Hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem

Article *in* Future Generation Computer Systems · March 2018

CITATIONS
198

READS
1,288

3 authors, including:

Mohamed Abdel-Basset
Zagazig University
263 PUBLICATIONS **6,891** CITATIONS

SEE PROFILE

Future Generation Computer Systems 85 (2018) 129–145

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem

Mohamed Abdel-Basset [a,*], Gunasekaran Manogaran [b], Doaa El-Shahat [a], Seyedali Mirjalili [c]

[a] Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah, Egypt
[b] University of California, Davis, United States
[c] School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia

HIGHLIGHTS

- An attempt is made to solve the NP-hard type combinatorial problems using WOA.
- The HWA integrates the WOA with a local search strategy for solving PFSSP.
- The proposed algorithm is validated using the benchmark problems.
- Results proved the effectiveness of the proposed algorithm.

ARTICLE INFO

Article history:
Received 5 February 2018
Received in revised form 3 March 2018
Accepted 10 March 2018
Available online 19 March 2018

Keywords:
Flow shop scheduling
Makespan
Whale optimization algorithm
Hybrid algorithm
Local search

ABSTRACT

The flow shop scheduling problem is one of the most important types of scheduling with a large number of real-world applications. In this paper, we propose a new algorithm that integrates the Whale Optimization Algorithm (WOA) with a local search strategy for tackling the permutation flow shop scheduling problem. The Largest Rank Value (LRV) requires the algorithm to deal with the discrete search space of the problem. The diversity of candidate schedules is improved using a swap mutation operation as well. In addition to the insert-reversed block operation is adopted to escape from the local optima. The proposed hybrid whale algorithm (HWA) is incorporated with Nawaz–Enscore–Ham (NEH) to improve the performance of the algorithm. It is observed that HWA gives competitive results compared to the existing algorithms.

1. Introduction

Recently, much attention is paid towards the flow shop scheduling problem (FSSP) because of its vital role in the procurement, production, computing designs, distribution, transportation, information processing and communications. FSSP is considered as an NP-hard problem since finding a solution in a polynomial time is difficult. Because of the importance of this problem, several attempts have been made in the literature to develop algorithms that achieve two objectives: minimizing the makespan of the best schedule as well as reducing the time complexity. FSSP was first introduced and formulated by Johnson 1954 [1]. Johnson obtained the optimal scheduling for the two-machine problem as well as the optimal scheduling for the three-machine problem but in a restricted case. Later, Nawaz et al. [2] introduced a heuristic algorithm for tackling *m*-machine and *n*-job FSSP called NEH (Nawaz–Enscore–Ham). NEH was compared with 15 algorithms and the results show that NEH works well for large FSSP. Although the time complexity of NEH was not excessive, it was much larger than the existing algorithms.

After NEH was proven to be effective, several algorithms used it in the literature. Marichelvam et al. [3] incorporated NEH with the cuckoo search algorithm to constitute an improved cuckoo search algorithm (ICS). Lately, Marichelvam et al. [4] proposed a hybrid monkey search algorithm (MSA) which was integrated with the dispatching rules, the shortest processing time, the longest processing time, and NEH heuristics to improve MSA performance. The author showed that MSA gives competitive results when it is compared with NEH. Liu et al. [5] designed an effective differential evolution algorithm (DE). Several methods are adopted to improve the DE efficiency such as Individual Improving Scheme (IIS), greedy based local search, and NEH heuristic. The greedy based local search was employed for escaping from the local minimum. IIS was

* Corresponding author.
E-mail addresses: analyst_mohamed@yahoo.com (M. Abdel-Basset), gmanogaran@ucdavis.edu (G. Manogaran), doaazidan@zu.edu.eg (D. El-Shahat), seyedali.mirjalili@griffithuni.edu.au (S. Mirjalili).

https://doi.org/10.1016/j.future.2018.03.020
0167-739X/

used to enhance the diversity in the population and ameliorate the solutions.

Rathinam et al. [6] used the decision tree algorithm (DT) to minimize the makespan of FSSP. Also, Govindan et al. [7] provided a hybrid algorithm that combined the scatter search algorithm with the decision tree (ADE). ADE outperformed other existing hybrid algorithms. A main drawback of ADE and DT algorithms is the increasing size of the tree as the number of jobs is increased in the large FSSP. Gao et al. [8] proposed two heuristic algorithms that combined the standard deviation heuristic, Bertolliso heuristic, and procedure of constructive heuristic. The algorithms were integrated with a local search strategy to ameliorate the quality of solutions. The results showed the effectiveness of the proposed algorithms, especially for the large FSSP. Tseng and Lin [9] developed a hybrid genetic algorithm that employs two different local search strategies: insertion search (IS) and the insertion search repair and cut (ISRC). The orthogonal-array-based crossover is used to increase the algorithm performance. Table 1 summarizes most of the algorithms developed for FSSP in the literature.

Although many algorithms have been developed to solve FSSP, there are still some disadvantages to these algorithms such as local optima and high computational cost especially when solving large-scale problems. All these algorithms seek to achieve objectives such as minimizing the makespan or the total flow time. Whale Optimization Algorithm (WOA) is novel algorithm that attracts the attention in many applications as power system [29], neural networks [30], and siting of capacitors in radial distribution network [31] and photovoltaic cells [32]. This motivates us to propose a hybrid whale algorithm that can minimize the makespan of the FSSP. HWA can converge quickly towards optimal solutions using different techniques such as local search strategy, swap mutation, insert-reversed block operation and combining a heuristic algorithm called NEH. All these techniques boost the performance of HWA and the experimental results show that HWA is an interesting algorithm and gives better results than the existing algorithms. The rest of this paper is organized as follows. Section 2 introduces the definition of the problem. Section 3 describes the standard whale optimization algorithm. Section 4 presents the proposed algorithm HWA. Section 5 shows the experimental results of HWA and Section 6 provides conclusions and future works. Note that the symbols used in this paper and their meaning as shown in Table 2.

2. Permutation flow shop scheduling problem

The Permutation Flow Shop Scheduling Problem (PFSSP) refers to the problem of running n jobs on m machines sequentially and on the same permutation with the objective of minimizing the makespan. It can be characterized by the following main points:

- Each job j_k can run on each machine only once where $k = 1, \ldots, n$
- Each machine i_h can process only one job at a time where $h = 1, \ldots, m$
- Each job j takes a time to be processed on each machine i namely, the processing timept
- The processing time of each job is changed depending on the machine as each job will run on machine 1 to m orderly.
- The processing time of each job includes the set-up time plus the running time on a machine.
- It is assumed that the start time is$zero$
- Each job has a completion time c to finish its processing in a machine that takes two parameters $c(j_k i_h)$ as it is the completion time of job j on a machine$i$.
- The objective of PFSSP is to find the best permutation of jobs that minimizes the makespan c^*which is defined as the completion time of the last job on the last machine.

PFSSP can be formulated as follows:

$$c\left(j_1, i_1\right) = pt_{j_1,i_1}, \quad k = 1, h = 1 \tag{1}$$

$$c\left(j_1, i_h\right) = c\left(j_1, i_{h-1}\right) + pt_{j_1,i_h}, \quad h = 2, \ldots m \tag{2}$$

$$c\left(j_k, i_1\right) = c\left(j_{k-1}, i_1\right) + pt_{j_k,i_1 i_1}, \quad k = 2, \ldots, n \tag{3}$$

$$c\left(j_k, i_h\right) = \max\left(c\left(j_{k-1}, i_h\right), c\left(j_k, i_{h-1}\right)\right) + pt_{j_k,i_h}, \quad k = 2, \ldots, n, h = 2, \ldots, m. \tag{4}$$

The permutation is the different order of the jobs on machines. The makespan is the completion time of the last processed job on the last machine which can be defined for each permutation as follows:

$$c^* = c\left(j_n, i_m\right). \tag{5}$$

The best permutation is defined as the permutation that has the minimum makespan known as (c^*) among all permutations.

3. Whale optimization algorithm

The WOA algorithm [33] is a novel meta-heuristic that mimics the social behavior of humpback whales in chasing their prey. The main steps of the standard WOA are described in algorithm 1. WOA implements the following strategies used by the humpback whales such as bubble-net attacking and searching for the prey.

3.1. Bubble-net attacking

While two or more whales swim in a shrinking circle, they produce bubbles to surround the prey along a spiral-shaped path. Some whales or search agents thrust the prey towards the surface while the others lead the prey towards the net. The positions of the search agents are updated according to the position of the best search agent. The strategy of encircling the prey can be formulated as follows:

$$D = |C \cdot w^* - x^t| \tag{6}$$

$$x^{t+1} = w^* - A \cdot D \tag{7}$$

$$A = 2a \cdot r - a \tag{8}$$

$$C = 2 \cdot r \tag{9}$$

where D is the distance between the current search agent x^t and the best search agent w^* at t iteration. A is a random value in the range $[-a, a]$ r is a random number in [0, 1]. Through iterations the value of a is decreasing from 2 to 0 and C is a coefficient. Thebehavior of the spiral-shaped path can be formulated as:

$$x^{t+1} = D' \cdot e^{bl} \cdot \cos\left(2\pi l\right) + w^* \tag{10}$$

$$D' = |w^* - x^t| \tag{11}$$

where D' is the absolute value for the distance between x^t and w^*. b defines the shape of the logarithmic spiral. l is a random value $\in [-1, 1]$. WOA implements the two behaviors of bubble-net

Table 1
The contributions made for solving the FSSP.

Author and reference	Year	Algorithm	Abbreviation	Summary
Ishibuchi et al. [10]	2003	Multi-objective genetic local search	MOGLS	– MOGLS is improved by selecting good individuals as initial solutions for the local search
Lian et al. [11]	2006	Novel particle swarm optimization	NPSO	– NPSO is integrated with mutation and crossover operations
Tasgetiren et al. [12]	2007	Discrete differential evolution with variable neighborhood descent	DDE_{VND}	– DDE is combined with variable neighborhood descent. – Initial population is constructed using NEH and NN (Nearest neighbor) heuristics. – The individual is updated using two-cut PTL crossover and the mutation operation.
Qian et al. [13]	2008	Hybrid differential evolution	HDE	– Largest-Order-Value is employed to convert continuous space into job permutation. – A simple local search is used to ensure exploitation.
Zobolas et al. [14]	2009	Hybrid meta-heuristic algorithm ($NEGA_{VNS}$)	$NEGA_{VNS}$	– $NEGA_{VNS}$ combined GA with VNS (Variable Neighborhood Search). – GA is combined with three heuristics: NEH, CDS heuristic of Campbell [15], Palmer's heuristic [16], and Gupta's heuristic [17].
Sayadi et al. [18]	2010	Discrete firefly	Discrete firefly	– The solution are ameliorated by a local search mechanism
Wang et al. [19]	2011	Hybrid modified global-best Harmony Search	hmgHS	– NEH is used in the initial harmony memory with a certain level. – Largest Position Value is used to convert continuous position into discrete one.
Ahmadizar [20]	2012	New Ant Colony Algorithm	NACA	– The generated sequence by Palmer's slope index [16] is considered as a seed sequence of NACA. – Local search procedure is adopted to improve the quality of the solutions.
Li and Yin [21]	2013	Opposition-based differential evolution algorithm	ODDE	– NEH is combined with the initial population. – Largest Rank Value is used to convert the continuous position of DE into discrete one. – Opposition based learning (OBL) is used to accelerates DE toward the global optimal solution.
Fong et al. [22]	2014	Firefly algorithm	FA	– Ranked Order Value (ROV) is used to convert the continuous search space of FA into discrete one. – NEH is used to initialize some individuals and the remaining individuals are randomly generated.
Luo et al. [23]	2014	Discrete bat algorithm	DBA	– Intensive Virtual Population Neighborhood Search is embedded to ameliorate the quality of solutions through using swap, insert, and single point move backward operations.
Saravanan et al. [24]	2015	Bicriteria-greedy randomized adaptive search procedure	B-GRASP	– GRASP was first addressed for bi-criteria flow shop scheduling problem.
Lin et al. [25]	2015	Hybrid backtracking search algorithm	HBSA	– Backtracking search algorithm is hybridized with crossover, mutation operations, simulated annealing mechanism, and random insertion local search.
Ding et al. [26]	2015	Tabu-mechanism improved iterated greedy	TMIIG	–Tabu search is added to the iterated greedy algorithm. –Neighborhood search operations are used such as swap, insert, and double-insert moves.
Ding et al. [27]	2015	Block-shifting simulated annealing	BSA	– BSA is integrated with the block-shifting operator based on k-insertion moves. – It has simple implementation and it is very effective.
Sanjeev et al. [28]	2016	Modified gravitational emulation local search algorithm	MGELS	– MGELS replaces the fitness of the new sequence with the parent sequence if it is better.

attacking encircling: the prey circle and the spiral movement with equal probabilities as follows:

$$x^{t+1} = \begin{cases} w^* - A \cdot D & p < 0.5 \\ D' \cdot e^{bl} \cdot \cos(2\pi l) + w^* & p \geq 0.5. \end{cases} \quad (12)$$

3.2. Searching for the prey

WOA makes a tradeoff between the exploration, which can be done by selecting a random search agent x_{rand}, and the exploitation, which is done by selecting the best search agent. Searching for the

Table 2
Summarizing the meaning of the used symbol.

Symbol	Meaning
n	Number of jobs
m	Number of machines
j	Job
i	Machine
j_k	a job in the sequence of n jobs
i_h	A machine in the sequence of m machines
pt	Processing time
$c(j_k i_h)$	Completion time of job j_k on a machine i_h
c^*	Completion time of executing the last job in the last machine
LSP	Local search probability
BS	Block size in the insert-reversed operation

prey can be defined as:

$$D = |C \cdot x_{rand} - x^t| \quad (13)$$

$$x^{t+1} = x_{rand} - A \cdot D. \quad (14)$$

4. The proposed algorithm

Since WOA has been proposed for tackling continuous problems, it is incorporated with several methods to solve PFSSP as a discrete problem. We proposed a hybrid whale algorithm that combines WOA with a local search strategy, swap mutation, and the insertion of reversed-block operations to enhance the performance of WOA. HWA consists of five main steps: representation, evaluation, swap mutation, inserting reversed-block operation, and the local search strategy. Fig. 1 shows the framework of HWA. All these five steps will be discussed in the following subsections.

Algorithm 1 The standard WOA

Initialize a population of n random search agents
Evaluate all n search agents
w^* = the best search agent
$t = 0$
While ($t < iterations$)
  for each search agent
    Update WOA parameters (a, $A, L, C,$ and p)
    if ($p < 0.5$)
      if ($|A| < L$)
        $x^{t+1} = w^* - A.D$
      else if ($|A| \geq L$)
        $x^{t+1} = x_{rand} - A.D$
      end if
    else if ($p \geq 0.5$)
      $x^{t+1} = D'.e^{bl}.\cos(2\pi l) + w^*$
    end if
  end for
  Evaluate the search agent x^{t+1}
  Update w^* if x^{t+1} if better
  $t = t + 1$
end while
return w^*

4.1. Representation

PFSSP is an NP-hard combinatorial problem which is represented by a discrete search space. As WOA is used for the continuous problems, we propose HWA for solving the PFSSP. At first, an initial population (pop) of distinct random numbers is generated. We must make sure that there is no any repeated numbers in the same search agent as shown in Fig. 2.

As the solution of the problem is a permutation of job sequence, we have to find a way to map the continuous search agents into a job sequence permutation. The previous study by Li and Yin [21] shows that LRV is an effective way of mapping the continuous values into job permutation. In LRV, the continuous values are ranked by in a decreasing order. As mentioned before, we must avoid any duplication of numbers in the population because of using LRV that will generate a wrong job permutation. In Fig. 3, the wrong job permutation can be produced because of contradiction to the PFSSP conditions (job 1 runs on two machines at the same time).

NEH is an efficient algorithm for solving the scheduling problem [2]. 10% of the initial population is initialized with NEH. To produce the search agent using NEH, we follow these steps:

1. Calculate the total processing time for each job.
2. Sort the jobs $j = 1, \ldots, n$ in a non-increasing order according to their total processing time.

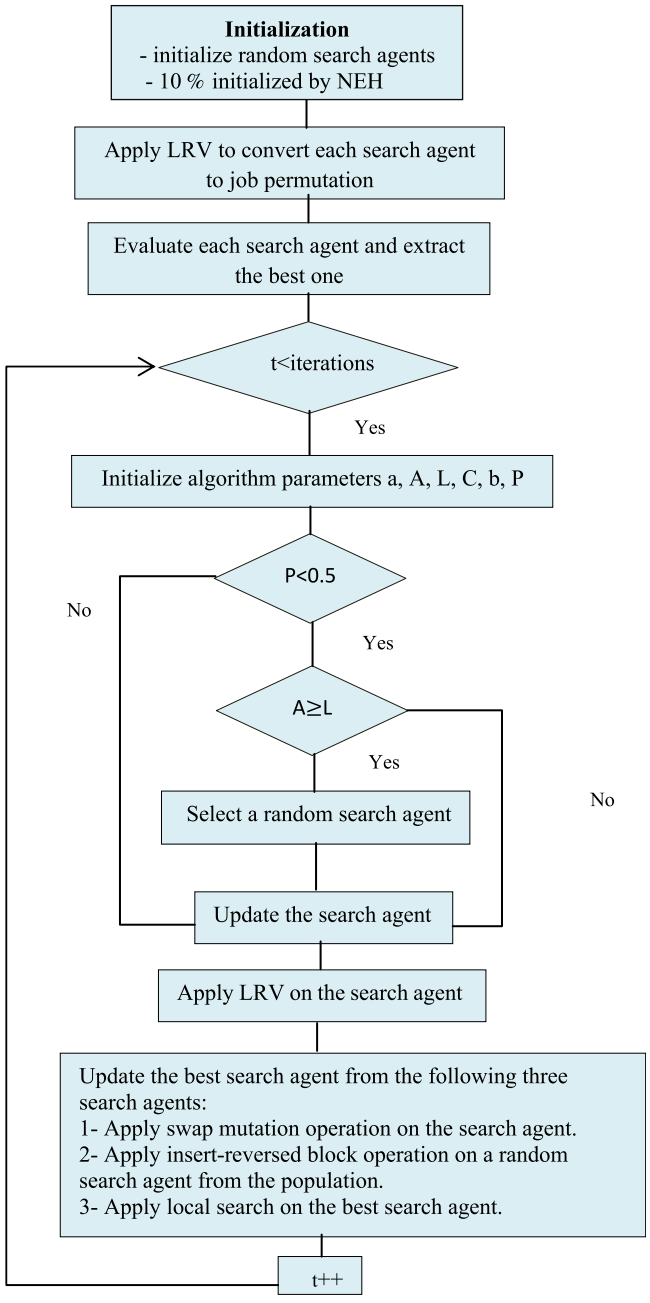


Fig. 1. Framework of HWA.

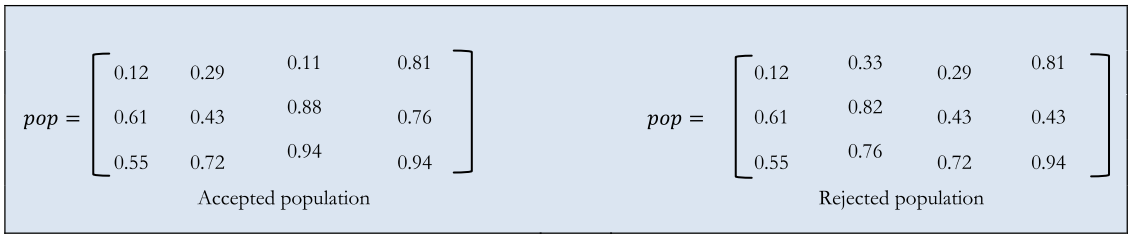
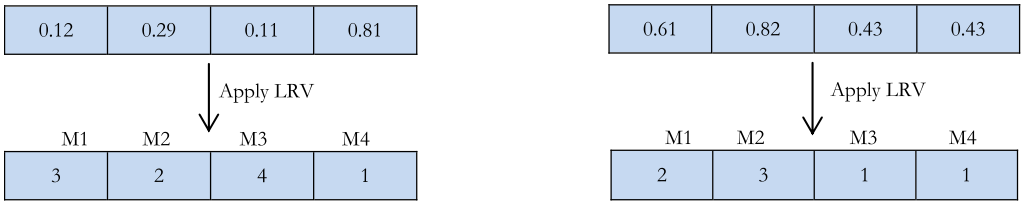


Fig. 2. Initial population.



(a) Correct job permutation.

(b) Wrong job permutation.

Fig. 3. LRV representation.

Table 3
PFSSP example.

	M1	M2	M3
J1	5	6	11
J2	8	4	7
J3	11	9	3
J4	14	15	20

Table 4
The calculation of the makespan for the job Seq. 1–4–3–2.

	M1	M2	M3
J1	5	11	22
J4	19	34	54
J3	30	43	57
J2	38	47	64

Table 5
Evaluation of different job sequences.

Seq. No.	Job Seq.	$c(n, m)$
1	2–3–4–1	79
2	4–1–3 –2	70
3	3–1–2–4	73
4	4–3–2–1	70
5	1–4–3–2	64

Table 6
The parameters of the proposed algorithm.

Parameter	Value
Population size	50
Number of iterations	3000
LSP	0.01
Number of runs	20

3. Select the first two jobs. Evaluate the different permutation of the two jobs. Choose the best partial permutation.
4. Add the job in the third rank to the partial permutation obtained in step 2 to all the three possible positions. Evaluate all the possible partial permutations and choose the best permutation of minimum makespan.
5. Add the next job in the next rank to the last best partial permutation. Evaluate the job in all the possible positions of the partial permutation and choose the best partial permutation.

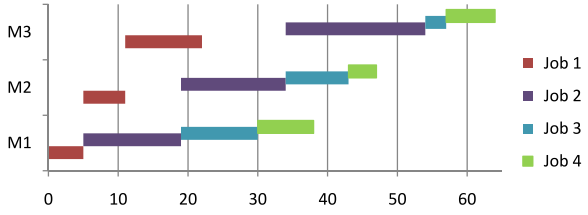


Fig. 4. The Gantt chart for the job sequence 1–4–3–2.

6. Repeat step 5 until all jobs are scheduled in the job permutation.

The search agent generated by NEH is set to 10 % of the initial population. The standard WOA is applied to the initial population and the new search agent positions are updated. Again, the WOA may generate the same value in same the search agent. HWA can overcome this problem by generating a random number that is not found in the search agent. Then, this continuous search agent is mapped to a job permutation using LRV.

4.2. Evaluation

Once the population is generated, the search agents are evaluated. Each search agent is a permutation of job. The makespan for each search agent in the population is computed. The search agent that has the minimum makespan is extracted as the best one. For example, we consider a PFSSP as described in Table 3. Table 4 presents the makespan of the job sequence 1–4–3–2. The makespan is computed using the Eqs. (1)–(4). In Table 5, five job sequences are provided and the maximum makespan ($c(n, m)$) of each sequence is recorded. It can be seen the job sequence 1–4–3–2 has the minimum makespan $c^* = 64$. Fig. 4 shows the Gantt chart for the job sequence 1–4–3–2.

4.3. Swap mutation

In this step, two random positions in the search agent are chosen. The two jobs in those two random positions are replaced with each other. The swap mutation operation is described in Fig. 5. In

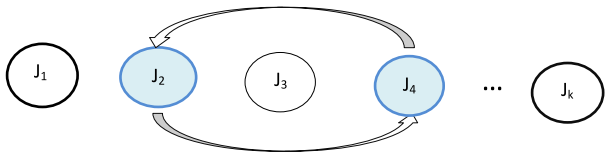


Fig. 5. Swap mutation operation.

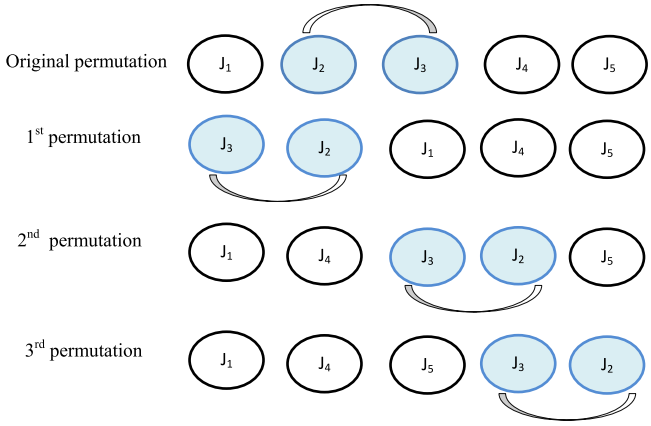


Fig. 6. Insert-reversed block operation.

HWA, there is a relation between the occurrence of swap operation and the number of jobs. The swap operation is repeated by $0.01 * n$ where n is the number of jobs.

4.4. Insert-reversed block operation

This operation inserts a reversed block of job permutation in all $(d-1)$ possible positions in a d-dimensional search agent. This operation can be described as follows:

- Choose two random positions in the search agent.
- Reverse the order of jobs from the first random position to the second random position (reversed block).
- Cut the reversed block from the original sequence.
- Insert the reversed block in all $(d-1)$ possible positions in the original sequence and evaluate each new permutation.
- If the new permutation is better than the original permutation then set the new permutation as the original one. The insert-reversed operation is described in Fig. 6.

4.5. local search strategy

The local search strategy is employed for ameliorating the quality of the best schedule W^*as follows:

- For each job in W^* that has a dimension d, move the job to all the other $(d-1)$ places while keeping the rest of W^*
- For each move, evaluate the new schedule if better set $W^* =$ new schedule.
- The local search is done using a small probability *LSP*.

Finally, the proposed algorithm HWA is presented in algorithm HWA is introduced in Algorithm 2. It is worth mentioning that the value of parameter p is changed at each iteration as described by Kaveh and Ghazaan [34] as follows:

$$p = 0.3 * \frac{iter}{\max\_iter}. \quad (15)$$

It should be noted that changing the value of p at each iteration improves the results.

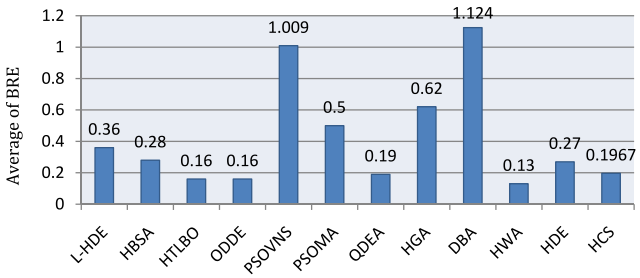


Fig. 7. Comparison on average of BRE on benchmarks Car and Rec.

Algorithm 2 The proposed HWA

Initialize a population of n random whales or search agents
Apply LRV on each search agent to be mapped into job permutation
Solve the PFSSP using NEH
Choose about 10 % random search agents from the population and replace them with NEH
Evaluate each search agent
W^* = the best search agent
Find W^*
$t = 0$
While ($t < \max\_iter$)
  for each search agent in the population
    Update WOA parameters (a, A, C, L, and p)
    if ($p < 0.5$)
      if ($|A| < L$)
        $x^{t+1} = W^* - A.D$
      else if ($|A| \geq L$)
        $x^{t+1} = x_{rand} - A.D$
      end if
    else if ($p \geq 0.5$)
      $x^{t+1} = D'.e^{bl}.\cos(2\pi l) + W^*$
    end if
  end for
  Ensure no repeated values in the same search agent x^{t+1}
  Apply LRV on each search agent x^{t+1}
  for i = 0:0.01*n
    Perform swap mutation on the search agent x^{t+1}
    if (evaluate (x^{t+1}) < evaluate (W^*))
      $W^* = x^{t+1}$
    end if
  end for
  Perform insert-reversed block operation on a random search agent x_r
  if (evaluate (x_r) < evaluate (W^*))
    $W^* = x_r$
  end if
  W^* =Perform local search strategy on the best search agent W^*
  Evaluate the search agent
  Update W^* if there is a better solution in the population
  $t = t + 1$
end while
return W^*

5. Experimental results

In the following experiments, the effectiveness of the proposed algorithm is tested on four widely-used and well-known benchmarks. The first benchmark is the Carlier dataset [35] which contains eight instances. The second benchmark consists of 21 instances introduced by Reeves [36]. The third benchmark is introduced by Heller [37] and includes only two instances. These benchmarks are available at http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt. The fourth benchmark is the Taillard dataset found at http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html. It consists of 120

Table 7
Comparison on Car and Rec benchmarks.

Problem	$n * m$	c^*		L-HDE	ODDE	PSOVNS	PSOMA	DBA	HBSA	HCS	HTLBO	QDEA	HGA	HBA	HWA
Car01	11 * 5	7038	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			WRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	–	–	–	**0**
Car02	13 * 4	7166	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			WRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	–	–	–	**0**
Car03	12 * 5	7312	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	0.420	**0**	0.397	0.06	**0**	**0**	**0**	**0**	–	**0**
			WRE	**0**	**0**	1.189	**0**	1.190	1.19	**0**	–	–	–	–	**0**
Car04	14 * 4	8003	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			WRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	–	–	–	**0**
Car05	10 * 6	7720	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	0.039	0.018	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			WRE	**0**	**0**	0.389	0.375	**0**	**0**	**0**	–	–	–	–	**0**
Car06	8 * 9	8505	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	0.076	0.114	**0**	**0**	**0**	**0**	**0**	0.04	–	**0**
			WRE	**0**	**0**	0.764	0.764	**0**	**0**	**0**	–	–	–	–	**0**
Car07	7 * 7	6590	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			WRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	–	–	–	**0**
Car08	8 * 8	8366	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			ARE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	**0**
			WRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	–	–	–	–	**0**
REC01	20 * 5	1247	BRE	**0**	**0**	0.160	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
			ARE	**0**	**0**	0.168	0.144	0.080	0.14	0.016	**0**	0.112	0.14	0.092	**0**
			WRE	**0**	**0**	0.321	0.160	0.160	0.16	0.1602	–	–	–	–	**0**
REC03	20 * 5	1109	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
			ARE	**0**	**0**	0.158	0.189	0.081	0.08	**0**	**0**	0.009	0.09	**0**	**0**
			WRE	**0**	**0**	0.180	0.721	0.180	0.18	**0**	–	–	–	–	**0**
REC05	20 * 5	1242	BRE	0.2415	**0**	0.242	0.242	0.242	0.24	0.2415	**0**	0.242	**0**	**0**	**0**
			ARE	0.2415	0.170	0.249	0.249	0.242	0.24	0.2415	**0**	0.242	0.29	0.121	**0**
			WRE	0.2415	0.242	0.420	0.402	0.242	0.24	0.2415	–	–	–	–	**0**
REC07	20 * 10	1566	BRE	**0**	**0**	0.702	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
			ARE	**0**	**0**	1.095	0.986	0.575	0.46	**0**	**0**	**0**	0.69	0.032	**0**
			WRE	**0**	**0**	1.405	1.149	1.149	1.15	**0**	–	–	–	–	**0**
REC09	20 * 10	1537	BRE	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
			ARE	0.0260	**0**	0.651	0.621	0.638	0.07	**0**	**0**	**0**	0.64	**0**	**0**
			WRE	0.2602	**0**	1.366	1.691	2.407	0.65	**0**	–	–	–	–	**0**
REC11	20 * 10	1431	BRE	**0**	**0**	0.071	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
			ARE	**0**	**0**	1.153	0.129	1.167	**0**	**0**	**0**	**0**	1.10	**0**	**0**
			WRE	**0**	**0**	2.656	0.978	2.655	**0**	**0**	–	–	–	–	**0**
REC13	20 * 15	1930	BRE	**0**	**0**	1.036	0.259	0.415	0.10	**0**	**0**	0.104	0.36	0.005	**0**
			ARE	0.2746	0.124	1.79	0.893	1.461	0.53	0.1917	**0**	0.225	1.68	0.114	**0**
			WRE	0.7772	0.259	2.643	1.502	3.782	1.14	0.3109	–	–	–	–	**0**
REC15	20 * 15	1950	BRE	**0**	**0**	0.769	0.051	0.154	0.05	**0**	**0**	**0**	0.56	**0**	**0**
			ARE	0.5231	0.062	1.487	0.628	1.226	0.64	0.1128	**0**	0.158	1.12	0.092	**0**
			WRE	1.1795	0.564	2.256	1.076	2.103	1.18	0.4615	–	–	–	–	**0**
REC17	20 * 15	1902	BRE	**0**	**0**	0.999	**0**	0.368	**0**	**0**	**0**	**0**	0.95	**0**	**0**
			ARE	0.3628	**0**	2.453	1.330	1.277	1.00	0.1157	**0**	0.126	2.32	0.078	**0**
			WRE	0.9464	**0**	3.365	2.155	2.154	2.16	0.4732	–	–	–	–	**0**
REC19	30 * 10	2093	BRE	0.2867	0.239	1.529	0.43	0.573	0.29	0.1433	0.239	0.287	0.62	0.262	**0**
			ARE	0.7023	0.373	2.099	1.313	0.929	0.81	0.5638	0.373	0.435	1.32	0.593	**0.257**
			WRE	1.2422	0.860	2.532	2.102	2.023	1.29	0.7645	–	–	–	–	**0.287**
REC21	30 * 10	2017	BRE	0.6445	0.149	1.487	1.437	1.438	0.69	0.2975	0.149	0.149	1.44	0.412	**0**
			ARE	1.2791	0.999	1.671	1.596	1.671	1.50	1.1651	0.999	1.041	1.57	1.298	**0.113**
			WRE	1.4378	1.438	2.033	1.636	2.231	1.83	1.4378	–	–	–	–	**0.198**
REC23	30 * 10	2011	BRE	0.3481	0.149	1.343	0.596	0.796	0.45	0.248	0.149	0.348	0.40	0.265	**0**
			ARE	0.4276	0.438	2.106	1.310	1.173	1.28	0.4625	0.438	0.597	0.87	0.321	**0.254**
			WRE	0.4973	0.497	2.884	2.038	2.381	3.08	0.4973	–	–	–	–	**0.398**
REC25	30 * 15	2513	BRE	0.5571	0.199	2.388	0.835	1.632	0.40	0.2786	0.199	0.119	1.27	0.004	**0**
			ARE	1.0824	0.454	0.160	2.085	2.921	1.29	0.8754	0.454	0.995	2.54	0.782	**0.040**
			WRE	1.6315	1.035	0.168	3.233	3.940	2.43	0.1.1540	–	–	–	–	**0.119**
REC27	30 * 15	2373	BRE	0.2528	0.126	1.728	1.348	1.011	0.25	0.2528	0.126	0.253	1.10	0.227	**0**
			ARE	0.8512	0.607	2.463	1.605	1.419	1.27	0.8175	0.607	0.954	1.83	0.73	**0.079**
			WRE	1.2221	0.927	3.203	2.402	2.298	2.57	1.0535	–	–	–	–	**0.126**

(*continued on next page*)

Table 7 (*continued*)

Problem	$n * m$	c^*		L-HDE	ODDE	PSOVNS	PSOMA	DBA	HBSA	HCS	HTLBO	QDEA	HGA	HBA	HWA
REC29	30 ∗ 15	2287	BRE	0.8308	**0**	1.968	1.442	1.049	0.57	0.0875	**0**	**0**	1.40	0.381	**0**
			ARE	1.0494	0.770	3.109	1.888	2.580	1.42	0.8833	0.770	0.824	2.70	0.671	**0**
			WRE	1.443	1.137	4.067	2.492	3.935	2.97	1.1369	–	–	–	–	**0**
REC31	50 ∗ 10	3045	BRE	0.427	**0.263**	2.594	1.510	2.299	0.43	**0.263**	**0.263**	**0.263**	0.43	0.343	**0.263**
			ARE	0.644	0.302	3.232	2.254	3.392	1.91	0.8998	0.302	0.565	1.34	1.196	**0.263**
			WRE	0.920	0.427	4.237	2.692	4.532	2.66	1.3793	–	–	–	–	**0.263**
REC33	50 ∗ 10	3114	BRE	**0**	**0**	0.835	**0**	0.610	**0**	**0**	**0**	**0**	**0**	0.008	**0**
			ARE	0.244	**0**	1.007	0.645	0.728	0.59	**0.5652**	**0**	0.297	0.78	0.362	**0**
			WRE	0.835	**0**	1.477	0.834	1.734	1.28	0.8349	–	–	–	–	**0**
REC35	50 ∗ 10	3277	BRE	**0**	**0**	**0**	**0**	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
			ARE	**0**	**0**	0.038	**0**	0.037	**0**	**0**	**0**	**0**	**0**	**0**	**0**
			WRE	**0**	**0**	0.092	**0**	0.092	**0**	**0**	–	–	–	–	**0**
REC37	75 ∗ 20	4951	BRE	2.565	1.737	4.383	2.101	3.373	1.92	1.6360	1.737	**1.717**	3.75	2.08	1.737
			ARE	3.001	2.165	4.949	3.537	4.872	2.93	2.2541	2.165	2.771	4.90	2.522	**1.90**
			WRE	3.555	2.868	5.736	4.039	5.979	4.20	2.5045	–	–	–	–	**2.363**
REC39	75 ∗ 20	5087	BRE	1.730	**0.649**	2.850	1.553	2.280	0.90	**0**.8060	**0.649**	0.845	2.20	1.028	**0.649**
			ARE	1.832	0.869	3.371	2.426	3.851	1.88	1.2463	0.869	1.485	2.79	1.423	**0.831**
			WRE	2.005	**0.983**	3.951	2.830	5.347	3.38	1.5530	–	–	–	–	**0.983**
REC41	75 ∗ 20	4960	BRE	2.661	**1.008**	4.173	2.641	3.810	1.69	**1**.4516	**1.008**	1.190	3.64	1.485	**1.008**
			ARE	3.350	2.085	4.867	3.684	5.095	2.72	2.1532	2.085	1.965	4.92	2.468	**1.925**
			WRE	3.770	2.742	5.585	4.052	6.532	3.55	2.4597	–	–	–	–	**2.459**

Bold values indicate the best results.

Table 8
Comparison on Heller benchmark.

Instance	$n * m$	Upper bound	HWA	NEH	SG	DSOMA
Hel1	20 ∗ 10	516	**515**	518	**515**	631
Hel2	100 ∗ 10	136	**135**	141	137	150

Bold values indicate the best results.

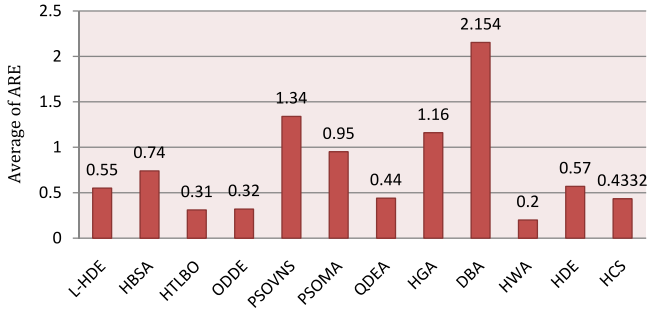


Fig. 8. Comparison on average of ARE on benchmarks Car and Rec.

instances provided by Taillard [38]. The proposed algorithm HWA is encoded in Java and run on a laptop with the following capabilities: Intel Core i5-3317U CPU @ 1.70 GHz with 4.0 GB Memory in windows 8.1 with 64-bit operating system. The parameters of the proposed algorithm are presented in Table 6. The block size is set as follows:

$$BS = \begin{cases} n/2 & \text{if } n < 10 \\ 5 & \text{otherwise} \end{cases} \tag{16}$$

where BS is the block size and n is the number of jobs. It is worth mentioning that the local search probability $LSP = 0.01$ for all benchmarks except for Taillard $500 * 20$ instance, $LSP = 0.001$.

5.1. The comparison on Carlier, Reeves, and Heller benchmarks

In this experiment, the performance of HWA is compared with other eleven algorithms on two benchmarks Car and Rec. The algorithms are as follows including conventional and recent algorithms:

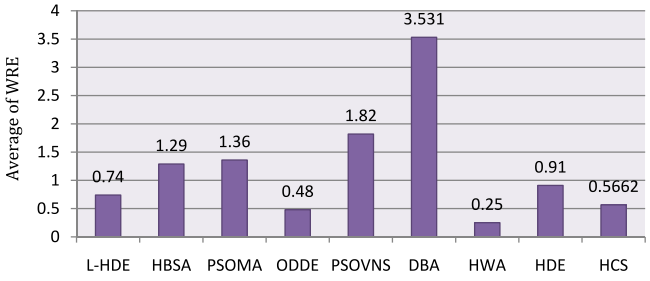


Fig. 9. Comparison on average of WRE on benchmarks Car and Rec.

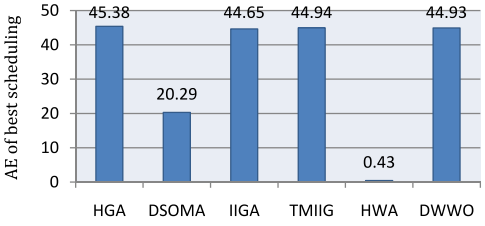


Fig. 10. Comparison on AE of the best scheduling on Taillard benchmark.

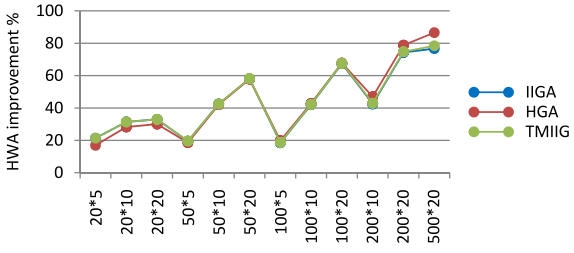


Fig. 11. IP of HWA based on the best scheduling.

Table 9
Comparison on Taillard benchmark.

Problem	Size	NCS		HWA	
		Mean	ARPD	Mean	ARPD
Ta010	20 ∗ 5	1 108	0.00	**1 108**	**0.00**
Ta020	20 ∗ 10	1 606	0.94	**1 591**	**0.00**
Ta030	20 ∗ 20	2 184	0.28	**2 178**	**0.00**
Ta040	50 ∗ 5	2 782	0.00	**2 782**	**0.00**
Ta050	50 ∗ 10	3 131.2	2.16	**3 081.7**	**0.54**
Ta060	50 ∗ 20	3 860.6	2.78	**3 772.38**	**0.44**
Ta070	100 ∗ 5	5 326	0.08	**5 328**	0.09
Ta080	100 ∗ 10	5 891.4	0.79	**5 871.91**	**0.46**
Ta090	100 ∗ 20	6 602.8	2.62	**6 531.54**	**1.52**
Ta100	200 ∗ 10	10 734	0.55	**10 727**	**0.49**
Ta110	200 ∗ 20	11 633.6	3.06	**11 521.46**	**2.07**
Ta120	500 ∗ 20	26 897.2	1.66	**26 697.22**	**0.91**
	Average		1.24		**0.54**

Bold values indicate the best results.

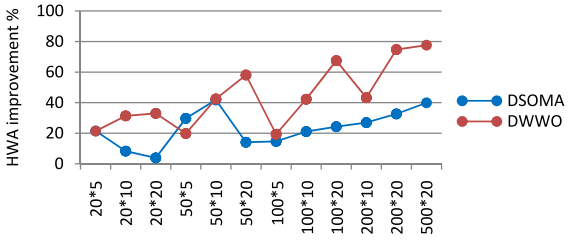


Fig. 12. IP of HWA based on the best scheduling.

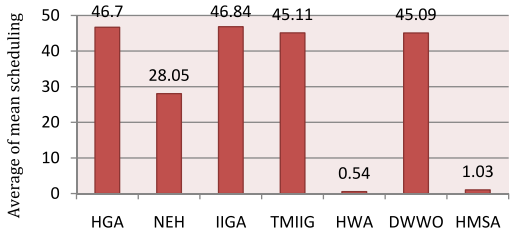


Fig. 13. Comparison on AE of the mean scheduling on Taillard benchmark.

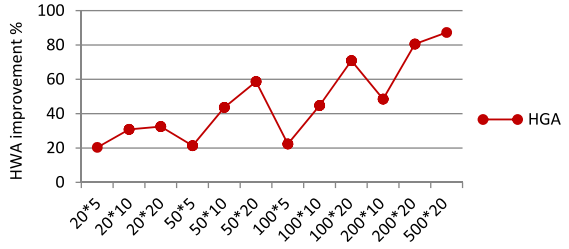


Fig. 14. IP achieved by HWA for HGA on Taillard benchmark.

- Differential evolution algorithm with the iterated improving scheme-based local search and the greedy based local search (L-HDE) [5].
- Opposition-based differential evolution algorithm (ODDE) [21].
- Discrete bat algorithm (DBA) [23].
- Hybrid backtracking search algorithm (HBSA) [25].

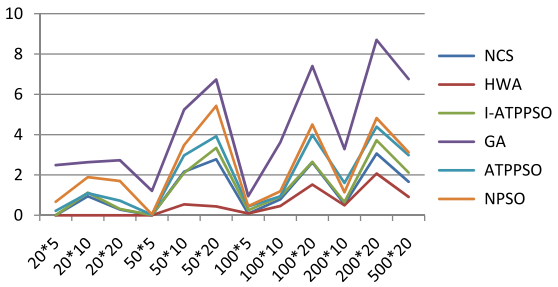


Fig. 15. Comparison of ARPD on Taillard benchmark.

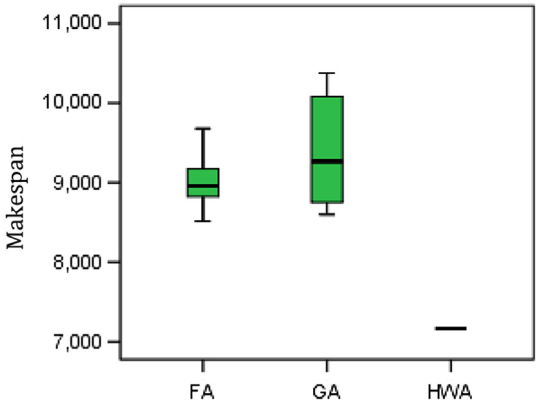


Fig. 16. Boxplot for instance Car2.

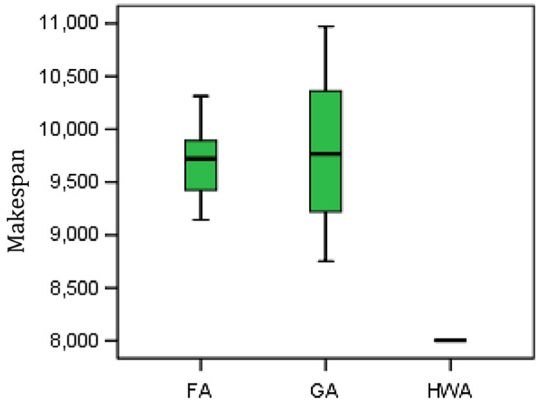


Fig. 17. Boxplot for instance Car4.

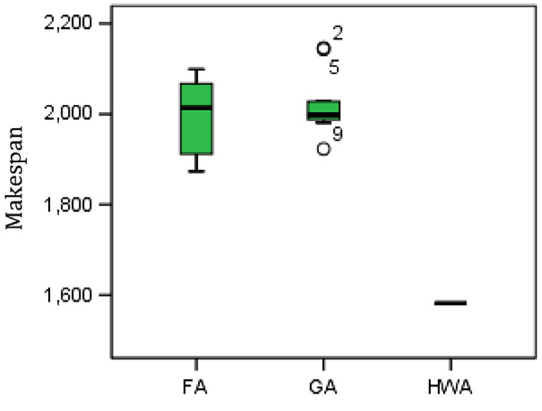


Fig. 18. Boxplot for instance Ta011.

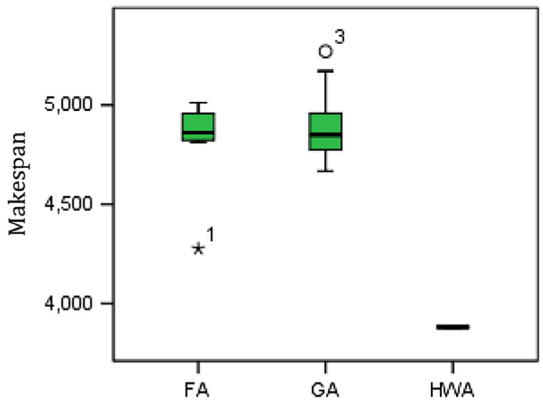


Fig. 19. Boxplot for instance Ta051.

- Particle swarm optimization based on variable neighborhood search (PSOVNS) [39].
- Particle swarm optimization based memetic algorithm (PSOMA) [40].
- Hybrid teaching–learning-based optimization algorithm (HTLBO) [41].
- Quantum differential evolutionary algorithm (QDEA) [42].
- Hybrid genetic algorithm (HGA) [43].
- Hybrid bat algorithm (HBA) [44].
- Hybrid cuckoo search (HCS) [45].

Three performance measures are used to quantitatively compare the algorithms: Best Relative Error (BRE), Average Relative Error (ARE), and Worst Relative Error (WRE) of the algorithm. They can be computed as follows:

$$\text{BRE} = \frac{z^* - z_{\text{best}}}{z^*} \tag{17}$$

$$ARE = \frac{z^* - z_{avg}}{z^*} \tag{18}$$

$$WRE = \frac{z^* - z_{worst}}{z^*} \tag{19}$$

where z_{best} is the best value obtained for each instance through all runs. z^* is the bestknown solution. z_{avg} is the average of the final solutions obtained through runs. z_{worst} is the worst value obtained through runs. The results are recorded in Table 7.

Figs. 7–9 show a comparison based on calculating the average of BRE, ARE, and WRE for the benchmarks of Car and Rec. HWA is compared with eleven algorithms. It can be seen that HWA provides the best results. In the average of BRE, it has the lowest value with 0.13, but DBA has the largest value of 1.124. Also, in the average of ARE, it is ranked first with value 0.2. DBA comes in the last rank with value of 2.154. For the average of WRE, HWA achieves 0.25 and ODDE comes in the second rank with value 0.48. It is evident in the result that HWA is an effective and reliable algorithm when solving PFSSP.

Table 8 presents a comparison on Heller benchmark which contains two instances (Hel1 and Hel2). HWA is compared with NEH and a stochastic greedy heuristic algorithm (SG) [46]. The upper bound is taken from [46]. It can be seen in the results that HWA achieves new upper bounds for the instances of Hel1 and Hel2 with value 515 and 135 respectively.

5.2. Comparison on benchmark Taillard

We test the effectiveness and stability of HWA on Taillard benchmark. The upper bounds of the instances are taken from [14]. This benchmark contains 120 instances of different sizes as follows: $20*5$, $20*10$, $20*20$, $50*5$, $50*10$, $50*20$, $100*5$, $100*10$, $100*20$, $200*10$, $200*20$, $500*20$. Three performance measures are used to evaluate the algorithms: best, mean, and worst. In Table A.1 in the Appendix, we conduct a comparison based on the best makespan between HWA and other six algorithms as follows:

- Hybrid genetic algorithm (HGA) [9].
- Tabu-mechanism improved iterated greedy (TMIIG) [26].
- Discrete self-organizing migrating algorithm (DSOMA) [47].
- Iterated greedy with referenced insertion scheme (IG_RIS) [48].
- Discrete water wave optimization algorithm (DWWO) [49].
- Improved iterated greedy algorithm (IIGA) [50].

The results obtained show the superiority of HWA. Also, we can see that HWA obtains the lowest known upper bounds in 50 out of 120 instances. The bold values indicate the best results. The values of upper bound are made bold if HWA can obtain the upper bounds values. HWA outperforms all the other algorithms in all instances. Fig. 10 investigates the average error (AE) of the best scheduling on Taillard benchmark which can be defined as:

$$AE = \frac{1}{k} * \sum_{i=1}^{k} \frac{z_{opt} - z_i}{z_{opt}} * 100 \tag{20}$$

where z_{opt} is the lowest known upper bound value for each instance. The parameter z_i is the makespan of the best, mean, and worst scheduling. The parameter K is the number of instances. HWA has the lowest average error with value 0.54 in the best scheduling. HWA outperforms all the other algorithms which prove the robustness and stability of our algorithm. Figs. 11 and 12 provide a comparison based on a performance measure called improvement percentage (IP) of HWA which can be defined as follows:

$$IP = \frac{c_{HWA} - c_x}{c_{HWA}} \tag{21}$$

where IP is the percentage improvement of HWA. The parameter c_{HWA} is the minimum makespan obtained by HWA in the best, mean, and worst scheduling. The parameter c_x is the minimum makespan obtained by any algorithm x in the best, mean, worst scheduling. IP is useful to determine the percentage of improvement that HWA achieves for each algorithm.

In Table A.2, a comparison is conducted based on the mean value between HWA and other six algorithms as follows:

- Nawaz–Enscore–Ham (NEH) [2].
- Hybrid monkey search algorithm (HMSA) [4].
- Hybrid genetic algorithm (HGA) [9].
- Tabu-mechanism improved iterated greedy (TMIIG) [26].
- Discrete water wave optimization algorithm (DWWO) [49].
- Improved iterated greedy algorithm (IIGA) [50].

HWA obtains the minimum known makespan in 43 out of 120 instances. HWA outperforms all algorithms in most of the instances. Fig. 13 shows the Average Error (AE) of the mean scheduling on Taillard benchmark. HWA has the lowest average error with value 0.54 in the best scheduling then HMSA with value 1.03.

Table A.1
Comparison on Taillard benchmark based on the best scheduling.

Instance	$n * m$	Upper bound	HWA	HGA	DSOMA	IIGA	TMIIG	DWWO	IG_RIS
Ta001	20 * 5	**1 278**	**1 278**	1 449	1 374	1 486	1 486	1 486	–
Ta002		**1 359**	**1 359**	1 460	1 408	1 528	1 528	1 528	–
Ta003		**1 081**	**1 081**	1 386	1 280	1 460	1 460	1 460	–
Ta004		**1 293**	**1 293**	1 521	1 448	1 588	1 588	1 588	–
Ta005		**1 235**	**1 235**	1 403	1 341	1 449	1 449	1 449	–
Ta006		**1 195**	**1 195**	1 430	1 363	1 481	1 481	1 481	–
Ta007		**1 239**	**1 239**	1 461	1 381	1 483	1 483	1 483	–
Ta008		**1 206**	**1 206**	1 433	1 379	1 482	1 482	1 482	–
Ta009		**1 230**	**1 230**	1 398	1 373	1 469	1 469	1 469	–
Ta010		**1 108**	**1 108**	1 324	1 283	1 377	1 377	1 377	–
Ta011	20 * 10	**1 582**	**1 582**	1 955	1 698	2 044	2 044	2 044	–
Ta012		**1 659**	**1 659**	2 123	1 833	2 166	2 166	2 166	–
Ta013		**1 496**	**1 496**	1 912	1 676	1 940	1 940	1 940	–
Ta014		**1 377**	**1 377**	1 782	1 546	1 811	1 811	1 811	–
Ta015		**1 419**	**1 419**	1 933	1 617	1 933	1 933	1 933	–
Ta016		**1 397**	**1 397**	1 827	1 590	1 892	1 892	1 892	–
Ta017		**1 484**	**1 484**	1 944	1 622	1 963	1 963	1 963	–
Ta018		**1 538**	**1 538**	2 006	1 731	2 057	2 057	2 057	–
Ta019		**1 593**	**1 593**	1 908	1 747	1 973	1 973	1 973	–
Ta020		**1 591**	**1 591**	2 001	1 782	2 051	2 051	2 051	–
Ta021	20 * 20	**2 297**	**2 297**	2 912	2 436	2 973	2 973	2 973	–
Ta022		**2 099**	**2 099**	2 780	2 234	2 852	2 852	2 852	–
Ta023		**2 326**	**2 326**	2 922	2 479	3 013	3 013	3 013	–
Ta024		**2 223**	**2 223**	2 967	2 348	3 001	3 001	3 001	–
Ta025		**2 291**	**2 291**	2 953	2 435	3 003	3 003	3 003	–
Ta026		**2 226**	**2 226**	2 908	2 383	2 998	2 998	2 998	–
Ta027		**2 273**	**2 273**	2 970	2 390	3 052	3 052	3 052	–
Ta028		**2 200**	**2 200**	2 763	2 328	2 839	2 839	2 839	–
Ta029		**2 237**	**2 237**	2 972	2 363	3 009	3 009	3 009	–
Ta030		**2 178**	**2 178**	2 919	2 323	2 979	2 979	2 979	–
Ta031	50 * 5	**2 724**	**2 724**	3 127	3 033	3 161	3 161	3 170	2 974
Ta032		**2 834**	**2 834**	3 438	3 045	3 432	3 440	3 441	3 171
Ta033		**2 621**	**2 621**	3 182	3 036	3 211	3 213	3 218	2 988
Ta034		**2 751**	**2 751**	3 289	3 011	3 339	3 343	3 349	3 113
Ta035		**2 863**	**2 863**	3 315	3 128	3 356	3 361	3 376	3 140
Ta036		**2 829**	**2 829**	3 324	3 166	3 347	3 346	3 352	3 158
Ta037		**2 725**	**2 725**	3 183	3 021	3 231	3 234	3 243	3 005
Ta038		**2 683**	**2 683**	3 243	3 063	3 235	3 241	3 239	3 040
Ta039		**2 552**	**2 552**	3 059	2 908	3 072	3 075	3 078	2 889
Ta040		**2 782**	**2 782**	3 301	3 120	3 317	3 322	3 330	3 094
Ta041	50 * 10	2 991	**3 021**	4 251	3 638	4 274	4 274	4 274	3 605
Ta042		2 867	**2 891**	4 139	3 511	4 177	4 179	4 180	3 470
Ta043		2 839	**2 869**	4 083	3 492	4 099	4 099	4 099	3 465
Ta044		3 063	**3 063**	4 480	3 672	4 399	4 399	4 407	3 649
Ta045		2 976	**3 001**	4 316	3 633	4 322	4 324	4 324	3 614
Ta046		3 006	**3 006**	4 282	3 621	4 289	4 290	4 294	3 574
Ta047		3 093	**3 126**	4 376	3 704	4 420	4 420	4 420	3 667
Ta048		3 037	**3 046**	4 304	3 572	4 318	4 321	4 323	3 549
Ta049		2 897	**2 897**	4 162	3 541	4 155	4 158	4 155	3 510
Ta050		3 065	**3 078**	4 232	3 624	4 283	4 286	4 286	3 603
Ta051	50 * 20	3 850	**3 876**	6 138	4 511	6 129	6 129	6 129	4 484
Ta052		3 704	**3 715**	5 721	4 288	5 725	5 725	5 725	4 262
Ta053		3 640	**3 653**	5 847	4 289	5 862	5 873	5 862	4 261
Ta054		3 720	**3 755**	5 781	4 378	5 788	5 789	5 789	4 338
Ta055		3 610	**3 649**	5 891	4 271	5 886	5 886	5 886	4 249
Ta056		3 681	**3 703**	5 875	4 202	5 863	5 874	5 871	4 271
Ta057		3 704	**3 723**	5 937	4 315	5 962	5 968	5 969	4 291
Ta058		3 691	**3 704**	5 919	4 326	5 926	5 940	5 926	4 298
Ta059		3 743	**3 763**	5 839	4 329	5 876	5 876	5 876	4 304
Ta060		3 756	**3 767**	5 935	4 422	5 958	5 959	5 958	4 398
Ta061	100 * 5	**5 493**	**5 493**	6 492	6 151	6 397	6 397	6 433	6 038
Ta062		**5 268**	**5 268**	6 353	6 064	6 234	6 246	6 268	5 933
Ta063		**5 175**	**5 175**	6 148	6 003	6 121	6 133	6 162	5 837
Ta064		5 014	**5 018**	6 080	5 786	6 026	6 028	6 055	5 661
Ta065		**5 250**	**5 250**	6 254	6 021	6 200	6 206	6 221	5 873
Ta066		**5 135**	**5 135**	6 177	5 869	6 074	6 088	6 121	5 732
Ta067		**5 246**	**5 246**	6 257	6 004	6 247	6 254	6 311	5 890
Ta068		**5 094**	**5 094**	6 225	5 924	6 130	6 150	6 197	5 785

(*continued on next page*)

Table A.1 (*continued*)

Instance	$n * m$	Upper bound	HWA	HGA	DSOMA	IIGA	TMIIG	DWWO	IG_RIS
Ta069		**5 448**	**5 448**	6 443	6 154	6 370	6 391	6 418	6 029
Ta070		5 322	**5 324**	6 441	6 186	6 381	6 396	6 404	6 049
Ta071	100 * 10	5 770	**5 776**	8 115	7 042	8 077	8 080	8 093	6 896
Ta072		5 349	**5 362**	7 986	6 813	7 880	7 888	7 891	6 622
Ta073		5 676	**5 691**	8 057	6 943	8 028	8 042	8 047	6 766
Ta074		5 781	**5 825**	8 327	7 198	8 348	8 350	8 364	7 037
Ta075		5 467	**5 491**	7 991	6 815	7 958	7 967	7 966	6 690
Ta076		5 303	**5 308**	7 823	6 685	7 801	7 808	7 791	6 517
Ta077		5 595	**5 608**	7 915	6 827	7 866	7 880	7 881	6 684
Ta078		5 617	**5 630**	7 939	6 874	7 913	7 912	7 924	6 729
Ta079		5 871	**5 891**	8 226	6 092	8 161	8 164	8 152	6 908
Ta080		5 845	**5 848**	8 186	6 990	8 114	8 130	8 126	6 832
Ta081	100 * 20	6 202	**6 280**	10 745	7 854	10 700	10 722	10 727	7 683
Ta082		6 183	**6 278**	10 655	7 910	10 594	10 611	10 604	7 739
Ta083		6 271	**6 368**	10 672	7 825	10 611	10 629	10 624	7 697
Ta084		6 269	**6 350**	10 630	7 902	10 607	10 615	10 615	7 730
Ta085		6 314	**6 377**	10 548	7 901	10 539	10 563	10 551	7 694
Ta086		6 364	**6 430**	10 700	7 921	10 690	10 684	10 680	7 745
Ta087		6 268	**6 354**	10 827	8 051	10 825	10 832	10 824	7 848
Ta088		6 401	**6 515**	10 863	8 025	10 839	10 846	10 839	7 879
Ta089		6 275	**6 396**	10 751	7 969	10 723	10 763	10 745	7 771
Ta090		6 434	**6 527**	10 794	8 036	10 798	10 797	10 787	7 818
Ta091	200 * 10	10 862	**10 885**	15 739	13 507	15 319	15 377	15 418	13 100
Ta092		10 480	**10 512**	15 534	13 458	15 085	15 167	15 252	13 048
Ta093		10 922	**10 965**	15 755	13 521	15 376	15 416	15 412	13 135
Ta094		**10 889**	**10 889**	15 842	13 686	15 200	15 250	15 304	13 112
Ta095		**10 524**	**10 524**	15 692	13 547	15 209	15 268	15 277	13 097
Ta096		10 326	**10 375**	15 622	13 247	15 109	15 163	15 222	12 869
Ta097		10 854	**10 868**	15 877	13 910	15 395	15 441	15 459	13 351
Ta098		10 730	**10 751**	15 733	13 830	15 237	15 295	15 307	13 225
Ta099		10 438	**10 465**	15 573	13 410	15 100	15 155	15 186	13 036
Ta100		10 675	**10 727**	15 803	13 744	15 340	15 382	15 378	13 119
Ta101	200 * 20	11 195	**11 335**	20 148	15 027	19 681	19 723	19 724	14 484
Ta102		11 203	**11 517**	20 539	15 211	20 096	20 127	20 091	14 690
Ta103		11 281	**11 481**	20 511	15 247	19 913	19 973	19 989	14 776
Ta104		11 275	**11 405**	20 461	15 174	19 928	19 997	19 940	14 694
Ta105		11 259	**11 374**	20 339	15 047	19 843	19 900	19 875	14 547
Ta106		11 176	**11 335**	20 501	15 212	19 942	20 003	19 980	14 734
Ta107		11 360	**11 438**	20 680	15 168	20 112	20 145	20 119	14 744
Ta108		11 334	**11 530**	20 614	15 247	20 056	20 053	20 053	14 763
Ta109		11 192	**11 439**	20 300	15 136	19 918	19 998	19 932	14 643
Ta110		11 288	**11 499**	20 437	15 243	19 935	19 932	19 916	14 703
Ta111	500 * 20	26 059	**26 388**	49 095	37 064	46 689	47 046	46 871	35 372
Ta112		26 520	**26 714**	49 461	37 419	47 275	47 630	47 294	35 743
Ta113		26 371	**26 648**	48 777	37 059	46 544	46 977	46 846	35 452
Ta114		26 456	**25 656**	49 283	37 014	46 899	47 328	47 185	35 687
Ta115		26 334	**26 579**	48 950	36 894	46 741	47 238	47 037	35 417
Ta116		26 477	**26 666**	49 533	37 372	46 941	47 553	47 166	35 747
Ta117		26 389	**26 594**	48 943	36 998	46 509	46 944	46 794	35 395
Ta118		26 560	**26 711**	49 277	36 944	46 873	47 346	47 127	35 568
Ta119		26 005	**26 228**	49 207	36 862	46 743	47 205	47 025	35 304
Ta120		26 457	**26 695**	49 092	37 098	46 847	47 374	47 105	35 643

Bold values indicate the best results.

In Table A.3, the worst scheduling obtained by HWA is recorded. We compare the performance of HWA with HGA. HWA outperforms HGA in all instances. Fig. 14 presents a comparison between HWA and HGA based on the HWA improvement percentage.

Table 9 presents a comparison between HWA and NCS [51] on Taillard benchmark. Two performance measures are used: mean and average relative percentage deviation (*ARPD*). *ARPD* can be calculated as follows:

$$ARPD = \frac{c_{opt} - c_x}{c_{opt}} \tag{22}$$

where c_{opt} is the known minimum make span. The parameter c_x is the minimum make span obtained by algorithm x The average of ARPD is computed. HWA has a lower average of ARPD with value 0.54. Fig. 15 introduces a comparison based on ARPD between HWA and the following algorithms:

- New cuckoo search (NCS) [51].
- An alternate two phases particle swarm optimization algorithm (ATPPSO) [52].
- Improved alternate two phases particle swarm optimization algorithm (I-ATPPSO) [53].
- Novel particle swarm optimization algorithm (NPSO) [54].
- Genetic algorithm (GA) [55].

It can be seen from the results that HWA provides the minimum ARPD for all the instances. Figs. 16–19 present the boxplots for three algorithms when solving the instances of Car2 and Car4, respectively. HWA is compared with firefly algorithm (FA) [22] and

Table A.2
Comparison on Taillard benchmark based on the mean scheduling.

Instance	$n * m$	Upper bound	HGA	HWA	TMIIG	DWWO	NEH	HMSA	IIGA
Ta001	20 ∗ 5	1 278	1 472.7	**1 278**	1 486	1 486	1 299	1 282	1 486
Ta002		1 359	1 477.6	**1 359**	1 528	1 528	1 365	1 362	1 528
Ta003		1 081	1 406.8	**1 081**	1 460	1 460	1 132	1 094	1 460
Ta004		1 293	1 546.4	**1 293**	1 588	1 588	1 329	1 304	1 588
Ta005		1 235	1 426.6	**1 235**	1 449	1 449	1 305	1 248	1 449
Ta006		1 195	1 446.6	**1 195**	1 481	1 481	1 251	1 204	1 481
Ta007		1 239	1 479.4	**1 239**	1 483	1 483	1 251	1 247	1 483
Ta008		1 206	1 459.2	**1 206**	1 482	1 482	1 215	1 208	1 482
Ta009		1 230	1 409.2	**1 230**	1 469	1 469	1 284	1 254	1 469
Ta010		1 108	1 345.5	**1 108**	1 377	1 377	1 127	1 124	1 377
Ta011	20 ∗ 10	1 582	1 972.6	**1 582**	2 044	2 044	1 681	1 616	2 044
Ta012		1 659	2 154.6	**1 659**	2 166	2 166	1 766	1 712	2 166
Ta013		1 496	1 931.1	**1 496**	1 940.4	1 940	1 562	1 510	1 940
Ta014		1 377	1 794.3	**1 377**	1 811	1 811	1 416	1 384	1 811
Ta015		1 419	1 934.4	**1 419**	1 933	1 933	1 502	1 456	1 933
Ta016		1 397	1 850.0	**1 397**	1 892	1 892	1 456	1 420	1 892
Ta017		1 484	1 951.5	**1 484**	1 963	1 963	1 531	1 496	1 963
Ta018		1 538	2 038.5	**1 538**	2 057	2 057	1 626	1 568	2 058.6
Ta019		1 593	1 953.9	**1 593**	1 973	1 973	1 639	1 604	1 973
Ta020		1 591	2 019.3	**1 591**	2 051	2 051	1 656	1 615	2 051
Ta021	20 ∗ 20	2 297	2 938.5	**2 297**	2 973	2 973	2 443	2 324	2 973
Ta022		2 099	2 814.2	**2 099**	2 852	2 852	2 134	2 112	2 852
Ta023		2 326	2 962.4	**2 326**	3 019.4	3 014.3	2 414	2 348	3 019.4
Ta024		2 223	2 982.5	**2 223**	3 001	3 001	2 257	2 242	3 001
Ta025		2 291	2 995.1	**2 291**	3 003	3 003	2 370	2 320	3 003
Ta026		2 226	2 932.8	**2 226**	2 998	2 998	2 349	2 249	2 998
Ta027		2 273	3 004.8	**2 273**	3 052	3 052	2 383	2 290	3 052
Ta028		2 200	2 789.5	**2 200**	2 839	2 839	2 249	2 224	2 839
Ta029		2 237	3 005.3	**2 237**	3 009	3 009	2 306	2 246	3 009
Ta030		2 178	2 956.3	**2 178**	2 979	2 979	2 257	2 192	2 979
Ta031	50 ∗ 5	2 724	3 198.3	**2 725.6**	3 162.4	3 171.8	2 729	2 728	3 222
Ta032		2 834	3 453.1	**2 834**	3 441	3 444.5	2 882	2 846	3 474.4
Ta033		2 621	3 242.5	**2 621**	3 216	3 231.8	2 650	2 642	3 262.8
Ta034		2 751	3 349.8	**2 751**	3 346.6	3 350.8	2 782	2 762	3 374.8
Ta035		2 863	3 369.1	**2 863**	3 364.6	3 376.7	2 868	2 866	3 406
Ta036		2 829	3 356.4	**2 829**	3 347.6	3 356.2	2 835	2 832	3 382.2
Ta037		2 725	3 246.4	**2 725**	3 235.8	3 245.3	2 806	2 748	3 267.6
Ta038		2 683	3 264.9	**2 686**	3 242.4	3 240.3	2 700	2 690	3 275.2
Ta039		2 552	3 088.2	**2 552**	3 078.8	3 086.8	2 606	2 564	3 123.2
Ta040		2 782	3 341.5	**2 782**	3 327.2	3 336.5	2 801	2 796	3 377.2
Ta041	50 ∗ 10	2 991	4 289.1	**3 021.28**	4 276.8	4 281.5	3 175	3 034	4 311.2
Ta042		2 867	4 193.5	**2 901.6**	4 185	4 184.5	3 073	2 994	4 201
Ta043		2 839	4 108.6	**2 869.25**	4 107.6	4 105.5	2 994	2 924	4 124
Ta044		3 063	4 517.4	**3 063.7**	4 405	4 405.7	3 218	3 082	4 439.2
Ta045		2 976	4 333.2	**3 012.33**	4 330.4	4 324.7	3 186	3 028	4 347
Ta046		3 006	4 301.6	**3 016.83**	4 297.2	4 295.2	3 148	3 064	4 330
Ta047		3 093	4 412.6	**3 127.625**	4 429.6	4 420	3 277	3 132	4 441.4
Ta048		3 037	4 331.9	**3 046.875**	4 327	4 323.3	3 170	3 072	4 357.6
Ta049		2 897	4 173.0	**2 901.5**	4 164.2	4 161.7	3 025	2 924	4 194.8
Ta050		3 065	4 279.0	**3 081.7**	4 286.2	4 286.2	3 267	3 134	4 301
Ta051	50 ∗ 20	3 850	6 149.7	**3 881.44**	6 139.6	6 138.5	4 006	3 896	6 154.6
Ta052		3 704	5 751.5	**3 723.05**	5 741	5 733.5	3 958	3 746	5 762.2
Ta053		3 640	5 884.4	**3 660.17**	5 882.4	5 865.5	3 866	3 694	5 907
Ta054		3 720	5 804.7	**3 762.4**	5 791.4	5 790.7	3 953	3 814	5 802.6
Ta055		3 610	5 909.7	**3 654.5**	5 899.4	5 893.5	3 872	3 686	5 930.6
Ta056		3 681	5 890.6	**3 708.75**	5 883.4	5 874.3	3 861	3 722	5 912.6
Ta057		3 704	5 974.2	**3 726.41**	5 974	5 974	3 927	3 766	6 012
Ta058		3 691	5 951.0	**3 707.975**	5 945.4	5 930.5	3 914	3 768	5 970.2
Ta059		3 743	5 873.5	**3 768.82**	5 883.2	5 876	3 970	3 812	5 900.6
Ta060		3 756	5 963.5	**3 772.38**	5 959	5 958.8	4 036	3 826	5 982
Ta061	100 ∗ 5	5 493	6 557.2	**5 493**	6 413.4	6 438.3	5 514	5 502	6 586.4
Ta062		5 268	6 409.4	**5 271.5**	6 252.2	6 285.5	5 284	5 272	6 428.2
Ta063		5 175	6 260.4	**5 175**	6 135.8	6 164.2	5 222	5 192	6 292.8
Ta064		5 014	6 159.0	**5 018.75**	6 031.4	6 050.7	5 023	5 020	6 184.8
Ta065		5 250	6 325.6	**5 251.4**	6 217.8	6 223.3	5 261	5 254	6 358.4
Ta066		5 135	6 225.7	**5 135**	6 096	6 122.7	5 154	5 144	6 269.2
Ta067		5 246	6 409.0	**5 255.57**	6 263.4	6 308.8	5 282	5 264	6 442.2
Ta068		5 094	6 308.6	**5 099.86**	6 156.6	6 210.8	5 140	5 114	6 338.6

(*continued on next page*)

Table A.2 (*continued*)

Instance	$n * m$	Upper bound	HGA	HWA	TMIIG	DWWO	NEH	HMSA	IIGA
Ta069		5 448	6 516.3	**5 453.125**	6 395	6 422	5 489	5 466	6 557.4
Ta070		5 322	6 542.5	**5 328**	6 401.6	6 427.2	5 336	5 332	6 561
Ta071	100 ∗ 10	5 770	8 173.5	**5 782.75**	8 094.2	8 100.8	5 897	5 792	8 224.6
Ta072		5 349	8 048.1	**5 367.83**	7 909	7 938.3	5 466	5 368	8 061.4
Ta073		5 676	8 142.2	**5 693.14**	8 054.4	8 051	5 747	5 694	8 152
Ta074		5 781	8 437.5	**5 825.8**	8 362.4	8 378.2	5 924	5 826	8 484.2
Ta075		5 467	8 046.4	**5 500.72**	7 981.2	7 980	7 991	5 514	8 087.6
Ta076		5 303	7 883.7	**5 316.25**	7 821.6	7 809	7 823	5 324	7 930.4
Ta077		5 595	8 007.0	**5 621.83**	7 887.2	7 889	7 915	5 628	7 983.8
Ta078		5 617	8 049.2	**5 647.92**	7 924.8	7 931.7	7 939	5 664	8 051.4
Ta079		5 871	8 290.4	**5 896.33**	8 172	8 183.8	8 226	5 912	8 312.6
Ta080		5 845	8 255.3	**5 871.91**	8 148.8	8 141	8 186	5 898	8 263.6
Ta081	100 ∗ 20	6 202	10 826.6	**6 301.08**	10 782.4	10 744	10 745	6 306	10 887
Ta082		6 183	10 762.5	**6 290.33**	10 623	10 608.7	10 655	6 278	10 755.6
Ta083		6 271	10 740.4	**6 375.5**	10 650	10 637	10 672	6 404	10 774.6
Ta084		6 269	10 679.7	6 354	10 647.2	10 629.8	10 630	**6 184**	10 765.8
Ta085		6 314	10 658.2	**6 396.08**	10 579.8	10 568.3	10 548	6 422	10 690.6
Ta086		6 364	10 753.0	**6 444.58**	10 696.2	10 690	10 700	6 526	10 819.4
Ta087		6 268	10 913.6	**6 369.67**	10 849.2	10 843.2	10 827	6 412	10 997.2
Ta088		6 401	10 905.2	6 519.09	10 862	10 850.2	10 863	**6 516**	11 011.6
Ta089		6 275	10 859.0	**6 399.71**	10 780.4	10 758	10 751	6 424	10 875
Ta090		6 434	10 928.8	6 531.54	10 817.2	10 808.5	10 794	**6 496**	10 944.6
Ta091	200 ∗ 10	10 862	15 843.5	**10 885**	15 397.4	15 449.3	15 739	10 932	15 751.4
Ta092		10 480	15 645.3	**10 527.78**	15 203	15 281.3	15 534	10 624	15 572.2
Ta093		10 922	15 882.4	**10 965**	15 433.2	15 454.5	15 755	11 006	15 762.4
Ta094		10 889	15 927.0	**10 892.21**	15 279.8	15 304.7	15 842	11 024	15 625
Ta095		10 524	15 763.8	**10 536.15**	15 280.8	15 295.8	15 692	**10 474**	15 626.2
Ta096		10 326	15 669.9	10 375	15 189.6	15 235.5	15 622	**10 369**	15 557
Ta097		10 854	15 962.3	**10 877**	15 476.6	15 508.3	15 877	10 907	15 842
Ta098		10 730	15 833.2	**10 767.86**	15 319.2	15 340	15 733	10 794	15 685.6
Ta099		10 438	15 626.2	**10 466.14**	15 185.2	15 240.2	15 573	10 482	15 525.8
Ta100		10 675	15 869.1	**10 727**	15 410.8	15 412.2	15 803	10 720	15 754.4
Ta101	200 ∗ 20	11 195	20 331.3	**11 341**	19 770.8	19 736.7	20 148	11 342	20 127.8
Ta102		11 203	20 763.5	**11 542.2**	20 148.8	20 118.8	20 539	11 584	20 536.6
Ta103		11 281	20 583.8	**11 495.27**	19 995.2	19 999	20 511	11 568	20 355.6
Ta104		11 275	20 594.6	**11 421.58**	20 032	19 967.2	20 461	11 480	20 349.6
Ta105		11 259	20 544.4	**11 390.55**	19 936	19 923.8	20 339	11 452	20 331.4
Ta106		11 176	20 661.9	**11 352.6**	20 050.6	20 011.3	20 501	11 378	20 407
Ta107		11 360	20 804.3	**11 471.44**	20 167.4	20 147.5	20 680	11 624	20 539.2
Ta108		11 334	20 665.7	**11 539.9**	20 095.6	20 084	20 614	11 613	20 449.8
Ta109		11 192	20 574.9	**11 452.27**	20 022.2	19 975.5	20 300	11 524	20 358.2
Ta110		11 288	20 587.6	**11 521.46**	19 968.6	19 946.3	20 437	11 574	20 284
Ta111	500 ∗ 20	26 059	49 289.4	**26 398.7**	47 147.2	46 965.5	49 095	26 552	48 253.4
Ta112		26 520	49 948.2	**26 720.75**	47 692.2	47 429.8	49 461	26 823	48 699
Ta113		26 371	49 139.0	**26 653**	47 049.6	46 878.7	48 777	26 627	48 116.4
Ta114		26 456	49 657.8	**26 673.43**	47 375.8	47 232.8	49 283	26 674	48 513.2
Ta115		26 334	49 423.1	**26 589.78**	47 280	47 099.2	48 950	26 622	48 345.8
Ta116		26 477	49 848.4	**26 678**	47 591.2	47 331.5	49 533	26 722	48 701.6
Ta117		26 389	49 366.5	26 597	47087.2	46 866.3	48 943	**26 572**	48 186
Ta118		26 560	49 675.8	26 819	47421.4	47 248.7	49 277	**26 720**	48 468.2
Ta119		26 005	49 429.3	26 238.13	47 248.6	47 083.5	49 207	**26 210**	48 297.2
Ta120		26 457	49 545.6	**26 697.22**	47 402.4	47 183.2	49 092	26 720	48 483.4

Bold values indicate the best results.

the genetic algorithm (GA) [22]. It is evident that HWA finds the optimal and minimum makespan among the algorithms.

Taken together, the results show that the new operators integrated into the WA and improvement were beneficial. They allowed this algorithm to handle high-dimensional case studies. This is mostly due to the integrated mutation operator that abruptly changes the solutions during the optimization process. The number of local solution is normally increased proportional to the number of iteration, so the mechanism proposed allowed the WOA algorithm to show a higher exploratory behavior and consequently better avoid local solutions. Another mechanism to increase the diversity of solutions and encourage a higher local optima avoidance was insert-reversed block operation. Increasing the randomness might result in degrading the accuracy of the solutions that we obtained at the end of the optimization process. The integrated local search alleviates this drawback by exploiting around the best solutions obtained so far.

6. Conclusions and future work

This paper presents a hybrid whale optimization algorithm incorporated with local search strategy, swap mutation, and insert-reversed block operations. Also, a well-known effective heuristic algorithm called NEH is combined with HWA. LRV is used to convert the continuous values into a job permutation. The local search improves the performance of HWA as it improves the quality of solutions. One of the HWA's limitations is the use of local search strategy. Misuse of the local search leads to increase in time. Excessive use of local search usually leads to trapping into local

Table A.3
Comparison on Taillard benchmark based on the worst scheduling.

Instance	$n * m$	Upper bound	HGA	HWA	Instance	$n * m$	Upper bound	HGA	HWA
Ta001	20 ∗ 5	1278	1485	**1278**	Ta061	100 ∗ 5	5 493	6 594	**5 493**
Ta002		1359	1505	**1359**	Ta062		5 268	6 469	**5 275**
Ta003		1081	1431	**1081**	Ta063		5 175	6 320	**5 175**
Ta004		1293	1573	**1293**	Ta064		5 014	6 198	**5 021**
Ta005		1235	1445	**1235**	Ta065		5 250	6 397	**5 253**
Ta006		1195	1471	**1195**	Ta066		5 135	6 296	**5 135**
Ta007		1239	1496	**1239**	Ta067		5 246	6 460	**5 262**
Ta008		1206	1475	**1206**	Ta068		5 094	6 359	**5 106**
Ta009		1230	1429	**1230**	Ta069		5 448	6 561	**5 465**
Ta010		1108	1368	**1108**	Ta070		5 322	6 592	**5 332**
Ta011	20 ∗ 10	1582	1998	**1582**	Ta071	100 ∗ 10	5 770	8 230	**5 794**
Ta012		1659	2166	**1659**	Ta072		5 349	8 101	**5 377**
Ta013		1496	1942	**1496**	Ta073		5 676	8 215	**5 699**
Ta014		1377	1811	**1377**	Ta074		5 781	8 505	**5 827**
Ta015		1419	1947	**1419**	Ta075		5 467	8 096	**5 516**
Ta016		1397	1879	**1397**	Ta076		5 303	7 947	**5 328**
Ta017		1484	1971	**1484**	Ta077		5 595	8 081	**5 633**
Ta018		1538	2066	**1538**	Ta078		5 617	8 105	**5 668**
Ta019		1593	1973	**1593**	Ta079		5 871	8 349	**5 903**
Ta020		1591	2032	**1591**	Ta080		5 845	8 340	**5 903**
Ta021	20 ∗ 20	2297	2972	**2297**	Ta081	100 ∗ 20	6 202	10 932	**6 327**
Ta022		2099	2835	**2099**	Ta082		6 183	10 847	**6 302**
Ta023		2326	2984	**2326**	Ta083		6 271	10 821	**6 389**
Ta024		2223	2994	**2223**	Ta084		6 269	10 797	**6 360**
Ta025		2291	3017	**2291**	Ta085		6 314	10 777	**6 416**
Ta026		2226	2964	**2226**	Ta086		6 364	10 853	**6 453**
Ta027		2273	3028	**2273**	Ta087		6 268	11 031	**6 411**
Ta028		2200	2826	**2200**	Ta088		6 401	10 992	**6 225**
Ta029		2237	3009	**2237**	Ta089		6 275	10 963	**6 404**
Ta030		2178	2979	**2178**	Ta090		6 434	11 067	**6 552**
Ta031	50 ∗ 5	2724	3229	**2728**	Ta091	200 ∗ 10	10 862	15 916	**10 885**
Ta032		2834	3475	**2834**	Ta092		10 480	15 764	**10 559**
Ta033		2621	3277	**2621**	Ta093		10 922	16 026	**10 965**
Ta034		2751	3384	**2751**	Ta094		10 889	16 111	**10 893**
Ta035		2863	3404	**2863**	Ta095		10 524	15 829	**10 537**
Ta036		2829	3377	**2829**	Ta096		10 326	15 731	**10 375**
Ta037		2725	3280	**2725**	Ta097		10 854	16 029	**10 882**
Ta038		2683	3288	**2689**	Ta098		10 730	15 933	**10 798**
Ta039		2552	3121	**2552**	Ta099		10 438	15 759	**10 473**
Ta040		2782	3383	**2782**	Ta100		10 675	15 934	**10 727**
Ta041	50 ∗ 10	2991	4306	**3025**	Ta101	200 ∗ 20	11 195	20 458	**11 360**
Ta042		2867	4235	**2907**	Ta102		11 203	20 889	**11 591**
Ta043		2839	4124	**2870**	Ta103		11 281	20 636	**11 521**
Ta044		3063	4549	**3064**	Ta104		11 275	20 753	**11 457**
Ta045		2976	4367	**3021**	Ta105		11 259	20 601	**11 432**
Ta046		3006	4332	**3021**	Ta106		11 176	20 780	**11 389**
Ta047		3093	4444	**3131**	Ta107		11 360	20 915	**11 520**
Ta048		3037	4347	**3049**	Ta108		11 334	20 814	**11 550**
Ta049		2897	4188	**2903**	Ta109		11 192	20 757	**11 470**
Ta050		3065	4304	**3090**	Ta110		11 288	20 712	**11 556**
Ta051	50 ∗ 20	3850	6172	**3887**	Ta111	500 ∗ 20	26 059	49 580	**26 415**
Ta052		3704	5790	**3728**	Ta112		26 520	50 354	**26 726**
Ta053		3640	5929	**3671**	Ta113		26 371	49 399	**26 661**
Ta054		3720	5827	**3775**	Ta114		26 456	50 004	**26 698**
Ta055		3610	5950	**3660**	Ta115		26 334	49 847	**26 594**
Ta056		3681	5911	**3715**	Ta116		26 477	50 046	**26 724**
Ta057		3704	6001	**3734**	Ta117		26 389	49 591	**26 600**
Ta058		3691	5971	**3723**	Ta118		26 560	49 942	**26 834**
Ta059		3743	5899	**3777**	Ta119		26 005	49 697	**26 250**
Ta060		3756	5979	**3781**	Ta120		26 457	50 002	**26 709**

Bold values indicate the best results.

optima in which we get sub-optimal solutions. We alleviated such drawbacks in HWA through:

(i) The local search strategy is imposed with a small probability.
(ii) The swap operation is used to escape from the local optima.
(iii) Further, the insert-reversed block operation helps to increase the quality of solutions through moving the reversed block to the other different places in the search agent to find better solutions.

HWA was tested on four different benchmarks: Carlier, Reeves, Heller, and Taillard benchmarks. The computational results showed the strength and the robustness of HWA. For future works, it is recommended to apply HWA to other types of scheduling such

as job shop scheduling and open shop scheduling. Solving traveling salesman problem using HWA is recommended.

Appendix. Proof of the main results

See Tables A.1–A.3.

References

[1] S.M. Johnson, Optimal two and three stage production schedules with setup times included, Naval Res. Logist. 1 (1) (1954) 61–68.
[2] M. Nawaz, E.E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, Omega 11 (1) (1983) 91–95.
[3] M.K. Marichelvam, T. Prabaharan, X.S. Yang, Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan, Appl. Soft Comput. 19 (2014) 93–101.
[4] M.K. Marichelvam, Tosun. Ö, M. Geetha, Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time, Appl. Soft Comput. 55 (2017) 82–92.
[5] Y. Liu, M. Yin, W. Gu, An effective differential evolution algorithm for permutation flow shop scheduling problem, Appl. Math. Comput. 248 (2014) 143–159.
[6] B. Rathinam, K. Govindan, B. Neelakandan, S.S. Raghavan, Rule based heuristic approach for minimizing total flow time in permutation flow shop scheduling, Teh. Vjesn. 22 (1) (2015) 25–32.
[7] K. Govindan, R. Balasundaram, N. Baskar, P. Asokan, A hybrid approach for minimizing makespan in permutation flowshop scheduling, J. Syst. Sci. Syst. Eng. 26 (1) (2017) 50–76.
[8] K. Gao, Q. Pan, P.N. Suganthan, J. Li, Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization, Int. J. Adv. Manuf. Technol. 66 (9–12) (2013) 1563–1572.
[9] L.Y. Tseng, Y.T. Lin, A hybrid genetic algorithm for no-wait flowshop scheduling problem, Int. J. Prod. Econ. 128 (1) (2010) 144–152.
[10] H. Ishibuchi, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Trans. Evol. Comput. 7 (2) (2003) 204–223.
[11] Z. Lian, X. Gu, B. Jiao, A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan, Chaos Solitons Fractals 35 (5) (2008) 851–861.
[12] M.F. Tasgetiren, Q.K. Pan, P.N. Suganthan, Y.C. Liang, A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion, in: IEEE Symposium on Computational Intelligence in Scheduling, 2007, SCIS'07, IEEE, 2007, pp. 251–258.
[13] B. Qian, L. Wang, R. Hu, W.L. Wang, D.X. Huang, X. Wang, A hybrid differential evolution method for permutation flow-shop scheduling, Int. J. Adv. Manuf. Technol. 38 (7) (2008) 757–777.
[14] G.I. Zobolas, C.D. Tarantilis, G. Ioannou, Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm, Comput. Oper. Res. 36 (4) (2009) 1249–1267.
[15] H.G. Campbell, R.A. Dudek, M.L. Smith, A heuristic algorithm for the n job, m machine sequencing problem, Manag. Sci. 16 (10) (1970) B–630.
[16] D.S. Palmer, Sequencing jobs through a multi-stage process in the minimum total time — a quick method of obtaining a near optimum, J. Oper. Res. Soc. 16 (1) (1965) 101–107.
[17] J.N. Gupta, A functional heuristic algorithm for the flowshop scheduling problem, J. Oper. Res. Soc. 22 (1) (1971) 39–47.
[18] M. Sayadi, R. Ramezanian, N. Ghaffari-Nasab, A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems, Int. J. Ind. Eng. Comput. 1 (1) (2010) 1–10.
[19] L. Wang, Q.K. Pan, M.F. Tasgetiren, A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem, Comput. Ind. Eng. 61 (1) (2011) 76–83.
[20] F. Ahmadizar, A new ant colony algorithm for makespan minimization in permutation flow shops, Comput. Ind. Eng. 63 (2) (2012) 355–361.
[21] X. Li, M. Yin, An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure, Adv. Eng. Softw. 55 (2013) 10–31.
[22] S. Fong, H.L. Lou, Y. Zhuang, S. Deb, T. Hanne, Solving the permutation flow shop problem with firefly algorithm, in: 2014 2nd International Symposium on Computational and Business Intelligence, ISCBI, IEEE, 2014, pp. 25–29.
[23] Q. Luo, Y. Zhou, J. Xie, M. Ma, L. Li, Discrete bat algorithm for optimal problem of permutation flow shop scheduling, Sci. World J. 2014 (2014).
[24] M. Saravanan, S.J.D. Vijayakumar, R. Srinivasan, S.P. Singarayar, Scheduling to minimize the sum of weighted total flow time and makespan in a permutation flow shop with setup time, Appl. Mech. Mater. 766 (2015) 989.
[25] Q. Lin, L. Gao, X. Li, C. Zhang, A hybrid backtracking search algorithm for permutation flow-shop scheduling problem, Comput. Ind. Eng. 85 (2015) 437–446.
[26] J.Y. Ding, S. Song, J.N. Gupta, R. Zhang, R. Chiong, C. Wu, An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem, Appl. Soft Comput. 30 (2015) 604–613.
[27] J.Y. Ding, S. Song, R. Zhang, S. Zhou, C. Wu, A novel block-shifting simulated annealing algorithm for the no-wait flowshop scheduling problem, in: 2015 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2015, pp. 2768–2774.
[28] R.Sanjeev. Kumar, K.P. Padmanaban, M. Rajkumar, Minimizing makespan and total flow time in permutation flow shop scheduling problems using modified gravitational emulation local search algorithm, Proc. Inst. Mech. Eng. B (2016) 0954405416645775.
[29] K. ben oualid Medani, S. Sayah, A. Bekrar, Whale optimization algorithm based optimal reactive power dispatch: A case study of the Algerian power system, Electr. Power Syst. Res. (2017).
[30] I. Aljarah, H. Faris, S. Mirjalili, Optimizing connection weights in neural networks using the whale optimization algorithm, Soft Comput. 22 (1) (2018) 1–15.
[31] D.B. Prakash, C. Lakshminarayana, Optimal siting of capacitors in radial distribution network using whale optimization algorithm, Alex. Eng. J. (2016).
[32] D. Oliva, M.A. El Aziz, A.E. Hassanien, Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm, Appl. Energy 200 (2017) 141–154.
[33] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[34] A. Kaveh, M.I. Ghazaan, Enhanced whale optimization algorithm for sizing optimization of skeletal structures, Mech. Based Des. Struct. Mach. 45 (3) (2017) 345–362.
[35] J. Carlier, Ordonnancements a contraintes disjonctives, RAIRO-Oper. Res. 12 (4) (1978) 333–350.
[36] C.R. Reeves, A genetic algorithm for flowshop sequencing, Comput. Oper. Res. 22 (1) (1995) 5–13.
[37] J. Heller, Some numerical experiments for an $M \times J$ flow shop and its decision-theoretical aspects, Oper. Res. 8 (2) (1960) 178–184.
[38] E. Taillard, Benchmarks for basic scheduling problems, European J. Oper. Res. 64 (2) (1993) 278–285.
[39] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European J. Oper. Res. 177 (3) (2007) 1930–1947.
[40] B. Liu, L. Wang, Y.H. Jin, An effective pso-based memetic algorithm for flow shop scheduling, IEEE Trans. Syst. Man Cybern. B 37 (1) (2007) 18–27.
[41] Z. Xie, C. Zhang, X. Shao, W. Lin, H. Zhu, An effective hybrid teaching–learning-based optimization algorithm for permutation flow shop scheduling problem, Adv. Eng. Softw. 77 (2014) 35–47.
[42] T. Zheng, M. Yamashiro, Solving flow shop scheduling problems by quantum differential evolutionary algorithm, Int. J. Adv. Manuf. Technol. 49 (5) (2010) 643–662.
[43] L.Y. Tseng, Y.T. Lin, A hybrid genetic algorithm for no-wait flowshop scheduling problem, Int. J. Prod. Econ. 128 (1) (2010) 144–152.
[44] Ö. Tosun, M.K. Marichelvam, Hybrid bat algorithm for flow shop scheduling problems, Int. J. Math. Oper. Res. 9 (1) (2016) 125–138.
[45] X. Li, M. Yin, A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem, Int. J. Prod. Res. 51 (16) (2013) 4732–4754.
[46] M. Ancău, On solving flowshop scheduling problems, Proc. Rom. Acad. Ser. A 13 (1) (2012) 71–79.
[47] D. Davendra, M. Bialic-Davendra, Scheduling flow shops with blocking using a discrete self-organising migrating algorithm, Int. J. Prod. Res. 51 (8) (2013) 2200–2218.
[48] M.F. Tasgetiren, D. Kizilay, Q.K. Pan, P.N. Suganthan, Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion, Comput. Oper. Res. 77 (2017) 111–126.
[49] F. Zhao, H. Liu, Y. Zhang, W. Ma, C. Zhang, A discrete water wave optimization algorithm for no-wait flow shop scheduling problem, Expert Syst. Appl. (2017).
[50] 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 (7) (2008) 778–786.
[51] H. Wang, W. Wang, H. Sun, Z. Cui, S. Rahnamayan, S. Zeng, A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems, Soft Comput. 21 (15) (2017) 4297–4307.
[52] C. Zhang, J. Sun, An alternate two phases particle swarm optimization algorithm for flow shop scheduling problem, Expert Syst. Appl. 36 (3) (2009) 5162–5167.
[53] C. Zhang, J. Ning, D. Ouyang, A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem, Comput. Ind. Eng. 58 (1) (2010) 1–11.
[54] Z. Lian, X. Gu, B. Jiao, A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan, Chaos Solitons Fractals 35 (5) (2008) 851–861.
[55] A.C. Nearchou, The effect of various operators on the genetic search for large scheduling problems, Int. J. Prod. Econ. 88 (2) (2004) 191–203.

Mohamed Abdel-Basset Received his B.Sc., M.Sc and the Ph.D. in operations research from Faculty of Computers and Informatics, Zagazig University, Egypt. He is a head of department of operations research and decision support, Faculty of Computers and Informatics, Zagazig University. His current research interests are Optimization, Operations Research, Data Mining, Computational Intelligence, Applied Statistics, Decision support systems, Robust Optimization, Engineering Optimization, Multi-objective Optimization, Swarm Intelligence, Evolutionary Algorithms, and Artificial Neural Networks. He is working on the application of multi-objective and robust meta-heuristic optimization techniques. He is also an/a Editor/reviewer in different international journals and conferences. He has published more than 100 articles in international journals and conference proceedings.

M. Gunasekaran is currently working as a big data scientist in University of California, United States. He received his Ph.D. from the Vellore Institute of Technology University, India. He received his Bachelor of Engineering and Master of Technology from Anna University and Vellore Institute of Technology University respectively. He has worked as a Research Assistant for a project on spatial data mining funded by Indian Council of Medical Research, Government of India. His current research interests include data mining, big data analytics and soft computing. He is the author/co-author of papers in conferences, book chapters and journals. He got an award for young investigator from India and Southeast Asia by Bill and Melinda Gates Foundation. He is a member of International Society for Infectious Diseases and Machine Intelligence Research labs.

Doaa El-Shahat Received her B.Sc. from Zagazig University, faculty of computers and informatics, department of computer science, Egypt. His area of interest includes is computation intelligence, Optimization, Swarm Intelligence, Evolutionary Algorithms, and Artificial Neural Networks.

Seyedali Mirjalili, is a lecturer in Griffith College, Griffith University. He received his B.Sc. degree in Computer Engineering (software) from Yazd University, M.Sc. degree in Computer Science from Universiti Teknologi Malaysia (UTM), and Ph.D. in Computer Science from Griffith University. He was a member of Soft Computing Research Group (SCRG) at UTM. His research interests include Robust Optimization, Multi-objective Optimization, Swarm Intelligence, Evolutionary Algorithms, and Artificial Neural Networks. He is working on the application of multi-objective and robust meta-heuristic optimization techniques.