Mathematical Modelling of Engineering Problems
Vol. 12, No. 3, March, 2025, pp. 1043-1052
Journal homepage: http://iieta.org/journals/mmep

# Discrete Fuzzy Multi-Objective Unrelated Parallel Machine Scheduling Problems: A Framework of Modified Artificial Fish Swarm Algorithm

Azhar Mahdi Ibadi[1,2*], Rosshairy Abd Rahman[1]

[1] Institute of Strategic Industrial Decision Modelling, School of Quantitative Sciences, Universiti Utara Malaysia, Sintok 06010, Malaysia
[2] Department of Physics, College of Science, University of Sumer, Al Rifaee 64005, Iraq

Corresponding Author Email: azhar_mahdi_ibadi@ahsgs.uum.edu.my

https://doi.org/10.18280/mmep.120331

**Received:** 8 October 2024
**Revised:** 20 December 2024
**Accepted:** 26 December 2024
**Available online:** 31 March 2025

***Keywords:***
*UPMSP, AFSA, fuzzy scheduling, metaheuristic, makespan optimization, tardiness*

## ABSTRACT

Unrelated Parallel Machines Scheduling Problem (UPMSP) is a significant discrete optimization problem employed in both industrial and non-industrial decision-making environments. In most real-world problems, the time parameters and constraint conditions often have imprecise information and cannot be quantified. Implementing fuzzy scheduling methodologies can improve schedule equality and reflect real-world scheduling complexity. Additionally, decision-makers regularly attempt to optimize multiple objectives simultaneously; thus, multi-objective optimization significantly influences scheduling performance and must be applied to establish tradeoffs among these objectives. Therefore, this paper aims to present a framework for developing a modified Artificial Fish Swarm Algorithm (AFSA) to solve multi-objective fuzzy UPMSP. This algorithm aims to overcome vital drawbacks of AFSA and improve it in three aspects: proposing new behavior, adapting improved visual and step parameters, and then employing a transformation method to make it more suitable for discrete UPMSP model. The evaluation process will involve a comparison of the proposed algorithm with standard AFSA and five modified AFSA algorithms from the literature, using three different sets of randomly generated problems that consist of various sizes of job and machine combinations. The findings are expected to effectively solve the proposed problem and identify optimal solutions, particularly for medium and large problems.

## 1. INTRODUCTION

Parallel Machine Scheduling Problem (PMSP) has received continuous attention among numerous types of manufacturing scheduling scholars and experts from scientific and engineering fields [1]. The increased interest in PMSP can be related to their widespread use in modern industrial systems such as flexible manufacturing systems [2] and plastic moulding industries [3], etc. Durasević and Jakobovi [4] defined the UPMSP as a problem model with many components. These components represent $N$ jobs that need to be distributed across $M$ machines with varying capacities. Consequently, the assigned jobs are consistently managed at varying rates. To address this issue, two necessary steps are involved: firstly, assigning jobs to machines and secondly, placing orders for them. UPMSP is beneficial in both theoretical and practical aspects. Theoretically, they can be regarded as an extension of the single machine problem and are the general form of other parallel machine problems that are identical and uniform. Due to the varying machine speeds, technologies, and machine types in production shops across many industries, production scheduling and manufacturing systems hold high practical significance in the present world.

The use of exact methods is limited in comparison to other algorithms because obtaining optimal results in an acceptable time is extremely costly. Swarm intelligence algorithms outperform evolutionary and single-solution algorithms and have interesting methodologies because they are flexible, capable of searching a larger search space, exploring, and providing opportunities for improvement [5]. Swarm intelligence algorithms fall under the broader category of metaheuristics, which are often classified based on their natural inspiration. Swarm intelligence is a set of artificial intelligence methods inspired by the collective behaviors of social insects, animals, and humans and provides numerous fundamental advantages for solving complex problems on a large scale. Until now researchers are still developing new swarm algorithms, such as Tuna Swarm Optimization Algorithm [6], and the Eagle Search Algorithm [7].

Compared to other swarm intelligence algorithms, the AFSA has fewer parameters to manage, making it easier to implement. The computational time required to develop satisfactory solutions is reasonable and it has parallelism, robust global search ability, rapid converging, and lower sensitivity to the demands of the objective functions [8]. These advantages have led to the successful application of AFSA in a variety of optimization problems based on advanced literature [9, 10]. Nevertheless, the algorithm has multiple drawbacks including inefficient convergence, local optimum trapping and weakness in the later optimization phases. In recent years, scholars have suggested several strategies to address the weaknesses of AFSA. For instance, Jin et al. [11] proposed a modified AFSA for unit commitment optimization in power systems to overcome the disadvantages of premature convergence and local extremes in the original algorithm. The improvements include a variable vision, adjusting the movement strategy, and combining the mutation operation of genetic algorithms. Wang et al. [12] proposed a modified AFSA in combination with a local path optimizer for the path planning problem for unmanned surface vehicles and customize four operators and an adaptive factor to improve the converging performance of the algorithm. Thus, until now the algorithm continues to be developed.

However, the NP-hardness of UPMSP and multi-objective adds the dimension of complexity to the problem, making it difficult to obtain optimal solutions, especially for large problems with uncertain time parameters. As a result, developing efficient algorithms is a challenge that adds to the complexity of the problem. Therefore, this paper presents a framework for the proposed modified AFSA that solves fuzzy multi-objective UPMSP. The proposed modified AFSA is a new algorithm developed to solve UPMSP by leveraging the strengths of the AFSA which include multi-stage solution updates and exploration of the search domain. The framework focuses on three modifications that improve convergence and optimality which encompasses three aspects: introducing a new behavior to the four primary behaviors by employing the best solution derived from AFSA, adapting improved visual and step parameters, which achieves a balance between global search capability and convergence speed, effectively applying the continuous AFSA to discrete machine scheduling problems and make it appropriate for combinatorial optimization problems such as UPMSP.

## 2. RELATED WORKS

Both industry and academia face the fundamental issue of effectively scheduling and sequencing jobs on machines. Extensive literature [4] suggests various approaches for solving UPMSP, ranging from exact methods and simple heuristics to complex hybrid metaheuristics. As a result, numerous research studies on scheduling issues have been done in recent years. Safarzadeh and Niaki [13] used a mathematical programming approach and the ϵ-constraint method to optimize the multi-objective makespan and the total cost, and their experimental findings validated the effectiveness of this approach. A study by Zheng et al. [14] created a Mixed Integer Linear Programming (MILP) model that solves small-scale problems more efficiently by lowering the weighted average of the total order completion time and the total processing cost of UPMSP machines. The researchers established two heuristic algorithms, type-based greedy and enhanced differential evolution, and demonstrated their efficiency and effectiveness in large-scale applications through computational experiments. Shafipour et al. [15] looked at the UPMSP using Simulated Annealing (SA) and the Artificial Immune System (AIS). They wanted to reduce total early and late costs while keeping different constraints in mind. The results show that SA using the repair strategy outperforms the other proposed methods. Åblad et al. [16] created and analyzed a method for exact solutions based on the Branch and Bound (BAB) method of the UPMSP to minimize the makespan. The suggested approach outperforms several MILP formulations and local search heuristics on various types of instances.

Arnaout [17] presented a Worm Optimization Algorithm (WO) for minimizing the makespan on the UPMSP with sequence-dependent setup times. The WO was compared to existing algorithms (TS, ACO, GA, SA), and the results of the tests demonstrated WO's superiority. Jouhari et al. [18] presented a new hybrid metaheuristic combining the Harris Hawk Optimizer (HHO) and Salp Swarm Algorithm (SSA) to solve UPMSPs to minimize the makespan. The proposed algorithm outperforms both SSA and HHO and other methods in terms of convergence to the optimal solution. Fanjul-Peyro et al. [19] suggested a new exact method that combines some improvements and a new procedure with the proposed MILP models of the UPMSP to minimize makespan with setup times. The results obtained prove that the suggested technique significantly improves the existing approaches and can obtain solutions for extremely large instances.

Due to the complexity of UPMSP, exact algorithms for the problem have only been proven effective for small to medium instances and until recently, real-world instances of the problem were not solved using these specific algorithms. When it comes to solving large-scale instances of the UPMSP, heuristic and metaheuristic approaches have usually been more effective than the exact ones. Much of the existing literature is concerned with heuristics and metaheuristics since they are simple to use, reproduce, and adaptable, thus these algorithms seem more appropriate. Despite the number of literatures dealing with UPMSPs, it is not surprising that many of the developed solution methods are based on modified and hybrid metaheuristic algorithms.

Chi et al. [20] enhanced the Iterative Greedy Algorithm (IG) with Tabu Search (TS) and provided an IG-TS algorithm for solving UPMSPs with makespan minimization. In their study, three commonly hybrid algorithms were used in scheduling, that is the GA-TS, ABC-TS and PSO-TS algorithms, were used as a comparison. As seen in the results, the scheduling decisions made using the IG-TS algorithm were the most efficient and reliable. Al-Qaness et al. [21] presented a method for minimizing the makespan for UPMSP with sequence-dependent setup times that combines the properties of two swarm intelligence algorithm, the Whale Optimization Algorithm (WOA) and the Firefly Algorithm (FA). Results show that the suggested method outperforms eight well-known metaheuristic optimization techniques.

Ewees et al. [22] suggested a hybrid algorithm that employs the operators of the Firefly Algorithm (FA) to enhance the SSA's exploitation ability while also providing a local search mechanism to improve the superiority of the solution to minimize the makespan and improve searchability for UPMSPs. Extensive comparisons show that the suggested approach outperforms existing metaheuristics. Meng et al. [23] proposed a hybrid GA with a variable neighborhood structure strategy to solve the multi-objective problems of total makespan and total tardiness for identical and UPMSPs. In comparison, the suggested method performs statistically better than the non-dominated sorting Genetic Algorithms II and III. Ding et al. [24] suggested a memetic method for makespan minimization that combines an efficient local search technique with an evolutionary approach of UPMSP.

Most recent UPMSP studies have used deterministic models with precise problem parameters. In many real-world situations with different degrees of uncertainty, it's not an acceptable assumption. Industry and academia have paid a lot of attention to fuzzy scheduling and many different objectives have been extensively studied. For instance, A fuzzy-based genetic algorithm has been developed by Yaghtin and Javid [25]. It could solve large instances and demonstrate good performance in terms of quality and run time factors to minimize total job tardiness, where process times and due dates are triangular fuzzy numbers (TFNs). Rezaeian et al. [26] established a fuzzy mathematical programming model to minimize the overall fuzzy weighted earliness and tardiness penalties UPMSP, where setup and processing times are TFNs and trapezoidal fuzzy numbers (TrFNs) for due dates. They proposed two algorithms: GA and modified SA to address such problems. Comparative results demonstrated that the modified SA algorithm outperformed both the GA and several other algorithms in solving the proposed problem. Cheng et al. [27] introduced a Random Forest-based metaheuristic for minimizing the makespan in UPMSP with uncertain setup times in the forging industry. Numerical experiments demonstrate a significant reduction in the error percentage for setup time estimation when the proposed method is implemented. This reduction is especially meaningful when dealing with large-scale problems. Sadati et al. [28] proposed two exact approaches and two metaheuristics to minimize fuzzy multi-objective model makespan and maximum tardiness for UPMSP, with each job determined by TrFNs of processing time-dependent machine, ready time and due date.

Liao and Su [29] created hybrid Ant Colony Optimization (ACO) for the fuzzy UPMSP meanwhile TrFNs was used to indicate processing time, release time, and setup time. Fuzzy makespan and other fuzzy numbers were ranked using a fuzzy ranking method. In comparison to a hybrid Particle Swarm Optimization (PSO) and two variants of SA algorithms, the proposed hybrid performs better. Manupati et al. [30] utilized TFNs to represent processing times and due dates to create efficient solutions for the UPMSP model in fuzzy multi-objective systems. The goal was to minimize the makespan, tardiness, flow time, and machine-load variation. The authors employed two metaheuristics for the purpose of achieving both high-quality and precise solutions. Naderi-Beni et al. [31] presented two metaheuristic techniques for solving a fuzzy multi-objective UPMSP where TrFNs are represented for processing times, release dates, setup times, and due dates to minimize workload imbalance, and overall tardiness at the same time. Torabi et al. [32] developed an efficient PSO algorithm for UPMSP by proposing a fuzzy programming model that reduces the sum of weighted tardiness and flow times, as well as the variance in machine load when processing times and due dates are TFNs. Test results show that the suggested algorithm surpasses the conventional method in all three-performance metrics.

Various machine environments with different objectives utilize AFSA in many scheduling problems. For instance, Li et al. [33] proposed an improved AFSA, formulating it as a multidimensional knapsack problem to find the optimal scheduling and minimize the time delay. Meanwhile, Sureja and Patel [34] explored a novel algorithm, inspired by the social behavior of food-seeking fish colonies, and applied it to the combinatorial problem known as the random travelling salesman problem. Other research by Krishnaveni and Janita [35] proposed modifying AFSA by incorporating oppositional behavior learning into standard AFSA, resulting in a reverse population that not only improved task scheduling but also significantly reduced multi-objective makespan and execution costs. Fortunately, the proposed algorithm outperforms PSO and GA algorithms. Similarly, Peng et al. [36] proposed a modified AFSA that integrates a random search strategy called Levy flight to enhance AFSA's preying behavior and firefly behavior. The algorithm incorporates the firefly algorithm's movement strategy to adjust random movement after determining the direction in AFSA. The proposed approach outperforms the tested algorithms in convergence speed and optimization accuracy on several benchmark problems.

Another research using AFSA was conducted by Sun et al. [37] utilizing a hybrid AFSA with Self-adaptive Differential Evolution (SADE) to solve the identical PMSP for minimizing the makespan. The hybrid model performed better than the three other algorithms in both efficiency and strength tests. Meanwhile, Tirkolaee et al. [38] also proposed a hybrid approach based on an interactive fuzzy solution technique and AFSA to reduce both total cost and total energy consumption in the flow shop scheduling problem. Finally, Zhu and Jiang [39] proposed a hybrid AFSA by incorporating TS strategy into the AFSA to prevent artificial fish from becoming stuck in the local optimum and to accelerate convergence for the job shop scheduling problem to reduce the makespan. These modifications attempted to improve the algorithm's performance in terms of convergence, optimality, and escape from local optimality. Nevertheless, the algorithm remains under development. Despite numerous enhancements and modifications to the fish swarm algorithm being regularly proposed, these efforts have predominantly concentrated on equilibrating exploration and exploitation processes, lacking comprehensive guidance for achieving accurate and precise directions [40].

## 3. METHODOLOGY

Constructing a framework for solving UPMSP involves five phases of the research methodology as illustrated in Figure 1.

1) Phase 1: Conduct relevant literature review:

Reviewing previous research and frameworks is an important phase in framework development, including an in-depth review of previously published works and established frameworks relevant to the subject. This phase comprehensively covered the existing literature on the use of deterministic and uncertain models to solve single and multi-objective functions in UPMSPs, followed by many related works of literature that use the AFSA in several machine scheduling problems.

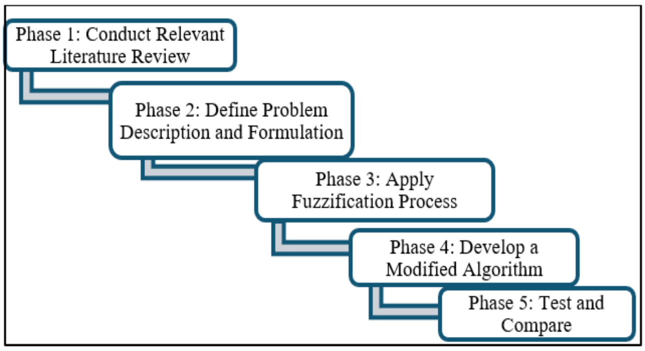
Phase 1: Conduct Relevant Literature Review
Phase 2: Define Problem Description and Formulation
Phase 3: Apply Fuzzification Process
Phase 4: Develop a Modified Algorithm
Phase 5: Test and Compare

**Figure 1.** Phases of the conceptual framework

2) Phase 2: Define problem description and formulation:

An optimization technique has been employed to maximize or minimize a singular objective. A common challenge faced by decision-makers is the need to optimize multiple objectives simultaneously, which results in multi-objective optimization. A solution may be optimal for one objective function but not for another, which makes it often vague to define what an optimal solution is [41]. In recent years, much attention has been focused on multi-objective optimization, with nearly half of the publications, particularly in 2019 and 2020, focusing on multi-objectives [42], which must be used in practice to determine a trade-off between them when making scheduling decisions. This phase aims to adapt and extend the linear multi-objective model proposed by Kongsri and Buddhakulsomsiri [43] for UPMSP by minimizing the makespan and total tardiness. This step will be done by simplifying the mathematical model and reducing sequence dependent setup times constraints with the purpose of making them more flexible and expandable.

3) Phase 3: Apply fuzzification process:

The majority of UPMSP work is conducted using the deterministic optimization methodology, where each parameter is assumed to be well-defined. While this assumption may be correct in some situations, it is often incorrect in practical implementations. There is a widespread assumption among researchers regarding the presence of predictable conditions for characteristics such as task processing times, release dates and due dates. However, in addition to the investigations conducted in a deterministic setting, the various parameters associated with UPMSP are often not well-known, resulting in uncertainties in real-world scenarios. Thus, this phase identifies and describes the fuzzification process of the UPMSP research problem model by embedding TFNs for the main parameters processing times and due dates in the proposed mathematical model to make it more realistic. This helps cover unexpected occurrences that are common in practice, such as errors in the estimation of values parameters, the changing weather and the health of the workforce, machine breakdown and human factors making and improving scheduling efficiency and achieving satisfactory results within the constraints of time, cost, and quality. Jobs' processing times and due dates are considered here along with TFNs to deal with the unpredictable nature of parameters in practical settings. The first fuzzy number indicated processing times $\tilde{p}_{ij} = ({p^1}_{ij}, {p^2}_{ij}, {p^3}_{ij})$, where ${p^1}_{ij}$ is the shortest processing time, ${p^2}_{ij}$ is the most possible processing time and ${p^3}_{ij}$ is the longest processing time. To represent job processing times, the following triangular fuzzy membership function is used:

$$\mu_{\tilde{p}_{ij}}(x) = \begin{cases} 0 & for\ x < {p^1}_{ij} \\ \dfrac{x - {p^1}_{ij}}{{p^2}_{ij} - {p^1}_{ij}} & for\ {p^1}_{ij} \le x \le {p^2}_{ij} \\ \dfrac{{p^3}_{ij} - x}{{p^3}_{ij} - {p^2}_{ij}} & for\ {p^2}_{ij} \le x \le {p^3}_{ij} \\ 0 & for\ x > {p^3}_{ij} \end{cases} \tag{1}$$

The second fuzzy number indicated the due dates $\tilde{d}_j = ({d^1}_j, {d^2}_j, {d^3}_j)$, where, ${d^1}_j$ is the earliest due date, ${d^2}_j$ is the most possible due date and ${d^3}_j$ is the latest due date. The following triangular fuzzy membership function is used to represent job due dates:

$$\mu_{\tilde{d}_j}(x) = \begin{cases} 0 & \text{for } x < {d^1}_j \\ \dfrac{x - {d^1}_j}{{d^2}_j - {d^1}_j} & \text{for } {d^1}_j \le x \le {d^2}_j \\ \dfrac{{d^3}_j - x}{{d^3}_j - {d^2}_j} & \text{for } {d^2}_j \le x \le {d^3}_j \\ 0 & \text{for } x > {d^3}_j \end{cases} \tag{2}$$

The fuzzy environment has an impact on the objective functions because the fuzzy completion time of each job will be a TFNs due to the triangular form of the processing times and is defined as a $\tilde{C}_{ij} = ({C^1}_{ij}, {C^2}_{ij}, {C^3}_{ij})$, the tardiness of the job $j$ on machine $i$ equals to the $\tilde{T}_{ij} = ({T^1}_{ij}, {T^2}_{ij}, {T^3}_{ij})$ where $\tilde{T}_{ij} = max(\tilde{C}_{ij} - \tilde{d}_j,\ 0)$, where, the zero fuzzy number =(0, 0, 0). The problem can be mathematically presented as follows:

$$MinF = w\widetilde{C_{max}} + (1 - w)\sum_{j=1}^{n}\sum_{i=1}^{m}\tilde{T}_{ij} \tag{3}$$

$$\tilde{C}_0 = 0 \tag{4}$$

$$\tilde{C}_{ij} - \tilde{C}_{ik} + v(1 - X_{ikj}) \ge \tilde{p}_{ik} \quad \forall i \in M, \forall k \in N_0, \forall j \in N: j \ne k \tag{5}$$

$$\widetilde{C_{max}} \ge \tilde{C}_j \quad \forall j \in N \tag{6}$$

$$\tilde{T}_{ij} = max(0, \tilde{C}_{ij} - \tilde{d}_j) \quad \forall i \in M, \forall j \in N \tag{7}$$

$$\tilde{P}_{ij}, \quad \tilde{C}_{ij}, \quad \tilde{T}_{ij}, \quad \widetilde{C_{max}} \ge 0 \tag{8}$$

$$X_{ikj} \in \{0,1\} \tag{9}$$

Eq. (3) represents the multi-objective functions to reduce the makespan and total tardiness. Constraint (4) sets the dummy job's completion time to zero. Constraint (5) guarantees the completion time of job $k$ equals the previous job's completion time plus the processing time of job k at each machine. Constraint (6) indicates that the makespan is equivalent to the maximum completion time among all machines, Constraint (7) determines the tardiness value for every job. Constraint (8) indicates that all decision variables must assume non-negative values. Constraint (9) clarifies the binary variables.

4) Phase 4: Develop a modified algorithm:

AFSA, described by Li [44] has been effectively applied to solve several optimization problems. Because of its simple principle and excellent features such as robustness and parameter setting tolerance [36], AFSA has grown in importance in swarm intelligence optimization. It is an iterative algorithm that progressively converges to the optimum global solutions. The basic idea is to mimic different environmental behaviors of schooling fish in the water by displaying social search behaviors and moving to regions with greater food availability, where a fish represents a point or fictitious entity of a true fish in a population and swarm movements occur at random [45].

In AFSA, all artificial fish (AF) are trained to react to real-time situations. Each AF learns how to execute four types of basic behaviors: prey, swarm, follow, and random. Figure 2 shows the flowchart of the standard AFSA algorithm.

• Prey behavior is the basis for the algorithm's convergence,

• Swarm behavior enhances the algorithm's global convergence and stability,

• Follow behavior accelerates the algorithm's convergence,

• Random behavior balances out the conflict between the other three behaviors.

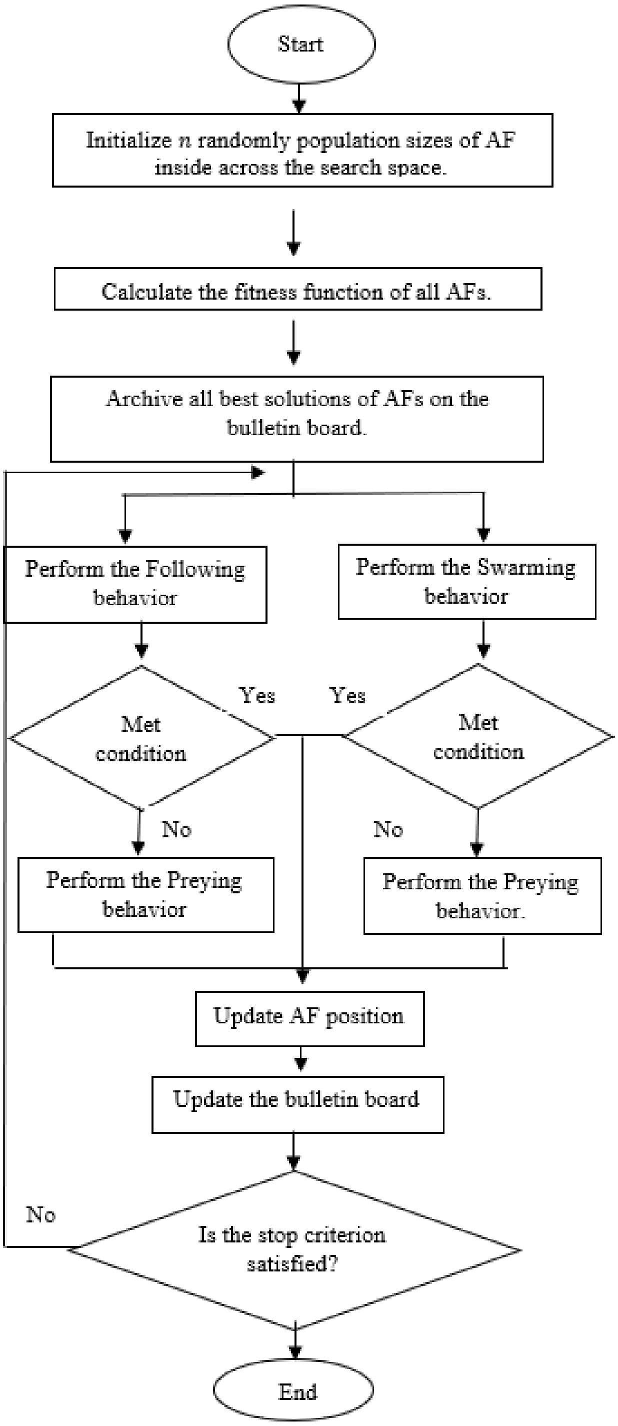
Start
Initialize $n$ randomly population sizes of AF inside across the search space.
Calculate the fitness function of all AFs.
Archive all best solutions of AFs on the bulletin board.
Perform the Following behavior
Perform the Swarming behavior
Met condition
Yes
Yes
Met condition
No
No
Perform the Preying behavior
Perform the Preying behavior.
Update AF position
Update the bulletin board
No
Is the stop criterion satisfied?
End

**Figure 2.** Flowchart of standard AFSA

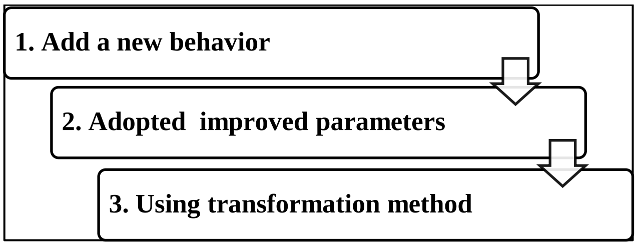
1. Add a new behavior
2. Adopted improved parameters
3. Using transformation method

**Figure 3.** Framework for a proposed modified algorithm

AF behaves in response to current conditions and environmental conditions, and it has an impact on the environment through its own and other AF's activities. Each AF receives two input parameters: visual (perception) and step (moving step length). AFs will collect information and then move using visual and step guidance. AFs' behaviors tend to affect each other's fitness as well.

We propose employing a three-stage framework that effectively modifies the AFSA algorithm. The proposed modified algorithm allows for increasing the effectiveness of traditional AFSA behaviors to make it suitable to the proposed problem. As illustrated in Figure 3, the framework can be divided into three stages: adding new behavior, adopting improved parameters, and using the transformation method. Each stage includes distinct activities as outlined below:

- **Add a new behavior**

Creating aspiration behavior depends on the best solution reached by AFs of all AFSA behaviors and adopted its position to compare with the current position in order to enhance the search process and help AFs to escape from the local optimums.

The update processes are formulated as: Suppose that $X_{best}$ represents the potential position of AF in the step range and $\| X_{best} - X_i \| \leq Step_2$, which can be stated as:

$$X_{Aspiration} = X_i + \frac{X_{best} - X_i}{\| X_{best} - X_i \|} \times Step_2 \times R(0,1) \qquad (10)$$

The aspiration behavior guarantees that the computing time is greatly reduced because AF can determine the best position in just one cycle of calculations. The following illustrates the pseudocode of the proposed aspiration behavior.

Pseudocode of the Aspiration behavior

```
Adapt the best solution found ever on the bulletin board
Generate and calculate food concentration at the new position
  within improved visual parameter
  Move along a particular direction based on Eq. (10)
  within improved step parameter to select the best optimal
     solution and update the current position of AF
     If the solution value gets no improvement
        Stay in the best position, and the best solution is the
          final optimal solution
     Else
     Move to a new position and consider it global
          optimal solution
    Update the current best solution of AF in the bulletin
          board.
```

- **Adopted improved parameters**

The visual and step parameters in the standard AFSA are fixed through iterative processes. Large visual and step values at the beginning of the algorithm speed up its convergence. However, the large values will result in problems like local optimum or iterative jumps when AFs approach the final position. However, if the values are too low, the effectiveness decreases [46]. As a result, the algorithm has specific requirements about the visual and step at different stages. Therefore, to achieve a balance between convergence rate and global search capability, the algorithm has adopted improved parameters as described in Tan and Mohamad-Saleh [40], and can presented as follows:

$$visual^{t+1} = visual^{t} - visual^{t} \times \lambda + visual_{min} \quad (11)$$

$$step^{t+1} = step^{t} - step^{t} \times \lambda + step_{min} \quad (12)$$

$$\lambda = e^{\left(-\frac{\sigma}{\sqrt[4]{(t_{max})^3}} * (t_{max} - t)\right)} \quad (13)$$

where, $visual_{min}$ is the minimum visual value; $step_{min}$ is the minimum step value; $t$=(1, 2, …, $t_{max}$) is the index number of iterations; $0.5 < \sigma < 1$ at any stage. Therefore, it improves the algorithm's local search at a later stage.

- **Using transformation method**

AFSA was initially developed as a continuous algorithm for solving continuous problems, making it challenging to deal with combinatorial optimization problems like UPMSP that have a discrete solution space. Hence, in the optimization algorithm, the process of solution encoding is critical. It is essential to enable the AFSA algorithm to solve discrete UPMSPs and enhance the search capabilities of the original algorithm. The major stage of the proposed algorithm is the use of the transformation method to transform the continuous search space of AFSA into a discrete search space to make it appropriate for discrete optimization problems such as UPMSP. The following illustrates the pseudocode for the transformation method.

Pseudocode of the transformation method

```
function [a]=trans(q)
   n=length(q);
   s=random('uniform',0.1,0.2,1,1);
for i=1: n

   a(q(i))=s;
   s=s+random('uniform',0.1,0.2,1,1);
end
a=a./norm(a);
```

This code uses a trans function to transform the input vector *q,* which represents the continuous solution, into the output vector *a,* representing the discrete solution. It starts by:

Step 1: Determine the length of *q* to find the number of elements in *q* and store it in *n*.

Step 2: Initialize *s* with a random value generated from a uniform distribution in the range [0.1, 0.2]. This value will serve as the starting point for the assignment process.

Step 3: Loop through each element of *q*. For every element in *q,* perform the following steps inside the loop:

- *a*(*q*(*i*))=*s*;: The current value of *s* would be assigned to the position in *a* for which *q*(*i*) is specified. This step maps the continuous solution to a discrete position.
- *s*=*s*+random('uniform', 0.1, 0.2, 1, 1);: A new random number from the same uniform distribution is added to update *s*. By doing this, the solution's randomness is maintained, and every next assignment in *a* is based on a new random value.

Step 4: Normalize the vector *a* to ensure its Euclidean norm (length) equals one. This step checks that the final discrete solution is appropriately scaled.

This method is used to compute the best solution in the population according to the value of multi objective function. Subsequently, the pseudocode of the proposed modified AFSA algorithm is presented as below, while Figure 4 represents the flowchart. The yellow color represents the modification steps that have been made in the proposed modified version of AFSA.

Pseudocode of the proposed modified AFSA algorithm:

```
Input: npop number of fish in the swarm
      MaxIt maximum number of iterations for optimization
       Visual the visual range
       Step the maximum step length that a fish can take at
each movement
       Try- number the number of attempts in preying
behavior, μ is the crowding factor
Input: Optimal fish position and its fitness
Initialize a population of npop fish' positions at random;
While the stopping condition is not satisfied
   for each AFs do
   compute the visual range
    if visual range is empty, then
         f ($X_i$) ◄———— random ($X_i$)
     else
       if visual range is crowded, then
         f ($X_i$) ◄———— follow ($X$)
         else
           calculate swarm center $x_{center}$, if fitness of $X_{center}$
             (f ($X_{center}$)) better than fitness of AFs, then
          f($X_i^1$) ◄———— swarm ($X_i$)
          else
           f($X_i^1$) ◄———— follow ($X_i$)
          end
           if fitness of AFs  worse than global best then
              f($X_i^2$) ◄———— pray ($X_i$)
           else
              f($X_i^2$) ◄———— follow ($X_i$)
           end
            f ($X_i$)=argmin fitness (f($X_i^1$), f($X_i^2$))
        end
    end
       for each AFs do
        $X_i$ ◄———— select ($X_i$, f ($X_i$))
        if $X_{best}$ is the best postion found ever in the globel
           best for t iterations then
            f ($X_{best}$ ) ◄———— aspiration ($X_{best}$)
        else
         randomly choose a fish $X_i$ and f ($X_i$)◄—— random ($X_i$)
        end
end
```

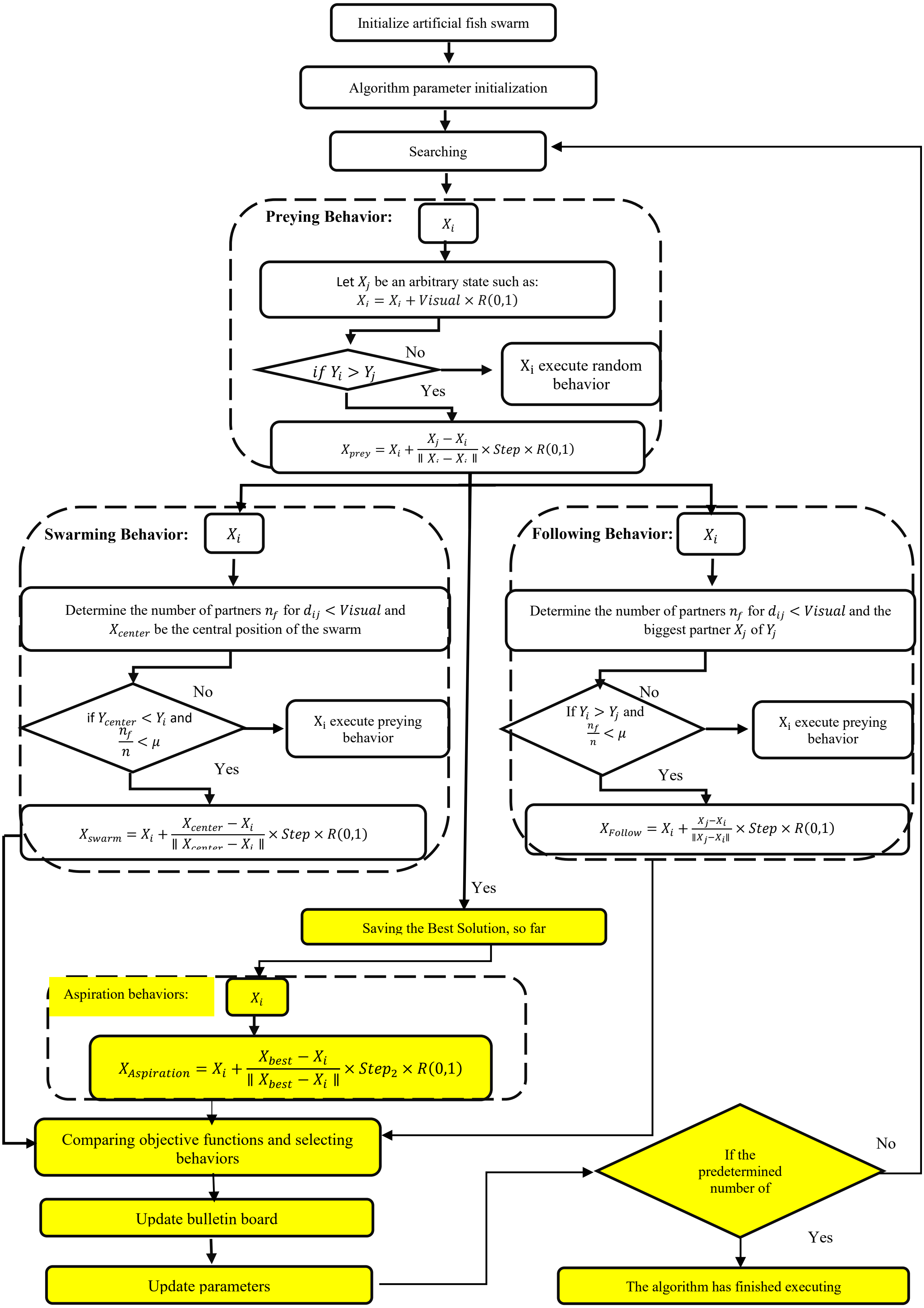
Initialize artificial fish swarm
Algorithm parameter initialization
Searching
Preying Behavior:
$X_i$
Let $X_j$ be an arbitrary state such as:
$X_i = X_i + Visual \times R(0,1)$
$if\ Y_i > Y_j$
No
Yes
$\mathrm{X_i}$ execute random behavior
$X_{prey} = X_i + \frac{X_j - X_i}{\| X_j - X_i \|} \times Step \times R(0,1)$
Swarming Behavior:
$X_i$
Determine the number of partners $n_f$ for $d_{ij} < Visual$ and $X_{center}$ be the central position of the swarm
if $Y_{center} < Y_i$ and $\frac{n_f}{n} < \mu$
No
Yes
$\mathrm{X_i}$ execute preying behavior
$X_{swarm} = X_i + \frac{X_{center} - X_i}{\| X_{center} - X_i \|} \times Step \times R(0,1)$
Following Behavior:
$X_i$
Determine the number of partners $n_f$ for $d_{ij} < Visual$ and the biggest partner $X_j$ of $Y_j$
If $Y_i > Y_j$ and $\frac{n_f}{n} < \mu$
No
Yes
$\mathrm{X_i}$ execute preying behavior
$X_{Follow} = X_i + \frac{X_j - X_i}{\|X_j - X_i\|} \times Step \times R(0,1)$
Yes
Saving the Best Solution, so far
Aspiration behaviors:
$X_i$
$X_{Aspiration} = X_i + \frac{X_{best} - X_i}{\| X_{best} - X_i \|} \times Step_2 \times R(0,1)$
Comparing objective functions and selecting behaviors
Update bulletin board
Update parameters
If the predetermined number of
No
Yes
The algorithm has finished executing

**Figure 4.** Flowchart of the proposed modified AFSA algorithm

The proposed modified AFSA algorithm outlined in Figure 4 initiates by setting up the population size of AFs along with the parameters for both the AFSA and the proposed algorithm. After each iteration, each AF is evaluated individually, and the best fitness is recorded on the bulletin board by employing preying, swarming, following, and random behaviors simultaneously. Current and environmental factors determine AF movement, which affects the environment through its and other artificial fish's actions. The modification steps that have been made in AFSA started by executing aspiration behavior by selecting the best possible solution after implementing these behaviors and improving the search process by comparing the aspiration state with the current state of the fish. The proposed algorithm employs preying behavior when the conditions for executing behaviors are unmet or when no improvement is seen. Furthermore, the AFSA's fixed value will be substituted with improved visual and step parameters, and the performance will be revised according to the best and contemporary individual employing aspiration behavior expression. The fitness value of the best individual is then determined through a transformation method that converts the continuous solution into a discrete solution. All solutions will be updated in accordance with previous steps and executed until the termination criterion is met. The three proposed modification steps, which are related and rely on the quality of the solution, will be performed and run simultaneously to optimize the entire process, which focuses on the primary shortcomings of the standard AFSA.

5) Phase 5: Test and compare:

The crucial phase in the framework development process is the test and compare phase, which involves a practical assessment of the proposed algorithm to confirm the algorithm's efficiency in UPMSP environment.

Throughout this phase, the framework is subjected to simulations and computational experiments using MATLAB R2023a. These experiments are conducted randomly and involve three different problem sizes: small, medium, and large sets of jobs and machine combinations. Each set of problems consists of 10 instances and will be run 10 times. Small size problems will consist of the number of jobs N=[5, 10, 15, 20, 25, 30, 35, 40, 45, 50], and the number of machines M =[3, 3, 4, 4, 5, 5, 6, 6, 7, 7]. Medium size problems instances, N=[60, 80, 100, 120, 140, 160, 180, 200, 220, 240], and M=[8, 8, 9, 9, 10, 10, 11, 11, 12, 12], while large size problems, N=[250, 270, 290, 310, 330, 350, 370, 390, 410, 430] and M=[13, 13, 14, 14, 15, 15, 16, 16, 17, 17]. The performance evaluation criteria for the whole computational experiments included minimum, maximum, mean, and standard deviation. The best performance metrics of the proposed algorithm on the test problems will be defined by the minimum values in the proposed minimization problem.

The performance of the proposed modified algorithm will be examined by comparing it with other versions of AFSA. The algorithms include standard AFSA [44], and five modified versions of AFSA which are presented in previous works by Tan and Mohamad-Saleh [40], Zhao et al. [46], and three methods from Huang et al. [47]. The comparison process will be employed based on the standardized predefined parameter settings to ensure the fairness of comparisons. These comparisons will be employed to evaluate the effectiveness of the proposed algorithm and identify any possible weaknesses in certain cases. Instead of comparing based solely on minimum values, the Wilcoxon signed-ranks test will also be conducted among all compared algorithms to assess significant differences.

## 4. THEORETICAL IMPLICATIONS OF THE PROPOSED ALGORITHM

The proposed algorithm has main theoretical implications to increase the effectiveness of traditional AFSA and make it suitable to solve the proposed problem. Firstly, the aspiration behavior aims to overcome the random searches of AF in its visual range for the next state before moving to the next position, which means AF must attempt every direction until the position is found and this position generated may not be optimal since the process is random. This has caused a drawback in the AFSA’s behavior and consumes a lot of time. Aspiration behavior is inspired by the best position of AFs that has been found recorded on the bulletin board after performing all AFSA behaviors. Secondly, in the AFSA, the AF seeks a local solution within its visual range, while the global optimal solution is achieved through the collaboration of the fish swarm. Therefore, the balance between local and local search is controlled by visual and step [10]. For reaching a balance between convergence rate and global search capability, the algorithm improves its local search in later stages through specific equations for visual and step parameters, which gradually decrease throughout the iterations before adjusting at a value that approximately corresponds to $visual_{min}$ and $step_{min}$, will therefore be incapable of performing any further local searches.

Finally, another theoretical importance is to be capable of making the algorithm solve the discrete machine scheduling models by converting solutions from the continuous vectors to discrete solutions space.

## 5. CONCLUSION

This paper proposes a framework for the development of a modified algorithm of AFSA aimed at addressing the multi-objective makespan and total tardiness in the UPMSP within a fuzzy environment, by using triangular fuzzy membership functions for the main parameters of processing times and due dates. The research framework is structured into five distinct phases: Conduct relevant literature review, adapt and extend a mathematical model, apply fuzzification process, modify the AFSA algorithm, and the evaluation process. The performance of the proposed modified algorithm will be evaluated using evaluation criteria that include Minimum, Maximum, Mean, Standard Deviation, and the Wilcoxon signed-rank test statistic. The proposed algorithm attempts to improve the exploitation capability of the AFSA, thereby further stabilizing the overall performance of the algorithm. The expected outcomes will involve efficient problem-solving and the determination of optimal solutions when evaluated against comparative algorithm results, especially for medium and large-size problems. Furthermore, for any possible shortcomings or unsatisfactory performance due to falling into a local optimum, future investigations could explore a hybrid approach by combining the proposed modified algorithm with other metaheuristic algorithms to enhance the performance and achieve more precise solutions.

## ACKNOWLEDGMENT

This work was a part of PhD study in the School of Quantitative Sciences, Universiti Utara Malaysia.

## REFERENCES

[1] Ezugwu, A.E., Akutsah, F. (2018). An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. IEEE Access, 6: 54459-54478. https://doi.org/10.1109/ACCESS.2018.2872110

[2] Dang, Q.V., van Diessen, T., Martagan, T., Adan, I. (2021). A matheuristic for parallel machine scheduling with tool replacements. European Journal of Operational Research, 291(2): 640-660. https://doi.org/10.1016/j.ejor.2020.09.050

[3] Salimifard, K., Mohammadi, D., Moghdani, R., Abbasizad, A. (2019). Green fuzzy parallel machine scheduling with sequence-dependent setup in the plastic moulding industry. Asian Journal of Management Science and Applications, 4(1): 27-48. https://doi.org/10.1504/AJMSA.2019.101423

[4] Đurasević, M., Jakobović, D. (2021). Heuristic and metaheuristic methods for the unrelated machines scheduling problem: A survey. arXiv Preprint arXiv: 2107.13106. https://doi.org/10.48550/arXiv.2107.13106

[5] Arık, O.A. (2020). Comparisons of metaheuristic algorithms for unrelated parallel machine weighted earliness/tardiness scheduling problems. Evolutionary Intelligence, 13(3): 415-425. https://doi.org/10.1007/s12065-019-00305-7

[6] Ledmi, A., Ledmi, M., Souidi, M.E.H., Haouassi, H., Bardou, D. (2024). Optimizing task scheduling in cloud computing using Discrete Tuna Swarm Optimization. Ingénierie des Systèmes d'Information, 29(1): 323-335. https://doi.org/10.18280/isi.290132

[7] Alsattar, H.A., Zaidan, A.A., Zaidan, B.B. (2020). Novel meta-Heuristic bald eagle search optimisation algorithm. Artificial Intelligence Review, 53: 2237-2264. https://doi.org/10.1007/s10462-019-09732-5

[8] Wang, H.B., Fan, C.C., Tu, X.Y. (2016). AFSAOCP: A novel artificial fish swarm optimization algorithm aided by ocean current power. Applied Intelligence, 45: 992-1007. https://doi.org/10.1007/s10489-016-0798-7

[9] Alhaqbani, A., Kurdi, H.A., Hosny, M. (2022). Fish-inspired heuristics: A survey of the state-of-the-art methods. Archives of Computational Methods in Engineering, 29(6): 3655-3675. https://doi.org/10.1007/s11831-022-09711-0

[10] Pourpanah, F., Wang, R., Lim, C.P., Wang, X.Z., Yazdani, D. (2023). A review of artificial fish swarm algorithms: Recent advances and applications. Artificial Intelligence Review, 56(3): 1867-1903. https://doi.org/10.1007/s10462-022-10214-4

[11] Jin, J., Zhang, Z., Zhang, L. (2023). A modified artificial fish swarm algorithm for unit commitment optimization. In the 8th International Conference on Electromechanical Control Technology and Transportation (ICECTT 2023), 2023, Hangzhou, China, pp. 760-767. https://doi.org/10.1117/12.2689449

[12] Wang, F., Zhao, L., Bai, Y. (2022). Path planning for unmanned surface vehicles based on modified artificial fish swarm algorithm with local optimizer. Mathematical Problems in Engineering, 2022(1): 1283374. https://doi.org/10.1155/2022/1283374

[13] Safarzadeh, H., Niaki, S. (2023). Unrelated parallel machine scheduling with machine processing cost. International Journal of Industrial Engineering Computations, 14(1): 33-48. https://doi.org/10.5267/j.ijiec.2022.10.004

[14] Zheng, F., Jin, K., Xu, Y., Liu, M. (2022). Unrelated parallel machine scheduling with processing cost, machine eligibility and order splitting. Computers & Industrial Engineering, 171: 108483. https://doi.org/10.1016/j.cie.2022.108483

[15] Shafipour, M., Rezaeian, J., Foroutan, R.A. (2022). Minimizing JIT scheduling of unrelated parallel machine with family setups and soft time window constraints. Research Square. https://doi.org/10.21203/rs.3.rs-1917932/v1

[16] Åblad, E., Strömberg, A.B., Spensieri, D. (2021). Exact makespan minimization of unrelated parallel machines. Open Journal of Mathematical Optimization, 2: 1-15. https://doi.org/10.5802/ojmo.4

[17] Arnaout, J.P. (2020). A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. Annals of Operations Research, 285(1): 273-293. https://doi.org/10.1007/s10479-019-03138-w

[18] Jouhari, H., Lei, D., Al-Qaness, M.A., Elaziz, M.A., Damaševičius, R., Korytkowski, M., Ewees, A.A. (2020). Modified Harris Hawks optimizer for solving machine scheduling problems. Symmetry, 12(9): 1460. https://doi.org/10.3390/sym12091460

[19] Fanjul-Peyro, L., Ruiz, R., Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. Computers & Operations Research, 101: 173-182. https://doi.org/10.1016/j.cor.2018.07.007

[20] Chi, X., Liu, S., Li, C. (2022). Research on optimization of unrelated parallel machine scheduling based on IG-TS algorithm. Bulletin of the Polish Academy of Sciences Technical Sciences, 70(4): e141724. https://doi.org/10.24425/bpasts.2022.141724

[21] Al-Qaness, M.A., Ewees, A.A., Abd Elaziz, M. (2021). Modified whale optimization algorithm for solving unrelated parallel machine scheduling problems. Soft Computing, 25(14): 9545-9557. https://doi.org/10.1007/s00500-021-05889-w

[22] Ewees, A.A., Al-Qaness, M.A., Abd Elaziz, M. (2021). Enhanced salp swarm algorithm based on firefly algorithm for unrelated parallel machine scheduling with setup times. Applied Mathematical Modelling, 94: 285-305. https://doi.org/10.1016/j.apm.2021.01.017

[23] Meng, R., Rao, Y., Luo, Q. (2020). Modeling and solving for bi-objective cutting parallel machine scheduling problem. Annals of Operations Research, 285(1): 223-245. https://doi.org/10.1007/s10479-019-03208-z

[24] Ding, J., Shen, L., Lü, Z., Xu, L., Benlic, U. (2019). A hybrid memetic algorithm for the parallel machine scheduling problem with job deteriorating effects. IEEE Transactions on Emerging Topics in Computational Intelligence, 4(3): 385-397. https://doi.org/10.1109/TETCI.2019.2935773

[25] Yaghtin, M., Javid, Y. (2023). Genetic algorithm based on greedy strategy in unrelated parallel-machine scheduling problem using fuzzy approach with periodic

maintenance and process constraints. International Journal of Supply and Operations Management, 10(3): 319-336. https://doi.org/10.22034/ijsom.2023.109377.2359

[26] Rezaeian, J., Mohammad-Hosseini, S., Zabihzadeh, S., Shokoufi, K. (2020). Fuzzy scheduling problem on unrelated parallel machine in JIT production system. Artificial Intelligence Evolution, 17-33. https://doi.org/10.37256/aie.112020202

[27] Cheng, C.Y., Pourhejazy, P., Ying, K.C., Li, S.F., Chang, C.W. (2020). Learning-based metaheuristic for scheduling unrelated parallel machines with uncertain setup times. IEEE Access, 8: 74065-74082. https://doi.org/10.1109/ACCESS.2020.2988274

[28] Sadati, A., Tavakkoli-Moghaddam, R., Naderi, B., Mohammadi, M. (2019). A bi-objective model for a scheduling problem of unrelated parallel batch processing machines with fuzzy parameters by two fuzzy multi-objective meta-heuristics. Iranian Journal of Fuzzy Systems, 16(4): 21-40. https://doi.org/10.22111/ijfs.2019.4779

[29] Liao, T.W., Su, P. (2017). Parallel machine scheduling in fuzzy environment with hybrid ant colony optimization including a comparison of fuzzy number ranking methods in consideration of spread of fuzziness. Applied Soft Computing, 56: 65-81. https://doi.org/10.1016/j.asoc.2017.03.004

[30] Manupati, V.K., Rajyalakshmi, G., Chan, F.T., Thakkar, J. (2017). A hybrid multi-objective evolutionary algorithm approach for handling sequence-and machine-dependent set-up times in unrelated parallel machine scheduling problem. Sādhanā, 42: 391-403. https://doi.org/10.1007/s12046-017-0611-2

[31] Naderi-Beni, M., Ghobadian, E., Ebrahimnejad, S., Tavakkoli-Moghaddam, R. (2014). Fuzzy bi-objective formulation for a parallel machine scheduling problem with machine eligibility restrictions and sequence-dependent setup times. International Journal of Production Research, 52(19): 5799-5822. https://doi.org/10.1080/00207543.2014.916430

[32] Torabi, S.A., Sahebjamnia, N., Mansouri, S.A., Bajestani, M.A. (2013). A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. Applied Soft Computing, 13(12): 4750-4762. http://doi.org/10.1016/j.asoc.2013.07.029

[33] Li, T., Yang, F., Zhang, D., Zhai, L. (2021). Computation scheduling of multi-access edge networks based on the artificial fish swarm algorithm. IEEE Access, 9: 74674-74683. https://doi.org/10.1109/ACCESS.2021.3078539

[34] Sureja, N.M., Patel, S.P. (2020). Solving a combinatorial optimization problem using artificial fish swarm algorithm. International Journal of Engineering Trends and Technology, 68(5): 27-32. https://doi.org/10.14445/22315381/ijett-v68i5p206s

[35] Krishnaveni, H., Janita, V.S. (2019). Modified artificial fish swarm algorithm for efficient task scheduling in cloud environment. International Journal of Computer Sciences and Engineering, 7(5): 1363-71. https://doi.org/10.26438/ijcse/v7i5.13631371

[36] Peng, Z., Dong, K., Yin, H., Bai, Y. (2018). Modification of fish swarm algorithm based on levy flight and firefly behavior. Computational Intelligence and Neuroscience, 2018(1): 9827372. https://doi.org/10.1155/2018/9827372

[37] Sun, Y., Qian, X., Liu, S. (2019). Scheduling deteriorating jobs and module changes with incompatible job families on parallel machines using a hybrid SADE-AFSA algorithm. In Learning and Intelligent Optimization: 12th International Conference, LION 12, Kalamata, Greece, pp. 455-472. https://doi.org/10.1007/978-3-030-05348-2_40

[38] Tirkolaee, E.B., Goli, A., Weber, G.W. (2020). Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option. IEEE Transactions on Fuzzy Systems, 28(11): 2772-2783. https://doi.org/10.1109/TFUZZ.2020.2998174

[39] Zhu, K., Jiang, M. (2010). The optimization of job shop scheduling problem based on Artificial Fish Swarm Algorithm with tabu search strategy. In Third International Workshop on Advanced Computational Intelligence, Suzhou, China, pp. 323-327. https://doi.org/10.1109/IWACI.2010.5585118

[40] Tan, W.H., Mohamad-Saleh, J. (2020). Normative fish swarm algorithm (NFSA) for optimization. Soft Computing, 24(3): 2083-2099. https://doi.org/10.1007/s00500-019-04040-0

[41] Deb, K., Sindhya, K., Hakanen, J. (2016). Multi-objective optimization. In Decision Sciences. Springer, Boston, MA, 2014, pp. 403-449. https://doi.org/10.1201/9781315183176-4

[42] Fu, Y., Hou, Y., Wang, Z., Wu, X., Gao, K., Wang, L. (2021). Distributed scheduling problems in intelligent manufacturing systems. Tsinghua Science and Technology, 26(5): 625-645. https://doi.org/10.26599/tst.2021.9010009

[43] Kongsri, P., Buddhakulsomsiri, J. (2020). A mixed integer programming model for unrelated parallel machine scheduling problem with sequence dependent setup time to minimize makespan and total tardiness. In 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA), Bangkok, Thailand, pp. 605-609. https://doi.org/10.1109/iciea49774.2020.9102086

[44] Li, X. (2003). A new intelligent optimization method-artificial fish school algorithm. Doctor Thesis of Zhejiang University, Hangzhou, China.

[45] Crawford, B., Soto, R., Olgun, E., Mansilla Villablanca, S., Gomez Rubio, A., Jaramillo, A., Salas, J. (2016). An artificial fish swarm algorithm to solve the set covering problem. https://ibook.pub/an-artificial-fish-swarm-optimization-algorithm-to-solve-set-covering-problem.html.

[46] Zhao, L., Wang, F., Bai, Y. (2023). Route planning for autonomous vessels based on improved artificial fish swarm algorithm. Ships and Offshore Structures, 18(6): 897-906. https://doi.org/10.1080/17445302.2022.2081423

[47] Huang, J., Zeng, J., Bai, Y., Cheng, Z., Feng, Z., Qi, L., Liang, D. (2021). Layout optimization of fiber Bragg grating strain sensor network based on modified artificial fish swarm algorithm. Optical Fiber Technology, 65: 102583. https://doi.org/10.1016/j.yofte.2021.102583