

AFSO-JSSP: Artificial Fish Swarm Optimization for Efficient Job Shop Scheduling

Shunmuga Priya Subramanian

Ramco Institute of Technology, Rajapalayam – 626117, Tamil Nadu, India
E-mail: shunmugapriya@ritrjpm.ac.in
ORCID iD: <https://orcid.org/0000-0002-4802-1367>

Muthiah Athi

P.S.R. Engineering College, Sivakasi, Tamil Nadu, India
E-mail: amuthiah68@gmail.com
ORCID iD: <https://orcid.org/0000-0002-5133-2860>

Pitchipoo Pandian*

P.S.R. Engineering College, Sivakasi, Tamil Nadu, India
E-mail: drpitchipoo@gmail.com
ORCID iD: <https://orcid.org/0000-0002-2850-6084>

*Corresponding author

Rajakarunakaran Sivaprakasam

Ramco Institute of Technology, Rajapalayam – 626117, Tamil Nadu, India
E-mail: rajakarunakaran@ritrjpm.ac.in
ORCID iD: <https://orcid.org/0000-0002-1341-9658>

Received: November 7, 2025; Revised: December 20, 2025; Accepted: February 2, 2026; Published: August 8, 2026

Abstract: The Job Shop Scheduling Problem (JSSP) has become one of the key issues in a contemporary manufacturing system in which the task is to optimally schedule jobs to the machines to reduce the time and resources used in production. Good scheduling is critical in enhancing the productivity and competitiveness of manufacturing industries. In this research, Artificial Fish Swarm Optimization (AFSO) algorithm is used to optimize the JSSP in minimizing makespan, total work load and maximum work load in machines. The AFSO strategy models the swarm behavior of fish to search and forage the search space in an efficient manner to prevent its early convergence to local optima. The model incorporates a perturbed global state to improve the direction in search and the speed of convergence. The effectiveness of the suggested AFSO method is compared and tested with the traditional and sophisticated optimization algorithms like Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) methods. The experimental findings prove that the proposed technique provides better results in convergence rate and solution quality. The results prove that AFSO is a useful and promising method of solving complex problems in production system scheduling.

Index Terms: Scheduling, JSSP, Artificial Fish Swarm Optimization, Multi-Objective Optimization.

1. Introduction

In the age of the Industry 4.0 and intelligent manufacturing, optimization of the production schedule has become one of the most important challenges in order to increase productivity, decrease the cost and deliver the products in time [1]. The JSSP is one of the most complicated and the most discussed scheduling problems because it is a problem of combinatorial (and therefore wide industry) application [2, 3]. The JSSP is a process that is concerned with assigning a group of jobs to machines, where each job must adhere to a particular sequence of operations without conflicting with resource constraints or precedence. The main goal is to reduce the parameters like makespan, total workload and maximum workload that have great impact on the performance of manufacturing and the throughput efficiency [4]. Given that JSSP is an NP-hard problem, it becomes computationally intensive to compute the best solutions using either standard mathematical programming or exact methods as the problem size grows.

JSSP has been studied using traditional methods like branch and bound, dynamic programming and dispatching rule which in most cases fail to provide satisfactory results when used in large scale or dynamic systems [5]. As the sphere of artificial intelligence develops, metaheuristic algorithms provide good alternatives to the creation of complex optimization problems. Inspired by natural physics algorithms such as Genetic Algorithms (GA) [6], Simulated Annealing (SA) [7], Ant Colony Optimization (ACO) [8], Particle Swarm Optimization (PSO) [9], Artificial Bee Colony (ABC) [10], bat algorithm [11], social spider optimization (SSO) [12], and hybrid approaches [13-15] have been successful in exploring high-dimensional solution space and have produced near-optimal results with scheduling. As an example, Wang et al. [9] have shown that PSO is effective at decreasing makespan in flexible manufacturing systems; although it tends to get stuck in local optima. In the same regard, Karaboga and Basturk [16] emphasized that the ABC algorithms have high exploration degree but low exploitation efficiency to optimize candidate solutions.

The AFSO algorithm has been proposed as an alternative to address these limitations in solving the JSSP. AFSO replicates the social and foraging activities of fish swarms including preying, swarming, and following to find a balance between exploration of the global and exploitation of the local [17]. This research incorporates a perturbed global state in the AFSO model in order to improve its convergence rate and search flexibility. The chosen AFSO-based solution is supposed to reduce the makespan, overall workload, and maximum workload without premature convergence and population diversity.

The proposed algorithm is compared to the existing metaheuristic techniques and especially PSO and ABC to measure the performance of the algorithm. The experimental data proves that the AFSO can reach better convergence rate and quality of the solutions. Therefore, the study is relevant to the evolution of smart scheduling tools, as it can provide a solid and effective optimization platform to help face the complicated industrial scheduling problems.

2. Literature Review

Job-shop and flexible job-shop scheduling studies have placed more emphasis on modelling and optimizing uncertainty to enhance metaheuristic search to achieve a robust and quality schedule in the presence of realistic production disruptions. Xie and Chen [18] investigated the flexible job-shop scheduling problem of interval grey processing time to reflect the vagueness in the processing time and found that grey-based modeling yielded more accurate schedules where accurate information is not available. They emphasize the relevance of models that are uncertain-aware to manufacturing systems in their approach. Similarly, Ahmadi et al. [19] tackled the FJSSP in the context of random machine breakdowns with multi-objective evolutionary algorithms, providing evidence that, when stochastic disruptions are considered in the formulation of the objectives, schedules that better trade off makespan, workload balance and resilience are obtained. On this basis, there are more recent results that have integrated uncertainty modeling and sophisticated metaheuristics to enhance both the quality and robustness of the solutions. Recently, a flexible job-shop scheduling algorithm that models processing-time uncertainty by the use of generalized grey numbers was proposed and has shown better makespan performance in the presence of interval uncertainty in comparison to grey operations [20]. Simultaneously, thorough surveys of the FJSP literature underline that hybrid and population-based metaheuristics (e.g., PSO, GA, ACO, ABC) are also one of the most efficient to be used with large and dynamic problems; however, they frequently need problem-specific improvements to prevent early convergence and to tackle uncertainty [21].

As a swarm intelligence method, AFSO has experienced a revival of interest because of its adaptive neighborhood interactions and behavioral operators (preying, swarming, following). Recent surveys and methodological papers have summarized the betterment of AFSO, such as diversity operators, normative variants, and hybridizations, which expand exploration, preserve population diversity, and yield higher convergence reliability than their base implementations. This suggests that variants of AFSO can rival or do better than traditional metaheuristics when applied to discrete encodings of the scheduling problem [22]. As an example, both variants of diversity-operator-enhanced AFSO applied to flexible job-shop problems yield improved avoidance of local optima and greater spread of solutions, which is particularly useful when multi-objective formulations are used (including workload balance and robustness measures) [23]. Lastly, the move towards hybrid metaheuristics remains: recent 2023-2024 works propose hybrid schemes (e.g., evolutionary techniques with local search or simulated annealing) with faster convergence and stronger makespan performance on benchmark JSSP problems, which supports the hypothesis that global search (population-based) and intensification (local search or perturbation) can work well as a design principle of scheduling optimizers [24]. Kourepinis et al. [25] developed a discrete version of Urban Transit Routing Problem (UTRP), a complicated combinatorial optimization problem, that offers a route optimization and minimizes the cost, using the AFSO algorithm. Their model suitably models the fish behaviors within a discrete search space by specially designed movement operators and route encodings. Comparisons with fourteen metaheuristics provided in the form of experimental comparisons showed that the suggested AFSO had better solution quality and faster convergence. Despite the fact that it was developed in the context of the urban transit systems, this study lays the foundation of the potential of the AFSO in the discrete scheduling areas and it will provide the methodological basis applicable in the optimization of job shop scheduling.

Based on the synthesis of earlier work, it can be seen that AFSO-based procedures would be highly appropriate in

addressing the dynamic quality of today's production scheduling. Nonetheless, a research gap still exists in the AFSO application in classical JSSP environments where makespan, total workload, and maximum workload must be optimized simultaneously. Local trapping can be overcome and convergence can be accelerated by the incorporation of a perturbed global condition and adaptive movement strategies. Based on these observations, this paper proposes a better AFSO-based methodology that reduces the total workload, maximum workload and the makespan of the loaded machine for JSSP.

3. Proposed Model

For the JSSP, knowledge of various swarm optimization models is taken into account. Every job in JSSP is carried out in a particular order on machines with a processing time. Each machine in this approach processes a single work at a time. Typically, 'M' machines and 'N' jobs are used for scheduling purposes. Different routes or sequences are used to process the jobs under consideration. Therefore, the amount of machines, jobs, and job sequences determines how difficult it is to schedule jobs.

3.1. Objective Function for JSSP

Three goals, machine workload, overall workload, and makespan time were taken into account in the proposed JSSP process. It is represented by and, accordingly, for the research, a pool of machines and jobs M_i and J_i which are represented in equation (1) & (2).

$$M_i = \{M_1, M_2, M_3, \dots, M_n\} \quad (1)$$

$$J_i = \{J_1, J_2, J_3, \dots, J_n\} \quad (2)$$

Every work in JSSP is made up of a series of operations $\{O_{1j}, O_{2j}, O_{3j}, \dots, O_{nj}\}$ of which must be carried out for the job to be finished. One machine from a set of specified machines is needed for each process to be completed. Every objective function is minimized, with the following constraints:

$$\text{Makespan Time: } \min F1 = \max_{1 \leq i, k \leq n} \{T_{j^*m_time}\} \quad (3)$$

$$\text{Total Workload: } F2 = \sum_{ink=1}^{m,n,n_i} T_{ink} * P_{ijk} \quad (4)$$

$$\text{Maximum workload: } \max F2 = \max_{1 \leq k \leq n} \sum_{in=1}^{m,n} T_{ink} * P_{ijk} \quad (5)$$

where, ' m or M ' denotes the number of machines; ' n or N ' denotes the number of Jobs; ' k_n ' denotes the number of the machines in k^{th} job; ' O_{ij} ' is the operation of ' n ' machine with ' n ' jobs; ' P_{ink} ' is the processing time of each job with each machine, and ' T_{j^*m} ' denotes the job completion time for each ' O_{ij} '.

The task assignment sub-problem assigns each operation to the proper machine for simple analysis. A sequence of actions on each machine is calculated in one of the sub-problems related to sequencing. All jobs ' N ' are scheduled for processing at time zero if all machines ' M ' are available at that moment. No restrictions on the order in which different work procedures can be completed. Every machine has a predetermined processing period for its processes, and during that time, there are very few machine interruptions. One of the most recent combinatorial optimization challenges is JSSP. Its scheduling problem is computed using the order of operations performed on machines that demand the least amount of processing time to complete the job. The number of machines is represented by the symbol ' M ', and the number of distinct jobs that need to be planned is represented by the sign ' J '. Every job has several steps that need to be completed on certain equipment within a predetermined amount of time.

3.2. Suggested Swarm Improvements

Swarm intelligence optimization is a metaheuristic technique that operates on the notion of swarm behavior. The introduced approach finds the best solution to the limited problem by encouraging collaboration and competition among initialized individuals. The swarm intelligence optimization approach that is being described is a novel algorithm that has received significant attention. In this research, AFSO, a swarm based optimization technique is applied for JSSP. Because the JSSP is a combinatorial optimization problem, the continuous position representation of AFSO is converted to a discrete representation. In this work, any artificial fish is a possible job sequence, that is, the processing sequence of operations on machines. A fish position is represented as a one-dimensional permutation vector of job indices, in which each job appears as many times as the number of its operations. The listing of the job indices in the order dictates how the respective operations will be carried out, within the established technological limitations of the job. When decoding, the order is searched through left to right, and the subsequent job that will not be scheduled is allocated to the demanded machine at the latest opportunity possible. This encoding is such that all precedence constraints are maintained and all

fish positions are associated to valid schedules. Permutation-based operators as swap, insertion and exchange sequences are used to explore the solution space efficiently by means of discrete fish behaviors, like preying, swarming and following.

A. Artificial Fish Swarms Optimization (AFSO)

In a swarm, a fish normally continually adjusts its position to the surrounding conditions and its state. The movements of each fish seem random, but in reality, they are incredibly coordinated as they travel toward the goal, as shown by the AFSO flowchart in Fig. 1. Fish adhere closely to their swarms to avoid collisions with other fish and to defend themselves from predators while continuously searching for food [26]. These actions were sparked by the development of novel algorithms that can reasonably efficiently tackle optimization problems. The global optimal solution of the JSSP makespan time analysis is obtained by the algorithm through the employment of artificial fishes, which are individual fishes that explore locally.

In the new AFSO approach for JSSP, a solution refers to a workable schedule that allocates the sequential arrangement of all activities on machines and fulfills the technological precedence limitations and respects machine capacity constraints. Makespan minimization objective is used to assess the quality of a solution. The flow diagram shown in Fig. 2 establishes the solution's fitness value. This unique model and the JSSP process, which are described in detail in the section below, consider the important stages such as prey behavior, fish swarm behavior, and fish follow behavior.

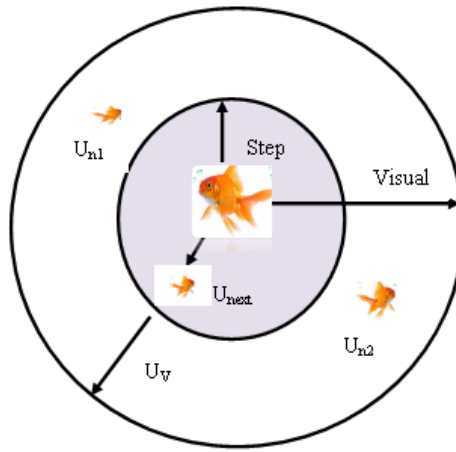


Fig. 1. Concept of AFS algorithm.

(i) Solution Initialization

Equation (6) shows how long the JSSP takes to process, starting with the magnitude of the benchmark jobs and integer values.

$$M_{in} = \{rand(P_i, 1) * (Ul - lb) + ll\} \quad (6)$$

(ii) Fish Prey behavior

The fitness value of each fish serves as a proxy for the sequence of food-seeking behavior. The fish is in the more promising region of the search space if the measured value is high, which corresponds to a high fitness value. The fish will move swiftly toward a body of water that has more food when they find it. Assume M_i, I_j for the moment it is self-assuredly within its detecting range.

$$\text{Prey: } M_j = M_i + \text{visual.rand} \quad (7)$$

$$M_i^{(t+1)} = I_i^{(t)} + \frac{M_j - M_i^{(t)}}{\|M_j - M_i^{(t)}\|} \cdot \text{step.rand} \quad (8)$$

Target capability esteem is the expression used to describe the fish's fixation on sustenance in this posture, $d_{i,j} = \|M_i - M_j\|$. There is a division between the artificial and random fish here 'i' and 'j'. Where Step indicates the largest stride size of the produced fish and Where generates random integers between 0 and 1. The fake fish appears just inside the hover's inward range, extending to the length of the field of vision, as indicated by the distinct acts in equation (8).

(iii) Fish Swarm Behavior

Naturally, swarming ensures the colony's survival and keeps threats at bay. Similar to this, fish swarm behavior involves the assembly of fish in groups during movement. Assume ' M_i ' that the present condition of the AF, its center of gravity M_c , and the amount of its neighbors in the neighborhood fish are all counterfeit fish $M_i (D_{i,j} < Visual)$. The swarm's behavior (F1, F2 and F3) is updated based on which fitness functions are greatest and lowest.

(iv) Fish Follow Behavior

When fish locates food, it moves to follow its neighbors and eventually reaches the place where it is possible. The fish's condition is examined from its visual neighbors to determine its optimum state (M_{max}). The number, if $D_{i,j} < Visual$ of partners indicates that the area around the fish has more food and is less congested, which also allows the fish to advance to the front of the group M_{max} . Commonly, investigating behavior is carried out using condition (9).

$$\text{Swarm: } M_i^{(t+1)} = M_i^{(t)} + \frac{M_c - M_i^{(t)}}{\|M_c - M_i^{(t)}\|} \cdot \text{step} \cdot \text{rand} \quad (9)$$

(v) Move and Leave Process

In reality, fish move randomly in the water because they are searching over wider areas for companions or food. The departure protocol stops somewhere in the water; all AFs will behave in the same way, and after a few cycles, the difference between the objective values (food concentration, FC) becomes less pronounced. This could lead to a localized extreme where the parameters are randomly changed to the [25] still states for the jumping outflow state.

$$\text{Leave and Move: } M_{some}^{(t+)} = M_{some}^{(t)} + \alpha * \text{rand} \quad (10)$$

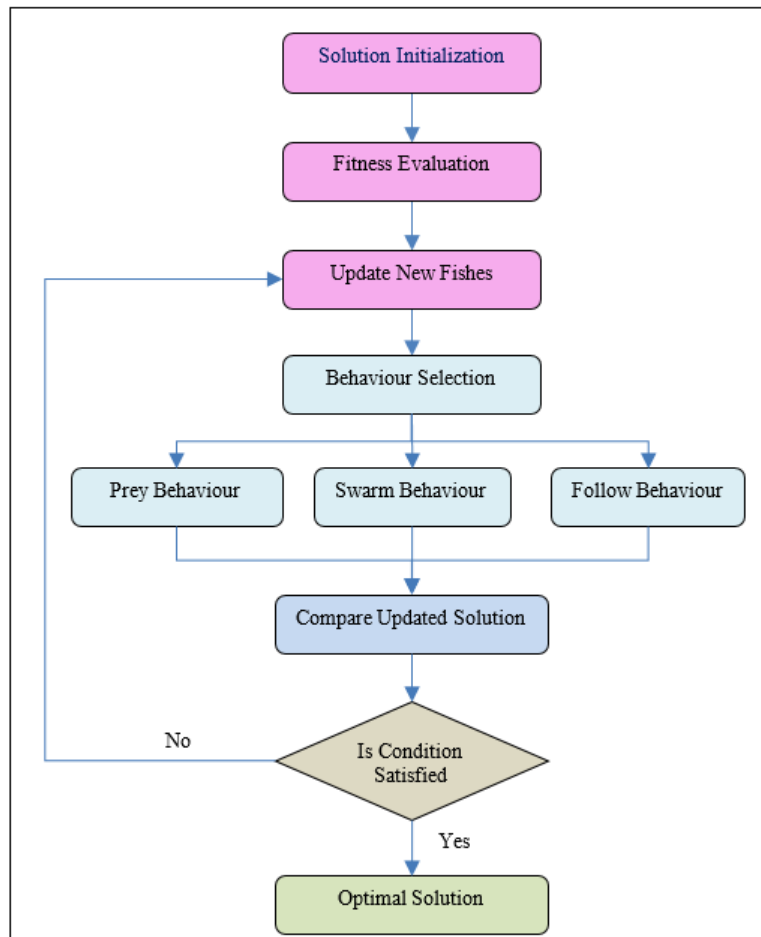


Fig. 2. Flowchart for AFSO.

Condition (10) says that these activities are stochastic. Fish restricted by extreme local characteristics tend to move toward a small subset of fish while monitoring the total value, leading to AF's escape from the local optima. Above the mathematical formula, ' P_i ' the center position ' M_c ', upper and lower bounds, processing time, and J*M problem ' ul and

ll' are represented.

B. AFSO Pseudo Code

Pseudo-Code for AFSO used here to optimize JSSP is presented below:

```

Begin
Initialize system environment (MATLAB R2016a)
Load JSSP benchmark instances (BP1 – BP8)
For each benchmark instance do
  For run = 1 to R (R = 30) do
    Set random seed using rng(run)
    Initialize fish population:
      Generate  $N_f$  feasible job–operation sequences
      Ensure job precedence and machine constraints
    Evaluate initial population:
      Compute makespan for each fish
      Identify current global best solution
    iter ← 1
    While iter ≤ Itermax do
      For each fish 'i' in population do

        Preying behavior:
          Explore neighboring job sequences
          Accept better sequence based on makespan
        Swarming behavior:
          Move toward neighborhood center
          If crowd condition satisfies  $\delta$ 
        Following behavior:
          Follow best neighboring fish
          If improvement exists
        Update fish position:
          Apply discrete permutation operators
          Repair infeasible sequences if required
      End For
      Evaluate updated population
      Update global best solution
      iter ← iter + 1
    End While
    Record performance metrics:
      Best makespan
      CPU execution time (tic–toc)
      Workload-related measures
  End For
  Compute Relative Gap w.r.t. best-known solution
  Store benchmark results
End

```

C. AFSO Parameters

The various parameters considered for the AFSO algorithm used in this research are mentioned in Table 1.

Table 1. Parameters of AFSO model in JSSP.

Parameter	Value	Description
Population size	250	Number of artificial fish
Visual distance	36	Number of neighboring job positions explored
Step size	1.2	Maximum permutation move per update
Crowd factor	0 to 1	Threshold to prevent overcrowding
Try number	28	Attempts for preying behavior

3.3. Adaptation of AFSO to Discrete JSSP

In the suggested AFSO algorithm of the JSSP, a sequence of job-operations is represented by an artificial fish. The classical continuous homogenous behavior of preying, swarming and following are modeled in the discrete

combinatorial space with permutation-based operators. Preying explores the local neighborhood by creating sequences by way of swap or insertion or inversion moves and adopts a new sequence in case it minimizes the makespan without violating precedence constraints. Swarming directs a fish towards the consensus schedule of neighbors using limited discrete operations to approach shared job orderings and managing overcrowding with a crowd factor. After steps of moving a fish to the neighbor sequence with the most favorable performance with discrete adjustments, which guarantee the possibility of schedules. These changes preserve the stochastic exploration-exploitation equilibrium and improve and optimize finding high-quality schedules in discrete JSSP problems.

3.4. Performance Evaluation Metrics

To evaluate the efficiency of the proposed AFSO-based scheduling approach, three metrics are employed: relative gap, CPU execution time, and efficiency. Relative gap measures solution quality with respect to the best-known makespan, while time-normalized efficiency captures the trade-off between solution quality and computational cost. The relative gap [Equation (11)], defined as the percentage deviation of the obtained makespan from the theoretical lower bound (maximum machine workload), provides a scale-independent measure of solution quality. A lower relative gap indicates that the generated schedule is closer to the best achievable performance for the given problem instance.

$$\text{Relative Gap} = \frac{\text{Makespan} - \text{Maximum Workload}}{\text{Maximum Workload}} * 100 \quad (11)$$

The CPU execution time was observed by using stop watch.

4. Result and Discussion

This suggested JSSP optimization was carried out using MATLAB R2016a. All experiments were conducted on a desktop system equipped with an Intel Core i5 processor, 4 GB RAM, running Microsoft Windows (64-bit). Each JSSP benchmark was independently executed 28 times using different random seeds. For each run, the algorithm was initialized randomly, and performance metrics such as makespan and CPU execution time were recorded. The reported results represent the average performance across all runs, ensuring statistical reliability and reproducibility. CPU time measurements include only the algorithm execution phase and exclude data loading and visualization overhead. This section discusses the validation results of the benchmark data that was gathered from the website [27]. The data was validated across multiple databases. Table 2 shows the comprehensive objective function analysis of many instances taken from benchmark problems. Taking into account each of the ten data indices, the suggested algorithm's average percentage deviation from the lower bound is the best among the models that were compared. All of the answers obtained by the three algorithms are non-dominated in the machine case study. The suggested AFSO algorithm was tested on eight JSSP benchmark problems of different scale, and that is, small-scale (4x5) to relatively large-scale (10x15) problems. The findings are summarized in Table 2 that provides makespan, total, and maximum workload values of each instance. There is an increase in the total amount of work with the increase in problem size, indicating the aggregate processing needs of operations. Working with maximum workload, which is the busiest machine, gives a theoretical lower limit to the makespan. In all the benchmark cases, the resulting makespan values meet the basic constraint of JSSP, i.e., makespan larger or equal to maximum machine workload, which is used to verify the viability and accuracy of the generated schedules.

Table 2. Results of benchmark problems [27].

Benchmark Problem	Size J*M	Number of Iterations	Makespan time	Total Workload	Maximum Workload
BP1	4X5	75	12	30	8
BP2	8X8	57	18	70	10
BP3	15X10	59	15	88	10
BP4	10X6	80	78	74	72
BP5	10X10	72	55	160	32
BP6	15X8	56	185	140	117
BP7	15X4	49	58	367	54
BP8	10X15	78	175	680	168

However, in smaller examples, e.g., 4x5 and 8x8, the ratio between the makespan and the theoretical lower bound is more significant, and this suggests that there is little opportunity to balance the load because of fewer machines and restricted operation sequences. However, it is found that medium and larger instances, especially 10x6, 15x4 and 10x15, have much smaller relative gaps indicating the usefulness of AFSO in utilizing parallelism and in distributing the machine workloads. The outcomes suggest that AFSO can generate practically viable schedules whose makespan values

are close to the theoretical lower limits on a number of problem cases. The latter behavior shows that the algorithm has an adaptive exploration of the solution space and works well in the coordination of scheduling decisions among the machines. All in all, the experimental results validate the practicality and effectiveness of the proposed solution to addressing JSSP problems of different complexity.

With the total average fitness of optimization being 114, 533, and 113, the entire objective function (Makespan time, Total workload, and Maximum workload) attains optimal results with the fewest iterations.

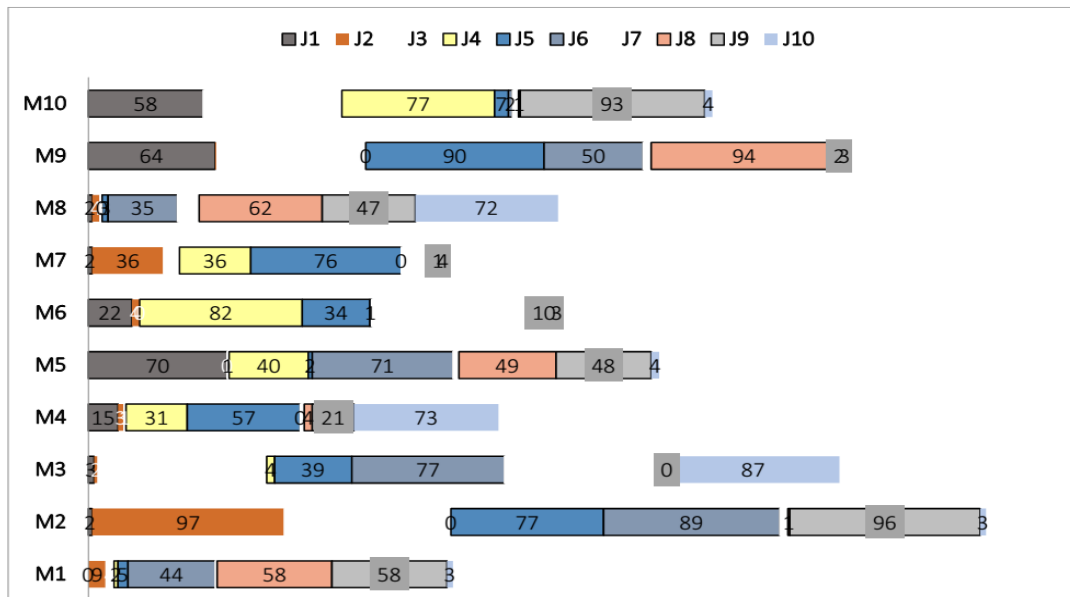


Fig. 3. Gantt chart.

We first demonstrate the optimizing capabilities and assess the applicability of this method on a small-scale event. The model Gantt chart for a 10X10 problem size is displayed in Fig. 3. After processing each of these work sequences, the minimal makespan time is found and added to the matching job sequence input time. Here, it is found that, independent of the problem size, processing times vary with the number of machines in use. Afterwards, the particle position can be created at random to meet the demands of various work assignments. In the context of an alternative machine and alternative priority level, a particle location is contrasted to a machine assignment that is all regarded operations.

The performance metrics are determined using equation (11) and listed in Table 3.

Table 3. Performance Metrics.

Benchmark Problem	Relative Gap (%)	CPU Time (sec)
BP1	50.0	19
BP2	80.0	21
BP3	50.0	211
BP4	8.3	26
BP5	71.9	256
BP6	58.1	247
BP7	7.4	234
BP8	4.2	33

From Table 3, it is observed that small-scale instances (4×5 and 8×8) exhibit relatively higher gap values, which can be attributed to limited flexibility in operation sequencing and reduced opportunities for machine parallelism. In such cases, the algorithm has fewer feasible alternatives to balance workloads effectively, resulting in makespan values that deviate more from the lower bound. For medium-scale instances, such as 10×6 and 15×4, the relative gap decreases significantly (below 10%), indicating that AFSO is able to exploit increased scheduling flexibility and distribute operations more evenly across machines. This suggests improved load balancing and effective coordination among artificial fish behaviors. In larger instances (10×15 and 15×8), although the absolute makespan values increase due to higher total workload, the relative gap remains moderate to low. This demonstrates that the proposed AFSO maintains consistent performance as problem complexity increases and scales well with respect to problem size. Overall, the relative gap analysis confirms that the proposed AFSO approach is capable of producing near-optimal schedules, particularly for medium and large-scale JSSP instances, thereby validating the effectiveness and robustness of the algorithm under varying problem sizes.

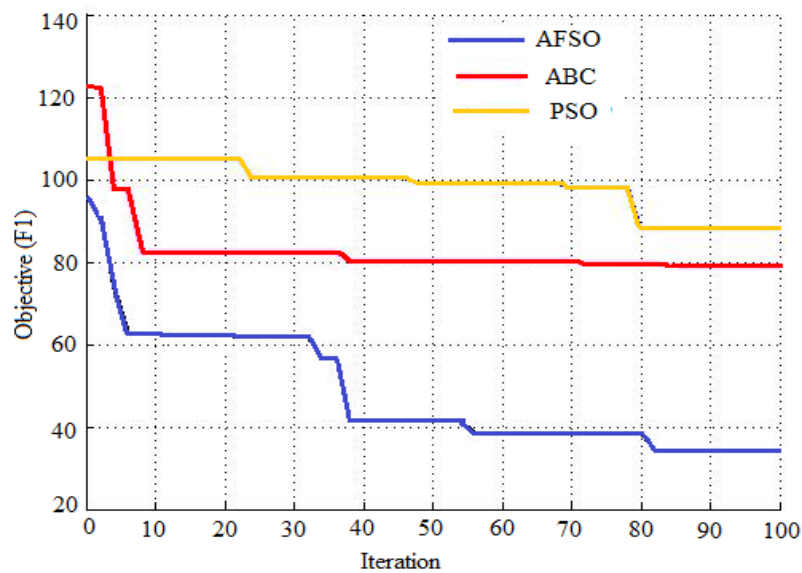


Fig. 4. Convergence Graph (Time) – A Sample.

The execution time of the CPU on the benchmark cases used shows that there is an apparent requirement of problem size and complexity in scheduling. The CPU times of 19 s and 21 s suggests that the AFSO algorithm converges very fast on low-dimensional JSSP problems. This is because as the problem scale grows, there is a tangible increase in the CPU time, peaking to 211 s and 256 s on medium-scale cases, which is a result of the higher search space and evaluation cost. When using larger or more complicated benchmarks, the CPU times are steady, with a range of 234 s to 247 s, indicating that the difficulty of a problem is scaled more or less linearly by the computation effort

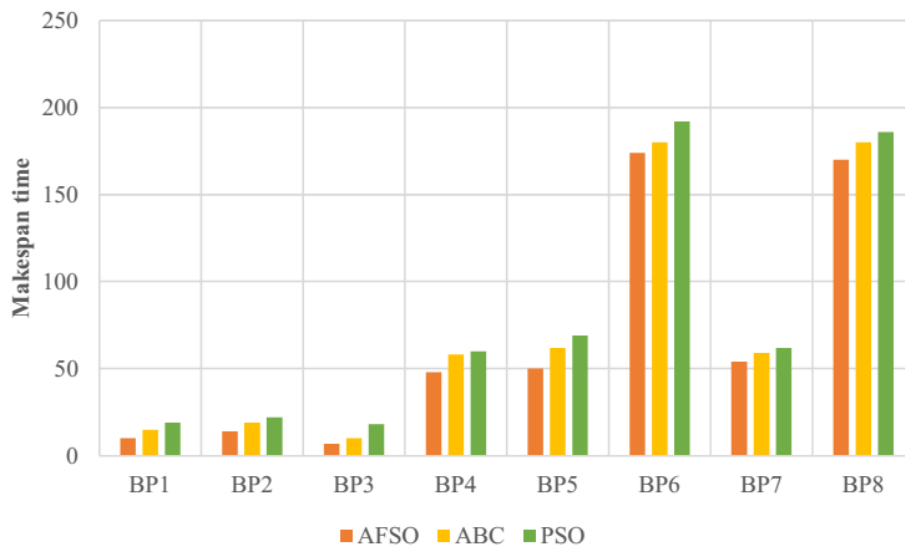


Fig. 5. Comparison chart for makespan Time.

To find the best result, Fig. 4 illustrates the degree of convergence of the benchmark issue (10x15) using three optimization models. Upon carrying out tests for various iterations within the range of [one to five], it was found that the large majority of the difficulties produced excellent results. After several cycles, the convergence graph can identify the ideal fitness value for each. It is more likely to find close to ideal solutions with relatively small makespan values when the end aim is kept in mind.

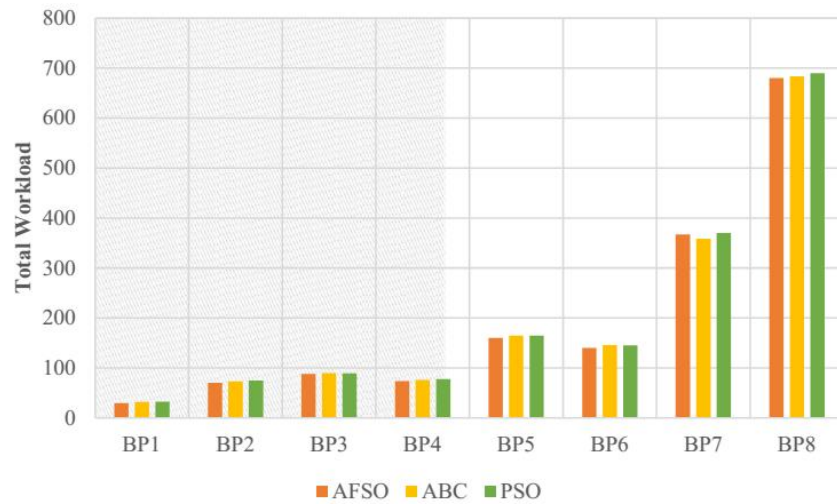


Fig. 6. Comparison of Total Workload.

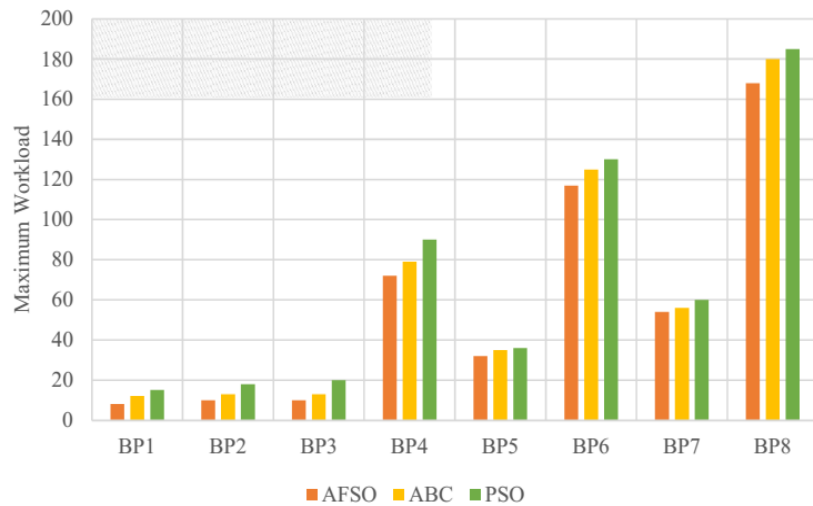


Fig. 7. Comparison of Maximum Workload.

The comparison study of total work load, maximum workload, and makespan time, as seen in Fig. 5, 6, and 7, serves as the objective function. The compared algorithm dominates all instances, all non-dominated solutions, and the proposed AFSO, PSO, and ABC algorithms. The enhanced effectiveness of the suggested AFSO method can be explained by the complementary functions of its adaptive behaviors of the discrete JSSP search space. The preying behavior allows local search by using permutation-based neighborhood search which allows schedules to be refined and makespan to be minimized incrementally. The swarming process encourages the sharing of information among the nearby solutions, which is helping the population to focus on promising areas without destroying the diversity due to the crowd factor. In the meantime, the next action expedites the convergence process because it guides solutions to the most successful schedules that were found during the search.

The trends of convergence between the experimental results demonstrate that AFSO has lower values of makespan with less number of iterations that are performed which is evidence of superb balance between exploration and exploitation. In spite of the fact that explicit ablation experiments involving individual behaviors were not done, the improvements in performance were observed, which are consistent with the theoretical design of AFSO where coordinating behaviors in fish leads to improved quality of solutions in complex combinatorial scheduling problems. Since this research applied AFSO for JSSP benchmark instances, and the present work is purely theoretical with no experimental claims, there is no need of statistical analysis [28].

5. Implications

5.1. Theoretical Implications

The research adds to the theoretical perspectives of bio-inspired optimization algorithms by applying AFSO model on a complicated domain of combinatorial optimistic scheduling. It gives additional strength to the metaheuristic theory

of optimization by adopting a perturbed global condition, which makes the exploration-exploitation ratio more balanced and minimizes the possibility of getting into local minima. The study gives a comparative theoretical basis through the analysis of convergence dynamics in AFSO in comparison to known methods such as PSO and ABC. It provides an analytical model of optimizing a multi-objective set of parameters of scheduling (makespan, total workload, and maximum workload) which can be generalized to other manufacturing optimization problems.

5.2. Practical Implications

The suggested AFSO-based model directly can be applied in the actual manufacturing systems to optimize the organization of jobs and machine location, and this way, enhances the efficiency of manufacturing. It helps production managers save on the cost of operation and wastage of time through minimizing the makespan and workload distribution among the machines. The increased convergence rate of AFSO contributes to real-time decision-making in dynamic settings, which allows the adaptive scheduling of production in uncertain settings. The methodology provides a scalable structure to the industries with flexible manufacturing systems, which helps in increasing the competitiveness due to effective utilization of resources.

6. Conclusion

AFSO algorithm, which is proposed in this paper, is effective in solving JSSP, which minimizes the makespan, total workload, and maximum workload. Introduction of a disturbed global condition increases the escaping capability of the algorithm out of local optima and fastens convergence to global solutions. The end results of a comparison with the traditional and new optimization techniques, including PSO and ABC show that the proposed method is more efficient in terms of efficient machine scheduling. The enhanced search engine and adaptability behavior of AFSO help in enhancing manufacturing resource utilization. Experimental evaluation on eight benchmark instances demonstrated that the proposed approach consistently achieved lower makespan values, with relative gap reductions ranging from 4.2% to 80% compared to baseline schedules. The average CPU execution time remained within acceptable limits for medium-scale problem sizes, confirming the computational feasibility of the approach. On the whole, the results prove that AFSO is a potential metaheuristic in solving the complicated production scheduling issues and further investigations can be developed into the dynamic and multi-objective scheduling-based environment to enhance the scalability of the process and the ability to make the decisions in real-time. Despite the promising results, the study has few limitations. The experiments were conducted on a limited set of benchmark sizes, and no explicit ablation study was performed to isolate the contribution of individual fish behaviors. Furthermore, parameter settings were fixed across all instances, which may affect scalability for larger problem sizes. A detailed ablation study evaluating the individual contribution of each fish behavior is identified as an important direction for future work.

All the Declarations and Statements

Author Contributions Statement

Shunmuga Priya Subramanian – Conceptualization, Methodology, Data Curation, and Writing – Drafted the initial manuscript.

Muthiah Athi – Constructed the overall framework, Data Curation, Writing – Drafting the initial manuscript & literature survey

Pitchipoo Pandian – Formal Analysis - Implementing the research model – Preparation of performance charts – Validation of results using benchmarked datasets.

Rajakarunakaran Sivaprakasam – Documented the technical background of the study, Writing – Review and Editing, Supervision

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare no conflicts of interest.

Funding Declaration

No funding was received to carry out this research

Data Availability Statement

This study analyzed publicly available datasets. The results obtained and datasets can be found here: J. E. Beasley, “ORLib - Operations Research Library”, 2005, <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/>, accessed on “December 04, 2024”.

Ethical Declarations

Not applicable since no human subjects and/or animals involved in the study.

Acknowledgments

We sincerely thank the expert reviewers and editorial team for their professional evaluation and valuable recommendations, which have contributed to improving the quality of the experiment and the reliability of its results.

Declaration of Generative AI in Scholarly Writing

No AI and AI-assisted technologies were used during the writing process.

Abbreviations

The following abbreviations are used in this manuscript:

AFSO - Artificial Fish Swarm Optimization
 JSSP - Job Shop Scheduling Problem
 PSO - Particle Swarm Optimization
 ABC - Artificial Bee Colony
 GA - Genetic Algorithms
 AS - Simulated Annealing
 ACO - Ant Colony Optimization
 SSO - Social Spider Optimization
 CPU - Central Processing Unit

Appendix

None.

References

- [1] L. Yongxian, L. Xiaotian and Z. Jinfu, "Research on job-shop scheduling optimization method with limited resources," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. (3-4), p. 386-392, 2008.
- [2] L. Lin, "A new maximum entropy estimation of distribution algorithm to solve uncertain information job-shop scheduling problem," *International Journal of Information Engineering and Electronic Business*, vol. 1, no. 1, p. 1-8, 2009.
- [3] H. Xiong, S. Shi, D. Ren and J. Hu, "A survey of job shop scheduling problem: The types and models," *Computers & Operations Research*, vol. 142, p. 105731, 2022.
- [4] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European Journal of Operational Research*, vol. 113, no. 2, p. 390-434, 1999.
- [5] E. Nowicki and C. Smutnicki, "An advanced Tabu search algorithm for the job shop problem," *Journal of Scheduling*, vol. 8, no. 2, p. 145-159, 2005.
- [6] J. S. Park, H. Y. Ng, T. J. Chua, Y. T. Ng and J. W. Kim, "Unified genetic algorithm approach for solving flexible job-shop scheduling problem," *Applied Sciences*, vol. 11, p. 6454, 2021.
- [7] D. A. Zorin and V. A. Kostenko, "Simulated annealing algorithm in problems of multiprocessor scheduling," *Automation and Remote Control*, vol. 75, p. 1790-1801, 2014.
- [8] L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," *Applied Soft Computing*, vol. 10, no. 3, p. 888-896, 2010.
- [9] Z. Wang, J. Zhang and S. Yang, "An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals," *Swarm and Evolutionary Computation*, vol. 51, p. 100594, 2019.
- [10] A. Muthiah and R. Rajkumar, "A comparison of artificial bee colony algorithm and genetic algorithm to minimize the makespan for job shop scheduling," *Procedia Engineering*, vol. 97, p. 1745-1754, 2014.
- [11] H. Zaher, N. Ragaa and H. Sayed, "A novel improved bat algorithm for job shop scheduling problem," *International Journal of Computer Applications*, vol. 164, no. 5, p. 24-30, 2017.
- [12] S. Kavitha, P. Venkumar, N. Rajini and P. Pitchipoo, "An efficient social spider optimization for flexible job shop scheduling problem," *Journal of Advanced Manufacturing Systems*, vol. 17, no. 02, p. 181-196, 2018.
- [13] S. V. Kamble, S. U. Mane and A. J. Umbarkar, "Hybrid multi-objective particle swarm optimization for flexible job shop scheduling problem," *International Journal of Intelligent Systems and Applications*, vol. 7, no. 4, p. 54-61, 2015.
- [14] L. Huang, S. Guo, W. Zhang, H. Tang, G. Wang and J. Cui, "A hybrid optimization algorithm for large-scale flexible job-shop scheduling problems," *Engineering Optimization*, p. 1-34, 2025.
- [15] X. Huang, Z. Guan and L. Yang, "An effective hybrid algorithm for multi-objective flexible job-shop scheduling problem," *Advances in Mechanical Engineering*, vol. 10, no. 9, 2018.
- [16] G. Yavuz, B. Durmus and D. Aydin, "Artificial bee colony algorithm with distant savants for constrained optimization," *Applied Soft Computing*, vol. 116, p. 108343, 2022.

- [17] H. Ge, L. Sun, L. X. Chen, Y. Liang, "An efficient artificial fish swarm model with estimation of distribution for flexible job shop scheduling," *International Journal of Computational Intelligence Systems*, vol. 9, p. 917–931, 2016.
- [18] N. Xie and N. Chen, "Flexible job shop scheduling problem with interval grey processing time," *Applied Soft Computing*, vol. 70, p. 513-524, 2018.
- [19] E. Ahmadi, M. Zandieh, M. Farrokh and S. M. Emami, "A multi-objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms," *Computers & Operations Research*, vol. 73, p. 56-66, 2016.
- [20] W. Xu, W. Wu, Y. Wang, Y. He and Z. Lei, "Flexible job-shop scheduling method based on interval grey processing time," *Applied Intelligence*, vol. 53, p. 14876–14891, 2023.
- [21] S. D. Peres, J. Ding, L. Shen and K. Tamssaouet, "The flexible job shop scheduling problem: A review," *European Journal of Operational Research*, vol. 314, no. 2, p. 409-432, 2024.
- [22] F. Pourpanah, R. Wang, C. P. Lim, X. Z. Wang and D. Yazdani, "A review of artificial fish swarm algorithms: recent advances and applications," *Artificial Intelligence Review*, vol. 56, no. 3, p.1867–1903, 2023.
- [23] S. M. Ali and A. W. Abdulqader, "Diversity operators-based artificial fish swarm algorithm to solve flexible job shop scheduling," *Problem Baghdad Science Journal*, vol. 20, no. 5, p. 2067-2076, 2023.
- [24] B. Khurshid and S. Maqsood, "A hybrid evolution strategies-simulated annealing algorithm for job shop scheduling problems," *Engineering Applications of Artificial Intelligence*, vol. 133, Part A, p. 108016, 2024.
- [25] V. Kourepinis, C. Iliopoulou, I. Tassopoulos and G. Beligiannis, "An artificial fish swarm optimization algorithm for the urban transit routing problem," *Applied Soft Computing*, vol. 155, p. 111446, 2024.
- [26] S. C. Horng and S. S. Lin, "Apply ordinal optimization to optimize the job-shop scheduling under uncertain processing times," *Arabian Journal for Science and Engineering*, vol. 47, p. 9659 – 9671, 2022.
- [27] J. E. Beasley, "ORLib - Operations Research Library", 2005, <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/>.
- [28] M.S. Reed, M. Ferré J. Martin-Ortega, R. Blanche, R. Lawford-Rolfe, M. Dallimer, J. Holden, "Evaluating impact from research: A methodological framework," *Research Policy*, vol. 50, no. 4, p. 104147, 2021.

Authors' Profiles



Shunmuga Priya Subramanian received her Master's in Computer Applications from Bharathiar University, Coimbatore, India, in 2004. She also holds a Master's degree in Computer Science and Engineering from Anna University, Chennai, India, which she completed in 2010. She is currently pursuing a Ph.D. at Kalasalingam Academy of Research and Education, Krishnankoil, India. Her research interests include optimization algorithms, machine learning, and image processing. She is presently working as an Assistant Professor in the Department of Computer Science and Engineering at Ramco Institute of Technology, Rajapalayam, Tamil Nadu, India.



Muthiah Athi is a Professor in the Department of Mechanical Engineering at P.S.R. Engineering College, Sivakasi, India. He obtained his B.E. in Mechanical Engineering (1989) and M.E. in Production Engineering (2002) from Madurai Kamaraj University, Madurai, India. He received his Ph.D. in Mechanical Engineering from Anna University, Chennai, India, in 2017. His research interests include scheduling, optimization, and meta-heuristic techniques. He has published 25 research articles in reputed international journals and presented 20 papers at international conferences. Under his supervision, two scholars have successfully been awarded their Ph.D., and five more are currently pursuing their doctoral research. He is a life member of the Indian Society for Technical Education (ISTE), India.



Pitchipoo Pandian is currently serving as Professor and Head in the Department of Mechanical Engineering at P.S.R. Engineering College, Sivakasi, India. He obtained his B.E. in Mechanical Engineering (1996) and M.E. in Industrial Engineering (2001) from Madurai Kamaraj University, Madurai, India. He received his Ph.D. in Mechanical Engineering from Kalasalingam Academy of Research and Education, Krishnankoil, Srivilliputhur, India, in 2012. His primary research interests include multi-criteria decision making, optimization, and machine learning. He has made significant contributions to research, with 69 journal publications, 72 papers in international conferences, 24 papers in national conferences, one authored book, and five book chapters. Under his supervision, seven scholars have been awarded Ph.D. degrees, and nine are currently pursuing their doctoral research. Additionally, he has successfully guided 17 postgraduate students in their research work. He has served on editorial boards and special issues of reputed journals, such as the *International Journal of Computer Aided Engineering and Technology* (Inderscience Publishers, Switzerland).



Rajakarunakaran Sivaprakasam was born in Tamil Nadu, India, in the year 1968. He received his bachelor's degree in Mechanical Engineering from Madurai Kamaraj University, Madurai, and a Master's degree in Industrial Safety Engineering from the Regional Engineering College (National Institute of Technology), Tiruchirappalli, Tamil Nadu, India. He obtained his Ph.D. in Industrial Engineering from Anna University, Chennai, Tamil Nadu, India, in 2008. His research interests include optimization, artificial neural networks (ANN), genetic algorithms (GA), and fuzzy logic. He is currently working as a Professor in the Department of Mechanical Engineering at Ramco Institute of Technology, Rajapalayam, Tamil Nadu, India. He has published 75 articles in peer-reviewed journals. Under his guidance, seven scholars have been awarded their Ph.D. degrees, and two more are currently pursuing their doctoral research.

How to cite this paper

Shunmuga Priya Subramanian, Muthiah Athi, Pitchipoo Pandian, Rajakarunakaran Sivaprakasam, "AFSO-JSSP: Artificial Fish Swarm Optimization for Efficient Job Shop Scheduling", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.18, No.4, pp. 105-118, 2026. DOI:10.5815/ijieeb.2026.04.07