problem, so algorithm was converted to make suitable for our problem by sorting and indexing solution. A simple example of the rule is shown in table 2. In table 2, there are eleven jobs, so the job is from 1 to 11, the candidate solution which was supposed is: positioni = [0.9883, 0.9652, 0.2030, 0.3092, 0.7251, 0.2849, 0.1744, 0.7852, 0.9984, 0.5376, 0.7035], by sorting the positioni in descending order. Finally, the order of job is the processing order π = [7_3_6_4_10_11_5_8_2_1_9] which refer to the candidate solution to the PFSP.

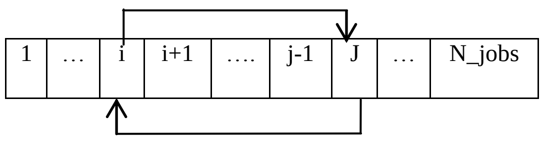
Table 2. Simple example 2

Jobs	1	2	3	4	5	6	7	8	9	10	11
position	.98	.96	.20	.30	.72	.28	.17	.78	.99	.53	.70
Π	7	3	6	4	10	11	5	8	2	1	9

Secondly: at the end of each iteration there is current candidate solution, this permutation solution is token, mutation permutation function is used to change permutation of jobs in scheduling problem. This mechanism consisted from 3 strategies developed in the following:

4.2 Swap Strategy

Random number is selected from 1 to 3 to choice between three strategies and i, j random number from 1 to number of jobs (N_jobs =11); Swap is the first strategy can be implemented. This strategy is shown in figure (3):



7	**3**	6	4	10	**11**	5	8	2	1	9

Parent (i=2 and j=6)

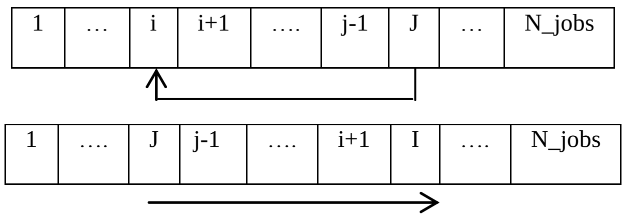
7	**11**	6	4	10	**3**	5	8	2	1	9

Offspring

Fig 3: An example of Swap

4.3 Reversion Strategy

Reversion is the second strategy that can be implemented by selection two random numbers from 1 to number of jobs (for example [4, 7]); i= min (two selected numbers), j=max (two selected numbers). This strategy is shown in figure (4):



7	3	6	**4**	10	11	**5**	8	2	1	9

Parent (i=4 and j=7)

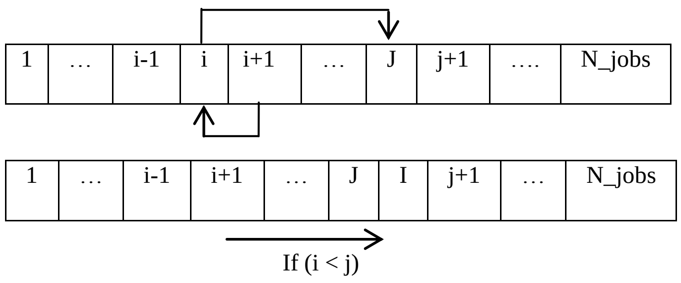
7	3	6	**5**	**11**	**10**	**4**	8	2	1	9

Offspring

Fig 4: An example of Reversion

4.4 Insertion Strategy

Insertion is the third strategy that can be implemented by selection two random numbers from 1 to number of jobs. This strategy is shown in figure (5a) and (5b):



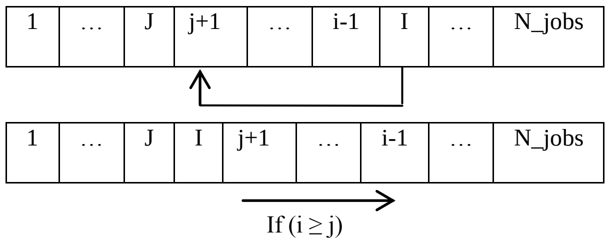
7	3	6	4	**10**	11	5	**8**	2	1	9

Parent (i=5 and j=8)

7	3	6	4	**11**	**5**	**8**	**10**	2	1	9

Offspring

Fig 5a: An example of Reversion



7	3	6	**4**	10	11	5	8	**2**	1	9

Parent (i=9 and j=4)

7	3	6	4	**2**	**10**	**11**	**5**	**8**	1	9

Offspring

Fig 5b: An example of Reversion

Also to improve the exploration, local optimal optima avoidance, exploitation and convergence of the WOA, the value of C vector in equation 4 that is important in determine the current best candidate solution in local search is replaced with Lévy flight equation used in dragonfly algorithm that is shown in the following equation [12].

$$\vec{C} = 2 \,.\, \text{Lévy}\,(\,) \qquad (13)$$

$$\text{Lévy}\,(\,) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \qquad (14)$$

where r1, r2 are two random numbers in [0,1], β is a constant (equal to 1.5 in this work), and σ is calculated as follows:

$$\sigma = x\left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}}\right)^{1/\beta} \qquad (15)$$

where Γ(x) = (x-1)!.

This update can maximize the diversification of search agents, which guarantees that the algorithm can explore the search place efficiently and accomplish local minima avoidance. This finding implies that the Lévy flight trajectory is helpful in obtaining a better trade-off between exploration and exploitation in the WOA.

The pseudo code of modified whale optimization algorithm is shown in Algorithm 2 in figure (6):

```
-------------------------------------------------------------------------
                        Algorithm 2 MOWA Algorithm
-------------------------------------------------------------------------
  Initialize the whales population Xi (i = 1; 2;……..; n)
  Calculate the fitness of each search agent
  X* = the best search agent
  while (t < maxi iteration)
  for each search agent
  Update a, A, l, p, and ω
  Update C vector based on Eq. (13)
  if1 (p < 0.5)
  if2 (|A| < 1)
  Update the position of the current search agent by Eq. (10)
  else if2 (|A| ≥ 1)
  Select a random search agent (Xrand)
  Update the position of the current search agent by Eq. (8)
```

```
end if2
else if1 ($p \geq 0.5$)
Update the position of the current search by Eq. (11)
end if1
end for
For each iteration
Create new solution by using the mutation permutation function
Calculate the fitness of the new solution
If3 (new solution ≤ fitness)
Update the position of current candidate solution based on the new solution of mutation function
end if3
end
Check if any search agent goes beyond the search space and amend it
Calculate the fitness of each search agent
Update X* if there is a better solution
t = t+1
end while
return X*
---------------------------------------------------------------------------
```

Fig 6: The pseudo code of MWOA

5. EXPERIMENTAL RESULTS

5.1 Testing Benchmarks

To test the quality of solution of proposed algorithm in solving permutation flowshop scheduling problem to obtain the minimum makespan, 54 instances is used that were selected from OR-Library benchmarks. The first eight instances are proposed by Carlier in [13], the next six instances were called rec01; rec03 through rec11 that are proposed by Reeves in [14] and the last 40 instances (Ta01-Ta40) are given by Taillard in [15].

5.2 The Carlier and Reeves datasets of PFSP

All computational experiments are implemented on MATLAB R2015a, and run on Intel (R) Core(TM) i7-4510U CPU, 2.60 GHz with 8 GB RAM. Each instance is run for 20 times with population size (500) and search agents (30). To compare between proposed algorithm and other meta-heuristic algorithms average Error Ratio (ERi) is used and it is calculated as the following equation:

$$ER_i = \frac{C^*(x) - Opt.}{Opt.} * 100\%$$

Where C*(x) is the makespan that obtained by proposed algorithm and Opt. is best known or optimal value available in the previous literature. The performance of the proposed work for Carlier and Reeves datasets are compared with MCS (Modified Cuckoo Search) [1], AWDAA [2] , hybridized version of PSO with EDA [5] and DT+SS [9]. Table 3 show total completion time (makespan) obtained from using WOA for solving permutation flowshop scheduling problem instances and from using Modified Whale Optimization Algorithm (MWOA) and from using previous researches for each instance.

Table 3. Results of MWOA algorithm for Carlier and Reeves datasets

P	n,m	Optimal Value available in the Literature	Results of the WOA algorithm	Error Ratio	Results of the MWOA algorithm	Error Ratio
Car1	11,5	7038	7038	0	7038	0
Car2	13,4	7166	7166	0	7166	0
Car3	12,5	7302	7312	0.14	7312	0.14
Car4	14,4	8003	8003	0	8003	0
Car5	10,6	7720	7767	0.61	7720	0
Car6	8,9	8505	8505	0	8505	0
Car7	7,7	6590	6590	0	6590	0
Car8	8,9	8366	8366	0	8366	0
Rec01	20,5	1247	1293	3.69	1247	0
Rec03	20,5	1109	1131	1.98	1109	0
Rec05	20,5	1242	1269	2.17	1245	0.24
Rec07	20,10	1566	1584	1.15	1566	0
Rec09	20,10	1537	1599	4.03	1537	0
Rec11	20,10	1431	1472	2.86	1431	0

Table 4 shows error ratio values for all fourteen datasets along with literature results. From table 4, except for the Car-3 instance AWDAA is the best but other 13 instances modified algorithm obtained optimal value such as modified cuckoo search algorithm. The proposed approach yielded much better solution quality compared to various hybrid algorithms, meta-heuristics and heuristics for Carlier and Reeves datasets.

Table 4. Average Error Ratio of algorithms

P	n,m	Modified WOA (MWOA)	WOA	MCS	AWDAA	PSO-EDA	DT+SS
Car1	11,5	0	0	0	0	0	0
Car2	13,4	0	0	0	0	0	0
Car3	12,5	0.14	0.14	0.14	0	0.14	0.14
Car4	14,4	0	0	0	0	0	0
Car5	10,6	0	0.61	0	0.93	0	0
Car6	8,9	0	0	0	2.35	0	0.02
Car7	7,7	0	0	0	0	0	0
Car8	8,9	0	0	0	0	0	0
Rec01	20,5	0	3.69	0	0.08	0.13	0
Rec03	20,5	0	1.98	0	0.63	0.04	0
Rec05	20,5	0.24	2.17	0.24	1.77	0.24	0.24
Rec07	20,10	0	1.15	0	0.38	0	0
Rec09	20,10	0	4.03	0	0.39	0.42	0
Rec11	20,10	0	2.86	0	8.04	0.50	0

5.3 The Taillard Datasets of PFSP

In this section, 40 well-known benchmark instances given by Taillard were experimented. For comparing modified algorithm efficiency, most literature resources used an upper bound and a few authors used a lower bound of Taillard to test dataset instances. Because most of the literature used upper bound, the proposed approach also uses upper bound as optimal value and compares it with the other algorithms from the literature such as Scatter Search (SS) [16], Self-guided GA [17], NEH+SA+C [18], and PSOspv [4]. Table 5 shows the test results of the modified algorithm along with completion time and Table 6 shows average error ratio of various literatures on all 40 instances.

From Table 6, it is seen that for all 10 problems of 20×5 dataset, the modified algorithm yields an optimal value and much lower average error ratio for 20×10 datasets. In the 20×20 and 50×5 datasets showed that hybrid algorithm of NEH, SA and composite heuristics produced lower values.

Table 5. Results of proposed method for Taillard (1993) datasets

P	n×m	Optimal Value available in the Literature	Results of the MWOA algorithm	Error Ratio
Ta01	20×5	1278	1278	0
Ta02		1359	1359	0
Ta03		1081	1081	0
Ta04		1293	1293	0
Ta05		1235	1235	0
Ta06		1195	1195	0
Ta07		1234	**1239**	0.41
Ta08		1206	1206	0
Ta09		1230	1230	0
Ta10		1108	1108	0
Average				0.04
Ta11	20×10	1582	1583	0.06
Ta12		1659	1674	0.90
Ta13		1496	1508	0.80

P	**n×m**	**Optimal Value available in the Literature**	**Results of the MWOA algorithm**	**Error Ratio**
Ta14		1377	1385	0.58
Ta15		1419	1430	0.77
Ta16		1397	1404	0.50
Ta17		1484	1484	0
Ta18		1538	1546	0.52
Ta19		1593	1603	0.63
Ta20		1591	1599	0.50
Average				0.526
Ta21	20×20	2297	2307	0.43
Ta22		2099	2119	0.95
Ta23		2326	2344	0.77
Ta24		2223	2235	0.54
Ta25		2291	2317	1.13
Ta26		2226	2230	0.18
Ta27		2273	2291	0.79
Ta28		2200	2218	0.82
Ta29		2237	2242	0.22
Ta30		2178	2198	0.92
Average				0.675
Ta31	50×5	2724	2724	0
Ta32		2834	2838	0.14
Ta33		2621	2621	0
Ta34		2751	2753	0.07
Ta35		2863	2864	0.03
Ta36		2829	2831	0.07
Ta37		2725	2725	0
Ta38		2683	2683	0
Ta39		2552	2555	0.12
Ta40		2782	2782	0
Average				0.043

Table 6. Average Error Ratio of algorithms

Problem	**n×m**	**Modified WOA (MWOA)**	**SS**	**Self-guided GA**	**NEH+SA+C**	**PSO_{SPV}**
Ta01-Ta10	20×5	**0.04**	0.33	1.10	0.91	1.75
Ta11-Ta20	20×10	**0.526**	0.88	1.90	0.665	3.25
Ta21-Ta30	20×20	0.675	0.497	1.60	**0.459**	2.82
Ta31-Ta40	50×5	0.043	0.166	0.52	**0.025**	1.14

5.4 Statistical Test

The Statistical tests were performed on all datasets. For Carlier and Reeves datasets, the error ratio for all the instances of the instances were specified in the previous researches and it is decided to use a Friedman’s Test for these two datasets. In the Taillard dataset, most researchers specified an error ratio of population means of dataset. For example Ta01-Ta10 consists of 10 instances, so most researchers would use an average error ratio of 10 instances: they would not specify error ratio of individual instances. Accordingly, a Friedman’s Test is used for grouped Taillard instances.

From Table 9, it is clear that for Carlier and Reeves problems sets modified algorithm (MWOA) & MCS gives statistically the same average error ratio and are more statistically significant than the other methods. Also from Table 10, it is clear that for Taillard problem sets modified algorithm (MWOA) & NEH+SA+C gives statistically the same average error ratio and are more statistically significant than the other methods.

Table 7. Test Statistics for Carlier and Reeves

N	14
Chi-Square	23.640
Df	5
Asymp. Sig.	.000
significance level α = 0.05	

Table 8. Ranks of algorithms for Carlier and Reeves datasets

	Mean Rank
MWOA	2.82
WOA	4.57
MCS	2.82
AWDAA	4.25
PSO_EDA	3.54
DT_SS	3.00

Table 9. Test Statistics for Taillard dataset

N	4
Chi-Square	13.400
Df	4
Asymp. Sig.	.009
significance level α = 0.05	

Table 10. Ranks of algorithms for Taillard dataset

	Mean Rank
MWOA	1.75
SS	2.50
Self_guided_GA	4.00
NEH_SA_C	1.75
PSO_SPV	5.00

6. CONCLUSION AND FUTURE WORK

Based on our knowledge, this is the first attempt of the whale optimization algorithm to solve the permutation flow shop scheduling problem with the makespan criteria in the literature. A modified WOA algorithm is proposed for solving PFSP. Three strategies of mutation methodology and Levy flight function are added to a whale optimization algorithm for improving WOA algorithm’s performance. The performance of the modified whale optimization algorithm is tested using the benchmark problem such as Carlier, Reeves and Taillard datasets. The statistical results show that the proposed algorithm is considerably effective to the PFSP. The proposed algorithm can be used in the future for solving other mathematical and industry problems to obtain the best solutions and researchers can made more improving on the whale optimization algorithm by using another optimization techniques to obtain optimal solution. In the future work, the modified algorithm may be used in hybrid with other heuristics and meta-heuristics and researchers can improve t. It would be interesting to apply the proposed algorithm to solve other stochastic scheduling problems with single or multiple objective functions in the future.

7. REFERENCES

[1] Hong-Qing. Z, Yong-Quan. Z, and Cong. X, 2016. “Modified Cuckoo Search Algorithm for Solving Permutation Flow Shop Problem”, Springer International Publishing Switzerland, pp. 714–721.

[2] M. K. Marichelvam1 & M. Geetha, 2013. “Solving Flowshop Sceduling Problem using a Discrete African Wild DOG Algorithm”, ICTACT journal on Soft Computing, pp. 555 – 558.

[3] Deepanshu .A and Gopal .A, 2016. “Meta-heuristic approaches for flowshop scheduling problems: a review”, Advanced Operations Management, pp. 1 – 16.

[4] M. F. Tasgetiren, Yun-Chia. L, Mehmet. S and Gunes. G, 2007. ‘”A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem”, European Journal of Operational Research, pp. 1930–1947.

[5] Hongcheng. L, Liang. G and Quanke. P, 2011. “A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem”, Expert Systems with Applications, pp. 4348–4360.

[6] Vanita. G. T and Pravin. K, 2013. “Solving Permutation Flowshop Scheduling Problem Using Improved Differential Evolutionary Algorithm”, International Journal of Engineering Research & Technology (IJERT), pp. 456-461.

[7] Simon. F, Hui-long. L, Yan. Z, Suash. D , Thomas. H, 2014. “Solving the Permutation Flow Shop Problem with Firefly Algorithm”, 2nd International Symposium on Computational and Business Intelligence, pp. 25-29.

[8] A. Baskar, 2015. “Minimizing the Makespan in Permutation Flow Shop Scheduling Problems using Simulation”, Indian Journal of Science and Technology, pp. 1-7.

[9] Kannan. G, R. Balasundaram, N. Baskar and P. Asokan, 2017. "A Hybrid Approach for Minimizing Makespan in Permutation Flowshop Scheduling", Systems Engineering Society of China and Springer-Verlag Berlin Heidelberg, pp. 50-76.

[10] Seyedali. M and Andrew. L, 2016. "The Whale Optimization Algorithm",Advances in Engineering Software, pp. 51–67.

[11] Hongping. H, Yanping. B and Ting. X, 2016. "A whale optimization algorithm with inertia weight", WSEAS Transaction on Computers, pp. 319-326.

[12] Seyedali. M, 2016. "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems", Neural Comput & Applic, pp. 1053–1073.

[13] Carlier. J, 1978. "Ordonnancements a contraintes disjonctives", R.A.I.R.O. Recherche Operationelle/Oper. Res. 12, 333–351.

[14] Reeves. C. R, 1995. "A genetic algorithm for flow-shop sequencing", Comput. Oper. Res. 22, 5–13.

[15] Taillard, E, 1993. "Benchmarks for basic scheduling instances", European Journal of Operational Research, 64(2): 278-285.

[16] M. Saravanan, A. N. Haq, A. R. Vivekraj and T. Prasad, 2008. "Performance evaluation of the scatter search method for permutation flowshop sequencing problems", Int J Adv Manuf Technol, pp.1200–1208.

[17] Shih-Hsin. Ch, Pei-Chann. Ch, T. C. E. Cheng, Qingfu. Z, 2012. "A Self-guided Genetic Algorithm for permutation flowshop scheduling problems", Computers & Operations Research, pp.1450–1457.

[18] Dipak. L and Uday. K. Ch, 2009. "An efficient hybrid heuristic for makespan minimization in permutation flow shop scheduling",Int J Adv Manuf Technol, pp.559–569.