AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 58(2) (2026) 277-292
DOI: 10.22060/miscj.2026.25711.5488

Fishing Boats Group: A novel swarm intelligence algorithm for global optimization

Ehsan Ehsaeyan*

Department of Electrical Engineering, Sirjan University of Technology, Sirjan, Iran.

ABSTRACT: This study presents the Fishing Boat Group (FBG), a new metaheuristic optimization method based on how fishing boats at sea work together when they hunt for fish. This paper's main contribution is the development of an algorithm that, through tuning only one parameter, achieves superior computational efficiency for high-dimensional problems while consistently demonstrating robust performance and competitive fitness values across multimodal benchmarks. FBG categorizes boats into three groups (A, B, and C) to search regions near the best boat (G_best), the virtual median boat, and unexplored areas. To check how well it works, FBG was tested against six well-known and newer metaheuristic algorithms using ten multimodal benchmark functions and two real-world engineering design problems. The experimental results show that FBG did better than the other methods in 8 of the 10 tests, suggesting it stays pretty robust and performs consistently across different cases. In practical applications, the algorithm achieved superior mean fitness values for the Three-bar truss (263.896) and Tension/compression spring (0.012989313) designs. Most notably, FBG maintained the shortest execution times across all benchmark functions, with an overall average of 0.68 seconds. These results suggest that FBG could be a strong option for tackling high-dimensional optimization problems, especially in situations where you need both accurate results and fast computation.

Review History:

Received: Apr. 21, 2026
Revised: Aug. 23, 2026
Accepted: Aug. 23, 2026
Available Online: Sep. 15, 2026

Keywords:

Fishing Boats Group
Human-Inspired Algorithm
Global Optimization
Benchmark Function

1- Introduction

Optimization algorithms are tools that improve performance and speed across various industries. Metaheuristic techniques are generally classified into four categories based on their underlying principles: evolutionary, swarm-based, physics-based, and human-based algorithms. Evolutionary algorithms (EAs) mimic processes like natural selection, genetics, crossover, mutation, and survival of the fittest to solve optimization problems. Key examples include the Genetic Algorithm (GA) [1], Evolution Strategies (ES) [2], the Biogeography-based Optimizer [3], and Differential Evolution [4]. Swarm-based algorithms are inspired by the collective behavior of natural systems, such as bird flocks, fish schools, ant colonies, and bee hives. By mimicking the social interaction and information-sharing strategies of these biological groups, these algorithms effectively solve complex optimization problems across continuous, discrete, multimodal, and constrained search spaces. Notable examples include the Artificial Bee Colony [5], Gray Wolf Optimizer [6], Grasshopper Optimization Algorithm [7], Dolphin Echolocation [8], and the Enhanced Whale Optimization Algorithm [9]. Physics-based algorithms draw inspiration from the fundamental laws of nature, utilizing principles of physical phenomena to explore search spaces and achieve convergence. Notable examples include Plasma Generation Optimization [10], Prism Refraction Search [11], Special Relativity Search [12], and Optics-Inspired Optimization [13], all of which model physical interactions to solve complex optimization challenges. Human-based metaheuristic algorithms are optimization tools inspired by how humans learn, think, and solve problems together. Instead of looking to nature, these methods use human-like strategies such as teaching, sharing knowledge, and making group decisions to find the best solutions to complex problems. Human-based algorithms include Harmony Search [14], Brain Storm Optimization [15], Teaching–Learning-Based Optimization [16], and Sine Cosine Algorithm [17]. Table 1 depicts examples of different algorithms of metaheuristic categories with advantages and disadvantages.

This study introduces a new metaheuristic algorithm named Fishing Boats Group (FBG), based on how fishing boats work together out at sea. The primary innovation of this study is the development of a unique three-group search mechanism (Groups A, B, and C). Unlike traditional metaheuristics that often rely on a single or dual-phase search, FBG utilizes three distinct mathematical operators: Group A follows the global best solution (*Gbest* to focus on known hotspots, Group B optimizes local exploration

*Corresponding author's email: ehsaeyan@sirjantech.ac.ir

Copyrights for this article are retained by the author(s) with publishing rights granted to Amirkabir University Press. The content of this article is subject to the terms and conditions of the Creative Commons Attribution 4.0 International (CC-BY-NC 4.0) License. For more information, please visit https://www.creativecommons.org/licenses/by-nc/4.0/legalcode.

UNCORRECTED PROOF

Table 1. Classification of famous metaheuristic methods.

category	algorithm	Advantage	Disadvantage
Evolutionary-based	GA [1]	Proven ability to execute on a global scale	Untapped regional potential
	ES [2]	High operational concurrency and rapid responsiveness	Limited optimization in data discovery
	BBO [3]	Maximum impact, minimum effort	Challenges arise in high-dimensional domains.
	DE [4]	Generating a diverse array of solutions	Suboptimal exploitation capacity
Swarm-based	ABC [5]	Superior exploration capability	Limited exploitation performance
	GWO [6]	Efficient and conceptually simple	Struggles in high-dimensional spaces
	GOA [7]	Diverse solution generation	Weak convergence precision
	DE [8]	Excellent global search ability	Prone to premature convergence
	EWOA [9]	Superior convergence speed	Increased computational complexity
physics-based	PGO [10]	High precision in stable plasma density control	Complex multi-parameter coupling and high computational cost
	PRS [11]	Highly effective at maintaining solution diversity through multi-directional scattering	Significant sensitivity to initial parameter settings and control variables
	SRS [12]	Superior balance between exploration and exploitation via time-dilation dynamics	Susceptible to local optima entrapment due to high parameter sensitivity
	OIO [13]	Superior ability to map complex search spaces through light propagation modeling	High susceptibility to parameter-tuning overhead in large-scale applications
Human-based	HS [14]	Exceptional ease of implementation and structural simplicity	Susceptibility to slow convergence and potential stagnation
	BSO [15]	Excellent global exploration through multi-cluster divergence and collaborative ideation	High sensitivity to clustering parameters and risk of premature convergence within local clusters
	TLBO [16]	Parameter-free mechanism with robust convergence speed and simplicity	Risk of falling into local optima when dealing with complex, highly multimodal landscapes
	SCA [17]	Simple design with highly efficient switching between exploration and exploitation	High risk of poor convergence accuracy and getting trapped in local optima

UNCORRECTED PROOF

by targeting positions near a virtual median boat, and Group C explores surrounding waters based on the mean and median of the fleet's positions. This mechanism allows FBG to strike an effective balance between exploration and exploitation, enabling it to navigate complex, multimodal search spaces without becoming trapped in local optima—a distinct advantage over similar swarm-based methods that lack this multi-directional, median-anchored coordination. The contributions of this paper can be summarized as follows:

1- Development of the FBG Algorithm: A new nature-inspired metaheuristic based on cooperative fishing boat behaviors is proposed.
2- Three-Group Innovation: A novel three-group mechanism (employing mean and median rules) is introduced to enhance navigation across complex search spaces and prevent premature convergence.
3- Computational Efficiency: The algorithm's design provides superior execution speeds and maintains low memory complexity $O(N \times D)$ making it highly suitable for high-dimensional problems.
4- Robust Performance: Extensive evaluation against both established and newer algorithms across ten multimodal benchmarks demonstrates FBG's competitive robustness, consistently achieving superior fitness values.
5- Practical Applicability: The effectiveness of the algorithm is validated through successful applications in solving complex real-world engineering design problems.

The rest of this paper is organized as follows. In Section 2, the proposed algorithm is described together with its mathematical formulation. Time complexity analysis of the FBG Algorithm is presented in this section. The effectiveness of the algorithm is evaluated in Section 3, where experiments involving multiple standard benchmarks and practical engineering design problems are presented. Concluding remarks along with potential future research paths are outlined in Section 4.

2- FBG optimization algorithm

2- 1- Inspiration

Fishing boats move across the sea, some small and old, others large and modern. Each one leaves with hope, guided by the ocean, the fish, and the crew's instincts. The sea can be calm or harsh, but the boats keep going, keeping people connected to the water. Some stay near proven spots, trusting fish will return. Others slip into empty spaces, searching patiently for hidden schools. A few explore far beyond familiar areas, testing new waters. Together, these boats balance each other, cover more sea, find more fish, and show that mixing old habits with new ideas keeps progress alive. Figure 1 shows fishing boats preparing for sea.

2- 2- Mathematical model

Similar to other metaheuristic algorithms, initialization is the first step of the FBG algorithm. The initial population of candidate solutions (representing boat positions) is generated randomly within the defined search space boundaries. This is mathematically expressed as:



Fig. 1. Fishing boats at sea.

UNCORRECTED PROOF

$$\vec{x}_i = \overrightarrow{lb} + U(0,1) * (\overrightarrow{ub} - \overrightarrow{lb}) \quad (1)$$

Where $\vec{x}_i$ is the i-th candidate solution vector, representing the position of the i-th boat in the search space. $\overrightarrow{lb}$ and $\overrightarrow{ub}$ are the vectors defining the lower bound and upper bound of the decision variables, respectively. $\vec{U}(0,1)$ is a d-dimensional vector of uniformly distributed random numbers in the interval $[0,1]$. This initialization formula ensures a uniform and diverse spread of the initial population across the entire feasible region, which is crucial for effective exploration in the early stages of the optimization process. For a population of size N, this step is repeated N times. This creates all the initial candidate solutions: $\vec{x}_1,...,\vec{x}_N$. They are collected into a position matrix. X in d dimensional space as follows:

$$X = \begin{bmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix} \quad (2)$$

After placing the fishing boats, they check how well each boat is doing. Fitness just means how good a spot is—basically, how many fish you can find there. A higher fitness value means the fishing spot is better and more likely to yield a good catch. After checking all the boats, they are arranged from the highest to the lowest fitness scores. Like in Particle Swarm Optimization (PSO), the boat with the best fitness is picked as the global best (*Gbest*). Figure 2 shows a boat called GBEST, which has caught the highest number of fish. This *Gbest* acts as a reference point for the swarm's best-known solution so far and helps guide how other boats move and explore in later steps, pushing the group to focus on better areas of the solution space. There are now three groups of boats: A, B, and C. The population is divided randomly into three identical groups: A, B, and C. Each group changes its position based on these rules:

$$x_a^t = Gbest + \exp\left(1 - \frac{r_1 t}{Iteration}\right)\sin(2\pi r_2) \times (x_a^{t-1} - Gbest) \quad (a) \quad (3)$$

$$x_b^t = median(X) + \left(x_p^{t-1} - x_q^{t-1}\right) \quad (b)$$

$$x_c^t = r_3 x_c^{t-1} * mean(X)/median(X) \quad (c)$$

Let x_a^t, x_b^t, and x_c^t denote the updated positions of boats in iteration t belonging to Groups A, B, and C, respectively. The variables r_1, r_2, and r_3 represent random numbers uniformly generated within the interval $[0,1]$. Similarly, p and q are random integers selected from the range? $[1,N]$, where N is the total number of boats. The terms *mean* and *median* correspond to the mean and median values of Matrix. X, which stores the positions of all boats. p and q may be selected from any of the three categories: Group A, Group B, or Group C boats. Group B boats are specifically designed to optimize local exploration by targeting positions near the virtual median boat. The virtual median boat determines its coordinates by calculating the median. X positions of the entire fleet. This mechanism significantly enhances the algorithm's sensitivity during the final stages of the search process. By anchoring search behavior to the median, the algorithm mitigates the tendency to oscillate around the global optimum when it is in proximity, effectively narrowing the search radius to converge on the target point. In Eq. (3-3), the scalar factor $\text{mean}(X)/\text{median}(X)$ is calculated as $mean(X) \times median(X)/median(X)^2$. The variable *iteration* specifies the maximum number of iterations allowed, effectively setting the termination condition for the algorithm.

After updating the positions, some boats might end up outside the designated fishing area, and that's still seen as an acceptable solution. To fix this problem, Equation (4) is used to adjust how the boats move and remap their coordinates, making sure every boat stays inside the fishing area's boundaries. This change improves how well the fleet works and helps make the most of the chances to catch fish in the chosen areas. Making this change will help us keep things running smoothly while following fishing rules.

$$x^t = \begin{cases} ub & x^{t-1} > ub \\ x^{t-1} & lb < x^{t-1} < ub \\ lb & x^{t-1} < lb \end{cases} \quad (4)$$

The whole process of the suggested algorithm is shown in the flowchart in Figure 3. This visual shows the steps involved and how different parts of the algorithm interact with each other. Algorithm 1 shows the pseudocode for the FBG.

The description of the FBG process in the flowchart and pseudocode is as follows: First, a random population is generated to create the X matrix. Then, the fishing boats are divided into three groups, with each boat updating its position according to its group rule (Eq. 3). Outlier boats are repositioned back into the fishing area, and this process repeats until the stopping criterion is satisfied. The most successful boat is identified as the solution (*Gbest*). Group A boats keep aiming for the spot they call the global best. Since fish often swim in groups, they stay near this area, expecting more catches, just like fishermen who return to a proven hotspot. Group B boats are specifically designed to optimize local exploration by focusing their search efforts around the position of a virtual median boat. This virtual boat acts as a dynamic reference point and determines its coordinates by calculating the median positions of all boats in the fleet. Group C boats adjust their positions to explore

UNCORRECTED PROOF



Fig. 2. Boat GBEST with the largest catch.

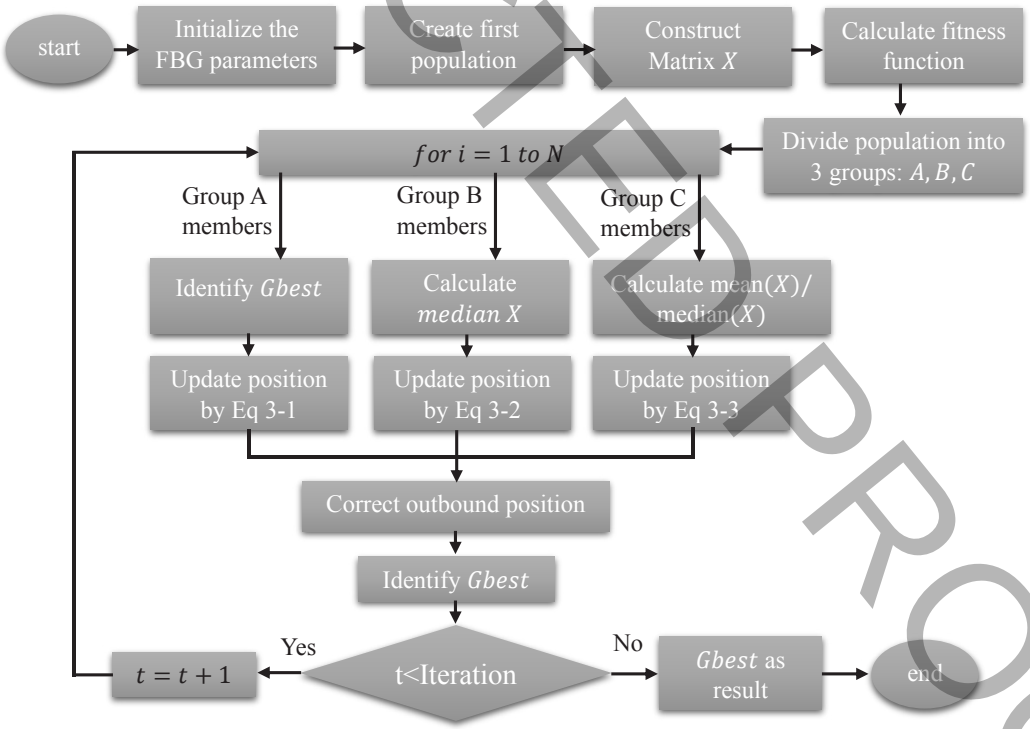


Fig.3. Flowchart of FBG algorithm.

UNCORRECTED PROOF

Algorithm 1. Pseudocode of the proposed algorithm.

input: FBG parameters (number of iterations $Iteration$, dimension D, feasible area $[lb, ub]$, objective function $f(x)$, population size N)

1. Initialize boat positions
2. Construct matrix X
3. calculate fitness functions and select $Gbest$
4. Divide the population into 3 groups: group A, group B, and group C
5. **while** $t < Iteration$
6. **for** i=1 to N
7. **if** i belongs to group A
8. identify $Gbest$ and update position using Eq. 3-1
9. **end if**
10. **if** i belong to group B
11. calculate $median(X)$ and update position using Eq. 3-2
12. **end if**
13. **if** i belong to group C
14. calculate $mean(X)/median(X)$and update position using Eq. 3-3
15. **end if**
16. correct unfeasible solutions by Eq. 4
17. identify $Gbest$
18. **end for**
19. **end while**

Output: the best solution ($Gbest$)

nearby waters. By spreading out and using different search strategies, they avoid crowding and work as a coordinated fleet to efficiently locate the best fishing grounds.

2- 3- Complexity Analysis of the FBG Algorithm

The initialization of the N candidate solutions in a D-dimensional search space requires $O(ND)$ operations. During each of the T iterations, the objective function is evaluated for all N candidate solutions, resulting in $O(NC)$ computational cost per iteration. The position-update operations for Groups A, B, and C require $O(ND)$ operations. For Groups B and C, the median of each decision-variable dimension is computed using Quickselect. Since Quickselect has an expected complexity of $O(N)$ for finding the median, computing the median over all D dimensions requires $O(ND)$ expected time. The mean computation also requires $O(ND)$. Boundary checking requires an additional $O(ND)$ operations per iteration. Therefore, the expected overall time complexity of the FBG algorithm is $O(T(NC + ND))$.

The total space complexity of the FBG algorithm is $O(N \times D)$, primarily dominated by the position matrix X and the storage required for the G_{best} and boundary vectors. Regarding the calculation of the median for Group B, while a standard sorting approach may require $O(N)$ auxiliary space per dimension, the implementation utilizes the Quickselect algorithm. This approach allows for the median to be computed with minimal additional memory overhead, ensuring that the overall space complexity remains low. $O(N \times D)$ and allowing the algorithm to remain lightweight and scalable for high-dimensional problems.

3- Results and discussions

3- 1- benchmark functions

The first experiment is done using test functions. Table 2 gives a clear rundown of the specs for these benchmark functions. It covers important details like the size of each function, the range of values they take, and any special features they might have.

Three traditional algorithms from existing research were picked for comparison: Particle Swarm Optimization (PSO) [18], Gravitational Search Algorithm (GSA) [19], and Whale Optimization Algorithm (WOA) [20]. Three newer algorithms are also included in the evaluation: Dhole Optimization Algorithm (DOA) [21], Cultural History Optimization Algorithm (CHOA) [22], and Divine Religions Algorithm (DRA) [23]. Using a mix of tried-and-true and newer methods will provide a clear picture of how the proposed algorithm works, making it easier to see where it does well and where it struggles with complex problems. Table 3 shows the parameters used for the algorithms. For all the problems, the algorithm uses a population size of 30 and limits the process to 600 iterations. Each algorithm is tested through 40 separate trials to get a solid sense of how well it performs. This method helps ensure their reliability and how well they

UNCORRECTED PROOF

Table 2. Benchmark function descriptions.

No	Name	Formula	f_{min}
1	Csendes	$f_1(\vec{x}) = \sum_{d=1}^{D} x_d^6 \left(2 + \sin\frac{1}{x_d}\right)$	0
2	Pinter	$f_2(\vec{x}) = \sum_{d=1}^{D} \left[dx_d^2 + 20d\sin^2(x_d) + d\sin^2\left(\frac{(d+1)x_{d+1}^2}{\pi}\right)\right], x_{D+1} = x_1$	0
3	DROP-WAV	$f_3(\vec{x}) = -\frac{1+\cos\left(12\sqrt{\sum_{d=1}^{D} x_d^2}\right)}{2+0.5\sum_{d=1}^{D} x_d^2}$	-1
4	Sinusoidal Quadratic	$f_4(\vec{x}) = \sum_{d=1}^{D} \left(\sin^2(x_d) + 0.1x_d^2\right)$	0
5	Salomon	$f_5(\vec{x}) = 1 - \cos\left(2\pi\sqrt{\sum_{d=1}^{D} x_d^2}\right) + 0.1\sqrt{\sum_{d=1}^{D} x_d^2}$	0
6	Styblinski-tank	$f_6(\vec{x}) = \frac{1}{2}\sum_{d=1}^{D} \left(x_d^4 - 16x_d^2 + 5x_d\right)$	-39.166D
7	Deb	$f_7(\vec{x}) = -\frac{1}{D}\sum_{d=1}^{D} \sin^6(5\pi x_i)$	-1
8	Pathological	$f_8(\vec{x}) = \sum_{d=1}^{D-1} \left(0.5 + \frac{\sin^2\sqrt{100x_d^2 + x_{d+1}^2} - 0.5}{1+0.001(x_d^2 - 2x_d x_{d+1} + x_{d+1}^2)^2}\right)$	0
9	Sinusoidal	$f_9(\vec{x}) = -2.5\prod_{d=1}^{D} \sin\left(x_d - \frac{\pi}{6}\right) - \prod_{d=1}^{D} \sin\left(5x_d - \frac{5\pi}{6}\right)$	-3.5
10	Vincent	$f_{10}(\vec{x}) = -\frac{1}{D}\sum_{d=1}^{D} \sin(10\ln(x_d))$	-1

work are tested in different situations. All the experiments were done using MATLAB 2019 on a Windows 11 system. The hardware used for this research had 8 GB of RAM and an Intel Core i7 processor, giving enough computing power to handle the tasks needed.

Table 4 sums up a comparison between FBG and other algorithms, showing how they perform on test functions with dimensions of 40. This study looks at how well the FBG algorithm can scale by putting its search ability to the test on tough high-dimensional optimization problems. The best result in each row is shown in bold. This table shows that FBG has better average results in 8 out of 10 benchmark cases, which means it's generally more consistent and delivers better solutions than the other methods. The results show that FBG is the only algorithm that can achieve the best scores for "best," "worst," and "standard deviation" metrics. It also tends to get better average results most of the time, no matter how many dimensions the problem has.

A systematic comparison of convergence performance was conducted between FBG and other leading algorithms over a set of ten benchmark functions (F1–F10). Figure 4 shows convergence plots that compare how each algorithm improves over time, giving a clear view of how their solutions and search patterns differ. When tested on all benchmark functions, FBG usually shows solid convergence performance. Its search behavior shows a quick move toward the best areas, steady progress with little back-and-forth, and clear stability over time.

UNCORRECTED PROOF

Table 3. Parameter configurations of optimization algorithms for assessment and comparison.

Algorithm	Parameter
PSO	$c_1 = c_2 = 2$ *Inertia weight = 0.9 to 0.4*
GSA	$G_0 = 100, \alpha = 40$ $Rnorm = 2, Rpower = 1$
WOA	$a = [0,2], b = 1$
DOA	$c_1 = 1, c_3 = 3$ $K = 0.5, \mu = 25$
CHOA	$N_{clus} = 10, P_{comp} = P_{comb} = 0.5$
DRA	$BPSP = 0.5, RP = 0.2$
FBG	$group_{a,b,c} = {}^{N}/_{3}$

Running time serves as a practical measure for evaluating the time complexity of the FBG algorithm. Figure 5 shows the execution times of the compared algorithms over all test functions, based on 40 independent runs per function. The FBG algorithm features a simple structure and can be implemented straightforwardly, as it requires no intermediate parameters. This practical advantage is confirmed in Figure 5, where FBG attains the best ranking in terms of running time (average of 0.68285 seconds overall the benchmarks), followed by DOA, CHOA, PSO, DRA, GSA and WOA. In order to provide a more detailed analysis, the execution times for the various components of the proposed algorithm are presented in Table 5. FBG demonstrates a faster execution time than the other algorithms because its spatial update process is performed simultaneously. This allows the mathematical operations in Equation 3 to be executed as matrix operations, eliminating the need for iterative loops in the program.

The Wilcoxon rank-sum test evaluates whether two groups come from the same distribution by ranking all observations from both groups and comparing the rank sums between them [24]. The test yields a p-value; if this p-value is less than a predefined significance level (α), the null hypothesis of equal distributions is rejected. In the present analysis, a 5% significance level was adopted for the Wilcoxon rank-sum test. The statistical findings are summarized in Table 6, with bold formatting applied to values above 0.05. Only 5 values surpass this threshold. This constitutes statistical evidence that the distributions of the paired algorithm result samples differ significantly.

3- 2- Three-bar truss design

The three-bar truss problem is a structural optimization task aimed at determining the ideal cross-sectional areas for each member. This optimization task aims to minimize the total volume of the truss structure under static load, while strictly enforcing stress constraints. It is formulated as

$$\text{Minimizing } f(x_1, x_2) = l \times \left(2\sqrt{2}x_1 + x_2\right) \tag{5}$$

Subject to

$$g_1(x_1, x_2) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0$$

$$g_2(x_1, x_2) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0$$

$$g_3(x_1, x_2) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leq 0$$

$$l = 100cm, \sigma = \ P = \frac{2kN}{cm^3}$$

$$0 \leq x_1, x_2 \leq 1$$

Where x_1 is the area of diagonal bars, x_2 is the area of the horizontal bar, l is the length, P is the *applied load*, σ is allowable stress, $f(x)$ is the total weight, and g_1, g_2, g_3 are stress constraints. In this experiment, each optimization

UNCORRECTED PROOF

Table 4. Comparing the proposed algorithm's performance on benchmark functions with recent techniques (Dim = 40).

No		CHOA	DOA	DRA	GSA	PSO	WOA	FBG
1	Best	0.000838932	0.001702181	8.7424E-05	2.84338E-11	0.000126844	0.452938071	**1.53956E-11**
	Mean	0.00973834	0.022752224	0.000158221	1.5298E-07	0.000189126	0.590997626	**3.99081E-10**
	Worst	0.023920684	0.089796571	0.000202436	6.84197E-07	0.000241809	0.688745054	**1.50611E-09**
	STD	0.009289306	0.031418801	3.17275E-05	2.4218E-07	3.36197E-05	0.071601708	**4.40547E-10**
2	Best	0.026001834	0.009536069	3.999112813	2.04417E-08	3.99949884	3.751920897	**6.27141E-10**
	Mean	0.279659101	0.670618823	3.999587261	0.399452052	3.999702857	3.829998323	**5.79994E-09**
	Worst	0.745219164	2.717658429	3.999918613	3.994500708	3.999915413	3.949435992	**1.32457E-08**
	STD	0.25622763	0.849615538	0.000250734	1.263171339	0.000144264	0.064902458	**3.75682E-09**
3	Best	1.43E-129	7.7289E-138	**0**	**0**	**0**	**0**	**0**
	Mean	1.88287E-96	9.8939E-82	**0**	**0**	**0**	**0**	**0**
	Worst	1.88287E-95	9.8939E-81	**0**	**0**	**0**	**0**	**0**
	STD	5.95416E-96	3.12872E-81	**0**	**0**	**0**	**0**	**0**
4	Best	2.69978E-05	0.000111365	**0**	9.29573E-10	**0**	0.000140698	**0**
	Mean	0.010612837	0.017577865	0.000391462	1.0276E-07	**6.47313E-29**	0.669499569	1.81591E-08
	Worst	0.086738758	0.067472661	0.003914618	5.20919E-07	**6.47313E-28**	5.552697124	1.69661E-07
	STD	0.026887105	0.023602476	0.001237911	1.80969E-07	**2.04698E-28**	1.749674634	5.32939E-08
5	Best	1.10168E-50	5.65405E-39	**0**	**0**	**0**	**0**	**0**
	Mean	2.57643E-30	1.3684E-23	**0**	**0**	**0**	**0**	**0**
	Worst	2.23085E-29	1.32628E-22	**0**	**0**	**0**	**0**	**0**
	STD	7.01746E-30	4.18005E-23	**0**	**0**	**0**	**0**	**0**
6	Best	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
	Mean	-0.99977196	-0.99362448	**-1**	**-1**	**-1**	-0.94899622	**-1**
	Worst	-0.99771961	-0.93624533	**-1**	**-1**	**-1**	-0.93624509	**-1**
	STD	0.000721123	0.02016098	**0**	**0**	**0**	0.026881354	**0**
7	Best	5795.584479	5428.812867	4621.255551	1.9455E-05	5439.02325	9327.146955	**8.74982E-06**
	Mean	7360.200175	6982.352116	6398.068172	**0.290247811**	6705.927502	9518.232068	263.2391608
	Worst	8339.298074	8501.063351	8036.358622	**1.771838283**	7425.404494	9663.627213	1447.239893
	STD	760.4255992	1023.16547	1097.249548	**0.54483797**	688.4186911	92.15960351	532.6331349
8	Best	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**
	Mean	7.99361E-16	1.86517E-15	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**
	Worst	3.9968E-15	7.54952E-15	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**	**4.44089E-16**
	STD	1.12347E-15	2.48408E-15	**0**	**0**	**0**	**0**	**0**
9	Best	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	**0**	1.11022E-17	**0**	**0**	**0**	**0**	**0**
	Worst	**0**	1.11022E-16	**0**	**0**	**0**	**0**	**0**
	STD	**0**	3.51083E-17	**0**	**0**	**0**	**0**	**0**
10	Best	0.124125216	0.090885693	1.088819943	8.39836E-09	1.178412053	10.93123592	**3.95676E-10**
	Mean	0.313430323	0.385605147	3.187705472	1.12666E-06	3.251330077	13.95047754	**2.78319E-08**
	Worst	0.731526392	0.768856103	4.633656014	7.98325E-06	5.632957052	17.66765777	**1.99888E-07**
	STD	0.178453344	0.200201266	1.030095629	2.45708E-06	1.278893948	1.932712276	**6.1092E-08**

UNCORRECTED PROOF

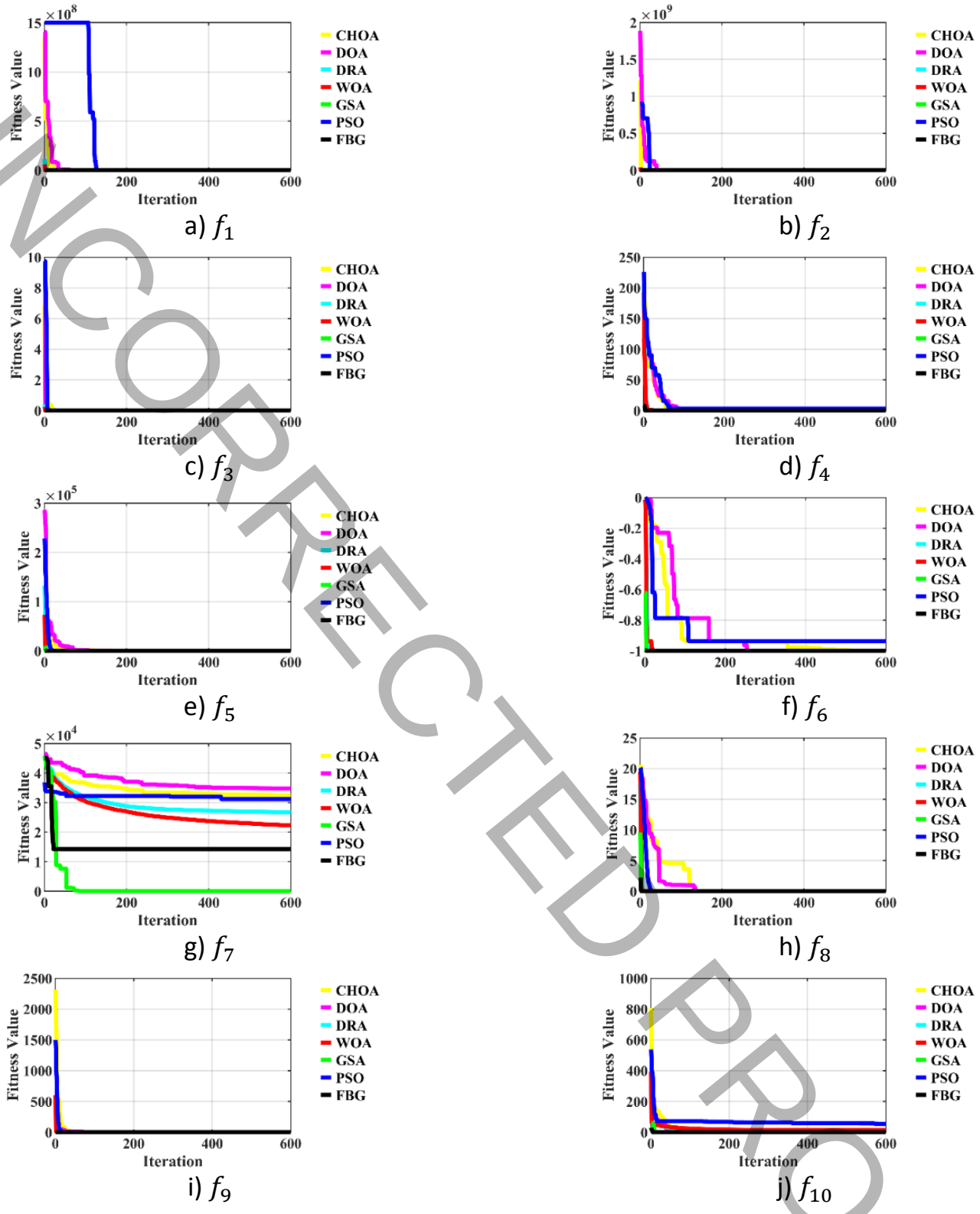


Fig.4. A comparison of convergence curves for the proposed algorithm and other algorithms evaluated on the benchmark function.

UNCORRECTED PROOF

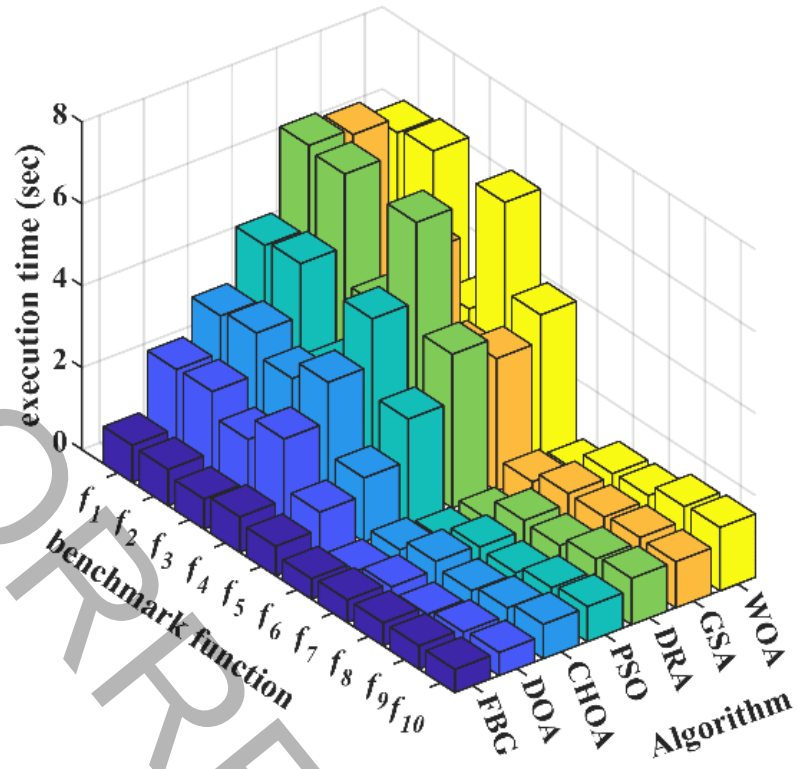


Fig. 5. A comparison of the running times for the FBG algorithm and other studied algorithms on the test functions.

Table 5. Comparison of execution times for FBG algorithm components in milliseconds (one iteration).

FBG component	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Initialization	0.01997	0.02071	0.02137	0.02205	0.02140	0.01993	0.02056	0.02112	0.02125	0.02136
Generate random parameters	0.05750	0.05762	0.05889	0.06010	0.06026	0.05699	0.05760	0.05859	0.05910	0.05958
Update boats' positions by Eq. (3)	0.75503	0.73039	0.57728	0.70690	0.55928	0.34564	0.39891	0.40621	0.41302	0.40649
Correct outbound locations by Eq. (4).	0.03109	0.03239	0.03206	0.03290	0.03330	0.03084	0.03139	0.03165	0.03258	0.03282
calculation Gbest	0.03980	0.04119	0.04160	0.04136	0.04216	0.03947	0.04025	0.04093	0.04096	0.04145

UNCORRECTED PROOF

Table 6. Wilcoxon's rank test results at the 5% significance level.

NUM	CHOA	DOA	DRA	GSA	PSO	WOA
1	6.28E-03	5.33E-07	1.59E-03	1.03E-02	1.54E-03	1.34E-02
2	2.44E-03	4.39E-04	1.02E-04	1.72E-07	1.69E-02	**6.80E-02**
3	3.25E-03	4.12E-04	3.04E-04	3.74E-05	1.27E-02	1.87E-02
4	8.80E-07	8.70E-03	2.00E-03	**6.19E-02**	2.70E-05	1.07E-03
5	2.33E-02	1.16E-08	1.45E-05	4.61E-02	9.00E-03	1.08E-08
6	2.91E-02	8.08E-06	4.70E-02	8.61E-07	2.65E-02	3.70E-06
7	1.11E-07	5.86E-07	2.07E-06	3.56E-07	1.94E-05	2.53E-10
8	2.14E-03	4.97E-07	2.07E-19	**5.93E-02**	2.80E-03	2.13E-03
9	2.62E-02	4.81E-02	4.18E-03	1.70E-02	2.80E-03	1.35E-03
10	1.16E-03	1.98E-04	3.95E-05	**5.08E-02**	3.40E-06	**5.56E-02**

Table 7. Results of different algorithms on three-bar truss design.

algorithm	CHOA	DOA	DRA	GSA	PSO	WOA	FBG
Best fitness	263.895843	263.895843	263.895843	263.895843	263.895843	263.895843	263.895843
Mean	265.499104	264.128408	264.7072823	266.590352	263.914660	265.400286	**263.896194**
Success rate	82.5	90	87.5	82.5	92.5	87.5	**97.5**
Best x1	0.788664	0.788670	0.788671	0.788665	0.788672	0.788656	0.788680
Best x2	0.408281	0.408262	0.408261	0.408276	0.408256	0.408302	0.408234
time (sec)	0.235181	0.215147	0.310509	0.316897	0.281651	0.379225	**0.180725**

algorithm was run 40 times independently to ensure statistical reliability. The results of these trials, shown in terms of best fitness, best $[x_1, x_2]$, mean values, and mean running times, are summarized in Table 7. Since this problem is a low-dimensional problem (two dimensions), algorithms easily find the global optimum. The success rate is calculated as follows to better distinguish between the abilities of different methods:

$$success\ rate = \frac{successful\ outcomes}{total\ runs\ (40)} \times 100 \quad (6)$$

The best result in each row is highlighted in bold. These results show that the FBG algorithm consistently finds the best solution (mean fitness=263.896194) across 40 independent runs. FBG records the fastest execution time, showing it operates more efficiently than the other methods tested. FBG shows strong performance in this problem domain because it achieves both solution quality and computational speed at the same time.

3- 3- Tension/compression spring design

The design of tension and compression springs is a fundamental discipline in mechanical engineering, centered on balancing force, deflection, and stress within spatial and

UNCORRECTED PROOF

Table 8. Results of different algorithms on tension/compression spring design.

algorithm	CHOA	DOA	DRA	GSA	PSO	WOA	FBG
Best fitness	0.013353118	0.013687458	0.013605104	0.013359773	0.014474539	0.01329911	**0.012989313**
Mean	0.015819338	0.016094085	0.014745832	0.014835987	0.015289121	0.014884925	**0.014673633**
Best x1	0.054875052	0.052254222	0.053509914	0.053663933	0.058365085	0.051570622	0.053895642
Best x2	0.437500827	0.369111561	0.3961862	0.399445587	0.507979949	0.353569821	0.408246131
Best x3	8.135704322	11.58070735	9.993170342	9.613856868	6.364729892	12.14303679	8.953598404
g_1	-0.04664236	-0.08815004	-0.055919	-0.02921673	-0.00155198	-0.05708447	-0.00580905
g_2	-0.00154946	-0.00296238	-0.01211586	-0.01349295	-0.05004944	-0.00069301	-0.00782454
g_3	-3.94929995	-3.65149702	-3.79130721	-3.91350424	-3.99116294	-3.77140658	-4.07263515
g_4	-0.67174941	-0.71908948	-0.70020259	-0.69792699	-0.62243664	-0.72990637	-0.69190548
time (sec)	0.07134666	0.06199715	0.095536554	0.103808412	0.086833587	0.122767446	**0.053933347**

material constraints. The model of tension/compression spring design can be expressed as

$$x = [x_1, x_2, x_3]$$

Minimize:

$$f(x) = (x_3 + 2)x_2x_1^2 \quad (7)$$

Subject to

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

$$0.05 \leq x_1 \leq 2$$

$$0.25 \leq x_2 \leq 1.3$$

$$2 \leq x_3 \leq 15$$

The performance of the FBG algorithm is evaluated by designing a tension/compression spring. To ensure consistency and comparability among the optimization algorithms, this experiment uses a uniform setting where the population size is fixed at 30, and the number of iterations is set to 600. Each algorithm is executed independently for a total of 40 trials. This approach provides a more equitable assessment of their performance across different runs, reducing variability and improving the reliability of the results. Similar to the previous experiment, Table 8 presents the best fitness, mean fitness, best vector x, and mean execution time values. The minimum values are shown in bold.

UNCORRECTED PROOF

The FBG algorithm demonstrates rapid convergence (best fitness: 0.012989313) and effective global search capability (mean fitness: 0.014673633). Its top ranking in mean fitness confirms its consistent superiority.

4- Conclusion

This paper introduces a new metaheuristic called the Fishing Boats Group (FBG), inspired by the cooperative behavior of fishing vessels. FBG has a simple structure that enables fast convergence, with carefully designed exploration and exploitation phases. Its performance was evaluated using ten multimodal benchmark functions and compared with three classical and three recent metaheuristics. Results show FBG is competitive across the tested functions. The algorithm also performed well on two engineering design problems, matching specialized algorithms from the literature. However, the FBG algorithm divides the population into three equal groups using a random selection process, which lacks a strategic mechanism for balancing exploration and exploitation. To address this limitation, future research will focus on transitioning from static, random partitioning to a more intelligent, dynamic grouping strategy. Specifically, a distance-based clustering approach (such as K-means or fuzzy c-means) is intended to be implemented to categorize agents based on their spatial distribution in the search space.

References

[1] Katoch, S., Chauhan, S. S. & Kumar, V. A review on genetic algorithm: past, present, and future. Multimed. Tools Appl. 80, 8091–8126 (2021).

[2] Beyer, H. G. & Schwefel, H. P. Evolution strategies–a comprehensive introduction. Nat. Comput. 1, 3–52 (2002).

[3] Simon, D. Biogeography-based optimization. IEEE Trans. Evol. Comput. 12, 702–713 (2008).

[4] Neri, F. & Tirronen, V. Recent advances in differential evolution: A survey and experimental analysis. Artif. Intell. Rev. 33, 61–106 (2010).

[5] Karaboga, D. & Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. 39, 459–471 (2007).

[6] Mirjalili, S., Mirjalili, S. M. & Lewis, A. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61 (2014).

[7] Saremi, S., Mirjalili, S. & Lewis, A. Grasshopper optimisation algorithm: Theory and application. Adv. Eng. Softw. 105, 30–47 (2017).

[8] Kaveh, A. & Farhoudi, N. A new optimization method: Dolphin echolocation. Adv. Eng. Softw. 59, 53–70 (2013).

[9] Nadimi-Shahraki, M. H., Zamani, H. & Mirjalili, S. Enhanced whale optimization algorithm for medical feature selection: a COVID-19 case study. Comput. Biol. Med. 148, 105858 (2022).

[10] Kaveh, A., Akbari, H. & Hosseini, S. M. Plasma generation optimization: a new physically-based metaheuristic algorithm for solving constrained optimization problems. Eng. Comput. 37, 4103–4128 (2021).

[11] Mirjalili, S. & Lewis, A. Prism refraction search: a novel physics-based metaheuristic algorithm. J. Supercomput. 80, 7546–7576 (2024).

[12] Goodarzimehr, V., Shojaee, S., Hamzehei-Javaran, S., et al. Special relativity search: A novel metaheuristic method based on special relativity physics. Knowl.-Based Syst. 257, 109484 (2022).

[13] Husseinzadeh Kashan, A. A new metaheuristic for optimization: Optics-inspired optimization (OIO). Comput. Oper. Res. 55, 99–125 (2015).

[14] Geem, Z. W., Kim, J. H. & Loganathan, G. V. A new heuristic optimization algorithm: harmony search. Simulation 76, 60–68 (2001).

[15] Shi, Y. Brain storm optimization algorithm. In International Conference on Swarm Intelligence, 303–309 (Springer, 2011).

[16] Rao, R. V., Savsani, V. J. & Vakharia, D. P. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput.-Aided Des. 43, 303–315 (2011).

[17] Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. Knowl. Based Syst. 96, 120–133 (2016).

[18] Shami, T. M. et al. Particle Swarm Optimization: A comprehensive survey. IEEE Access 10, 10031–10061 (2022).

[19] Rashedi, E., Nezamabadi-pour, H. & Saryazdi, S. GSA: A Gravitational Search Algorithm. Inf. Sci. 179, 2232–2248 (2009).

[20] Mirjalili, S. & Lewis, A. The Whale Optimization Algorithm. Adv. Eng. Softw. 95, 51–67 (2016).

[21] Mohammed, B. O., Aghdasi, H. S. & Salehpour, P. Dhole optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Cluster Comput. 28, 430 (2025).

[22] Sharifi, T., Mirsalim, M., Gharehchopogh, F. S. & Mirjalili, S. Cultural history optimization algorithm: a new human-inspired metaheuristic algorithm for engineering optimization problems. Neural Comput. Appl. 37, 21009–21068 (2025).

[23] Mozhdehi, A. T. et al. Divine Religions Algorithm: a novel social-inspired metaheuristic algorithm for engineering and continuous optimization problems. Cluster Comput. 28, 253 (2025).

[24] Derrac, J., García, S., Molina, D., & Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1(1), 3–18 (2011)

UNCORRECTED PROOF

UNCORRECTED PROOF

HOW TO CITE THIS ARTICLE
E. Ehsaeyan, Fishing Boats Group: A novel swarm intelligence algorithm for global optimization, AUT J. Model. Simul., 58(2) (2026) 251-266.

DOI: 10.22060/misci.2026.25711.5488

UNCORRECTED PROOF