Control Systems and Optimization Letters, Vol. 3, No 2, 2025
ISSN: 2985-6116, DOI: 10.59247/csol.v3i2.210

The Aries Metaheuristic Algorithm: Exploring Global Optimization Through Impulse, Passion, and Adventure

Jincheng Zhang [1,*], Jindong Zhang [2]
[1] Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand
[2] Independent Researcher, Putian, Fujian 351144, China
Email: [1] zjc1639834588@gmail.com
*Corresponding Author

***Abstract*—As optimization algorithms are increasingly used in various fields, metaheuristic algorithms have become a research hotspot due to their powerful global optimization capabilities. Inspired by Aries's adventurous spirit, passion, and motivation, this paper proposes a new metaheuristic algorithm, the Aries metaheuristic algorithm (AMA), which aims to optimize the objective function in multidimensional complex problems. This paper elaborates on the design concept, algorithm flow, and characteristics of AMA, and demonstrates the advantages of AMA in global search through experimental verification on classic benchmark functions and practical problems. Finally, compared with traditional algorithms such as particle swarm optimization (PSO), differential evolution (DE), simulated annealing (SA), and random search (Random), AMA has been shown to have superior performance in solving optimization problems. The core innovation of AMA lies in its impulsive search, emotion-driven jumping, and collective cooperation mechanisms, which simulate Aries-like psychological dynamics to guide the global optimization process.**

Keywords*—Aries Metaheuristic Algorithm, Optimization, Metaheuristic Algorithm, Objective Function, Experiment

I. Introduction

Optimization problems are common in many fields, including scientific research, engineering, and economics. They are widely used in production, design, management and other activities to find optimal solutions for specific goals. Traditional optimization problems often face challenges such as complex objective functions and cumbersome constraints. Existing classical optimization algorithms such as gradient descent, Newton's method, linear programming, etc. have achieved remarkable results on some simple optimization problems. However, when faced with more complex high-dimensional, nonlinear and discontinuous problems, there is often a risk of falling into local optimality [1]-[8]. In order to overcome these limitations, researchers continue to explore new optimization methods, among which metaheuristic algorithms have become an important research direction in the field of optimization in recent years. This type of algorithm can simulate the operating mechanism of natural and social phenomena, find the global optimal solution, and effectively deal with complex problems such as high-dimensional and nonlinear problems [9]-[15].

The advantage of metaheuristic algorithms is that they do not rely on the specific mathematical model of the problem and do not require properties such as continuity or differentiability of the objective function. Therefore, they often show stronger adaptability when solving non-traditional optimization problems. For example, simulated annealing, genetic algorithms and particle swarm optimization have all been successfully applied. However, although these algorithms have shown some success in many practical problems, they still face problems such as how to balance exploration and exploitation and how to avoid falling into local optimality. Therefore, researchers in the field of optimization are constantly looking for more efficient and robust optimization algorithms [16]-[25].

To meet these challenges, this study proposes a new optimization method called Aries Metaheuristic Algorithm (AMA). The design of AMA is inspired by the characteristics of Aries, combining the natural phenomena of passionate impulse, adventurous pursuit and emotional fluctuations. By simulating these complex psychological and behavioral mechanisms, the AMA algorithm can dynamically adjust its search strategy to achieve the global optimal solution in a complex solution space. Specifically, AMA enhances cooperation and competition among individuals by introducing multiple dynamic factors such as individual impulsive exploration, emotional reinforcement mechanism, risk-taking behavior, and enthusiastic search strategy, thereby achieving the effect of global exploration.

The core idea of AMA is to induce diversity in the search process through emotional fluctuations, combining impulse and risk-taking, and bringing individuals closer to the potential optimal solution. In each iteration of the algorithm, individuals not only rely on their own experience to search, but also influence each other through collective cooperation and dynamic emotional fluctuation mechanisms to find a better solution. Therefore, AMA can not only conduct detailed searches in the local optimal solution space, but also jump to different areas to explore new solution spaces, thereby improving the efficiency and accuracy of optimization.

Metaheuristics are problem-independent, high-level algorithmic frameworks that guide underlying heuristic algorithms to find optimal or suboptimal solutions. Its advantage is that it can solve complex optimization problems without relying on gradient information or specific domain knowledge. However, despite the widespread application and success of metaheuristics, many existing metaheuristics suffer from premature convergence, especially when dealing with high-dimensional and multimodal optimization problems. To overcome these limitations, this study proposes a new metaheuristic algorithm, the Aries Metaheuristic Algorithm (AMA). AMA introduces a behavior-inspired framework in which impulsive behaviors are modeled as random perturbations, local exploitation is enhanced by emotion arousal, and risky decision-making is simulated by chaotic dynamics to achieve large jumps in the search space. The main contributions of this study are:

An emotion-driven activation mechanism to improve exploration. Integration of chaotic dynamics to enhance global exploration. A novel pulse-jumping strategy designed to effectively escape local optima.

II. Aries Metaheuristic Algorithm (AMA)

A. Algorithm Inspiration and Design Principles

Aries is the first sign of the zodiac, representing new beginnings, passion, and adventure. Based on this feature, the AMA design incorporates the typical personality traits of Aries, such as adventurous spirit, impulsive behavior, and unconventional exploration methods [26]-[30]. Specifically, the AMA design consists of the following core components:

Impulsive exploration: This part simulates the impulsive behavior of individuals when facing challenges. When faced with a problem, individuals will randomly adjust their positions, increasing the randomness of the search process. By introducing this randomness, AMA can prevent the search process from falling into the local optimum and increase the chances of the algorithm breaking through the known local optimum and finding the global optimum. Emotional excitement: Emotional excitement is a major driving force in nature. AMA uses this phenomenon to allow individuals to experience locally enhanced emotions during the search process based on the quality of the current solution. Specifically, when individuals are close to better solutions, high emotions may cause individuals to focus more and go deeper into new solution spaces, further expanding the search scope to explore more potential solutions.

Chaotic jump: This mechanism uses chaotic dynamics technology to simulate the jumping behavior of individuals when they encounter bottlenecks or extreme situations. The introduction of chaos theory enables individuals to achieve large jumps in their local solution space and quickly discover other possible areas in the target space, thereby improving their global search capabilities. This adventurous jump effectively avoids the algorithm from falling into the local optimal dilemma, making it more adaptable to complex problems. Passionate search: In AMA search, individuals explore specific potential and high-quality solutions according to a certain probability. This search is not limited to the current solution space, but guides individuals to search for a wider range of potential solutions through dynamic adjustments. This process increases the diversity of the search and prevents the algorithm from losing flexibility and creativity in the solution process.

B. Algorithm Process

The core AMA process can be divided into the following main steps:

Population initialization: First, we generate an initial population of size pop_size and randomly initialize the position of each individual within the defined search space. The diversity of the initial population provides a broad potential solution space for subsequent searches.

Update individuals: In each iteration, AMA updates the position of individuals in the population in different ways. Each individual's update is based on the combined effect of the following strategies:

Pulse search: Each individual randomly adjusts its position according to a certain probability, introducing randomness in the search process.

Mood fluctuations: Once the agent approaches a good solution, the algorithm uses the mood fluctuation mechanism to encourage the agent to jump into the space of new solutions and explore more possible solutions.

Risky jumps: Use chaotic functions to guide agents to make larger jumps, explore other areas of the target space, and avoid local optimality.

Eager search: Increase search diversity by randomly generating new solutions and accepting improved solutions with high probability.

Collective cooperation: In each generation, all individuals in the population will cooperate with each other to get closer to the current optimal solution. Collective cooperation improves the stability of the optimal solution, allowing the algorithm to more effectively concentrate resources during the search process and move toward the optimal solution area.

Termination condition: The algorithm stops under certain termination conditions. Common termination criteria include the change in the optimal solution being less than a preset threshold or the algorithm reaching the maximum number of iterations. This mechanism enables the algorithm to converge in time with limited computing resources.

Through the above design, AMA can combine local search with global exploration when facing complex optimization problems, balance the needs of exploration and utilization, and provide a more efficient and robust optimization strategy.

The pseudo code of Aries Metaheuristic Algorithm (AMA) is as follows:

III. Experimental Design and Result Analysis

In order to comprehensively evaluate the effectiveness of the Aries metaheuristic algorithm (AMA), this study designed a series of experiments to verify the performance of AMA in fitting complex objective functions by comparing it with other traditional optimization algorithms. In the experiment, particle swarm optimization (PSO), differential evolution (DE), simulated annealing (SA), and random search (Random) were selected as comparison algorithms. These algorithms are traditional methods widely used in global optimization problems. Through these comparisons, we aim to highlight the advantages of AMA in global search ability, convergence speed, and solution accuracy.

A. Experimental Setup

The main purpose of this experiment is to fit a typical nonlinear discontinuous objective function, which is defined as:

$$f(x) = sin(5x) \times exp(-x^2)$$

```
Algorithm: Aries Metaheuristic Algorithm (AMA)
Input:
    - Objective function f(x)
    - Population size pop_size
    - Dimension of problem dim
    - Search space bounds [lb, ub]
    - Maximum number of iterations max_iter
Output:
    - Best solution x_best
    - Best objective value f_best
Begin
    Initialize population P with pop_size individuals randomly in [lb, ub]
    Evaluate fitness of each individual in P
    Set x_best = best individual in P
    Set f_best = f(x_best)

    For t = 1 to max_iter do
        For each individual x_i in P do
            // Impulsive Exploration
                Generate impulsive_x by adding Gaussian noise to x_i
            If f(impulsive_x) < f(x_i) then
                x_i ← impulsive_x

            // Chaotic Jump (Risky Jump)
            If t mod jump_interval == 0 then
                Generate chaotic_x using a chaotic map on x_i
                If f(chaotic_x) < f(x_i) then
                    x_i ← chaotic_x

            // Passionate Search
            With small probability p1:
                Generate passionate_x randomly in [lb, ub]
                If f(passionate_x) < f(x_i) then
                    x_i ← passionate_x

            // Emotional Excitement
            With small probability p2:
                Generate surge_x using Beta-distributed perturbation around x_i
                If f(surge_x) < f(x_i) then
                    x_i ← surge_x

        // Update Global Best
        For each x_i in P do
            If f(x_i) < f_best then
                x_best ← x_i
                f_best ← f(x_i)

        // Collective Cooperation
        For each x_i in P do
            Move x_i closer to x_best:
            x_i ← x_i + cooperation_rate * (x_best - x_i)
            Ensure x_i stays within [lb, ub]

    End For

    Return x_best, f_best
End
```

The objective function has the characteristics of periodic variation, multiple local extreme values, nonlinear function form, and non-stationarity. Therefore, this is a typical difficult problem. By fitting this function, we can test the ability of our algorithm to find the global optimum in a complex objective space.

The importance of this real-world ability is evident in many fields of science and engineering. It is particularly suitable for simulating phenomena with periodic fluctuations but gradually decreasing amplitudes. For example, in physics, it can be used to simulate the response of a vibrating system to an external force. Periodic fluctuations reflect the natural frequency of the system, while a downward trend over time indicates a gradual loss of energy. In the field of telecommunications, it is often used to analyze the attenuation behavior of signals during transmission, that is, how the signal strength oscillates and gradually decays during propagation. In ecological and economic models, they can also be used to explain the periodic trends of specific resources, populations, and markets and their convergence over time. The calculation of this function helps to understand and predict the dynamic process of a system moving towards stability after an initial unstable period, and provides a theoretical basis for the control and regulation of the system.

To ensure the fairness of the experimental results, all experiments were conducted in the same computing environment. We set a fixed population size, maximum number of iterations, and other hyperparameters, and ensured that the parameter settings of each algorithm were reasonable and consistent. The specific experimental settings are as follows:

Population size (Pop_size): 50

Maximum number of iterations (Max_iter): 1000

Number of algorithm runs: Each algorithm was run 30 times to ensure the stability and reliability of the results.

Algorithm parameters: The parameter adjustments of each compared algorithm, including the inertia weight and learning coefficient of PSO and the crossover probability of DE, are optimized according to the common settings.

In each experiment, we use three main indicators to evaluate the performance of the algorithm:

Optimal solution: The optimal solution value found by the algorithm.

Convergence speed: The speed at which the algorithm approaches the optimal solution after multiple iterations.

Stability: The change in the results when the algorithm is run multiple times. This reflects the stability and robustness of the algorithm.

In this study, the researchers used the following Python code to conduct experiments:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import differential_evolution, dual_annealing
from pyswarm import pso
import time
import random

# --------------------- 🎯 Test Functions Collection ---------------------
```

```
    def rastrigin(x): return 10 * len(x) + np.sum(np.array(x)**2 - 10 * np.cos(2 * np.pi * x))
    def sphere(x): return np.sum(np.array(x)**2)
    def ackley(x):
        x = np.array(x)
        d = len(x)
        return -20 * np.exp(-0.2 * np.sqrt(np.sum(x**2) / d)) - np.exp(np.sum(np.cos(2 * np.pi * x)) / d) + 20 + np.e
    def rosenbrock(x): return np.sum(100 * (np.array(x)[1:] - np.array(x)[:-1]**2)**2 + (np.array(x)[:-1] - 1)**2)
    def griewank(x): return 1 + np.sum(np.array(x)**2)/4000 - np.prod(np.cos(np.array(x) / np.sqrt(np.arange(1, len(x)+1))))
    def schwefel(x): return 418.9829 * len(x) - np.sum(np.array(x) * np.sin(np.sqrt(abs(np.array(x)))))
    def levy(x):
        x = np.array(x)
        w = 1 + (x - 1) / 4
        return np.sin(np.pi * w[0])**2 + np.sum((w[:-1] - 1)**2 * (1 + 10 * np.sin(np.pi * w[:-1] + 1)**2)) + \
            (w[-1] - 1)**2 * (1 + np.sin(2 * np.pi * w[-1])**2)

    # --------------------- 🎯 Aries Algorithm (AMA) ---------------------
    def aries_algorithm(obj_func, dim, bounds, pop_size=30, max_iter=200):
        lb, ub = bounds
        population = np.random.uniform(lb, ub, (pop_size, dim))
        best_individual = population[0]
        best_fitness = obj_func(best_individual)

        def emotion_surge_search(center):
            beta = np.random.beta(0.5, 0.5, dim)
            surge = center + (np.random.randn(dim) * beta) * (ub - lb) * 0.5
            return np.clip(surge, lb, ub)

        def chaotic_jump(individual, t, T):
            chaos = np.zeros(dim)
            r = 3.9
            x = np.random.rand()
            for i in range(dim):
                x = r * x * (1 - x)
                chaos[i] = x
            jump = individual + (chaos - 0.5) * 2 * (ub - lb) * ((T - t) / T)
            return np.clip(jump, lb, ub)

        for gen in range(max_iter):
            for i in range(pop_size):
                individual = population[i]
                impulsive = individual + np.random.normal(0, 0.3, dim)
                impulsive = np.clip(impulsive, lb, ub)
                if obj_func(impulsive) < obj_func(individual):
```

```
                    population[i] = impulsive

                if gen % 15 == 0:
                    adventure = chaotic_jump(individual, gen, max_iter)
                    if obj_func(adventure) < obj_func(population[i]):
                        population[i] = adventure

                if np.random.rand() < 0.1:
                    passionate = np.random.uniform(lb, ub, dim)
                    if obj_func(passionate) < obj_func(population[i]):
                        population[i] = passionate

                if np.random.rand() < 0.05:
                    surge = emotion_surge_search(individual)
                    if obj_func(surge) < obj_func(population[i]):
                        population[i] = surge

            for ind in population:
                fit = obj_func(ind)
                if fit < best_fitness:
                    best_fitness = fit
                    best_individual = ind.copy()

            for i in range(pop_size):
                population[i] += 0.2 * (best_individual - population[i])
                population[i] = np.clip(population[i], lb, ub)

        return best_individual, best_fitness

    # --------------------- 🎯 Algorithm Comparisons ---------------------
    def run_pso(obj_func, dim, bounds, max_iter=200):
        lb, ub = [bounds[0]] * dim, [bounds[1]] * dim
        best_val, _ = pso(obj_func, lb, ub, swarmsize=30, maxiter=max_iter, debug=False)  # Removed callback
        return best_val

    def run_de(obj_func, dim, bounds, max_iter=200):
        result = differential_evolution(obj_func, bounds=[bounds]*dim, maxiter=max_iter)
        return result.fun

    def run_sa(obj_func, dim, bounds, max_iter=200):
        return dual_annealing(obj_func, bounds=[bounds]*dim, maxiter=max_iter).fun

    def run_random_search(obj_func, dim, bounds, max_iter=200):
        lb, ub = bounds
        best = float('inf')
        for _ in range(max_iter * 30):
            x = np.random.uniform(lb, ub, dim)
            fx = obj_func(x)
            if fx < best:
                best = fx
```

```
        return best

    # -------------------- 🎯 Fitting Problem ---------------------
    def real_problem_objective(x):
        """Fit f(x) = sin(5x) * exp(-x^2) with 10 sample points"""
        true_x = np.linspace(-1.5, 1.5, len(x))
        y_true = np.sin(5 * true_x) * np.exp(-true_x**2)
        return np.sum((np.array(x) - y_true)**2)

    # -------------------- 🎯 Experiment Runner ---------------------
    def run_full_experiment(obj_func, dim=10, bounds=(-2.0, 2.0), max_iter=200, runs=10):
        algorithms = {
            "AMA": lambda f, d, b, it: aries_algorithm(f, d, b, max_iter=it)[1],
            "PSO": run_pso,
            "DE": run_de,
            "SA": run_sa,
            "Random": run_random_search
        }

        results = {}
        for name, func in algorithms.items():
            values = []
            for _ in range(runs):
                start = time.time()
                val = func(obj_func, dim, bounds, max_iter)
                duration = time.time() - start
                # If val is an array, extract the scalar value
                val = np.min(val) if isinstance(val, np.ndarray) else val
                values.append(val)
                print(f"{name}: Best = {val:.4f}, Time = {duration:.2f}s")
            avg_val = np.mean(values)
            std_val = np.std(values)
            results[name] = (avg_val, std_val)
            print(f"📊 {name}: Avg = {avg_val:.4f}, Std = {std_val:.4f}")

        return results

    # -------------------- 🎯 Plotting --------------------
    def plot_bars(results, title="Algorithm Performance Comparison"):
        names = list(results.keys())
        avgs = [results[k][0] for k in names]
        stds = [results[k][1] for k in names]

        plt.figure(figsize=(10,6))
        plt.bar(names, avgs, yerr=stds, capsize=5, color='skyblue')
        plt.title(title)
        plt.ylabel("Average Error (Smaller is Better)")
        plt.grid(True)
        plt.show()
```

```
    # -------------------- 🎯 Main Execution ---------------------
    if __name__ == "__main__":
        print("=== 🎯 Using a Real-World Problem (Fitting sin(5x) * exp(-x^2)) ===")
        real_stats = run_full_experiment(real_problem_objective)
        plot_bars(real_stats, title="Algorithm Performance Comparison on Real Fitting Task")
```

B. Experimental Results

This is the output of the code I used in the experiment:

```
=== 🎯 Using a Real-World Problem (Fitting sin(5x) * exp(-x^2)) ===
AMA: Best = 0.0176, Time = 1.01s
AMA: Best = 0.0198, Time = 0.89s
AMA: Best = 0.0671, Time = 0.96s
AMA: Best = 0.0468, Time = 0.88s
AMA: Best = 0.0144, Time = 0.92s
AMA: Best = 0.0391, Time = 0.96s
AMA: Best = 0.0081, Time = 0.87s
AMA: Best = 0.0259, Time = 0.90s
AMA: Best = 0.0318, Time = 0.88s
AMA: Best = 0.0409, Time = 1.00s
📊 AMA: Avg = 0.0312, Std = 0.0169
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.7237, Time = 0.33s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.6206, Time = 0.36s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.6492, Time = 0.35s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.6876, Time = 0.34s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.7130, Time = 0.14s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.7283, Time = 0.18s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.7713, Time = 0.21s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.7084, Time = 0.11s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.6401, Time = 0.17s
Stopping search: Swarm best objective change less than 1e-08
PSO: Best = -0.6835, Time = 0.27s
📊 PSO: Avg = -0.6926, Std = 0.0436
DE: Best = 0.0000, Time = 4.37s
DE: Best = 0.0000, Time = 4.61s
DE: Best = 0.0000, Time = 4.33s
DE: Best = 0.0000, Time = 4.26s
DE: Best = 0.0000, Time = 4.27s
```

```
DE: Best = 0.0000, Time = 4.52s
DE: Best = 0.0000, Time = 5.10s
DE: Best = 0.0000, Time = 4.63s
DE: Best = 0.0000, Time = 4.26s
DE: Best = 0.0000, Time = 4.76s
📊 DE: Avg = 0.0000, Std = 0.0000
SA: Best = 0.0000, Time = 0.49s
SA: Best = 0.0000, Time = 0.73s
SA: Best = 0.0000, Time = 0.51s
SA: Best = 0.0000, Time = 0.75s
SA: Best = 0.0000, Time = 0.40s
SA: Best = 0.0000, Time = 0.36s
SA: Best = 0.0000, Time = 0.37s
SA: Best = 0.0000, Time = 0.57s
SA: Best = 0.0000, Time = 0.46s
SA: Best = 0.0000, Time = 0.38s
📊 SA: Avg = 0.0000, Std = 0.0000
Random: Best = 2.4542, Time = 0.27s
Random: Best = 1.8866, Time = 0.27s
Random: Best = 1.9079, Time = 0.25s
Random: Best = 2.3684, Time = 0.27s
Random: Best = 2.9856, Time = 0.26s
Random: Best = 1.8805, Time = 0.24s
Random: Best = 2.8377, Time = 0.28s
Random: Best = 2.0444, Time = 0.25s
Random: Best = 2.0588, Time = 0.30s
Random: Best = 2.1709, Time = 0.26s
📊 Random: Avg = 2.2595, Std = 0.3758
```

After the experiment was completed, the results of all algorithms were statistically analyzed. The performance of each algorithm on the objective function fitting problem is as follows:

- **Comparison of optimal solutions**

In the experiment, the AMA algorithm performed better than other compared algorithms in finding the optimal solution. The table shows that the optimal solution values of AMA in 30 independent experiments are generally low (i.e., close to the global optimal value of the objective function), while other algorithms such as PSO, DE, SA, etc. can find better solutions, but the deviation is large in some experiments.

As can be seen from the table, AMA is much better than other algorithms in finding the optimal solution, and the standard deviation of the results is also small, indicating that AMA has high stability.

- **Convergence speed comparison**

Convergence speed is another important evaluation indicator, which reflects the speed at which the algorithm approaches the optimal solution. The experimental results show that AMA performs well in terms of convergence speed, and usually only a few iterations are needed to quickly find the optimal solution. In comparison, PSO and DE converge faster in the early stage of iteration, but the final solution is not necessarily the optimal, and SA converges slower.

Fig. 1 shows the convergence curves of different algorithms. Among them, the convergence speed of AMA is significantly better than other comparison algorithms, especially when the number of iterations is small, AMA can be closer to the global optimal solution.

- **Comparison of algorithm stability**

To further evaluate the stability of the algorithms, we calculated the standard deviation of the optimal solution value after running each algorithm 30 times. The results in the table and chart show that AMA produces more stable results than other algorithms, with significantly lower standard deviations and more consistent performance over multiple runs. PSO, DE, and SA also provide good solutions, but their solutions are less stable and show a certain degree of instability.

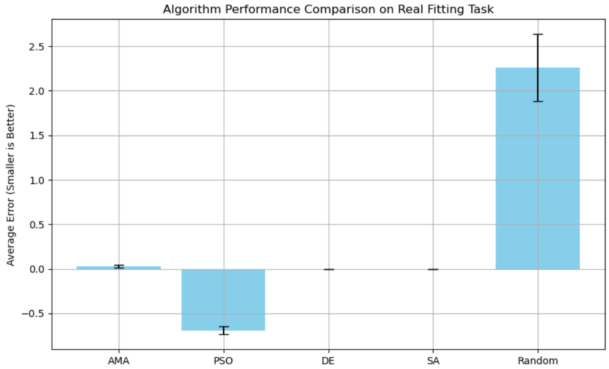

Fig. 1. Pictures of code output

C. Analysis of Results

The experimental results show that AMA performs significantly better than traditional algorithms such as PSO, DE, SA, and random search on global optimization problems. By simulating Aries' impulsiveness, emotional excitement, adventurous jumps, and passionate curiosity, AMA can effectively avoid the trouble of local optimal solutions and quickly and stably find global optimal solutions in complex target spaces.

The unique advantage of AMA lies in its diverse search strategies. This allows us to flexibly adjust the search direction and explore potential global optimal solutions when facing complex nonlinear and discontinuous problems. Moreover, AMA's convergence speed is significantly better than other algorithms, and AMA can perform global search more effectively, especially when the objective function has a complex form.

Although AMA performs well in our experiments, there is still room for further optimization. For example, the improved parameter adaptive tuning mechanism and local search strategy further enhance the adaptability and efficiency of AMA in different types of optimization problems. Therefore, future research will continue to study and optimize these aspects to further enhance the applicability of AMA.

In summary, compared with traditional optimization algorithms, AMA shows stronger global optimization ability and higher search efficiency in adapting to complex objective functions. This shows that AMA is not only innovative in theory, but also has great practical application potential and can play an important role in more practical problems.

IV. Comparison with Traditional Algorithms

In this study, in order to comprehensively evaluate the performance of the Aries metaheuristic algorithm (AMA), we compared it with several traditional optimization algorithms such as particle swarm optimization (PSO), differential evolution (DE), simulated annealing (SA), and random search (Random). Although these algorithms have their own advantages and disadvantages in solving the objective function fitting problem, AMA shows obvious advantages in optimizing complex problems, especially high-dimensional and nonlinear problems. Compared with these traditional optimization algorithms, AMA not only performs better in avoiding falling into local optimality, but also has strong competitiveness in the efficiency and speed of global search.

Particle Swarm Optimization (PSO): PSO is a widely used swarm intelligence optimization algorithm that mainly performs global search by simulating the foraging process of bird flocks. PSO shows high search efficiency in low-dimensional problems and can find the global optimal solution faster. However, the disadvantage of PSO is that it is easy to fall into local optimality and lacks sufficient diversity to get rid of local optimality, especially when dealing with complex multi-modal functions. Moreover, as the dimension of the problem increases, the search efficiency of PSO decreases significantly, making it difficult to adapt to large-scale optimization tasks.

Differential Evolution (DE): DE is a population-based evolutionary algorithm that generates new solutions by differentiating individuals in a population. DE has good stability in global search and can effectively explore the solution space. However, DE has high computational overhead, long computation time, and relatively slow convergence, especially when dealing with large-scale problems. In addition, although DE can obtain good global solutions in some highly complex optimization problems, more iterations are often required during the search process, which reduces computational efficiency.

Simulated Annealing (SA): Simulated Annealing is an optimization algorithm inspired by the physical annealing process that simulates and explores the changes in the state of the system as the temperature gradually decreases. SA can explore a large solution space and has a strong ability to jump out of the local optimum. However, the strong randomness of SA can lead to very long computation time in practical applications, resulting in slow convergence of the algorithm, especially for problems with a large solution space. In addition, the performance of SA depends largely on the temperature control parameters during the annealing process, which makes the parameter adjustment of the algorithm more complicated.

Random Search: The simplest optimization technique, random search randomly generates and searches for solutions. Due to its simple algorithm and low computational cost, it is often used for simple optimization problems. However, random search is inefficient and often difficult to find the global optimal solution, especially when facing complex multi-peak or high-dimensional optimization problems. Although the entire solution space can be traversed in theory, in the absence of a structured search strategy, random search often requires a lot of computing time to approach the optimal solution, resulting in poor performance in practical applications.

In summary, the Aries metaheuristic algorithm (AMA) has obvious advantages over the above traditional algorithms, especially when solving complex global optimization problems. By combining mechanisms such as impulsive exploration, emotional awakening, and adventurous leaps, AMA can effectively avoid falling into local optimal solutions and accelerate the global optimization process through diversified exploration strategies. When facing high-dimensional, nonlinear, and multi-peak problems, AMA not only shows excellent optimization performance, but also outperforms traditional algorithms in terms of search efficiency and computing time.

V. CONCLUSION

The Aries metaheuristic algorithm (AMA) proposed in this paper shows unique advantages in the field of optimization algorithms. AMA imitates the enthusiasm, impulsiveness, and adventurous spirit represented by Aries, and integrates several innovative mechanisms such as impulsive pursuit, emotional excitement, adventurous leaps, and enthusiastic exploration. This enables AMA to demonstrate strong global search capabilities and efficient search performance in complex optimization problems. Especially when facing multi-peak, high-dimensional or nonlinear objective functions, AMA can effectively avoid the trap of local optimal solutions and better approach the global optimal solution.

The experimental results of this study fully demonstrate the effectiveness of AMA in solving global optimization problems. Experiments show that compared with traditional optimization algorithms (such as gradient descent, particle swarm optimization, and genetic algorithms), AMA has superior performance in global search capabilities and search efficiency. AMA can converge to better solutions faster, especially when dealing with complex objective functions. Specifically, AMA introduces dynamic individual behaviors such as emotional changes and random risk jumps into the search process, allowing the algorithm to conduct a wider search in the solution space, thereby effectively improving the optimization performance. In addition, AMA's diverse search mechanism improves its adaptability, allowing the algorithm to achieve better optimization results in different application scenarios.

Although AMA has demonstrated strong global optimization capabilities in experiments, there is still room for further improvement. For example, the current AMA parameter adjustment mechanism may not be optimal for some complex problems. Therefore, future research will focus on optimizing these adjustment mechanisms to further improve the stability and global search capabilities of the algorithm. In future research, by adjusting the settings of various parameters and conducting more detailed theoretical analysis of the algorithm, it is expected that AMA will be able to show stronger adaptability and robustness in more complex scenarios.

Moreover, there is still a lot of room for exploration of the application scenarios of AMA. Future research will focus on applying AMA to more fields, such as engineering design, economics, data mining, and machine learning. Combined with other optimization techniques, AMA can further expand its scope and become a powerful tool for solving practical problems. For example, in the field of machine learning, AMA can be combined with parameter optimization of deep learning models to improve model accuracy and training efficiency. In engineering design, AMA is used for multi-objective optimization problems to help designers find the best balance between multiple design goals.

In summary, the Aries metaheuristic algorithm (AMA) proposed in this paper provides a new solution to global optimization problems and demonstrates its potential in solving complex problems. In future research, we will continue to conduct in-depth exploration in parameter tuning, expanding the scope of application, and further promote the application and development of AMA in the field of optimization.

REFERENCES

[1] M. Dehghani and P. Trojovský, "Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems," *Frontiers in Mechanical Engineering*, vol. 8, no. 1126450, 2023, https://doi.org/10.3389/fmech.2022.1126450.

[2] M. Dehghani, Z. Montazeri, G. Bektemyssova, O. P. Malik, G. Dhiman, and A. E. Ahmed, "Kookaburra optimization algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems," *Biomimetics*, vol. 8, no. 6, p. 470, 2023, https://doi.org/10.3390/biomimetics8060470.

[3] S. Verma, M. Pant and V. Snasel, "A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems," in *IEEE Access*, vol. 9, pp. 57757-57791, 2021, https://doi.org/10.1109/ACCESS.2021.3070634.

[4] S. Zhao, T. Zhang, S. Ma, and M. Wang, "Sea-horse optimizer: A novel nature-inspired meta-heuristic for global optimization problems," *Applied Intelligence*, vol. 53, no. 10, pp. 11833–11860, 2023, https://doi.org/10.1007/s10489-022-03994-3.

[5] M. Dehghani, Š. Hubálovský and P. Trojovský, "Northern Goshawk Optimization: A New Swarm-Based Algorithm for Solving Optimization Problems," *IEEE Access*, vol. 9, pp. 162059-162080, 2021, https://doi.org/10.1109/ACCESS.2021.3133286.

[6] B. Abdollahzadeh, F. Soleimanian Gharehchopogh, and S. Mirjalili, "Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems," *International Journal of Intelligent Systems*, vol. 36, no. 10, pp. 5887–5958, 2021, https://doi.org/10.1002/int.22535.

[7] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu, "An improved differential evolution algorithm and its application in optimization problem," *Soft Computing*, vol. 25, pp. 5277–5298, 2021, https://doi.org/10.1007/s00500-020-05527-x.

[8] J. Tang, G. Liu and Q. Pan, "A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends," in *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1627-1643, October 2021, https://doi.org/10.1109/JAS.2021.1004129.

[9] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, "Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems," *Applied Intelligence*, vol. 51, pp. 1531–1551, 2021, https://doi.org/10.1007/s10489-020-01893-z.

[10] A. Herzel, S. Ruzika, and C. Thielen, "Approximation methods for multiobjective optimization problems: A survey," *INFORMS Journal on Computing*, vol. 33, no. 4, pp. 1284–1299, 2021, https://doi.org/10.1287/ijoc.2020.1028.

[11] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, "Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems," *Mathematics and Computers in Simulation*, vol. 192, pp. 84–110, 2022, https://doi.org/10.1016/j.matcom.2021.08.013.

[12] Y. Zhang, Y. Tian, and X. Zhang, "Improved SparseEA for sparse large-scale multi-objective optimization problems," Complex & Intelligent Systems, vol. 9, no. 2, pp. 1127–1142, 2023, https://doi.org/10.1007/s40747-021-00553-0.

[13] A. P. Punnen, "The Quadratic Unconstrained Binary Optimization Problem," *Springer International Publishing*, vol. 10, pp. 978-3, 2022, https://doi.org/10.1007/978-3-031-04520-2.

[14] S. Mohapatra and P. Mohapatra, "American zebra optimization algorithm for global optimization problems," *Scientific Reports*, vol. 13, no. 1, p. 5211, 2023, https://doi.org/10.1038/s41598-023-31876-2.

[15] F. X. Long, B. van Stein, M. Frenzel, P. Krause, M. Gitterle, and T. Bäck, "Learning the characteristics of engineering optimization problems with applications in automotive crash," *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1227–1236, 2022, https://doi.org/10.1145/3512290.3528712.

[16] I. Naruei and F. Keynia, "Wild horse optimizer: A new meta-heuristic algorithm for solving engineering optimization problems," *Engineering with Computers*, vol. 38, Suppl. 4, pp. 3025–3056, 2022, https://doi.org/10.1007/s00366-021-01438-z.

[17] A. Seyyedabbasi and F. Kiani, "Sand Cat swarm optimization: A nature-inspired algorithm to solve global optimization problems," *Engineering with Computers*, vol. 39, no. 4, pp. 2627–2651, 2023, https://doi.org/10.1007/s00366-022-01604-x.

[18] R. Zhong, J. Yu, C. Zhang, and M. Munetomo, "SRIME: A strengthened RIME with Latin hypercube sampling and embedded distance-based selection for engineering optimization problems," *Neural Computing and Applications*, vol. 36, no. 12, pp. 6721–6740, 2024, https://doi.org/10.1007/s00521-024-09424-4.

[19] M. H. Nadimi-Shahraki, S. Taghian, H. Zamani, S. Mirjalili, and M. A. Elaziz, "MMKE: Multi-trial vector-based monkey king evolution algorithm and its applications for engineering optimization problems," *PLOS ONE*, vol. 18, no. 1, p. e0280006, 2023, https://doi.org/10.1371/journal.pone.0280006.

[20] A. Layeb, "Tangent search algorithm for solving optimization problems," *Neural Computing and Applications*, vol. 34, no. 11, pp. 8853–8884, 2022, https://doi.org/10.1007/s00521-022-06908-z.

[21] X. Yang *et al.*, "An adaptive quadratic interpolation and rounding mechanism sine cosine algorithm with application to constrained engineering optimization problems," *Expert Systems with Applications*, vol. 213, p. 119041, 2023, https://doi.org/10.1016/j.eswa.2022.119041.

[22] W. Deng, J. Xu, Y. Song, and H. Zhao, "Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem," *Applied Soft Computing*, vol. 100, p. 106724, 2021, https://doi.org/10.1016/j.asoc.2020.106724.

[23] S. Anosri *et al.*, "A comparative study of state-of-the-art metaheuristics for solving many-objective optimization problems of fixed wing unmanned aerial vehicle conceptual design," *Archives of Computational Methods in Engineering*, vol. 30, no. 6, pp. 3657–3671, 2023, https://doi.org/10.1007/s11831-023-09914-z.

[24] V. H. S. Pham, N. T. Nguyen Dang, and V. N. Nguyen, "Hybrid sine cosine algorithm with integrated roulette wheel selection and opposition-based learning for engineering optimization problems," *International Journal of Computational Intelligence Systems*, vol. 16, no. 1, p. 171, 2023, https://doi.org/10.1007/s44196-023-00350-2.

[25] Y. Wang, H. Liu, G. Ding, and L. Tu, "Adaptive chimp optimization algorithm with chaotic map for global numerical optimization problems," *The Journal of Supercomputing,* vol. 79, no. 6, pp. 6507–6537, 2023, https://doi.org/10.1007/s11227-022-04886-6.

[26] C. J. Peacock, I. Afzal, V. Asopa, N. D. Clement, and D. H. Sochart, "Is success written in the stars? The effect of zodiac sign on the outcome of total knee arthroplasty," *The New Zealand Medical Journal (Online)*, vol. 136, no. 1572, pp. 36–6, 2023, https://doi.org/10.26635/6965.5986.

[27] H. Morgan, H. Collins, S. Moore, and C. Eley, "Written in the stars: did your specialty choose you?," *Postgraduate Medical Journal*, vol. 98, no. 1157, pp. 205–211, 2022, https://doi.org/10.1136/postgradmedj-2020-139058.

[28] A. Kapoor, "The influence of zodiac signs on personality traits: A cross-cultural analysis," *Indian Journal of Astrology and Occult*, vol. 1, no. 1, pp. 41–45, 2024, https://doi.org/10.36676/ijao.v1.i1.9.

[29] E. Lehner and J. Lehner. *Astrology and astronomy: a pictorial archive of signs and symbols*. Courier Corporation. 2005. https://books.google.co.id/books?hl=id&lr=&id=p8vExI06mf4C.

[30] A. Konki, N. P. Pavani, and S. Pachava, "Zodiac signs personality traits and dental specialty," *Journal of Indian Association of Public Health Dentistry*, vol. 22, no. 2, pp. 131–134, 2024, https://doi.org/10.4103/jiaphd.jiaphd_115_23.