Journal of Research in Science and Engineering (JRSE)
ISSN: 1656-1996 Volume-7, Issue-12, December 2025
DOI: 10.53469/jrse.2025.07(12).12

Enhancing Kepler Optimization: An Improved Algorithm with Superior Search Capabilities

Yi Huang[1], Lanyue Yang[2]

[1,2]College of Computer Science and Engineering, Chongqing University of Technology, Chongqing, China
[1]*hy6@stu.cqut.edu.cn,* [2]*yanglanyue0128@outlook.com*

Abstract: ***To enhance the global search capability, convergence accuracy, and stability of optimization algorithms, this paper improves the basic Kepler optimization algorithm and proposes an enhanced Kepler optimization algorithm. This algorithm incorporates a tent map to enhance the uniformity and diversity of population initialization; utilizes Lévy flight strategy to balance the algorithm's global exploration and local exploitation capabilities; and designs an adaptive perturbation mechanism to enable the algorithm to effectively escape from local optima in the later stages of iteration. To verify the performance of the proposed algorithm, comprehensive experiments were conducted on the CEC2022 benchmark test function set, and it was compared with six classic swarm intelligence optimization algorithms. The experimental results show that the improved algorithm exhibits significant advantages in convergence speed, solution accuracy, and robustness, especially when dealing with complex, high-dimensional optimization problems. This study provides new insights for the design of optimization algorithms, and the improved algorithm is expected to be widely applied in fields such as engineering optimization and machine learning parameter tuning.***

Keywords: Kepler Optimization Algorithm, Kent Chaos Mapping, Dynamic Lévy Flight, Opposition-Based Learning.

1. Introduction

Swarm intelligence optimization algorithms are optimization methods inspired by collective behavior in nature, possessing advantages such as strong global search capability, ease of implementation, and strong adaptability. Currently, mainstream swarm intelligence optimization algorithms include the classic Particle Swarm Optimization (PSO) [1], Ant Colony Optimization (ACO) [2], Firefly Algorithm (FA) [3], Grey Wolf Optimizer (GWO) [4], Bat Algorithm (BA) [5], Artificial Bee Colony (ABC) [6], Whale Optimization Algorithm (WOA) [7], as well as newly proposed algorithms such as Parrot Optimization Algorithm [8] and Kepler Optimization Algorithm (KOA) [9] in recent years.

The Particle Swarm Optimization (PSO) algorithm, one of the most classic swarm intelligence methods, was proposed by Kennedy and Eberhart in 1995, which simulates the social behavior of bird flocks to solve problems [10]. This algorithm guides particles to search for the optimal solution in the solution space through the collaboration of individual and group experiences. To enhance its performance, researchers have proposed various improvement strategies, such as introducing dynamic inertia weight, designing adaptive learning factors, and integrating local search mechanisms, effectively balancing the exploration and exploitation capabilities of the algorithm.

The Ant Colony Optimization (ACO) algorithm simulates the positive feedback mechanism of pheromone in the foraging paths of ants to solve combinatorial optimization problems. Each ant constructs a feasible solution, and the pheromone concentration represents the quality of the path (or solution component). ACO performs well in discrete search spaces, but its convergence speed may be limited on high-dimensional complex problems. Therefore, many studies have focused on integrating it with other algorithm frameworks such as Particle Swarm Optimization (PSO) [11] and Genetic Algorithm (GA) [12] to form hybrid strategies to enhance overall search efficiency.

In recent years, novel meta-heuristic algorithms inspired by ecosystems or physical models have emerged continuously, providing new insights for solving complex optimization problems. For instance, the Grey Wolf Optimization Algorithm simulates the social hierarchy and hunting strategies of wolf packs, guiding the search direction by defining α, β, and δ wolves, demonstrating excellent global search performance and robustness [13], [14]. The Firefly Algorithm moves based on the attractiveness of individual light emission intensity, suitable for handling continuous optimization problems [15]; the Bat Algorithm combines frequency modulation and pulse response strategies, enhancing the ability for local fine-grained search [16]. The Artificial Bee Colony Algorithm mimics the division of roles (worker bees, forager bees, scout bees) during honey collection by bees, achieving a balance between global exploration and local exploitation through the collaboration of various search behaviors [17]. These algorithms, with their unique bionic mechanisms, have been widely validated and applied in various benchmark tests and engineering optimization problems.

2. Kepler Optimization Algorithm

The Kepler Optimization Algorithm (KOA) is a nature-inspired metaheuristic algorithm derived from the three laws of the planetary motion of KOA. It is developed primarily to address single-objective and continuous optimization problems. The fundamental principle of KOA is to simulate the elliptical orbits of planets around a central mass, usually the Sun. This simulation helps guide candidate solutions toward the global optimum by imitating orbital dynamics. KOA demonstrates outstanding performance in terms of global search capability and solution precision. The following section presents a detailed exposition of the mathematical model and implementation framework of the algorithm.

2.1 Population Initialization

In the initialization phase of KOA, N planetary individuals are randomly generated within the whole solution space. Each individual has a dimensionality of d. The mathematical expression is given by formula (1).

$$X_i^j = X_{i,lb}^j + r \times \left(X_{i,ub}^j - X_{i,lb}^j\right), \begin{cases} i = 1,2,\dots\dots,N \\ j = 1,2,\dots\dots,d \end{cases} \quad (1)$$

Where X_i^j denotes the initial position of the $i\text{-}th$ planet in the $j\text{-}th$ dimension, lb and ub are the lower and upper bounds of the $j\text{-}th$ dimension, respectively, and $r \in [0,1]$ is a uniformly distributed random value.

2.2 Velocity Calculation

When a planet approaches the sun, its orbital velocity increases. Conversely, when it moves away from the star, its orbital velocity decreases. As an object gets closer to the Sun, the gravitational force exerted by the Sun becomes significantly stronger. To counteract this intense gravitational pull, each planet tries to accelerate its motion. The corresponding mathematical model is provided below.

$$\vec{v}_i(t) = \begin{cases} \delta \times \left(2r_4 \times \vec{X}_i - \vec{X}_b\right) + \ddot{\delta} \times \left(\vec{X}_a - \vec{X}_b\right) + (1 - R_{i-norm}(t)) \\ \times \sigma \times \vec{U}_1 \times \vec{r}_5 \times \left(\vec{X}_{i,ub} - \vec{X}_{i,lb}\right), if\ R_{i-norm}(t) \le 0.5 \\ r_4 \times \kappa \times \left(\vec{X}_a - \vec{X}_i\right) + (1 - R_{i-norm}(t)) \\ \times \sigma \times U_2 \times \vec{r}_5 \times \left(r_3 \times \vec{X}_{i,ub} - \vec{X}_{i,lb}\right), else \end{cases} \quad (2)$$

$$\delta = \vec{U} \times M \times \kappa, \ddot{\delta} = \left(1 - \vec{U}\right) \times \vec{M} \times \kappa \quad (3)$$

$$\kappa = \left[\mu(t) \times (m_i + M_s) \times \left(\frac{2}{R_i(t)+\varepsilon} - \frac{1}{a_i(t)+\varepsilon}\right)\right]^{\frac{1}{2}} \quad (4)$$

$$M = (r_3 \times (1 - r_4) + r_4), \vec{M} = (r_3 \times (1 - \vec{r}_5) + \vec{r}_5) \quad (5)$$

$$\vec{U} = \begin{cases} 0, if\ \vec{r}_5 \le \vec{r}_6 \\ 1, else \end{cases} \quad (6)$$

$$\sigma = \begin{cases} 1, if\ r_4 \le 0.5 \\ -1, else \end{cases} \quad (7)$$

Where $\vec{v}_i(t)$ denotes the velocity of the $i\text{-}th$ planet at time t, r_3 and r_4 are probabilistic variables that follow a uniform distribution in the interval $[0,1]$, $\vec{r}_5$ and $\vec{r}_6$ are two binary vectors whose elements are restricted to values of 0 or 1, $\vec{X}_a$ and $\vec{X}_b$ represent two planets randomly selected from the current population, σ is a randomly selected scalar used to adjust the direction of motion.

2.3 Local Optimum Escape Strategy

Most planets rotate counterclockwise when they are close to the Sun, but some planets are exceptions and orbit the Sun in the opposite direction. In the original KOA, this phenomenon is utilized to adjust the search direction and facilitate escape from local optima. To achieve this, KOA introduces a control variable σ that dynamically adjusts the search direction and thereby controls the orbital trajectory of planets around the Sun.

2.4 Position Update

Planets follow Kepler's laws and orbit the Sun along elliptical paths. During their approach to the Sun, the planets continuously revolve: they gradually move closer to the star and then slowly move away. KOA is designed based on this behavior and includes two main steps, exploration and exploitation. When a planet moves away from the Sun, the algorithm performs exploration to search for a broader solution space; when a planet moves closer to the Sun, the algorithm performs exploitation to focus on optimizing the current solution. The mathematical model can be expressed by formula (8).

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \sigma \times \vec{V}_i(t) + \vec{U} \times \left(F_{gi}(t) + |r|\right) \times \left(\vec{X}_S(t) - \vec{X}_i(t)\right) \quad (8)$$

2.5 Elite Retention Mechanism

To ensure that the optimal solution is not destroyed, KOA employs the following elite retention strategy.

$$\vec{X}_i(t+1) = \begin{cases} \vec{X}_i(t+1), if\ f\left(\vec{X}_i(t+1)\right) \le f\left(\vec{X}_i(t)\right) \\ \vec{X}_i(t), \quad else \end{cases} \quad (9)$$

3. Enhancing Kepler Optimization

To address the inherent limitations of the original KOA, such as premature con vergence and insufficient global search capability. Specifically, a tent map mechanism is integrated to improve population diversity during initialization, a Lévy flight strategy is embedded to strengthen global exploration, and dynamic opposition-based learning (DOBL) is adopted to prevent stagnation and accelerate convergence. Furthermore, the fit ness evaluation process is upgraded through a multi-objective fitness function, which simultaneously optimizes classification accuracy and feature subset compactness.

3.1 Initialization based on Tent Map

We adopt tent map during the population initialization phase to improve the quality of population distribution and enhance the coverage of the search space. Chaotic mapping is a nonlinear dynamic system that shows sensitivity to initial conditions, ergodicity, and pseudo-randomness. The tent map improves KOA's initialization by promoting diversity and preventing early convergence.

The tent map is a classical one-dimensional discrete chaotic system, and the mathematical expression is given by formula (10).

$$x_{n+1} = \begin{cases} \frac{x_n}{\mu}, & 0 \le x_n < \mu \\ \frac{1-x_n}{1-\mu}, & \mu \le x_n \le 1 \end{cases} \quad (10)$$

The parameter $\mu \in (0,1)$ is a control parameter, and $\mu = 0.5$ is commonly selected to achieve the most desirable chaotic behavior. The tent map exhibits excellent ergodicity and can generate uniformly distributed chaotic sequences within the interval $[0,1]$. The sample points generated by this map can more effectively fill the search space, which is beneficial to overcome the problem of premature convergence in the algorithm.

The tent map is applied during the population initialization phase. First, an initial seed $x_0 \in (0,1)$ is set, and a chaotic sequence of length $N \times D$ is generated using the iterative Tent map, where N is the population size and D is the dimension of problem. Then, the chaotic sequence is converted into values within the range of each dimension in the search space by formula (11).

$$X_{i,j} = X_{min}^{(j)} + x_{i,j} \cdot (X_{max}^{(j)} - X_{min}^{(j)}) \quad (11)$$

where the chaotic sequence values $x_{i,j} \in [0,1]$ are mapped to the specific interval of the search space. Finally, the transformed values $x_{i,j}$ are used as the initial positions of the particles, which are then substituted into the main loop of the KOA to begin the optimization process.

The initial individuals generated by the tent map make the population spread evenly in the search space. This helps explore complex solution spaces and reduces the chance of getting stuck in local optima.

3.2 Dynamic Opposition-Based Learning Strategy

Tizhoosh first proposed the Opposition-Based Learning (OBL) strategy in 2005, which was inspired by the concept of opposition [18]. To date, the OBL strategy and its improved versions have been widely applied in various optimization algorithms, demonstrating significant advantages in improving algorithm performance [19]. The basic definition of the OBL strategy is given by formula (12).

$$\hat{X}_i = LB + UB - X_i \quad (12)$$

Where X_i and $\hat{X}_i$ represent the original solution and its opposite solution, respectively; the symbols LB and UB denote the lower and upper limits of the search space.

In OBL strategy, the individual with the higher fitness value between the original solution and its opposite solution is selected for the next generation iteration. Although this strategy can improve population diversity and quality to some extent, it maintains a fixed distance between the original and opposite solutions during generation, which leads to a lack of necessary randomness in the search process. This limitation may restrict the algorithm's optimization capability during global search.

This paper introduces an improved Dynamic Opposition-Based Learning (DOBL) strategy to solve the low-randomness problem of the traditional OBL strategy. The aim is to increase population diversity and improve the quality of solutions. The DOBL strategy adds a dynamic boundary adjustment to the original OBL strategy. The mathematical model of the DOBL strategy is presented by formula (13).

$$\hat{X}_{i,j}^t = a_j^t + b_j^t - X_{i,j}^t \quad (13)$$

Where $\hat{X}_{i,j}^t$ and $X_{i,j}^t$ represent the opposite solution and the original solution of the vector $i\text{-}th$ in the dimension $j\text{-}th$ during the iteration t, respectively; a_j^t and b_j^t denote the lower and upper bounds in the dimension $j\text{-}th$ at the iteration t, respectively. The mathematical model of a_j^t and b_j^t is expressed as follows.

$$a_j^t = min(X_j^t), b_j^t = max(X_j^t) \quad (14)$$

The pseudo-code of the DOBL is given in Table 1.

Table 1: Pseudo-code of DOBL Strategy

Algorithm 1 Pseudo-code of DOBL Strategy
1: Input: D, N, X // dimension, population size, original solutions
2: Output: X
3: for i = 1 to N do
4: for j = 1 to D do
5: $\hat{X}_{i,j}^t = a_j^t + b_j^t - X_{i,j}^t$ // generate opposite solution
6: end for
7: end for
8: Calculate the fitness values of the original and opposite solutions
9: X ← select the optimal solutions from the set$\{X, \hat{X}\}$

3.3 Lévy Flight Strategy

To enhance the global exploration ability of KOA and avoid premature convergence, recent studies have increasingly focused on integrating KOA with other mechanisms. In particular, the incorporation of random jump mechanisms has proven effective in improving search diversity. Among these, the Lévy Flight strategy has demonstrated remarkable optimization potential due to its distinctive behavior of combining frequent small steps with occasional long-distance jumps, thereby introducing nonlinear perturbations and long-range exploration into the algorithm.

Lévy flight follows a random walk mechanism governed by the Lévy stable distribution. The step length of this mechanism exhibits a heavy-tailed distribution characterized by frequent small local searches interspersed with occasional long distance jumps. This behavior has been widely observed in nature, including animal foraging patterns, photon trajectories, and even human mobility modeling. The probability distribution function of the Lévy flight is given by formula (15).

$$L(s) \sim |s|^{-1-\beta}, 0 < \beta \leq 2 \quad (15)$$

Where β is the stability index of the L´evy distribution, which determines the frequency of long jumps. A commonly used value is $\beta = 1.5$, which balances search stability and diversity.
Lévy flight can be numerically simulated using the Mantegna algorithm, with the jump step length L generated using the formula below.

$$L = \frac{u}{|v|^{1/\beta}}, u \sim \mathcal{N}(0, \sigma^2), v \sim \mathcal{N}(0,1) \quad (16)$$

$$\sigma = \left[\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma((1+\beta)/2)\cdot\beta\cdot 2^{(\beta-1)/2}}\right]^{1/\beta} \quad (17)$$

The random perturbations generated by this strategy can be used for position updates in the algorithm, allowing individuals to jump unpredictably to regions far from the current solution. This enhances the algorithm's ability to escape local optima.

The main goal of incorporating the Lévy flight strategy into the KOA is to improve population diversity and global search capability. This paper uses Lévy perturbations in the position update phase. The original KOA updates the positions of particles by simulating the motion of celestial bodies. After each update, the Lévy flight adds a jump-based perturbation. The modified update equation is given by formula (18).

$$X_i^{t+1} = X_i^t + \alpha \cdot \text{Lévy}(\beta) \cdot (X_i^t - X_{\text{best}}^t) \quad (18)$$

Where, X_i^t denotes the position of the particle $i\text{-}th$ in iteration t, X_{best}^t is the current global best solution, α is the scaling factor controlling the jump magnitude, and Lévy(β) represents the generated Lévy distributed perturbation. The parameter β is the Lévy index, which determines the shape of the distribution.

3.4 Procedure of EKOA

Table 2: Pseudo-code of EKOA

Algorithm 2 Pseudo-code of EKOA

1: Input: N, D, T_{max},ub,lb,fobj
2: Output: X_s
3: Initialize population Positions using Tent chaotic mapping
4: Evaluate fitness of each individual
5: Select best individual X_s
6: while t < Tmax do
7: Compute β and α parameters for DOBL strategy
8: Update X_s using DOBL strategy
9: for i=1:N do
10: Update semi-major axis a(t), orbit radius R(i), normalized masses M, MS, m, and gravitational factor F g(i)
11: Randomly select two individuals a, b
12: Generate dynamic mass V and random value r
13: if r < 0.5 then
14: Update position using Lévy flight strategy
15: else
16: Update position using original formula
17: end if
18: end for
19: Use elitism to update best individual X_s
20: t ← t + 1
21: end while

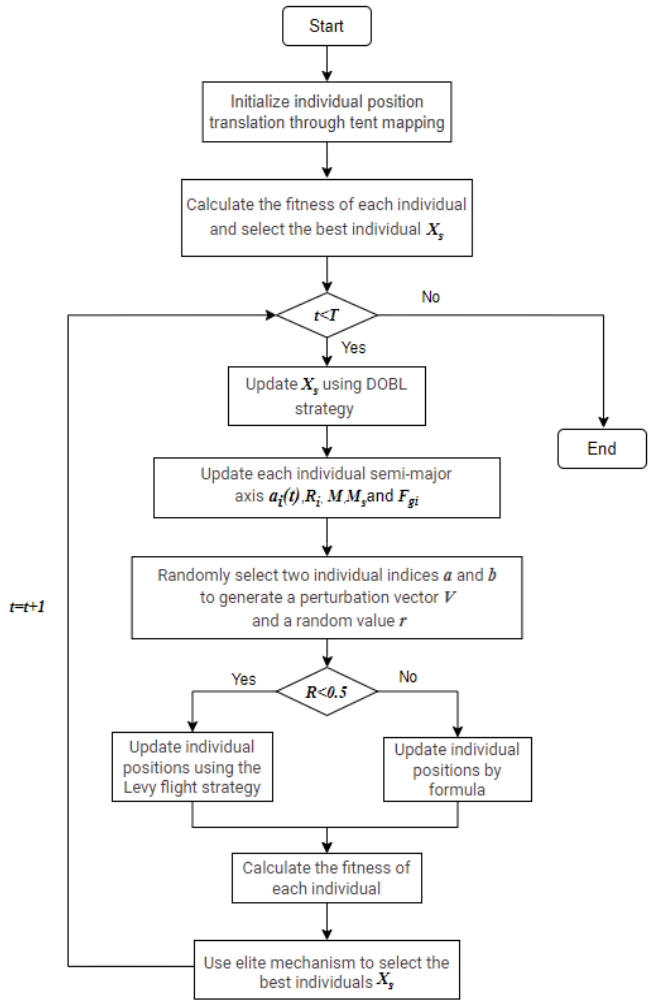

Figure 1: Flowchart of the EKOA Algorithm

In this study, we improve the classical KOA by introducing tent chaotic mapping, Lévy flight strategy, and Dynamic Opposition-Based Learning. These methods help increase population diversity, speed up convergence, and improve the ability to escape local optima. The pseudocode of EKOA is shown in Table 2, and Figure 1 illustrates the flowchart of the proposed EKOA algorithm.

The algorithm begins by initializing the population positions using Tent chaotic mapping to ensure diversity in the search space. The fitness of each individual is then evaluated and the best solution is selected. During each iteration, a dynamic opposition-based learning strategy is applied to adjust the current best solution, enhancing the global search capability. Based on a random threshold, an individual's position is updated either using the Lévy flight strategy to improve exploration and avoid local optima or by the original position update formula to maintain search stability. At the end of each iteration, an elitism strategy is used to preserve and update the best global solution. This process continues until the maximum number of iterations reached, enabling effective optimization of the problem.

4. Numerical Experiments

In this section, we designed two groups of experiments to comprehensively evaluate the performance and limitations of EKOA. To evaluate the numerical optimization performance of EKOA, a set of experiments is conducted on 12 benchmark functions derived from the CEC2022 test suite. The experimental results are compared with six mainstream metaheuristic algorithms, namely: Sine Cosine Algorithm (SCA), Seagull Optimization (SO), Harris Hawks Optimization (HHO), Grey Wolf Optimizer (GWO), Whale Optimization Algorithm (WOA), and the original KOA.

4.1 Experiment Setting

To ensure the accuracy and reliability of the experimental results, a consistent experimental setting is strictly followed in all algorithms. The same parameter configurations are applied uniformly for a fair comparison and reduce experimental bias, comparative algorithms are executed using their default parameters listed in Table 3.

All experiments were carried out on a computer configured with a 64-bit Windows operating system, an Intel(R) Core (TM) i7-8750H CPU with a base frequency of 2.20 GHz and 8 GB of RAM. The programming environment used was MATLAB 2024a.

Table 3: Parameter Settings of Algorithms

Algorithm	Parameters
Common Parameters	Population Size (nPop) = 30 Maximum Iterations (Maxiter) = 1000 Problem Dimension (dim) = 30 Number of Runs (N) = 3
SCA	A=2(Default)
SO	θ1=0.25, θ2=0.6, c1=2=0.5, c3=2
HHO	b=[2,0], β=1.5(Default)
GWO	α=2-t*((2)/Max_iter)
WOA	α=2-t*((2)/Max_iter) α2=-1+t*((-1)/Max_iter)
KOA, EKOA	Tc=3, M0=0.1, λ=15

4.2 Benchmarking one CEC2022 Benchmark Functions

In this paper, we use the CEC2022 benchmark functions suite of complex test functions for evaluation, which cover a wide range of types, including unimodal functions, basic functions, hybrid functions, and composition functions. The detailed characteristics of the CEC2022 benchmark functions are listed in Table 4, where Fun_i^* denotes the global optimum of the $i\text{-}th$ function.

Table 4: Test functions utilized in CEC2022

NO.	Function	Fun_i^*
	Unimodal Function	
1	Shifted and full Rotated Zakharov Function	300
	Basic Function	
2	Shifted and full Rotated Rosenbrock's Function	400
3	Shifted and full Rotated Expanded Schaffer's f6 Function	600
4	Shifted and full Rotated Non-Continuous Rastrigin's f6 Function	800
5	Shifted and full Rotated Levy Function	900
	Hybrid Function	
6	Hybrid Function 1 (N=3)	1800
7	Hybrid Function 2 (N=6)	2000
8	Hybrid Function 3 (N=5)	2200
	Composition Function	
9	Composition Function 1 (N=5)	2300
10	Composition Function 2 (N=4)	2400
11	Composition Function 3 (N=5)	2600
12	Composition Function 4 (N=6)	2700

Search range: $[-100, 100]^D$

Table 5 presents the performance comparison results of the EKOA algorithm with other comparative algorithms on the CEC2022 test function set. The data reveals that MOKOA excels in solving the 12 benchmark test functions in CEC2022, achieving a success rate of up to 75%. This advantage is reflected in multiple evaluation metrics, including the optimal solution, standard deviation, and average value.

EKOA achieved the best performance in terms of optimal values on F1, F3, F4, F6, F7, F8, F9, F10, and F12. Although some algorithms, such as SCA, HHO, and GWO, surpassed EKOA in terms of optimal value metrics on F11, F2, and F5, respectively, their "Std" values were significantly larger, indicating poor convergence stability. However, EKOA showed almost no deviation on F3, F4, F5, and F10, demonstrating high convergence stability. Furthermore, in functions F3, F4, high-dimensional function F6, and complex function F10, EKOA significantly outperformed all algorithms, exhibiting good adaptability.

Figure 2 illustrates the boxplots of the optimization results for each algorithm. Boxplots effectively reflect data distribution, variability, and stability. The proposed EKOA algorithm exhibits noticeably narrower boxes in most test functions, indicating lower variability and higher consistency over multiple runs. This highlights the algorithm's robustness and stability, as its performance is less influenced by randomness and remains reliable between experiments.

The convergence curves show how the algorithms improve during the iterations. They also show how fast and efficiently the algorithms reach the optimal solution. Observing the convergence curves, one can evaluate the stability and accuracy of the algorithms to approaching the optimal solution. In this study, based on the CEC2022 benchmark function, the convergence characteristics of the proposed EKOA and several comparative algorithms were analyzed. The relevant results are shown in Figure 3. The experimental results show that EKOA performs better than the other algorithms in convergence accuracy and speed in the CEC2022 benchmark functions. It has strong global search ability and good local optimization ability.

Table 5: Performance of algorithms across 5 metrics and Friedman ranking on CEC2022 benchmark functions

		SCA	SO	HHO	GWO	WOA	KOA	EKOA
F1	Best	1337.877	6921.200	302.485	407.147	13625.210	17893.408	300.143
	Std	505.869	1376.100	1.549	744.102	4293 348	1721.139	16.916
	Mean	1850.014	8363.176	304.257	1266.356	18582 333	19850.635	313 482
F2	Best	437.778	560.514	402.496	407.200	414.395	701.256	405.497
	Std	15.296	205.403	14.339	15.658	115.374	360.753	10.433
	Mean	452.990	724.438	408.816	424.590	501.162	1116.870	414.985
F3	Best	617.894	619.247	628.973	600.067	628 363	665.021	600.000
	Std	2.867	19.820	9.840	0.030	11.613	6.543	0.011
	Mean	621.117	637 895	636.157	600.099	640.930	672.503	600.007
F4	Best	843.040	842.625	815.070	811.204	820.941	903.087	804.975
	Std	1.195	8.692	11.281	2.600	23 509	7.244	0.995
	Mean	844.270	852.434	823.062	814.158	843.210	910.836	805.970
F5	Best	945.472	1427.859	1398.878 105.168	900.195	978.977 709.134	2698.584	903.656
	Std	48.753	94.049		8.345		466.348	0.156
	Mean	994.935	1490.118	1468.077	905.827	1538.476	3149.453	903.812
F6	Best	1218415.423	1076547.097	2811.627 2500.936	2451.408 3330.429	1944.258 1160.941	104298946.800	1832.953 1010.280
	Std	2619875.427	7439253.270				135555257.700	
	Mean	4241034.379	51711006.360	4944.686	6296.914	2988.297	224972310.400	2512.740
F7	Best	2052.086	2058.801	2035.412 17.963	2029.690 16.061	2052.271 16.906	2146.406	2002.234 10.895
	Std	7.848	30.770				18.003	
	Mean	2059.612	2084.092	2053.748	2039.014	2069.476	2162.536	2014.763
F8	Best	2229.352	2241.202	2226.681 19.402	2225.131	2227.244	2272.448	2220.933
	Std	1.705	20.984		1.377	4.520	38.089	2.175
	Mean	2231.172	2254.252	2248.872	2226.675	2232.412	2309.206	2223.270
F9	Best	2554.886	2659.032	2529.562 19.182	2529.307 44.060	2542.534 44.615	2797.998	2485.508 17.559
	Std	14.037	43.827				75.438	
	Mean	2567.397	2690.768	2548.901	2572.403	2571.694	2880.637	2495.726
F10	Best	2501.504	2522.292	2500.733 488.293	2500.315 64.150	2502.832 92.576	2700.499	2500.267
	Std	0.258	90.683				242.706	0.068
	Mean	2501.709	2626.823	2782.691	2537.400	2606.093	2974.837	2500.338
F11	Best	2776.259	4331.238	2605.440 181.124	2730.879 107.777	2736.308 145.667	33210.625 10912.531	2900.602 18.418
	Std	36.286	127.968					
	Mean	2817.573	4431.454	2710.326	2854.971	2904.072	41369.811	2916.959
F12	Best	2869.201	2894.030	2868.183 29.409	2863.206 11.203	2867.322 27.218	2956.326	2847.137
	Std	1.484	51.598				40.365	2.776
	Mean	2870.804	2930.703	2886.841	2870.026	2898.655	2982.112	2848.909

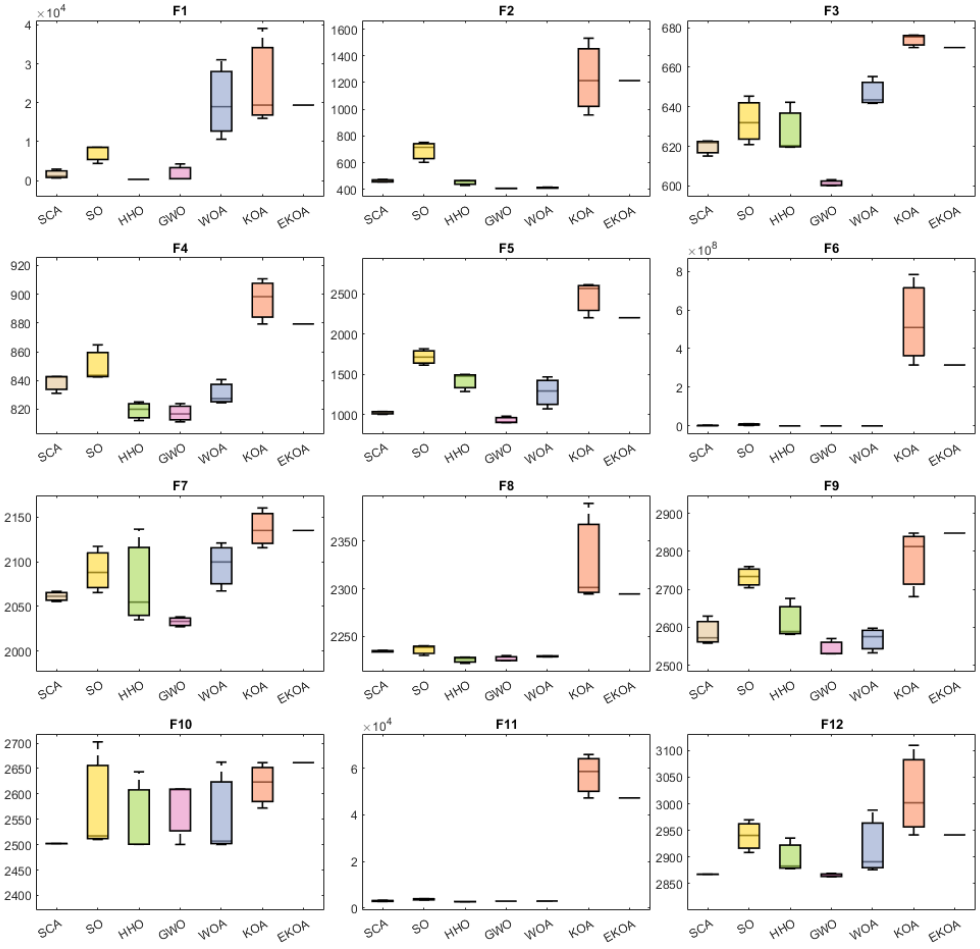

Figure 2: Boxplots of optimization results obtained by each algorithm during the process

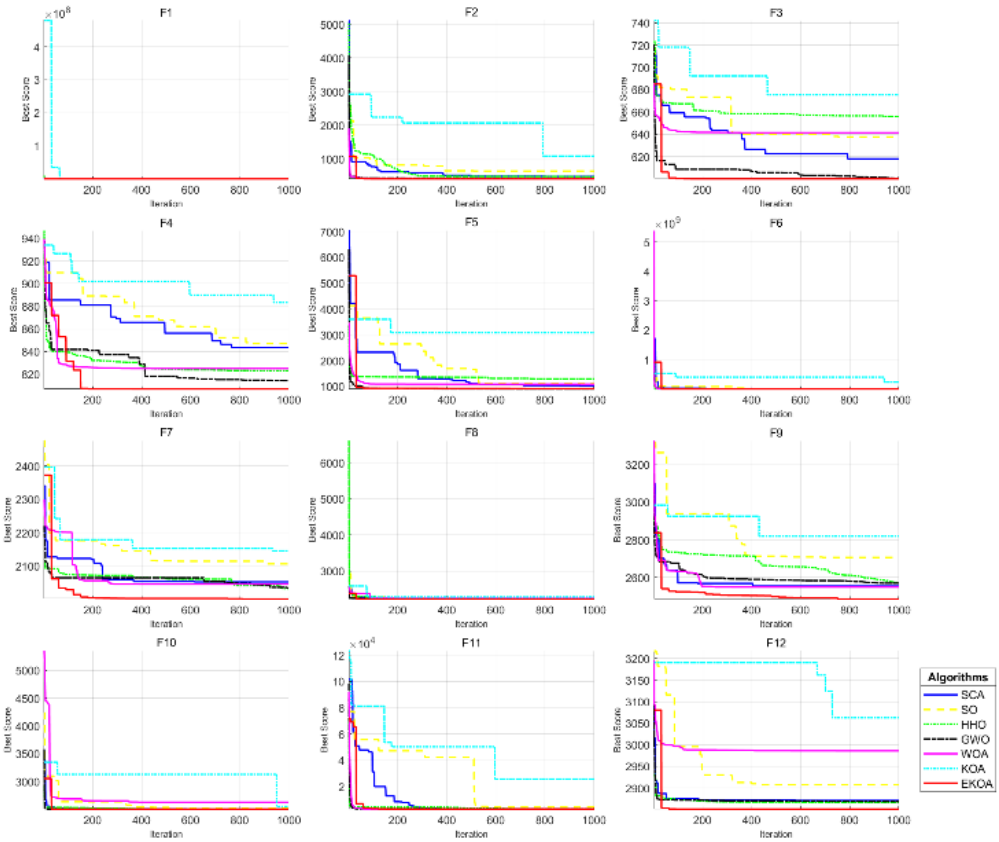

Figure3: Convergence curves of EKOA and comparative algorithms on CEC2022 benchmark functions

5. Conclusion and Future Work

Aiming at the limitations of the traditional Kepler Optimization Algorithm in handling complex optimization problems, such as susceptibility to local optima and insufficient convergence stability, this paper proposes EKOA incorporating multiple improvement strategies. During the population initialization stage, Tent chaotic mapping is introduced to generate a more diverse initial population, laying a foundation for global search. In the planetary motion simulation stage, a dynamically adjusted Lévy flight strategy is integrated to balance the algorithm's exploration and exploitation capabilities. In the orbital update mechanism, DOBL is adopted to enhance the efficiency of local refined search. To validate the performance of EKOA, systematic experiments were conducted on the CEC2022 benchmark function, and comparisons were made with several intelligent optimization algorithms such as KOA, GWO, and HHO. The experimental results show that EKOA significantly outperforms the compared algorithms in terms of convergence accuracy, stability, and robustness. Its best solution, average solution, and standard deviation metrics all demonstrate superior performance, and the convergence curves indicate faster convergence speed and stronger global optimization ability.

In the future, this research can be further deepened in the following directions: First, further optimizing the parameter adaptation mechanism and strategy integration of EKOA to enhance its performance in high-dimensional, dynamic, and multi-constraint optimization problems. Second, applying EKOA to practical engineering fields, such as neural architecture search, power system scheduling, image segmentation, and path planning in complex scenarios, to verify its practical utility. Third, exploring hybrid models combining EKOA with other intelligent computing methods, such as deep learning and fuzzy systems, to expand its application scope and problem-solving capabilities.

References

[1] Sakri, S. B., Rashid, N. B. A., & Zain, Z. M. (2018). Particle swarm optimization feature selection for breast cancer recurrence prediction. IEEE Access, 6, 29637-29647.

[2] Aghdam, M. H., Ghasem-Aghaee, N., & Basiri, M. E. (2009). Text feature selection using ant colony optimization. Expert systems with applications, 36(3), 6843-6853.

[3] Selvakumar, B., & Muneeswaran, K. (2019). Firefly algorithm based feature selection for network intrusion detection. Computers & Security, 81, 148-155.

[4] Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016). Binary grey wolf optimization approaches for feature selection. Neurocomputing, 172, 371-381.

[5] Nakamura, R. Y., Pereira, L. A., Costa, K. A., Rodrigues, D., Papa, J. P., & Yang, X. S. (2012, August). BBA: a binary bat algorithm for feature selection. In 2012 25th SIBGRAPI conference on graphics, patterns and images (pp. 291-297). IEEE.

[6] Nakamura, R. Y., Pereira, L. A., Costa, K. A., Rodrigues, D., Papa, J. P., & Yang, X. S. (2012, August). BBA: a binary bat algorithm for feature selection. In 2012 25th SIBGRAPI conference on graphics, patterns and images (pp. 291-297). IEEE.

[7] Mafarja, M., & Mirjalili, S. (2018). Whale optimization approaches for wrapper feature selection. Applied Soft Computing, 62, 441-453.

[8] Chen, T., & Yi, Y. (2024). Multi-Strategy Enhanced Parrot Optimizer: Global Optimization and Feature Selection. Biomimetics,9(11), 662.

[9] Abdel-Basset, M., Mohamed, R., Azeem, S. A. A., Jameel, M., & Abouhawwash, M. (2023). Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary motion. Knowledge-Based Systems, 268, 110454.

[10] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). ieee.

[11] Menghour, K., & Souici-Meslati, L. (2016). Hybrid ACO-PSO based approaches for feature selection. Int J Intell Eng Syst, 9(3), 65-79.

[12] Nemati, S., Basiri, M. E., Ghasem-Aghaee, N., & Aghdam, M. H. (2009). A novel ACO–GA hybrid algorithm for feature selection in protein function prediction. Expert systems with applications, 36(10), 12086-12094.

[13] Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016). Binary grey wolf optimization approaches for feature selection. Neurocomputing, 172, 371-381.

[14] Al-Tashi, Q., Md Rais, H., Abdulkadir, S. J., Mirjalili, S., & Alhussian, H. (2020). A review of grey wolf optimizer-based feature selection methods for classification. Evolutionary machine learning techniques: algorithms and applications, 273-286.

[15] Zhang, L., Mistry, K., Lim, C. P., & Neoh, S. C. (2018). Feature selection using firefly optimization for classification and regression models. Decision Support Systems, 106, 64-85.

[16] Yang, X. S., & He, X. (2013). Bat algorithm: literature review and applications. International Journal of Bio-inspired computation, 5(3), 141-149.

[17] Hancer, E., Xue, B., Zhang, M., Karaboga, D., & Akay, B. (2018). Pareto front feature selection based on artificial bee colony optimization. Information Sciences, 422, 462-479.

[18] Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 2005) and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC 2005) (Vol. 1, pp. 695–701). Vienna, Austria.

[19] Yang, Q., Song, G. W., Gao, X. D., & Others. (2023). A random elite ensemble learning swarm optimizer for high-dimensional optimization. Complex & Intelligent Systems, 9, 5467–5500.