Journal of Computers and Applied Science Education ©2026 ISSN 2409-7284(online)
Copyright ©Ubiquitous International Volume **6**, Number **1**, March **2026**

Genetic Programming based Position Update Formula Generation for Optimization with Application of Feature Selection

Shu-Chuan Chu

Nanjing University of Information Science and Technology, Nanjing, 210044, China
Shandong University of Science and Technology, Qingdao 266590, China
scchu0803@gmail.com

Xiaoping Cao

Shandong University of Science and Technology, Qingdao 266590, China
xiaoping120608@163.com

Xingsi Xue

Fujian University of Technology, Fuzhou 350000, China
jack8375@gmail.com

Han-Chieh Chao

Fo Guang University, Yilan 26247, Taiwan
Tamkang University, New Taipei 25137, Taiwan
UCSI University, Kuala Lumpur 56000, Malaysia
president@mail.fgu.edu.tw

Jeng-Shyang Pan

Nanjing University of Information Science and Technology, Nanjing 210044, China
Shandong University of Science and Technology, Qingdao 266590, China
Chaoyang University of Technology, Taichung City 41349, Taiwan
jspan@ieee.org

Received October 28, 2025, revised January 18, 2026, accepted February 20, 2026.

Abstract. *Feature selection is a crucial step in preprocessing high-dimensional data. It aims to eliminate irrelevant and redundant features, thereby significantly improving classification performance and reducing computational cost. Although evolutionary algorithms demonstrate strong global search capabilities in this field, their performance is often limited by manually designed and fixed-position update strategies. These strategies lack adaptability to complex and dynamic data structures. To overcome this limitation, this paper proposes a novel genetic programming (GP)-based feature selection method. It automatically evolves position update formulas through symbolic expression mechanisms, thereby enhancing the search efficiency and adaptability of the algorithm. Furthermore, a binary version of the proposed method is developed and successfully applied to feature selection tasks on multiple UCI benchmark datasets. Experimental results show that the method consistently outperforms several traditional binary evolutionary algorithms in classification accuracy and feature subset size. This validates the effectiveness and robustness of the GP-based strategy generation approach in complex feature spaces.*
Keywords: Genetic Programming (GP), Evolutionary Algorithm, Optimization Formula Generation, Feature Selection.

1. **Introduction.** With the rapid development of artificial intelligence and big data technologies, the processing and analysis of high-dimensional data have become a focal point of current research. Feature selection is a crucial preprocessing step for high-dimensional data that has attracted extensive attention in recent years. It effectively removes redundant, irrelevant, and noisy features, thereby reducing data dimensionality and enhancing both computational efficiency and model generalization. Feature selection plays a vital role in various practical applications such as medical diagnosis, financial risk control, image recognition, and bioinformatics. Its importance becomes especially prominent in the analysis of high-dimensional sparse data, including gene expression data, text classification, and remote sensing imagery.

Evolutionary Computation (EC), as a population-based global optimization method, simulates natural selection and genetic variation mechanisms, demonstrating excellent search capabilities and adaptability, making it an effective tool for feature selection. Typical EC algorithms include Genetic Algorithm (GA) [1], Particle Swarm Optimization (PSO) [2], and Differential Evolution (DE) [3]. These algorithms utilize diverse evolutionary operators and collaborative population strategies to identify feature subsets with high discriminative power within complex search spaces, thereby enhancing the effectiveness of feature selection. Besides feature selection [4], EC methods have also achieved remarkable success in fields such as image processing [5], computer networks [6], and production scheduling [7].

However, despite their promising performance in feature optimization, these metaheuristic algorithms [8] still face several challenges. In particular, premature convergence to local optima and difficulties in maintaining population diversity are more evident in high-dimensional and complex problems.Most current EC algorithms rely on manually designed position update strategies and fixed mathematical expressions. This "designer-driven" approach limits the algorithms' ability to dynamically adapt to varying problem characteristics, increases the difficulty of parameter tuning, and negatively impacts robustness and search efficiency.

On the other hand, the designer's subjective preferences may affect the generalization ability of the formula, leading to inconsistent algorithm performance across different problems. Furthermore, according to the "No Free Lunch Theorem," no single optimal algorithm can be universally applied to all optimization problems [9]. Therefore, overcoming the limitations of manually designed update formulas is a critical challenge in evolutionary algorithm research. Enhancing their adaptability and generalization ability is essential for improving algorithm performance.

This manuscript proposes a Genetic Programming-based Rafflesia Optimization Algorithm (GP-ROA) to address these issues. As an evolutionary computation method, Genetic Programming (GP) can dynamically generate mathematical expressions using a tree-based structure, optimizing the evolution of search rules for complex problems [10, 11]. Meanwhile, the Rafflesia Optimization Algorithm (ROA) is an emerging metaheuristic algorithm that simulates the resource competition and adaptive behaviors of Rafflesia [12]. It demonstrates strong global search capabilities and effectively maintains population diversity. It has also shown good performance in fields such as engineering optimization [13], water pipeline network design [14], and multithreshold image segmentation [15]. However, GP may converge prematurely when evolving complex expressions, while ROA relies on manually designed and fixed position update formulas, which limits its adaptability. In ROA, the search behavior of individuals is completely determined by predefined update formulas. Once these formulas are fixed, the algorithm cannot adjust its search pattern according to different problem characteristics. In contrast, GP is particularly suitable for automatically generating and optimizing mathematical expressions.

Therefore, GP is introduced to evolve the position update formulas of ROA automatically, enabling the search strategy to be adjusted during the optimization process rather than remaining fixed.

In this manuscript, GP is utilized to evolve the position update formula of ROA, and its optimization performance is evaluated on the CEC2017 benchmark test set [16]. Furthermore, to verify its effectiveness in real-world problems, GP-ROA is extended to a binary version and applied to feature selection tasks. In high-dimensional data environments, Feature Selection(FS) aims to eliminate redundant or irrelevant features to reduce computational costs and improve classification performance [17]. Experimental results on UCI datasets show that the optimization formulas generated by GP can improve search efficiency compared to traditional methods. In certain cases, they also achieve superior feature selection results.

The main contributions of this manuscript are as follows:

1). An automatic GP-based optimization formula generation method for improving the search strategy of evolutionary algorithms is proposed and validated using GP-ROA as an example;

2). The optimization power of GP-ROA is evaluated on the CEC2017 benchmark test set and compared with classical and popular evolutionary algorithms;

3). Extend GP-ROA to binary form and apply it to feature selection and validate its effectiveness with experimental analysis on multiple UCI datasets.

This manuscript is organized as follows: section 2 introduces the basic techniques of ROA and GP in detail; Section 3 comprehensively describes the GP-ROA algorithm proposed in this manuscript and its newly derived position update formula; Section 4 evaluates the performance of the GPROA algorithm on the CEC2017 test set and analyzes it in comparison with other algorithms; section 5 applies the GPROA algorithm to feature selection, shows the experimental results and provides a detailed analysis; Section 6 summarizes the main points of this manuscript and discusses potential directions for future research.

2. **Related Work.**

2.1. **ROA.** The king flower is a decaying plant with an extremely short main axis, no leaves, and no underground stems. When it blooms, it emits a special odor that attracts insects for pollination, which are sometimes trapped and killed by its unique structure. After the flowers are gone, the fruit ripens and contains many tiny seeds. These seeds spread through various pathways in search of a suitable environment for germination. Based on these characteristics, the ROA can be realized in three stages. The process of attracting insects involves two strategies. The first strategy focuses on replacing poorly performing individuals with better ones. Under this strategy, 1/3 of the poorly adapted individuals are replaced by a new individual. To calculate the position of the new individual, The manuscript considers each of its dimensions as a point in a three-dimensional space, and the position model is shown in Fig.1. The formulas for updating the position of a single X_i are (1) to (3).

$$X_{i_k} = X_{\text{best }_k} + d \cdot \sin\beta_k \cos\gamma_k \tag{1}$$

$$d = \sqrt{\sum_{k=1}^{D} (X_{R_k} - X_{\text{best }_k})^2} \tag{2}$$

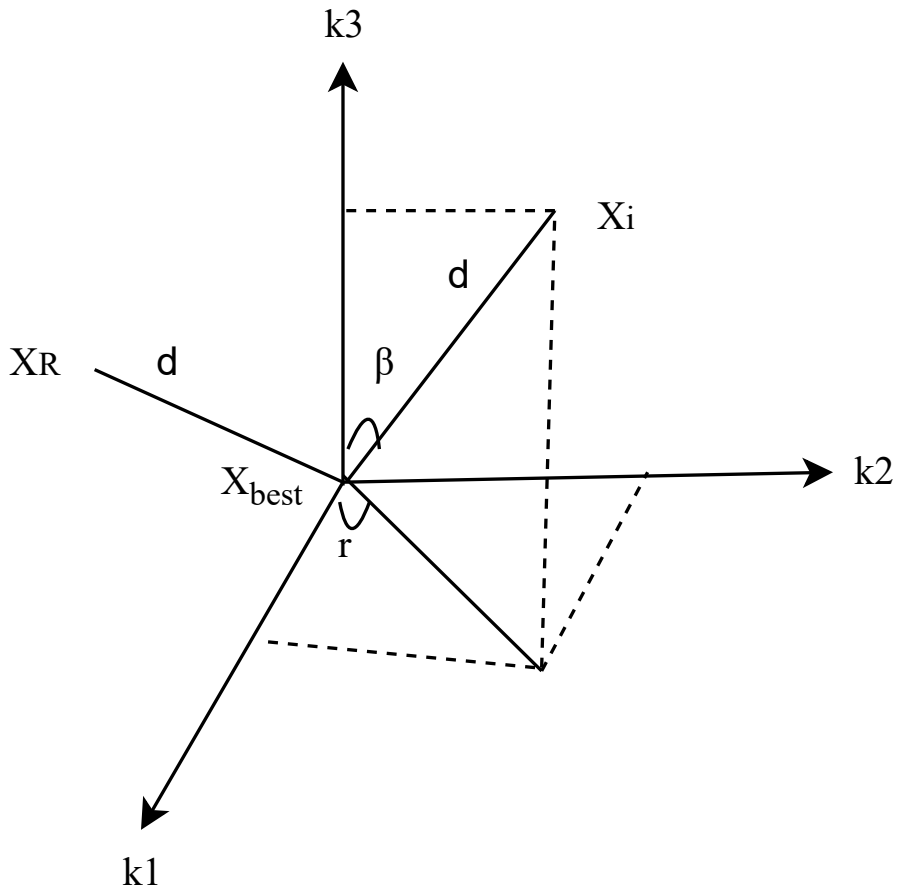


FIGURE 1. The model of the calculated dimensions

$$X_{\text{worst }_i} = X_i \tag{3}$$

where d is the distance between X_i and X_{best}. X_i is a random individual in the population. X_{best} represents the top-performing individual in the population, while X_{worst} denotes the lowest-performing one. β_k is a random value chosen from the range [0, $\pi/2$], and γ_k is a random value within the interval [0, π]. The second strategy involves updating 2/3 of the individuals to enhance their adaptation. The velocity update equations for individuals are based on the insect flapping wing flight model and are presented in equations (4)-(7).

$$\overrightarrow{v_1} = \frac{\omega_0}{2}\sqrt{A^2 \sin^2(\omega_0 t + \theta) + B^2 \cos^2(\omega_1 t + \theta)} \tag{4}$$

$$\overrightarrow{v_2} = \overrightarrow{v_2}\omega_0 \cos(\omega_0 t + \theta + \phi) \tag{5}$$

$$\vec{v} = \overrightarrow{v_1} + \overrightarrow{v_2} \tag{6}$$

$$X_{(t)} = X_{(t)} + C \cdot \vec{v} \cdot t + \left(X_{\text{best}} - X_{(t)}\right) \cdot (1 - C) \cdot \text{ rand} \tag{7}$$

v_1 and v_2 denote the translational and rotational velocities, respectively. ω_0 and ω_1 denote the frequency period of the flutter and side-flutter wings, respectively, and both have a value of 0.025. a is the amplitude of the wing during its motion and has a value of 2.5.

b is the transverse offset and has a value of 0.1. ϕ denotes the phase difference between the translational and rotational velocities and has a value of -0.78545. θ has a range of values (0,2 π). v_2 has an initial value of (0,2 π). C is a random number in (-1,1).

Based on the principle of "survival of the fittest," the individual with the lowest fitness is eliminated after several iterations during the insect swallowing stage. This process enhances solution quality and improves the algorithm's efficiency.

In the seed sowing stage, the Rafflesia's position is represented by the initial optimal individual, while other individuals randomly search for a suitable growth environment. The individual update equation in this stage is as follows:

$$X_{(t)_k} = X_{\text{best}_k} + rd \cdot \exp\left(\frac{\text{iter}}{\text{Max\_iter}} - 1\right) \cdot \text{sign}(\text{rand} - 0.5) \tag{8}$$

Here, $k(k = 1, \ldots, D)$ denotes the population dimension. The terms $iter$ and Max_iter represent the current and maximum number of iterations, respectively. The function $sign(rand - 0.5)$ outputs 1 or -1. rd indicates the range of individual values in the population, and its expression is given by:

$$rd = rand \cdot (ub - lb) + lb \tag{9}$$

ub represents the upper bound of the search space, while lb denotes the lower bound.

In summary, the ROA algorithm consists of three key stages: insect attraction, individual elimination, and seed dispersal. The core update mechanisms have been fully defined in Equations (1)-(9).

2.2. **GP.** Genetic Programming (GP) is an evolutionary computational method proposed by John R. Koza in the early 1990s, which is an expansion and extension of Genetic Algorithm (GA) [18]. The core idea of GP is to simulate the selection, crossover, and mutation processes in biological evolution. It aims to achieve the automatic generation of computer programs through random search. Additionally, it relies on optimization techniques to improve the generated programs. Its biggest advantage lies in its low demand for human intervention and its ability to solve the target problem autonomously. As a result, it has been widely used in Multi-intelligence reinforcement learning [19], feature extraction [20], and project scheduling problems [21].

In GP, each solution is represented as a tree whose nodes include operators in the set of functions and variables or constants in the set of terminals. Population initialization usually adopts a mixed strategy, where some individuals are generated by the depth-first method and some by the breadthfirst method to ensure the diversity of the population. Meanwhile, the depth of each tree needs to satisfy a preset limit to avoid overly complex structures.

The process of GP operation is usually divided into three main steps: selection, crossover, and mutation [22].

Selection:

The selection phase selects individuals from the population that need modification to provide the basis for the next population generation. Commonly used selection methods include random, roulette, and tournament selection. Roulette selection is more likely to fall into a local optimum, as it relies on probability proportional to fitness. In contrast, tournament selection allows for more flexible adjustment of selection pressure through small-scale competition.

Crossover:

The crossover operation is one of the central steps in GP, aiming at information sharing through genetic recombination. Specifically, crossover points are randomly selected among two-parent individuals, and their corresponding subtrees are exchanged to generate new

individuals. The crossover operation incorporates the advantages of the parents and potentially introduces more diversity. However, the size of the crossover subtree needs to be controlled to prevent the solution space from becoming too large, as shown in Fig.2.

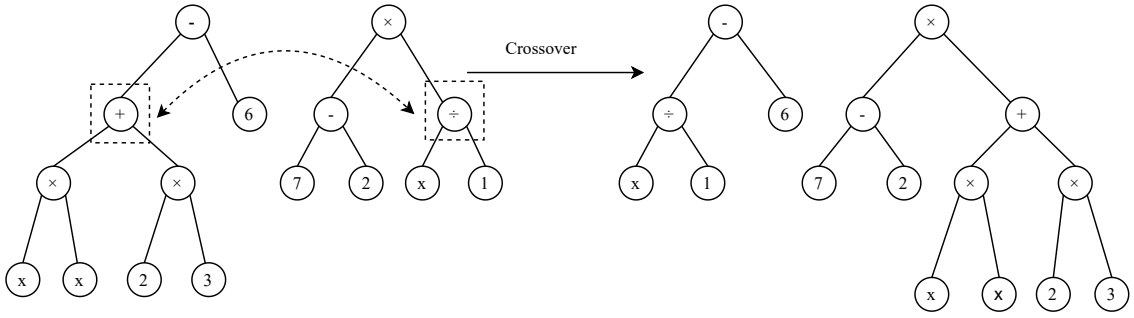


FIGURE 2. Example illustration of the crossover process

Mutation:

Mutation is a key genetic operation that introduces a new structure by randomly modifying a portion of an individual to increase population diversity and avoid falling into local optima. Mutation usually involves replacing a subtree in an expression tree, e.g., replacing the original subtree with a randomly generated new subtree that allows individuals to explore a new search space. Mutation can be classified into various types based on the modification. One type is subtree mutation, which selects a subtree and replaces it with a newly generated subtree, usually resulting in a significant change in the individual's behavior and improving global search. Another type is node mutation, which introduces a slight perturbation by randomly selecting and replacing a single node with another operator or terminal variable. Together, these mutation operations enable genetic programming to maintain high search efficiency and diversity of solutions in complex optimization problems, as shown in Fig.3. In summary, with its flexibility

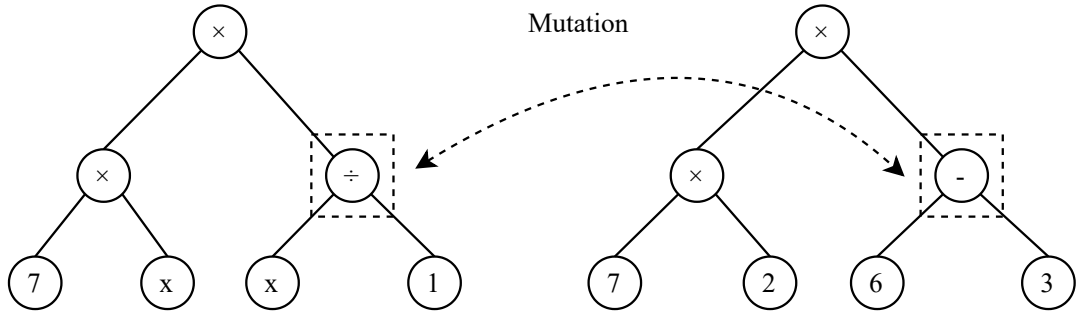


FIGURE 3. Example illustration of the mutation process

and scalability, GP provides a powerful tool for modeling and solving complex problems. Meanwhile, optimization studies on GP, such as stack-based GP, linear structure-based GP, and multi-objective GP, have further enhanced its performance. These advancements have also expanded its application scope.

3. **GP-ROA.** This chapter proposes a method based on Genetic Programming (GP) to optimize the Rafflesia Optimization Algorithm (ROA), i.e., GP-ROA.GP generates the positional updating formula of ROA through evolution to enhance the search capability

Table 1. Parameter Settings for GP-Based Formula Generation

Parameter	Setting
Size of the population	30
Method of initialization	half in depth and half in breadth
Maximum number of iterations	600
Upper and lower bounds of tree depth	3/5
Retention rate of high-quality samples	0.2
Crossover-to-mutation ratio	0.9/0.05

and adaptability of the algorithm. Unlike the traditional hand-designed update formulas, the formulas generated by GP can be dynamically adjusted according to the problem characteristics, making the optimization process more intelligent.

In GP-ROA, GP is responsible for evolving individual position update formulas to replace the original fixed formula of ROA (Equation (7). This method allows GP to explore more complex mathematical expressions and improve individuals' search efficiency. To verify the effectiveness of GP-ROA, we evaluate its performance using the CEC2017 benchmark test set in this chapter.

In the position update formula generated by GP, designing the terminal and function sets is crucial for Genetic Programming (GP). Based on the research of ROA, the terminal set and function set designed for ROA are shown below:

$$\begin{cases} \text{Terminal} : \{C, X_{\text{best}}, X_{\text{t}}, v, \text{rand}, 1\} \\ \text{Function} : \{+, -, \times, \div\} \end{cases}$$

The value of X_t denotes the position of the current individual in the population, while v denotes the speed of the current individual. The value of X_{best} denotes the position of the optimal individual in the population. *Rand* is a random number in (0, 1), C is a random number in (-1,1).

Through GP evolution, these elements will be combined to form new position update formulas to improve the adaptability of ROA in different optimization tasks.

Table 1 provides an overview of the parameters associated with the GP generation formula. With the GP generation formula, the depth of the individuals is set in the range of 3 to 5. At the beginning of the process, a random depth is selected for each individual within the specified limits. To improve efficiency in finding suitable formulas within a shorter time and to accelerate convergence, the algorithm directly retains the top 20% of individuals from the previous generation into the next generation. The remaining 80% of individuals are generated through crossover and mutation operations. To enhance population diversity, individuals selected for mutation undergo one of the following randomly chosen mutation methods: terminal node mutation, non-terminal node mutation, or growth mutation. The resulting Equation generated by GP is shown in Equation (10).

$$X_{(t)} = ((C + X_{\text{best}} \cdot (v + C))) \cdot \left(\frac{(C + \text{rand})}{(1 - \text{rand})} \right) + X_{(t)} \tag{10}$$

Unlike the original equation, the GP-generated formula dynamically incorporates an individual's position X_{best} and velocity v, enhancing the adaptiveness of velocity updates. In addition, the formulation contains a nonlinear scaling factor $(C + rand)/(1 - rand)$, which increases the perturbation magnitude and helps to improve the search capability and prevent premature convergence. At the same time, the learning constant C functions in multiple computational terms, guiding individuals based on the global optimal solution. It also allows individuals to adjust to their state, improving search diversity. In contrast,

Algorithm 1 GP-Generated ROA Algorithm

Input: $f(x)$; N; d; lb; ub; Max_iter; GP Parameters; Terminal set; Function set.
Output: Global optimal solution X_{gbest} and best GP-generated evolutionary formula.
Initialize ROA-related parameters.
Randomly initialize the positions of insect individuals.
Generate an initial GP population.
for $iter = 1$ **to** Max_iter **do**
    Compute the fitness of each insect.
    Sort individuals based on fitness and select the current best X_{best} (lowest fitness value).
    if $\text{rem}(iter, 600) < 300$ **then**
        for $i = 1$ **to** N **do**
            if $i \leq N/3$ **then**
                Update individuals using ROA Equations (1)–(3).
            else
                Select an evolutionary formula X_{new} from the GP population.
                if X_{new} has a lower fitness value than X_{best} **then**
                    Update X_{best}.
                    Store the best evolutionary formula.
                end if
            end if
        end for
    end if
    if $\text{rem}(iter, 600) = 300$ **then**
        Perform the **swallowing stage**, deleting the worst-performing individual.
    end if
    if $\text{rem}(iter, 600) > 300$ **then**
        for $i = 1$ **to** N **do**
            Compute random perturbation rd (using Equation (9)).
            Update the individual's position using Equation (8).
        end for
    end if
    Record the best fitness value.
end for
Return X_{gbest} and the best GP-generated evolutionary formula.

the original formulation adopts a fixed linear combination with a single search pattern. The GP formulation, on the other hand, improves adaptability and global optimization ability in complex problems through evolutionary optimization.

The pseudo-code of GP-ROA is shown in Algorithm 1:

4. Experimentation and Analysis of the CEC2017 Benchmark Test.

4.1. **Experimental Design.** The performance of GP-ROA on global optimization problems was evaluated using the CEC2017 benchmark test functions. The set of test functions covers a variety of complex optimization scenarios, including multi-peak function, single-peak function, and composite function, which can comprehensively evaluate the optimization capability of the algorithms. The problem dimensions are 30D and 50D, and the comparison algorithms include PSO [23], Sine Cosine Algorithm(SCA) [24], Grey Wolf Optimizer(GWO) [25], Butterfly Optimization Algorithm(BOA) [26] and ROA and

all the experiments are run under the same parameter settings. The population size of all algorithms involved in the experiments is set to 30, and 600 iterations are performed for each algorithm. Considering the effect of randomness, 10 independent runs were performed, and the results were averaged for comparison. Table 2 provides information about each algorithm and the parameters used.

TABLE 2. Parameter settings for each related algorithm

Algorithm	**Parameter**	**Setting**
PSO	Vmax	10
	c1,c2	2
	w	0.3
SCA	a	2
WOA	a	linearly decreases from2 to 0
	a2	linearly decreases from -1 to -2
	Spiral factor b	1
BOA	probabibility switch	0.6
	power exponent	0.1
	sensory modality	0.01
ROA	A	2.5
	f	40
	B	0.1
	φ	-0.78545

4.2. **Statistical Results.** Table 3 demonstrates the mean and standard deviation of GPROA versus the other five comparative algorithms for the 30-dimensional test conditions. In terms of mean performance, GP-ROA achieved better results than the other algorithms on 17 of the 30 test functions, showing its significant advantage in optimization capability. In addition, Table 5 provides these six algorithms' mean and standard deviation data for 10 runs under 50-dimensional test conditions. From the analysis of the mean results, GP-ROA achieved the smallest mean value among the 16 test functions. This result indicates that its optimization performance in higher dimensions remains strong and outperforms the other compared algorithms.

These data show that GP-ROA achieves a significant improvement in terms of optimization-seeking ability compared to the original ROA algorithm. This improvement is attributed to the optimization of GP-ROA's algorithmic structure. As a result,it becomes more efficient in navigating the search space and can approximate the global optimal solution more quickly and stably. Meanwhile, GP-ROA demonstrates greater adaptability and competitiveness compared to several other mainstream metaheuristic algorithms. It is also able to provide better solutions to complex optimization problems.

Overall, GP-ROA's performance in the CEC2017 benchmark test fully validates its advantages in solving highdimensional complex optimization problems. It significantly improves the optimality-finding capability. Additionally, it demonstrates excellent performance in terms of reliability and stability.

Although GP-ROA outperforms other algorithms in most test functions, it performs poorly in certain cases, such as f3 and f9. One possible reason is that the GP-generated update formula introduces additional randomness, which may not benefit functions with sharp minima or highly structured landscapes. Additionally, the increased complexity of the evolved formula could lead to slower convergence in some scenarios.

TABLE 3. The mean and standard deviation of each algorithm on the CEC2017 test function 30 dimensions

Function		PSO	SCA	GWO	BOA	ROA	GP-ROA
f1	adv	5.8613e+10	6.1563e+10	5.5952e+10	5.6807e+10	5.6028e+10	**5.5913e+10**
	std	1.2629e+09	1.0698e+09	7.6398e+07	6.4083e+08	2.8413e+08	3.7728e+06
f3	adv	**6.1182e+04**	7.5205e+04	9.2364e+04	7.3742e+04	6.1340e+04	6.5689e+04
	std	2.0031e+03	2.6965e+03	7.3196e+03	2.5848e+03	851.2229	2.1818e+03
f4	adv	2.0433e+04	2.1479e+04	1.9466e+04	1.9870e+04	1.9464e+04	**1.9415e+04**
	std	700.7208	314.9285	106.8213	221.6552	57.5069	7.7814
f5	adv	823.6780	900.3818	**803.5171**	830.8670	826.6032	831.0400
	std	19.0153	16.8607	30.6984	20.7403	16.7732	25.5675
f6	adv	666.9098	688.7066	**666.8944**	673.4806	674.2606	671.5596
	std	3.8502	6.0366	6.1037	3.3803	10.0106	4.8091
f7	adv	1.2369e+03	1.3390e+03	1.3147e+03	1.3420e+03	1.2687e+03	**1.2360e+03**
	std	38.3663	22.9136	41.2305	28.7245	53.2866	48.4255
f8	adv	1.0235e+03	1.1070e+03	1.0316e+03	1.0337e+03	1.0207e+03	**1.0096e+03**
	std	19.4174	18.2647	21.7258	20.3644	20.8074	21.6145
f9	adv	**5.4381e+03**	1.0586e+04	7.6856e+03	8.8410e+03	5.8786e+03	7.6606e+03
	std	270.5955	854.4512	889.6814	1.2676e+03	450.9296	1.4149e+03
f10	adv	6.1703e+03	7.8839e+03	**5.5024e+03**	5.9877e+03	6.3942e+03	6.1344e+03
	std	296.3532	342.0533	535.7652	643.3580	900.0843	451.5844
f11	adv	1.8278e+07	1.8278e+07	1.8277e+07	1.8278e+07	1.8277e+07	**1.8277e+07**
	std	514.6487	323.7334	98.9618	733.9448	76.9601	12.4580
f12	adv	1.8698e+10	1.9768e+10	1.8509e+10	1.8733e+10	1.8508e+10	**1.8484e+10**
	std	2.5263e+08	3.0923e+08	7.1675e+07	1.7131e+08	4.5487e+07	4.1568e+05
f13	adv	2.8636e+10	2.8720e+10	2.8636e+10	2.8636e+10	2.8636e+10	**2.8636e+10**
	std	2.1222e+03	9.3101e+07	300.1942	3.8296e+03	1.6142e+03	103.9557
f14	adv	2.8593e+08	2.8593e+08	2.8593e+08	2.8593e+08	**2.8593e+08**	2.8593e+08
	std	22.6553	13.5946	17.4428	11.9249	14.6795	14.6664
f15	adv	2.9709e+09	2.9708e+09	2.9696e+09	2.9696e+09	**2.9696e+09**	2.9696e+09
	std	2.8661e+06	1.7503e+06	2.1848e+03	1.0465e+04	2.8595e+03	3.4448e+03
f16	adv	**1.5101e+04**	1.6133e+04	1.5116e+04	1.5591e+04	1.5178e+04	1.5221e+04
	std	226.3241	302.9799	323.8113	329.8678	328.9471	365.7376
f17	adv	**5.6995e+04**	5.7280e+04	5.7162e+04	5.7371e+04	5.7175e+04	5.6996e+04
	std	241.6408	155.1574	392.0526	294.2413	305.6322	168.9302
f18	adv	1.3711e+09	1.3893e+09	1.3727e+09	1.3874e+09	1.3720e+09	**1.3707e+09**
	std	1.9470e+06	1.2245e+07	3.4349e+06	1.1563e+07	2.0130e+06	8.0377e+05
f19	adv	3.1217e+09	3.1228e+09	3.1228e+09	3.1232e+09	3.1212e+09	**3.1211e+09**
	std	2.5429e+06	7.2252e+05	9.8466e+05	1.2598e+06	2.9588e+05	3.4417e+05
f20	adv	3.3478e+03	3.3635e+03	3.4109e+03	3.3885e+03	3.3856e+03	**3.3263e+03**
	std	89.5929	15.9486	100.3447	78.2122	98.3696	65.9431
f21	adv	2.8304e+03	2.8686e+03	**2.7840e+03**	2.8187e+03	2.8028e+03	2.8009e+03
	std	25.0614	14.4004	25.1713	23.9600	34.6272	19.0677
f22	adv	8.3537e+03	9.3628e+03	8.8652e+03	8.4555e+03	8.4273e+03	**8.3381e+03**
	std	203.1595	286.8143	425.4134	456.8509	401.5261	245.2027

Future work could explore adaptive GP constraints to more effectively balance exploration and exploitation. This approach would be especially useful for problems where maintaining structural information is crucial.

TABLE 4. Table 3 (continued)

Function		PSO	SCA	GWO	BOA	ROA	GP-ROA
f23	adv	4.9812e+03	5.7581e+03	**4.9543e+03**	5.1566e+03	5.2136e+03	5.1889e+03
	std	275.2749	113.8120	226.8182	201.8920	245.8657	254.5374
f24	adv	4.7686e+03	4.8179e+03	4.6952e+03	4.7449e+03	**4.6600e+03**	4.6695e+03
	std	41.7189	17.8951	38.7376	54.8537	18.3181	23.3132
f25	adv	5.3689e+03	5.6703e+03	5.1352e+03	5.2819e+03	5.1095e+03	**5.0959e+03**
	std	134.7647	89.2992	48.6920	99.2599	21.6686	0.5018
f26	adv	1.1953e+04	1.2404e+04	1.1982e+04	1.1808e+04	1.1658e+04	**1.1544e+04**
	std	268.6301	149.8357	204.9464	128.6254	104.3689	125.3570
f27	adv	6.8126e+03	7.6180e+03	7.4575e+03	7.1622e+03	7.2259e+03	**6.7572e+03**
	std	281.7018	164.0198	270.1017	246.3120	304.2252	308.4007
f28	adv	7.5252e+03	7.8400e+03	7.2225e+03	7.2879e+03	7.2086e+03	**7.1932e+03**
	std	106.8322	80.8229	27.9510	38.2323	25.8411	6.5390e-04
f29	adv	3.5988e+04	3.7848e+04	3.4013e+04	3.4895e+04	3.2525e+04	**3.2358e+04**
	std	2.0277e+03	973.1293	962.3694	844.4327	642.3407	788.8754
f30	adv	6.0339e+09	6.2177e+09	5.7074e+09	5.8317e+09	5.6854e+09	**5.6529e+09**
	std	1.5147e+08	8.8531e+07	8.5070e+07	4.6142e+07	2.8744e+07	4.8731e+05
win		21	29	24	25	23	

TABLE 5. The mean and standard deviation of each algorithm on the CEC2017 test function 50 dimensions

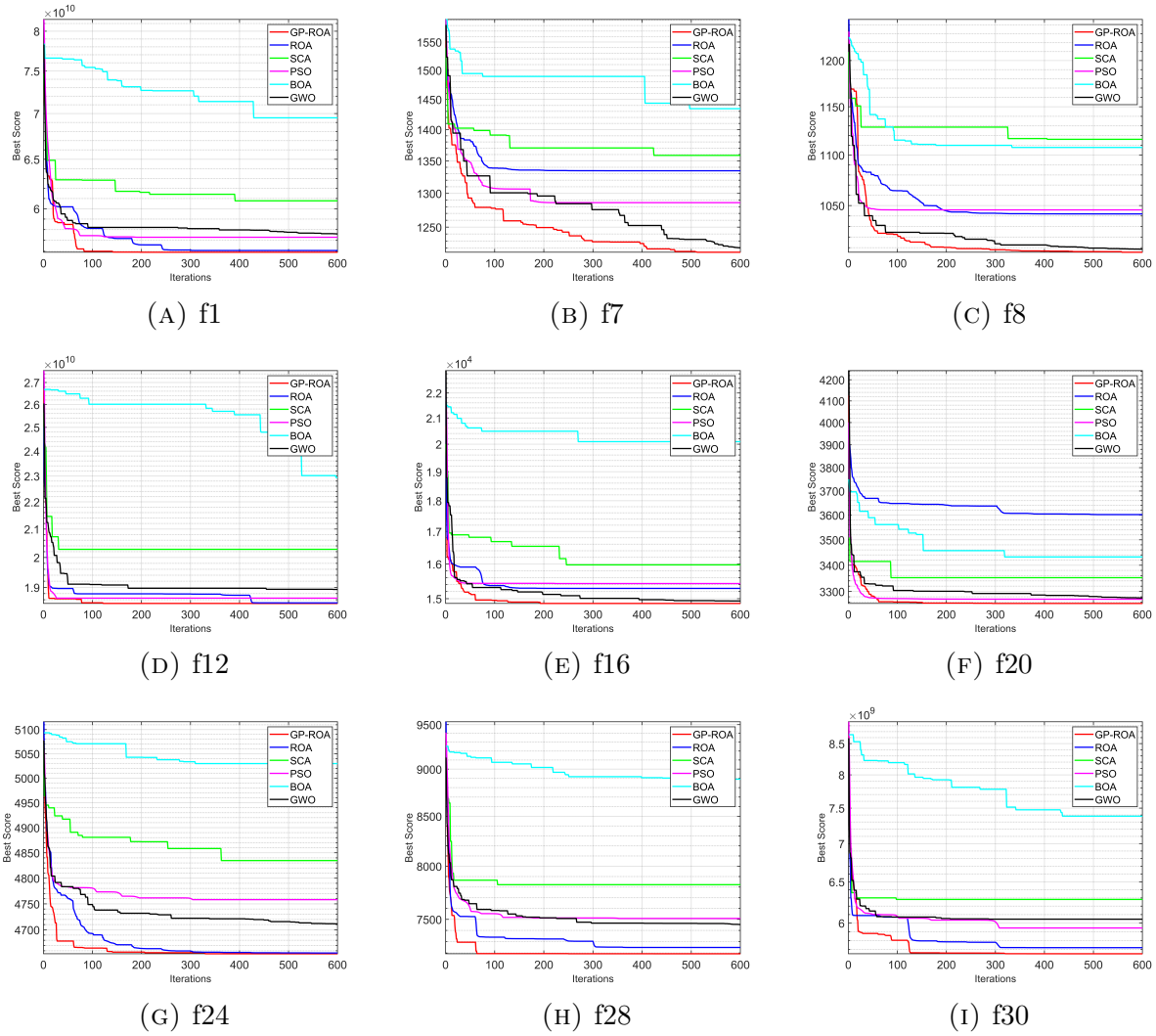
Function		PSO	SCA	GWO	BOA	ROA	GP-ROA
f1	adv	9.8879e+10	1.0437e+11	8.7093e+10	9.0427e+10	8.6781e+10	**8.6296e+10**
	std	2.3954e+09	1.5104e+09	9.1381e+08	7.2598e+08	5.9200e+08	7.3074e+07
f3	adv	1.4307e+05	1.8399e+05	1.7536e+05	1.8660e+05	**1.4295e+05**	1.4948e+05
	std	5.7887e+03	1.1244e+04	7.5754e+03	1.1215e+04	2.9115e+03	5.7410e+03
f4	adv	3.5453e+04	3.7846e+04	3.1990e+04	3.2966e+04	3.1771e+04	**3.1388e+04**
	std	958.6003	748.6903	286.4387	419.1781	453.8050	111.1920
f5	adv	**914.8757**	1.1409e+03	932.6246	974.1920	928.3570	955.0585
	std	26.5727	21.3618	24.2556	32.7617	26.3926	31.1430
f6	adv	**669.4468**	699.9238	671.5099	684.9015	679.1084	677.7425
	std	2.5437	3.4650	5.8673	5.8413	6.1330	5.2398
f7	adv	1.7043e+03	1.8757e+03	1.8654e+03	1.9335e+03	1.7547e+03	**1.6917e+03**
	std	32.5590	22.8602	51.7154	54.3239	57.3510	60.0635
f8	adv	**1.2704e+03**	1.4672e+03	1.2840e+03	1.3354e+03	1.2883e+03	1.3269e+03
	std	21.2486	27.7701	27.4950	17.7619	19.7144	63.4742
f9	adv	**1.4409e+04**	3.6468e+04	1.8940e+04	3.1927e+04	1.5464e+04	2.6293e+04
	std	897.7735	2.6011e+03	2.1977e+03	1.9735e+03	1.9196e+03	3.3209e+03
f10	adv	**9.6009e+03**	1.4027e+04	1.1881e+04	1.0674e+04	1.0443e+04	1.0216e+04
	std	617.4744	313.0143	994.9414	455.0823	595.8999	930.9808
f11	adv	1.8795e+04	2.1174e+04	1.9075e+04	1.9045e+04	**1.8323e+04**	1.8510e+04
	std	804.6566	394.0034	477.4467	408.3526	44.4140	181.7796
f12	adv	9.9722e+10	1.0464e+11	9.3752e+10	9.5739e+10	9.4014e+10	**9.3346e+10**
	std	2.6997e+09	1.5829e+09	5.2372e+08	8.2999e+08	7.1553e+08	2.0777e+07

4.3. **Convergence Analysis.** In addition, this manuscript provides a detailed analysis of GP-ROA in terms of convergence performance. It evaluates the algorithm in 30- and

TABLE 6. Table 5 (continued)

Function		PSO	SCA	GWO	BOA	ROA	GP-ROA
f13	adv	7.8681e+10	8.0308e+10	7.8576e+10	7.8715e+10	7.8576e+10	**7.8576e+10**
	std	2.4496e+08	5.6163e+08	7.2366e+03	1.2875e+08	1.9885e+05	8.8777e+03
f14	adv	5.9518e+08	5.9915e+08	**5.9199e+08**	5.9224e+08	5.9199e+08	5.9199e+08
	std	6.8137e+06	4.2144e+06	49.8139	7.7114e+05	31.1858	37.5011
f15	adv	1.3734e+10	1.3825e+10	**1.3734e+10**	1.3736e+10	1.3734e+10	1.3734e+10
	std	1.7407e+03	1.0372e+08	1.3982e+03	5.2768e+06	3.2421e+03	2.0638e+03
f16	adv	1.2587e+04	1.4698e+04	1.2761e+04	1.2617e+04	1.2493e+04	**1.2380e+04**
	std	308.1008	402.4916	569.2414	561.7313	508.8584	269.5650
f17	adv	1.0385e+04	1.2006e+04	1.0425e+04	4.2720e+04	1.0284e+04	1.0162e+04
	std	857.0690	556.7555	470.0053	355.4603	419.3220	359.6343
f18	adv	3.7464e+08	3.8003e+08	3.4891e+08	3.5436e+08	**3.4808e+08**	3.4808e+08
	std	2.8091e+07	1.4528e+07	1.8999e+06	8.9332e+06	2.1979e+03	8.5958e+03
f19	adv	8.1643e+09	8.2028e+09	8.1642e+09	8.1644e+09	8.1644e+09	**8.1641e+09**
	std	5.8199e+05	2.5011e+07	3.7365e+05	5.5539e+05	2.8811e+05	1.1649e+05
f20	adv	3.8159e+03	3.8895e+03	4.1033e+03	3.8760e+03	3.7809e+03	**3.7213e+03**
	std	189.0315	179.3187	346.4464	324.9531	258.8873	252.9844
f21	adv	3.4346e+03	3.6261e+03	3.4470e+03	3.4283e+03	3.3690e+03	**3.3550e+03**
	std	39.4247	18.8088	54.0204	37.6977	75.6613	49.1541
f22	adv	**1.3233e+04**	1.6865e+04	1.3411e+04	1.4464e+04	1.4063e+04	1.4284e+04
	std	471.6573	680.9610	880.7762	447.7342	909.1677	582.6394
f23	adv	5.6455e+03	6.9715e+03	**5.5704e+03**	6.0310e+03	5.7193e+03	5.9982e+03
	std	314.9463	154.0551	374.9592	383.6129	169.9682	265.8626
f24	adv	6.1367e+03	6.3074e+03	6.0265e+03	6.6374e+03	5.9211e+03	**5.8876e+03**
	std	60.7887	23.0423	68.8808	61.6568	37.3037	54.7487
f25	adv	1.3215e+04	1.4204e+04	1.1816e+04	1.2178e+04	1.1693e+04	**1.1603e+04**
	std	414.5204	263.0342	149.0675	188.0484	152.0673	10.8845
f26	adv	1.5745e+04	1.6506e+04	1.5520e+04	1.5395e+04	1.4979e+04	**1.5039e+04**
	std	236.1051	163.3587	551.5178	385.7774	227.2274	199.9157
f27	adv	1.2158e+04	1.4073e+04	1.2992e+04	1.2628e+04	1.2135e+04	**1.2019e+04**
	std	407.8386	308.7395	669.2688	433.1520	535.2649	528.8172
f28	adv	1.3539e+04	1.4240e+04	1.2402e+04	1.2897e+04	1.2398e+04	**1.2241e+04**
	std	260.1482	196.0637	148.9098	276.3332	120.1326	14.7402
f29	adv	1.2665e+06	1.3564e+06	1.1815e+06	1.2558e+06	1.1565e+06	**1.1280e+06**
	std	4.3984e+04	1.3877e+04	2.9316e+04	2.3187e+04	2.0576e+04	7.1568e+03
f30	adv	1.4553e+10	1.5090e+10	1.3376e+10	1.3962e+10	1.3524e+10	**1.3143e+10**
	std	2.1355e+08	1.4257e+08	1.4477e+08	2.1062e+08	1.1721e+08	9.3755e+06
win		19	29	20	28	21	

50-dimensional conditions on four problems selected from the CEC2017 test function set. Fig. 4 illustrates the convergence performance of the algorithm in the 30-dimensional case, while Fig. 5 shows its performance in the 50-dimensional condition. In the figure, the convergence curve of GP-ROA is highlighted in red color to evaluate its performance more clearly. As can be observed from Fig. 4 and Fig. 5, GP-ROA exhibits fast convergence in the initial phase of the iteration and can advance steadily towards the global optimum. These observations indicate that the algorithm possesses efficient navigation capabilities in the search space and can reliably converge to the global optimum solution. In addition, the convergence curves of the other algorithms are also shown in the figure.



(A) f1 (B) f7 (C) f8

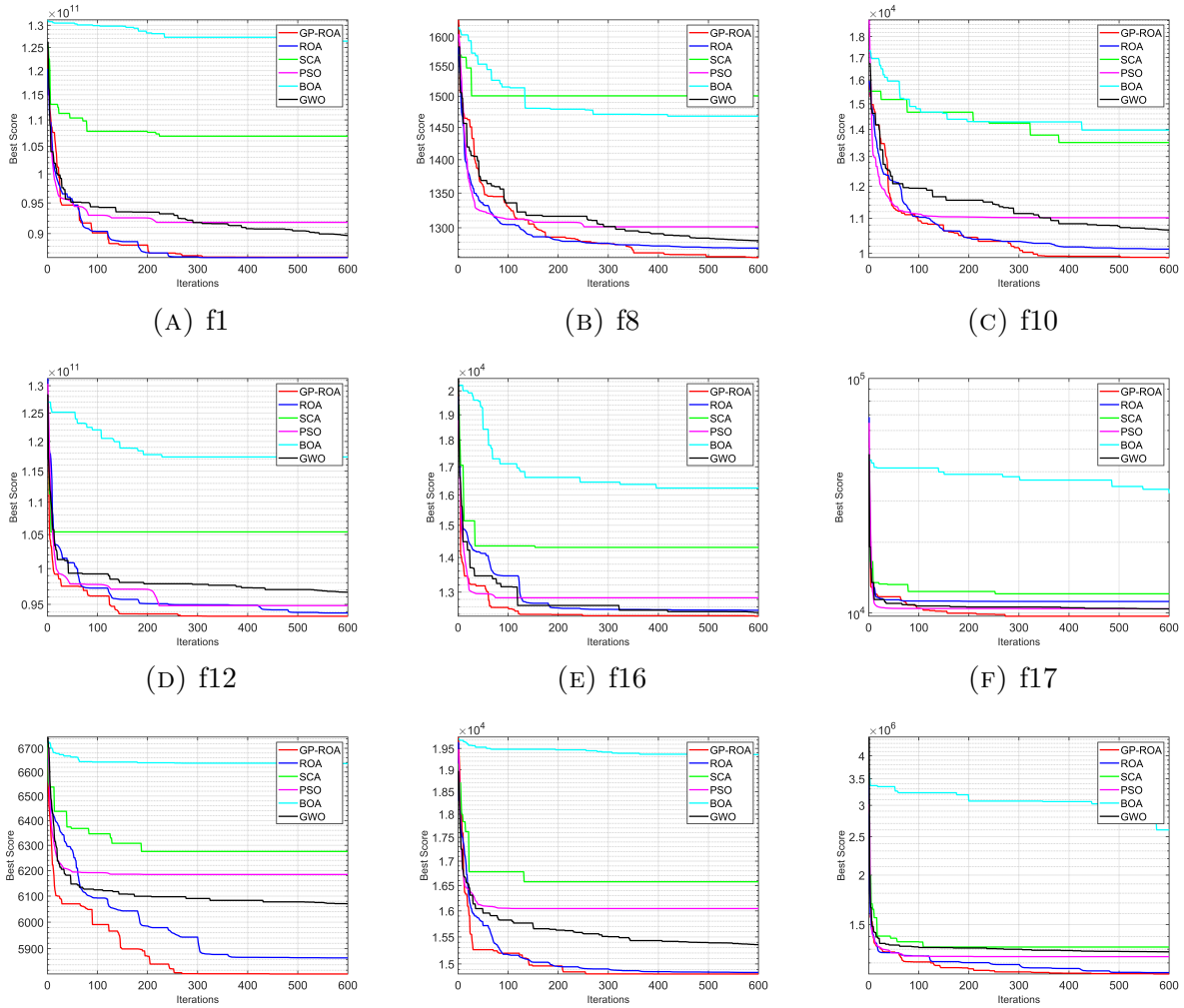
(D) f12 (E) f16 (F) f20

(G) f24 (H) f28 (I) f30

FIGURE 4. Convergence curves of GP-ROA and comparative algorithms on 30D CEC2017 benchmark functions.

The comparison provides a comprehensive understanding of the differences in convergence behavior and relative performance of the algorithms. Overall, the convergence analysis on the CEC2017 benchmark function set fully demonstrates GP-ROA's excellent ability to achieve fast convergence and approximation of the optimal solution. These findings provide strong support for its application in complex optimization problems.

4.4. **Statistical Analysis.** In this section, the results of the data provided in the previous paragraph are statistically analyzed to compare the performance of GP-ROA with other competing algorithms objectively. The analysis results are shown in Table 7. First, the Friedman test calculates the average ranking of each algorithm on all test functions. The p-value of this test is used to measure the reliability of the statistical results, and the smaller the p-value, the more statistically significant the results are. The p-values are 1.6339e-14 and 7.3859e-17 for the two dimensions, which are close to zero, so the statistical results can be considered to have a high degree of reliability. The average rankings of GP-ROA in the 30D and 50D dimensions are 1.8276 and 1.9310, respectively, which rank first. These rankings indicate that GP-ROA performs better when solving unknown problems. This strong performance suggests that GP-ROA has a stronger overall capability in solving unknown problems. In addition, the Wilcoxon rank sum test was



(A) f1 (B) f8 (C) f10

(D) f12 (E) f16 (F) f17

(G) f24 (H) f26 (I) f29

FIGURE 5. Convergence curves of GP-ROA and comparative algorithms on 50D CEC2017 benchmark functions.

TABLE 7. Statistical results for each competing algorithm

	Algorithm	GP-ROA	ROA	PSO	SCA	GWO	BOA
30D	Friedman	**1.8276**	2.7241	3.2414	5.6552	3.2069	4.3498
				p=1.6339e-14			
	Wilcoxon(+/=/-)		22/0/7	21/0/8	29/0/0	23/0/6	26/0/3
50D	Friedman	**1.9310**	2.4138	3.2759	5.8621	3.1034	4.4138
				p=7.3859e-17			
	Wilcoxon(+/=/-)		20/0/9	20/0/9	29/0/0	20/0/9	29/0/0

used with the significance level set at 0.05. The symbols indicate the results: "+" means GP-ROA significantly outperforms the competitive algorithms, "=" indicates no significant difference, and "-" shows that GP-ROA significantly underperforms the competitive algorithms. Where "+" means that GP-ROA is considerably better than the competitive algorithm, "=" means that there is no significant difference between GP-ROA and the competitive algorithm, and "-" indicates that GP-ROA is significantly lower than the

TABLE 8. Parameter settings for each related algorithm

Dataset	**Instances**	**Number of Features**	**Number of Categories**	**Attribute Types**
Breast Cancer Wisconsin	569	30	2	Real
Abalone	4177	8	3	Real, Integer, Categorical
Glass Identification	214	9	7	Real
Wine	178	13	3	Real, Integer
Raisin	900	7	2	Real, Integer
Ionosphere	351	34	2	Real, Integer
Connectionist Bench	208	60	2	Real
Image Segmentation	2310	19	7	Real
MAGIC Gamma Telescope	19020	10	2	Real
Wall-Following Robot Navigation Data	5456	24	4	Real
Yeast	1484	8	10	Real
Zoo	101	16	7	Categorical, Integer

competitive algorithm. According to the statistical test data in the table, the GP-CSO algorithm improved by GP shows superior performance compared to the original algorithm in the 30D dimension. It achieves advantages in 22 test functions, further validating the algorithm's effectiveness. In addition, GP-ROA performs better on most test functions than other competing algorithms. GP-ROA also outperforms the original ROA algorithm on most benchmark functions at 50D.

5. **Application of Feature Selection.** Due to the presence of a large amount of redundant information in the original data, data processing becomes complex and time-consuming [27]. Feature selection serves as a crucial preprocessing step to eliminate irrelevant features, making the data more compact and informative, thus improving overall processing efficiency [28].In this study, an improved algorithm, termed BGP-ROA, which is the binary version of the previously proposed Genetic Programming-based ROA, is applied to perform feature selection on several benchmark datasets. By adaptively evolving position update formulas according to the data characteristics, BGP-ROA enhances both search efficiency and solution diversity.

5.1. **Experimental Setup and Dataset Description.** This study utilizes twelve benchmark datasets from the UCI Machine Learning Repository, covering a range of sample sizes and feature dimensions to comprehensively evaluate the performance of the proposed

feature selection algorithm [29]. Feature selection is conducted using a binary representation, where "1" indicates that a feature is selected and "0" indicates it is discarded. All algorithms adopt the Sigmoid function [30] to map continuous values into the binary space, with a threshold of 0.5 used to determine feature selection. This transformation method is simple and efficient, contributing to improved algorithm stability and convergence speed. Detailed information about the datasets is provided in Table 8.

5.2. **Evaluation Algorithms and Parameterization.** The efficiency of the proposed algorithm was evaluated by comparing it with five popular feature selection algorithms: Binary Particle Swarm Optimization(BPSO) [31, 32], Binary Grey Wolf Optimization(BGWO) [33], Binary Sine cosine algorithm(BSCA) [34], Binary Butterfly Optimization Algorithm(BBOA) [35], and Binary Remora Optimization Algorithm(BROA) [36]. All tests are run under the same settings, with the population size set to 30 and the maximum number of iterations to 600. A K-NN classifier is used to evaluate the feature subset during the optimization process [37]. The experimental parameters were set based on relevant literature, and K was 5 to ensure the best results [38]. The fitness function combines the classification error rate and the number of features. The classification error rate is calculated using the leave-out method, where 80% of the data is used for training, and the remaining 20% is used for evaluation. The experiment was conducted with twenty independent tests, and the results were averaged to minimize the effect of randomness. The number of features in each dataset is the problem dimension, and Table 2 lists the algorithms and their parameters. The formula for the fitness function in this manuscript is as follows:

$$\text{fitness} = \alpha \cdot \text{error} + \beta \frac{\text{Fs}}{\text{Fa}} \tag{11}$$

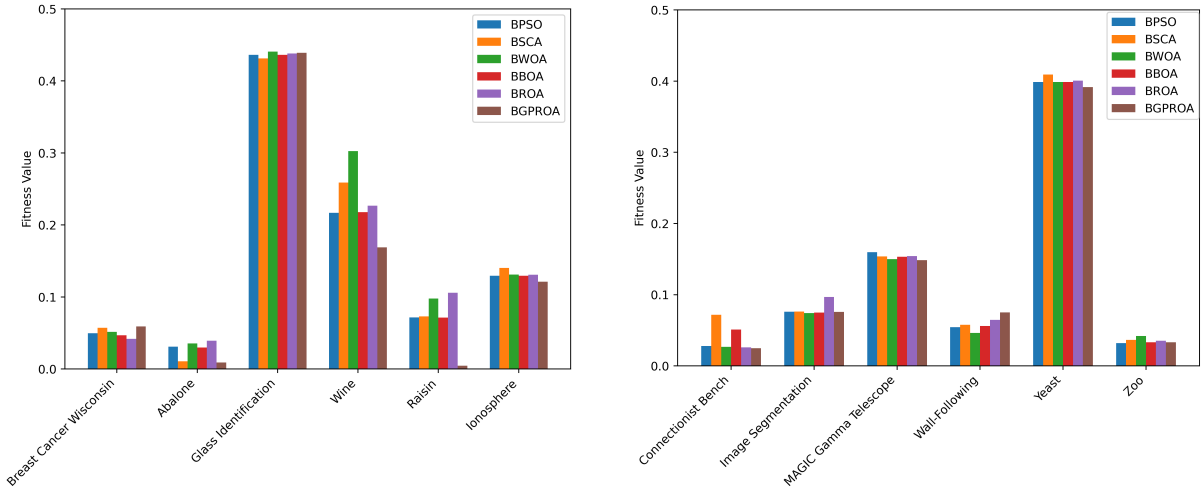
In this research, the variable error denotes the classification error. In addition, α and β are parameters that determine the feature and classification weights, respectively. α takes values ranging from 0 to 1, and β can be estimated as $(1 - \alpha)$. Fs denotes the number of features selected, and Fa denotes the total number of features in the dataset. Since classifier performance is considered the most important metric in this research, α is 0.99, and β is 0.01.

5.3. **Experimental Results and Analysis.** The experimental results show that BGP-ROA generates formulas for ROA through the tree structure of GP. It achieves high classification accuracy on several datasets and performs best on some datasets (Table 9). For example, in the Breast Cancer Wisconsin, Wine, and Raisin datasets, BGP-ROA outperforms other algorithms in accuracy, demonstrating its effectiveness in feature selection tasks. In addition, on the Abalone datasets, BGP-ROA achieves near-optimal classification results. These results indicate that the algorithm has a strong adaptive ability under different data distributions.

Meanwhile, BGP-ROA also performs well in reducing the number of features (Table 10). BGP-ROA selects fewer features in several datasets than most of the compared algorithms. For example, the Breast Cancer Wisconsin, Glass Identification, and Connectionist Bench datasets significantly reduce the feature dimensions and computational overhead. In addition, Fig. 6 shows that BGP-ROA also performs better in fitness value, which can achieve better feature selection results in the optimization process. From a comprehensive perspective, BGP-ROA optimizes ROA through the GP structure, effectively reducing the number of features while ensuring classification accuracy. It achieves efficient and accurate feature selection, outperforming other methods.

TABLE 9. The result of accuracy value

Dataset	BPSO	BSCA	BGWO	BBOA	BROA	BGP-ROA
Breast Cancer Wisconsin	0.943859	0.937719	0.965789	0.943859	0.949122	**0.967982**
Abalone	**0.562201**	0.511423	0.525956	0.526495	0.550358	0.554306
Glass Identification	0.658139	0.633720	0.729069	0.661627	0.754651	**0.761627**
Wine	0.829762	0.839285	0.902381	0.841661	0.871428	**0.946428**
Raisin	0.842500	0.815555	0.861666	0.846944	0.863055	**0.864166**
Ionosphere	0.905633	0.893661	0.905633	0.898591	**0.931690**	0.900000
Connectionist Bench	0.880952	0.765476	0.896428	0.828571	0.911904	**0.923809**
Image Segmentation	0.800000	0.839285	0.891666	0.834523	0.860714	**0.892857**
MAGIC Gamma Telescope	0.790036	0.795202	0.848606	0.779153	0.839589	**0.868375**
Wall-Following Robot Navigation Data	**0.938003**	0.888003	0.862821	0.866529	0.912545	0.897985
Yeast	0.593602	0.434680	0.515488	0.457575	0.603030	**0.684848**
Zoo	0.980952	0.804761	0.923809	0.876190	0.952381	**0.985714**



(A) (B)

FIGURE 6. The result of fitness value.

TABLE 10. The number of selected features

Dataset	BPSO	BSCA	BGWO	BBOA	BROA	BGP-ROA
Breast Cancer Wisconsin	10.65	14.9	11.75	11.55	12.1	**9.9**
Abalone	6.0	**5.2**	6.3	6.5	6.1	6.3
Glass Identification	6.6	4.7	7.1	6.4	5.05	**4.25**
Wine	5.65	6.6	**4.18**	5.6	5.95	6.0
Raisin	2.1	2.2	3.1	2.45	2.6	**2.0**
Ionosphere	**9.8**	13.55	13.6	10.05	11.95	11.1
Connectionist Bench	25.45	29.2	27.75	25.75	26.1	**25.0**
Image Segmentation	5.35	8.3	6.45	5.55	6.75	**5.05**
MAGIC Gamma Telescope	**5.0**	6.0	6.15	6.8	5.6	5.3
Wall-Following Robot Navigation Data	**6.8**	13.7	14.7	12.5	9.5	10.5
Yeast	7.1	6.3	5.6	7.6	5.7	**5.0**
Zoo	**5.1**	9.1	10.8	9.8	6.2	5.3

6. **Conclusions.** This manuscript proposes a novel approach that leverages Genetic Programming(GP) to automatically generate adaptive optimization strategies, applied specifically to the feature selection problem. Traditional feature selection algorithms often rely on manually designed position update formulas, which are constrained by fixed structures and heuristic assumptions. This makes it difficult for them to adapt to diverse and complex optimization scenarios. To address these limitations, this work employs GP to dynamically evolve the position update formulas, thereby enhancing the adaptability and generalization capability of the optimization process.

Extensive experiments on multiple UCI datasets demonstrate that the GPgenerated optimization strategies effectively reduce redundant features while improving classification accuracy. These strategies outperform conventional evolutionary algorithms and several mainstream methods. Moreover, the proposed approach achieves better optimization performance with reduced computational overhead, resulting in increased data processing efficiency and practical applicability.

Future research directions include: (1) developing adaptive evolution strategies by incorporating dynamic parameter tuning mechanisms to better accommodate varying types and scales of optimization problems; (2) leveraging parallel computing technologies, such as distributed or cloud-based resources, to accelerate the GP evolution process and scale

to large optimization tasks; and (3) integrating hybrid optimization frameworks by combining GP with deep learning or reinforcement learning techniques to enhance search efficiency and avoid local optima. These directions are expected to further improve the effectiveness and applicability of GP-based feature selection methods in complex real-world scenarios.

REFERENCES

[1] E. Aličković and A. Subasi, Breast cancer diagnosis using GA feature selection and rotation forest, *Neural Computing and Applications*, vol. 28, no. 4, pp. 753–763, 2017.

[2] Y. Liu, G. Wang, H. Chen, H. Dong, X. Zhu, and S. Wang, An improved particle swarm optimization for feature selection, *Journal of Bionic Engineering*, vol. 8, no. 2, pp. 191–200, 2011.

[3] T. Vivekanandan and N. C. S. N. Iyengar, Optimal feature selection using a modified differential evolution algorithm and its effectiveness for prediction of heart disease, *Computers in Biology and Medicine*, vol. 90, pp. 125–136, 2017.

[4] K. Chen, F.-Y. Zhou, and X.-F. Yuan, Hybrid particle swarm optimization with spiral-shaped mechanism for feature selection, *Expert Systems with Applications*, vol. 128, pp. 140–156, 2019.

[5] M. Mafarja and S. Mirjalili, Whale optimization approaches for wrapper feature selection, *Applied Soft Computing*, vol. 62, pp. 441–453, 2018.

[6] N. A. El-Hefnawy, O. A. Raouf, and H. Askr, Dynamic routing optimization algorithm for software defined networking, *Computers, Materials & Continua*, vol. 70, no. 1, pp. 69–89, 2022.

[7] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[8] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, Metaheuristic algorithms: A comprehensive review, *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, pp. 185–231, 2018.

[9] T. Joyce and J. M. Herrmann, A review of no free lunch theorems and their implications for metaheuristic optimisation, *Nature-Inspired Algorithms and Applied Optimization*, pp. 27–51, 2017.

[10] P. Bartashevich, I. Bakurov, S. Mostaghim, and L. Vanneschi, PSO-based search rules for aerial swarms against unexplored vector fields via genetic programming, *Proc. of the International Conference on Parallel Problem Solving from Nature*, Springer, pp. 41–53, 2018.

[11] C. Di Chio and P. Di Chio, Group-foraging with particle swarms and genetic programming, *Proc. of the European Conference on Genetic Programming*, Springer, pp. 331–340, 2007.

[12] J.-S. Pan, Z. Fu, C.-C. Hu, P.-W. Tsai, and S.-C. Chu, Rafflesia optimization algorithm applied in the logistics distribution centers location problem, *Journal of Internet Technology*, vol. 23, no. 7, pp. 1541–1555, 2022.

[13] J.-S. Pan, Z. Zhang, S.-C. Chu, Z.-J. Lee, and W. Li, Application of diversity-maintaining adaptive rafflesia optimization algorithm to engineering optimisation problems, *Symmetry*, vol. 15, no. 11, p. 2077, 2023.

[14] Y.-C. Huang, Q. Yang, Y.-C. Huang, and J.-S. Pan, Optimization of water distribution network design using rafflesia optimization algorithm based on opposition-based learning, *Journal of Internet Technology*, vol. 24, no. 5, pp. 1079–1087, 2023.

[15] J.-L. Zhou, S.-C. Chu, S. Weng, and W.-M. Zheng, An efficient multilevel threshold image segmentation approach based on rafflesia optimization algorithm, *Proc. of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Springer, pp. 1–13, 2022.

[16] G. Wu, R. Mallipeddi, and P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization, Technical Report, National University of Defense Technology, Changsha, China; Kyungpook National University, Daegu, South Korea; Nanyang Technological University, Singapore, 2017.

[17] A. A. Soofi and A. Awan, Classification techniques in machine learning: applications and issues, *Journal of Basic & Applied Sciences*, vol. 13, pp. 459–465, 2017.

[18] K. E. Kinnear, P. J. Angeline, and L. Spector, *Advances in Genetic Programming*, vol. 3, MIT Press, 1994.

[19] T. Wang, X. Peng, T. Wang, T. Liu, and D. Xu, Automated design of action advising trigger conditions for multiagent reinforcement learning: a genetic programming-based approach, *Swarm and Evolutionary Computation*, vol. 85, p. 101475, 2024.

[20] I. Azzali, N. D. Cilia, C. De Stefano, F. Fontanella, M. Giacobini, and L. Vanneschi, Automatic feature extraction with vectorial genetic programming for Alzheimer's disease prediction through handwriting analysis, *Swarm and Evolutionary Computation*, vol. 87, p. 101571, 2024.

[21] L. Wei, M. Chen, L. Xing, Q. Wan, Y. Song, Y. Chen, and Y. Chen, Knowledge-transfer based genetic programming algorithm for multi-objective dynamic agile earth observation satellite scheduling problem, *Swarm and Evolutionary Computation*, vol. 85, p. 101460, 2024.

[22] S. Luke and L. Spector, A revised comparison of crossover and mutation in genetic programming, *Genetic Programming*, vol. 98, pp. 55–72, 1998.

[23] D. Wang, D. Tan, and L. Liu, Particle swarm optimization algorithm: an overview, *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018.

[24] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[25] S. Mirjalili, S. M. Mirjalili, and A. Lewis, Grey wolf optimizer, *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[26] S. Arora and S. Singh, Butterfly optimization algorithm: a novel approach for global optimization, *Soft Computing*, vol. 23, no. 3, pp. 715–734, 2019.

[27] J. Zhao, T. Zhang, T. Gao, and Z. Luo, Interference range-Doppler image detection by stacking classifier design for over-the-horizon radar, *Proc. of the 6th International Conference on Electronic Engineering and Informatics (EEI)*, IEEE, pp. 52–57, 2024.

[28] Z. Meng, W. Zhang, S. Shen, C. Yu, and K. Zhang, Secure interaction-based feature selection for vertical federated learning, *Proc. of the IEEE International Conference on Communications (ICC)*, IEEE, pp. 4608–4613, 2024.

[29] A. Asuncion and D. J. Newman, *UCI Machine Learning Repository*, University of California, Irvine, CA, USA, 2007.

[30] A. Menon, K. Mehrotra, C. K. Mohan, and S. Ranka, Characterization of a class of sigmoid functions with applications to neural networks, *Neural Networks*, vol. 9, no. 5, pp. 819–835, 1996.

[31] P. Hu, J.-S. Pan, S.-C. Chu, and C. Sun, Multi-surrogate assisted binary particle swarm optimization algorithm and its application for feature selection, *Applied Soft Computing*, vol. 121, p. 108736, 2022.

[32] T.-G. Ngo, H. Nguyen-Cong, T.-T.-T. Nguyen, and T.-K. Dao, Optimal parameter-feature selection using binary PSO for enhanced classification performance, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 14, no. 4, pp. 172–183, 2023.

[33] M. Abdel-Basset, D. El-Shahat, I. El-Henawy, V. H. C. De Albuquerque, and S. Mirjalili, A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection, *Expert Systems with Applications*, vol. 139, p. 112824, 2020.

[34] S. Taghian and M. H. Nadimi-Shahraki, Binary sine cosine algorithms for feature selection from medical data, *arXiv preprint arXiv:1911.07805*, 2019.

[35] S. Arora and P. Anand, Binary butterfly optimization approaches for feature selection, *Expert Systems with Applications*, vol. 116, pp. 147–160, 2019.

[36] J.-S. Pan, H.-J. Shi, S.-C. Chu, P. Hu, and H. A. Shehadeh, Parallel binary rafflesia optimization algorithm and its application in feature selection problem, *Symmetry*, vol. 15, no. 5, p. 1073, 2023.

[37] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, Efficient KNN classification with different numbers of nearest neighbors, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2017.

[38] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, Metaheuristic algorithms on feature selection: a survey of one decade of research (2009–2019), *IEEE Access*, vol. 9, pp. 26766–26791, 2021.