Octopus Inspired Optimization Algorithm: Multi-Level Structures and Parallel Computing Strategies

Abstracts

This paper introduces a novel bionic intelligent optimisation algorithm, Octopus Inspired Optimization (OIO) algorithm, which is inspired by the neural structure of octopus, especially its hierarchical and decentralised interaction properties. By simulating the sensory, decision-making, and executive abilities of octopuses, the OIO algorithm adopts a multi-level hierarchical strategy, including tentacles, suckers, individuals and groups, to achieve an effective combination of global and local search. This hierarchical design not only enhances the flexibility and efficiency of the algorithm, but also significantly improves its search efficiency and adaptability. In performance evaluations, including comparisons with existing mainstream intelligent optimisation algorithms, OIO shows faster convergence and higher accuracy, especially when dealing with multimodal functions and high-dimensional optimisation problems. This advantage is even more pronounced as the required minimum accuracy is higher, with the OIO algorithm showing an average speedup of 2.27 times that of conventional particle swarm optimisation (PSO) and 9.63 times that of differential evolution (DE) on multimodal functions. In particular, when dealing with high-dimensional optimisation problems, OIO achieves an average speed of 10.39 times that of DE, demonstrating its superior computational efficiency. In addition, the OIO algorithm also shows a reduction of about 5% in CPU usage efficiency compared to PSO, which is reflected in the efficiency of CPU resource usage also shows its efficiency. These features make the OIO algorithm show great potential in complex optimisation problems, and it is especially suitable for application scenarios that require fast, efficient and robust optimisation methods, such as robot path planning, supply chain management optimisation, and energy system management.

Keywords:

1. Introduction

Heuristic algorithms are a class of optimization algorithms that mimic biological or physical phenomena found in nature [1-4]. These algorithms seek approximate optimal solutions to complex problems by emulating mechanisms such as evolution, group behaviors, and immune system functions. Compared to traditional mathematical planning methods, heuristic algorithms can often find satisfactory solutions in a shorter time and effectively avoid the problem of falling into local optima.

Among these, Swarm Intelligence (SI) is one of the most well-known branches of heuristic algorithms [7-8]. SI simulates the collective behaviors of organisms in nature, such as ants, flocks of birds, and schools of fish, achieving group optimization through simple interactions between individuals. Classic SI algorithms include Ant Colony Optimization (ACO) [9-10], Particle Swarm Optimization (PSO) [11-12], and others. These algorithms showcase the great potential of heuristic approaches in solving complex optimization problems. However, existing bionic intelligence optimization algorithms primarily focus on the "emergence" of intelligence from cooperative behaviors between individuals, with less emphasis on the role of individual intelligence in this process. In fact, the intelligent behaviors of many organisms are shaped by intricate neural structures. For instance, ants have numerous olfactory receptors on their antennae, and the ganglia in their abdomen integrate multiple sensory inputs [13-14]. These complex perceptual and neural computational mechanisms form the basis of ants' intelligent behavior. Therefore, we believe it is essential to consider the neuromorphological characteristics of individuals more thoroughly when designing bionic intelligence optimization algorithms.

In this paper, we propose a new bionic intelligence optimization algorithm inspired by the unique behavioral features of the octopus, such as foraging and predator avoidance, and the hierarchical structure of the octopus nervous system. The algorithm is designed around key features of the octopus, including the hierarchical structure of the suckers, the decentralized interaction of the tentacles, and information exchange between individuals. Mechanisms such as multi-level search, asynchronous parallelism, adaptive parameter control, and region regeneration

are incorporated. Through reasonable abstraction and mapping, we construct a flexible and efficient Octopus Inspired Optimization (OIO). Compared to traditional swarm intelligence algorithms, this work focuses more on extracting intelligent mechanisms from neuromorphological features, aiming to enhance the adaptability and emergent abilities of the algorithm. This approach can avoid local optima while maintaining algorithmic simplicity, demonstrating excellent performance in various benchmark functions and practical application problems.

2. Related Work

In recent years, many researchers have proposed various novel optimization algorithms inspired by biological mechanisms in nature. For example, the Particle Swarm Optimization (PSO) algorithm proposed by Kennedy and Eberhart in 1995 simulates the flocking behavior of birds when foraging for food and achieves optimization through information exchange between individuals. Dorigo et al. drew on the pheromone mechanism of ants searching for food and proposed the Ant Colony Optimization (ACO) algorithm in 2006. Karaboga proposed the Artificial Bee Colony (ABC) algorithm in 2005, inspired by the honey harvesting behavior of honey bees. The ABC algorithm cleverly balances the relationship between exploration and exploitation, showing good optimality searching ability on several benchmark functions[15-16]. Mirjalili et al. proposed the Grey Wolf Optimizer (GWO) in 2014, inspired by the grey wolf's hunting mechanism. GWO demonstrated excellent performance in solving multiple benchmark functions and engineering problems by simulating the social hierarchy and hunting behavior of grey wolf packs [17-18]. Xu et al. proposed the Bat Sonar Algorithm (BSA) in 2021 [19-20]. Although these algorithms show the superiority of group intelligence, they mainly focus on modeling individual behaviors and lack sufficient exploration of the organizational structure of the group. For example, the interaction between particles in the PSO algorithm is relatively simple, making it difficult to portray the complex group decision-making process [21-22].

Although the above algorithms have achieved notable results in their respective fields, they either focus only on modeling the behavior of individual organisms and lack in-depth exploration of the higher intelligence of the group, or they qualitatively focus on individual organisms without adequately addressing the structural complexity of group relationships. In contrast, octopuses possess a unique hierarchical and autonomous nervous system, including multi-layered structures such as suckers, tentacles, and individuals, which support their flexible and varied sensing, decision-making, and execution abilities. Additionally, there is a certain degree of synergistic cooperation and optimization among groups [23-24]. Inspired by this, this paper proposes the Octopus Inspired Optimization (OIO) to achieve a breakthrough in intelligent optimization algorithms by simultaneously considering individual learning and group cooperation.

The innovations of the Octopus Inspired Optimization (OIO) include:

1.Proposal of a New Bionic Intelligence Optimization Algorithm: Inspired by the nervous system of the octopus, the OIO algorithm designs mechanisms such as multi-level search, asynchronous parallelism, and adaptive parameter control to fully utilize the octopus's perception, decision-making, and execution capabilities.

2.Algorithm Implementation Strategy: Based on various neuromorphological features of the octopus, such as suckers, tentacles, individuals, and swarms, specific implementation strategies for the OIO algorithm are tailored.

3.Performance Evaluation: The OIO algorithm is compared with existing mainstream intelligent optimization algorithms in a multi-dimensional benchmark test. The results show that the OIO algorithm excels in convergence speed, solution quality, and robustness.

4.Scalability and Application Prospects: The bionic design of the OIO algorithm can be generalized to other intelligent computing models such as neural network design and has potential applications in various fields such as pattern recognition, robot control, and intelligent optimization.

Subsequent chapters of this paper will detail the design of the OIO algorithm. Chapter 3 will explore the morphological features of the octopus nervous system and their implications for algorithm design. Chapter 4 will detail the framework and methodological principles of the OIO algorithm, focusing on the specific implementations of multi-level search, asynchronous parallelism, and adaptive control. Chapter 5 will systematically evaluate the performance of the OIO algorithm through simulation experiments and analyze it in comparison with other intelligent optimization algorithms.

3. Morphological Feature Analysis of the Octopus

Octopuses are highly evolved cephalopod mollusks with complex nervous systems and remarkable intelligence. Their nervous system comprises approximately 500 million neurons, most of which are distributed in their tentacles, enabling them to perform functions such as grasping and exploration independently [25-27]. The suckers on octopus tentacles are embedded with multimodal receptors that can integrate and transmit environmental information to higher neural centers [28]. These features allow octopuses to interact precisely with their environment, providing important insights for the design of intelligent algorithms.

Overall, the information processing flow of the octopus nervous system is as follows: the suckers selectively perceive environmental information, which is integrated and summarized in the ganglia of the tentacles. Based on proprioceptive information and inputs from other tentacles, the tentacles independently execute various behaviors such as movement, extension, growth, and retraction. The individual octopus gains global information and controls and adjusts the movement of the tentacle structure accordingly. At the group level, when information is obtained from different octopuses, it is possible to abstract an overall profile of the problem solution. Based on the needs of problem-solving, the group adjusts through master-slave co-evolution to obtain targeted information.

3.1 Octopus Basic Nervous System

The nervous system of the octopus has a unique structure, from the suckers to the ganglia and then to the tentacles, as shown in Figure 1. This paper systematically analyzes the key features at these structural levels and their impact on algorithm design.

Inspired by the morphological characteristics of the octopus, we abstract four levels: sucker, tentacle, individual, and group to simulate the emergence of intelligent behavior. After aggregating individual information, different information sources from octopuses are integrated to establish an abstract concept of the group, achieving multimodal source integration.
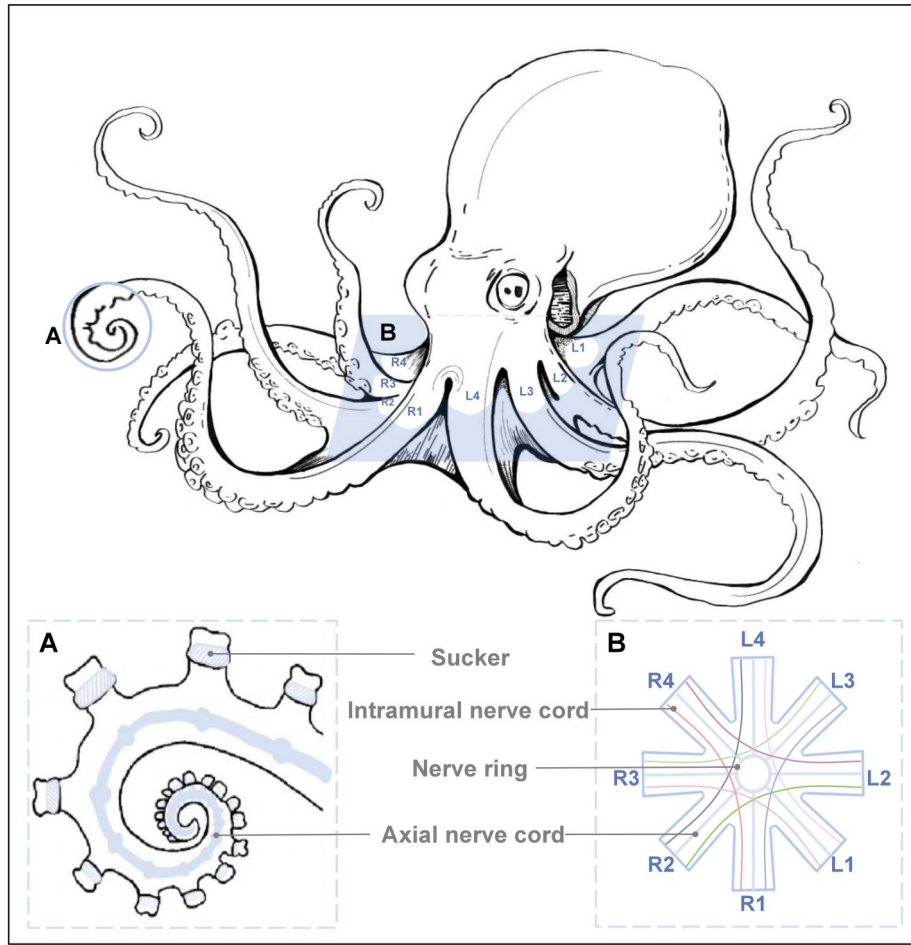
Figure 1. Diagram of the Nervous System of Octopuses

3.2 Suckers: Information Perception and Local Processing

In group optimization problems, we use multidimensional function values to evaluate the function of the sucker structure, simulating the independent control of suckers by the tentacles. This design allows suckers to perform efficient local information perception and processing, providing necessary feedback to the upper structures, thereby demonstrating the distributed processing characteristics of the octopus nervous system. The specific details of information transmission and parameter adjustment are shown in Figure 2.

In the algorithm design, the suckers are initialized at the beginning of each iteration. They first collect information based on the evaluation function, then aggregate and feedback this information to the tentacle structure. Before the next iteration, the tentacle structure adjusts the relevant parameters, updates the performance indicators of the suckers, and optimizes the

information processing process, outputting the aggregated information to the upper structures. This process can be represented by the following formula:

$$S_i^{t+1} = f\left(S_i^t, E_i^t, P_i^t\right) \tag{1}$$

where S_i^t represents the state of the i th sucker at iteration t, E_i^t represents the environmental information it acquires, P_i^t represents the control parameters returned by the tentacle structure, and f is the information processing function of the sucker.
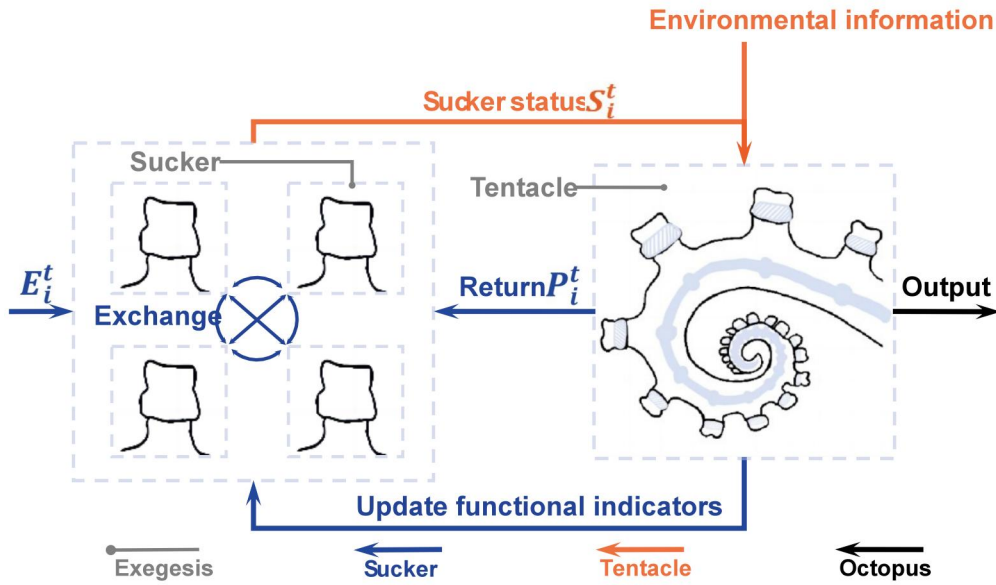

Figure 2. Diagram of the Suckers

3.3 Tentacles: Distributed Search and Autonomous Coordination

During each iteration, the tentacles first integrate information from the suckers and, after evaluating the environment, make autonomous decisions involving movement adjustments (e.g., moving, extending) and sucker function regulation. All information is ultimately fed back to the individual, influencing the motion instructions for the next iteration. This process can be represented by the following formula:

$$T_i^{t+1} = g\left(T_i^t, S_i^t, I^t\right) \tag{2}$$

where T_i^t is the state of the i th tentacle, S_i^t is the set of sucker states, I^t is the motion

instruction, and g is the decision function.

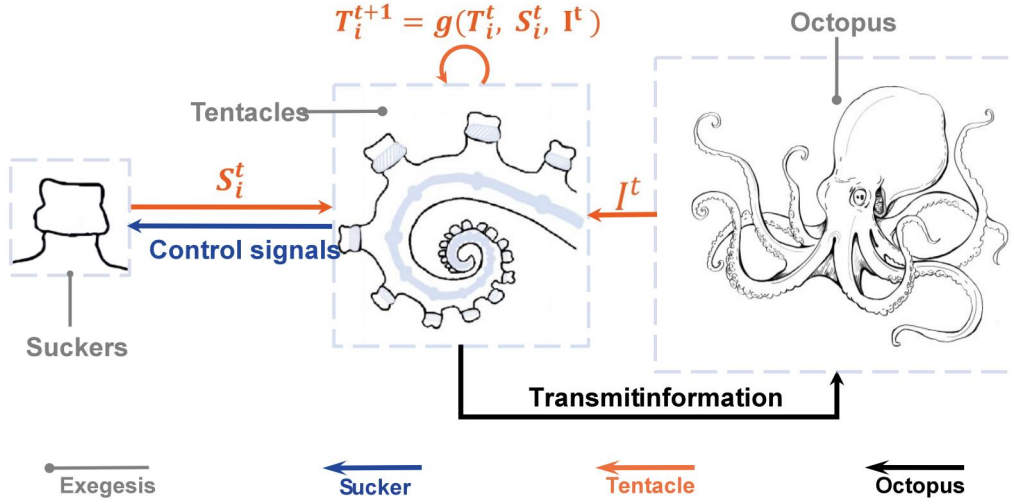$$T_i^{t+1} = g(T_i^t, S_i^t, \mathbf{I}^t)$$

Figure 3. Diagram of the Tentacles

In group optimization, the behavior of the tentacles is simulated by internal sucker particles, adjusting the search range and position of the suckers on the tentacles to simulate tentacle extension and retraction. The tentacles elongate when moving towards optimal regions and contract or degenerate otherwise, balancing the exploration and exploitation of the algorithm to optimize performance. The growth and degeneration mechanisms of the tentacles need to be designed according to the problem and parameters, selecting appropriate trigger conditions and adjustment strategies to ensure the stability and efficiency of the algorithm while considering computational overhead to avoid performance degradation.

3.4 Individuals: Macro Decision-Making and Adaptive Regulation

In the Octopus Inspired Optimization, individuals not only act as macro decision-makers and adaptive regulators by integrating tentacle information and making decisions based on environmental characteristics, but also control the group optimization algorithm through a reward and punishment mechanism. Specifically, individuals reward or punish tentacles based on their performance in the global environment, and reinitialize tentacles that remain in low-value or local optimal regions for extended periods. This control structure allows individuals to guide the group optimization algorithm to adaptively adjust search strategies, thereby enhancing overall algorithm performance. This process can be represented by the following formula:

$$I^{t+1} = h\left(I^t, T^t, M^t\right) \qquad (3)$$

Where I^t is the individual state, T^t is the set of tentacle states, M^t is the memory information, and h is the decision function. By dynamically adjusting the forgetting factor ϵ and the memory factor γ, individuals use the update function ϕ to balance the preservation of historical information with the forgetting of past optimal values, optimizing search efficiency and algorithm convergence. The process of individuals adaptively adjusting search strategies is illustrated in Figure 4.
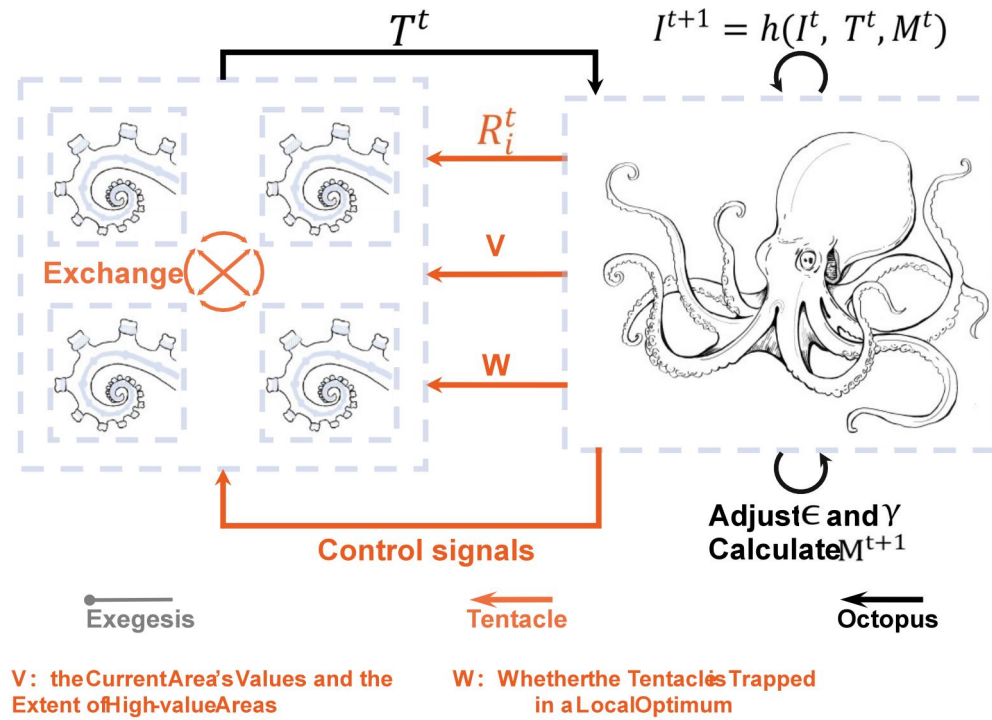

Figure 4. Diagram of the Octopus Individuals

3.5 Group Intelligence: Emergent Properties and Collaborative Optimization

The group aggregates information from octopus individuals of the same or different "types" and synthesizes multiple data sources to comprehensively describe the problem-solving process. For each octopus's data, the group abstracts the information and integrates all types of data into the same dimension. For each octopus's movement, the group regulates based on the needs of problem-solving, enabling each octopus to collaboratively address the issue.

In the group optimization algorithm, since the input data are merely the extreme value solutions of multidimensional functions, we establish a reinforcement learning model and a global optimal control mechanism for each octopus. This design allows the algorithm to gain stronger regulation capabilities (e.g., avoiding local optima) and more comprehensive search abilities for the global optimum during the search process.

4. Methodology of the Octopus Inspired Optimization

4.1 Group Co-evolution

The Octopus Intelligence Optimization (OIO) algorithm is inspired by the high cooperation capability of octopus groups, adopting a master-slave group co-evolution strategy. This strategy decomposes the overall problem into multiple sub-problems, which are solved through information sharing and cooperation among sub-groups, thereby overcoming the local optimal traps commonly encountered by traditional algorithms in high-dimensional complex problems [29-30].
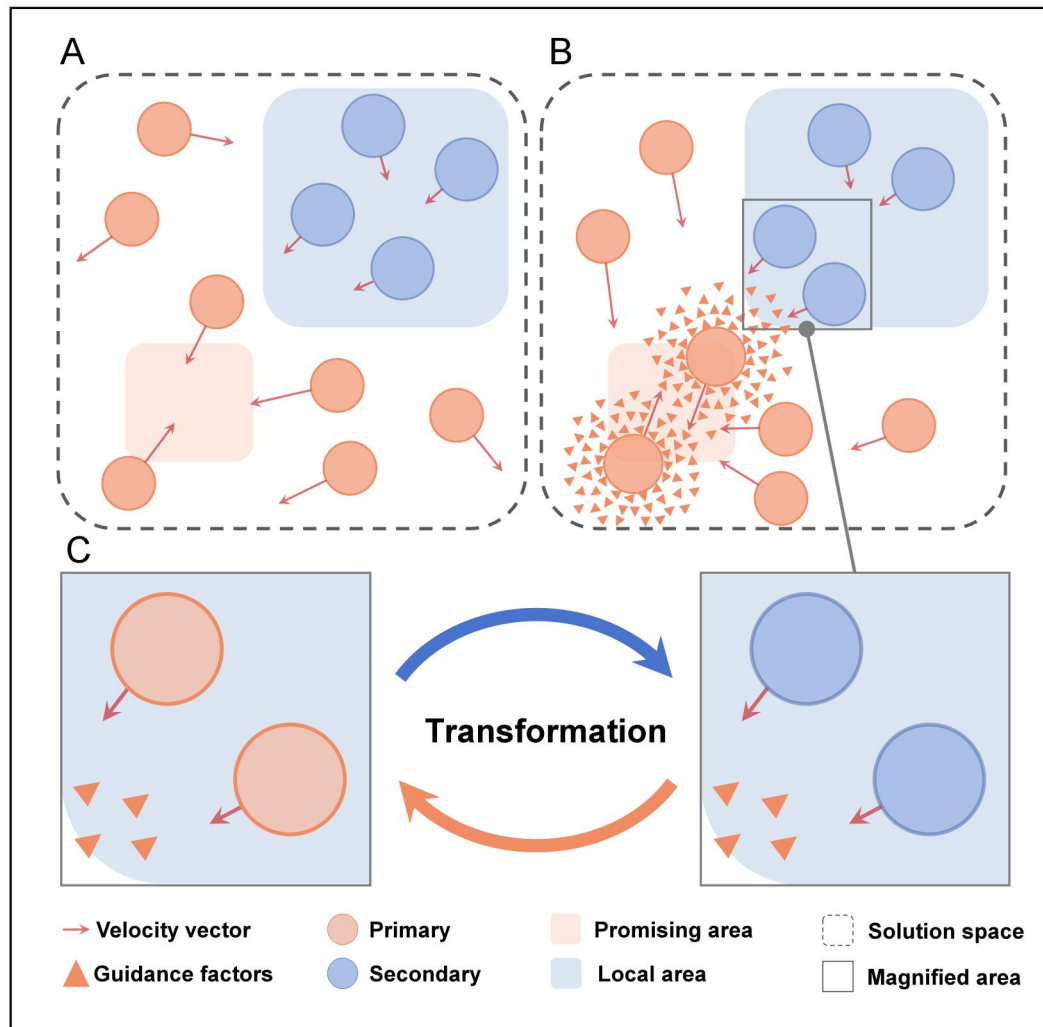

Figure 5. Diagram of Group Co-evolution

(A) Master points searching for areas of dominance;

(B) Some subject points finding areas of dominance and guiding other follower points to adjust their stride and direction;

(C) Search points close to the search area switching from follower object to subject object.

As shown in Figure 5, the main group deeply exploits the current optimal region, while the subordinate group extensively explores new areas. When the subordinate group discovers a potential area, it converts into the main group through a "guidance factor," and the main group adjusts the search direction based on the concentration of the "guidance factor." The algorithm also includes a master-slave identity switching mechanism to maintain a balance between exploration and exploitation [4]. The pseudocode for the algorithm's partial principles is shown in Pseudocode 1.

Pseudocode 1 Guidance Factor

1: Initialize n-dimensional space with random instances
2: **while** termination condition not met **do**
3:    Evaluate instances in the environment
4:    Differentiate 20% elites (main agents) and 80% followers
5:    **for** each elite **do**
6:        Adjust search parameters and step size
7:        Traverse high-value regions intensively
8:        Communicate restricted search regions to followers
9:    **end for**
10:   **for** each follower **do**
11:       Randomly explore the global space with larger step sizes
12:   **end for**
13:   **for** each follower **do**
14:       **if** follower discovers a higher value region than an elite **then**
15:           Promote follower to elite
16:           Demote the current elite to follower
17:       **end if**
18:   **end for**
19: **end while**

4.2 Memory-Based Evolution Strategy

The memory and learning abilities of octopuses in nature also provide inspiration for the design of the OIO algorithm. Our memory-based strategy uses evolutionary historical information such as individual heredity and fitness statistics to guide the evolution process, balancing global and local searches [31-32]. The algorithm maintains a population memory bank that records optimal solutions and decision paths, allowing octopus individuals to use the memory bank to guide their search direction. To prevent excessive reliance on specific strategies, a tentacle

forgetting mechanism is introduced:

$$M_i^{t+1} = \lambda M_i^t + (1 - \lambda) \bar{M}_i^t \tag{4}$$

Where M_i^t is the tentacle memory, $\bar{M}_i^t$ is the new memory, and λ is the forgetting coefficient that controls memory retention and forgetting, maintaining strategy diversity. Additionally, we integrate the forgetting mechanism with the tentacle growth and degradation mechanism. When dependence exceeds a threshold, the tentacle degrades, promoting the growth of new tentacles, and maintaining global search capability [33-34]. This design enhances the algorithm's robustness and global optimization capability.

By designing an effective tentacle forgetting mechanism combined with the growth and degradation mechanism, the OIO algorithm retains useful prior knowledge while avoiding over-reliance on a single strategy, thereby improving the algorithm's robustness and global optimization capability. The core code is shown in Pseudocode 2.

Pseudocode 2 Pseudo-Memetic Memory Model

1: Initialize memory M with random instances
2: Set $t = 0$
3: **while** not converged **do**
4:     Increment t
5:     $best\_solution, best\_position \leftarrow$ FindBestSolution(M)
6:     $mean, variance \leftarrow$ CalculateMeanAndVariance(M)
7:     PartitionSearchSpace($M, mean, variance$)
8:     **for** each instance I in M **do**
9:       **if** I has found global best solution more than once **then**
10:         MigratePoorSolutions(M, I)
11:       **end if**
12:       EvaluateValue(I, t)
13:     **end for**
14: **end while**

4.3 Sucker-Tentacle Coordination Mechanism

The Octopus Inspired Optimization (OIO) algorithm, inspired by the octopus nervous system, designs a localized structure simulating the division of labor between suckers and tentacles, improving global optimization efficiency.

In the OIO algorithm, suckers perform local search and direction guidance, with each sucker representing a point in the solution space partition, focusing on local sampling and exploitation.

The workflow includes: initializing with randomly deployed suckers, each sucker performing local perturbations and position updates based on location and objective function values, and the updated suckers interacting through a tentacle mechanism to guide the global search direction.

The tentacle structure is abstracted into a spatial plane covering parts of the solution space, responsible for overall search and information transmission. Multiple tentacles are generated during initialization to cover the solution space. Tentacles adjust their shape through interactions with suckers, exchanging information and influencing each other to achieve efficient overall search.

As shown in Figure 6, the tentacles work in tandem with the suction cups and will dynamically adjust the search area of the tentacles during the iteration process
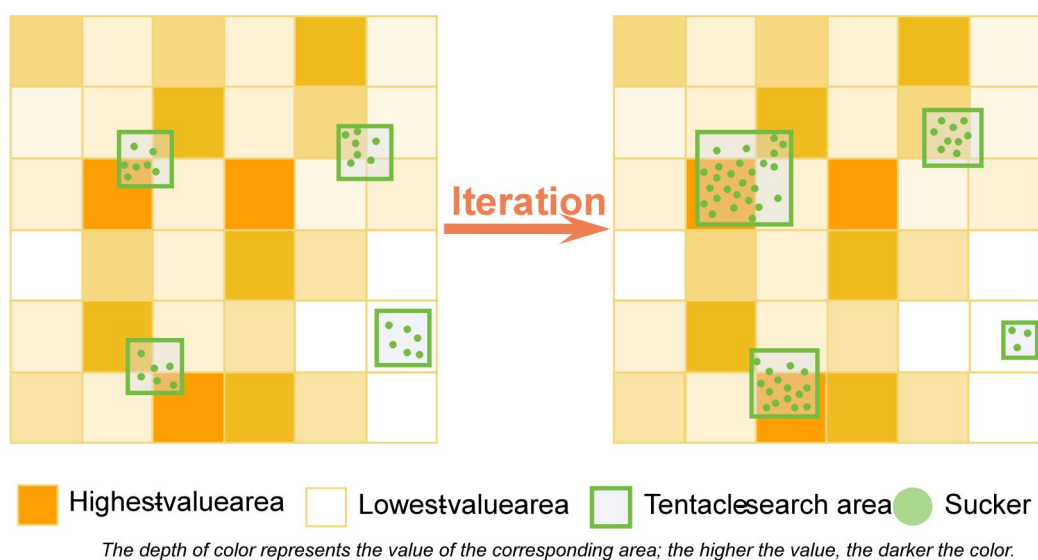

Highest value area Lowest value area Tentacle search area Sucker

The depth of color represents the value of the corresponding area; the higher the value, the darker the color.

Figure 6. Schematic diagram of the iterative process

Through the collaboration of suckers and tentacles, the OIO algorithm maintains a dynamic balance between fine-tuning known good solutions (exploitation) and broadly exploring unknown areas (exploration), enhancing performance. This coordination mechanism fully utilizes the octopus nervous system's high division of labor and precise control characteristics, providing a novel bio-inspired solution for complex optimization problems. The core process of the algorithm can be represented by the pseudocode 3.

Pseudocode 3 Octopus-Inspired Optimization

1: Initialize parameters, instances, memory model
2: **for** each iteration **do**
3:     Perform classification search using a master-slave coordination model
4:     **for** each instance **do**
5:         Retrieve parameter adjustments from the memory model
6:         Perform search based on adjusted parameters
7:         Record search results
8:     **end for**
9:     **for** each search result **do**
10:        **if** result is poor or search range overlaps **then**
11:           Remove the instance from the instances set
12:           Set up new instance at an appropriate position
13:        **end if**
14:     **end for**
15: **end for**

4.4 Adaptive Search and Diversification Strategy

The Octopus Inspired Optimization (OIO) algorithm integrates adaptive search mechanisms and diversification strategies to enhance its adaptability to complex environments and avoid premature convergence .

Adaptive Search Mechanism: Includes tentacle regeneration and plasticity strategies. Tentacle regeneration is achieved through the condition $f(x_i) < \theta_r$, allowing tentacles to regenerate in promising areas and reduce ineffective searches. The plasticity strategy adjusts weights through:

$$w_j = score_j \Big/ \sum_{k=1}^{N_s} score_k \tag{5}$$

enabling the algorithm to autonomously adjust strategy combinations based on previous performance.

Diversification Strategy: Includes asynchronous iteration queue search and multi-objective optimization guidance. Asynchronous queue search mode:

$$x_i^{t+1} = \begin{cases} x_i^t & \text{if } i \notin Q^t \\ \mathcal{U}(x_i^t) & \text{if } i \in Q^t \end{cases} \tag{6}$$

enhances population diversity, avoiding concentration on non-global optima due to synchronous updates. Multi-objective optimization:

$$\min_{x \in \Omega} \mathbf{F}(x) = (f_1(x), f_2(x), \ldots, f_m(x))^{\mathrm{T}} \qquad (7)$$

helps the algorithm maintain population diversity while effectively avoiding excessive convergence to boundaries.

Through adaptive adjustments of tentacle regeneration and plasticity strategies, combined with asynchronous iteration and multi-objective optimization, the OIO algorithm demonstrates excellent global search capabilities and robust convergence performance, effectively addressing dynamic and complex optimization problems.

5. Experimental Section

5.1 Basic Information of the Experiment

To evaluate the performance of the Octopus Inspired Optimization (OIO) algorithm, we selected a series of representative test functions, such as the multimodal Rastrigin function and the unimodal Easom function. These test functions are diverse and representative, commonly used in the industry to examine the performance of algorithms in handling different types of optimization problems. As shown in Table 1 below, we recorded their mathematical expressions, function names, and function domains according to their classifications. Based on the dimensional characteristics of the functions, we categorized the test functions into four types: Fixed Dimension (F), Random Dimension(R), Unimodal (U), and Multimodal (M).

Fixed dimensionality functions are used to test the algorithm's ability to optimise at a specific complexity; random dimensionality functions assess the algorithm's adaptability and flexibility with respect to dimensional changes; unimodal functions help analyse the algorithm's efficiency in local search, while multimodal functions are used to verify the algorithm's ability to avoid local optimums and to explore the potential solution space in a global search. This categorisation ensures that the performance of OIO algorithms in all standard scenarios is fully and fairly tested.

Additionally, to better compare the time efficiency, resource utilization, and other parameters of the algorithms horizontally, we also selected several classic heuristic algorithm models such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) as control groups. The control groups were tested multiple times in parallel under the same server environment and with the same hyperparameters to ensure fairness. The details of the operating system and hardware configuration are as follows:

Operating System and Software Environment:
 - Python 3.7
 - CUDA 11.1

- Pytorch 1.8.1

- Ubuntu 18.04

- NVCC 11.1

- OpenMPI 4.0.0

Speed Test Hardware Configuration:

- GPU: NVIDIA GeForce RTX 3090, 24GB Graphics Memory

- CPU: Intel® Xeon® Gold 6230R Processor, 24 cores, 35.75M Cache, 2.10 GHz

- Memory: 120GB RAM

- Storage: 1500GB ROM

Accuracy Test Hardware Configuration:

- CPU: Intel® Xeon® Gold 6230R Processor, 1 core, 35.75M Cache, 2.10 GHz

- Memory: 4GB RAM

- Storage: 150GB ROM

Table 1:Test Functions

Function	Mathematical expression (d is the dimension)	Function Name	Min	Search Scope	Feature
f_1	$-20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)} + 20 + e$	ACKLEY	0	$[-32,32]^d$	RM
f_2	$\sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}}) + 1$	Griewank Function	0	$[-600,600]^d$	RM
f_3	$\left(\sin^2(\pi w_1) + \sum_{i=1}^{d-1}((w_i-1)^2(1+10\sin^2(\pi w_i+1))) + (w_d-1)^2(1+\sin^2(2\pi w_d))\right)$	Levy Function	0	$[-10,10]^d$	RM
f_4	$10d + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$	Rastrigin Function	0	$[-5.12,5.12]^d$	RM
f_5	$418.98288727243369n - \sum_{i=1}^{n}x_i\sin(\sqrt{|x_i|})$	Schwefel Function	0	$[-500,500]^d$	RM
f_6	$(x_1-1)^2 + \sum_{i=2}^{n}i(2x_i^2 - x_{i-1})^2$	Dixon-Price Function	0	$[-10,10]^d$	RM

f_7	$\sum_{i=1}^{n} i\,x_i^2$	Sum Squares Function	0	$[-10,10]^d$	RU
f_8	$\sum_{i=1}^{n}(x_i-1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	Trid Function	$-d(d+4)(d-1)$	$[-d^2,d^2]^d$	RU
f_9	$\sum_{i=1}^{n-1}[100\,(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	Rosenbrock Function	0	$[-5,10]^d$	RU
f_{10}	$\sum_{i=1}^{n} x_i^2$	Sphere Function	0	$[-5.12,5.12]^d$	RU
f_{11}	$\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j^2\right)$	Rotated Hyper-Ellipsoid Function	0	$[-65,65]^d$	RU
f_{12}	$\sum_{i=1}^{n} |x_i|^{\,i+1}$	Sum of Different Powers Function	0	$[-1,1]^d$	RU
f_{13}	$\dfrac{1}{2}+\dfrac{\sin^2(x_1^2-x_2^2)-0.5}{(1+0.001\,(x_1^2+x_2^2))^2}$	Schaffer Function N. 2	0	$[-100,100]^2$	FM
f_{14}	$(1.5-x_1+x_1 x_2)^2+(2.25-x_1+x_1 x_2^2)^2+(2.625-x_1+x_1 x_2^3)^2$	Beale Function	0	$[-4.5,4.5]^2$	FM
f_{15}	$[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1 x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1 x_2+27x_2^2)]$	Goldstein-Price Function	3	$[-2,2]^2$	FM
f_{16}	$-\dfrac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5\,(x_1^2+x_2^2)+2}$	Drop-Wave Function	-1	$[-5.12,5.12]^2$	FM
f_{17}	$\sin^2(3\pi x_1)+(x_1-1)^2(1+\sin^2(3\pi x_2))+(x_2-1)^2(1+\sin^2(2\pi x_2))$	Levy Function N. 13	0	$[-10,10]^d$	FM
f_{18}	$-\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$	Easom Function	-1	$[-100,100]^2$	FU

5.2 Experimental Parameters and Performance Testing

Considering the requirements of actual production and physical realities, our speed and accuracy tests were conducted in a three-dimensional physical field to solve extremum problems.

5.2.1 Speed Test Setup

To ensure efficiency while controlling the minimum search unit number comparable to other algorithms, we set the following experimental parameters:

Number of Iterations: 50 iterations for the population, a number sufficient for

convergence across various algorithms.

Octopus Configuration: Initialization with 4 octopuses, each equipped with 5 tentacles.

Tentacles and Suckers: Each tentacle is equipped with 50 suckers, and each tentacle undergoes 200 iterations.

Adaptive Parameter Adjustment: Given that the OIO algorithm employs adaptive hyperparameter adjustment, we set thresholds for related hyperparameters (total_range = max_range - min_range, where max_range and min_range are the upper and lower limits of the search space, respectively). The hyperparameter ranges are shown in Table 2:

Table2. Parameter constraints

Tentacle Rebirth Parameters	(20,40,0.06,0.08)
Expansion Factor Range	(0.4,1.6)
Radius Range	(0.02*total_range, 0.3*total_range)
Number of Individual Tentacle Suckers	(10,400)
Range of Tentacle Iterations	(20,400)

To test the speed performance of the OIO algorithm, we set acceptable threshold values for the search results at 1e-10, 1e-12, 1e-14, and 1e-16. The search ends as soon as the algorithm finds a point meeting the threshold condition, and the time consumed for this process is recorded. These thresholds represent 5 progressive experimental groups, each group undergoing 10 parallel tests consecutively.

5.2.2 Accuracy Test

In the accuracy test, to increase the experimental difficulty and demonstrate the performance of a single octopus in a two-dimensional search problem, we reduced the configuration from the original four octopuses to a single octopus. This adjustment was made to investigate the robustness and effectiveness of the OIO algorithm under more challenging conditions. The

experiment finely adjusts the number of tentacle iterations and suckers, dividing it into the following three experimental groups, with each group undergoing ten independent tests to ensure the reliability and repeatability of the results:

    (a) Experimental Group 1: Single tentacle with 50 suckers, 200 iterations.

    (b) Experimental Group 2: Single tentacle with 40 suckers, 180 iterations.

    (c) Experimental Group 3: Single tentacle with 30 suckers, 160 iterations.

Under these settings, we adjust the step size for tentacle iterations and the number of suckers to 20 and 10, respectively, to accurately evaluate the algorithm's sensitivity to parameter changes. This testing configuration allows us to thoroughly investigate the algorithm's optimization capabilities under various parameter settings, particularly its performance under extreme or boundary conditions.

5.3 Speed Benchmark and Methodology

This study involves various population optimization algorithms, including Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Genetic Algorithm (GA), Artificial Immune System (AIS), Differential Evolution (DE), and Octopus Inspired Optimization (OIO). To ensure consistency and fairness, the total number of particles for all algorithms is uniformly set to 1000 (calculated as 4*5*50), and the total number of iterations is set to 10000 (calculated as 50*200). Referencing the research by D. Karaboga et al. [1], in this experiment, the mutation rate for GA is set to 0.01, the crossover rate to 0.9, and the generational gap selection to 0.9. For PSO, the cognitive and social influence factors are both set to 1.8, and the inertia weight is set to 0.6.

The acceptable threshold values for speed testing are set to 1e-8, 1e-10, 1e-12, 1e-14, and 1e-16. Whenever an algorithm finds a solution below these thresholds, the search is immediately terminated and the time consumed is recorded. For each threshold, all algorithms undergo 10 independent continuous runs to ensure the reliability and consistency of the results.

5.4 CPU Usage Test

Given that AIS, ABC, and GA did not find solutions meeting the threshold in previous experiments, this part of the test will exclude these three algorithms. Although DE was able to find solutions meeting the threshold, it took too long, rendering it unsuitable for further CPU usage comparison. This section aims to investigate whether the speed improvement of the OIO algorithm is due to higher CPU usage. To this end, we designed an experiment where both OIO and PSO algorithms optimize 12 test functions under the set acceptable threshold of 1e-14, recording CPU usage throughout the experiment. This experiment not only measures algorithm performance but also evaluates their resource efficiency in practical operations, providing a more comprehensive perspective on the applicability of the algorithms in real-world scenarios.

6. Results

## 6.1 Octopus Inspired Optimization Performance

The OIO speed test is measured in terms of elapsed time and its mean and standard deviation are shown in Table 3.

When the acceptable threshold is decreasing, the OIO elapsed time shows a more significant increase in some of the functions (e.g., Rosenbrock, Trid, Griewank, etc.), but all of them are able to find the acceptable threshold.

The OIO accuracy test was performed 10 consecutive times for each function separately, and the statistical means and standard deviations are shown in Table 4. (50,200) indicates a tentacle with 50 suckers and 200 iterations; the same is true for the remaining parameters.

According to Table 4 it can be found that the OIO algorithm has excellent performance in terms of accuracy for all types of test functions in 2 dimensions and is extremely stable.

The convergence plot of the octopus optimisation algorithm for different test functions in the accuracy test is shown in the second column of figure 6. To ensure that the OIO algorithm does not fall into any other local minima in the accuracy experiments, we visualise the positions of the final suction cups and tentacles. x represents the position of the centre of the tentacle, the square box represents the tentacle coverage, the red dots represent the suction cups, and the star-shaped indicator represents the target position. An example of (50, 200) is shown in the third column of figure 7.
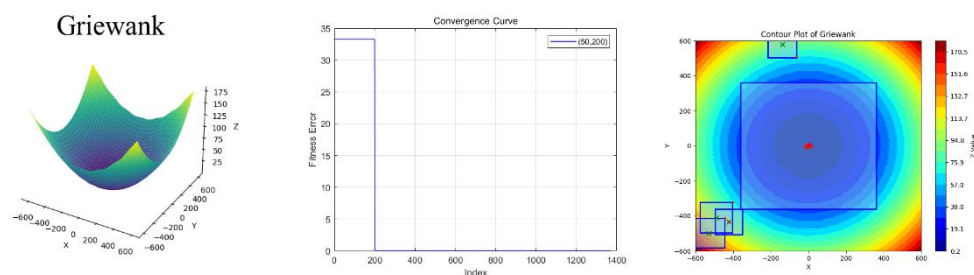
Levy
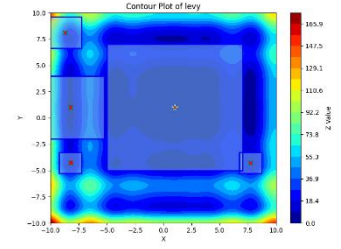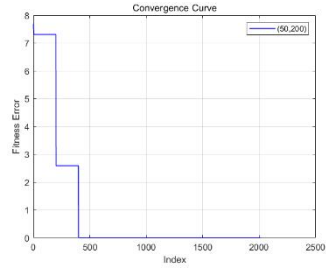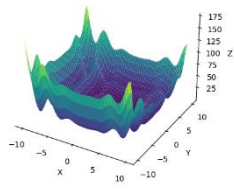
Rastrigin
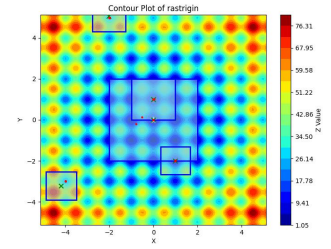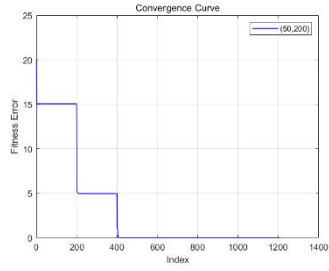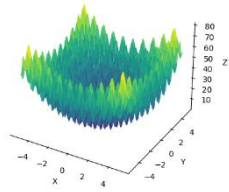
Dixon_Price
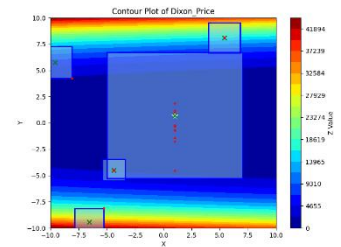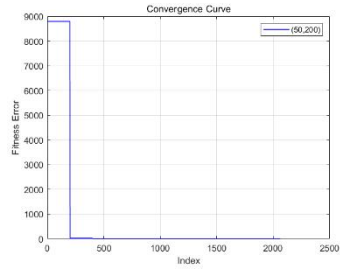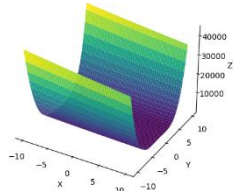
Sum_square

Trid

Rosenbrock

Sphere

Convergence Curve
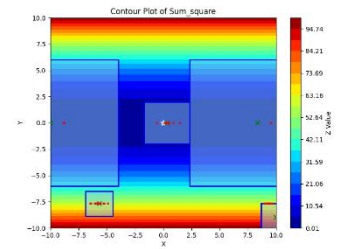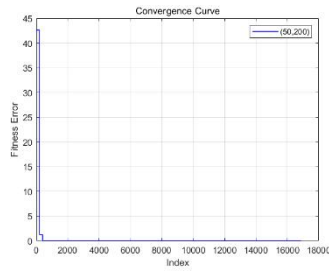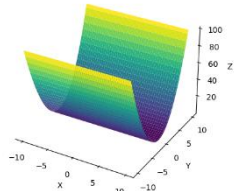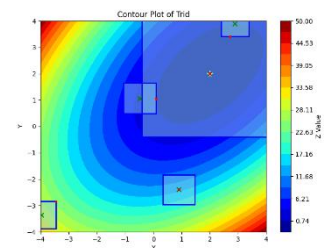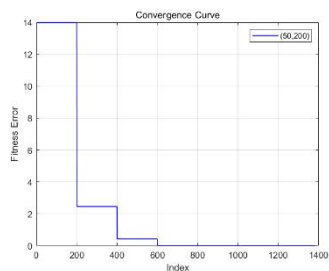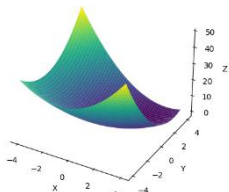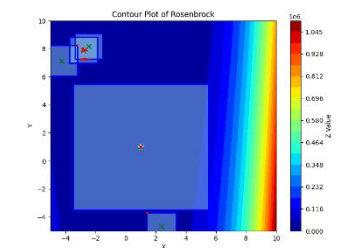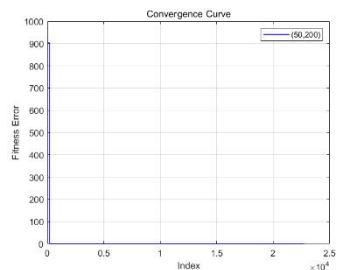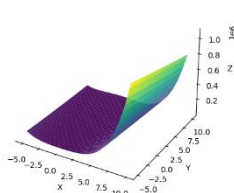
Contour Plot of sphere

Rotated_Hyper_Ellipsoid

Convergence Curve
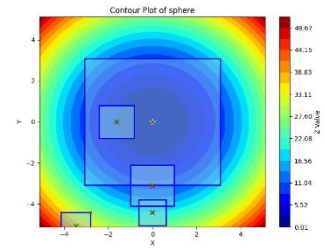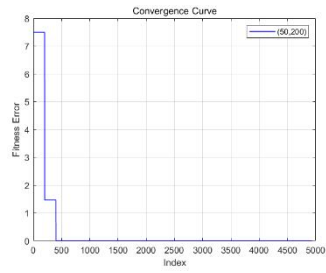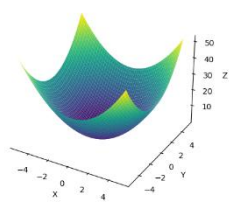
Contour Plot of Rotated_Hyper_Ellipsoid

Sum_of_different_powers

Convergence Curve
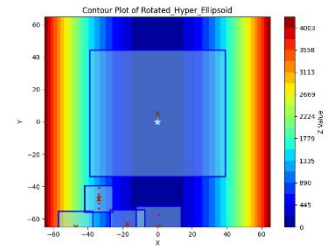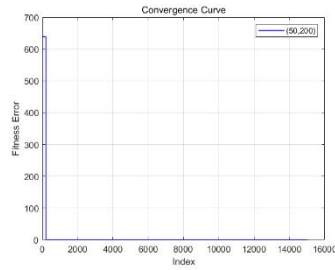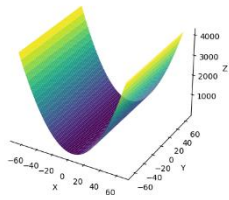
Contour Plot of sum_of_different_powers

Schaffer_function_n2

Convergence Curve

Contour Plot of schaffer_function_n2

Beale

Convergence Curve

Contour Plot of Beale

Drop_wave

Convergence Curve
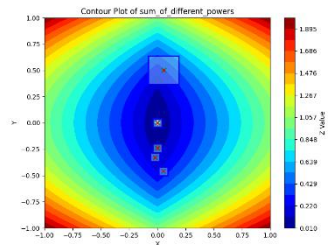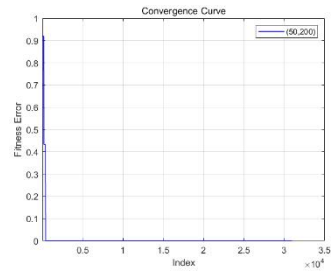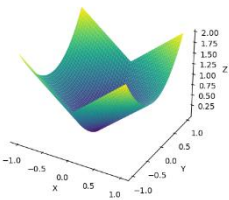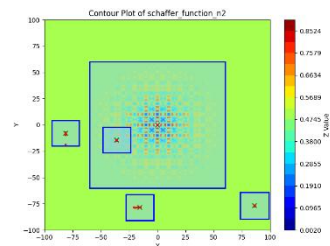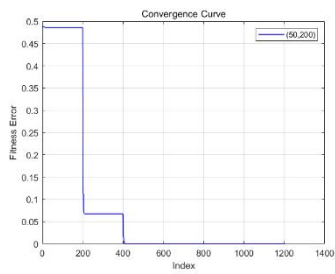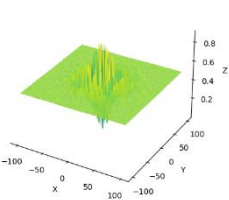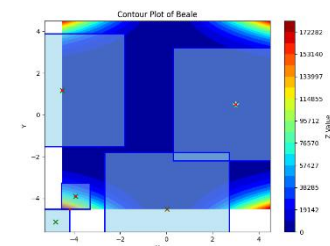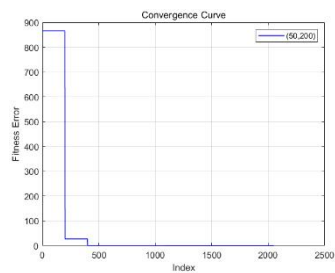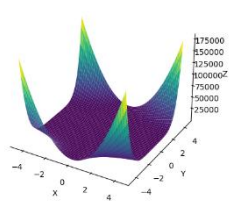
Contour Plot of drop_wave
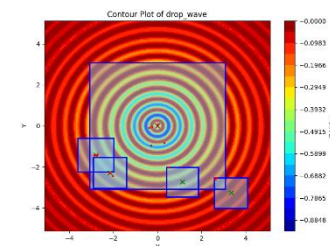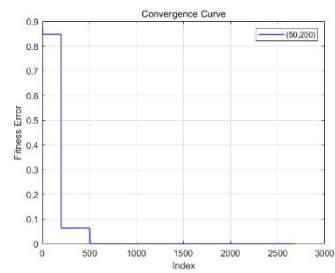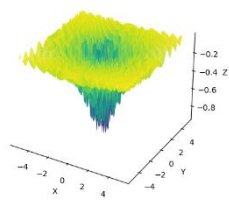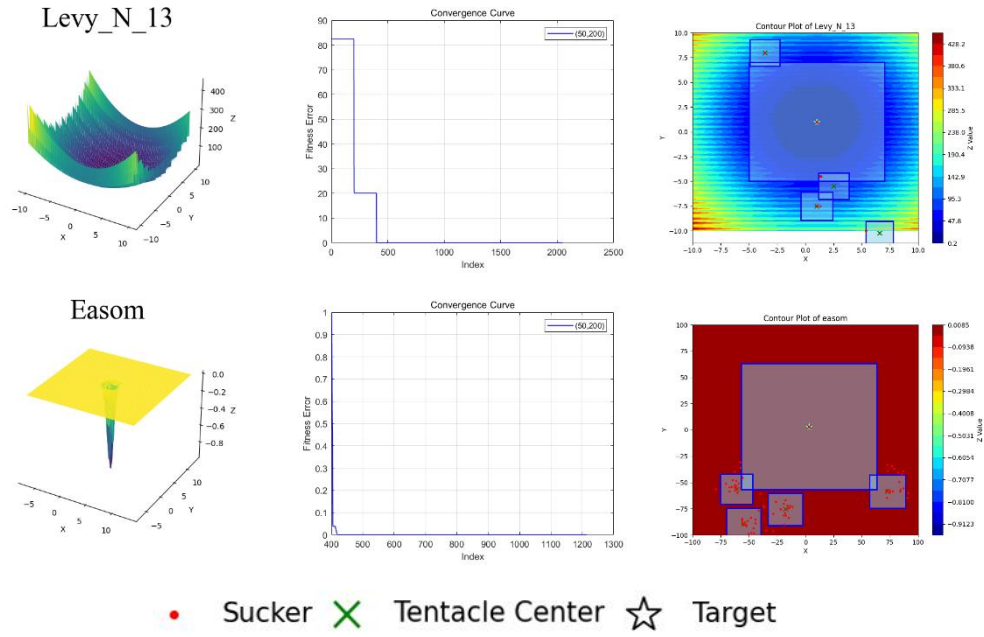
Figure7. Search Result

Based on the visualisation it can be found that the OIO algorithm does not overlap the tentacle positions at the same point when running. This is due to the region regeneration competition mechanism designed in the OIO algorithm. Although there is still some overlap in the region covered by the tentacles, the tentacles with less adaptation will be forced to randomly respawn at any point in the search space in the next iteration. This not only avoids the situation of multiple tentacles searching the same region repeatedly, but also improves the ability of the OIO algorithm to jump out of local minima.

6.2 Speed cross-comparison results

Tables 5, 6, 7, 8 and 9 summarise the mean and standard deviation of the elapsed time for each algorithm for each test function and compare the elapsed time while ensuring that acceptable thresholds were found, and the data with the lowest elapsed time will be marked in bold. Referring to the experimental tables in the appendix it can be seen that GA, AIS, and ABC do not satisfy the thresholds in the vast majority of cases and therefore are not involved in the comparison.
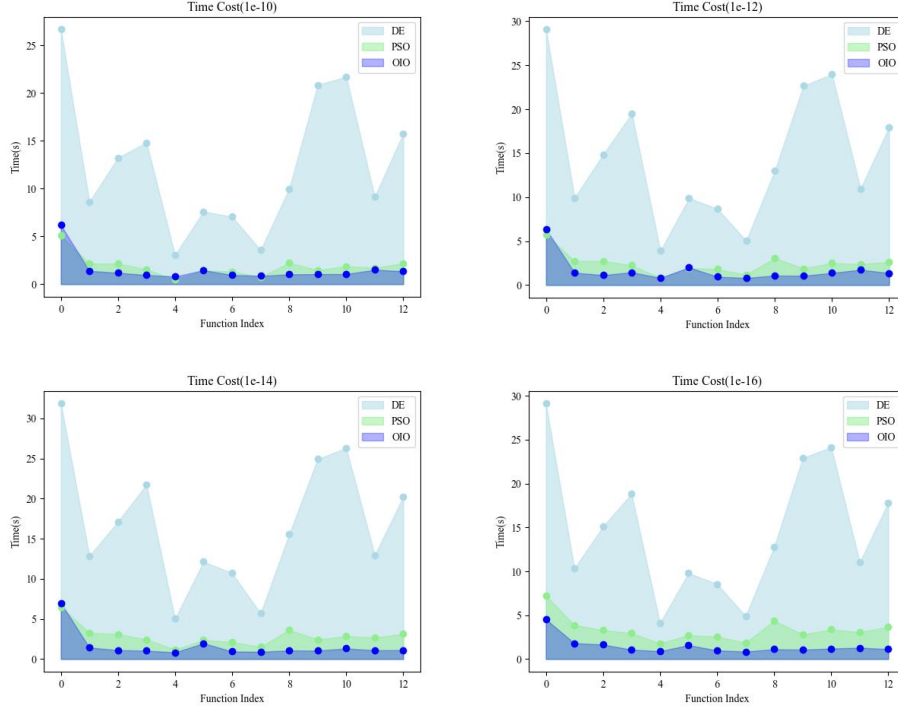
Figure8. Search Speed Comparison Graph

Figure 8 shows the speed comparison graphs obtained in the four sets of speed tests. We can intuitively see that in the vast majority of cases, OIO takes less time than PSO and DE. in the four sets of tests, OIO achieves an average speed of 1.69 times that of PSO and 9.30 times that of DE. In the single-peak function used above, the average speed of OIO reaches 1.89 times that of PSO and 8.63 times that of DE, and in the multimodal function, the average speed of OIO reaches 1.59 times that of PSO and 9.63 times that of DE. Therefore, we can infer that OIO outperforms DE and PSO in terms of speed, whether it is used to search for multimodal or single-peak functions, and when the threshold is set to 1e-14, the average speed of OIO opens the largest gap with DE, reaching 10.39 times that of DE. And when the threshold is set to 1e-16, the average speed of OIO opens the biggest gap with PSO, reaching 2.27 times of PSO.

6.3 CPU Usage Test Results

Considering the significantly longer time consumption of the DE algorithm compared to PSO and OIO in reaching the acceptable threshold, the comparison of CPU usage rates will no longer include the DE algorithm. To rigorously investigate the impact of algorithms on CPU usage, we first run the threads responsible for recording CPU usage separately 10 times, each for 2 minutes, to obtain the baseline system load. Through measurements from the 10 experiments, the average

baseline load is calculated to be 0.65%.

Table11:Average total CPU usage (%)

	Average resource usage			Resource consumption after debasement	
	OIO	PSO		OIO	PSO
Test1	4.340299	4.801111		3.690299	4.151111
Test2	4.332787	4.68625		3.682787	4.03625
Test3	4.616883	4.78022		3.966883	4.13022
Test4	4.806944	4.761538		4.156944	4.111538
Test5	4.460274	4.845455		3.810274	4.195455
Test6	4.464634	4.73956		3.814634	4.08956
Test7	4.746667	4.690217		4.096667	4.040217
Test8	4.496667	4.628889		3.846667	3.978889
Test9	4.271698	4.412088		3.621698	3.762088
Test10	4.501124	4.722093		3.851124	4.072093

In the ten experiments, the average net CPU usage rates for OIO and PSO were 3.85% and 4.05%, respectively. OIO demonstrates an advantage in CPU usage, achieving faster results without significantly increasing CPU computational resources. This efficiency arises from OIO's optimized scheduling and resource allocation, enhancing computational efficiency and reducing CPU load. Consequently, even during high-intensity computing tasks, the system can maintain lower resource consumption. This characteristic makes OIO more competitive in scenarios requiring rapid processing of large amounts of data with limited resources. Therefore, OIO not only outperforms PSO in computational speed but also optimizes CPU resource usage, making it suitable for scenarios with limited computational resources or requiring efficient energy consumption ratios.
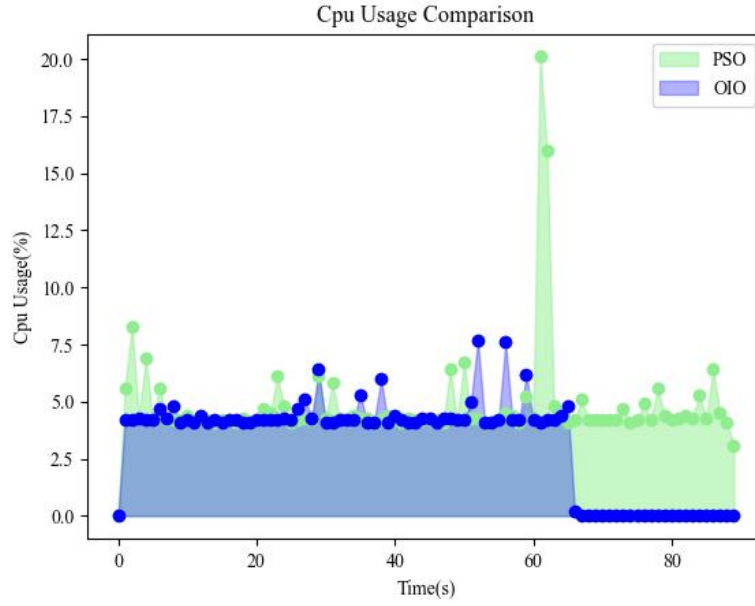
Figure10. CPU total usage area chart

Figure 10 presents a comparison of the total CPU usage between OIO and PSO during the first experiment. We can visually observe that the peak CPU usage of the OIO algorithm is smaller than that of PSO, and OIO also terminates the search process earlier. The frequent significant spikes in CPU usage observed during the search process of PSO further confirm the superiority of the OIO algorithm in reducing resource usage.

7. Conclusion and Discussion

This paper presents a novel bionic intelligence optimisation algorithm, the Octopus Inspired Optimization (OIO) algorithm, inspired by the unique neuromorphic features of the octopus, including hierarchical structure, decentralised interactions, and region regeneration.The design of the OIO algorithm not only reflects a deep understanding of individual and collective intelligence, but also has important foresight in the field of heuristic algorithms.

7.1 Performance and Efficiency

Experimental results show that the OIO algorithm outperforms established algorithms such as Particle Swarm Optimisation (PSO) and Differential Evolution (DE) in terms of speed, accuracy and robustness. In particular, at a tighter threshold 1e-16, OIO achieves an average speed of 2.27

times that of PSO and 9.63 times that of DE on multimodal functions. For example, in the test of the Griewank function, when the threshold is set to 1e-16, the convergence time of the OIO is 4.53 seconds, which is much better than that of the PSO at 7.24 seconds and that of the DE at 29.12 seconds. In addition, when the threshold is set to 1e-14, the average speed of OIO opens the largest gap with DE, reaching 10.39 times that of DE.

7.2 Algorithm Adaptation and Resource Utilisation

A distinguishing feature of the OIO algorithm is its adaptability. The algorithm effectively balances exploration and exploitation through its multi-layer search and asynchronous parallelism mechanisms, adapting to a wide range of optimisation problems. In addition, the region regeneration mechanism enhances the algorithm's ability to escape from local optima by allowing less adaptable tentacles to randomly regenerate, thus promoting diversity in the search process.

OIO demonstrates significant resource efficiency in CPU usage tests. In ten experiments, OIO has a lower average CPU usage compared to PSO, 3.85% versus 4.05%, respectively. This result shows that OIO does not additionally use more CPU resources while getting results faster, showing its energy saving and efficiency.

7.3 Directions for Follow-up Research

The success of OIO raises several implications for the field of optimisation. Firstly, it highlights the potential for bio-inspired algorithms to be more effective than traditional methods in solving complex and dynamic problems. Second, the principles demonstrated by OIO can be extended to other domains, such as robotics and decision support systems, where adaptive and efficient problem solving capabilities are valuable.

For future research, exploring the combination of OIO with machine learning approaches may enhance its predictive capabilities, allowing for more efficient handling of dynamically changing environments. In addition, extending the algorithm to handle multi-objective

optimisation problems may significantly expand its application in various disciplines where navigating trade-offs between conflicting objectives is required.

7.4 Conclusion

In summary, the Octopus Inspired Optimization (OIO) algorithm represents an important advancement in the field of heuristic algorithms, providing a robust, efficient, and adaptable approach to solving optimisation problems. Its design is inspired by the complex neuromorphic structure of the octopus, providing a new perspective on the synthesis of individual and collective intelligence in artificial systems.The continued development and application of OIO is expected to have a far-reaching impact in a number of areas of theoretical research and practical problem solving.

Reference

[1]Kokash, N. (2005). An introduction to heuristic algorithms. Department of Informatics and Telecommunications, 1-8.

[2]Beheshti, Z., & Shamsuddin, S. M. H. (2013). A review of population-based meta-heuristic algorithms. Int. j. adv. soft comput. appl, 5(1), 1-35.

[3]Eiselt, H. A., Sandblom, C. L., Eiselt, H. A., & Sandblom, C. L. (2000). Heuristic algorithms. Integer Programming and Network Models, 229-258.

[4]Ibarra, O. H., & Kim, C. E. (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM (JACM), 24(2), 280-289.

[5]Silver, E. A. (2004). An overview of heuristic solution methods. Journal of the operational research society, 55, 936-956.

[6]Ball, M. O. (2011). Heuristics based on mathematical programming. Surveys in Operations Research and Management Science, 16(1), 21-38.

[7]Kennedy, J. (2006). Swarm intelligence. In Handbook of nature-inspired and innovative computing: integrating classical models with emerging technologies (pp. 187-219). Boston, MA: Springer US.

[8]Dorigo, M., Birattari, M., Garnier, S., Hamann, H., de Oca, M. M., Solnon, C., & Stützle, T. (2007). Swarm intelligence. Scholarpedia, 2(9), 1462.

[9]Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE computational intelligence magazine, 1(4), 28-39.

[10]Ribeiro, C. C., Hansen, P., Maniezzo, V., & Carbonaro, A. (2002). Ant colony optimization: an overview. Essays and surveys in metaheuristics, 469-492.

[11]Marini, F., & Walczak, B. (2015). Particle swarm optimization (PSO). A tutorial. Chemometrics and Intelligent Laboratory Systems, 149, 153-165.

[12]Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. Soft computing, 22, 387-408.

[13]Cervero, F., & Tattersall, J. E. H. (1986). Somatic and visceral sensory integration in the thoracic spinal cord. Progress in brain research, 67, 189-205.

[14]Furness, J. B., Callaghan, B. P., Rivera, L. R., & Cho, H. J. (2014). The enteric

nervous system and gastrointestinal innervation: integrated local and central control. Microbial endocrinology: The microbiota-gut-brain axis in health and disease, 39-71.

[15]Karaboga, D. (2010). Artificial bee colony algorithm. scholarpedia, 5(3), 6915.

[16]Jahjouh, M. M., Arafa, M. H., & Alqedra, M. A. (2013). Artificial Bee Colony (ABC) algorithm in the design optimization of RC continuous beams. Structural and Multidisciplinary Optimization, 47, 963-979.

[17]Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.

[18]Faris, H., Aljarah, I., Al-Betar, M. A., & Mirjalili, S. (2018). Grey wolf optimizer: a review of recent variants and applications. Neural computing and applications, 30, 413-435.

[19]Su, Y., Liu, L., & Lei, Y. (2021). Structural damage identification using a modified directional bat algorithm. Applied Sciences, 11(14), 6507.

[20]Bajaj, A., Sangwan, O. P., & Abraham, A. (2022). Improved novel bat algorithm for test case prioritization and minimization. Soft Computing, 26(22), 12393-12419.

[21]King, A. J., & Cowlishaw, G. (2009). Leaders, followers, and group decision-making. Communicative & integrative biology, 2(2), 147-150.

[22]Wong, E. M., Ormiston, M. E., & Tetlock, P. E. (2011). The effects of top management team integrative complexity and decentralized decision making on corporate social performance. Academy of Management Journal, 54(6), 1207-1228.

[23]Li, C., Xu, H., & Fan, S. (2020). Synergistic effects of self-optimization and imitation rules on the evolution of cooperation in the investor sharing game. Applied Mathematics and Computation, 370, 124922.

[24]Zhou, B., Chan, K. W., Yu, T., & Chung, C. Y. (2013). Equilibrium-inspired multiple group search optimization with synergistic learning for multiobjective electric power dispatch. IEEE Transactions on Power Systems, 28(4), 3534-3545.

[25]Tramacere, F., Beccai, L., Kuba, M., Gozzi, A., Bifone, A., & Mazzolai, B. (2013). The morphology and adhesion mechanism of Octopus vulgaris suckers. PLoS One, 8(6), e65074.

[26]Garcia, A. (2010). Comparative study of the morphology and anatomy of octopuses of the family Octopodidae (Doctoral dissertation, Auckland University of Technology).

[27]Amor, M. D., Norman, M. D., Roura, A., Leite, T. S., Gleadall, I. G., Reid, A., ... & Strugnell, J. M. (2017). Morphological assessment of the Octopus vulgaris species complex evaluated in light of molecular-based phylogenetic inferences. Zoologica Scripta, 46(3), 275-288.

[28]Ferreira, J., Callou, G., Josua, A., Tutsch, D., & Maciel, P. (2019). An artificial neural network approach to forecast the environmental impact of data centers. Information, 10(3), 113.

[29]Assent, I. (2012). Clustering high dimensional data. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(4), 340-350.

[30]Muja, M., & Lowe, D. G. (2014). Scalable nearest neighbor algorithms for high dimensional data. IEEE transactions on pattern analysis and machine intelligence, 36(11), 2227-2240.

[31]Aras, S., Gedikli, E., & Kahraman, H. T. (2021). A novel stochastic fractal search algorithm with fitness-distance balance for global numerical optimization. Swarm and Evolutionary Computation, 61, 100821.

[32]Buche, D., Schraudolph, N. N., & Koumoutsakos, P. (2005). Accelerating evolutionary algorithms with Gaussian process fitness function models. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 35(2), 183-194.

[33]Cheng, C. T., Wang, W. C., Xu, D. M., & Chau, K. W. (2008). Optimizing hydropower reservoir operation using hybrid genetic algorithm and chaos. Water Resources Management, 22, 895-909.

[34]Shen, H. X., Casalino, L., & Luo, Y. Z. (2015). Global search capabilities of indirect methods for impulsive transfers. The Journal of the Astronautical Sciences, 62, 212-232.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) NA
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) yes
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper (yes/no) yes

Does this paper make theoretical contributions? (yes/no) yes

If yes, please complete the list below.

- All assumptions and restrictions are stated clearly and formally. (yes/partial/no) yes
- All novel claims are stated formally (e.g., in theorem statements). (yes/partial/no) yes
- Proofs of all novel claims are included. (yes/partial/no) yes
- Proof sketches or intuitions are given for complex and/or novel results. (yes/partial/no) yes
- Appropriate citations to theoretical tools used are given. (yes/partial/no) yes
- All theoretical claims are demonstrated empirically to hold. (yes/partial/no/NA) yes
- All experimental code used to eliminate or disprove claims is included. (yes/no/NA) yes

Does this paper rely on one or more datasets? (yes/no) no

If yes, please complete the list below.

- A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA)
- All novel datasets introduced in this paper are included in a data appendix. (yes/partial/no/NA)
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no/NA)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes/no/NA)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes/partial/no/NA)
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (yes/partial/no/NA)

Does this paper include computational experiments? (yes/no) yes

If yes, please complete the list below.

- Any code required for pre-processing data is included in the appendix. (yes/partial/no). yes
- All source code required for conducting and analyzing the experiments is included in a code appendix. (yes/partial/no) yes
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no) yes
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) yes
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes/partial/no/NA) yes
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes/partial/no) yes
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes/partial/no) yes
- This paper states the number of algorithm runs used to compute each reported result. (yes/no) yes
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes/no) yes
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes/partial/no) no
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes/partial/no/NA) partial
- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes/partial/no/NA) partial

Supporting Materials

Table 3.OIO elapsed time (time/s)

Function	Acceptable	1.00E-08	1.00E-10	1.00E-12	1.00E-14	1.00E-16
Griewank	Mean	5.741724	6.188067	6.36652	6.999679	4.533972
	Std./Mean	0.812342	0.912028	0.714705	0.745545	0.534402
Levy	Mean	1.684242	1.400081	1.398468	1.429915	1.806029
	Std./Mean	0.567357	0.125137	0.019156	0.009957	0.499387
Rastrigin	Mean	1.10316	1.194469	1.135208	1.113722	1.657665
	Std./Mean	0.150621	0.034087	0.192189	0.193329	0.637898

Dixon_Price	Mean	1.406462	0.963744	1.453894	1.044185	1.074503
	Std./Mean	0.651836	0.150012	0.622538	0.014187	0.01571
Sum_Square	Mean	0.737754	0.791872	0.813084	0.832736	0.898983
	Std./Mean	0.303963	0.267169	0.24071	0.260441	0.199391
Trid	Mean	1.735214	1.464731	2.011118	1.964295	1.581761
	Std./Mean	0.427796	0.107016	0.489038	0.509212	0.013935
Rosenbrock	Mean	1.007514	1.66239	4.230964	8.0003	9.78229
	Std./Mean	0.668706	0.545263	0.645342	0.397721	0.057744
Sphere	Mean	0.939359	0.966379	0.946484	0.951027	0.985048
	Std./Mean	0.00851	0.009343	0.138287	0.137066	0.137804
Rotated_Hyper_Ellipsoid	Mean	0.811142	0.870507	0.804912	0.904883	0.852174
	Std./Mean	0.220457	0.154054	0.258098	0.142306	0.239823
Sum_Of_Different_Powers	Mean	0.931518	1.031799	1.082279	1.071653	1.123941
	Std./Mean	0.140066	0.014347	0.017394	0.123393	0.11667
Schaffer_Function_N.2	Mean	1.01016	1.057394	1.056161	1.05103	1.071514
	Std./Mean	0.021466	0.010542	0.023057	0.141186	0.128248
Beale	Mean	5.570163	5.487116	4.170626	5.267392	3.904259
	Std./Mean	0.59432	0.493344	0.792101	0.746996	0.887484
Drop_Wave	Mean	0.96957	1.06181	1.38016	1.325116	1.204039
	Std./Mean	0.193394	0.119166	0.525539	0.569184	0.058057
Levy_N.13	Mean	0.963707	1.509239	1.74332	1.088987	1.290307
	Std./Mean	0.15146	0.634362	0.624015	0.017901	0.564722
Easom	Mean	1.080786	1.357562	1.365644	1.112898	1.143276
	Std./Mean	0.015286	0.560356	0.538549	0.137081	0.130198

Table 4-1 Error table for changing the number of suction cups

Function	Metric	30,200	40,200	50,200	60,200	70,200
Ackley	Mean	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16
	Std.	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Griewank	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std.	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Levy	Mean	1.50E-32	1.50E-32	1.50E-32	1.50E-32	1.50E-32
	Std.	2.88E-48	2.88E-48	2.88E-48	2.88E-48	2.88E-48
Rastrigin	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std.	0	0	0	0	0

Function	Metric					
Schwefel	Mean	-2.30E-13	-2.30E-13	-2.30E-13	-2.30E-13	-2.30E-13
	Std.	0	0	0	0	0
Dixon_Price	Mean	2.47E-32	2.47E-32	2.47E-32	2.47E-32	2.47E-32
	Std.	5.77E-48	5.77E-48	5.77E-48	5.77E-48	5.77E-48
Sum_Square	Mean	1.20E-76	1.03E-78	1.21E-77	1.71E-78	2.26E-76
	Std.	2.16E-76	1.11E-78	1.38E-77	4.37E-78	5.50E-76
Trid	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std.	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rosenbrock	Mean	7.70E-31	7.96E-31	1.40E-30	9.68E-31	1.21E-30
	Std.	8.44E-31	2.08E-30	2.79E-30	1.64E-30	2.28E-30
Sphere	Mean	3.76E-62	1.83E-61	4.55E-62	1.13E-61	1.54E-61
	Std.	3.98E-62	2.20E-61	2.96E-62	1.20E-61	1.31E-61
Rotated_Hyper_Ellipsoid	Mean	1.49E-76	5.69E-76	5.28E-74	3.50E-76	3.20E-76
	Std.	2.05E-76	1.17E-75	1.09E-73	4.21E-76	6.45E-76
Sum_Of_Different_Powers	Mean	5.77E-38	3.79E-38	4.04E-38	5.04E-38	1.62E-37
	Std.	1.14E-37	5.17E-38	7.62E-38	1.06E-37	2.46E-37
Schaffer_Function_N.2	Mean	0	0	0	0	0
	Std.	0	0	0	0	0
Beale	Mean	0	0	0	0	0
	Std.	0	0	0	0	0
Goldstein_Price	Mean	-8.10E-14	-8.10E-14	8.05E-14	-8.00E-14	-8.00E-14
	Std.	1.93E-16	2.11E-16	1.69E-16	7.78E-16	1.26E-16
Drop_Wave	Mean	0	0	0	0	0
	Std.	0	0	0	0	0
Levy_N.13	Mean	1.35E-31	1.35E-31	1.35E-31	1.35E-31	1.35E-31
	Std.	2.31E-47	2.31E-47	2.31E-47	2.31E-47	2.31E-47
Easom	Mean	0	0	0	0	0
	Std.	0	0	0	0	0

Table 4-2 .Error table for changing the number of iterations

Function	Metric	50,160	50,180	50,200	50,220	50,240
Ackley	Mean	4.44E-16	4.44E-16	4.44E-16	4.44E-16	4.44E-16
	Std.	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Griewank	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std.	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Levy	Mean	1.50E-32	1.50E-32	1.50E-32	1.50E-32	1.50E-32
	Std.	2.88E-	2.88E-	2.88E-	2.88E-	2.88E-48

		48	48	48	48	
Rastrigin	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std.	0	0	0	0	0
Schwefel	Mean	-2.3E-13	-2.3E-13	-2.3E-13	-2.3E-13	-2.3E-13
	Std.	0	0	0	0	0
Dixon_Price	Mean	2.47E-32	2.47E-32	2.47E-32	2.47E-32	2.47E-32
	Std.	5.77E-48	5.77E-48	5.77E-48	5.77E-48	5.77E-48
Sum_Square	Mean	4.72E-78	4.21E-77	1.21E-77	1.04E-76	7.64E-77
	Std.	7.78E-78	1.05E-76	1.38E-77	2.08E-76	1.60E-76
Trid	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std.	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rosenbrock	Mean	9.32E-31	8.01E-31	1.40E-30	1.84E-31	5.69E-31
	Std.	1.48E-30	9.35E-31	2.79E-30	2.78E-31	7.85E-31
Sphere	Mean	2.05E-61	1.02E-61	4.55E-62	6.51E-62	1.04E-61
	Std.	4.20E-61	1.09E-61	2.96E-62	8.57E-62	1.31E-61
Rotated_Hyper_Ellipsoid	Mean	2.37E-76	4.57E-76	5.28E-74	4.18E-76	3.67E-76
	Std.	4.85E-76	6.27E-76	1.09E-73	5.23E-76	1.12E-75
Sum_Of_Different_Powers	Mean	1.59E-37	8.35E-38	4.04E-38	4.94E-38	1.50E-37
	Std.	3.82E-37	1.36E-37	7.62E-38	8.02E-38	3.02E-37
Schaffer_Function_N.2	Mean	0	0	0	0	0
	Std.	0	0	0	0	0
Beale	Mean	0	0	0	0	0
	Std.	0	0	0	0	0
Goldstein_Price	Mean	-8E-14	-8.1E-14	8.05E-14	-8E-14	-8E-14
	Std.	1.69E-16	2.11E-16	1.69E-16	3.54E-16	3.95E-16
Drop_Wave	Mean	0	0	0	0	0
	Std.	0	0	0	0	0
Levy_N.13	Mean	1.35E-31	1.35E-31	1.35E-31	1.35E-31	1.35E-31
	Std.	2.31E-47	2.31E-47	2.31E-47	2.31E-47	2.31E-47
Easom	Mean	0	0	0	0	0
	Std.	0	0	0	0	0

Table 5.Table of elapsed time （Acceptable:1e-8）

Function	Metric	PSO	DE	OIO	AIS	GA	ABC
Griewank	Mean	**4.150111**	24.76257	5.741724	22.71635	308.1214	650.3767
	Std./Mean	0.129429	0.023299	0.812342	0.001335	0.000253	0.001058
Levy	Mean	**1.353939**	6.764352	1.684242	12.94923	197.457	510.7123
	Std./Mean	0.195083	0.072718	0.567357	0.047874	0.016349	0.001055
Rastrigin	Mean	1.682712	11.09525	**1.10316**	6.611436	141.8939	424.2892
	Std./Mean	0.101307	0.043616	0.150621	0.002962	0.013819	0.001309
Dixon_Price	Mean	**1.161798**	11.72637	1.406462	2.723215	101.751	362.4495
	Std./Mean	0.111377	0.109422	0.651836	0.006024	0.018707	0.001459
Sum_Square	Mean	**0.294829**	1.859835	0.737754	0.082874	58.79886	2.189965
	Std./Mean	0.442125	0.265529	0.303963	1.097617	0.733724	0.311005
Trid	Mean	**1.032021**	5.812206	1.735214	3.948256	113.3602	397.34
	Std./Mean	0.108503	0.092045	0.427796	0.003102	0.000537	0.003336
Rosenbrock	Mean	1.813837	14.37608	**1.007514**	2.741521	100.7789	358.5214
	Std./Mean	0.091199	0.042316	0.668706	0.007522	0.000259	0.001759
Sphere	Mean	**0.937772**	5.22159	0.939359	2.643382	100.3239	355.6785
	Std./Mean	0.05752	0.08406	0.00851	0.013105	0.009964	0.001872
Rotated_Hyper_Ellipsoid	Mean	**0.487025**	2.459416	0.811142	0.100037	79.33034	3.367999
	Std./Mean	0.288342	0.302576	0.220457	0.767783	0.370295	0.329116
Sum_Of_Different_Powers	Mean	1.474839	7.293044	**0.931518**	1.851535	91.07604	341.1465
	Std./Mean	0.135683	0.063349	0.140066	0.01159	0.023916	0.001331
Schaffer_function_N.2	Mean	1.040904	17.69117	**1.01016**	4.52061	119.0329	380.94
	Std./Mean	0.106365	0.071751	0.021466	0.002505	0.000314	0.001219
Beale	Mean	**1.090452**	8.004226	5.570163	3.407871	107.4548	351.8095
	Std./Mean	0.084624	0.077231	0.59432	0.004779	0.031899	0.002247
Drop_Wave	Mean	1.505673	20.33826	**0.96957**	5.640463	130.9503	387.2101
	Std./Mean	0.066389	0.02885	0.193394	0.003744	0.037512	0.001539
Levy_N.13	Mean	1.282261	7.457867	**0.963707**	3.771574	111.3559	372.5849
	Std./Mean	0.094228	0.021857	0.15146	0.001279	0.023494	0.001603
Easom	Mean	1.613746	14.00182	**1.080786**	6.323894	138.4347	406.2244
	Std./Mean	0.146929	0.033615	0.015286	0.001705	0.006713	0.001152

Table 6. Table of elapsed time （Acceptable:1e-10）

Function	Metric	PSO	DE	OIO	AIS	GA	ABC
Griewank	Mean	**5.120116**	26.6586	6.188067	23.75453	326.3725	750.3454
	Std./Mean	0.165551	0.026642	0.912028	0.042972	0.029374	0.00092
Levy	Mean	2.156131	8.630126	**1.400081**	13.35455	210.41	593.0117
	Std./Mean	0.071349	0.057354	0.125137	0.04991	0.022777	0.000933
Rastrigin	Mean	2.164205	13.23251	**1.194469**	6.820684	149.4181	495.4236
	Std./Mean	0.067422	0.02743	0.034087	0.001938	0.017215	0.001214
Dixon_Price	Mean	1.560746	14.82201	**0.963744**	2.693343	104.5691	424.1581
	Std./Mean	0.142253	0.093025	0.150012	0.003187	0.020812	0.001901
Sum_Square	Mean	**0.487518**	3.067631	0.791872	0.825671	95.75756	4.2847
	Std./Mean	0.456657	0.110409	0.267169	1.107406	0.294717	0.476724
Trid	Mean	**1.44406**	7.604722	1.464731	4.136783	120.2932	460.4977
	Std./Mean	0.086068	0.104111	0.107016	0.00104	0.027689	0.001312
Rosenbrock	Mean	2.427417	17.32397	**1.66239**	2.733574	104.3866	422.1888
	Std./Mean	0.105989	0.023064	0.545263	0.021404	0.014805	0.001283
Sphere	Mean	1.332293	7.076579	**0.966379**	2.777539	105.5117	420.2953
	Std./Mean	0.072704	0.064156	0.009343	0.033969	0.014218	0.000776
Rotated_Hyper_Ellipsoid	Mean	**0.769272**	3.574589	0.870507	1.058892	96.62095	9.481768
	Std./Mean	0.291967	0.155919	0.154054	0.826095	0.013199	0.183518
Sum_Of_Different_Powers	Mean	2.23992	9.985262	**1.031799**	1.790721	92.69818	407.6053
	Std./Mean	0.073515	0.022813	0.014347	0.001579	0.010642	0.001089
Schaffer_function_N.2	Mean	1.472666	20.84638	**1.057394**	4.478356	125.6203	451.7385
	Std./Mean	0.062299	0.020838	0.010542	0.003539	0.013461	0.000684
Beale	Mean	**1.538096**	11.03384	5.487116	3.443204	110.9274	411.3184
	Std./Mean	0.09564	0.033855	0.493344	0.02687	0.011191	0.001504
Drop_Wave	Mean	1.890358	21.70964	**1.06181**	5.536673	136.0943	412.6167
	Std./Mean	0.111611	0.054839	0.119166	0.001858	0.021377	0.078579
Levy_N.13	Mean	1.716771	9.159263	**1.509239**	3.643332	116.0413	375.4466
	Std./Mean	0.097177	0.078981	0.634362	0.002394	0.014216	0.001116
Easom	Mean	2.196847	15.77514	**1.357562**	6.203204	142.2855	412.1029
	Std./Mean	0.064935	0.043593	0.560356	0.002803	0.010149	0.000392

Table 7.Table of elapsed time（Acceptable:1e-12）

Function	Metric	PSO	DE	OIO	AIS	GA	ABC
Griewank	Mean	**5.758299**	29.04583	6.36652	22.71843	320.4427	642.5988
	Std./Mean	0.093315	0.038646	0.714705	0.001661	0.003166	0.000707
Levy	Mean	2.724082	9.928845	**1.398468**	12.34368	201.4017	506.4917
	Std./Mean	0.068631	0.071212	0.019156	0.002691	0.002483	0.001381
Rastrigin	Mean	2.741055	14.85306	**1.135208**	6.518731	144.5299	420.5592
	Std./Mean	0.055742	0.04426	0.192189	0.001236	0.005335	0.001339
Dixon_Price	Mean	2.240487	19.48927	**1.453894**	2.724012	103.3059	359.5647
	Std./Mean	0.068441	0.097374	0.622538	0.00165	0.00312	0.001023
Sum_Square	Mean	0.842899	3.897546	**0.813084**	2.739547	102.8596	8.685232
	Std./Mean	0.375862	0.209218	0.24071	0.001655	0.004469	0.247585
Trid	Mean	**1.92432**	9.877959	2.011118	3.923895	118.6116	393.2193
	Std./Mean	0.069481	0.0843	0.489038	0.001014	0.005168	0.001561
Rosenbrock	Mean	**3.461905**	20.70815	4.230964	2.74874	102.5651	359.026
	Std./Mean	0.077531	0.030162	0.645342	0.001861	0.003454	0.000765
Sphere	Mean	1.812573	8.656818	**0.946484**	2.602898	101.9811	356.5148
	Std./Mean	0.053603	0.043488	0.138287	0.003237	0.005663	0.001663
Rotated_Hyper_Ellipsoid	Mean	1.1708	5.045653	**0.804912**	2.019528	93.1532	11.07505
	Std./Mean	0.211255	0.062912	0.258098	0.004327	0.003877	0.206102
Sum_Of_Different_Powers	Mean	3.087205	13.00794	**1.082279**	1.831578	90.79551	340.2233
	Std./Mean	0.051939	0.018289	0.017394	0.005751	0.003216	0.001151
Schaffer_function_N.2	Mean	1.889367	22.65672	**1.056161**	4.433171	119.6673	381.5241
	Std./Mean	0.071266	0.034615	0.023057	0.001642	0.0038	0.000754
Beale	Mean	**2.065522**	13.40765	4.170626	3.41407	107.3414	354.7624
	Std./Mean	0.045787	0.059634	0.792101	0.002669	0.00293	0.001151
Drop_Wave	Mean	2.51205	23.96628	**1.38016**	5.541868	128.8584	388.5166
	Std./Mean	0.083472	0.030999	0.525539	0.002167	0.004874	0.001792
Levy_N.13	Mean	2.370653	10.96674	**1.74332**	3.785729	110.9691	373.6242
	Std./Mean	0.047936	0.093355	0.624015	0.00167	0.003777	0.000731
Easom	Mean	2.661788	17.96128	**1.365644**	6.320979	137.3055	407.3709
	Std./Mean	0.089779	0.025593	0.538549	0.004496	0.002609	0.000656

Table 8.Table of elapsed time（Acceptable:1e-14）

Function	Metric	PSO	DE	OIO	AIS	GA	ABC
Griewank	Mean	**6.501565**	31.83202	6.999679	23.45174	321.5481	640.1049
	Std./Mean	0.067926	0.018931	0.745545	0.001231	0.028744	0.001482
Levy	Mean	3.277244	12.85423	**1.429915**	12.80924	208.8177	507.7562
	Std./Mean	0.071585	0.024544	0.009957	0.002518	0.024776	0.001425
Rastrigin	Mean	3.111076	17.11876	**1.113722**	6.880304	147.0575	421.9303
	Std./Mean	0.050817	0.035539	0.193329	0.00442	0.001316	0.001066
Dixon_Price	Mean	2.447319	21.73384	**1.044185**	2.823881	106.298	360.3045
	Std./Mean	0.069938	0.122666	0.014187	0.003067	0.016454	0.000596
Sum_Square	Mean	1.199011	5.019165	**0.832736**	2.840054	105.379	13.22378
	Std./Mean	0.16827	0.091475	0.260441	0.001713	0.017287	0.218352
Trid	Mean	2.375264	12.13656	**1.964295**	4.140257	119.7874	393.7792
	Std./Mean	0.036267	0.044306	0.509212	0.002638	0.021225	0.00125
Rosenbrock	Mean	**4.103158**	23.87009	8.0003	2.813445	105.8452	359.1283
	Std./Mean	0.047667	0.050567	0.397721	0.00193	0.019351	0.000448
Sphere	Mean	2.126858	10.72363	**0.951027**	2.674969	104.8224	357.4763
	Std./Mean	0.077852	0.040592	0.137066	0.002481	0.024217	0.001172
Rotated_Hyper_Ellipsoid	Mean	1.535709	5.68187	**0.904883**	2.078267	97.87827	16.62687
	Std./Mean	0.178987	0.125927	0.142306	0.002699	0.017016	0.136557
Sum_Of_Different_Powers	Mean	3.644701	15.53834	**1.071653**	1.853026	93.34616	342.0732
	Std./Mean	0.045579	0.018577	0.123393	0.002464	0.018168	0.000843
Schaffer_Function_N.2	Mean	2.444901	24.91335	**1.05103**	4.587549	123.7221	381.9203
	Std./Mean	0.051015	0.027116	0.141186	0.00142	0.00459	0.002107
Beale	Mean	**2.487051**	16.70951	5.267392	3.432029	109.8748	354.1861
	Std./Mean	0.048967	0.045322	0.746996	0.001399	0.011214	0.001359
Drop_Wave	Mean	2.859051	26.29869	**1.325116**	5.690425	133.5877	389.2991
	Std./Mean	0.059839	0.028364	0.569184	0.001781	0.006785	0.001408
Levy_N.13	Mean	2.665374	12.94997	**1.088987**	3.974641	111.3743	375.7356
	Std./Mean	0.064717	0.032108	0.017901	0.002366	0.001232	0.000879
Easom	Mean	3.129896	20.24751	**1.112898**	6.357879	144.0042	414.082
	Std./Mean	0.048175	0.020506	0.137081	0.003913	0.014414	0.001883

Table 9.Table of elapsed time （Acceptable:1e-16）

Function	Metric	PSO	DE	OIO	AIS	GA	ABC
Griewank	Mean	7.24397359	29.11755	4.533972	23.84387412	314.5254	666.8623
	Std./Mean	0.02502536	0.043305	0.534402	0.002918369	0.033835	0.001468
Levy	Mean	3.88878439	10.32649	1.806029	12.3126343	199.5628	521.6451
	Std./Mean	0.03818811	0.066476	0.499387	0.001963862	0.000436	0.001547
Rastrigin	Mean	3.29537838	15.09525	1.657665	6.677716303	145.2885	434.172
	Std./Mean	0.04389171	0.029072	0.637898	0.00498335	0.000436	0.002237
Dixon_Price	Mean	2.96643715	18.82834	1.074503	2.698420405	101.5288	377.2782
	Std./Mean	0.05310544	0.065887	0.01571	0.002842285	0.000819	0.002939
Sum_Square	Mean	1.78254576	4.123675	0.898983	2.727951646	102.0037	9.649681
	Std./Mean	0.13507084	0.106175	0.199391	0.011082873	0.000562	0.186023
Trid	Mean	2.74078002	9.796526	1.581761	4.017462444	115.4904	409.2938
	Std./Mean	0.0482944	0.071439	0.013935	0.001894483	0.00041	0.002042
Rosenbrock	Mean	5.15432875	20.24328	9.78229	2.751228285	101.7115	375.8151
	Std./Mean	0.10554766	0.033836	0.057744	0.013401405	0.000852	0.002312
Sphere	Mean	2.55244815	8.565935	0.985048	2.625390673	101.1165	368.97
	Std./Mean	0.04708961	0.056631	0.137804	0.018479112	0.000756	0.002475
Rotated_Hyper_Ellipsoid	Mean	1.864943	4.913963	0.852174	1.992052841	94.95243	12.46418
	Std./Mean	0.10278724	0.151184	0.239823	0.022651842	0.035155	0.278332
Sum_Of_Different_Powers	Mean	4.37806702	12.77827	1.123941	1.758338785	89.80916	357.1348
	Std./Mean	0.05397275	0.029037	0.11667	0.004191448	0.013738	0.001533
Schaffer_function_N.2	Mean	2.81845226	22.91213	1.071514	4.489289975	121.939	397.1624
	Std./Mean	0.06273317	0.034797	0.128248	0.005297317	0.000508	0.001909
Beale	Mean	3.02192376	13.76873	3.904259	3.358512115	108.0523	368.7202
	Std./Mean	0.0703482	0.042239	0.887484	0.001552766	0.00065	0.001865
Drop_Wave	Mean	3.37993839	24.14356	1.204039	5.567820978	130.7706	402.7617
	Std./Mean	0.04742879	0.04462	0.058057	0.005100557	0.000303	0.00191
Levy_N.13	Mean	3.07376905	11.03845	1.290307	3.734168887	110.9802	392.4343
	Std./Mean	0.05146962	0.032883	0.564722	0.007472344	0.00069	0.002444
Easom	Mean	3.67135949	17.82801	1.143276	6.344906807	139.5839	422.5718
	Std./Mean	0.03994282	0.027393	0.130198	0.004826897	0.012542	0.004799

Code Open-source Statement

In our research, we have utilized and developed the Octopus Inspired Optimization (OIO) algorithm to address multi-objective optimization problems. To promote academic exchange and further research, we have open-sourced this code on GitHub. Readers and researchers can access and download the code through the link:

https://anonymous.4open.science/r/Octopus_Inspired_Optimization_OIO-4550