http://dx.doi.org/10.14500/aro.11609

Flexible Job Shop Scheduling Problem-Solving Using Apiary Organizational-Based Optimization Algorithm

Mais A. Al-Sharqi[1†], Ahmed T. Sadiq[2] and Safaa O. Al-mamory[3]

[1]Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers and Informatics, Baghdad, Iraq

[2]Department of Computer Science, University of Technology, Baghdad, Iraq

[3]Department of Cybersecurity, College of Information Technology, University of Babylon, Babylon, Iraq

***Abstract*—Flexible job shop scheduling problem (FJSSP) is a complex and challenging problem that plays a crucial role in industrial and manufacturing production. FJSSP is an expansion of the standard job shop scheduling problem (JSSP). One of FJSSP's objectives that the manufacturing system competing for is minimizing the makespan. This paper uses a new nature-inspired metaheuristic optimization algorithm called the Apiary Organizational-Based Optimization algorithm (AOOA) to solve the FJSSP. This Algorithm simulates the organizational behavior of honeybees inside the apiary and translates their activities and vital processes during their lifecycle into phases that can solve such NP-hard problems. Two benchmark datasets, Brandimarte and Hurink, with 10 MK instances and 24 (edata, rdata, and vdata) instances respectively, were used to demonstrate the ability of AOOA to solve FJSSP. Moreover, the results of AOOA were compared with a set of state-of-the-art algorithms and statistically measured using the paired samples t-test and p-value, RPD, and group-based superiority statistical analysis to test its performance. AOOA outperformed Elitism GA, Enhanced GA, Improved GA, and MOGWO in solving all 10 MK instances and HICSA in solving 9 MK instances out of 10. Moreover, AOOA overcame CS, CS-BNG, CS-ILF, CHA, and MCA in solving 24, 12, 12, 23, and 24 instances of edata, rdata, and vdata, respectively. AOOA proved its robustness, showing promising outcomes.**

***Index Terms*—Apiary Organizational-Based Optimization Algorithm, Flexible job shop scheduling, Makespan, Metaheuristic nature-inspired.**

I. Introduction

In the era of advanced technology and intelligence, time is important in manufacturing companies' competition. Production systems' scheduling is the means to utilize available resources and achieve more productivity in less time (Meng, et al., 2024). The flexible job shop scheduling problem (FJSSP) is an expansion of the standard job shop scheduling problem (JSSP) which was debuted by Burcker and Schlie in 1990 (Jiang, Yu and Chen, 2023). FJSSP is a more difficult nondeterministic polynomial hard (NP-hard) problem compared to JSP (Wang, et al., 2024). This is because every job operation can be allocated to more than one machine and might have variable processing times (Song, et al., 2022). FJSSP is a single objective and multiobjective based on the optimization problem and the aim of the researcher. Reducing the entire processing time, or makespan, is one of the most important objective functions.

The FJSS has significant applications and advantages that gained the researchers' attention (Xie, et al., 2019). Some of these are: (1) Maximizing resource utilization by allocating machines, tools, and labor in the best possible way. (2) Minimizing makespan or the total time needed to finish all tasks to maximize time efficiency. (3) Meeting client delivery criteria and deadlines by effectively scheduling jobs and resources. (4) Handling intricate production processes because jobs in such industries may involve several procedures and call for various resources at different times. (5) Scheduling systems are meant to be flexible to adapt to unexpected changes such as machine failures, urgent orders, and job priority shifting. and (6) Cost reduction via FJSS effective scheduling which not only saves cost but also enhances competitiveness.

Nature-inspired metaheuristic algorithms are commonly used to address NP-hard problems. The Organizational-Based Optimization Algorithm (AOOA) proposed by Al-Sharqi, Al-Obaidi and Al-mamory (2024) is a new nature-inspired optimization algorithm that takes its steps from the organizational behavior of honeybees inside the apiary. The contribution of this paper is to employ the concept of multiple populations of the apiary to solve the NP-hard FJSSP and minimize the makespan using the AOOA. A benchmark function of thirty-four instances distributed on four datasets was used to test AOOA's ability to solve FJSSP. AOOA performance was compared with several well-known

ARO-The Scientific Journal of Koya University
Vol. XII, No. 2 (2024), Article ID: ARO.11609. 13 pages
Doi: 10.14500/aro.11609
Received: 28 April 2024; Accepted: 02 August 2024
Regular research paper; Published: 24 August 2024
†Corresponding author's e-mail: Phd202110697@iips.edu.iq

metaheuristic algorithms and showed superiority in the results of scheduling time reduction (i.e., makespan).

The rest of this research is structured as follows. In Section 2, the related works are reviewed. Problem definition and an elucidating example are discussed in Section 3. Section 4 explains the AOOA used to solve FJSSP. In Section 5, the experimental results of FJSSP are presented and discussed. Finally, Section 6 concludes the key points of this work and highlights its possible future directions.

II. Related Work

Many metaheuristic algorithms were proposed to solve FJSSP. In FJSSP, time is the most crucial factor represented by makespan where the objective function is used to minimize it. As the proposed algorithm in this present research is inspired by the organizational behavior and lifecycle of honeybees in the apiary, there is no similar related literature. Accordingly, earlier works that were conducted to solve FJSSP are outlined here. Previous literature can be categorized into five groups based on natural operations; so many types of metaheuristic algorithms were proposed (Rajwar, Deep and Das, 2023). These algorithms can be categorized based on their underlying precept into four essential groups.

The first group can refer to evolutionary algorithms. They adopt Charles Darwin's theory of natural selection or survival of the fittest individuals. These algorithms are population-based such as Genetic Algorithms (GA), Differential Evolution (DE), Evolutionary Strategies (ES), Genetic Programming (GP), and Gene Expression Programming (GEP).

Xie and Chen, 2018) designed the elitism genetic algorithm (ELGA) that was paired with the elitism strategy in external memory to define inaccurate time quota as interval grey processing time. The proposed algorithm solved FJSSP by minimizing the interval grey makespan.

Saving energy through production scheduling is a complicated nonlinear programming problem in flexible manufacturing systems with resource flexibility and complex constraints. This led Dai, et al. (2019) to utilize intelligent scheduling techniques to reduce energy consumption. Accordingly, a multi-objective optimization model for FJSSP with transportation restrictions was developed to decrease energy usage and makespan. This was integrated with the enhanced genetic algorithm (EHGA) in which the results supported decision-makers in flexible manufacturing systems.

In another research study, an improved genetic algorithm (IGA) was proposed by (Zhang, et al., 2020). This was performed by combining the setup and transportation times with the processing time. This was to minimize the makespan time, total setup time, and total transportation time. The research introduced three various methods to improve the quality and diversity of the initial population. The crossover was improved by artificial pairing to keep reasonable solutions and improve poor solutions effectively. Furthermore, a method for adaptive weighting was executed to modify mutation probability and search ranges for individuals within the population. Standard datasets were used to support the efficacy of the proposed methodology.

Another group of metaheuristics is the physical law-based algorithms. These algorithms depend on physical phenomena such as Simulated Annealing (SA), Tabu Search (TS), Variable Neighborhood Search (VNS), and Iterated Local Search (ILS) or chemical reactions such as ions motion optimization algorithm (IMOA), and thermal exchange optimization (TEO). A research study examined the benefits and drawbacks of using variable-size batching in manufacturing scheduling (Li, et al., 2022). It was noticed that besides the flexibility of this technique, it can increase not only energy consumption but also scheduling searches. To solve this difficulty, both the makespan and the total energy consumption were taken into account. Hence, a two-stage multi-objective hybrid algorithm (HICSA) was proposed. It combined the simulated annealing algorithm (SAA) with the imperialist competitive algorithm (ICA). Through various stages, the task sequence optimization and lot-splitting approach were enhanced by performing SAA with an iterated column generation algorithm. Experiments demonstrate the efficiency of the proposed HICSA algorithm.

The third group is miscellaneous algorithms (Rajwar, Deep and Das, 2023). These algorithms have influenced various theories on human behavior, game theory, mathematical theorems, politics, artificial thought, and other subjects. Zeidabadi and Dehghani proposed the Puzzle Optimization Algorithm (POA) to solve different optimization problems. The basic idea of the proposed POA is the mathematical simulation of solving a puzzle process (Zeidabadi and Dehghani, 2022). They considered that each member of the population is a puzzle that can be updated based on the guidance of other members and tried to complete its puzzle using the pieces suggested by other members.

The fourth group is represented by the nature-inspired and swarm intelligence algorithms. The nature-inspired algorithms are primarily based on animals' natural organizational behavior or biological processes during the lifecycle. This can help them survive and adapt to their environment to achieve their task efficiently. Swarm intelligence is a decentralized and self-organized system in which individuals perform tasks via collective and cooperative behavior. This is a population-based system that reflects the primary characteristics of a swarm system in adaptability (learning by doing), effective communication, and knowledge-sharing. Examples of such algorithms are Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Firefly Algorithm (FA), Bat Algorithm (BA), Cuckoo Search (CS), and Artificial Bee Colony (ABC).

Al-Obaidi and Hussein, (2016) enhanced the Cuckoo Search algorithm (CS). The initial modification was made on the Best Neighbors Generation (CS-BNG) prediction, whereas the second was based on the utilization of Iterative Levy Flight (CS-ILF). Adjustments were made to the defining characteristics of the CS algorithm to improve search performance in the discrete state space. The proposed algorithms enhanced the solutions' quality and convergence rate. Several FJSSP benchmark situations were tested to evaluate the performance of the enhanced algorithms where they outperformed the original cuckoo search method.

To make a balance between production competence and environmental leverage, energy-efficient scheduling was

 http://dx.doi.org/10.14500/aro.11609

prioritized for green manufacturing (Luo, Zhang and Fan, 2019). The research addressed the multi-objective flexible job shop scheduling problem (MOFJSP) by employing variable processing speed to minimize makespan and total energy usage simultaneously. A multi-objective grey wolf optimization (MOGWO) algorithm, which utilizes a three-vector chromosomal encoding form, was introduced. The proposed algorithm takes into consideration the assignment of machines, speeds, and operation order. Extensive numerical experiments on thirty-five different scale benchmarks were conducted to verify the effectiveness of the suggested algorithm.

Based on the behavior of meerkats in the Kalahari Desert, Sadiq Al-Obaidi, Abdullah and Ahmed (2018) introduced the meerkat clan algorithm (MCA) as a new swarm intelligence algorithm. By utilizing meerkat behavior such as sentry, foraging, and babysitter, the algorithm partitions solution sets and executes operations specifically on the foraging set. It was found that the sentry identified the best solution. MCA was used in solving FJSSP where the experimental results showed better efficiency than the original Cuckoo Search, Artificial Fish Search, and Camel Herd algorithms.

Al-Obaidi et al. proposed a new nature-inspired algorithm that mimics camel herds' behavior (Sadiq Al-Obaidi, Abdullah and Ahmed, 2017). The role of the leader of each herd was considered, which guided the herd to the source of food and water depending on humidity value with neighboring strategy. FJSSP was undertaken as a case study to verify the proposed algorithm, Camel Herds Algorithm (CHA) in which the results proved the ability of CHA to find the optimal solution in the problem space. Table I summarizes the metaheuristic algorithms that tackle the FJSSP.

The Apiary Organizational-Based Optimization algorithm (AOOA) is a new nature-inspired algorithm that adopts the concept of multiple populations by introducing the apiary concept. AOOA simulates the organizational behavior of honeybees (the queen, drones, and workers) inside the apiary and the biological activities of workers' bees during their lifecycle. In this work, AOOA is employed to solve the FJSSP where the concept of multiple populations is utilized to minimize the makespan.

III. Problem Definition

FJSSP is widely regarded as an NP-hard problem because of (i) deciding which operation can be served by the machine, which is known as the routing problem, and (ii) decisions on the sequence of operations on each machine, which is known as the sequence problem (Alzaqebah, et al., 2022). According to Wu and Cai (2021); Jedrzejowicz and Wierzbowska (2022), the n multiplies m FJSSP is described as:

A set of n jobs $J= \{J1, J2,\ldots,Jn\}$ must be handled on a set of M machines $M=\{M1, M2,\ldots,Mn\}$. Each job $j \in J$ is characterized by a sequence of nj operations $Oj= \{Oj,1, Oj,2,\ldots, Oj, nj\}$. Each task might have a unique sequence of operations. These operations must be executed in the specified order. The following constraints characterize the problem (Bissoli, et al., 2018; Amirteimoori, et al., 2022):

1. For every operation $o \in Oj$ of job j, there is a subset MSo of machines that execute the operation o with the duration dr for each $r \in MSo$.
2. Each operation o must be processed on precisely one machine $r \in MSo$ and each machine can process exactly one operation at a time.
3. For each operation o processed on r machine, an uninterrupted duration dr is assigned.
4. Similar tasks adhere to a predetermined order.
5. Only one machine can be selected for each operation.
6. All jobs can be executed at the beginning of the work, and all machines are available.
7. As the processing proceeds, no preemptions or cancellations are permitted.
8. Each operation o has its beginning and end time co. A schedule is a set of all completion times $\{co \mid o \in Oj, 1 \leq j \leq n\}$ that satisfies all aforementioned constraints.
9. The time necessary to complete whole tasks is known as makespan: $C_{max} = \max_{o \in O_{j,1 \leq j \leq n}} c_o$

The makespan is an optimization criterion which means identifying the schedule that minimizes it. Some constraints that may be considered with makespan finding in FJSSP:

(i) All machines are obtainable at time zero, and job groups are launched.

(ii) Machine setup time and the transportation time between operations are included in the processing time (Lu et al., 2019; Şahman, 2021; Zhang, et al., 2021; Zhu, et al., 2022)

A. An Elucidation Example

An example is provided here to explain FJSSP. Given a set of three jobs (J=3) and three machines (M=3). Each job consists of a sequence of operations (Oj,i, and i is the number of operations for the job), which must be processed

TABLE I
Optimization Objective of Different Algorithms for Solving FJSSP

Algorithm	Objective	Classification
ELGA	Minimize process time and makespan.	Evolutionary algorithm
IGA	Minimize processing time, set-up time, and transportation time.	Evolutionary algorithm
EHGA	Minimize Makespan	Evolutionary algorithm
HICSA	Minimize makespan and the total energy consumption	Physical law-based algorithms
CS, CS-BNG, CS-ILF	Minimize Makespan	Nature-inspired and swarm intelligence algorithms
MOGWO	Minimize Makespan and total energy usage	Nature-inspired and swarm intelligence algorithms
MCA	Minimize Makespan	Nature-inspired and swarm intelligence algorithms
CHA	Minimize Makespan	Nature-inspired and swarm intelligence algorithms

consecutively. As shown in Table I, *J1* has *O11, O12*, and *O13*. Every operation can be assigned to at least one elected machine out of a predefined subset of machines with a specific operation processing time on that machine. In other words, it can be suggested that *M1* or *M3* for *O11* with times 3 or 6, *M2* or *M3* for *O12* with times 9 or 2, and *M1* or *M2* for *O13* with times 1 or 4, respectively.

J2 also consists of three operations: *O21, O22,* and *O23*. Each has a subset of machines that can be allocated for each operation within the determined time. *O21* can be implemented on *M1, M2* or *M3* with time =3, 7 or 1, respectively. Correspondingly, *O22* can be implemented on *M2* or *M3* with time = 7 or 1. *O23* can be performed on *M1* or *M3* with time = 8 or 5, respectively.

J3 has only two operations, *O31* and *O32*, with a subset of machines *M1* or *M2* for *O31* with times 3 or 6, and *M2* or *M3* for OS with times 5 or 9, respectively as they can be processed within the associated time. The details of the explained example are depicted in Table II.

IV. The Use of AOOA for FJSSP Solving

To employ AOOA for FJSSP solving, the population representation and the AOOA phases were determined and the phases were adapted to be convenient for FJSSP.

A. AOOA Stages

AOOA is a new nature-inspired metaheuristic optimization algorithm proposed by Al-Sharqi, Al-Obaidi and Al-mamory (2024). Its concept depends on organizational behavior and biological processes of honeybees inside the apiary. Moreover, AOOA simulates the activities of hives' members (queen, drone, and workers) during their lifecycle. AOOA consists of seven phases corresponding to honeybees' different activities inside the apiary. These phases are initialization, drone exchange, Fertilization and Bees Breeding, Worker Lifecycle, Queen Investiture, Fading out, and Swarming. Fig. 1 illustrates the AOOA phases.

In the AOOA first phase, the population is randomly initiated considering the constraints of each job. The population consists of (*h*) numbers of hives each hive has a (*b*) number of bees. The population size is calculated according to Equation (1).

$$N = h \times b \tag{1}$$

TABLE II
The Processing Time and Operation Assignment for a Set of Jobs

Job	Operation	Machine		
		M1	*M2*	*M3*
J1	*O11*	3	-	6
	O12	-	9	2
	O13	1	4	-
J2	*O21*	3	7	1
	O22	-	7	1
	O23	8	-	5
J3	*O31*	3	6	-
	O32	-	5	9

For FJSSP, the population consists of (N) numbers of hives and each hive has (*r*) jobs and (*m*) machines with (*k*) operations for each machine. The population was represented as a set of hives, each with a three-dimensional array of jobs (*job*), operations (*op*), and machines (*M*). Each 2D array (i.e. an individual) represents a possible scheduling solution as a sequence of different operations of jobs for each machine. Fig. 2 shows the population representation of FJSSP.

Drone exchange is the second phase of AOOA in which a randomly selected drone of one hive is exchanged with a drone from another hive as shown in Fig. 3.

In the third phase, the queen of each hive is fertilized with a list of fittest drones (those who have the minimum makespan) to produce (*k*) new bees according to the fertilization equation shown in (2) under the control of $fert_{ratio}$.

$$New\_b_{H_i} = \begin{cases} Q_{H_i} + rand\left(d_{H_i} - Q_{H_i}\right), & if \left(\dfrac{d_{H_i} - Q_{H_i}}{d_{H_i}}\right) < 0.5 \\ \frac{1}{2}(Q_{H_i}) + rand\ \frac{1}{2}\left(d_{H_i}\right), & O.W \end{cases} \tag{2}$$

Where Q_{H_i}, d_{H_i} represent the queen and drone of the ith hive, respectively. At the same time, (*rand*) represents a random number within [0,1].

To adapt AOOA to the FJSSP, a random number of machines (i.e., columns) is selected for each drone to fertilize the queen. The queen is searched for each job and operation ($J_{n,nj}$) pair in the selected machine of the drone. If found, it will be the first index of the newly generated bee. If it is not found, search for the next pair. The remaining pairs of ($J_{n,\,nj}$) will be given the rest of the indices as depicted in Fig. 4.

The newly generated bees are either drones or workers according to randomly generated value (*rand*).

The worker lifecycle is the fourth phase. This phase simulates the behavior of workers bees. Each worker is assigned an age representing the likelihood of transforming an existing bee into a new one. From a biological standpoint, the lifespan of workers spans from a minimum of 1 day to a maximum of 60 days. The worker's life cycle is defined as the duration of the worker's life in terms of the number of days to perform simple or 1-opt (i.e., one-optima), 2-opt, and 3-opt based on the worker's age translocation as follows:

1–10: applying simple translocation on the worker (i),
11–25: applying 2-opt worker (i),
26–50: applying 3-opt worker (i),
51–60: applying 2-opt worker (i),
Otherwise, dropping worker (i).

As noted in (3), each period involves a certain number of modifications on the chosen worker to generate a new bee.

$$NW_{H_i} = W_{H_i} + rand.\left(\frac{W_{H_i}^{\max\_g} - W_{H_i}^{g}}{W_{H_i}^{\max\_g}}\right) \times W_{H_i} \tag{3}$$

Where W_{H_i} is the worker of the i^{th} hive, (*rand*) is a random number within [0,1), and w^{max_g} is the maximum age of the

http://dx.doi.org/10.14500/aro.11609

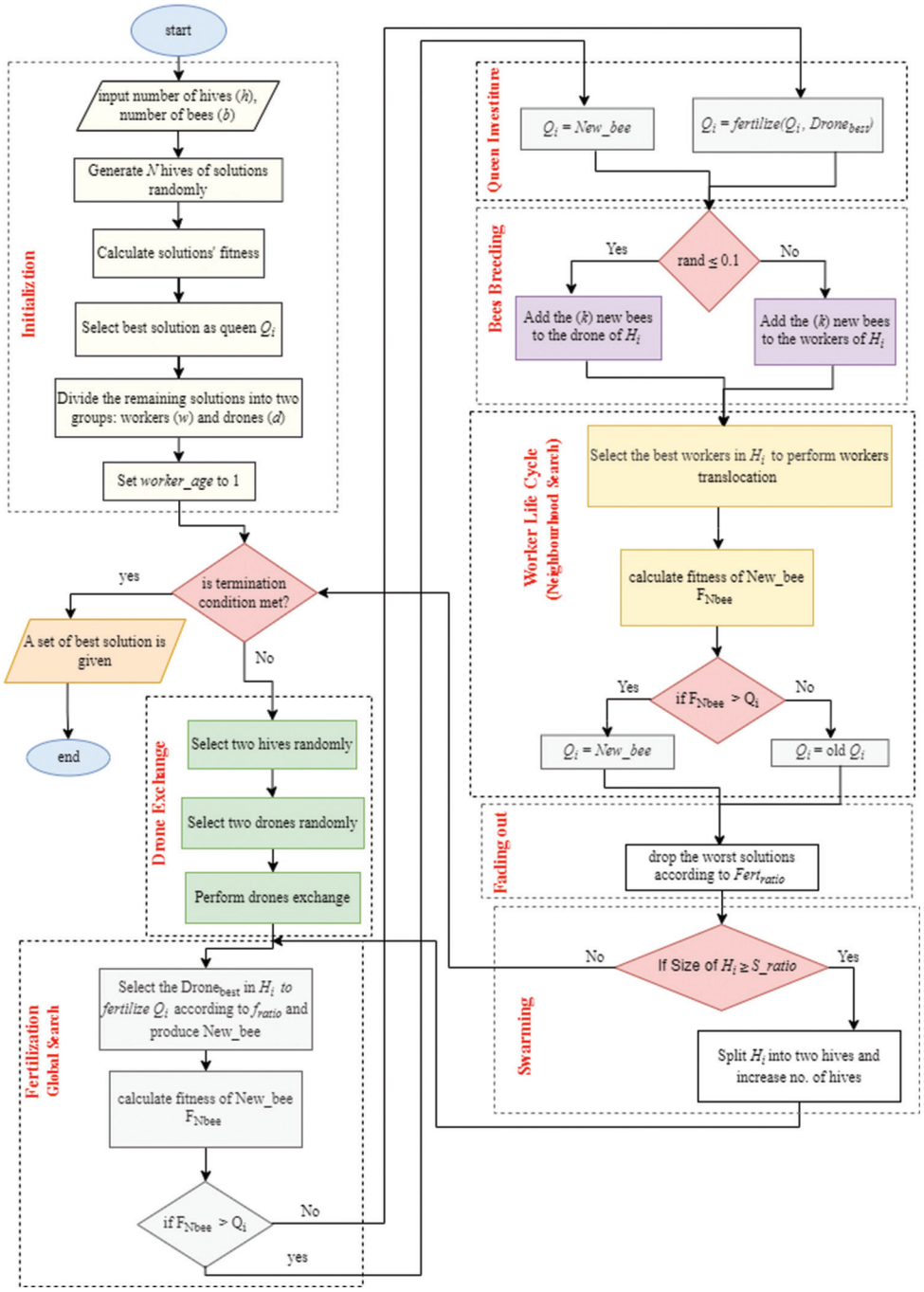

Fig. 1. Apiary Organizational-Based Optimization algorithm phases.

worker. To adopt AOOA for FJSSP, some columns are selected randomly where each column represents a specific machine. According to (*rand*), one, two, or three positions are randomly chosen to perform simple, 2-opt, or 3-opt translocations, respectively. An example of the worker translocation process is illustrated in Fig. 5, where the selected worker consists of five machines M1, M2, M3, M4, and M5. The randomly selected machines M1, M2, and M4 perform 3-opt, 2-opt, and simple (1-opt) respectively to generate a new bee. The fitness of new bees is calculated and it will be added to the population.

In the fifth phase, queen investiture, if the fitness of the new bee is better than the queen then the newly generated bee will be invested as the queen of this hive. This step guides the search toward enhanced solutions by exploiting good ones to find the optimal solution(s).

Naturally, bees' max age is 60 days (sometimes less because of climate or diseases). They die at the end of their lifecycle. On the other hand, drones fade out after the queen fertilization process. AOOA simulates this behavior into the sixth phase, the fading out phase. AOOA controls the increasing size of populations in each hive of the apiary by adding a control parameter called fd_{ratio} which dominates the number of dying bees. The fd_{ratio} is ranged between 10% and 30% of bees (a nature-inspired ratio).

The last phase of AOOA's phases is the swarming phase. This phase mimics the swarming phenomenon of honeybee

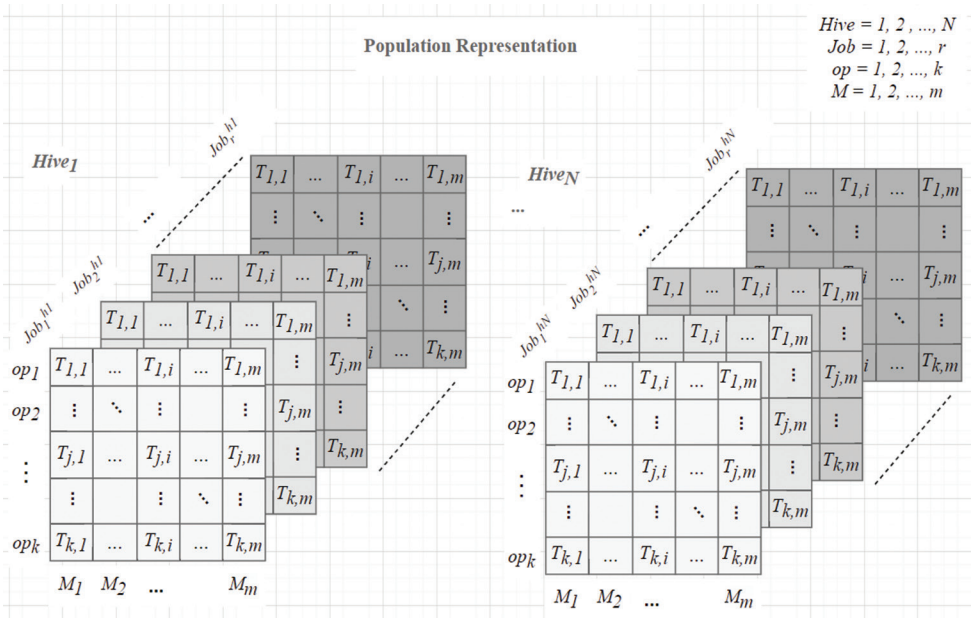

Fig. 2. Flexible job shop scheduling problem population representation.

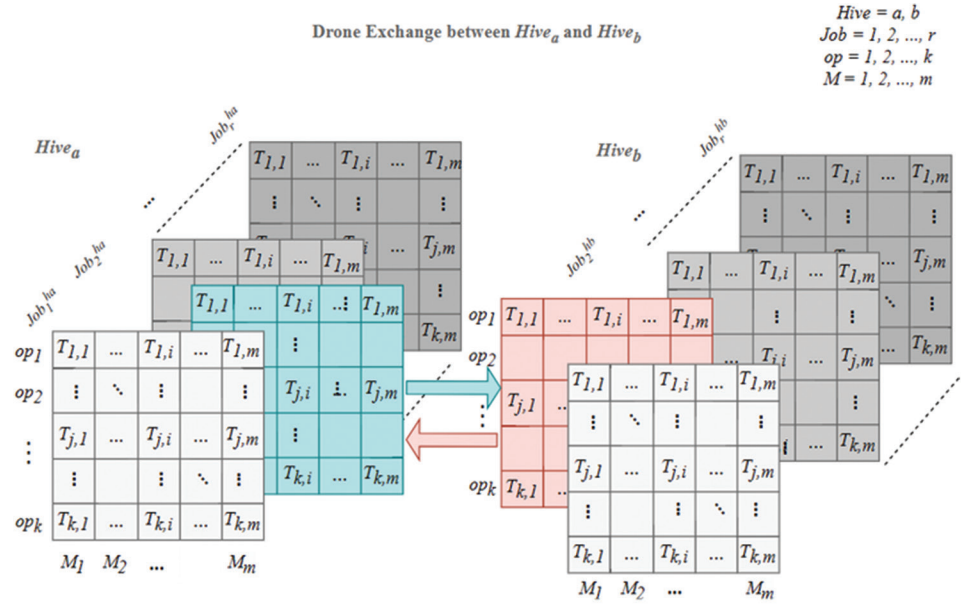

Fig. 3. Drone exchange between two randomly selected hives.

behavior in which the bees split the crowded hive into two hives. S_{ratio} is a control parameter added to manage the hive separation process. AOOA assumed that S_{ratio} is a user-defined value tuned based on the problem.

V. Experiments and Results

A. Comparison of the AOOA Performance with the Competing Algorithms

To confirm the efficacy and resilience of the suggested AOOA, it was compared to some competing metaheuristic algorithms using a total of 34 benchmark instances, with different sizes (small, medium, and large) scales, were used in the experiments. These benchmark instances consist of Brandimarte Data (BRdata) (MK01-MK10) (Brandimarte, 1993) and Hurink edata (la01-la08), rdata (la01-la08), and vdata (la01-la08) (Hurink, Jurisch and Thole, 1994).

The first test was to investigate the AOOA Performance of MK01-MK10 Instances. The effectiveness of AOOA's performance was validated by comparing the FJSSP results with other metaheuristic algorithms, namely, the ELGA, EHGA, IGA (Zhang et al., 2020), MOGWO (Luo, Zhang and Fan, 2019), and HICSA (Li, et al., 2022). The experiments were performed through 10 independent runs and 50 iterations for each run. The experimental results were evaluated statistically using the relative percentage difference (RPD) and compared to the lower bound (LB). FJSSP has a known LB to which the results would be comparable. The calculation method of RPD is shown in (4)

http://dx.doi.org/10.14500/aro.11609

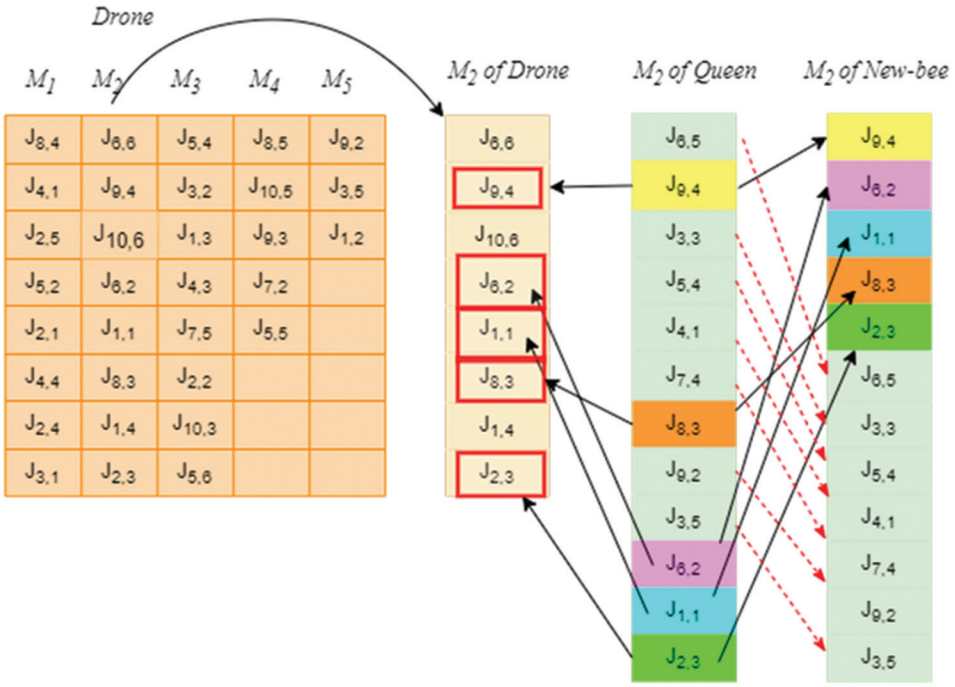

Fig. 4. Queen fertilization with the best drone.

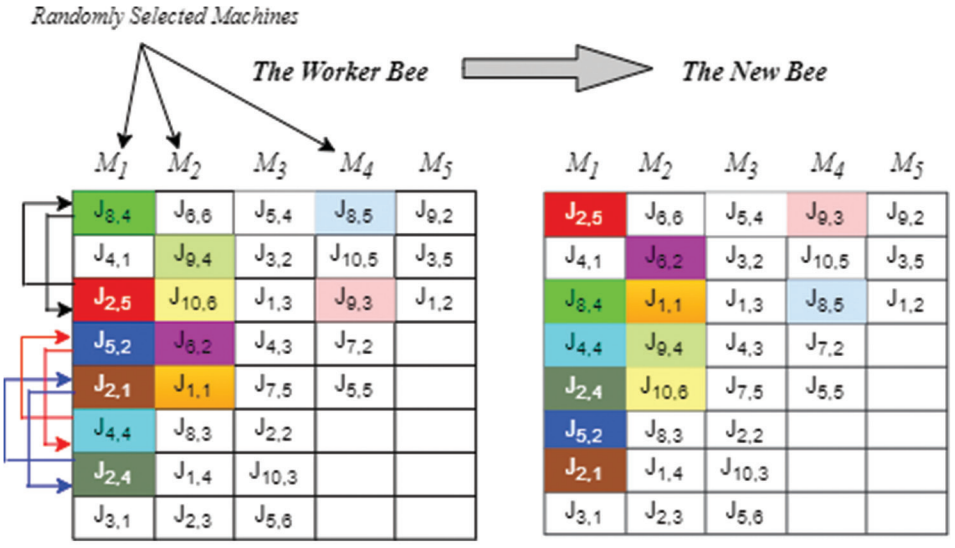

Fig. 5. Worker translocation process.

$$RPD = \frac{Best\ C_{max} - LB}{\left(\frac{Best\ C_{max} + LB}{2}\right)} \times 100\% \quad (4)$$

Where Best C_{max} denotes the optimal makespan value produced by the algorithm, and LB denotes each instance's known lower bound value. Based on Equation 4, it can be noticed that a decrease in the RPD value could correspond to a reduction in the gap between an algorithm solution result and LB, indicating a closer match between the two values. Hence, if the optimal solution found by AOOA matches LB, the result of RPD equals zero. Table III displays the problem instance, its size, the LB, the minimum makespan (best C_{max}), RPD, and Rank which ranks the AOOA and the competing algorithms according to the RPD value. The optimal value acquired by AOOA has lower RPD values than other algorithms for MK01-MK09 and a competitive near-lower value for MK10. This reflects the excellent performance of AOOA. Best RPD results are denoted in bold and a zero-value means achieving LB. The optimal values that match LB are highlighted in bold with the (*) symbol.

The outcomes confirm that AOOA achieves first rank in comparison with ELGA, EHGA, IGA, MOGWO, and HICSA in MK01-MK09 and comes in second rank in MK10. AOOA successfully found the optimal solution for MK08. This validates the superiority of the proposed algorithm in

TABLE III
COMPARISON RESULTS OF MAKESPAN FOR INSTANCES (MK01-MK10) WITH OTHER ALGORITHMS OF (LUO, ZHANG AND FAN, 2019) AND (LI, *ET AL.*, 2022)

Instance	Size	LB	Parameter	ELGA	EHGA	IGA	MOGWO	HICSA	AOOA
MK01	10×6	36	Best C_{max}	82	68	65	70	87	**45**
			RPD	78	61.5	57.4	64.2	82.9	**22.2**
			Rank	5	3	2	4	6	**1**
MK02	10×6	24	Best C_{max}	63	57	52	61	80	**41**
			RPD	62.9	89.7	81.5	73.7	87.1	**52.3**
			Rank	5	3	2	4	6	**1**
MK03	15×8	204	Best C_{max}	350	330	313	322	445	**247**
			RPD	27.8	52.7	47.2	42.2	44.9	**19.1**
			Rank	2	6	5	3	4	**1**
MK04	15×8	48	Best C_{max}	120	112	107	106	162	**65**
			RPD	85.7	80	76.1	75.3	108.6	**30.1**
			Rank	5	4	3	2	6	**1**
MK05	15×4	168	Best C_{max}	314	284	276	287	447	**197**
			RPD	60.6	51.3	48.6	52.3	90.7	**15.9**
			Rank	5	3	2	4	6	**1**
MK06	10×15	33	Best C_{max}	174	158	129	159	207	**121**
			RPD	136.2	130.9	118.5	131.3	145	**114.3**
			Rank	5	3	2	4	6	**1**
MK07	20×5	133	Best C_{max}	311	265	267	255	418	**197**
			RPD	80.2	66.3	67	62.9	103.4	**38.8**
			Rank	5	3	4	2	6	**1**
MK08	20×10	523	Best C_{max}	805	783	781	792	543	**523***
			RPD	42.5	39.8	39.6	40.9	3.8	**0**
			Rank	6	4	3	5	2	**1**
MK09	20×10	299	Best C_{max}	615	566	567	548	480	**424**
			RPD	69.1	61.7	61.9	58.8	46.5	**34.6**
			Rank	6	4	5	3	2	**1**
MK10	20×15	165	Best C_{max}	474	494	424	457	304	357
			RPD	96.7	99.8	87.9	93.9	59.3	73.6
			Rank	5	6	3	4	1	2

exploring the search region and guiding the search toward optimal solutions.

The second test was to verify the robustness and effectiveness of AOOA on 24 instances of the Hurink dataset. The test was partitioned into three subtests: edata, rdata, and vdata with (la1- la8) for each. A set of five metaheuristic optimization algorithms: CS, CS-BNG, CS-ILF (Al-Obaidi and Hussein, 2016), CHA (Al-Obaidi, Abdullah and Ahmed, 2017), and MCA (Al-Obaidi, Abdullah and Ahmed, 2018) was selected for comparison with AOOA. Table IV illustrates the comparison results of Hurink datasets for the compared algorithms. The comparison was based on the RPD and ranked the competing algorithms accordingly. The best values are identified in bold type and the optimal values that match LB are highlighted in bold with the (*) symbol.

The results show the superiority of AOOA rank in edata la1-la8 over other algorithms. In rdata, AOOA was superior in la2 and came in the second rank in la3 while it came in the third rank in la1, la4, la5, la6, la7, and la8. For the vdata, the results proved the supremacy of AOOA in la1, la3, and la7 instances while it came in the second rank in la4 and la5. AOOA came in the third rank in la6 and la7. For la2, AOOA is in the fourth rank of RPD.

TABLE IV
COMPARISON RESULTS OF MAKESPAN FOR HURINK WITH COMPETING ALGORITHMS

Instance	Size	LB	Parameter	CS	CS-BNG	CS-ILF	CHA	MCA	AOOA
edata-la1	10×5	609	Best C_{max}	729	636	634	888	780	**614**
			RPD	17.9	4.3	4	37.3	24.6	**0.8**
			Rank	4	3	2	6	5	**1**
edata-la2	10×5	655	Best C_{max}	783	707	694	823	724	**655***
			RPD	17.8	7.6	5.8	22.7	10	**0**
			Rank	5	3	2	6	4	**1**
edata-la3	10×5	550	Best C_{max}	663	593	588	732	706	**550***
			RPD	18.6	7.5	6.7	28.4	24.8	**0**
			Rank	4	3	2	6	5	**1**
edata-la4	10×5	568	Best C_{max}	709	620	619	830	703	**568***
			RPD	22.1	8.8	8.6	37.5	21.2	**0**
			Rank	5	3	2	6	4	**1**
edata-la5	10×5	503	Best C_{max}	605	525	526	681	655	**522**
			RPD	18.4	4.3	4.5	30.1	26.3	**3.7**
			Rank	4	2	3	6	5	**1**
edata-la6	15×5	833	Best C_{max}	976	864	861	1332	1229	**833***
			RPD	15.8	3.7	3.3	46.1	38.4	**0**
			Rank	4	3	2	6	5	**1**
edata-la7	15×5	762	Best C_{max}	960	818	819	1255	1136	**785**
			RPD	23	7.1	7.2	48.9	39.4	**3**
			Rank	4	2	3	6	5	**1**
edata-la8	15×5	845	Best C_{max}	1001	880	868	1257	1129	**852**
			RPD	16.9	4.1	2.7	39.2	28.8	**0.8**
			Rank	4	3	2	6	5	**1**
rdata-la1	10×5	570	Best C_{max}	723	607	609	665	789	621
			RPD	23.7	6.3	6.6	15.4	32.2	8.6
			Rank	5	1	2	4	6	3
rdata-la2	10×5	529	Best C_{max}	680	573	567	633	770	**554**
			RPD	25	8	6.9	17.9	37.1	**4.6**
			Rank	5	3	2	4	6	**1**
rdata-la3	10×5	477	Best C_{max}	621	518	512	590	708	514
			RPD	26.2	8.2	7.1	21.2	39	7.5
			Rank	5	3	1	4	6	2
rdata-la4	10×5	502	Best C_{max}	646	542	538	623	720	551
			RPD	25.1	7.7	6.9	21.5	35.7	9.3
			Rank	5	2	1	4	6	3
rdata-la5	10×5	457	Best C_{max}	577	484	480	568	667	489
			RPD	23.2	5.7	4.9	21.7	37.4	6.8
			Rank	5	2	1	4	6	3
rdata-la6	15×5	799	Best C_{max}	974	832	821	1044	1252	857
			RPD	19.7	4	2.7	26.6	44.2	7
			Rank	4	2	1	5	6	3
rdata-la7	15×5	749	Best C_{max}	917	779	776	1005	1123	802
			RPD	20.2	3.9	3.5	29.2	40	6.8
			Rank	4	2	1	5	6	3
rdata-la8	15×5	765	Best C_{max}	938	793	790	1061	1203	823
			RPD	20.3	3.6	3.2	32.4	44.5	7.3
			Rank	4	2	1	5	6	3
vdata-la1	10×5	570	Best C_{max}	728	613	609	671	815	**607**
			RPD	24.3	7.3	6.6	16.3	35.4	**6.3**
			Rank	5	3	2	4	6	**1**
vdata-la2	10×5	529	Best C_{max}	675	565	564	560	770	574
			RPD	24.3	6.6	6.4	5.7	37.1	8.2
			Rank	5	3	2	1	6	4
vdata-la3	10×5	477	Best C_{max}	627	515	520	574	706	**515**
			RPD	27.2	7.7	8.6	18.5	38.7	**7.7**
			Rank	4	1	2	3	5	**1**
vdata-la4	10×5	502	Best C_{max}	652	534	531	641	706	534

(Contd...)

http://dx.doi.org/10.14500/aro.11609

TABLE IV
(Continued)

Instance	Size	LB	Parameter	CS	CS-BNG	CS-ILF	CHA	MCA	AOOA
			RPD	26	6.2	5.6	24.3	33.8	6.2
			Rank	4	2	1	3	5	2
vdata-la5	10×5	457	Best C_{max}	587	485	499	531	645	499
			RPD	24.9	5.9	8.8	15	34.1	8.8
			Rank	4	1	2	3	5	2
vdata-la6	15×5	799	Best C_{max}	981	826	821	1042	1212	839
			RPD	20.4	3.3	2.7	26.4	41.1	4.9
			Rank	4	2	1	5	6	3
vdata-la7	15×5	749	Best C_{max}	941	774	773	968	1130	796
			RPD	22.7	3.3	3.2	25.5	40.6	6.1
			Rank	4	2	1	5	6	3
vdata-la8	15×5	765	Best C_{max}	952	779	787	908	1181	**777**
			RPD	21.8	1.8	2.8	17.1	42.8	**1.6**
			Rank	5	2	3	4	6	**1**

B. Sensitivity Analysis

AOOA population consists of two critical parameters: the number of hives (*h*) and bees (*b*). As the number of machines and jobs vary in FJSSP, the population size of AOOA should also vary to fit the search space. Therefore, it makes sense to inspect the impact of different (*h*) and (*b*). For this reason, two scales were used to fit all sizes of NxM where N is the number of jobs and M is the number of machines. SSA and MSA apiary were tested for each instance of benchmark data with small, medium, and large-sized jobs and machines. This can disclose whether SSA can find the best results, so there is no need to increase computational complexity by increasing population size and its consequent computations using MSA. Table V illustrates the apiary scales of FJSSP. Each instance was tested against all hive sizes with all bees' numbers. However, experiments manifest that increasing (*h*) and (*b*) to be convenient for LSA yields increasing computational time at a dramatic rate. Therefore, MSA was adopted to include both medium and large-scale FJSS problem sizes.

By observing the results of all 34 instances illustrated in Table VI, it is clear that the minimum makespan was found using SSA for small factories with small-scale jobs and machines of MK instances (i.e., MK1 and MK2). When the size of factories increases, the minimum makespan was found in MSA, MK3, MK5, MK7, MK10. In MK4 and MK6, there was a slight difference in SSA and MSA results in both best and average makespan where the best was found in SSA. In MK8 and MK9 both SSA and MSA gave the same results. Larger values than eight and seventy for h and b, respectively failed to achieve closer results to the optimal solution but this significantly increased the implementation time. Moreover, most optimal solutions (i.e., minimum makespan) appeared in the early iterations, and the ultimate solution appeared in iteration 50.

It was evident in the results of edata, rdata, and vdata that 70% of best C_{max} was found using MSA in edata (la1, la6, la7, and la8), rdata (la1, la4, la6, la7, and la8), and vdata (la2, la4, la5, la7, and la8). Edata la2, la3, and la4, SSA, and MSA gave the same results, but the mean C_{max} of MSA was better than the C_{max} of SSA. Therefore, edata la2, la3, and la4 results were counted within the 17 instances of the best C_{max} of MSA (i.e. 70%). The best C_{max} resulting from SSA was 7 instances with 30% in edata (la5), rdata (la2, la3, and la5), and vdata (la1, la3, and la6). The results proved that the effect of increasing population size using MSA on enhancing the minimum and mean makespan is bigger for MS factories than for SS factories. The best values of (Best Cmax) and (Mean Cmax) are identified in bold type.

TABLE V
Apiary Parameter Values According to FJSSP Problem Size

Problem size			Apiary Scale	Number of Hives	Number of Bees
Job		Machines			
Small	10	5	Small-scale apiary (SSA)	3	30
	10	6		4	40
Medium	15	4		5	50
	15	5	Medium scale apiary (MSA)	6	50
	15	8		7	60
	20	5		8	70
Large	10	15			
	20	10			
	20	15			

C. Performance Analysis

In this section, the results and findings analysis will be discussed. The discussion is divided into two parts: statistical performance analysis of AOOA with the competing algorithms and limitations.

The first part is divided into two subparts. The first subpart investigates the relationship between the AOOA and the five competing algorithms using a paired samples t-test. We analyzed the data from samples of the six participant algorithms, categorizing them based on (ELGA, EHGA, IGA, MOGWO, HICSA, and AOOA) for MK instances and (CS, CS-BNG, CS-ILF, CHA, MCA, and AOOA) for Hurink dataset. Table VII shows the p-values for the MK01-MK10, edata, rdata, and vdata instances of the AOOA vs. the competing algorithms with degrees of freedom (*df*=8), sample size (*N*=9), and significance level (α = 0.05). The p-values $\geq$ (α) are highlighted in bold type.

The paired samples t-test yielded a *t* statistic and the calculated p-value shown in Table VII, indicating an extremely statistically significant difference between AOOA and other algorithms at the α = 0.05 level. The low p-values of AOOA (except for edata5, rdata2 - rdata5 of CS-BNG, and CS-ILF) indicate strong evidence against the null hypothesis proving that the results are significant.

The second subpart measures the AOOA performance for the 34 instances compared to the competing algorithms using group-based superiority in finding the best makespan. AOOA outperformed ELGA, EHGA, IGA, and MOGWO in 10 MK instances with 100% superiority. On the other hand, it outperformed HICSA in 9 MK instances (MK1-MK9) and competed with HICSA on MK10. For edata, AOOA was superior to all competing algorithms: CS, CS-BNG, CS-ILF, CHA, and MCA with 100% superiority. In rdata and vdata, AOOA was superior to CS and MCA in all 16 instances while it was superior to CHA in all eight rdata instances and seven

TABLE VI
BEST AND MEAN MAKESPAN RESULTS USING SSA AND MSA OF MK1-MK10

Instance	Apiary Scale	Best C_{max}	Mean C_{max}	Instance	Apiary Scale	Best C_{max}	Mean C_{max}
MK1	SSA	**45**	54.1	edata-la8	SSA	854	**878.7**
	MSA	46	53.9		MSA	**852**	888.2
MK2	SSA	**41**	47.8	rdata-la1	SSA	632	678.9
	MSA	43	**46**		MSA	**621**	**667.2**
MK3	SSA	257	274.3	rdata-la2	SSA	**554**	640
	MSA	**247**	**270.2**		MSA	588	**622.3**
MK4	SSA	**65**	85.2	rdata-la3	SSA	**514**	592.4
	MSA	70	**81.1**		MSA	525	**573.9**
MK5	SSA	199	213.4	rdata-la4	SSA	562	611.1
	MSA	**197**	**209.7**		MSA	**551**	**595.3**
MK6	SSA	**121**	138.8	rdata-la5	SSA	**489**	549.1
	MSA	125	**135.5**		MSA	517	**548.2**
MK7	SSA	203	228.6	rdata-la6	SSA	894	927.9
	MSA	**197**	**227.1**		MSA	**857**	**914.73**
MK8	SSA	523	523	rdata-la7	SSA	814	863.9
	MSA	523	523		MSA	**802**	**852.8**
MK9	SSA	424	454.5	rdata-la8	SSA	834	885.4
	MSA	424	**450.8**		MSA	**823**	**884.2**
MK10	SSA	363	**378.6**	vdata-la1	SSA	**607**	685.2
	MSA	**357**	381.6		MSA	644	**677.4**
edata-la1	SSA	637	672.9	vdata-la2	SSA	576	637.3
	MSA	**614**	**650**		MSA	**574**	**627.6**
edata-la2	SSA	655	673.9	vdata-la3	SSA	**515**	573.7
	MSA	655	**663.2**		MSA	539	**565.1**
edata-la3	SSA	550	566.3	vdata-la4	SSA	545	603.7
	MSA	550	**563.9**		MSA	**534**	**586.4**
edata-la4	SSA	568	597.5	vdata-la5	SSA	503	557.8
	MSA	568	**587.8**		MSA	**499**	**538.2**
edata-la5	SSA	**522**	560.3	vdata-la6	SSA	**839**	**911.3**
	MSA	524	**551.9**		MSA	870	920.4
edata-la6	SSA	849	909.3	vdata-la7	SSA	810	871.4
	MSA	**833**	**892.4**		MSA	**796**	**866.7**
edata-la7	SSA	795	844.7	vdata-la8	SSA	834	902.2
	MSA	**785**	**826.7**		MSA	**777**	**874.5**

TABLE VII
PAIRED SAMPLES T-TEST AND *P* VALUE FOR AOOA VERSUS OTHER ALGORITHMS OF INSTANCES MK01-10, EDATA01-08, RDATA01-08, AND VDATA01-08, *DF*=8, *N*=9, A=0.05

PI	AOOA VS.	t	Sig. (2-tailed) p-value	PI	AOOA VS.	t	Sig. (2-tailed) p-value
MK1	ELGA	26.407	4.54629E-09	MK6	ELGA	21.964	1.94932E-08
	EHGA	18.141	8.75736E-08		EHGA	55.763	1.18696E-11
	IGA	15.656	2.76462E-07		IGA	7.069	0.00011
	MOGWO	20.346	3.56036E-08		MOGWO	26.382	4.58033E-09
	HICSA	26.328	4.65453E-09		HICSA	52.178	2.01724E-11
MK2	ELGA	20.084	3.94225E-08	MK7	ELGA	35.207	4.63615E-10
	EHGA	17.605	1.10756E-07		EHGA	17.519	1.15079E-07
	IGA	9.197	1.57994E-05		IGA	18.973	6.16255E-08
	MOGWO	10.670	5.22089E-06		MOGWO	14.493	5.02867E-07
	HICSA	27.086	3.71812E-09		HICSA	69.046	2.15518E-12
MK3	ELGA	27.721	3.09392E-09	MK8	ELGA	149.000	4.60432E-15
	EHGA	28.156	2.73463E-09		EHGA	229.399	1.45963E-16
	IGA	20.616	3.20934E-08		IGA	188.259	7.09268E-16
	MOGWO	23.233	1.25126E-08		MOGWO	155.568	3.26094E-15
	HICSA	46.764	4.83285E-11		HICSA	19.228	5.54874E-08
MK4	ELGA	66.287	2.98493E-12	MK9	ELGA	70.795	1.76492E-12
	EHGA	31.825	1.03469E-09		EHGA	39.941	1.69834E-10
	IGA	33.942	6.20127E-10		IGA	40.510	1.51749E-10

(Contd...)

http://dx.doi.org/10.14500/aro.11609

TABLE VII
(Continued)

PI	AOOA VS.	t	Sig. (2-tailed) p-value	PI	AOOA VS.	t	Sig. (2-tailed) p-value
	MOGWO	22.277	1.74305E-08		MOGWO	40.827	1.42600E-10
	HICSA	92.655	2.05504E-13		HICSA	16.397	1.92877E-07
MK5	ELGA	83.326	4.79935E-13	MK10	ELGA	33.070	7.62695E-10
	EHGA	45.667	5.84033E-11		EHGA	36.765	3.28452E-10
	IGA	59.713	6.87372E-12		IGA	21.805	2.06362E-08
	MOGWO	50.579	2.58547E-11		MOGWO	43.756	8.21037E-11
	HICSA	153.139	3.69824E-15		HICSA	-9.330	1.42094E-05
edata-la1	CS	23.465	1.15698E-08	edata-la5	CS	13.496	8.71455E-07
	CS-BNG	1.982	0.08280		CS-BNG	1.197	**0.26552**
	CS-ILF	2.366	0.04555		CS-ILF	0.965	**0.36276**
	CHA	38.736	2.16744E-10		CHA	37.789	2.63954E-10
	MCA	18.805	6.60635E-08		MCA	25.873	5.34432E-09
edata-la2	CS	73.041	1.37505E-12	edata-la6	CS	39.804	1.74550E-10
	CS-BNG	39.179	1.98018E-10		CS-BNG	11.994	2.15235E-06
	CS-ILF	21.290	2.49111E-08		CS-ILF	6.337	0.00022
	CHA	97.054	1.41831E-13		CHA	133.905	1.08180E-14
	MCA	74.046	1.23295E-12		MCA	132.679	1.16434E-14
edata-la3	CS	131.109	1.28065E-14	edata-la7	CS	64.626	3.65552E-12
	CS-BNG	23.170	1.27823E-08		CS-BNG	6.818	0.00014
	CS-ILF	36.156	3.75183E-10		CS-ILF	7.230	8.98005E-05
	CHA	231.587	1.35290E-16		CHA	95.006	1.68205E-13
	MCA	130.255	1.34932E-14		MCA	81.851	5.53566E-13
edata-la4	CS	44.260	7.49525E-11	edata-la8	CS	57.522	9.26340E-12
	CS-BNG	20.171	3.81019E-08		CS-BNG	14.661	4.60020E-07
	CS-ILF	14.648	4.63250E-07		CS-ILF	6.058	0.00030
	CHA	54.677	1.38874E-11		CHA	137.617	8.69330E-15
	MCA	45.279	6.25117E-11		MCA	104.127	8.08257E-14
rdata-la1	CS	46.593	4.97624E-11	rdata-la5	CS	8.074	4.08754E-05
	CS-BNG	-3.386	0.00956		CS-BNG	−1.584	**0.15175**
	CS-ILF	-3.146	0.01369		CS-ILF	−1.460	**0.18245**
	CHA	4.441	0.00217		CHA	7.829	5.09974E-05
	MCA	31.532	1.11355E-09		MCA	20.961	2.81675E-08
rdata-la2	CS	14.019	6.50345E-07	rdata-la6	CS	16.055	2.27228E-07
	CS-BNG	1.609	**0.14633**		CS-BNG	4.677	0.00159
	CS-ILF	0.192	**0.85243**		CS-ILF	4.191	0.00303
	CHA	6.378	0.00021		CHA	9.994	8.52846E-06
	MCA	25.276	6.42756E-09		MCA	4.314	0.00257
rdata-la3	CS	9.365	1.38234E-05	rdata-la7	CS	25.866	5.35586E-09
	CS-BNG	−0.487	**0.63925**		CS-BNG	−7.338	8.08969E-05
	CS-ILF	−1.953	**0.08656**		CS-ILF	−11.057	3.99117E-06
	CHA	6.789	0.00014		CHA	61.686	5.30169E-12
	MCA	16.631	1.72711E-07		MCA	85.750	3.81619E-13
rdata-la4	CS	11.872	2.32641E-06	rdata-la8	CS	34.023	6.08574E-10
	CS-BNG	−0.527	**0.61264**		CS-BNG	−7.780	5.33524E-05
	CS-ILF	−0.192	**0.85272**		CS-ILF	−14.344	5.44727E-07
	CHA	8.752	2.27471E-05		CHA	57.098	9.82709E-12
	MCA	17.356	1.23754E-07		MCA	112.547	4.34061E-14
vdata-la1	CS	17.314	1.26166E-07	vdata-la5	CS	32.655	8.43151E-10
	CS-BNG	4.825	0.00131		CS-BNG	4.206	0.00297
	CS-ILF	4.851	0.00127		CS-ILF	5.145	0.00088
	CHA	10.120	7.76489E-06		CHA	7.075	0.00010
	MCA	7.542	6.66079E-05		MCA	8.051	4.17167E-05
vdata-la2	CS	17.408	1.20921E-07	vdata-la6	CS	11.565	2.83930E-06
	CS-BNG	4.513	0.00197		CS-BNG	4.736	0.00147
	CS-ILF	4.529	0.00193		CS-ILF	4.664	0.00161
	CHA	4.489	0.00203		CHA	8.006	4.34181E-05
	MCA	7.665	5.93336E-05		MCA	4.348	0.00245
vdata-la3	CS	14.976	3.90236E-07	vdata-la7	CS	17.341	1.24636E-07
	CS-BNG	5.267	0.00076		CS-BNG	4.316	0.00256

(Contd...)

TABLE VII
(Continued)

PI	AOOA VS.	t	Sig. (2-tailed) p-value	PI	AOOA VS.	t	Sig. (2-tailed) p-value
	CS-ILF	5.827	0.00039		CS-ILF	4.138	0.00326
	CHA	8.835	2.12313E-05		CHA	14.593	4.76762E-07
	MCA	8.721	2.33440E-05		MCA	5.555	0.00054
vdata-la4	CS	20.845	2.94166E-08	vdata-la8	CS	11.476	3.01076E-06
	CS-BNG	5.574	0.00053		CS-BNG	4.787	0.00138
	CS-ILF	5.513	0.00057		CS-ILF	5.662	0.00047
	CHA	18.180	8.61057E-08		CHA	14.528	4.93665E-07
	MCA	8.517	2.77472E-05		MCA	5.558	0.00054

TABLE VIII
Group-based Superiority of AOOA

Dataset	Number of Instances Where AOOA is Better				
	ELGA	EHGA	IGA	MOGWO	HICSA
MK1-10	8	8	8	8	7
	CS	CS-BNG	CS-ILF	CHA	MCA
edata	8	8	8	8	8
rdata	8	2	1	8	8
vdata	8	2	3	7	8
Total instances	24	12	12	23	24

vdata instances (la1, la3-la8). AOOA outperformed CS-BNG in two rdata instances (la2, la3) while overcame CS-ILF algorithm in la2. However, AOOA overcame CS-BNG in two vdata instances (la1, la8) and was equal to it in rank in la3 and la4, while overcame CS-ILF in three vdata (la1, la3, and la8) and was equal to it in rank in la2. Finally, it was found that AOOA achieved better results over CS, CS-BNG, CS-ILF, CHA, and MCA in solving 24, 12, 12, 23, and 24 FJSSP instances, respectively. Table VIII shows the AOOA superiority over competing algorithms for edata, rdata, and vdata datasets.

We believe that there are several factors behind the superiority of AOOA which can be summarized in (i) The concept of multiple populations generating a large diverse population, (ii) good exploration and exploitation mechanisms, (iii) two sources to generate new solutions represented in queen fertilization and worker lifecycle, and (iv) drone exchange, fading out, and swarming increase diversity and prevent stagnation.

Although AOOA presented a superior performance, it still has some limitations. AOOA used classical 34 instances of FJSS benchmark datasets while other benchmark datasets can be used such as (Kacem, Hammadi and Borne, 2002; Fattahi, Saidi Mehrabad and Jolai, 2007), etc. More objective functions and constraints can be considered such as maximizing the profit by resource utilization or machine performance, minimizing transportation time, process time, job delay, and energy consumption. Furthermore, enhancement can be made to AOOA so that better results are obtained.

VI. Conclusion

This paper introduced a solution to one of the NP-hard real-world problems, namely the FJSSP using AOOA, a new nature-inspired metaheuristic optimization algorithm. Its main structure was based on the organizational behavior of honey bees inside the apiary. Benchmark datasets of thirty-four instances of various sizes (small, medium, and large) scales were employed in the experiments to find the minimum makespan (C_{max}). The results were compared to ten of the state-of-the-art algorithms, five for BRdata and five for Hurink edata, rdata, and vdata. They were statistically assessed using the paired samples t-test and p-value, RPD, and group-based superiority statistical analysis to prove the AOOA performance. The p-values of AOOA at $\alpha = 0.05$ level (except for MK09 and edata02) indicate strong evidence against the null hypothesis proving that the results are significant. AOOA outperformed ELGA, EHGA, IGA, and MOGWO in solving all 10 MK instances and HICSA in solving 9 MK instances. Moreover, AOOA overcame CS, CS-BNG, CS-ILF, CHA, and MCA in solving 24, 12, 12, 23, and 24 instances of edata, rdata, and vdata, respectively. In many instances, AOOA achieved LB or near-optimal solutions to compete with other algorithms. The research demonstrated that the new nature-inspired algorithm can solve NP-hard problems effectively and efficiently. The overall findings were promising in comparison with some metaheuristic optimization algorithms.

References

Al-Obaidi, A.T.S., Abdullah, H.S., and Ahmed, Z.O., 2017. Camel herds algorithm: A new swarm intelligent algorithm to solve optimization problems. *International Journal on Perceptive and Cognitive Computing*, 3(1), pp.6–10.

Al-Obaidi, A.T.S., Abdullah, H.S., and Ahmed, Z.O., 2018. Meerkat clan algorithm: A new swarm intelligence algorithm. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(1), pp.354-360.

Al-Obaidi, A.T.S., and Hussein, S.A., 2016. Two improved Cuckoo search algorithms for solving the flexible job-shop scheduling problem. *International Journal on Perceptive and Cognitive Computing*, 2(2), pp.25-31.

Al-Sharqi, M.A., Al-Obaidi, A.T.S., and Al-mamory, S.O., 2024. Apiary organizational-based optimization algorithm: A new nature-inspired metaheuristic algorithm. *International Journal of Intelligent Engineering and Systems*, 17(3), pp.783-801.

Alzaqebah, M., Jawarneh, S., Alwohaibi, M., Alsmadi, M.K., Almarashdeh, I., and Mohammad, R.M.A., 2022. Hybrid brain storm optimization algorithm and late acceptance hill climbing to solve the flexible job-shop scheduling problem. *Journal of King Saud University -Computer and Information Sciences*, 34(6), pp.2926-2937.

Amirteimoori, A., Mahdavi, I., Solimanpur, M., Ali, S.S., and Tirkolaee, E.B.,

2022. A parallel hybrid PSO-GA algorithm for the flexible flow-shop scheduling with transportation. *Computers and Industrial Engineering*, 173, p.108672.

Bissoli, D.C., Altoe, W.A.S., Mauri, G.R., and Amaral, A.R.S., 2018. A Simulated Annealing Metaheuristic for the Bi-objective Flexible Job Shop Scheduling Problem. In: *2018 International Conference on Research in Intelligent and Computing in Engineering* (*RICE*). pp.1-6.

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by Tabu search. *Annals of Operations Research*, 41(3), pp.157-183.

Dai, M., Tang, D., Giret, A., and Salido, M.A., 2019. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics and Computer-Integrated Manufacturing*, 59, pp.143-157.

Fattahi, P., Saidi Mehrabad, M., and Jolai, F., 2007. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18(3), pp.331-342.

Hurink, J., Jurisch, B., and Thole, M., 1994. Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, 15(4), pp.205-215.

Jiang, M., Yu, H., and Chen, J., 2023. Improved self-learning genetic algorithm for solving flexible job shop scheduling. *Mathematics*, 11(22), p.4700.

Kacem, I., Hammadi, S., and Borne, P., 2002. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 32(1), pp.1-13.

Li, Y., Yang, Z., Wang, L., Tang, H., Sun, L., and Guo, S., 2022. A hybrid imperialist competitive algorithm for energy-efficient flexible job shop scheduling problem with variable-size sublots. *Computers and Industrial Engineering*, 172, p.108641.

Lu, C., Gao, L., Pan, Q., Li, X., and Zheng, J., 2019. A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution. *Applied Soft Computing Journal*, 75, pp.728-749.

Luo, S., Zhang, L., and Fan, Y., 2019a. Energy-efficient scheduling for multi-objective flexible job shops with variable processing speeds by grey wolf optimization. *Journal of Cleaner Production*, 234, pp.1365-1384.

Luo, S., Zhang, L., and Fan, Y., 2019b. Energy-efficient scheduling for multi-objective flexible job shops with variable processing speeds by grey wolf optimization. *Journal of Cleaner Production*, 234, pp.1365-1384.

Meng, L., Duan, P., Gao, K., Zhang, B., Zou, W., Han, Y., and Zhang, C., 2024. MIP modeling of energy-conscious FJSP and its extended problems: From simplicity to complexity. *Expert Systems with Applications*, 241, p.122594.

Piotr Jedrzejowicza, I.W., 2022. Implementation of the Mushroom Picking Framework for Solving Flexible Job Shop Scheduling Problems in Parallel. In: *26th International Conference on Knowledge-Based and Intelligent Information and Engineering 26th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems* (*KES 2022*). pp.292-298.

Rajwar, K., Deep, K., and Das, S., 2023. An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges. *Artificial Intelligence Review*, 56(11), pp.13187-13257.

Sadiq Al-Obaidi, A.T., Abdullah, H.S., and Ahmed, Z.O., 2018. Solving flexible job shop scheduling problem using meerkat clan algorithm. *Iraqi Journal of Science*, 59(2A), pp.754-761.

Şahman, M.A., 2021. A discrete spotted hyena optimizer for solving distributed job shop scheduling problems. *Applied Soft Computing*, 106, p.107349.

Song, L., Liu, C., Shi, H., and Zhu, J., 2022. An improved immune genetic algorithm for solving the flexible job shop scheduling problem with batch processing. *Wireless Communications and Mobile Computing*, 2022, p.2856056.

Wang, Y., Xia, T., Xu, Y., Ding, Y., Zheng, M., Pan, E., and Xi, L., 2024. Joint optimization of flexible job shop scheduling and preventive maintenance under high-frequency production switching. *International Journal of Production Economics*, 269, p.109163.

Wu, L., and Cai, H., 2021. Energy-efficient adaptive sensing scheduling in wireless sensor networks using Fibonacci tree optimization algorithm. *Sensors* (*Basel*), 21(15), p.55002.

Xie, J., Gao, L., Peng, K., Li, X., and Li, H., 2019. Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing*, 1, p.67-77.

Xie, N., and Chen, N., 2018. Flexible job shop scheduling problem with interval grey processing time. *Applied Soft Computing Journal*, 70, pp.513-524.

Zeidabadi, F.A., and Dehghani, M., 2022. POA: Puzzle optimization algorithm. *International Journal of Intelligent Engineering and Systems*, 15(1), pp.273-281.

Zhang, C., Tan, J., Peng, K., Gao, L., Shen, W., and Lian, K., 2021. A discrete whale swarm algorithm for hybrid flow-shop scheduling problem with limited buffers. *Robotics and Computer-Integrated Manufacturing*, 68, p.102081.

Zhang, G., Hu, Y., Sun, J., and Zhang, W., 2020. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. *Swarm and Evolutionary Computation*, 54, p.100664.

Zhu, N., Zhao, F., Wang, L., Ding, R., Xu, T., and Jonrinaldi, 2022. A discrete learning fruit fly algorithm based on knowledge for the distributed no-wait flow shop scheduling with due windows. *Expert Systems with Applications*, 198, p.116921.