Multi-Strategy Enhanced Hippopotamus Optimization for Robot Path Planning

Lizhen Wu, Xudong Li, Xiang Fang, Tingting Zhang, Shaofei Chen, Chaoyang Chen, Chang Wang*

***Abstract*—As a core technology for autonomous driving and robot navigation, robot path planning often exhibits slow convergence and a tendency to become trapped in local optima in complex obstacle environments. The Hippopotamus Optimization (HO) algorithm can be used for path planning by simulating hippopotamus herd behavior. However, it suffers from insufficient population diversity and a fixed search step size. In this paper, we propose a Multi-Strategy Enhanced Hippopotamus Optimization (MSEHO) algorithm. Firstly, a Logistic chaotic mapping initializes the population, and a similarity-based classification mechanism divides the population into elite groups that search in different directions to optimize resource allocation. Secondly, a Levy-Logistic chaotic disturbance operator is introduced in early iterations to enhance global exploration, and a dynamic step-size Brownian motion strategy is employed in later stages to improve local exploitation precision. The simulation results demonstrate that MSEHO outperforms the comparative algorithms in both convergence speed and solution diversity across 29 CEC2017 benchmark functions and three path-planning scenarios.**

***Index Terms*—Hippopotamus optimization, path planning, Chaotic mapping, Elite population**

I. Introduction

Path planning aims to determine a collision-free and feasible trajectory that connects predefined start and goal positions within a specified operational environment [1]. Because robotic applications must account for path planning alongside robot kinematics and environmental uncertainty, robot path planning problems are typically formulated as multi-objective optimization problems that must simultaneously ensure collision avoidance while optimizing path length, energy consumption, motion smoothness, and task timeliness. The inherent complexity arising from diverse environmental constraints and multifaceted objective functions renders the problem NP-hard, with computational complexity increasing exponentially with environmental scale and constraint strength [2].

Metaheuristic algorithms are effective for solving NP-hard problems. Currently, there are four types of algorithms: bio-inspired evolutionary algorithms, swarm intelligence-based algorithms, human behavior-inspired algorithms, and physics/chemistry-based algorithms. Evolutionary algorithms, such as the Genetic Algorithm (GA) [3] and Differential Evolution (DE) [4], emulate natural selection principles to improve solution quality iteratively. Swarm intelligence algorithms include Particle Swarm Optimization (PSO) [5], Ant Colony Optimization (ACO) [6], and Whale Optimization Algorithm (WOA) [7]. Human behavior-inspired approaches include Human Learning Optimization (HLO) [8], and a physics-based method is Simulated Annealing (SA) [9]. By simulating various intelligent behaviors, these algorithms offer diverse approaches to solving high-dimensional, constrained optimization problems in robot path planning. However, they often face inherent trade-offs, such as premature convergence, sensitivity to parameter tuning, poor adaptability to unstructured landscapes, and insufficient late-stage exploitation, which limit their overall performance on complex path-planning tasks.

The Hippopotamus Optimization Algorithm (HO), recently introduced by Amiri et al. [9], is a novel metaheuristic inspired by the social behavior of hippopotamuses in natural ecosystems. HO mathematically models three primary behavioral patterns: position updating, predator defense, and predator evasion. HO's framework incorporates dominant individual guidance for population updates, defensive position adjustments against predators, and evasion tactics toward the nearest water bodies, corresponding to the global exploration and local exploitation phases, which collectively enable efficient search-space navigation while mitigating entrapment in local optima. HO has demonstrated success across diverse domains, including engineering optimization, neural network parameter tuning, and financial forecasting [10]–[12]. Subsequent enhancements by researchers have further expanded HO's capabilities. Mehta et al. incorporated adaptive weight mechanisms and opposition-based learning to prevent premature convergence in mechanical component design [13]. Tejani et al. applied HO to optimize truss structure parameters and determine optimal cross-sectional dimensions [14]. Yuan et al. integrated HO with Long Short-Term Memory (LSTM) networks to address gradient vanishing in pressure vessel design while maintaining prediction accuracy and robustness [15]. Yang et al. combined HO with an improved Strength Pareto Evolutionary Algorithm (SPEA2) to optimize task offloading in edge cloud computing environments [16]. However, the HO

Lizhen Wu, Xudong Li, Shaofei Chen and Chang Wang are with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China, 410073. {lzwu, lixudong, chenshaofei01, wangchang07}@nudt.edu.cn

Xiang Fang and Tingting Zhang are with the Key Laboratory of Maritime Intelligent Cyberspace Technology, Ministry of Education, Hohai University, Nanjing, China, 210024, fangxiang202009@yeah.net, 20151951@hhu.edu.cn.

Chaoyang Chen is with the School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan 411201, China, cychen@hnust.edu.cn.

This work was supported by the National Natural Science Foundation of China under Grant 62576354 and 62533010.

* Chang Wang is the corresponding author.

exhibits slow convergence and is susceptible to local optima in complex problems.

In this paper, we develop a multi-strategy enhanced hippopotamus optimization (MSEHO) algorithm based on an adaptive differential elite group strategy, a Levy-Logistic chaotic mapping, and a dynamic step-size Brownian motion strategy. We have validated, using the CEC2017 benchmark functions [17], that MSEHO outperforms existing high-quality algorithms in comprehensive optimization performance on complex problems. The main contributions are as follows:

- We propose a similarity-based adaptive differential elite group strategy that partitions the population into multiple directional elite groups, guiding position updates toward high-quality solutions to accelerate convergence.
- We adopt the Levy-Logistic chaotic mapping for initial solution generation and iterative disturbance, ensuring uniform coverage of the solution space and enhancing early-stage global exploration.
- We integrate dynamic step-size Brownian motion to adapt to late-stage iteration needs, improving local exploitation precision and balancing global exploration with local exploitation.

II. Problem Formulation

A. Raster map model

In this paper, we use a two-dimensional raster topology model to build the simulation environment. The environment is modeled as a raster map, with a corresponding binary matrix representing obstacles (see Fig. 1). The passable area is represented by a white grid cell, marked 0 in the matrix, while the obstacle area is identified by a black grid cell, marked 1 in the matrix. The Start node is labeled S (Start), and the target node is labeled G (Goal). By establishing a mapping between the Cartesian coordinate system and the grid matrix, the position of each grid cell can be converted from a coordinate value to its grid serial number using Eq. (1), thereby building a complete spatial topological model.

$$\begin{cases} x_{num} = \begin{cases} m, \text{ otherwise, if } mod(num, m) = 0 \\ mod(num, m) - 0.5, \text{ otherwise} \end{cases} \\ y_{num} = n + 0.5 - ceil(num/n) \end{cases} \tag{1}$$

In Eq. (1), the sequence number of the grid is expressed as *num*, where *m* and *n* are the number of rows and columns of the raster map, (x_{num}, y_{num}) is the coordinate of the *num* th grid, x_{num} is the corresponding horizontal coordinate of the grid, y_{num} is the corresponding vertical coordinate of the grid, and $ceil()$ and $mod()$ are the rounding up and remainder functions, respectively.

B. Constraint condition

Path planning mainly considers three constraint indices: path length, path safety, and path smoothness. Path length *L* is the primary optimization goal in path planning, representing the distance the robot travels from the start to the end.

$$\begin{aligned} L &= \sum_{i=1}^{k-1} \text{dist}(P_i, P_{i+1}) \\ \text{dist}(P_i, P_{i+1}) &= \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \end{aligned} \tag{2}$$

where k is the total number of grids through which the path passes, P_i represents the i th grid in the path, (x_i, y_i) is the coordinate of the i th grid in the path, and $dist(P_i, P_{i+1})$ represents the Euclidean distance between the two grid elements P_i and P_{i+1}.

The path safety degree *S* quantifies the distance between the path and the obstacle, ensuring the robot remains clear of the obstacle and reducing the risk of collision.

$$\begin{aligned} S &= \min \sum_{i=1}^{k} FO((x_i, y_i)) \\ FO &= \begin{cases} \inf, \ if(x_i, y_i) \in obstacle \\ 0, \ else \end{cases} \end{aligned} \tag{3}$$

where $\inf$ is an infinite penalty cost when the path intersects any obstacles, thereby rendering the trajectory invalid, and *obstacle* denotes the set of obstacle grids.

Path smoothness *R* reduces the number of turns and angle changes of the robot, and improves movement efficiency.

$$\begin{aligned} R &= \frac{1}{k-2} \sum_{i=2}^{k-1} |\theta_i - \theta_{i-1}| \\ \theta_i &= \arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) - \arctan\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right) \end{aligned} \tag{4}$$

where θ_i represents the steering angle between the i th grid cell in the path and the front and rear grid cells. The smaller the path smoothness, the smaller the path steering angle changes.

The goal of path planning is to solve for the minimum value of the comprehensive constraint function:

$$\begin{aligned} \hat{L} &= \frac{L_{min}}{L} \\ \hat{R} &= \frac{R}{\pi} \\ \min F &= \begin{cases} \omega_1 \cdot \hat{L} + \omega_2 \cdot \hat{R}, S = 0 \\ \inf, otherwise \end{cases} \end{aligned} \tag{5}$$

III. Hippopotamus optimization algorithm

The HO algorithm consists of three stages: initialization, exploration, and development. At initialization, the hippopotamus population's locations are randomly generated, and each hippopotamus represents a candidate solution to the optimization problem. The initialization formula is as follows:

$$\begin{aligned} X_i &= \{x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots\} \\ x_{ij} &= lb_j + rand \cdot (ub_j - lb_j) \end{aligned} \tag{6}$$

where x_{ij} is the dimensional position of the i th hippo, ub_j and lb_j are the upper and lower bounds of the i th dimension, respectively, and *rand* represents the random number taking the range $[0, 1]$.

Discovery Phase 1: The hippo group comprises female hippos, young adult males, and dominant males. Young, immature hippos may leave the group out of curiosity, while adult males defend their territory and drive out other males. Simulating hippopotamus position updating behavior in water, the dominant male hippo (optimal solution) leads other male hippos' position updating to:

$$xp_{ij}^{1} = x_{ij} + r_1 \cdot (D_{hippo} - I_1 \cdot x_{ij}) \tag{7}$$

where xp_{ij}^{1} is the location of the other male hippos, D_{hippo} is the location of the dominant hippo, r_1 is the random number of the range $[0, 1]$, and $I_1 \in [0, 1]$ is the control parameter.

When a juvenile hippo moves away from the female hippo, the position of the male, female, or juvenile hippo in the group is updated as follows:

$$\begin{aligned} xp_{ij}^{2} &= \begin{cases} x_{ij} + r_5 \cdot (D_{hippo} - I_2 \cdot MG_i), T > 0.6 \\ x_{ij}, \text{ otherwise} \end{cases} \\ T &= \exp(-\frac{t}{T_{max}}) \end{aligned} \tag{8}$$

where xp_{ij}^{2} represents the position of the female or juvenile hippo, r_5 is the random vector of the upper and lower bounds range, I_2 represents the random number of the range $[1, 2]$, representing the average of some hippopotamus locations randomly selected, t is the current iteration number, and T_{max} is the maximum iteration number.

Discovery Phase 2: Hippos protect themselves and the group by turning toward predators and making calls to deter them. The formula for predator position is:

$$\begin{aligned} \mathrm{RP} &= \overrightarrow{\mathrm{RL}} \oplus \mathrm{P}_j \\ \mathrm{RC} &= \frac{b}{c - d\cos(2\pi g)} \\ \overrightarrow{D} &= \mathrm{P}_j - X_i \\ xp_{ij}^{3} &= \begin{cases} \mathrm{RP} + \mathrm{RC} \cdot \left(\frac{1}{D}\right), & \mathrm{F}_{p_j} < \mathrm{F}_i \\ \mathrm{RP} + \mathrm{RC} \cdot \left(\frac{1}{2D + r_9}\right), & \mathrm{F}_{p_j} \geq \mathrm{F}_i \end{cases} \end{aligned} \tag{9}$$

where xp_{ij}^{3} represents the location where the predator invaded the hippo's territory, $\overrightarrow{\mathrm{RL}}$ represents a random vector with a Levy distribution, P_j represents the location of the predator in the search space; b, c, d, and g are uniformly distributed random numbers, where $b \in [1, 1.5]$, $c \in [2, 3]$, $d \in [2, 4]$, and $g \in [2, 4]$. $\overrightarrow{D}$ represents the distance from the ith hippo to the predator, F_{p_j} is the fitness value of the predator's position, F_i is the fitness value corresponding to the hippo's position, and r_9 represents the random variable of the range $[-1, 1]$.

In the Development stage, when hippos are unable to repel predators through defensive behavior, they attempt to flee to the nearest water area to avoid predation. The position update of the escape behavior is as follows:

$$xp_{1j}^{4} = lb_j + r_{10}[\frac{lb_j}{t} + r_{11}(\frac{ub_j - lb_j}{t})] \tag{10}$$

where xp_{ij}^{4} is the safe location found by the hippo, r_{10} is a random number in the range $[0, 1]$, r_{11} represents a random number conforming to a normal distribution, and t is the current iteration number.

IV. Hippo optimization algorithm with multi-strategy integration

A. Adaptive differential elite group strategy

To address the waste of computational resources caused by traditional optimization algorithms that randomly explore the solution space, an adaptive differential elitist strategy is proposed in [18]. By screening and retaining elite individuals with excellent fitness, the poor solution is forced to adopt directional development guided by the elite solution, thereby significantly improving convergence speed and computational efficiency.

$$N_e = \sqrt{N} \cdot \exp(1 - \frac{t}{T_{\max}}) \tag{11}$$

where N_e represents the total number of elite populations, N represents the number of populations, t represents the number of current iterations, and T_{max} represents the maximum number of iterations.

Because elite solutions may cluster in local optima and overlook nearby global optima, a similarity-based mechanism for classifying elite solutions is proposed. By calculating the similarity between the elite solution and the current global optimal solution, the elite solution is classified into two types: highly similar to the global optimal solution and significantly different. In the early stages of the algorithm, most poor solutions are based on the first type of elite, which is randomly selected, and perform large-scale exploration in their neighborhoods. In the middle and late stages, they turn to deep development, with the second type of elite as the core.

$$\begin{aligned} s(X_e, gbest) &= \frac{X_e \cdot gbest}{||X_e||||gbest||} \\ X_e &\in \begin{cases} X_{E_1}, if s(X_e, gbest) \geq \delta \\ X_{E_2}, if s(X_e, gbest) < \delta \end{cases} \end{aligned} \tag{12}$$

where $s(X_e, gbest)$ denotes the similarity between the elite individual and the global optimal solution, X_e denotes an elite individual, X_{E_1} and X_{E_2} denote the first and second elite groups, respectively. δ denotes the similarity difference factor.

B. Levy-Logistic Chaotic Mapping

To address the issue that random initial solution generation can lead to an uneven population distribution, as reported in [19], [20], we use the Lochis chaotic map to generate the initial solution, ensuring a uniformly distributed initial population across the solution space.

$$\begin{aligned} x_{1j} &= \mathrm{rand}(lb, ub), j = 1, \ldots n \\ x_{i+1,j} &= rd \cdot x_{ij} \cdot \left(1 - \frac{x_{ij}}{ub - lb}\right) \\ rd &= \mathrm{rand}(3.57, 4) \end{aligned} \tag{13}$$

where x_{ij} represents the ith sub-variable of bubble X_i in bubble population X, $\mathrm{rand}(lb, ub)$ represents any number in the lower limit lb and upper limit ub, $\mathrm{rand}(3.57, 4)$ represents any number from 3.57 to 4. Random values are drawn from

the specified upper and lower bounds. Each subvariable $x_{i+1,j}$ in the subsequent individual X_{i+1} is updated according to the above formula based on the corresponding location subvariable x_{ij} of the previous individual X_i, i.e., X_2 is calculated according to X_1, and then X_3 is calculated based on X_2 and so on to obtain the initial values of all individuals.

To address the problem of reduced convergence speed caused by random movement during algorithm execution to explore the surrounding solution space, as shown in [21]–[23], we propose the Levy-Logistic chaotic perturbation operator, which generates a large, long disturbance by leveraging the heavy-tailed distribution characteristics of Levy flights to traverse the obstacle area quickly. The current position oscillates at a small amplitude under the logistic chaotic map, and the mixed weights of the two strategies are adaptively adjusted. Smooth transition from coarse-grained global scanning to fine-grained local optimization.

Levy step size s follows the distribution:

$$\begin{aligned} S &\sim \frac{\phi \cdot \beta}{|\tau|^{1/\gamma}} \\ \phi &= \left(\frac{\Gamma(1+\gamma) \cdot \sin(\pi\gamma/2)}{\Gamma((1+\gamma)/2) \cdot \gamma \cdot 2^{(\gamma-1)/2}} \right)^{1/\gamma} \end{aligned} \tag{14}$$

where parameter $\gamma \in (1, 3]$ controls the step length tail thickness, β is the scale factor, and $\tau \sim N(0, 1)$ is the standard normally distributed random number.

The individual update rules are:

$$\begin{aligned} x_{i+1,j} &= x_{ij} + \lambda(t) \cdot s \cdot \Delta_{chaos} \\ \lambda(t) &= \lambda_{max} - \frac{t}{T_{max}}(\lambda_{max} - \lambda_{min}) \\ \Delta_{chaos} &= rd \cdot x_{ij} \cdot (1 - \frac{x_{ij}}{ub\text{-}lb}) \\ rd &= rand(3.57, 4) \end{aligned} \tag{15}$$

where x_{ij} is the jth subvariable of individual ith, λ_{max} and λ_{min} are control parameters satisfying $\lambda_{max} + \lambda_{min} = 1, \lambda_{max} - 0.3 \geq \lambda_{min}$, rand$(3.57, 4)$ represents a random number from 3.57 to 4, t represents the number of current iterations, and T_{max} represents the maximum number of iterations.

C. Dynamic step size Brownian motion strategy

To prevent the algorithm from falling into a local optimal solution, as shown in [23], [24], a dynamic step size Brownian motion strategy is proposed, which simulates the random walk characteristics of Brownian particles, introduces controllable random perturbations into the algorithm, combines the nonlinear weight attenuation and boundary constraint mechanism, and gives the dynamic step size Brownian motion the ability to dynamically adjust the search range while retaining the small step length high-frequency perturbation characteristics.

$$\begin{aligned} \Delta_{\text{brown}} &= N(0, \sigma(t)^2) \\ \sigma(t)^2 &= \sigma_{\max} \cdot e^{-2t/T} \\ w(t) &= w_{\min} + (w_{\max} - w_{\min}) \cdot e^{-3t/T} \\ x_{i+1,j} &= x_{ij} + (g_j - x_{ij}) \cdot w(t) \cdot \Delta_{\text{brown}} \end{aligned} \tag{16}$$

where $\sigma_{max} = 1.2$ is the initial maximum disturbance intensity, t represents the number of current iterations, T_{max} represents the maximum number of iterations, w_{max} and w_{min} are the control parameters, satisfying $w_{max} + w_{min} = 1, w_{max} - 0.6 \geq w_{min}$, and g_j is the jth subvariable in the global optimal solution gbest.

D. Time Complexity Analysis

The time complexity of both HO and MSEHO algorithms is closely related to the number of iterations *T*, population size *N*, and the location update mechanism. The computational complexity of the original HO is:

$$O(\text{HO}) = O(N \cdot (1 + (5 \cdot T)/2)) \tag{17}$$

where $O(N)$ represents the computational complexity of the initial allocation of the algorithm, the computational complexity of the initial phase in HO is expressed as $O(N \cdot T)$, the computational complexity of the second stage in HO is $O(N \cdot T/2)$, and the computational complexity of the third stage is $O(N \cdot T)$.

Similarly, the complexity of MSEHO is:

$$O(\text{MSEHO}) = O\left(N \cdot (5 + T/2)\right) \tag{18}$$

where the computational complexity of the initial phase in MSEHO is $O(T)$, the computational complexity of the initial solution generated by using Logistic chaos mapping is $O((N-1) \cdot T)$, the computational complexity of the adaptive differential elite strategy is $O(N \cdot T)$, the computational complexity of the second stage in HO is $O(N \cdot (T/2))$, and the computational complexity of the Levy-Logistic chaotic disturbance is $O(c_1 \cdot N \cdot T + 2)$. The computational complexity of the third stage is $O(N \cdot T)$, the computational complexity of the dynamic step size Brownian motion strategy is $O(c_2 \cdot N \cdot T + 1)$, c_1 and c_2 are constant factors, both less than 1.

Through our improvements, the original HO algorithm has enhanced its ability to explore the global space while optimizing locally, and it can adapt better to complex optimization problems.

V. Simulations and Results

A. Performance comparison on benchmark test set

1) Experimental parameter setting: To evaluate the efficiency of the proposed algorithm, two other improved Hippopotamus Optimization Algorithms, MHO [13] and MOHO [14], and five population-based optimization algorithms were compared on the CEC 2017 benchmark functions [17]. According to the criteria provided by CEC 2017, the search space for each variable was $[-100, 100]$, population size $N = 30$, dimension $D = 100$, and maximum number of iterations $T_{max} = 300$. To reduce randomness, each algorithm was repeated 10 times, and the mean and standard deviation were computed.

2) Comparative experiment on CEC 2017: The experimental results are presented in Table I, where the best results are shown in bold.

Table I presents the mean, standard deviation, and Friedman test results for each algorithm on the CEC2017 test dataset. Overall, MSEHO exhibits the strongest comprehensive optimization performance and outperforms HO across all test functions. MSEHO achieves the lowest mean and standard deviation across most test functions, indicating not only improved optimization performance relative to the Hippopotamus Optimization Algorithm but also stronger search ability compared with MHO and MOHO. For functions F_5, F_{14}, F_{17}, and F_{19}, MHO obtains the optimal mean value, but it can be seen that MSEHO yields a mean value only slightly worse than MHO on these four functions. MOHO obtains the optimal solution for functions F_2, F_{13}, F_{15}, and F_{26}. From the perspective of standard deviation and Friedman's test, the standard deviation of MOHO and MHO is often larger than that of MSEHO, indicating that the solving performance of these two improved algorithms is less stable than that of MSEHO.

For comparative experiments, we selected classical and novel intelligent optimization algorithms, including the Genetic Algorithm (GA) and the Improved Artificial Fish Swarm Algorithm [25] (IDEAFSA) and the Improved Sparrow Search Algorithm (SSA) [26]. The comparative experimental results are also shown in Table I.

From the perspective of the overall average, MSEHO achieves the optimal value on most functions, indicating stronger global search ability. Since swarm intelligence algorithms often exhibit slow search in the middle and late stages and are prone to falling into local optima, MSEHO exhibits stronger local search ability, as measured by the minimum standard deviation.

B. Multi-scenario robot path planning simulation experiment

1) Experimental parameter setting: To verify that MSEHO outperforms other algorithms in robot path planning tasks with respect to search and optimization performance, we design three terrain scenarios of varying complexity for comparative experiments. The population size for each algorithm was set to 50, and the maximum number of iterations was 100. To avoid accidental influence on the experimental results, the comparison experiment for each terrain scene was repeated 10 times to assess the differences between MSEHO and other algorithms. The starting point of each scene was set to (0, 0), and the endpoint to (20, 20).

2) Comparative Simulations for Robot Path Planning: The experiment was repeated 10 times across 3 terrain scenarios, and the total path length and time for the paths planned by MSEHO and other algorithms are shown in Table II. The Wilcoxon rank-sum test can determine whether there are significant differences between multiple samples. The Wilcoxon rank-sum test results for various algorithms in the robot path-planning experiment are shown in Table II.

From the path lengths obtained by the algorithms in Figure 1 and Table II, we note that each algorithm has advantages and disadvantages in finding the optimal path across three terrain scenarios with varying complexity.

In scenario 1, both SA and MSEHO find the optimal path, whereas PSO has the longest path, which may be due to suboptimal solutions in the particle population affecting the search for the optimal solution in the middle and late stages of the algorithm.

In scenario 2, GA, SA, and MSEHO obtain the optimal solution, while PSO obtains a suboptimal solution. APF performs slightly worse than PSO, possibly because of the smaller resultant force in APF.

In scenario 3, except for MSEHO, none of the other algorithms can find the optimal solution, and the complexity of the scene greatly increases the difficulty for the swarm intelligence algorithm to escape from local optima.

We note that as the number of obstacles increases, the performance of the swarm intelligence algorithm first increases and then decreases. This may be because the search efficiency is low when the search space is too large, and there are fewer obstacles. As the number of obstacles increases, the search space gradually decreases, but the solution algorithm improves. However, when there are more obstacles, the algorithm falls into a local optimal solution and reduces solution efficiency. Based on the experimental results presented above, MSEHO outperforms other algorithms in solving path-planning problems.

3) Policy Validity Analysis: To verify the effectiveness of the improved strategies and to explore their specific impact on UAV path planning, we evaluated the performance of DEHO, LLHO, and BMHO in robot path planning using only the adaptive differential elite group strategy, the Levy-Logistic chaotic mapping strategy, and the dynamic step-size Browne motion strategy, respectively. To avoid redundancy and focus on core experimental analysis, the above three single-strategy improvement algorithms were applied to the complex terrain scenarios 1 to 3. In each scenario, each algorithm was run 10 times independently, and the mean and standard deviation of the experimental results were also recorded in Table II.

We observe that the average cost functions of the planned paths of DEHO, LLHO, and BMHO are lower than that of the original HO but slightly higher than that of the MSEHO with multi-strategy integration. The results show that the three single strategies proposed in this paper can effectively improve the performance of the HO algorithm in path-planning problems. Therefore, combining them can further enhance the algorithm's optimization capability. In addition, further analysis of the data in Table II shows that the mean cost function of DEHO across scenarios 1 to 3 is lower than that of LLHO and the BMHO. Still, its standard deviation is significantly higher than those of the latter two, indicating that DEHO's robustness is weak. This phenomenon effectively prevents elite genetic strategies from producing high-quality candidate solutions. Once high-quality candidate solutions emerge from the algorithm, DEHO can quickly integrate and guide other

TABLE I

RESULTS OF CEC TEST SET COMPARISON BETWEEN MSEHO AND OTHER IMPROVED HO ALGORITHMS AND COMPARISON BETWEEN MSEHO AND OTHER SWARM INTELLIGENCE ALGORITHMS.

Function	Algorithm (*Mean*)								Algorithm (*Std*)							
	HO	MHO [13]	MOHO [14]	MSEHO	GA	IDEAFSA [25]	ISSA [26]	MSEHO	HO	MHO [13]	MOHO [14]	MSEHO	GA	IDEAFSA [25]	ISSA [26]	MSEHO
F_1	2.31e10	9.12e09	2.75e09	7.52e05	2.31e10	9.15e09	2.80e09	7.59e05	4.94e09	3.85e09	2.49e09	6.51e05	4.94e09	3.89e09	2.48e09	6.46e05
F_2	3.27e05	3.29e05	2.52e05	2.59e05	3.27e05	3.24e05	2.62e05	2.63e05	2.53e04	1.84e04	2.31e04	1.86e04	2.53e04	1.92e04	2.26e04	1.93e04
F_3	3.46e03	2.13e03	1.22e03	8.49e02	3.46e03	2.19e03	1.13e03	8.58e02	7.89e02	3.45e02	1.60e02	7.19e01	7.89e02	3.53e02	1.63e02	7.19e01
F_4	1.36e03	1.25e03	1.21e03	1.23e03	1.36e03	1.19e03	1.30e03	1.27e03	8.19e01	8.47e01	7.23e01	8.52e01	8.19e01	8.53e01	7.14e01	8.49e01
F_5	6.65e02	6.47e02	6.54e02	6.48e02	6.65e02	6.53e02	6.61e02	6.39e02	7.87e00	7.10e00	7.20e00	6.14e00	7.87e00	7.10e00	7.23e00	6.08e00
F_6	3.26e03	2.72e03	2.76e03	2.27e03	3.26e03	2.71e03	2.77e03	2.24e03	2.03e02	2.79e02	1.87e02	1.46e02	2.03e02	2.84e02	1.79e02	1.52e02
F_7	1.86e03	1.72e03	1.59e03	1.52e03	1.86e03	1.70e03	1.68e03	1.53e03	5.90e01	6.56e01	1.12e02	1.17e02	5.90e01	6.56e01	1.16e02	1.07e02
F_8	3.38e04	2.64e04	2.44e04	2.41e04	3.38e04	2.57e04	2.46e04	2.36e04	3.22e03	2.47e03	2.62e03	3.04e03	3.22e03	2.45e03	2.65e03	2.77e03
F_9	1.68e04	1.59e04	1.68e04	1.54e04	1.68e04	1.62e04	1.60e04	1.55e04	1.53e03	1.43e03	2.08e03	1.62e03	1.53e03	1.44e03	2.08e03	1.68e03
F_{10}	5.79e04	4.23e04	9.43e03	6.61e03	5.79e04	4.32e04	9.36e03	6.69e03	2.32e04	9.03e03	3.04e03	2.69e03	2.32e04	9.04e03	2.99e03	2.69e03
F_{11}	1.36e09	7.37e08	6.12e08	3.87e07	1.36e09	7.32e08	6.15e08	3.82e07	4.69e08	2.02e08	1.78e09	1.27e07	4.69e08	1.99e08	1.70e09	1.20e07
F_{12}	8.31e04	2.99e05	2.35e04	5.92e03	8.31e04	2.92e05	2.42e04	5.88e03	4.55e05	4.89e04	9.20e03	3.18e03	4.55e05	4.80e04	8.40e03	3.27e03
F_{13}	5.88e06	3.91e06	8.70e05	1.85e06	5.88e06	3.91e06	8.60e05	1.89e06	2.40e06	1.76e06	5.08e05	6.31e05	2.40e06	1.74e06	4.98e05	6.25e05
F_{14}	3.36e04	8.47e03	8.90e03	8.51e03	3.36e04	8.11e03	8.56e03	8.24e03	1.10e04	3.22e03	4.15e03	1.22e03	1.10e04	3.15e03	4.11e03	1.28e03
F_{15}	6.87e03	6.06e03	5.91e03	6.15e03	6.87e03	5.97e03	5.91e03	6.12e03	6.34e02	6.53e02	5.24e02	4.66e02	6.34e02	6.55e02	5.22e02	4.68e02
F_{16}	5.59e03	5.35e03	5.74e03	5.05e03	5.59e03	5.28e03	5.81e03	5.06e03	5.89e02	5.49e02	6.81e02	4.80e02	5.89e02	5.46e02	6.72e02	4.77e02
F_{17}	4.69e06	1.44e06	2.37e06	1.65e06	4.69e06	2.52e06	2.34e06	1.63e06	2.14e06	1.68e06	7.47e05	5.29e05	2.14e06	1.70e06	7.44e05	5.34e05
F_{18}	4.44e04	9.40e03	8.43e03	4.75e03	4.44e04	9.31e03	8.46e03	4.78e03	1.27e04	4.92e03	6.59e03	3.34e03	1.27e04	4.90e03	6.54e03	3.39e03
F_{19}	5.28e03	5.11e03	5.72e03	5.12e03	5.28e03	5.02e03	5.73e03	5.07e03	1.84e03	5.78e02	4.85e02	5.35e02	1.84e03	5.79e02	5.25e02	4.48e02
F_{20}	3.44e03	3.14e03	3.24e03	3.01e03	3.44e03	3.04e03	3.30e03	2.98e03	1.57e02	8.89e01	1.51e02	8.76e01	1.57e02	8.84e01	1.43e02	8.72e01
F_{21}	2.12e04	2.04e04	2.08e04	1.88e04	2.12e04	2.04e04	2.11e04	1.92e04	1.63e03	1.64e03	1.91e03	1.17e03	1.63e03	1.67e03	1.97e03	1.07e03
F_{22}	3.89e03	3.54e03	3.72e03	3.44e03	3.89e03	3.54e03	3.81e03	3.36e03	1.91e02	1.42e02	1.56e02	1.14e02	1.91e02	1.34e02	1.50e02	1.07e02
F_{23}	5.27e03	4.63e03	4.86e03	4.49e03	5.27e03	4.71e03	4.78e03	4.52e03	2.84e02	2.75e02	2.74e02	1.24e02	2.84e02	2.70e02	2.73e02	1.20e02
F_{24}	5.12e03	4.49e03	3.74e03	3.39e03	5.12e03	4.45e03	3.74e03	3.43e03	3.61e02	3.28e02	1.70e02	6.49e01	3.61e02	3.28e02	1.77e02	6.42e01
F_{25}	2.70e04	1.84e04	2.34e04	1.72e04	2.70e04	1.22e04	2.39e04	1.29e04	2.58e03	1.98e03	3.55e03	6.25e03	2.58e03	1.95e03	3.54e03	6.30e03
F_{26}	4.62e03	4.38e03	3.91e03	4.14e03	4.62e03	4.35e03	3.88e03	4.04e03	3.47e02	1.52e02	1.94e02	2.28e02	3.47e02	1.59e02	1.91e02	2.27e02
F_{27}	6.62e03	5.98e03	4.04e03	3.74e03	6.62e03	6.07e03	4.02e03	3.82e03	6.99e02	6.02e02	2.77e02	4.67e01	6.99e02	6.11e02	2.83e02	4.75e01
F_{28}	8.88e03	7.92e03	8.09e03	7.20e03	8.88e03	8.02e03	8.11e03	7.26e03	6.27e02	4.95e02	7.63e02	5.27e02	6.27e02	4.87e02	7.63e02	5.26e02
F_{29}	1.09e07	4.75e06	3.98e05	9.81e04	1.09e07	4.67e06	4.00e05	9.74e04	9.84e06	3.56e06	2.37e05	6.89e04	9.84e06	3.59e06	2.28e05	6.91e04
	Rankmean								*Ranking*							
Friedman	3.78	2.62	2.43	1.67	3.12	2.41	2.27	1.73	4	3	2	1	4	3	2	1

TABLE II

THE PATH LENGTH OBTAINED BY EACH ALGORITHM AND THE FITNESS VALUE DATA OF PATHS PLANNED BY THREE SINGLE-STRATEGY IMPROVEMENT ALGORITHMS.

Environmental	Indicator	Algorithm					Indicator	Algorithm			
		GA	PSO	APF	SA	MSEHO		HO	DEHO	LLHO	BMHO
1 (6 obstacles)	length	30.0416	31.2132	30.0416	28.6247	28.6274	Mean	32.6274	30.0416	30.3848	31.2312
							Std	0.8422	1.0727	0.5635	0.7681
2 (8 obstacles)	length	29.799	30.3848	30.6274	29.799	29.799	Mean	31.4558	30.3848	30.9706	30.6274
							Std	1.5164	1.2462	0.8162	1.0547
3 (12 obstacles)	length	30.3848	30.6274	31.4558	30.6274	29.799	Mean	32.0416	29.799	31.2312	31.2312
							Std	1.7506	1.6849	1.2711	1.1407

solutions to update, thereby approaching the global optimal solution. However, in the absence of high-quality candidate solutions in the initial population, DEHO's performance may fluctuate significantly. In contrast, LLHO and BMHO improve stability by introducing chaos disturbance and a random-walk mechanism, thereby maintaining more consistent performance across different scenarios.

VI. CONCLUSION

In this paper, a multi-strategy enhanced hippopotamus optimization algorithm (MSEHO) is proposed, integrating an adaptive differential elite population strategy, Levy-Logistic chaotic mapping, and dynamic step-size Brownian motion strategy. Simulation results on the CEC2017 benchmark functions show that MSEHO significantly improves optimization accuracy and stability while retaining the original algorithm's fast convergence and enhancing global search robustness in complex discrete scenarios. However, this work has limitations: it focuses solely on discrete map scenarios and does not consider continuous optimization problems, and hyperparameters are tuned empirically for specific benchmarks. Future research will extend MSEHO to continuous optimization by adjusting strategy-mapping mechanisms, integrating adaptive hyperparameter tuning to reduce manual intervention, and validating its performance on practical engineering tasks (e.g., real UAV path planning) to strengthen its practical applicability.

REFERENCES

[1] Y. Zhi, K. Shi, and X. Liu, "Review of path planning algorithms for mobile robots," in *Advances in Guidance, Navigation and Control*, L. Yan, H. Duan, and Y. Deng, Eds. Singapore: Springer Nature Singapore, 2025, pp. 580–589.

[2] L. Guangshun and S. Hongbo, "Study of technology on path planning for mobile robots," in *2008 Chinese Control and Decision Conference*, 2008, pp. 3295–3300.

[3] A. Anwaar, A. Ashraf, W. H. K. Bangyal, and M. Iqbal, "Genetic algorithms: Brief review on genetic algorithms for global optimization problems," in *2022 Human-Centered Cognitive Systems (HCCS)*, 2022, pp. 1–6.

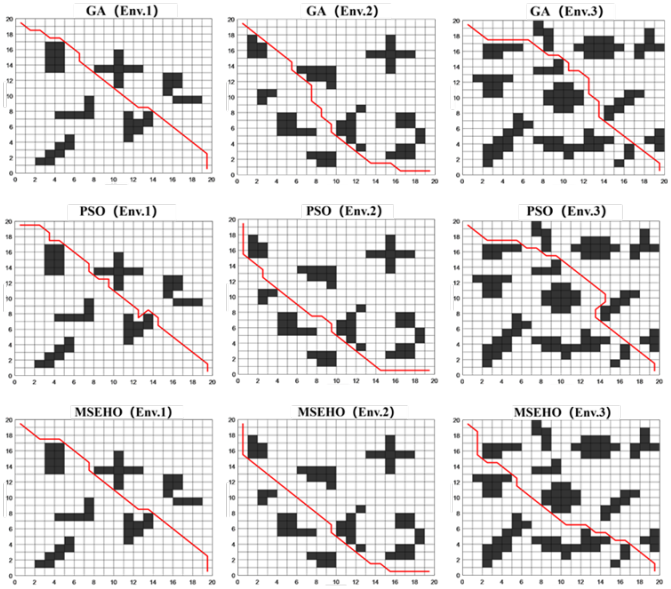
[4] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Particle swarm optimization or differential evolution—a comparison," *Engineering Applications of Artificial Intelligence*, vol. 121, p. 106008, 2023.

[5] M. Jain, V. Saihjpal, N. Singh, and S. B. Singh, "An overview of variants and advancements of pso algorithm," *Applied Sciences*, vol. 12, no. 17, 2022.

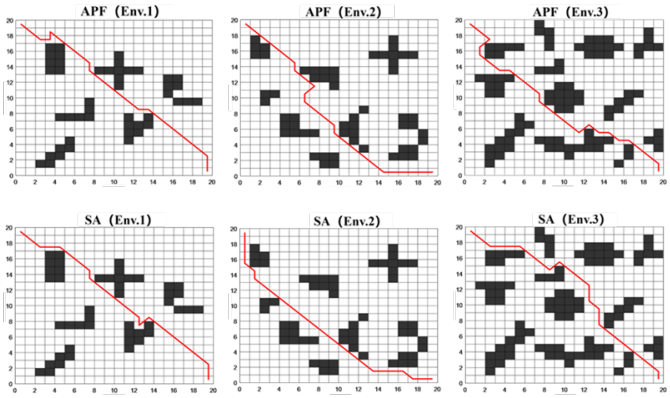
[6] M. García, N. López, and I. Rodríguez, "A full process algebraic representation of ant colony optimization," *Inf. Sci.*, vol. 658, no. C, Feb. 2024.

[7] K.-W. Huang, Z.-X. Wu, C.-L. Jiang, Z.-H. Huang, and S.-H. Lee, "Wpo: A whale particle optimization algorithm," *Int. J. Computational Intelligence Systems*, vol. 16, no. 1, p. 115, December 2023.

[8] J. Du, L. Wang, and M. F. . M. I. Menhas, "A human learning optimization algorithm with competitive and cooperative learning," *Complex Intelligent Systems*, pp. 797–823, 2023.



(a) Path planning results of swarm intelligent algorithm.



(b) Path planning reslts of non-swarm intelligent algorithm.

Fig. 1. Path planning results of the swarm intelligent algorithm and the non-swarm intelligent algorithm.

[9] M. H. Amiri, N. Mehrabi Hashjin, M. Montazeri, S. Mirjalili, and N. Khodadadi, "Hippopotamus optimization algorithm: A novel nature-inspired optimization algorithm," *Scientific Reports*, vol. 14, no. 1, p. 5032, feb 2024.

[10] H. Wang, N. N. Binti Mansor, and H. B. Mokhlis, "Novel hybrid optimization technique for solar photovoltaic output prediction using improved hippopotamus algorithm," *Applied Sciences*, vol. 14, no. 17, 2024.

[11] M. A. Abdelaziz, A. A. Ali, R. A. Swief, and R. Elazab, "Optimizing energy-efficient grid performance: Integrating electric vehicles, DSTATCOM, and renewable sources using the Hippopotamus Optimization Algorithm," *Scientific Reports*, vol. 14, no. 1, p. 28974, nov 2024.

[12] W. Wang and J. Tian, "An improved hippopotamus optimization algorithm utilizing swarm-elite learning mechanism's levy flight and quadratic interpolation operators for optimal parameters extraction in photovoltaic models," in *Advanced Intelligent Computing Technology and Applications*, D.-S. Huang, X. Zhang, and W. Chen, Eds. Singapore: Springer Nature Singapore, 2024, pp. 264–277.

[13] P. Mehta, S. M. Sait, B. S. Yildiz, and A. R. Yildiz, "Enhanced hippopotamus optimization algorithm and artificial neural network for mechanical component design," *Materials Testing*, vol. 67, pp. 655 – 662, 2025.

[14] G. G. Tejani, S. K. Sharma, N. Mashru, P. Patel, and P. Jangir, "Optimization of truss structures with two archive-boosted moho algorithm," *Alexandria Engineering Journal*, vol. 120, pp. 296–317, 2025.

[15] Y. Lei, "Application of improved hippopotamus optimization algorithm integrating multi-strategy," *Journal of Hainan University*, vol. 43, pp. 289–296, 2025.

[16] H.-J. Wang, J.-S. Pan, T.-T. Nguyen, and S. Weng, "Distribution network reconfiguration with distributed generation based on parallel slime mould algorithm," *Energy*, vol. 244, p. 123011, 2022.

[17] Q. Zhang, "CEC-Benchmark-Functions," jan 2026. [Online]. Available: https://github.com/tsingke/CEC-Benchmark-Functions

[18] X. Meng and X. Zhu, "Autonomous obstacle avoidance path planning for grasping manipulator based on elite smoothing ant colony algorithm," *Symmetry*, vol. 14, no. 9, 2022.

[19] H. Zhang, W. Hong, and M. Chen, "A path planning strategy for intelligent sweeping robots," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2019, pp. 11–15.

[20] K. Yu and B. Xu, "Mobile robot path planning based on improved elite ant colony algorithm," in *2024 8th International Conference on Robotics, Control and Automation (ICRCA)*, 2024, pp. 63–67.

[21] X.-J. Wu, L. Xu, R. Zhen, and X.-L. Wu, "Global and Local Moth-flame Optimization Algorithm for UAV Formation Path Planning Under Multi-constraints," *International Journal of Control, Automation and Systems*, vol. 21, no. 3, pp. 1032–1047, Mar. 2023.

[22] J. Lian, W. Yu, and W. Liu, "A chaotic adaptive particle swarm optimization for robot path planning," in *2019 Chinese Control Conference (CCC)*, 2019, pp. 4751–4756.

[23] C. Lv and F. Yu, "The Application of a Complex Composite Fractal Interpolation Algorithm in the Seabed Terrain Simulation," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–6, Jul. 2018.

[24] H. E. Espitia and J. I. Sofrony, "Path planning of mobile robots using potential fields and swarms of brownian particles," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 123–129.

[25] H. Li, Z. Sun, and M. Luo, "Agv path planning based on improved hybrid artificial fish swarm algorithm," in *2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 11, 2023, pp. 1676–1681.

[26] D. Wu, F. Hao, C. Yuan, and Y. Li, "An improved sparrow search algorithm for mobile robot path planning," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 1899–1903.