AIMS Mathematics, 9(12): 35326–35354.
DOI: 10.3934/math.20241679
Received: 07 July 2024
Revised: 29 November 2024
Accepted: 04 December 2024
Published: 18 December 2024

https://www.aimspress.com/journal/Math

Research article

Modified artificial fish swarm algorithm to solve unrelated parallel machine scheduling problem under fuzzy environment

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

[1] School of Quantitative Sciences, Universiti Utara Malaysia, Sintok, Kedah 06010, Malaysia
[2] Department of Physics, College of Science, University of Sumer, Al Rifaee, Thi-Qar 64005, Iraq

* **Correspondence:** Email: azhar_mahdi_ibadi@ahsgs.uum.edu.my.

Abstract: Unrelated parallel machine scheduling problem (UPMSP) in a fuzzy environment is an active research area due to the fuzzy nature of most real-world problems. UPMSP is an NP-hard problem; thus, finding optimal solutions is challenging, particularly when multiple objectives need to be considered. Hence, a metaheuristic algorithm based on a modified artificial fish swarm algorithm (AFSA) is presented in this study to minimize the multi-objective makespan and total tardiness. Three modifications were made to the proposed algorithm. First, aspiration behavior was added to AFSA behaviors to increase effectiveness. Second, improved parameters such as step and visual were used to balance global search capability and convergence rate. Finally, a transformation method was injected to make the algorithm suitable for discrete optimization problems such as UPMSP. The proposed algorithm was compared with AFSA and five modified versions of AFSA to verify and measure the algorithm's effectiveness by conducting three different sizes of problems. Afterward, the Wilcoxon signed-rank test was used to statistically evaluate the algorithm's performance. The results indicate that the proposed algorithm significantly outperformed the other algorithms, especially for medium and large-sized problems.

Keywords: unrelated parallel machine scheduling problem; fuzzy sets; multi-objective; artificial fish swarm algorithm; modified algorithm
Mathematics Subject Classification: 90C27, 90C70

1. Introduction

In the domain of operations research, the parallel machine scheduling problem (PMSP) is important and difficult. The importance of PMSP comes from its ability to optimize the use of several machines and allocate a specific set of jobs to satisfy the customer's demands. This is vital for industries where resource optimization is essential, such as maximizing productivity and minimizing overall machine time. Additionally, a wide range of industries, especially in high-demand environments like data centers or manufacturing facilities, can adapt PMSP models for various scenarios, ranging from small-scale operations to large, complicated systems [1]. The challenges of PMSP stem from its complex nature, which has gained popularity and led to its classification as NP-hard [1,2]. This means that finding an optimal solution is computationally costly and becomes more difficult as the number of jobs and machines increases. Real-world scenarios frequently include dynamic changes and uncertainties, such as machine breakdowns or different job durations. Machine constraints and multiple objectives must be considered and balancing them can complicate the scheduling process. Therefore, the challenge of developing effective algorithms contributes to the complexity of the problem. According to [3], the successful solution of the PMSP requires simultaneous determination of assignment and sequencing policies for the available parallel machines and jobs.

Academic literature primarily categorizes research on PMSP into three main groups [3]: identical, uniform, and unrelated PMSP (UPMSP). UPMSP can be regarded as a general representation of the other two groups, wherein distinct machines are employed to execute identical jobs but possess varying processing capacities or capabilities. However, addressing real-world UPMSPs poses a significant challenge for both practitioners and researchers, primarily due to their inherent complexity. Notably, UPMSPs are typically classified as NP-hard, even without considering the multi-objective functions [4]. In addition, UPMSPs are considered more realistic than the other groups because machine speeds, technology, and machine types in the shop may differ from one another. Therefore, solving UPMSPs has attracted many researchers and professionals from scientific and engineering disciplines [5]. UPMSPs have found widespread application in production scheduling and manufacturing systems, particularly in semiconductor manufacturing [6], electronic assembly [4], and manufacturing electricity costs [7].

The most effective method for both exact and approximate solutions in small problem instances intended for single-objective optimization is linear programming [8]. In recent years, considerable attention has been given to multi-objective optimization, as seen by a substantial body of literature, particularly in the years 2019 and 2020 [9]. For instance, Sarçiçek [10] proposed a multi-objective UPMSP model that aims to minimize makespan and maximize machine preferences for jobs with sequence-dependent setup times and introduces a simulated annealing metaheuristic to address large-scale problems. Additionally, Meng et al. [11] presented a mathematical model that integrates identical UPMSPs in a structural metal-cutting facility to minimize total makespan and total tardiness. A new metaheuristic algorithm combines the variable neighborhood structure strategy (VNSGA-III) and the non-dominated sorting genetic algorithm III (NSGA-III). Experimental results show that the suggested algorithm statistically outperforms the comparison algorithms. In the meantime, in [12], the UPMSP model was adapted and expanded based on previous mathematical models by focusing on minimizing makespan and a set of constraints for small instances. Then, different types of simulated annealing were suggested to solve large-scale instances in the packaging industry. The comprehensive evaluation and performance analysis show that the proposed methods often outperform state-of-the-art methods.

The most multi-objective scheduling challenges across all fields were minimizing makespan and total tardiness, which also happened to be the multi-objective that most researchers looked at [13,14]. The application of fuzzy sets enhanced the accuracy of schedules to evaluate UPMSPs and deal with the complexities of real-world problems [1,15]. For example, in the transport industry, where the operators are the resources and the jobs are the freight delivery of the real-world UPMSP problem, Rivera Zarate [16] employed a fuzzy relational system to represent the decision makers' preferences and solve the three objective functions: minimizing completion time and risk and maximizing delivery reliability. The outranking-based particle swarm optimization algorithm (O-PSO) is a metaheuristic developed in that study. The study demonstrated its ability to generate high-quality solutions in comparison to a commonly used policy. Another study conducted by [17] aimed to minimize makespan and total cost simultaneously within manufacturing production systems, and a fuzzy programming method was employed to solve large-scale instances of the proposed UPMSP problem. Meanwhile, Zhou et al. [18] developed a tri-objective optimization model for UPMSP to minimize the total cost of the order delay and early penalty, makespan, and workload imbalance; the processing time is fuzzy due to many uncertain variables associated with actual production. A metaheuristic of Pareto-based discrete particle swarm optimization (PDPSO) was employed to evaluate the model across various scales. The findings indicate that the proposed PDPSO surpasses the other algorithms, particularly for small and medium-sized instances.

Prior studies have proposed fuzzy UPMSPs to enhance performance metrics and determine appropriate schedules across a range of scenarios. Such methods encompass exact methods [15,19], heuristic methods [20], and artificial swarm intelligence methods [21]. Therefore, quite a few metaheuristic algorithms have been proposed for UPMSP, with many demonstrating the capability to achieve near-optimal solutions within reasonable time frames [9]. Even though various algorithms have exhibited commendable performance results, the multiple and conflicting objectives remain an open issue [22]. The primary objectives under consideration are the minimization of makespan and total tardiness. Machine scheduling commonly employs makespan, but it can surpass due dates, thereby requiring the inclusion of an additional objective. As a result, even if the optimal makespan solution is identified, many jobs will probably be completed after their due dates. Consequently, simultaneously considering total tardiness and makespan minimization could lead to increased efficiency. While scheduling machines commonly use this multi-objective [14], the information required to define the time parameters and constraint conditions may be vague or not precisely measurable. This is because most real-world problems are fuzzy. For improvement in the quality of the schedule and making it more reflective of scheduling issues that occur in the real world, fuzzy sets can be utilized to evaluate UPMSPs.

Limited research has been conducted on UPMSP with the multi-objective makespan and total tardiness in the fuzzy environment [15]. Recently, Pourpanah et al. [23] conducted a comprehensive review of the artificial fish swarm algorithm (AFSA). The authors revealed that most modifications of AFSA focus on controlling the parameter values of the algorithm, particularly the visual distance, maximum step length, and crowding factor, along with updates and enhancements to existing behaviors or the development of new ones. AFSA has seen a lot of modifications recently to enhance optimization's effectiveness, most of which aimed for a balance among the exploration and exploitation procedures. Even so, currently used fish swarm algorithms have not yet achieved a global optimum at remarkably high convergence rates. Thus, AFSA development still has a tremendous deal of potential.

In recent years, the improved and modified AFSA became a valuable tool that can be efficiently

applied in a variety of applications. Zhao et al. [24] adopted the improved AFSA for route planning of autonomous vessels by introducing a directional operator to improve efficiency, a probability weight factor to avoid local optima, and an adaptive operator to achieve better convergence performance. Tan & Mohamad-Saleh [25] presented a new algorithm of AFSA to improve the performance of global and local search techniques in optimization. The proposed algorithm includes several improvements, such as hybridizing characteristics, introducing normative communication and memory behaviors, and adapting parameters in terms of visuals and steps. Meanwhile, Huang et al. [26] proposed three adaptive step methods to speed up and address the shortcomings of the standard AFSA for solving sensor layout optimization problems in fiber grating networks. Research by Wang et al. [27] customized four operators and an adaptive factor to enhance the algorithm's convergence performance and suggested a modified AFSA when combined with a local path optimizer to resolve the path planning issue. Jin et al. [28] proposed a modified AFSA for unit commitment optimization to overcome the disadvantages of premature convergence and local extremes in the original algorithm. The improvements include variable vision, adjusting the movement strategy, and combining the mutation operation of genetic algorithms. A study by Gao et al. [29] introduced novel AFSA by using Cauchy mutation to accelerate the algorithm's speed of convergence to solve the parameter selection problem for the twin support vector machine. Li et al. [30] proposed improving AFSA to determine the best scheduling to minimize time delay for the multidimensional knapsack problem. These modifications attempted to improve the algorithm's performance in terms of convergence, optimality, and escape from local optimality, among others. However, the development of the algorithm continues to this day.

While there is limited research on the use of modified AFSA in machine scheduling, Tirkolaee et al. [31] introduced an improved version of AFSA to solve flow shop scheduling (FSS) problems. The aims were to minimize total cost and total energy consumption by adding self-adaptive behavior to enhance the algorithm's performance. To our knowledge, AFSA has never been used to solve the scheduling problem on unrelated machines for the proposed multi-objective problem under fuzzy environments. Therefore, in this study, a modified AFSA is proposed to minimize the proposed multi-objective problem with considerations of fuzzy triangular numbers to mitigate the uncertainty associated with the processing times and due dates. The proposed modified AFSA is a new algorithm designed to solve UPMSP by exploiting the capabilities of the AFSA, which is modified through three aspects: the addition of new behavior for the four main behaviors, the use of improved visual and step parameters, and converting AFSA from a continuous to a discrete solution space to deal with the discrete proposed model. Therefore, after creating a random fish population to represent UPMSP solutions, the proposed modified AFSA starts with the best individual solution generated by the standard AFSA. Additionally, the AFSA's constant value is replaced with improved visual and step parameters, and the performance is updated based on the best and current individual using aspiration behavior expression. Subsequently, the fitness value of the best individual is calculated using a transformation method to convert the continuous solution into a discrete solution. Therefore, all solutions will be updated according to the previous steps and implemented until the termination criterion is satisfied.

In short, the main contributions of this research are described as follows:

- Adapt and extend the linear multi-objective model introduced by [32] for UPMSP by embedding triangular fuzzy parameters.
- Apply the total integral value defuzzification method that converts a fuzzy output into a crisp

output value.

- Modify standard continuous AFSA to deal with the discrete UPMSP model in solving the proposed UPMSP model.
- Evaluate the performance of the proposed modified algorithm by conducting computational experiments and comparing the results with AFSA and five versions of modified AFSA algorithms. The evaluation results confirmed the superior performance of the modified algorithm.

The performance of the proposed algorithm was compared against standard AFSA and three modified versions proposed by Zhao et al. [24] and Tan & Mohamad-Saleh [25] and three versions of methods outlined by Huang et al. [26].

This paper is structured as follows: Section 2 presents the problem formulation, while Section 3 explains the preliminaries. The standard AFSA is described in Section 4, and the proposed modified AFSA is detailed in Section 5. Section 6 provides computational and statistical results, along with the analysis of the evaluation process. Finally, Section 7 discusses the conclusions and future research directions.

2. Problem formulation

Based on the classification scheme of scheduling problems by Graham et al. [33], the UPMSP for the multi-objective can be indicated as $R//(C_{max} + \sum T_j)$, where R denotes an unrelated machine and $C_{max} + \sum T_j$ represents a proposed multi-objective function. The following assumptions are considered before formulating the mathematical model for the proposed multi-objective UPMSP.

Assumptions:

1- Every machine and job are available at the start of time.
2- Parallel machines are unrelated (each job's processing time differs and is determined by the machine).
3- Preemption is not allowed.
4- There is only one job that each machine can process at once.

Indices and sets:

N: A set of jobs.
M: A set of machines.
j, k : Index for jobs, $j, k \in N = \{1, \dots, n\}$.
N_0 : Indicates a set of jobs that involve a dummy job, $N_0 = \{0,1, \dots, n\}$.
i: Index for machine, $i \in M = \{1, \dots, m\}$.

Parameters and decision variables:

p_{ij} *:* The processing time of job j on machine i, which could differ depending on different machines.
d_j *:* The due date of job j.
C_j : The completion time of job j.
C_{max} : The maximum completion time (makespan).
w: The value of weight.
v: Sufficiently large number.
T_j : The tardiness of job j.

$$Xikj = \begin{cases} 1, & \text{if job } j \text{ immediately follows job k on machine } i \\ 0, & \text{Otherwise} \end{cases}$$

The notation definitions above and the current model have been inspired by the models that were introduced in papers [34,35]. Meanwhile, the proposed multi-objective model for UPMSP is extended to the linear multi-objective model introduced by [32], which is formulated as follows:

$$Min\ F\ = wC_{max} + (1-w)\sum_{j=1}^{n} T_j \tag{1}$$

Subject to:

$$\sum_{i\in M}\sum_{k\in N_0, k\neq j} X_{ikj} = 1 \qquad \forall\, j \in N, \tag{2}$$

$$\sum_{i\in M}\sum_{j\in N_0, j\neq k} X_{ikj} = 1 \qquad \forall\, k \in N, \tag{3}$$

$$\sum_{j\in N_0, j\neq k} X_{ikj} - \sum_{h\in N_0, j\neq k} X_{ihk} = 0 \quad \forall\, i \in M, k \in N, \tag{4}$$

$$\sum\nolimits_{j\in N} X_{i0j} \leq 1 \quad \forall i \in M, \tag{5}$$

$$C_0 = 0, \tag{6}$$

$$C_j - C_k + v\left(1 - X_{ikj}\right) \geq p_{ij} \quad \forall i \in M, \forall k \in N_0, \forall j \in N: j \neq k, \tag{7}$$

$$C_{max} \geq C_j \quad \forall j \ \in N, \tag{8}$$

$$T_j \geq C_j - d_j \quad \forall \mathrm{j} \in N, \tag{9}$$

$$p_{ij}, C_j, T_j, C_{max} \ \geq 0, \tag{10}$$

$$X_{ikj} \in \{0,1\}. \tag{11}$$

Equation (1) works to minimize multi-objective functions' makespan and total tardiness simultaneously. Constraint (2) guarantees that every job is done by just one machine. When job j^{th} is executed after the job k^{th} on the machine i^{th}, the X_{ikj} value is equal to 1; otherwise, the value will be 0. Constraint (3) demonstrates that there is only one job that is scheduled first at each machine, which means $X_{ikj} = 1$ when the k^{th} job is the first job on a machine i^{th}, otherwise $X_{ikj} = 0$. Constraint (4) specifies the job's flow balance at each machine, which means only one preceding job and one succeeding job. Constraint (5) determines the machine's initial job. Constraint (6) gives zero completion time at job 0. Constraint (7) ensures that the completion time of job j^{th} equals the completion time of the preceding job plus the processing time of job j^{th}at each machine; this can be performed using a large number v. When $X_{ikj} = 1$ if the job j^{th} is ordered after job k^{th}, then

$v(1 - X_{ikj}) = 0$ and $C_j = C_k + p_{ij}$. Otherwise, when job j^{th} is not ordered after job k^{th}, then $X_{ikj} = 0$, and thus, $v(1 - X_{ikj}) = v$. Constraint (8) demonstrates that the maximum completion time for all machines is equal to the makespan. Constraint (9) calculates each job's tardiness value. Constraint (10) ensures that no value for any of the decision variables can be negative. Constraint (11) illustrates binary variables.

3. Preliminaries

In the literature, fuzzification refers to the process of converting a crisp set into a specific fuzzy set [36]. The primary objective of a fuzzy scheduling process is to identify the "best" solution, or decision, given the presence of uncertain data. Zadeh [37] introduced the fuzzy set theory, which expands the classical set theory's 0–1 integer range to include the values [0,1]. This section introduces some basic prerequisites for fuzzy sets and triangular fuzzy numbers that will be used later.

Definition 1 [38]: A fuzzy set $\tilde{A}$ in X is specified by the function $\mu_{\tilde{A}}: X \longrightarrow [0, 1]$, a set of ordered pairs if X is a collection of items represented by x:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) : x \in X\} \tag{12}$$

As shown in Figure 1, the membership function $\mu_{\tilde{A}}(x)$ of the fuzzy set $\tilde{A}$ is the relationship between various crisp values x and values in the interval [0,1].

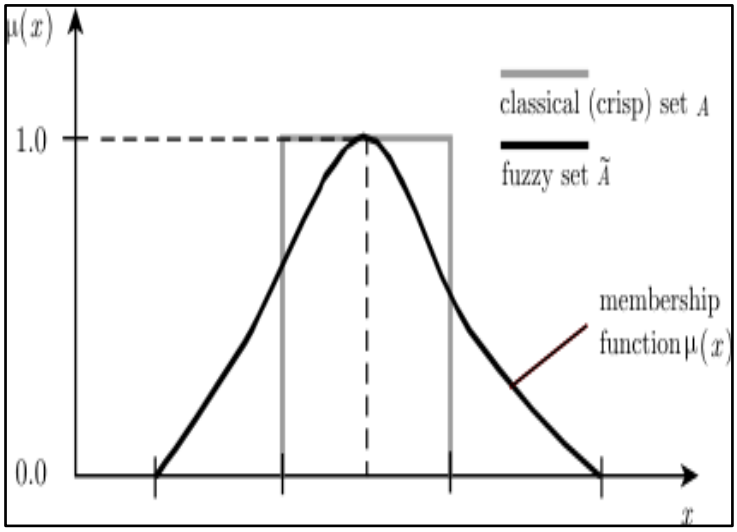

Figure 1. Crisp set A with fuzzy set $\tilde{A}$.

In fuzzy theory applications, the most widely used and basic model of fuzzy numbers is called a triangular fuzzy number (TFN), which is represented by $\tilde{A}$=(a_1, a_2, a_3). The parameter a_2 is the maximum possible evaluation data value of x, and parameters $a_1$and a_3 denote the lower and upper bounds of the evaluation data's available area [39]. The graph of the membership function (x) in Figure 2 has a triangular shape, with the base over the interval $[a_1, a_3]$ and the vertex at $x = a_2$.

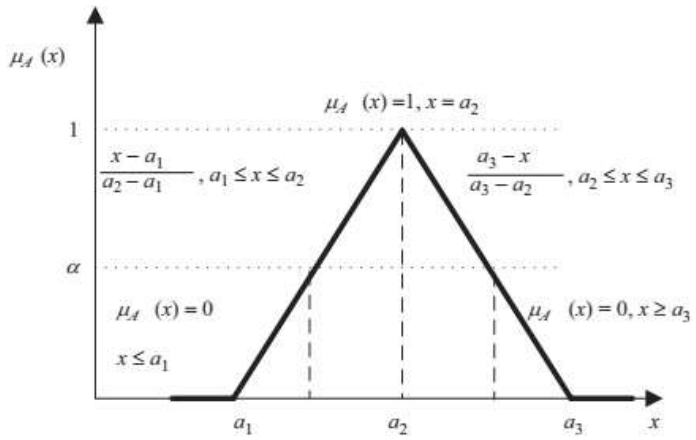

Figure 2. Membership function for the triangular fuzzy number $\widetilde{\mathrm{A}} = (\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3)$.

Definition 2: Let $\tilde{A} = (a_1, a_2, a_3)$ and $\tilde{B} = (b_1, b_2, b_3)$ be two TFNs. The main operations are also TFNs and can be performed as follows [40]:

$\tilde{A} + \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3)$

$\tilde{A} - \tilde{B} = (a_1 - b_3, a_2 - b_2, a_3 - b_1)$

The fuzzy mathematical model is the proposed deterministic model that has been fuzzified using TFNs for the primary parameters of processing time and due date stated here as follows:

$$Min\ F\ = w\widetilde{C_{max}} + (1-w)\sum_{j=1}^{n}\sum_{i=1}^{m}\tilde{T}_j, \tag{13}$$

$$\tilde{C}_0 = 0, \tag{14}$$

$$\tilde{C}_j - \tilde{C}_k + v\left(1 - X_{ikj}\right) \geq \tilde{p}_j \quad \forall i \in M, \forall k \in N_0, \forall j \in N: j \neq k, \tag{15}$$

$$\widetilde{C_{max}} \geq \tilde{C}_j \quad \forall j \in N, \tag{16}$$

$$\tilde{T}_j \geq \tilde{C}_j - \tilde{d}_j \quad \forall i \in M, \forall \mathrm{j} \in N, \tag{17}$$

$$\tilde{p}_{ij}, \tilde{C}_j, \tilde{T}_j, \widetilde{C_{max}} \geq 0. \tag{18}$$

Considering the above model, an optimal solution cannot be obtained in such a fuzzy environment [41]. Defuzzification is the process of transforming a fuzzy output into a single, crisp output value. A study conducted a performance evaluation and comparison of the nine defuzzification methods introduced in [42] to understand the impact of fuzziness on the fuzzy parameters in the UPMSP. The integral method introduced by [43] was found to have significantly better performance than the other methods based on comprehensive testing. To calculate the maximum fuzzy makespan and tardiness, it is necessary to use a method that compares TFNs. According to [43], the total integral value can be used and denoted as:

$$I_T^{\alpha}(A) = \frac{1}{2}\alpha(b+c) + \frac{1}{2}(1-\alpha)(a+b)$$
$$= \frac{1}{2}[\alpha c + b + (1-\alpha)a] \tag{19}$$

for triangular fuzzy number $A = (a, b, c)$, and given $\alpha \in [0, 1]$.

4. Artificial fish swarm algorithm

A novel swarm intelligence-based optimization technique known as the artificial fish swarm algorithm (AFSA) was introduced by Li Xiao Lei [44], which involves randomly searching for a set of solutions to solve NP-hard problems [23]. Compared to other intelligent algorithms, AFSA has a limited number of parameters to manage, making it relatively simple to implement. The amount of time needed to find good solutions is reasonable, and the algorithm is parallel, has a strong global search capability, converges quickly, and is less sensitive to the needs of the objective functions. These benefits have led to the successful application of AFSA and its variants in a diverse array of optimization problems [25], such as flexible job shop scheduling problems [45], logistics distribution [46], network design problems [47], wireless sensor networks [48], cloud computing [49], the traveling salesman problem [50], and the exam timetabling problem [51]. Furthermore, Peraza et al. [52] demonstrated the successful implementation of AFSA in various benchmarking optimization problems. The basic idea behind AFSA is to mimic the various environmental behaviors that schooling fish exhibit in the water, where a fish serves as a fictitious representation of a true fish within a population, and the movements of the swarm are random. This algorithm trains each artificial fish (AF) to react to its current position and teaches it four basic behaviors: prey, swarm, follow, and random. Prey behavior is the foundation of the algorithm's convergence; swarm behavior improves the algorithm's stability and global convergence; follow behavior accelerates the convergence of the algorithm; and random behavior balances the contradiction of the other three behaviors. Here is a description of AFSA's behaviors:

Preying behavior: This essential behavior enables AF to get to locations where there is a greater concentration of food. This behavior is modeled within a radius of a fish's neighborhoods by using an AF's current location and its nearest neighbors as determined by the AF's field of view. Let X_i be the current position of the i^{th}AF, and X_j be an arbitrary state of an AF as follows:

$$X_j = X_i + Visual \times R\ (0,1) \tag{20}$$

$Visual$ is the visual length between two AFs that are placed in X_i and X_j, where the Euclidean distance can be defined as $dist_{ij} = \| X_i - X_j \|$, and R is a random vector with each element between 0 and 1.

In minimization problems, if $f(X_j) \leq f(X_i)$, the i^{th}AF moves a step toward X_j and, using the following formula:

$$X_{prey} = X_i + \frac{X_j - X_i}{\|X_j - X_i\|} \times Step \times R\ (0,1) \tag{21}$$

where X_{prey} is the position of the i^{th} AF after executing praying behavior, which is next to the $X_j - X_i$, and $\| X_{prey} - X_i \| \leq S$. However, if $f(X_j) > f(X_i)$, another position X_j is randomly selected by applying Eq (1). If the fitness value (objective function), which could be the amount of food the AF is currently concentrating, fails to be satisfied after trying for a given number of iterations, X_i

randomizes its next step as follows:

$$X_{prey} = X_i + Visual \times R(0,1) \tag{22}$$

Swarming behavior: In this behavior, the fish collect and move toward the center position to avoid crowding. Let n denote the total number of AFs, n_f be the number of neighbors within the i^{th}AF's visual range with $dist_{ij} < Visual$, and X_{center} be the central position of the swarm, which is the average of all AFs' positions:

$$X_{center} = \frac{1}{n}\sum_{n=1}^{k} X_k \tag{23}$$

The i^{th}AF relocates in the direction of X_{center} if $f(X_{center}) \leq f(X_i)$ and $\frac{n_f}{n} \geqslant \mu$, where $\mu \in [0,1]$ is the crowding factor; this indicates that there is greater availability of food in the center and that the location is not overcrowded. As a result, X_i takes a movement toward the neighbor's center as follows:

$$X_{swarm} = X_i + \frac{X_{center} - X_i}{\|X_{center} - X_i\|} \times Step \times R(0,1) \tag{24}$$

otherwise, the i^{th}AF will carry out the preying behavior.

Following behavior: For i^{th}AF, if at least another j^{th}AF is represented where $\| X_j - X_i \| \leq Visual$ and $f(X_j) > f(X_i)$ with low crowded factor, i.e., $\frac{n_f}{n} < \mu$, then i^{th}AF move one step toward the j^{th} AF position is as follows:

$$X_{Follow} = X_i + \frac{X_j - X_i}{\|X_j - X_i\|} \times Step \times R(0,1) \tag{25}$$

If there are no neighbors around X_i or if none of them satisfy the condition, then i^{th} AF performs preying behavior.

Random behavior: Each AF makes its random search process and lets the fish swim freely or follow a swarm in a larger space. The next state of AF is X_{next} and defined as follows:

$$X_{next} = X_i + Visual \times R(0,1) \tag{26}$$

5. The proposed modified artificial fish swarm algorithm

In standard AFSA, AF typically randomly searches its visual range for the next state before proceeding to the next position. This method is particularly time-consuming. The AF must explore every direction until it finds its position. To address this weakness, we incorporate aspiration behavior into the four primary behaviors of the AFSA, which are influenced by the best position of AFs. The proposed algorithm starts by initializing the population size of a fish swarm, followed by the initiation of the AFSA and the proposed AFSA parameters. Every AF signifies a solution to the problem, receives a fitness value to assess its performance, and records its results on the bulletin board following each iteration. Like standard AFSA, each AF in the proposed algorithm changes its position based on its individual best solution for the whole swarm through each iteration. However, there is a major difference in how the best solution is updated. The modified algorithm compares the current position of an AF with the best position achieved across all behaviors. This enhances the search process and helps AFs to escape from the local optimums. Figure 3 illustrates the flowchart of the modified algorithm, highlighting the modified steps in yellow. Creating new behavior depends on the best,

current, and random movement solutions. The update processes are formulated as Eq (27).

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

$$X_{Aspiration} = X_i + \frac{X_{best} - X_i}{\|X_{best} - X_i\|} \times Step \times R(0,1) \tag{27}$$

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

However, through iterative processes, the visual and step parameters in the standard AFSA are fixed. The algorithm's convergence speed is increased at the beginning with large visual and step values. Nevertheless, the large values will result in problems like local optimum or iterative jumps when AFs approach the final position. However, values that are too low reduce effectiveness [24,25]. As a result, there are specifications for the algorithm for the visual's size and step at various phases. To achieve a balance between convergence rate and global search capability, the improved adaptive parameters (visual, step) utilized in [25] were adapted by the modified algorithm and are given as follows:

$$visual^{t+1} = visual^t - visual^t \times \lambda + visual_{min} \tag{28}$$

$$step^{t+1} = step^t - step^t \times \lambda + step_{min} \tag{29}$$

$$\lambda = e^{\left(-\frac{\sigma}{\sqrt[4]{(t_{max})^3}} * (t_{max} - t)\right.} \tag{30}$$

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.

The AFSA was originally created to address continuous problems; solving combinatorial optimization problems such as UPMSP remains challenging. Therefore, it is crucial to make the algorithm suitable for solving discrete mathematical scheduling problems and improving the original algorithm's search capabilities. The major contribution of the proposed algorithm is the use of the transformation method to transform the continuous solution into a discrete solution. To achieve this, the modified algorithm's continuous solution was encoded, and the fractional parts were sorted in a non-decreasing order. Let us illustrate the proposed procedure as follows: Assume that a UPMSP has five jobs and three machines and let the lower bound for algorithm search space be equal to 0 and the upper bound be equal to 1. Let the generated solution by AFSA, S = [0.23 0.43 0.1 0.7 0.2], be continuous, and then encode the solution by integer numbers N = [1 2 3 4 5]. After a combination of solution and index, the solution turns out to be:

1	2	3	4	5
0.23	0.43	0.1	0.7	0.2

continuous solution

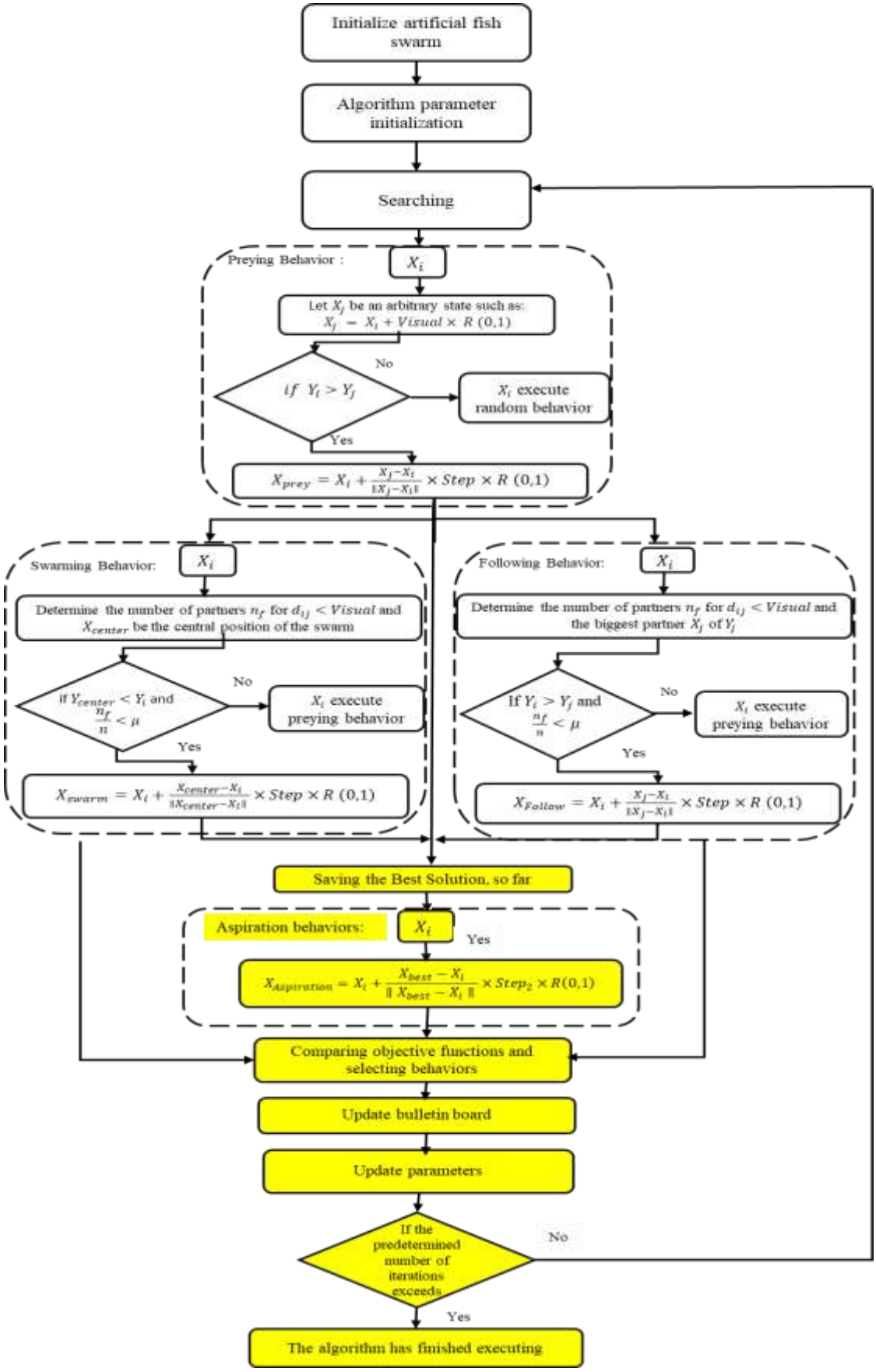

Figure 3. Flowchart of the proposed modified algorithm.

Now, the solution is sorted by increasing order, and the index is arranged based on the solution. After sorting, the corresponding solution should be:

3	5	1	2	4
0.1	0.2	0.23	0.43	0.7

discrete solution

Thus, the discrete solution D = [3 5 1 2 4].

This method is used to compute the best solution for each solution in the population, and then the best solution is selected according to the value of the multi-objective function. The entire optimization process is performed and runs simultaneously by employing the three proposed modification steps, which are interconnected and depend on the quality of the solution.

The proposed algorithm was applied for each machine and job combination, which were randomly generated, with three types of problem instances: small, medium, and large size. Each instance was repeated 10 times, and the results were recorded. The smallest minimum value is determined to be the optimum value for each instance. The details of the results and comparisons are discussed in the following section.

6. Results and discussion

Various experiments were conducted on three differently sized problem instances to confirm the proposed algorithm's performance for the multi-objective UPMSP. Each size of the problem consists of 10 instances and was run 10 times. Small size problems 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 had 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 had N = [250,270,290,310,330,350,370,390,410,430] and M = [13,13,15,15,16,16,17,17]. The proposed AFSA results are compared with standard AFSA, other versions of modified AFSA (presented in previous works by Zhao et al [24] and Tan & Mohamad-Saleh [25]), and three methods outlined by Huang et al [26]. The performance evaluation of the solution methodology is made based on the minimum values, mean, and standard deviation of the objective function obtained. Further analysis of the proposed algorithm's performance was also conducted using the Wilcoxon signed-rank test. The experiments were conducted based on different parameter values, as given in Table 1, using 12th Gen Intel(R) Core (TM) i7-1280P 2.00 GHz and 16 GB RAM under Windows 11 (64-bit) with MATLAB R2023a.

Table 1. Parameter settings for the proposed modified algorithm.

Parameter	Values
Try-number	10
Fish population ($nPop$)	40
$visual_{min}$	30
$step_{min}1$	1
$step_{min}2$	2
Crowd factor, μ	0.3
σ	0.6
Max-Iterations	1000

a) Comparison of results based on the minimum values

In minimization problems, the minimum values are used to determine the best performance of the proposed solution approach. Tables 2, 3, and 4 illustrate the minimum values of the proposed algorithm for 10 instances of different size problems. The results are then compared with the standard AFSA, Modified 1 [24], Modified 2 [25], and three methods presented in Modified 3 [26].

Table 2. Minimum values of proposed AFSA and their comparison with other versions of AFSA for small-sized instances.

	N	*M*	**Proposed**	**AFSA**	**Modified 1**	**Modified 2**	**Modified 3 - method 1**	**Modified 3 - method 2**	**Modified 3 - method 3**
1	5	3	**23.25**	23.25	23.25	23.25	23.25	23.25	23.25
2	10	3	**42**	42.75	42	42	42	42	42
3	15	4	**47.25**	48	50.25	48.75	49.5	51	48
4	20	4	62.25	60	62.25	59.25	62.25	63.75	**58.5**
5	25	5	63.75	63	65.25	64.5	66.75	64.5	**62.25**
6	30	5	74.25	72	75	70.5	76.5	78.75	**70.5**
7	35	6	83.25	81.75	82.5	78.75	86.25	91.5	**78.75**
8	40	6	96.25	94	102.25	94	96.25	100	**92.5**
9	45	7	90	**87.75**	96	88.5	94.5	102	89.25
10	50	7	91.5	**89.25**	97.5	90	98.25	99	91.5

Table 3. Minimum values of proposed AFSA and their comparison with other versions of AFSA for medium-sized instances.

	N	*M*	**Proposed**	**AFSA**	**Modified 1**	**Modified 2**	**Modified 3 - method 1**	**Modified 3 - method 2**	**Modified 3 - method 3**
1	60	8	**62.5**	63	66	63	67.5	68.5	65
2	80	8	107.5	**101**	118.5	102.5	123	133	110
3	100	9	**105.5**	106	117.5	108.5	117.5	117.5	113.5
4	120	9	**123.5**	141.5	147.5	131.5	173	164	136
5	140	10	**136.5**	136.5	201	142.5	193	221	154
6	160	10	**161.5**	173.5	230.5	178.5	221.5	348	192
7	180	11	**154**	154.5	208.5	163	269.5	326	183.5
8	200	11	**181.5**	192.5	269.5	187.5	282.5	372	209.5
9	220	12	**205**	238	538	262.5	660.5	697.5	340
10	240	12	**251.5**	308.5	621	294.5	664.5	1206	351

Table 4. Minimum values of proposed AFSA and their comparison with other versions of AFSA for large-sized instances.

	N	*M*	**Proposed**	**AFSA**	**Modified 1**	**Modified 2**	**Modified 3 - method 1**	**Modified 3 -method 2**	**Modified 3 - method 3**
1	250	13	**262**	287.5	629	287	684	1058	384.5
2	270	13	**270**	276	715	347	971.5	1876	429
3	290	14	**409**	452.5	1463.5	432.5	1249.5	2580	715.5
4	310	14	**303**	404	1191	327.5	1313	3215	514
5	330	15	559	542.5	1695.5	**448.5**	1543	4593	716.5
6	350	15	**565**	898	2014.5	884	2624.5	8196.5	1121
7	370	16	**449.5**	475.5	1510	463.5	1772	6986	668
8	390	16	**847**	891	2647.5	801	3150	10149.5	1369.5
9	410	17	**517**	649	2584.5	841.5	2437.5	9709	1191
10	430	17	**696.5**	938	2752	977	3666.5	11626.5	1411.5

As shown in Tables 2, 3 and 4, the bold value is the minimum value obtained that represents the best-found solution in comparison with all seven versions of AFSA. The proposed algorithm exhibits relative performance improvements when dealing with small-sized problems, as demonstrated by the results in Table 2. It is notable that the best minimum values that the proposed algorithm yields are 23.25, 42, and 47.25, which happened in the instances of (N=5, M=3), (N=10, M=3), and (N=15, M=4), respectively. However, as the number of jobs and machines increases to medium and large sizes (as seen in Table 3 and Table 4), the algorithm demonstrates an even more pronounced enhancement in its superior performance. We noticed that the proposed algorithm provides the best minimum values in nine instances of medium-sized problems, 62.5, 105.5, 123.5,136.5,161.5,154, 181.5, 205, 251.5, which happened in the instances of (N=60, M=8), (N=100, M=9), (N=120, M=9), (N=140, M=10), (N=160, M=10), (N=180, M=11), (N=200, M=11), (N=220, M=12), and (N=240, M=12), respectively. Meanwhile, the standard AFSA provides the best minimum values in the second instance when (N=80, M=8). In large-sized problems, the proposed algorithm also provides the best minimum values in nine instances and gives the values 262, 270,409, 303, 565, 449.5, 847, 517, 696.5, which happened in the instances of (N=250, M=13), (N=270, M=13), (N=290, M=14), (N=310, M=14), (N=350, M=15), (N=370, M=16), (N=390, M=16), (N=410, M=17), and (N=430, M=17), respectively. The standard AFSA provides the best minimum value in the fifth instance when (N=330, M=15). We observe that in medium and large size problems, the AFSA algorithm converges with an unsatisfactory solution. However, when modifications are added to the AFSA, the proposed modified algorithm converges to the best solutions in 9 out of 10 instances for each problem for the proposed multi-objective function. This reflects the effectiveness of the proposed modifications, making the modified algorithm more effective in solving the proposed multi-objective UPMSP.

b) Comparison of results based on statistical values

A descriptive statistical analysis of the performance of AFSA and several modified AFSA algorithms was conducted to demonstrate the superior accuracy and computational efficiency of the proposed algorithm. The results in Tables 5, 6, and 7 present the numerical outcomes of all algorithms after 10 independent runs of the proposed UPMSP model. In the tables, “Mean” refers to the average value, and “STD” refers to the standard deviation, which was used to verify the reliability of the performance of the proposed algorithm. The results show that as more jobs and machines are added to the problem, the mean and STD of all algorithms increase. A smaller mean value represents the best average solution performed by the algorithms, and a lower STD indicates that the results are closer to the mean, suggesting more consistent or stable performance.

Table 5. Statistical values of proposed AFSA and their comparison with other versions of AFSA for small-sized instances.

	N	*M*		Proposed	AFSA	Modified 1	Modified 2	Modified 3 -method 1	Modified 3 -method 2	Modified 3 -method 3
1	5	3	**Mean**	**23.475**	23.475	23.7	24.6	23.925	24.075	23.7
			STD	**0.711512**	0.711512	0.9486833	1.161895	1.08685326	1.38969421	0.9486833
2	10	3	**Mean**	**43.275**	43.8	44.55	43.95	44.55	44.475	43.575
			STD	1.325236	**0.880341**	2.12720474	1.665833	2.15638587	2.0630681	1.02774024
3	15	4	**Mean**	**51.975**	53.625	53.175	52.125	54.825	56.025	51.99
			STD	2.646932	3.14742	2.67978544	**2.430278**	4.09954266	2.69374275	2.97489496
4	20	4	**Mean**	64.05	62.55	66.3	62.775	66.375	66.9	**61.95**
			STD	**1.589025**	1.627882	2.94108823	2.265226	2.65165043	1.6507574	1.73925271
5	25	5	**Mean**	67.075	65.325	70.275	67.875	70.025	72.075	**65.3**
			STD	**1.852363**	1.982738	2.84421928	2.430278	3.06741386	4.61586937	1.97132217
6	30	5	**Mean**	76.275	73.65	79.275	74.25	78.975	80.625	**72.6**
			STD	**1.371587**	1.573213	3.85762232	2.236068	2.39921862	2.3782872	1.6881943
7	35	6	**Mean**	87.9	85.125	91.65	83.55	90.85	97.35	**83.325**
			STD	3.274905	**2.378287**	7.58122681	3.28253	4.74517299	4.79322438	2.67978544
8	40	6	**Mean**	98.875	98.5	108.55	97.75	105.55	109.375	**98**
			STD	**2.243045**	3.391165	9.38216393	3.221025	5.61471282	11.3976423	3.61132478
9	45	7	**Mean**	95.025	**94.725**	105.025	95.85	107.025	116.8	96.45
			STD	4.184047	**3.708942**	8.32453702	3.933828	7.31574596	11.5426744	4.86312657
10	50	7	**Mean**	94.65	**93.6**	108.65	93.125	107.475	112.95	96.375
			STD	**1.864135**	2.202272	7.22284185	2.118864	4.89961733	9.4647005	3.10745072

Table 6. Statistical values of proposed AFSA and their comparison with other versions of AFSA for medium-sized instances.

	N	*M*		Proposed	AFSA	Modified 1	Modified 2	Modified 3 -method 1	Modified 3 -method 2	Modified 3 -method 3
1	60	8	**Mean**	65.9	**64.65**	69	65.45	69.8	72.35	66.45
			STD	1.449138	**1.155903**	2.47206616	1.403369	1.70293864	4.0828231	1.32182534
2	80	8	**Mean**	117.35	**112.2**	148.5	113.65	155.2	164.8	128.1
			STD	8.3335	**6.028635**	29.3342803	7.742559	25.2005291	19.3249982	12.8101522
3	100	9	**Mean**	**110**	112.8	133.9	113.9	147.1	141.3	121.25
			STD	**3.146427**	3.851407	12.8491245	4.677369	24.4151592	14.2150155	9.98123239
4	120	9	**Mean**	**141.65**	148.2	195.85	146.75	218.1	254.9	167.3
			STD	**5.869885**	10.43245	37.7131351	18.29731	39.0894985	65.5802308	26.462972
5	140	10	**Mean**	**151.75**	156.15	231.25	153.9	238.2	297.3	177
			STD	**181**	196.2	307.35	193.8	363.1	449.6	240.85
6	160	10	**Mean**	17.02449	17.20982	59.1960256	**9.724539**	90.8407642	105.066064	26.5842999
			STD	177.95	**175.15**	299.65	181.45	326.7	456.3	217.45
7	180	11	**Mean**	12.71143	15.90606	70.8876458	**12.38604**	46.1923515	94.8976642	23.2360472
			STD	3.274905	**2.378287**	7.58122681	3.28253	4.74517299	4.79322438	2.67978544
8	200	11	**Mean**	**203.85**	213	310.3	204.5	393.2	580.55	240.6
			STD	18.0909	20.45863	33.4964343	**11.19524**	101.628736	99.4899353	26.4215821
9	220	12	**Mean**	**281.1**	317.2	752.55	302.9	837.35	1138.65	402.9
			STD	60.7096	50.07949	116.875397	**27.08403**	141.488133	306.002637	58.3784587
10	240	12	**Mean**	**324.25**	373.45	944.45	376.15	931.85	1702	442.95
			STD	**47.46768**	47.62379	231.365085	52.06942	162.443845	418.957038	66.6468512

Table 7. Statistical values of proposed AFSA and their comparison with other versions of AFSA for large-sized instances.

	N	*M*		Proposed	AFSA	Modified 1	Modified 2	Modified 3 -method 1	Modified 3 -method 2	Modified 3 -method 3
1	250	13	**Mean**	**344.6**	377.95	918.3	356.85	956.35	1705.35	481.45
			STD	56.49916	**45.5689**	195.281876	67.45412	166.979049	448.824146	65.4393061
2	270	13	**Mean**	**336.15**	430.4	1142.9	410.5	1181.6	2809.5	515.65
			STD	**43.07361**	78.93943	193.103714	47.35387	157.181919	954.20147	80.3202099
3	290	14	**Mean**	**573.9**	634.95	1753.05	566.2	1790.5	3777.8	866.9
			STD	118.7923	**76.78341**	138.85473	90.48732	292.947853	681.76899	118.181076
4	310	14	**Mean**	**432.8**	495	1440.4	448.3	1586.95	4748.05	608.55
			STD	106.8431	80.23611	180.931202	**79.73156**	247.053689	1065.97672	78.5367748
5	330	15	**Mean**	**637.45**	674.55	2187.55	699.15	2123.25	6812.7	985.8
			STD	**79.19822**	142.6311	339.222104	124.8408	317.492192	1292.51091	178.974579
6	350	15	**Mean**	**779.25**	1068.15	2767.5	1060.1	3106.25	9727.05	1408.65
			STD	144.6231	**131.5367**	475.518664	148.3487	367.185864	804.673172	154.263852
7	370	16	**Mean**	652.7	**634.75**	1965.45	689.65	2187.8	8790.5	927.1
			STD	**133.5825**	144.9935	257.429219	135.1053	309.22225	752.913858	208.926064
8	390	16	**Mean**	993.9	1304.75	3357.75	1120.05	3575.9	11184.05	1705.4
			STD	**117.9894**	253.0275	637.852833	199.1058	375.70303	627.130744	325.749034
9	410	17	**Mean**	866.35	1003.4	2927.1	983.95	2892.75	10599.05	1325.1
			STD	221.4315	201.1979	212.070193	207.7379	449.840666	611.269465	**109.244069**
10	430	17	**Mean**	1178.25	1325.9	3723.55	1270.55	4417.7	12701.75	1798.85
			STD	269.3216	270.0505	541.578631	**241.2811**	561.6097	871.22099	266.459508

Table 5 demonstrates that the proposed algorithm produced the best mean values of the proposed model in the first three instances, with values of 23.475, 43.275, and 51.975. Additionally, the algorithm achieved the best STD value in six instances: 0.711512, 1.589025, 1.852363, 1.371587, 2.243045, and 1.864135, for (N=5, M=3), (N=*20*, M=4), (N=*25*, M=5), (N=*30*, M=5), (N=*40*, M=5), and (N=50, M=6), respectively, for small-sized problems. Table 6 and Table 7 show that the proposed algorithm achieved the best mean values in six instances, such as 110 and 336.15 for (N=*100*, M=9), and (N=*27*0, M=13), respectively, and the best STD values in four instances, such as 3.146427 and 79.19822 for (N=*100*, M=9), and (N=*33*0, M=15), respectively, compared to the other algorithms for medium and large-sized problems. The summarized results based on the mean criterion indicate that the superiority and efficiency of the proposed algorithm improves as the number of machines and jobs increases and low values of STD imply more consistency in the performance of the algorithm.

The best possible solutions for the proposed model are analyzed and illustrated in figures after 10 runs of each algorithm. It is guaranteed that the proposed algorithm will converge to the best domain solutions for each of the three size test problems. The provided graph samples illustrate the performance of the proposed algorithm in Figures 4, 5, and 6, on two instances of each size problem, clarifying the best convergence to the best solution and demonstrating that the proposed algorithm can stably converge when compared to other algorithms. Small-sized problems are illustrated in Figure 4, with (N=10, M=3) and (N=15, M=4); medium-sized problems are illustrated in Figure 5, with (N=60, M=8) and (N=220, M=12); and large-sized problems are illustrated in Figure 6, with (N=290, M=14) and (N=370, M=18). In all figures, the x-axis represents the number of iterations used in the comparison process, and the y-axis represents the function evaluation in terms of getting closer to minimum values. As the number of jobs and iterations increases, the proposed algorithm has a notable impact when approaching the best values. As shown in Figure 4, the proposed algorithm has a slight impact on getting close to the best values. Figures 5 and 6 demonstrate a significant impact, confirming the effectiveness of the proposed algorithm. This is evident from the smaller minimum values of the objective function, which expanded the convergence range and accelerated the algorithm toward the best feasible solutions. Additionally, the update process is guided to enhance the global search ability. As a result, the convergence results and quality of the solutions produced by the proposed algorithm improve as the number of jobs and iterations increase.

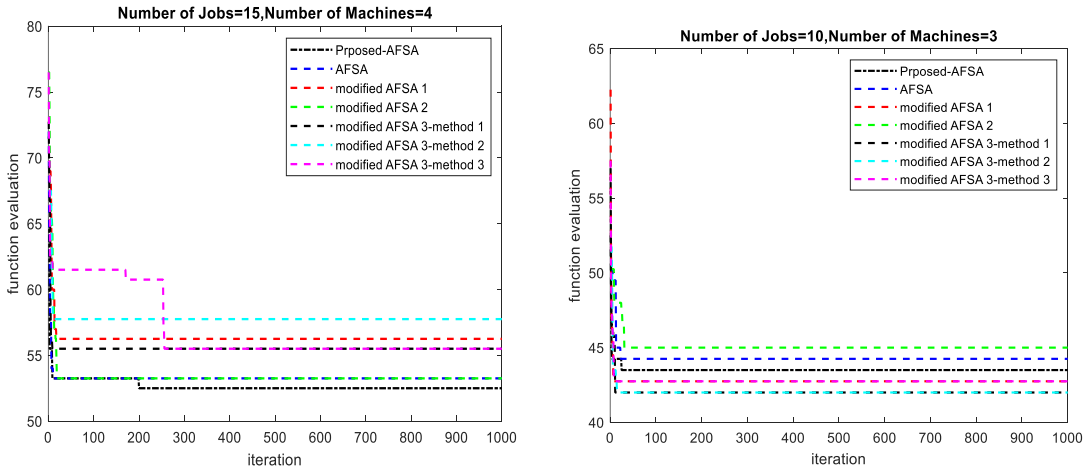

Figure 4. Performance of modified AFSA for small-size problems.

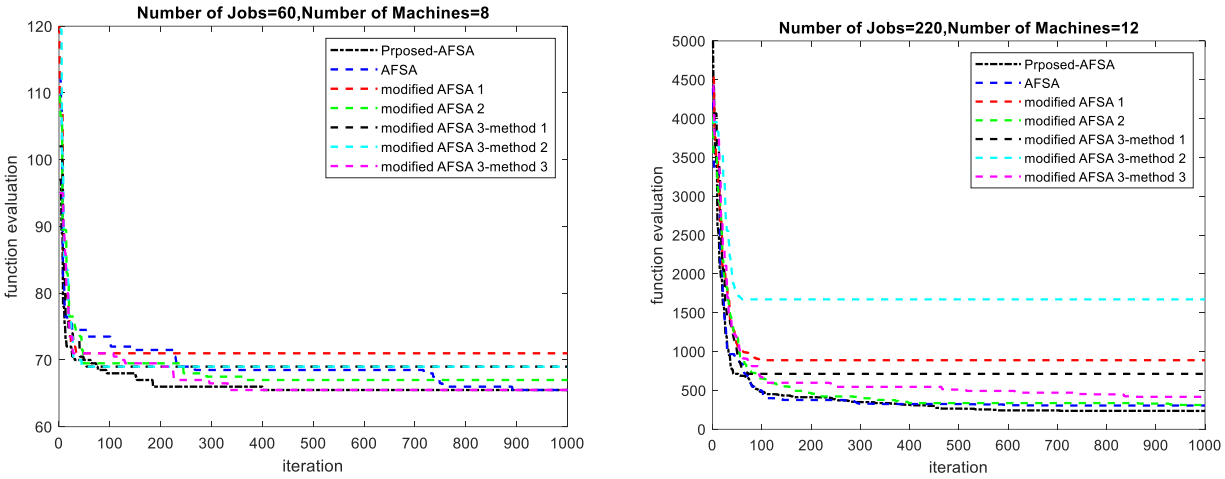

Figure 5. Performance of modified AFSA for medium-size problems.

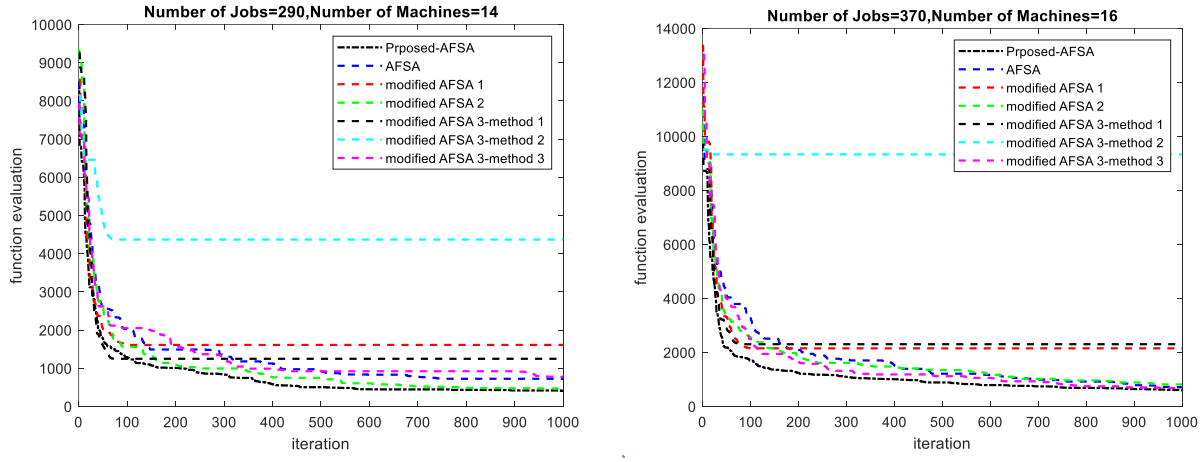

Figure 6. Performance of modified AFSA for large-size problems.

c) Comparison of results based on the Wilcoxon signed-ranks test

A common practice in computational intelligence is to use statistical tests as part of the performance evaluation process of a new method [54]. Therefore, the Wilcoxon signed-ranks test [55] is utilized in this research since it is non-parametric, meaning it does not assume a normal distribution of the data and is appropriate for small sample sizes. It is specifically formulated for pairwise comparisons to determine significant differences between the proposed algorithm and the algorithms under comparison, which is advantageous when evaluating two related samples. It is also robust, safe, and easy to use. Tables 11, 12, and 13 summarize the results for small, medium, and large instances, respectively, in comparison of the proposed algorithm with other AFSA versions

Table 11. Results of the Wilcoxon signed-rank test for small-size problems.

Problems		1	2	3	4	5	6	7	8	9	10
Proposed VS AFSA	***p*-value**	9.77E-02	1.95E-02	7.81E-02	1.86E-01	3.16E-01	1.39E-01	3.22E-01	4.92E-01	3.22E-01	2.93E-02
	R+	14.5	19	50.5	45	47	45	27	45	49	52
	R-	40.5	36	4.5	10	8	10	28	10	6	3
	Ind.	-1	-1	1	1	1	1	-1	1	1	1
Proposed VS Modified 1	***p*-value**	0.021484	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03
	R+	52	55	55	55	55	55	55	55	55	55
	R-	3	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 2	***p*-value**	0.578125	0.125	1.05E-01	8.28E-01	9.80E-01	9.96E-02	4.80E-01	9.22E-01	3.75E-01	3.71E-02
	R+	33.5	14.5	47	40	40	52	45	45	45	52
	R-	21.5	40.5	8	15	15	3	10	10	10	3
	Ind.	1	-1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 method 1	***p*-value**	0.001953	0.001953	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03
	R+	55	55	55	55	55	55	55	55	55	55
	R-	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 method 2	***p*-value**	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	R+	55	55	55	55	55	55	55	55	55	55
	R-	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 method 3	***p*-value**	0.375	0.003906	0.003906	0.013672	0.009766	0.001953	0.009766	0.019531	0.005859	0.009766
	R+	47	54	54.5	52	54	55	52	52	54	52
	R-	8	1	0.5	3	1	0	3	3	1	3
	Ind.	1	1	1	1	1	1	1	1	1	1

Table 12. Results of the Wilcoxon signed-rank test for medium-size problems.

Problems		1	2	3	4	5	6	7	8	9	10
Proposed VS AFSA	***p*-value**	1.00E+00	3.20E-01	2.71E-01	1.56E-02	2.34E-02	1.17E-02	1.13E-01	8.59E-01	9.49E-01	5.16E-01
	R+	32	50.5	49	18.5	14.5	14.5	30.5	41.5	42.5	26.5
	R-	23	4.5	6	36.5	40.5	40.5	24.5	13.5	12.5	28.5
	Ind.	1	1	1	-1	-1	-1	1	1	1	-1
Proposed VS Modified 1	***p*-value**	1	2.34E-01	2.85E-01	2.54E-02	2.54E-02	9.77E-02	4.16E-01	3.91E-03	1.56E-02	1.95E-03
	R+	36.5	45	50.5	52	54	50.5	40	54.5	54	55
	R-	18.5	10	4.5	3	1	4.5	15	0.5	1	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 2	***p*-value**	0.0625	0.449219	7.93E-01	2.73E-01	3.11E-01	6.64E-02	4.30E-02	1.95E-01	6.45E-01	9.77E-02
	R+	47.5	40	42.5	30.5	45	23	14.5	30.5	47	30.5
	R-	7.5	15	12.5	24.5	10	32	40.5	24.5	8	24.5
	Ind.	1	1	1	1	1	-1	-1	1	1	1
Proposed VS Modified 3 method 1	***p*-value**	0.5	0.1875	1.95E-01	3.52E-02	5.86E-03	5.86E-03	6.25E-02	1.17E-02	3.91E-03	1.95E-03
	R+	37	48.5	48.5	53	54	54	49	54	54	55
	R-	18	6.5	6.5	2	1	1	6	1	1	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified3 method 2	***p*-value**	0.5	0.128906	0.015625	0.015625	0.001953	0.001953	0.003906	0.007813	0.001953	0.001953
	R+	40.5	50.5	54	53	55	55	54.5	53	55	55
	R-	14.5	4.5	1	2	0	0	0.5	2	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 method 3	***p*-value**	1	0.6875	0.929688	0.039063	0.0625	0.001953	0.027344	0.46875	0.425781	0.080078
	R+	36.5	41.5	42.5	23	19	0	19	37	42.5	54
	R-	18.5	13.5	12.5	32	36	55	36	18	12.5	1
	Ind.	1	1	1	-1	-1	-1	-1	1	1	1

Table 13. Results of the Wilcoxon signed-rank test for large-size problems.

Problems		1	2	3	4	5	6	7	8	9	10
Proposed VS AFSA	***p*-value**	1.93E-01	1.37E-02	1.31E-01	1.93E-01	6.25E-01	1.95E-03	7.70E-01	5.86E-03	1.60E-01	3.75E-01
	R+	45	54	49	49	45	55	34	54	52	45
	R-	10	1	6	6	10	0	21	1	3	10
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 1	***p*-value**	0.001953	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03
	R+	55	55	55	55	55	55	55	55	55	55
	R-	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 2	***p*-value**	0.769531	0.005859	9.22E-01	4.92E-01	2.75E-01	3.91E-03	4.32E-01	1.93E-01	5.57E-01	6.95E-01
	R+	49	54	45	45	45	54	49	49	45	45
	R-	6	1	10	10	10	1	6	6	10	10
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 method 1	***p*-value**	0.001953	0.001953	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.95E-03
	R+	55	55	55	55	55	55	55	55	55	55
	R-	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 method 2	***p*-value**	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953
	R+	55	55	55	55	55	55	55	55	55	55
	R-	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 method 3	***p*-value**	0.001953	0.001953	0.001953	0.001953	0.001953	0.001953	0.013672	0.001953	0.001953	0.001953
	R+	55	55	55	55	55	55	54	55	55	55
	R-	0	0	0	0	0	0	1	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1

Tables 11, 12, and 13 present the results of the Wilcoxon signed-rank test obtained by the proposed algorithm, AFSA, Modified 1, Modified 2, Modified 3-method 1, Modified 3-method 2, and Modified 3- method 3. The *p*-value is the assessment index between zero and one, carried out with a significance level of 0.05. R+ represents the total ranking indicating that the first method is superior to the second one, while R− represents the total ranking indicating that the second method is superior to the first one. Indicate (Ind) denotes a difference in performance scores between the two algorithms; if Ind = +1, the proposed modified AFSA is superior to the other; if Ind = -1, the other algorithm is superior to the proposed modified AFSA; and if Ind = 0, the performance of the two algorithms is equal.

Subsequently, Table 11 shows that the *p*-value in most pairwise comparisons is lower than the significance level of 0.05. For instance, the *p*-value in all ten instances in the comparison between the proposed algorithm and AFSA is lower than 0.05, which proves that the proposed algorithm outperforms AFSA. The comparison results with other algorithms based on the Wilcoxon signed-ranks test demonstrate that, except for the results with AFSA for small-size problems and the results with AFSA and Modified 3-method 3 for medium-size problems, the proposed algorithm outperforms the other compared algorithms and has great significant differences in performance. Furthermore, the proposed algorithm outperforms all compared algorithms and shows great significant differences for large-sized problems in each pairwise comparison.

To sum up the discussions above, it is evident that the proposed algorithm's effectiveness stems from leveraging AFSA's advantages, including its multi-stage solution updates and search domain exploration. The algorithm enhances the exploitation and exploration phases by utilizing the best solution extracted from AFSA. Additionally, by adapting improved visual and step parameters, it balances global search capability with convergence speed and effectively applies AFSA to machine scheduling problems. However, the proposed algorithm has limitations, particularly in achieving optimal results for machines 4 to 10. Thus, improvements are necessary to solve small-sized jobs and machine problems. While the stability of the proposed algorithm is commendable, further enhancements are required.

7. Conclusions and future research

This research adopts UPMSP to minimize multi-objective makespan and total tardiness. A fuzzy programming model is presented, which is transformed into a deterministic model using the total integral value defuzzification method. The proposed algorithm is developed by adding aspiration behavior to improve the performance of AFSA. Visual and step are crucial parameters for optimizing AFSA; thus, improved visual and step parameters are adopted into the modified algorithm. Additionally, transforming the continuous solution of AFSA into a discrete solution enables the algorithm to handle the discrete model of UPMSP. Three sets of problem sizes, each including 10 randomly generated instances, are used to evaluate the performance of the algorithm. The proposed algorithm's best performance metrics on the test problems are determined by the minimum values in the proposed minimization problem. The first three instances provide good performance for small-sized problems with up to 50 jobs and up to 7 machines, whereas the best performance is achieved in nine instances for medium and large-sized problems with up to 240 jobs and 12 machines, and up to 430 jobs and 17 machines, respectively. According to computational results, the proposed algorithm improves the exploitation capability of the AFSA and demonstrates superior performance compared to the AFSA and other modified AFSA variants in identifying the best solutions for almost all test

problems. The Wilcoxon signed-rank test supports the conclusion that the proposed algorithm significantly outperforms the other algorithms. Based on the good performance of the proposed modified algorithm, it might be used in the future to solve more complex scheduling problems, such as job shop machine scheduling or flow shop scheduling, as well as various performance measures.

Author contributions

Azhar Mahdi Ibadi conceptualized the research methodology and led the development of the algorithm and programming, focusing on results analysis and writing the initial draft. Rosshairy Abd Rahman led the supervision and validation, thoroughly checking it for significant conceptual content. All authors participated in carefully editing the manuscript. All authors have read and approved the final version of the manuscript for publication.

Acknowledgments

The authors are deeply grateful to the School of Quantitative Sciences, Universiti Utara Malaysia, for their invaluable support in facilitating this research.

Conflict of interest

The authors declare no conflicts of interest.

Use of AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

References

1. K. Salimifard, D. Mohammadi, R. Moghdani, A. Abbasizad, Green fuzzy parallel machine scheduling with sequence-dependent setup in the plastic moulding industry, *Asian J. Manag. Sci. Appl.,* **4** (2019), 27–48. https://doi.org/10.1504/AJMSA. 2019.101423
2. M. R. Garey, D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, *Freeman, San Francisco,* (1979). https://doi.org/10.1137/1024022
3. A. E. Ezugwu, A. K. Shukla, R. Nath, A. A. Akinyelu, J. O. Agushaka, H. Chiroma, et al., Metaheuristics: A comprehensive overview and classification along with bibliometric analysis. *Artif. Intell. Rev.,* **54** (2021), 4237–4316. https://doi.org/10.1007/s10462-020-0952-0
4. D. Y. Lin, T. Y. Huang, A hybrid metaheuristic for the unrelated parallel machine scheduling problem, *Mathematics*, **9** (2021). 1–20. https://doi.org/10.3390/math9070768
5. A. E. Ezugwu, F. Akutsah, An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times, *IEEE Access*, **6** (2018), 54459–54478. https://doi.org/10.1109/access.2018.2872110

6. C. Chen, M. Fathi, M. Khakifirooz, K. Wu, Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times, *Comput. Ind. Eng.*, **165** (2022), 107915. https://doi.org/10.1016/j.cie. 2021.107915
7. J. B. Abikarram, K. McConky, R. Proano, Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing, *J. Clean. Prod.,* **208** (2019), 232–242. https://doi.org/10.1016/j.jclepro.2018.10.048
8. N. Vakhania, On preemptive scheduling on unrelated machines using linear programming, *AIMS Math.,* **8** (2023), 7061–7082. https://doi.org/10.3934/math.2023356
9. Y. Fu, Y. Hou, Z. Wang, X. Wu, K. Gao, L. Wang, Distributed scheduling problems in intelligent manufacturing systems, *Tsinghua Sci. Technol.*, **26** (2021), 625–645. https://doi.org/10.26599/tst.2021.9010009
10. İ. Sarıçiçek, Multi-objective scheduling by maximizing machine preferences for unrelated parallel machines, *Sigma J. Eng. Nat. Sci.*, **38** (2020), 405–419. https://doi.org/10.5829/idosi.ije.2017.30.02b.09
11. R. Meng, Y. Rao, Q. Luo, Modeling and solving for bi-objective cutting parallel machine scheduling problem, *Ann. Oper. Res.*, **285** (2020), 223–245. https://doi.org/10.1007/s10479-019-03208-z
12. M. Moser, N. Musliu, A. Schaerf, F. Winter, Exact and metaheuristic approaches for unrelated parallel machine scheduling, *J. Scheduling*, **25** (2022), 507–534. https://doi.org/10.1007/s10951-021-00714-6
13. M. Đurasević, D. Jakobović, Heuristic and metaheuristic methods for the unrelated machines scheduling problem: A Survey, *IEEE T. Cybernetics,* (2021). http://arxiv.org/abs/2107.13106
14. A. F. Guevara-Guevara, V. Gómez-Fuentes, L. J. Posos-Rodríguez, N. Remolina-Gómez, E. M. González-Neira, Earliness/tardiness minimization in a no-wait flow shop with sequence-dependent setup times, *J. Proj. Manag.,* **7** (2022). 177–190. https://doi.org/10.5267/j.jpm.2021.12.001
15. A. Sadati, R. Tavakkoli-Moghaddam, B. Naderi, M. Mohammadi, A bi-objective model for a scheduling problem of unrelated parallel batch processing machines with fuzzy parameters by two fuzzy multi-objective meta-heuristics, *Iran. J. Fuzzy Syst.*, **16** (2019), 21–40. https://doi.org/10.3233/ifs-151846
16. G. Rivera Zarate, Outranking-based multi-objective PSO for scheduling unrelated parallel machines with a freight industry-oriented application, *Instituto de Ingeniería Tecnología*. (2021). https://doi.org/10.1016/j.engappai.2021.104556
17. A. Fallahi, B. Shahidi-Zadeh, S. T. A. Niaki, Unrelated parallel batch processing machine scheduling for production systems under carbon reduction policies: NSGA-II and MOGWO metaheuristics, *Soft Comput.*, **27** (2023), 17063–17091. https://doi.org/10.1007/s00500-023-08754-0
18. W. Zhou, F. Chen, X. Ji, H. Li, J. Zhou, A Pareto-based discrete particle swarm optimization for parallel casting workshop scheduling problem with fuzzy processing time, *Knowl. Based Syst.*, (2022), 256, 109872. https://doi.org/10.1016/j.knosys.2022.109872
19. M. Z. Erişgin Barak, M. Koyuncu, Fuzzy order acceptance and scheduling on identical parallel machines, *Symmetry*, **13** (2021), 1236. https://doi.org/10.3390/sym13071236

20. J. Rezaeian, S. Mohammad-Hosseini, S. Zabihzadeh, K. Shokoufi, Fuzzy scheduling problem on unrelated parallel machine in JIT production system, *Artif. Intell. Evol.*, **1** (2020), 17–33. https://doi.org/10.37256/aie.112020202
21. K. Li, J. Chen, H. Fu, Z. Jia, W. Fu, Uniform parallel machine scheduling with fuzzy processing times under resource consumption constraint, *Appl. Soft Comput.*, **82** (2019), 1–13. https://doi.org/10.1016/j.asoc.2019.105585
22. H. A. Alsattar, A. A. Zaidan, B. B. Zaidan, Novel meta-heuristic bald eagle search optimisation algorithm, *Artif. Intell. Rev.*, **53** (2020), 2237–2264. https://doi.org/10.1007/s10462-019-09732-5
23. F. Pourpanah, R. Wang, C. P. Lim, X. Z. Wang, D. Yazdani, A review of artificial fish swarm algorithms: Recent advances and applications, *Artif. Intell. Rev.,* **56** (2020), 1867–1903. https://doi.org/10.1007/s10462-022-10214-4
24. L. Zhao, F. Wang, Y. Bai, Route planning for autonomous vessels based on improved artificial fish swarm algorithm, *Ships Offshore Struc.*, **18** (2023), 897–906. https://doi.org/10.1080/17445302.2022.2081423
25. W. H. Tan, J. Mohamad-Saleh, Normative fish swarm algorithm (NFSA) for optimization, *Soft Comput.,* **24** (2020), 2083–2099. https://doi.org/10.1007/s00500-019-04040-0
26. J. Huang, J. Zeng, Y. Bai, Z. Cheng, Z. Feng, L. Qi, et al., Layout optimization of fiber Bragg grating strain sensor network based on modified artificial fish swarm algorithm, *Opt. Fiber Technol.,* **65** (2021), 102583. https://doi.org/10.1016/j.yofte.2021.102583
27. F. Wang, L. Zhao, Y. Bai, Path planning for unmanned surface vehicles based on modified Artificial Fish Swarm Algorithm with local optimizer, *Math. Probl. Eng.*, (2022). https://doi.org/10.1155/2022/1283374
28. J. Jin, Z. Zhang, L. Zhang, A modified artificial fish swarm algorithm for unit commitment optimization, *in 2023 Eighth International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, 760–767. https://doi.org/10.1117/12.2689449
29. Y. Gao, L. Xie, Z. Zhang, Q. Fan, Twin support vector machine based on improved artificial fish swarm algorithm with application to flame recognition, *Appl. Intell.,* **50** (2020), 2312–2327. https://doi.org/10.1007/s10489-020-01676-6
30. T. Li, F. Yang, D. Zhang, L. Zhai, Computation scheduling of multi-access edge networks based on the artificial fish swarm algorithm, *IEEE Access*, **9** (2021), 74674–74683. https://doi.org/10.1109/ACCESS.2021.3078539
31. E. B. Tirkolaee, A. Goli, G. W. Weber, Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option, *IEEE T. Fuzzy Syst.*, **28** (2020), 2772–2783. https://doi.org/10.1109/TFUZZ.2020.2998174
32. P. Kongsri, J. Buddhakulsomsiri, 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)*, 605–609. https://doi.org/10.1109/iciea49774.2020.9102086
33. R. L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals Discrete Math*., **5** (1979), 287–326. https://doi.org/10.1016/s0167-5060(08)70356-x

34. Y. Li, J. F. Côté, L. Callegari-Coelho, P. Wu, Novel formulations and logic-based benders decomposition for the integrated parallel machine scheduling and location problem, *INFORMS J. Comput.*, **34** (2022), 1048–1069. https://doi.org/10.1287/ijoc.2021.1113
35. Y. Li, J. F. Côté, L. C. Coelho, P. Wu, Novel efficient formulation and matheuristic for large-sized unrelated parallel machine scheduling with release dates, *Int. J. Prod. Res.*, **60** (2022), 6104–6123. https://doi.org/10.1080/00207543.2021.1983224
36. S. Chakraverty, D. M. Sahoo, N. R. Mahato, Defuzzification, *In Concepts of Soft Computing, Springer Singapore,* 2019, 117–127. https://doi.org/10.1007/978-981-13-7430-2_7
37. L. A. Zadeh, Fuzzy sets, *Inf. Control*, **8** (1965), 338–353. https://doi.org/10. 1016/s0019-9958(65)90241-x
38. H. J. Zimmermann, Fuzzy set theory, *WIRES Comput. Stat.*, **2** (2010), 317–332. https://doi.org/10.1002/wics.82
39. H. T. Lee, S. H. Chen, H. Y. Kang, Multicriteria scheduling using fuzzy theory and tabu search, *Int. J. Prod. Res.,* **40** (2002), 1221–1234. https://doi.org/10.1080/ 00207540110098832
40. S. Banerjee, T. K. Roy, Arithmetic operations on generalized trapezoidal fuzzy number and its applications, *Turkish J. Fuzzy Syst.,* **3** (2012), 16–44.
41. J. Shen, An uncertain parallel machine problem with deterioration and learning effect, *Comput. Appl. Math*., 38 (2019), 1–17. https://doi.org/10.1007/s40314-019-0789-5
42. T. W. Liao, P. Su, 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, *Appl. Soft Comput. J.*, **56** (2017), 65–81. https://doi.org/10.1016/j. asoc.2017.03.004
43. T. S. Liou, M. J. Wang, Ranking fuzzy numbers with integral value, *Fuzzy Sets Syst.*, **50** (1992), 247–255. https://doi.org/10.1016/0165-0114(92)90223-q
44. X. L. Li, An optimizing method based on autonomous animats: fish-swarm algorithm, *Syst. Eng. Theory Practice*, **22** (2002), 32–38.
45. A. W. Abdulqader, S. M. Ali, Diversity operators-based Artificial Fish Swarm Algorithm to solve flexible job shop scheduling problem, *Baghdad Sci. J.,* (2023). https://doi.org/10.21123/bsj.2023.6810
46. L. Zhang, M. Fu, T. Fei, X. Pan, Application of FWA-Artificial Fish Swarm Algorithm in the Location of Low-Carbon Cold Chain Logistics Distribution Center in Beijing-Tianjin-Hebei Metropolitan Area, *Sci. Programming*, (2021), 1–10. https://doi.org/10.1155/2021/9945583
47. Y. Liu, X. Feng, L. Zhang, W. Hua, K. Li, A pareto artificial fish swarm algorithm for solving a multi-objective electric transit network design problem, *Transportmetrica A,* **16** (2020),1648–1670. https://doi.org/10.1080/23249935.2020.1773574
48. S. Gorgich, S. Tabatabaei, Proposing an energy-aware routing protocol by using fish swarm optimization algorithm in WSN (wireless sensor networks), *Wireless Pers. Commun.,* **119** (2021), 1935–1955. https://doi.org/10.1007/s11277-021-08312-7
49. R. A. Hasan, R. A. I. Alhayali, M. A., Mohammed, T. Sutikno, An improved fish swarm algorithm to assign tasks and cut down on latency in cloud computing, *TELKOMNIKA (Telecommunication Computing Electronics and Control),* **20** (2022), 1103–1108. https://doi.org/10.12928/telkomnika.v20i5.22645
50. N. M. Sureja, S. P. Patel, Solving a combinatorial optimization problem using artificial fish swarm algorithm, *Int. J. Eng. Trends Technol*, **68** (2020), 27–32. https://doi.org/10.14445/22315381/ijett-v68i5p206s

51. M. Yadollahi, S. S. Razavi, Using Artificial Fish Swarm Algorithm to solve university exam timetabling problem, *J. Adv. Comput. Res.*, **10** (2019), 109–117.
52. C. Peraza, P. Ochoa, L. Amador, O. Castillo, Artificial Fish Swarm Algorithm for the Optimization of a Benchmark Set of Functions, In New Perspectives on Hybrid Intelligent System Design based on Fuzzy Logic, Neural Networks and Metaheuristics, *Studies in Computational Intelligence*, **1050** (2022), (77–92). https://doi.org/10.1007/978-3-031-08266-5_6
53. S. P. M. Villablanca, An artificial fish swarm algorithm to solve the set covering problem doctoral dissertation, pontificia universidad católica de valparaíso, (2016).
54. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol Comput*, **1** (2011), 3–18. https://doi.org/10.1016/j.swevo.2011.02.002
55. F. Wilcoxon, Individual comparisons by ranking methods, *Biometrics Bulletin*, **1** (1945), 80–83. https://doi.org/10.2307/3001968

AIMS Press