mathematics

MDPI

Article

Application of the Improved Cuckoo Algorithm in Differential Equations

Yan Sun

School of Mathematics, Harbin Institute of Technology, Harbin 150001, China; 21s012023@stu.hit.edu.cn

Abstract: To address the drawbacks of the slow convergence speed and lack of individual information exchange in the cuckoo search (CS) algorithm, this study proposes an improved cuckoo search algorithm based on a sharing mechanism (ICSABOSM). The enhanced algorithm reinforces information sharing among individuals through the utilization of a sharing mechanism. Additionally, new search strategies are introduced in both the global and local searches of the CS. The results from numerical experiments on four standard test functions indicate that the improved algorithm outperforms the original CS in terms of search capability and performance. Building upon the improved algorithm, this paper introduces a numerical solution approach for differential equations involving the coupling of function approximation and intelligent algorithms. By constructing an approximate function using Fourier series to satisfy the conditions of the given differential equation and boundary conditions with minimal error, the proposed method minimizes errors while satisfying the differential equation and boundary conditions. The problem of solving the differential equation is then transformed into an optimization problem with the coefficients of the approximate function as variables. Furthermore, the improved cuckoo search algorithm is employed to solve this optimization problem. The specific steps of applying the improved algorithm to solve differential equations are illustrated through examples. The research outcomes broaden the application scope of the cuckoo optimization algorithm and provide a new perspective for solving differential equations.

Keywords: cuckoo search algorithm; differential equations; optimization problem; approximate solution

MSC: 68W30

Citation: Sun, Y. Application of the Improved Cuckoo Algorithm in Differential Equations. *Mathematics* **2024**, *12*, 345. https://doi.org/10.3390/math12020345

Academic Editors: Tien Anh Tran, Roman Rodriguez Aguilar, Gerhard-Wilhelm Weber, Igor Litvinchev, Joshua Thomas, Pandian Vasant and Ioannis G. Tsoulos

Received: 18 December 2023
Revised: 9 January 2024
Accepted: 19 January 2024
Published: 21 January 2024

1. Introduction

Since the seventeenth century, the widespread application of calculus has resolved numerous practical problems in fields such as physical chemistry, engineering, and population statistics, fostering the emergence of various new disciplines, including differential equations. Many problems in these fields can be described and understood through the use of differential equations, making the study of their solutions essential. However, most differential equations encountered in real-life scenarios and scientific research are complex, and determining their analytical solutions can be challenging. Although there exists a series of mature numerical methods for solving these equations, the effectiveness of the solutions often depends on the careful selection of the initial values. Achieving optimal performance and convergence characteristics requires precise choices for the initial solutions. Selecting an excellent initial point for solving differential equations is not a straightforward task. To address these challenges, researchers have developed various optimization algorithms to seek the best solutions.

In the past few decades, nature-inspired metaheuristic algorithms and their improved versions have gained popularity due to their simplicity and flexibility. Examples include the genetic algorithm (GA) [1,2], differential evolution (DE) [3], particle swarm optimization (PSO) [4,5], grey wolf optimization (GWO) [6], ant colony optimization (ACO) [7], wind-driven optimization (WDO) [8], CS [9–11], memetic algorithm (MA) [12], artificial bee

colony algorithm (ABC), and others. With the evolution of intelligent algorithms, those with global optimization performance have been widely applied to solve differential equations. Intelligent algorithms overcome the challenge of rational initial guesses for solutions, are suitable for nonsmooth, non-differentiable, or noisy objective functions, and possess certain global search capabilities. Grosan and Biazar [13], utilized evolutionary computation techniques to address the conversion of systems into multi-objective optimization problems. Jaberipour et al. [14] employed an improved particle swarm optimization (PPSO) to solve nonlinear differential equations. Oliveira and Petraglia [15] proposed the fuzzy adaptive simulated annealing algorithm (ASA) for finding solutions to arbitrary nonlinear systems of equations. Abdollahi et al. [16] used the imperialist competitive algorithm to solve systems of differential equations and demonstrated its effectiveness on some well-known test problems. Raja et al. [17] introduced another memetic approach (GA-SQP), using an improved GA algorithm as a global search tool and employing sequential quadratic programming (SQP) for efficient local search, yielding satisfactory results. Zhang proposed a niching cuckoo search algorithm (NCSA) based on the principle of fitness sharing to solve systems of nonlinear equations. The introduced algorithm enhances the CS's ability to solve nonlinear differential equations by incorporating a niching strategy. This algorithm can handle highly nonlinear problems and outperforms many algorithms mentioned in the literature. Verma [18] presented a novel hybrid algorithm of PSO and DE for finding solutions to nonlinear differential equations.

As the optimization problems we encounter in practical applications become increasingly complex, and the requirements for achieving the goals of problem-solving become higher, it is crucial for us to drive algorithms in more efficient and effective directions. The no free lunch theorem [19] proves that there is no one-size-fits-all optimization algorithm capable of solving all mathematical optimization problems. This is due to the diverse characteristics and constraints associated with different problems. Given the distinct features and constraints of various problems, it is essential to choose an appropriate optimization algorithm to obtain the optimal solution or the best possible approximate solution. Hence, to achieve better results when solving different types of optimization problems, experts and scholars optimize existing algorithms or develop new ones.

In recent years, the cuckoo search (CS) algorithm has been extensively applied in numerical optimization [20,21] and multi-objective optimization [22,23], among other domains. The CS algorithm is widely employed in diverse scenarios to search for robust solutions with fast convergence. While the efficiency of the CS algorithm generally surpasses that of other algorithms, its performance in precisely searching for optimal solutions is suboptimal, exhibiting immature convergence characteristics near local minima. Striking a balance between exploration and exploitation has become a crucial aspect when utilizing the CS algorithm. In order to fully unleash the potential of the CS, researchers have conducted numerous attempts, proposing new techniques to further enhance its performance. Improvements to the CS algorithm primarily focus on three aspects, namely:

(1) Parameter Adjustment

The CS relies primarily on two crucial parameters: the Levy flight step-size control factor and the elimination probability. In the original CS proposed by Yang et al. [24], the step-size control factor was set as a fixed value, which, in practice, diminishes the algorithm's performance. Ong, based on the assumption that cuckoos lay eggs in regions where the host bird's egg survival rate is higher, introduced an adaptive step-size adjustment strategy [25], dynamically adjusting the step size. Wang et al. proposed a variable parameter strategy [26], where the step-size proportion factor is randomly generated, and in each iteration, this value follows a uniform distribution between 0 and 1. Cheng et al. introduced an algorithm with dynamic feedback information [27], utilizing the randomness and stability trends of cloud models, dynamically adjusting the step size and discovery probability based on individual fitness values.

(2) Strategy Improvement

In the realm of strategy improvement, Ma et al. introduced an enhanced CS employing a search trend strategy [28], which enhances the overall solving performance of the algorithm. Test results demonstrate that this algorithm exhibits strong competitiveness in addressing multi-peaked and high-dimensional function optimization problems. Meng et al. developed a locally enhanced CS algorithm targeting multimodal numerical problems [29]. In this algorithm, the global optimal individual's position guides the movement of all cuckoos, improving local search capabilities. The introduction of an inertia weight achieves a balance between exploration and exploitation. Meng utilized individual constraint and collective constraint techniques (CGC) for population initialization, enhancing the diversity of initial population decision variable values. Additionally, a cosine decay strategy [30] was employed for dynamic adaptive change in probability, speeding up convergence. In the same year, Gao et al. proposed the MSACS algorithm [31], incorporating five search strategies with different characteristics to replace a single Levy flight strategy. Each cuckoo adaptively selects a search strategy with an adaptive probability in each iteration, generating new solutions in every iteration.

(3) Hybrid Algorithms

Combining the cuckoo search (CS) algorithm with other diverse algorithms can maximize the utilization of their respective strengths. Some scholars have attempted to enhance the performance of the CS algorithm through various hybrid strategies. In 2018, Saha [32] introduced the concept of population ranking in the grey wolf optimizer (GWO). After each iteration, the three best positions of the search individuals are recorded, guiding the cuckoo search in exploring new solutions. Simultaneously, the Cauchy distribution replaced the Levy flight. Testing the proposed algorithm on standard benchmark functions revealed its high competitiveness compared to existing techniques.

The improved CS algorithms described above have enhanced the performance of the CS algorithm to varying degrees in relevant studies. However, for the CS and most of its improved versions, there is a lack of effective information sharing among cuckoo individuals during the evolutionary process. This deficiency may result in the underutilization of useful information within the population, limiting the performance potential of these CS-based algorithms. We define our objective function $F(x)$ with the aim of minimizing its value. Formally, the optimization problem is articulated as follows:

$$\min \quad F(x)$$

Here, $x = (x_1, x_2, \ldots, x_n) \in R^n$. The improved cuckoo search algorithm proposed in this paper is designed to effectively solve this optimization problem, especially in the context of complex differential equations.

The main contributions of this paper are as follows:

(1) The proposal of an improved CS based on a sharing mechanism.
(2) The introduction of a numerical solution method for differential equations using the coupling of function approximation and intelligent algorithms.
(3) The application of the improved algorithm to solve differential equations.

The structure of this paper is as follows. Section 2 provides an overview of the CS and summarizes the algorithm's optimization process. Section 3 introduces the improved CS based on a sharing mechanism. Section 4 briefly outlines the general form of the optimization problems and elucidates the fundamental approach of applying the improved CS to the solution of differential equations. And Section 5 concludes this paper.

2. Cuckoo Search Algorithm

2.1. Introduction to the Cuckoo Search Algorithm

The CS is designed to mimic the behavior of cuckoos in nature, specifically their parasitic egg-laying and hatching behavior. Cuckoos exhibit parasitic egg-laying behavior,

characterized by not building their nests and not caring for their offspring. To maximize the survival rate of their eggs, cuckoos preferentially choose nests that resemble their own eggs, referring to the owner of such a nest as the host. When the host is absent from the nest, the cuckoo lays its eggs, allowing the host to feed and nurture the chicks. If the parasitic eggs go unnoticed by the host, they may be incubated and cared for by the host, but there is also a possibility of the host identifying the parasitic eggs. Once discovered as eggs not belonging to the host, the host may destroy the cuckoo eggs or abandon the current nest to build a new one, resulting in the failure of the cuckoo's parasitic plan. In order to gain more space and food, the chicks may push the eggs of the host birds out of the nest.

The design of the CS corresponds to the biological behavior of cuckoos. The algorithm begins by randomly selecting initial solutions, evaluating and ranking them, and then choosing solutions with lower fitness as seed solutions for the next iteration. In each iteration, the seed solutions undergo random perturbation and recombination, generating a new set of solutions that is subsequently ranked and selected based on fitness. Ultimately, the algorithm converges to a global or local optimal solution.

2.2. Optimization Process of the Cuckoo Search Algorithm

The foundation of the CS lies in the survival efforts of cuckoos. Due to the large size of cuckoo eggs, it is challenging for cuckoos to carry multiple eggs at once. Moreover, carrying eggs makes flight more difficult, requiring more energy to maintain stamina and evade predators. To evade predators, cuckoos must find a secure place to hatch their eggs. Cuckoos continually strive to improve their mimicry of host bird eggs, while hosts continuously refine methods for detecting parasitic eggs. Essentially, the CS algorithm is an idealized algorithm based on the following assumptions:

Assumption 1. *Cuckoos randomly select nests, and each time, they lay only one egg in the host nest.*

Assumption 2. *Based on the survival of the fittest rule, only a portion of the randomly chosen host nests, the best ones, are retained for the next generation.*

Assumption 3. *The number of host nests available for cuckoo egg laying is fixed, and there is a probability* $p_a \in [0, 1]$ *that hosts discover eggs not their own. This may lead to the abandonment of cuckoo eggs or the host nest. In such cases, the host birds migrate to new habitats and establish new nests to start afresh.*

Similar to other nature-inspired algorithms, the CS algorithm initiates its optimization process from the initialization stage and then proceeds to the iterative phase.

(1) Population Initialization Based on a Random Distribution Strategy

To begin, the search space and population size are initialized, assuming the dimensions of the search space and the population size are denoted as D and d, respectively. Before the algorithm starts running, it is necessary to distribute the search individuals across the search space.

Let the position of the *i-th* cuckoo individual in the population at the *t-th* iteration be denoted as $x_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,D}^t\right]$. The population information is initialized using a random distribution strategy, specifically through the following expression:

$$x_i^k = x_k^{low} + rand_1 \cdot (x_k^{up} - x_k^{low}),\ i = 1, 2, \ldots, d,\ k = 1, 2, \ldots, D \tag{1}$$

where $rand_1$ represents a random number following a uniform distribution $U(0, 1)$, and x_k^{up} and x_k^{low} denote the upper and lower bounds, respectively, of the *k-th* dimension. $U(0, 1)$ signifies a continuous uniform distribution defined over the interval [0, 1].

(2) Global Search Based on Lévy Flight

In most cases, animals can freely choose their direction of movement and alter it during their motion. Since the direction and shift to the next position depend on the current position, the present state, and the transition probability to the next position,

animal movement generally adheres to the principle of random walking. After numerous experiments and deepening research by experts and scholars, attempts have been made to use probability-related mathematical formulas to describe this behavior. When the step size in random walking follows a Lévy distribution, it is termed a Lévy flight. A notable characteristic of Lévy flight is that its values can be positive or negative, allowing for the generation of suitable distributions.

The step length S in Lévy flight can be expressed as Equation (2):

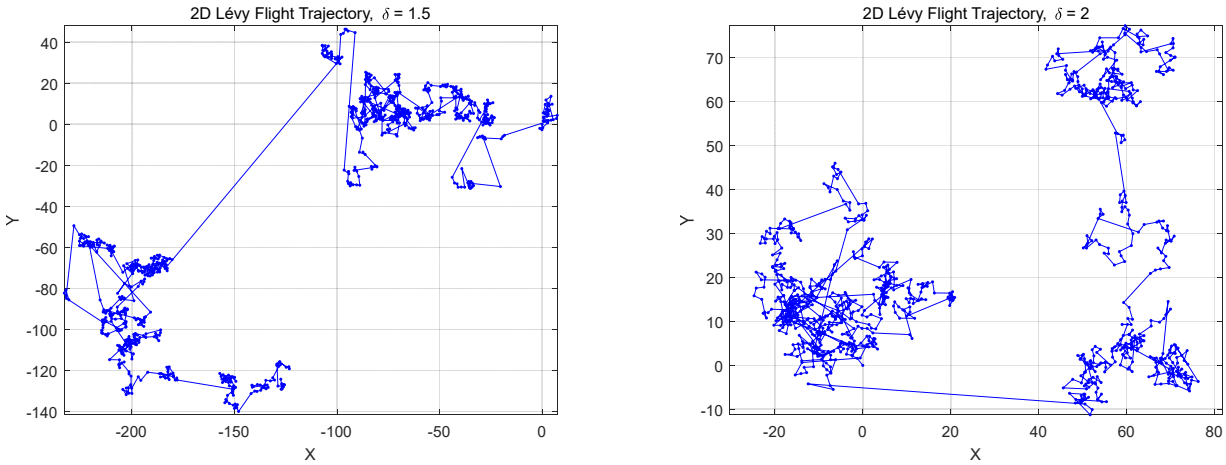
$$S = \frac{U}{|V|^{\frac{1}{\delta}}} \tag{2}$$

where U, V follow normal and Gamma distributions, respectively: $N(0, \sigma_U^2)$, $N(0, \sigma_V^2)$. Additionally, for $1 \leq \delta \leq 2$, Equation (3) holds:

$$\sigma_U = \left\{ \frac{\Gamma(1+\delta)\sin(\frac{\pi\delta}{2})}{\Gamma\left(\frac{1+\delta}{2}\right)\delta 2^{\frac{1}{\delta-1}}} \right\}^{\frac{1}{\delta}}, \sigma_V = 1 \tag{3}$$

where Γ is the Gamma distribution defined as follows: $\Gamma(k) = \int_0^\infty t^{k-1}e^{-t}\mathrm{dt}$.

In a two-dimensional space, the trajectory of a Lévy flight walking 1000 steps under different parameters is shown in Figure 1. Observing the image, it can be noticed that after multiple instances of clustering, Lévy flight exhibits significant leaps.



(**a**) (**b**)

Figure 1. Trajectory of Lévy flight: (**a**) $\delta = 1.5$; (**b**) $\delta = 2.0$.

In each iteration, the CS algorithm first employs a global search based on Lévy flight to generate new nest positions x_i^{t+1}, $i = 1, 2, \ldots, D$. The update formula for x_i^{t+1} can be formalized as follows:

$$\begin{aligned} x_i^{t+1} &= x_i^t + \alpha \otimes S \\ &= x_i^t + \alpha_0 \cdot S \otimes (x_i^t - x_{best}^t) \end{aligned} \tag{4}$$

where:

α—the step size factor;
α_0—the proportionality factor;
x_{best}^t—the best nest position at the *t-th* iteration;
$\otimes$—denotes element-wise multiplication.

The movement strategy of the CS algorithm can be intuitively observed from Figure 2.

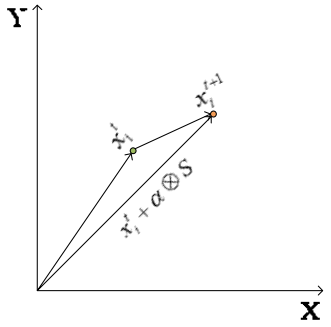


Figure 2. Global search strategy of the CS algorithm.

(3) Local Search Based on Preference Mechanism

In the strategy of random preference wandering, there is a certain probability that the cuckoo's parasitic eggs will be discovered by the host. Assuming the probability of the host discovering the cuckoo's eggs is p_a, let P_a be a D-dimensional vector, and its elements are all p_a. Once the host discovers that the egg does not belong to itself, the host nest with a high fitness value will be destroyed. The new host nest's position x_i^{t+1} $(i = 1, 2, \ldots, D)$ will be reconstructed based on the random mechanism in Equation (5).

$$x_i^{t+1} = x_i^t \otimes (1 - G(P_a - rand_2)) + v_i^t \otimes G(P_a - rand_2) \tag{5}$$

where:

$$v_i^t = x_i^t + rand_3(x_m^t - x_n^t) \tag{6}$$

$rand_2$ is a random vector whose all elements follow the distribution $U(0,1)$, $rand_3$ is a random number that follows $U(0,1)$, and x_m^t, x_n^t representing two randomly chosen and distinct host nests from the current population. Here, $G(x)$ is the Heaviside function:

$$G(x) = G(x_1, x_2, \ldots, x_k, \ldots, x_d) = [G_1, G_2, \ldots, G_k, \ldots, G_d] \tag{7}$$

where $G_k = \begin{cases} 1, x_k > 0 \\ 0, x_k \leq 0 \end{cases}$.

After completing the global or local random walk, the fitness function is determined as $f(x)$ and the greedy strategy is employed to decide whether the newly generated host nest should replace the corresponding old host nest or be retained. Following the update of host-nest positions according to Equations (4) or (5), the fitness values are calculated. Subsequently, the new and old fitness values are compared to determine which one has a superior fitness, thus establishing the new solution.

For minimization problems, the selection strategy is formalized as follows:

$$x_i^{t+1} = \begin{cases} x_i^{t+1}, f(x_i^{t+1}) < f(x_i^t) \\ x_i^t \quad , others \end{cases} \tag{8}$$

The bird nest where the host bird's egg is located serves as a solution, and the cuckoo searches for a suitable solution through Lévy flights. Once a suitable nest is found, the cuckoo removes the host bird's egg and lays its own egg in the host bird's nest. These suitable nests represent new solutions generated by the update formula of the CS. The search for the optimal solution mainly relies on the mechanisms of Lévy flights and random walks to explore new solutions. The process of using the algorithm to find the optimal solution involves continually replacing the previous inferior solution with a new one.

The CS algorithm is derived from idealized rules, and its implementation process is shown in Figure 3. Through the above steps, it can be observed that the CS has a relatively simple and straightforward approach. While the CS is widely used for optimization problems, it still faces challenges such as slow convergence speed, long simulation time, and a certain degree of dependence on randomness.

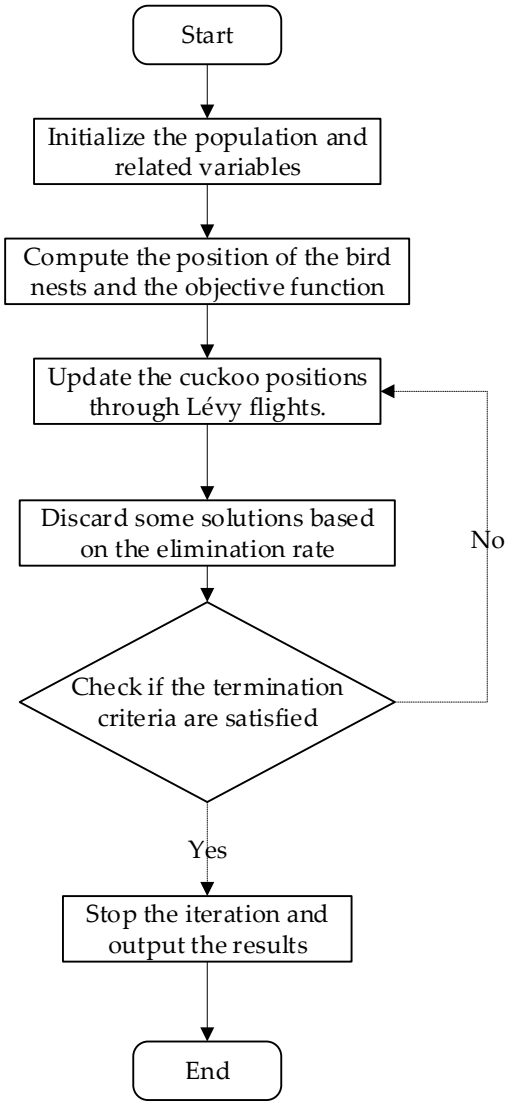


Figure 3. Cuckoo search algorithm process.

In intelligent optimization algorithms, the initialization of the population in the search space plays a crucial role in determining the optimization performance of the algorithm. The distribution of initial positions in the population determines the environmental adaptability of the initial population. The initial population should represent individuals from the entire space as much as possible. Therefore, the distribution of individuals in the initial population can have a certain impact on the optimization effect of the algorithm. Random distribution policy for initializing the population is uncontrollable, and its coverage in the space is uncertain. This approach has a certain probability of obtaining an initial population with better fitness. However, when the randomly acquired population clusters in a certain area in the exploration space, it is not conducive to global optimization. When dealing with complex systems, the initial solution set cannot be obtained randomly, nor can it traverse the entire space. If the generated initial solutions deviate significantly from the actual solutions, it may be impossible or difficult to find the actual solutions. In other words, to better solve practical problems, it is essential to have an initial set of solutions that represents the potential distribution of solutions in the entire search space.

The CS uses a random approach to initialize the cuckoo population in the initialization phase. This results in an uneven distribution of the initial cuckoo population, leading to low algorithm efficiency and hindering global search. Equations (4) and (5) indicate that in the CS algorithm, each cuckoo individual's search for a nest is guided only by the cuckoo

that found the best host nest. This undoubtedly leads to the underutilization of useful information carried by other cuckoo individuals. Due to the lack of effective information from previous iterations, there is a phenomenon of repeated searching in the next iteration. This not only causes resource waste but also reduces the algorithm's performance potential.

3. Improved Cuckoo Search Algorithm

3.1. Algorithm Design

(1) Initialization of the Population Based on the Best Point Set Method

The uniform distribution method can effectively characterize the characteristics of the solution space. Scholars have proposed methods to construct a uniform distribution. The best point set method [33] is a way to uniformly sample points in space. The initial solution set obtained using the best point set method is uniformly distributed and has good diversity. In a D-dimensional space, a best point set with d points is constructed as follows:

$$P_d(i) = \left\{ \left(\left\{ l_1^{(d)} \times i \right\}, \left\{ l_2^{(d)} \times i \right\}, \ldots, \left\{ l_D^{(d)} \times i \right\} \right), i = 1, 2, \ldots, d \right\} \tag{9}$$

Taking $l_i = 2\cos\left(\frac{2\pi i}{p}\right)(1 \le i \le d)$, p as the minimum prime number satisfying the conditions $\frac{p-3}{2} \ge D$. $P_d(i)$ is considered a best point set, and $l = \{l_i, i = 1, \ldots, d\}$ is a best point, which is the distribution of the improved initial cuckoo bird.

In a two-dimensional space, 200 points are randomly selected, and simultaneously, a best point set containing 200 points is constructed with values ranging $[0, 1]$. The distribution effects are shown in Figure 4. By comparing Figure 4a with Figure 4b, it is evident that the distribution obtained by random point selection is highly scattered, while the best point set method yields a more uniform distribution. As long as the number of points is fixed, the best point set method ensures a uniform distribution of points regardless of the dimensionality of the search space, thereby guaranteeing good diversity in the population.

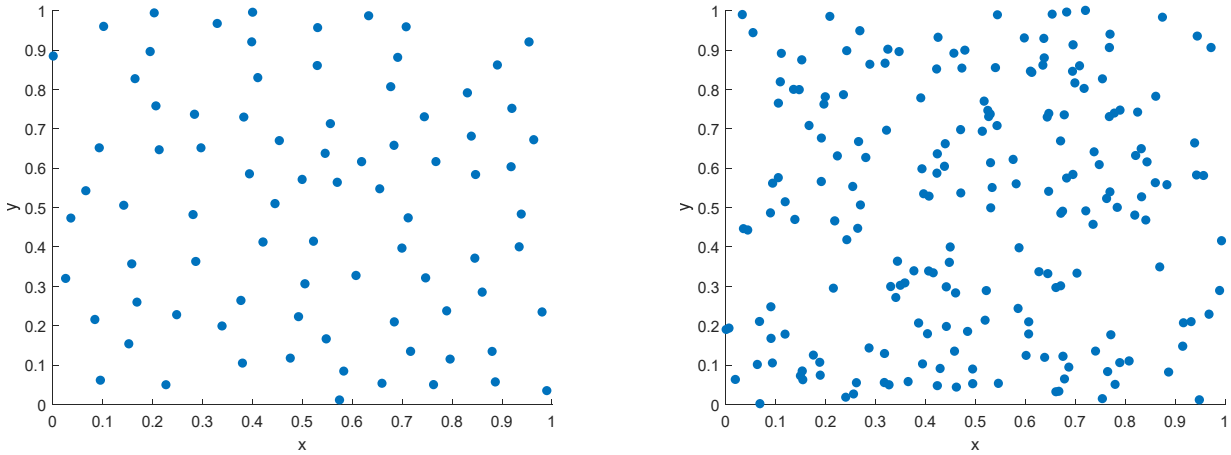


Figure 4. Comparison of the distribution in two-dimensional space between 200 best points and 200 random points: (**a**) two-dimensional best point set; (**b**) two-dimensional random point set.

To ensure the uniformity of the population, the best point set can be used instead of a random method for population initialization. Theoretically, applying the best point set mapping to the target solution space can effectively enhance the performance of the CS. The specific steps for initializing the operation using the best point set principle are as follows:

Step 1: Construct a best point set containing d points, where d is the population size;

Step 2: The initial position of the *i-th* cuckoo is given by $x_i^0 = (x_{i,1}^0, x_{i,2}^0, \ldots, x_{i,D}^0)$, $i = 1, \ldots, d$. Here, D represents the dimensionality of the search space.

Step 3: The specific calculation formula is:

$$x_{i,j}^0 = j_{low} + 2(j_{up} - j_{low}) \cos\left(\frac{2\pi j}{p}\right) \tag{10}$$

where j_{up} and j_{low} represent the upper and lower bounds of the *j-th* dimension in the search space, respectively.

(2) Shared Mechanism-Based Global Search Strategy

The generation of new solutions in the CS algorithm is primarily based on the information difference between individuals and their peers, leading to slow convergence in the later stages of the algorithm and a decrease in optimization accuracy. Information sharing among cuckoo individuals helps alleviate this problem, and previous research [34–37] has attempted to use information sharing to improve the performance of the CS algorithm, resulting in some new and improved CS algorithms. In order to further strengthen the information sharing among cuckoo individuals, an improved CS based on shared mechanism (ICSABOSM) is proposed, enhancing the collaboration within the population. The improved algorithm introduces a new iteration strategy to replace the CS algorithm's globally search strategy based on Levy flights, referred to in this paper as the shared mechanism-based global search strategy. The improved CS modifies the traditional operation mode, allowing the algorithm to focus more on exploration in the early stages of evolution and more on exploitation in the later stages.

The CS algorithm is sensitive to parameter settings, and the process of updating nest positions, as per Equations (2) and (4), depends on two parameters, α_0 and δ, with a broad range of possible values. Typically, in research studies, a fixed value is assigned to the step-size factor α_0 throughout the algorithm's stages. Parameter δ is a crucial factor in adjusting the convergence speed of the CS algorithm. When δ is fixed, it remains constant throughout the entire iteration process and cannot be changed. In cases where the iteration count is not sufficiently large, it can result in poor performance of the CS algorithm. To achieve the best global solution with an acceptable error, a large number of iterations is often required. To enhance the reliability and accuracy of the CS algorithm, many studies have introduced dynamic changes in the handling of parameter δ, transitioning from a constant to dynamic values. Numerous research works have demonstrated that dynamic variations in parameters can be controlled through linear or nonlinear functions [38]. Users can choose appropriate parameter values based on the specific problem they aim to solve, making the convergence results dependent on user choices. To overcome this limitation, an analysis is conducted based on the characteristics of parameter δ.

A notable characteristic of optimization algorithms is their reliance on sufficiently long moves in the initial iterations. If these steps progress towards the global optimum, the convergence speed of the algorithm will be enhanced. When the parameter δ linearly increases to 2, the search space gradually decreases. This indicates that if sufficiently long moves can be made in the initial iterations, the convergence speed of the algorithm will improve. At the same time, to avoid reaching a local optimum, it is necessary to reduce the movement space before the end of the iteration process.

Let t and T^{max} denote the current iteration number and the total number of iterations, respectively. Set

$$\delta = \frac{t}{T^{max}} \tag{11}$$

Control parameter δ in Equation (11) through Equation (12):

$$f_4(\delta) = random[f_1(\delta), f_2(\delta), f_3(\delta)] \tag{12}$$

where

$$f_1(\delta) = 1 + \sin(\pi\delta)$$

$$f_2(\delta) = \delta + 1$$
$$f_3(\delta) = e^{0.693\delta}$$

Combining the characteristics of the function, the advantages of using function $f_4(\delta)$ to calculate parameter δ can be analyzed. As shown in Figure 5, function $f_1(\delta)$ is a nonlinear convex function characterized by a significant increase in the value of $f_1(\delta)$ in the early iterations. This characteristic enhances the algorithm's ability to explore feasible regions in the search space. However, in cases of flight failure, it cannot obtain the global optimum. Adding the linear function E helps strike a balance between convergence speed and precision, but when the convergence speed is too fast, it may fall into a local optimum. Therefore, adding the concave function $f_2(\delta)$ controls the convergence speed. Function $f_3(\delta)$ has a slower convergence speed than functions $f_1(\delta)$ and $f_2(\delta)$. Thus, according to function $f_3(\delta)$, to achieve the desired precision, the algorithm requires more iterations. Function $f_4(\delta)$ is used to control the step size S. Compared to any fixed value δ, it allows for sufficiently long moves in the initial iterations and short moves in the final iterations. Based on the advantages of the new vector step size S controlled by function $f_4(\delta)$, a new step size parameter S^* is proposed.

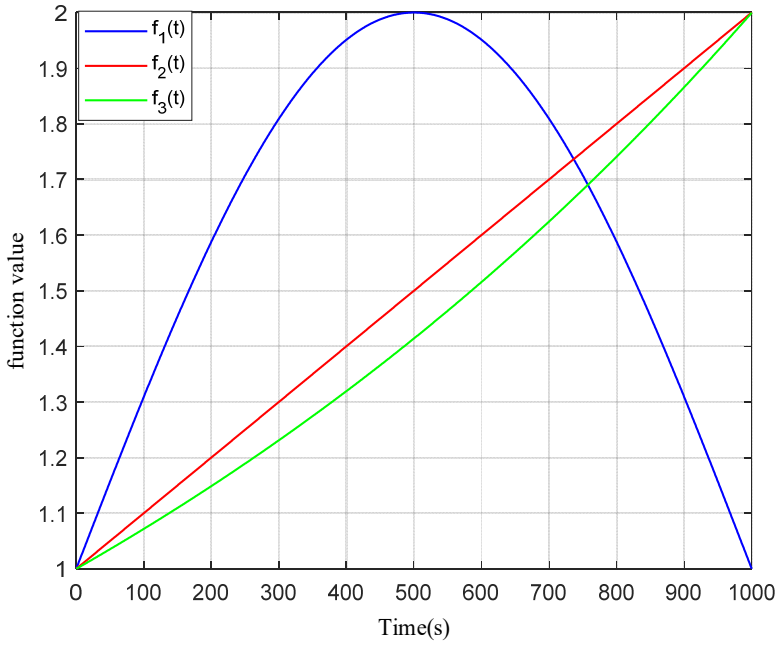


Figure 5. Partial control function graphs.

The cuckoo seeks the parasitic bird's nest that is most suitable for egg-laying to maximize the survival rate of its eggs. In the CS algorithm, finding the host nest relies on completely random movements because of the lack of good information about the optimal parasitism. Therefore, multiple attempts are needed to find the optimal parasitic bird's nest, meaning several iterations are required to achieve global optimality. Addressing this drawback of the CS algorithm, this paper introduces a new parameter, the feasible sharing area, established based on information sharing among cuckoos in the population.

Similar to the population structure in the grey wolf optimization algorithm, the population structure of the improved CS is defined as k_{best}, calculated by Equation (13):

$$k_{best} = d - (d-3)\sqrt{\frac{t}{T^{max}}} \tag{13}$$

As the number of iterations increases, k_{best} shows a decreasing trend. Moreover, when $t = T^{max}$, it holds that $k_{best} = 3$.

To improve the success rate, a comparison is made among the cuckoo birds in the population to find all the nests. The fitness values of all the current positions of host nests are calculated, and after sorting, the position information of the excellent parasitic nests is recorded. The top k_{best} nests in terms of fitness values are considered feasible nests. As per the definition of feasible nests, these nests contain host nests in the search space with better properties. Based on feasible nests, a feasible sharing area $Feas_{best}$ is established. $Feas_{best}$ is calculated using Equation (14), representing the average position of nests ranked in the top k_{best} in terms of fitness values.

$$Feas_{best} = \frac{\sum_{i=1}^{k_{best}} X_i}{k_{best}} \tag{14}$$

where

k_{best} is calculated by Equation (13);
X_i is the position of the cuckoo ranked *i-th* in fitness.

Clearly, after each iteration, $Feas_{best}$ will be updated. Let the host nests in the feasible sharing area take on the role of new leaders. In each iteration, guided by $Feas_{best}$ as the leader, each cuckoo bird's movement is directed, and the new direction always aligns with the movement trend of the cuckoo bird.

In the improved algorithm, a new step size is established based on $Feas_{best}$:

$$S^* = S + \alpha_0(\alpha_0 Feas_{best} - S) \tag{15}$$

In the equation, S is defined consistently with the step size in the CS algorithm, δ is controlled by the function $f_4(\delta)$, updating the step size S in the CS algorithm to maintain a certain level of randomness in the defined step size. The dominant individuals included in $Feas_{best}$ increase the likelihood of the cuckoo birds finding better solutions. Additionally, based on the information contained in $Feas_{best}$, the optimal movement direction for each cuckoo bird can be determined. Such movement directions reduce the number of iterations, thereby improving the convergence speed of the algorithm.

The global search strategy based on the sharing mechanism can be explained using Equation (16):

$$x_i^{t+1} = x_i^t + \alpha_0 \cdot S^* \otimes \left(x_{best}^t - x_i^t\right) \tag{16}$$

From Equations (15) and (16), it can be observed that in the new global exploration strategy based on the sharing mechanism, both the best information from the population and useful information from other cuckoo individuals guide the search simultaneously. This approach not only relies on the information from the best cuckoo individual but also helps the algorithm maintain population diversity. Additionally, using the new step-size operator instead of the single operator based on the Lévy flight further enhances the exploration capability.

(3) Local Search Strategy Based on Sharing Mechanism

After global search completion, to enhance the ability of information sharing during local search, one cuckoo is randomly selected from set $Feas_{best}$ to share information with the *i-th* cuckoo, denoted as the q_i-*th* cuckoo. The current fitness values of the nests found by these two cuckoos are compared, and the nest position with the smaller fitness value is chosen for updating in set $x_{q_i}^t$. The updating process is illustrated in Figure 6.

Hosts with probability p_a of finding alien bird eggs with high fitness values will have their nests destroyed. To ensure that cuckoos stay away from abandoned nests, more cuckoos need to provide location information. Four cuckoos with distinct positions are randomly selected from the current population, and their positions are denoted as x_m^t, x_n^t, x_p^t, x_q^t, respectively. The new nest position x_i^{t+1}, $i = 1, 2, \ldots, D$ is reconstructed based on the shared information from multiple cuckoos. The double difference Formula (17)

is used to update v_i^t in Formula (5), resulting in a local search strategy based on the sharing mechanism:

$$v_i^{*t} = rand_5 \otimes x_{q_i}^t + (1 - rand_5) \otimes x_{best} + rand_6 \cdot (x_m^t - x_n^t) + rand_4 \cdot (x_p^t - x_q^t) \quad (17)$$

where $rand_5$, $rand_6$, $rand_4$ is a random number generated by $U(0,1)$.

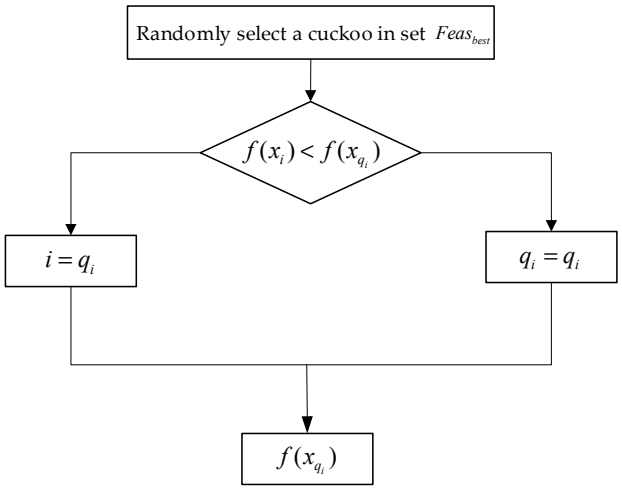


Figure 6. Selecting the cuckoo to share information with the *i-th* cuckoo.

From Equations (6) and (17), double -difference vectors have taken the place of a single-difference vector used in the original CS. Double-difference vectors have taken the place of single-difference vectors used in the original CS. This contributes to the reinforcement. From Equation (17) described above, it can be seen that the modified operation mode offers more flexibility for the search process, since this new operation mode enables the population at the early and later evolution stages to still have the opportunity of exploiting the promising regions that have been located already and exploring new regions that have not been visited in the search space, respectively.

3.2. Algorithm Flow

The main steps of the improved CS are as follows, and the algorithm flowchart is shown in Figure 7.

(1) Set the population size d and other variables T^{max}, P_a, and the fitness function to be optimized as $F(x)$. Assume the position of the bird nest found by the *i-th* cuckoo is: $x_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t)), i = 1, 2, \ldots, D$. Initialize the population using the optimal points set method: $x_i(i = 1, 2, \ldots, d)$;
(2) Calculate the fitness value $F(x_i)$ of the initial bird nest position. Through comparison, designate the minimum fitness value in the current individuals as F_{best}, and record the corresponding best position as x_{best};
(3) Search and update $Feas_{best}$ using Equations (13) and (14);
(4) Update the position of each cuckoo bird using Equations (15) and (16), denoted as $\hat{x}_i(t)$ and calculate the fitness value of the new position as $F(\hat{x}_i(t))$.
(5) If $F(\hat{x}_i(t)) < F(x_i(t))$, then update $x_i(t)$ to $\hat{x}_i(t)$, and correspondingly update $F(x_i(t))$ to $F(\hat{x}_i(t))$. If $F(\hat{x}_i(t)) \geq F(x_i(t))$, then keep them unchanged.
(6) Generate a random number $\theta \in [0, 1]$. If $\theta > P_a$, then eliminate solution $\hat{x}_i(t)$. Update the eliminated solution according to Equations (5) and (17), then update its fitness.
(7) Sort the fitness of all cuckoo birds, obtaining the current best position x_{best}^{t+1} and the best value $F(x_{best}^{t+1})$.
(8) Check the termination criteria. If satisfied, output the optimal solution; otherwise, iterate back to step (3).

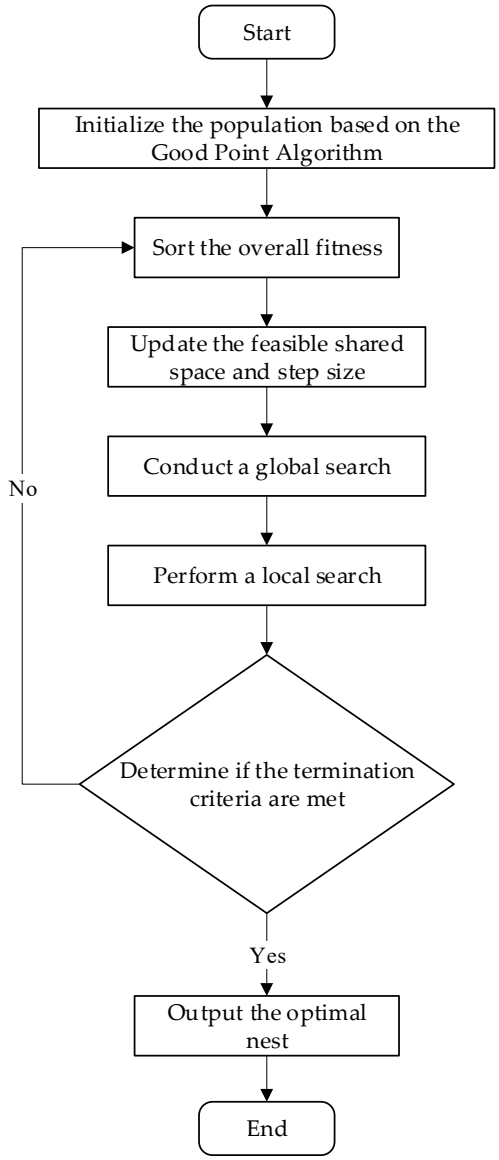


Figure 7. Improved algorithm flowchart.

3.3. Improved Algorithm Performance Test

3.3.1. Experimental Design

To evaluate the performance of the proposed improved CS, we compared it with the original CS on global optimization problems. We conducted a comprehensive experimental assessment using four well-known standard test functions, including two low-dimensional test functions and two high-dimensional test functions. The following are the four test functions [39] along with the optimal value $f(x^*)$ of the functions in their defined domains.

(1) $f_1 = 14.203125 + x_1(3x_2 - 4.5x_2^2 - 5.25x_2^3 - 3) + x_1^2(x_2 + x_2^2 + x_2^4 + x_2^6 + 1)$

Whereas $x_1, x_2 \in [-100, 100]$, in function f_1, the global minimum point is at $x^* = (3, 0.5)$, and it has a global minimum value of $f_1(x^*) = 0$. As shown in Figure 8a, the terrain of the function is relatively flat.

(2) $f_2 = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{1 + 0.002\left(x_1^2 - x_2^2\right) + \left(x_1^2 - x_2^2\right)^2}$

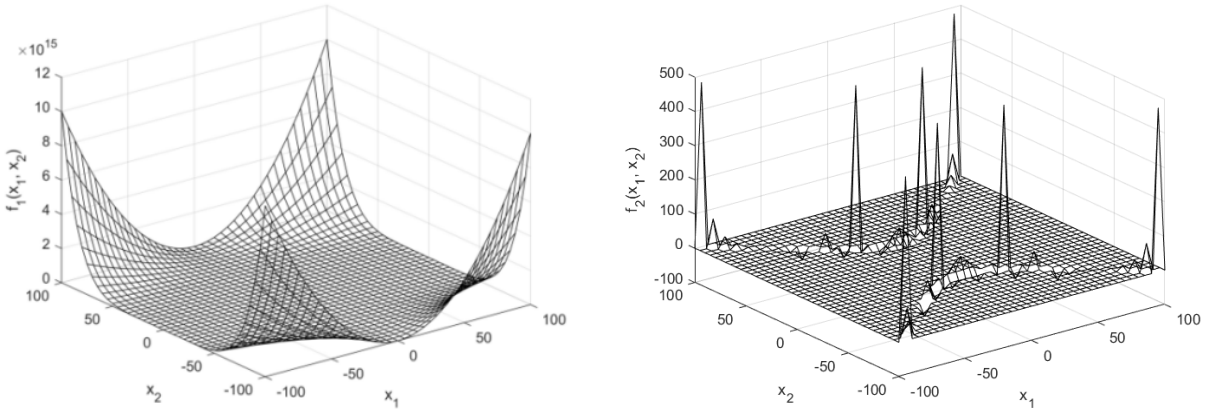
Whereas $x_1, x_2 \in [-100, 100]$, the global minimum point for function f_2 is at $x^* = (0, 0)$, and it has a global minimum value of $f_2(x^*) = 0$. As shown in Figure 8b, the function exhibits strong oscillations within the given domain.

(3) $f_3(x) = \sum_{i=1}^{15} |x_i| + \prod_{i=1}^{15} |x_i|$

Whereas $x_i \in [-10, 10]$, in function f_3, the global minimum point is $x^* = (0, 0, \ldots, 0)$, and it has a global minimum value of $f_3(x^*) = 0$. Typically, finding the global minimum point of this function is somewhat challenging.

(4) $f_4(x) = \sum_{i=1}^{15} \left[x_i^2 - 10(\cos(2\pi x_i) + 1)\right]$

Whereas $x_i \in [-5.2, 5.2]$, in function f_4, the global minimum point is $x^* = (0, 0, \ldots, 0)$, and it has a global minimum value of $f_4(x^*) = 0$. The function exhibits peaks and valleys in its overall shape.



(**a**) (**b**)

Figure 8. Partial schematic diagrams of standard test functions in three dimensions: (**a**) three-dimensional schematic diagram of function f_1; (**b**) three-dimensional schematic diagram of function f_2.

3.3.2. Comparative Analysis of Algorithms

The optimal solution often requires multiple executions of the algorithm to achieve it. In the experiment, the algorithm was set to run independently 20 times, with a maximum iteration count of 2500. Under the same parameter settings, both the CS and the improved CS were tested. The population size was set to 30, discovery probabilities were set to 0.75 and 0.25, and the step-size control factor was set to 0.1 for two sets of experiments. The data obtained from running the algorithm independently 20 times were organized, and the best, worst, and average values were recorded as shown in Tables 1 and 2.

Table 1. Algorithm test results with a discovery probability of 0.75.

Function	Algorithm	Best Value	Worst Value	Average Value
f_1	ICSABOSM	7.5378×10^{-39}	3.4674×10^{-32}	4.5386×10^{-36}
	CS	4.5726×10^{-31}	2.6935×10^{-25}	6.5320×10^{-29}
f_2	ICSABOSM	0	7.0054×10^{-35}	5.3022×10^{-40}
	CS	0	6.2378×10^{-26}	4.2057×10^{-32}
f_3	ICSABOSM	4.2648×10^{-31}	7.2538×10^{-23}	8.5478×10^{-28}
	CS	7.4875×10^{-22}	4.5902×10^{-15}	4.7326×10^{-20}
f_4	ICSABOSM	2.0018×10^{-15}	1.4608×10^{-8}	9.4010×10^{-11}
	CS	1.2450×10^{-12}	3.1634×10^{-3}	3.7025×10^{-6}

Table 2. Algorithm test results with discovery probability of 0.25.

Function	Algorithm	Best Value	Worst Value	Average Value
f_1	ICSABOSM	1.4473×10^{-33}	5.5632×10^{-28}	1.9824×10^{-31}
	CS	6.3557×10^{-27}	8.4367×10^{-23}	5.2564×10^{-25}
f_2	ICSABOSM	0	6.5837×10^{-17}	5.3704×10^{-30}
	CS	8.3642×10^{-17}	2.4579×10^{-11}	5.3578×10^{-15}
f_3	ICSABOSM	4.8346×10^{-25}	3.6849×10^{-13}	7.2841×10^{-21}
	CS	7.3648×10^{-14}	4.3574×10^{-9}	9.3572×10^{-13}
f_4	ICSABOSM	4.2602×10^{-10}	9.4738×10^{-5}	5.6173×10^{-7}
	CS	7.6469×10^{-5}	4.6328×10^{-2}	3.7468×10^{-3}

Comparing the contents of Tables 1 and 2, it can be observed that the improved CS with a discovery probability of 0.25 outperforms the CS in three evaluation metrics. Therefore, in terms of search accuracy and stability, the improved CS does show improvement compared to the original CS. These standard test functions exhibit complex properties and numerous local optima. Typically, optimization in high-dimensional spaces poses certain challenges as position updates cannot guarantee flexible movement. The experimental results for standard test functions indicate that, based on the newly proposed parameters, ICSABOSM's position updates are more intelligent and flexible than those of the CS. The improved algorithm demonstrates a certain superiority, as it can perform sufficiently distant movements in the initial iterations, thereby expanding the search space.

3.3.3. Comparative Analysis of Algorithm Convergence

To more intuitively demonstrate the optimization performance of the ICSABOSM algorithm, it was compared with other modified CS algorithms and several commonly used intelligent optimization algorithms in engineering. This includes the classical CS, the firefly algorithm (FA), and particle swarm optimization (PSO). Additionally, a hybrid of the cuckoo search and firefly algorithms (CS-FA) was also included in the test. For function f_1, f_2, the iteration count was set to $T_{\max} = 1000$, $d_1 = 2$. The parameter settings for the algorithms involved in the test are shown in Table 3. For function f_3, f_4, the iteration count was set to $T_{\max} = 8000$, $d_1 = 13$.

Table 3. Parameter settings.

ICSABOSM Parameter Settings								
n	α	—	—	—	—	P_a	$T_{\max}$	d
20	0.01	—	—	—	—	0.75	1000/8000	2/13
CS Parameter Settings								
n	α	—	—	—	—	P_a	$T_{\max}$	d
20	0.01	—	—	—	—	0.75	1000/8000	2/13
CS-FA Parameter Settings								
n	α	η_0	τ	η	$\vec{\varepsilon}_1$	P_a	$T_{\max}$	d
20	0.01	1.0	1.0	1.5	0.5	0.75	1000/8000	2/13
FA Parameter Settings								
n	α	η_0	τ	—	—	—	$T_{\max}$	d
20	0.5	0.2	1.0	—	—	—	1000/8000	2/13
PSO Parameter Settings								
n	ω	c_1	c_2	—	—	—	$T_{\max}$	d
20	0.9	2.0	2.0	—	—	—	1000/8000	2/13

Figure 9a–d shows the convergence curves of five algorithms regarding the test functions. The horizontal axis represents the number of iterations, and the vertical axis represents the logarithmic value of the best fitness.

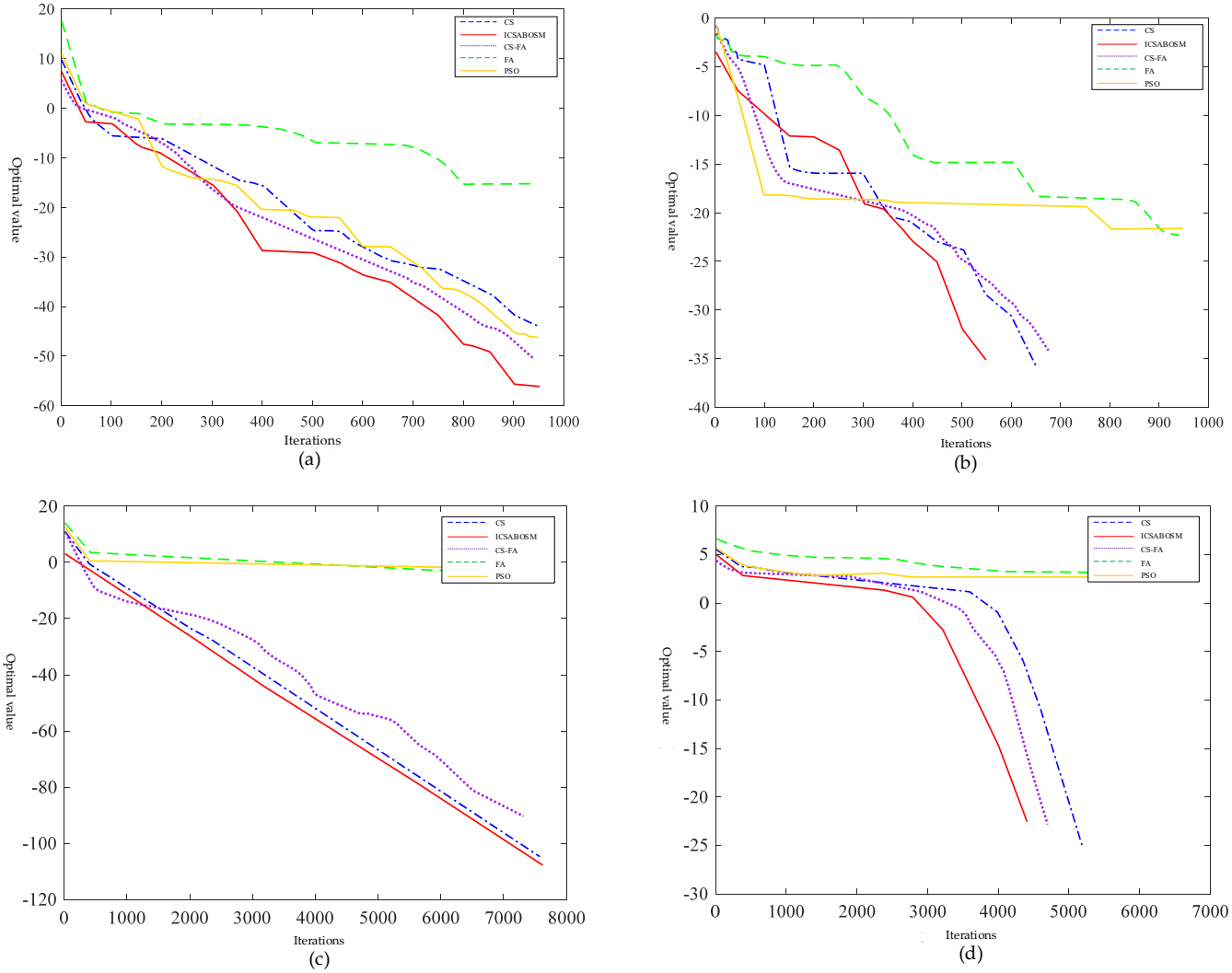


Figure 9. Convergence curve graph: (**a**) the convergence curves of function f_1; (**b**) the convergence curves of function f_2; (**c**) the convergence curves of function f_3; (**d**) the convergence curves of function f_4.

Function f_1: Multimodal function, characterized by a relatively flat fitness landscape. Traditional swarm intelligence algorithms and CS successfully find the global optimum in solving this problem, demonstrating good solutions. This indicates that the ICSABOSM exhibits commendable characteristics.

Function f_2: A multimodal function with innumerable local minima within a given feasible domain, marked by strong oscillations, making the search for the global minimum challenging. Traditional swarm intelligence algorithms show lower precision in optimization, while the classic CS, CS-FA, and other improved CS algorithms all successfully identify the global optimum.

Function f_3: A difficult-to-optimize unimodal function. For this function, all algorithms did not converge to the global minimum within 8000 iterations. However, the CS, CS-FA, and the improved algorithm ICSABOSM achieved significantly better optimal values compared to traditional swarm intelligence algorithms.

Function f_4: A typical nonlinear multimodal function, with peaks varying greatly in height, rendering the search for its global optimum exceedingly difficult. Experimental results indicate that the PSO and FA became trapped in local optima, highlighting the strong local search capability of the ICSABOSM.

The ICSABOSM algorithm demonstrates a significant advantage in convergence accuracy and can converge rapidly to the optimal value or its vicinity within a small number of iterations.

4. Application of the Improved Algorithm in Differential Equations

Optimization problems can be formulated as the minimization or maximization of the objective function under variable constraints. The general form of a constrained minimization optimization problem is:

$$\begin{aligned} &\min \ F(T) \\ &s.t. \ \ h_i(T) = 0, \ \ i = 1, 2, \ldots, n_1 \\ &\qquad \mathrm{g}_j(T) \leq 0, \ \ j = 1, \ldots, n_2 \end{aligned} \tag{18}$$

where $\Omega = \{T \in R^n\}$ is the feasible solution space, T is the decision variables, $F(T)$ is the objective function, $h_i(T) = 0$ represents equality constraints, and $\mathrm{g}_j(T) \leq 0$ represents inequality constraints.

By transforming a system of differential equations into an optimization problem, the solution of the differential equations can be tackled using existing optimization algorithms. The fundamental idea behind this approach is to treat the unknown functions in the system of differential equations as optimization variables, and the residuals of the differential equations serve as the cost function. The solution to the system of differential equations is obtained by minimizing this cost function.

4.1. Construction of Approximate Solutions

Optimization is a mathematical method widely applied in various fields such as science, engineering, and business. Its purpose is to find the optimal solution within given constraints to meet specific requirements and objectives. In this process, the selection and adjustment of input variables play a crucial role in influencing the final results. The core of optimization methods lies in the cost function, objective function, or fitness function. These functions are used to measure the relationship between input variables and output results. By optimizing these functions, the optimal solution can be found.

Assuming the general form of a system of differential equations defined on the interval $[t_0, t_n]$ can be described as follows:

$$\begin{aligned} &F_1(t, X_1, \ldots, {X_1}^{(n)}, X_2, \ldots, {X_2}^{(n)}, \ldots, X_n, \ldots, {X_n}^{(n)}) = 0 \\ &F_2(t, X_1, \ldots, {X_1}^{(n)}, X_2, \ldots, {X_2}^{(n)}, \ldots, X_n, \ldots, {X_n}^{(n)}) = 0 \\ &\qquad\qquad \vdots \\ &F_n(t, X_1, \ldots, {X_1}^{(n)}, X_2, \ldots, {X_2}^{(n)}, \ldots, X_n, \ldots, {X_n}^{(n)}) = 0 \end{aligned} \tag{19}$$

The boundary value conditions are:

$$\begin{cases} X_1(t_0) = X_{1,0} \\ X_2(t_0) = X_{2,0} \\ \quad \vdots \\ X_n(t_0) = X_{n,0} \end{cases}, \begin{cases} X_1'(t_0) = X_{1,0}' \\ X_2'(t_0) = X_{2,0}' \\ \quad \vdots \\ X_n'(t_0) = X_{n,0}' \end{cases}, \cdots \begin{cases} X_1(t_n) = X_{1,n} \\ X_2(t_n) = X_{2,n} \\ \quad \vdots \\ X_n(t_n) = X_{n,n} \end{cases}, \begin{cases} X_1'(t_n) = X_{1,n}' \\ X_2'(t_n) = X_{2,n}' \\ \quad \vdots \\ X_n'(t_n) = X_{n,n}' \end{cases} \tag{20}$$

Or the initial value conditions are:

$$\begin{cases} X_1(t_0) = X_{1,0} \\ X_2(t_0) = X_{2,0} \\ \quad\vdots \\ X_n(t_0) = X_{n,0} \end{cases}, \begin{cases} X_1'(t_0) = X_{1,0}' \\ X_2'(t_0) = X_{2,0}' \\ \quad\vdots \\ X_n'(t_0) = X_{n,0}' \end{cases}, \cdots, \begin{cases} X_1^{(n-1)}(t_0) = X_{1,0}^{(n-1)} \\ X_2^{(n-1)}(t_0) = X_{2,0}^{(n-1)} \\ \quad\vdots \\ X_n^{(n-1)}(t_0) = X_{n,0}^{(n-1)} \end{cases} \tag{21}$$

By constructing an approximate solution to the differential equation, the original problem can be transformed into a constrained optimization problem. Consider an n-th order initial-boundary value problem for a differential equation:

$$F(t, x, x', \ldots, x^{(n)}) = 0 \tag{22}$$

with boundary conditions:

$$x(t_0) = x_0, x(t_n) = x_n, \ldots, x'(t_0) = x_0', \ldots, x'(t_n) = x_n' \tag{23}$$

or the initial value conditions of:

$$x(t_0) = x_0, x'(t_0) = x_0', \ldots, x^{(n-1)}(t_0) = x_0^{(n-1)} \tag{24}$$

At this point, solving the above Equation (19) can be addressed by utilizing the improved CS to find an approximate solution. For a general continuous function, it is known from the Fourier series convergence theorem that the convergence of the Fourier series is satisfied. In other words, a continuous function can be expressed in the form of a Fourier series expansion. As is well known, the sine and cosine functions can be infinitely differentiated, and it holds that:

$$\sin^{(n)}(x) = \sin(x + \frac{n\pi}{2})$$

The approximate solution to high-order differential equations can be obtained by solving them using a finite number of terms in the Fourier series. Moreover, during the differentiation process, the derivatives of the function are not reduced, and an improved algorithm is employed to find the optimal values for these coefficients.

Construct a Fourier series expansion centered around $\bar{t}$ as follows:

$$x(t) \approx \widetilde{x}(t) = a_0 + \sum_{j=1}^{M} [a_j \cos(\frac{j\pi(t-\bar{t})}{L}) + b_j \sin(\frac{j\pi(t-\bar{t})}{L})] \tag{25}$$

Compute the derivative as follows:

$$\begin{aligned} x'(t) &\approx \widetilde{x}'(t) = \sum_{j=1}^{M} \frac{j\pi}{L} [a_j \cos(\frac{j\pi(t-\bar{t})}{L} + \frac{\pi}{2}) + b_j \sin(\frac{j\pi(t-\bar{t})}{L} + \frac{\pi}{2})] \\ x''(t) &\approx \widetilde{x}''(t) = \sum_{j=1}^{M} (\frac{j\pi}{L})^2 [a_j \cos(\frac{j\pi(t-\bar{t})}{L} + \pi) + b_j \sin(\frac{j\pi(t-\bar{t})}{L} + \pi)] \\ &\vdots \\ x^{(n)}(t) &\approx \widetilde{x}^{(n)}(t) = \sum_{j=1}^{M} (\frac{j\pi}{L})^n [a_j \cos(\frac{j\pi(t-\bar{t})}{L} + \frac{n\pi}{2}) + b_j \sin(\frac{j\pi(t-\bar{t})}{L} + \frac{n\pi}{2})] \end{aligned} \tag{26}$$

where $L = t_n - t_0$ is the interval length, M is the number of terms for sin and cos. To find the approximate solution to the differential equation, it is only necessary to determine the unknown coefficients $a_j,\ b_j$ in Equation (25).

Based on the above discussion, for a general system of differential equations, the approximate solution can be constructed in the following form:

$$\begin{aligned} X_1(t) \approx \widetilde{X}_1(t) &= a_0^1 + \sum_{j=1}^{M} [a_j^1 \cos(\tfrac{t-\bar{t}}{L} j\pi) + b_j^1 \sin(\tfrac{t-\bar{t}}{L} j\pi)] \\ X_2(t) \approx \widetilde{X}_2(t) &= a_0^2 + \sum_{j=1}^{M} [a_j^2 \cos(\tfrac{t-\bar{t}}{L} j\pi) + b_j^2 \sin(\tfrac{t-\bar{t}}{L} j\pi)] \\ &\vdots \\ X_n(t) \approx \widetilde{X}_n(t) &= a_0^n + \sum_{j=1}^{M} [a_j^n \cos(\tfrac{t-\bar{t}}{L} j\pi) + b_j^n \sin(\tfrac{t-\bar{t}}{L} j\pi)] \end{aligned} \tag{27}$$

For the derivative $\widetilde{X}_1(t)$, $\widetilde{X}_2(t), \ldots,$ $\widetilde{X}_n(t)$ of the approximate solution $\widetilde{X}_1(t)$, $\widetilde{X}_2(t), \ldots,$ $\widetilde{X}_n(t)$, it is only necessary to calculate according to Equation (26).

4.2. Constraint Conditions

When using differential equations to construct optimization problems, the found solution must satisfy two conditions. First, this solution must comply with the differential Equation (19). Secondly, this solution must also meet the requirements of the optimization problem, i.e., it needs to satisfy its initial or boundary conditions. To achieve this, it is necessary to transform the forms of the homogeneous and non-homogeneous boundary or initial conditions, establishing the forms as (28) or (29), respectively:

$$\begin{cases} x(t_0) = 0 \\ x'(t_0) = 0 \\ x''(t_0) = 0 \\ \vdots \\ x^{(n)}(t_0) = 0 \end{cases} \Rightarrow \begin{cases} h_0(t_0) = |x(t_0)| = |\widetilde{x}(t_0)| \\ h_1(t_0) = |x'(t_0)| = |\widetilde{x}'(t_0)| \\ h_2(t_0) = |x''(t_0)| = |\widetilde{x}''(t_0)| \\ \vdots \\ h_n(t_0) = \left|x^{(n)}(t_0)\right| = \left|\widetilde{x}^{(n)}(t_0)\right| \end{cases} \tag{28}$$

$$\begin{cases} x(t_0) = x_0 \\ x'(t_0) = {x'}_0 \\ x''(t_0) = {x''}_0 \\ \vdots \\ x^{(n)}(t_0) = x_0^{(n)} \end{cases} \Rightarrow \begin{cases} h_0(t_0) = \left|\frac{x(t_0)}{x_0} - 1\right| \approx \left|\frac{|\widetilde{x}(t_0)|}{x_0} - 1\right| \\ h_1(t_0) = \left|\frac{x'(t_0)}{{x'}_0} - 1\right| \approx \left|\frac{|\widetilde{x}'(t_0)|}{{x'}_0} - 1\right| \\ h_2(t_0) = \left|\frac{x''(t_0)}{{x''}_0} - 1\right| \approx \left|\frac{|\widetilde{x}''(t_0)|}{{x''}_0} - 1\right| \\ \vdots \\ h_n(t_0) = \left|\frac{x^{(n)}(t_0)}{x_0^{(n)}} - 1\right| \approx \left|\frac{|\widetilde{x}^{(n)}(t_0)|}{x_0^{(n)}} - 1\right| \end{cases} \tag{29}$$

where

$h_1, h_2, \ldots, h_n$ represent the constraints of optimization problems.

4.3. Objective Function and Fitness Function

Replace x, $x'(t)$, $x''(t), \ldots,$ $x^{(n)}(t)$ in the differential Equation (19) with the constructed approximate solution $\widetilde{x}(t)$ and its derivative $\widetilde{x}'(t)$, $\widetilde{x}''(t), \ldots,$ $\widetilde{x}^{(n)}(t)$, respectively, to obtain the residual:

$$R(t) = F(t, \widetilde{x}, \widetilde{x}', \widetilde{x}'', \ldots, \widetilde{x}^{(n)}).$$

Choosing an appropriate evaluation function to test the accuracy of the approximate solution, the weighted residual function is adopted as the evaluation criterion for the approximate solution, in the form of:

$$W = \int_{t_0}^{t_n} |W(t)||R(t)|dt$$

where $W(t)$ is the weight function. A smaller W value is preferable, indicating higher accuracy in the established approximate solution. Perform numerical calculations on the integration using trapezoidal or Simpson integration methods.

For the standard form of the penalty function, with a penalty factor of 1, the penalty function is given by:

$$P = \sum_{j=1}^{m_1+m_2} h_j$$

where

h_j is the constraint condition;
m_1 is the number of boundary value conditions;
m_2 is the number of initial value conditions.

Build an appropriate fitness function to evaluate the quality of individual positions and find the optimal solution through iterative loops.

Add the penalty function value to both the residual function and the mean square error function to obtain the fitness function F:

$$F = W + P$$

Transform the problem of solving definite solutions to ordinary differential equations into a minimization problem of residual functions in the feasible domain Ω, with boundary or initial values as its constraint conditions, which will transform the solved differential equation problem into a optimization problem in the following form:

$$\min \quad F$$

In the context of mechanical engineering and other fields, this approach offers significant potential. Mechanical engineering often involves complex systems governed by differential equations, such as dynamics, fluid mechanics, and thermal systems. By applying this improved algorithm, engineers and researchers can obtain more accurate and efficient solutions to these equations, leading to better modeling, analysis, and design of mechanical systems.

4.4. Algorithm Procedure for Problem Solving

The specific steps for applying the improved CS to solve differential equations are as follows:

(1) Express the differential equation in the implicit function form on the solution interval $[t_0, t_n]$, as in Equation (19): $F(t, x, x', \ldots, x^{(n)}) = 0$
(2) Transform the boundary conditions or initial value conditions into constraint forms (28) or (29);
(3) Based on Equation (25), select an appropriate number M of terms in the Fourier series expansion;
(4) Assign values to each undetermined coefficient $a_0, a_1, a_2, \ldots, a_M, b_1, \ldots, b_M$ in the approximate function and introduce them into the ICSABOSM algorithm;
(5) Call the ICSABOSM algorithm to search for the undetermined coefficients.
(6) Calculate the values of the approximate solution $\widetilde{x}(t)$ at various points with Δt:$t_j = t_0 + j\Delta t$, $x(t_j) \approx \widetilde{x}(t_j)$ as the step size
(7) Calculate the approximate value of the derivative at point t_j using Equation (26): $\widetilde{x}'(t_j), \widetilde{x}''(t_j), \ldots, \widetilde{x}^{(n)}(t_j)$
(8) Construct the residual function $R(t)$;
(9) Choose an appropriate fitness function based on the target function equation and calculate the fitness function value for each cuckoo's current position;
(10) Repeat steps (6) to (10) until the stopping criteria of the ICSABOSM algorithm are met.

4.5. Numerical Examples and Results Analysis

4.5.1. First-Order Linear Differential Equation

Solving the first-order linear differential equation with initial condition: $y(0) = 1$:

$$\frac{d}{dt}y(t) + 2y(t) = \sin(t), \quad t \in [0, 3]$$

The exact solution of this equation is $y(t) = \frac{2}{5}\sin(t) - \frac{1}{5}\cos(t) + \frac{6}{5}e^{-2t}$.

The search space of the improved CS is influenced by the population size, resulting in noticeable differences in optimization results. When the population size is small, individuals lack diversity and are prone to converge to local optimal values. Conversely, when the population size is large, the search space increases, making it easier to obtain global optimal solutions but at the cost of increased time complexity. Population size, the number of iterations, and constraints all impact the optimization results. Based on the algorithm workflow and multiple experiments, the algorithm's population size was set to 400, and the number of iterations was set to 10,000. For the above first-order linear differential equation, we used the least squares basis functions and the Fourier series to construct the approximate functions of the equation.

Figure 10 presents a comparative analysis between the numerical and analytical solutions of a first-order linear differential equation, and Figure 11 showcasing the deviations between them. For first-order linear differential equations, the enhanced algorithm significantly improves the accuracy of the exact solutions. This is evidenced by the reduced mean squared error and absolute error between the numerical and analytical solutions. Furthermore, these findings underscore the precision of the algorithm in solving boundary value problems for higher-order differential equations.

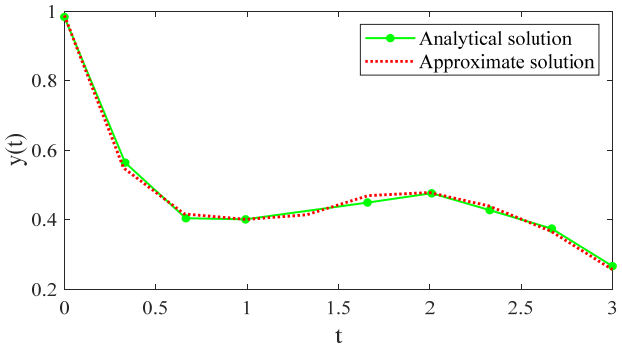


Figure 10. Comparison of numerical solution and analytical solution for first-order linear differential equation.

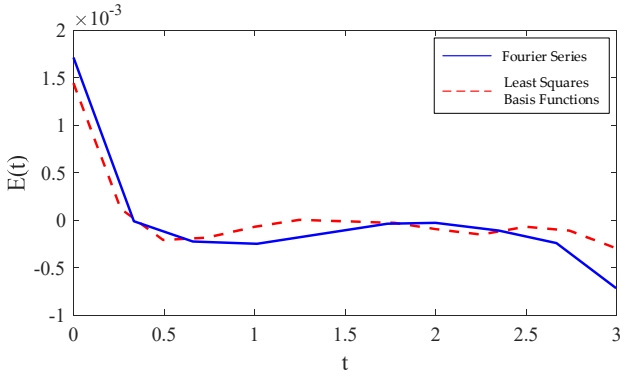


Figure 11. The relative error between the numerical solution obtained and the analytical solution.

The graphs and tables mentioned above clearly illustrate that the improved algorithm significantly enhances the accuracy of the approximate solutions. This algorithm effectively increases the precision of these solutions in solving differential equations.

4.5.2. Second-Order Nonlinear Differential Equation

Selecting a second-order nonlinear differential equation:

$$\begin{cases} y(y'' + \pi^2 y - y) = -2\pi \sin \pi x \cos \pi x \frac{e^{-2x} - e^{2x}}{(e^5 - e^{-5})^2} \\ y(0) = 0, y'(0) = 0 \end{cases}$$

The analytical solution of this equation is $y(t) = \frac{(e^{-t} - e^{2t}) \sin \pi x}{e^5 - e^{-5}}$.

For the above second-order linear differential equation, we used the least squares basis functions and the Fourier series to construct the approximate functions of the equation.

In accordance with the algorithmic procedure, the population size of the algorithm is set at 500, and the number of iterations is fixed at 10,000. The relative error between the numerically approximated solution and the exact solution is presented in Table 4.

Table 4. Comparison of mean squared error and maximum absolute error between numerical solution and analytical solution for first-order linear differential equation.

	Fourier Series	**Least Squares Basis Functions**
Mean squared error	8.4×10^{-9}	2.1×10^{-7}
Maximum absolute error	0.0013	0.0122

As indicated by Table 4 and Figure 12, the Fourier function approximation method employed in this study yields numerical solutions of higher precision compared to the least squares basis function approximation, as evidenced by the smaller relative error between the numerical and analytical solutions.

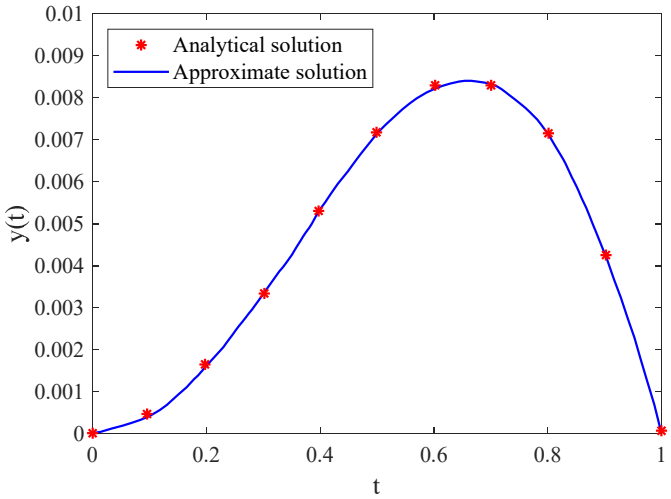


Figure 12. Comparison between numerical and analytical solutions of second-order nonlinear differential equations.

In summary, empirical evidence demonstrates that when extending from first-order to higher-order, whether dealing with first-order linear differential equations or second-order nonlinear differential equations, the analytical solutions obtained through the method proposed in this paper exhibit superior approximation results. Consequently, the algorithm presented in this paper demonstrates a higher degree of accuracy in solving boundary value problems for both first-order and higher-order differential equations.

4.5.3. No Analytic Solution to Differential Equation

Consider a differential equation with no analytic solution:

$$\frac{d}{dt}y(t) = t^2 + y^2(t), \quad t \in [0,1]$$

The interval $[0,1]$ is divided equally into ten parts for point selection and training. An approximate solution to the differential equation is sought using the improved algorithm in conjunction with the fourth-order Runge–Kutta method, with the results presented in Figure 13. As can be observed from Figure 13, the approximate solution obtained in this study is closely aligned with that derived from the fourth-order Runge–Kutta method, demonstrating the efficacy of the solution method proposed in this paper.

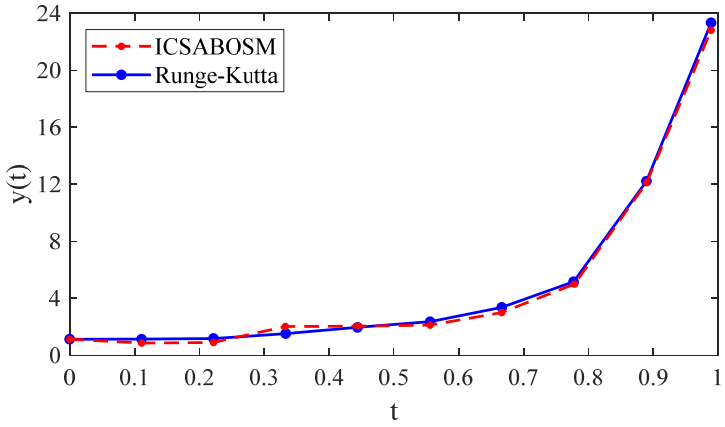


Figure 13. Comparison with the fourth-order Runge–Kutta method.

5. Conclusions

To address the deficiency of the CS algorithm in lacking information sharing among individuals, this paper introduces an enhanced CS algorithm based on a sharing mechanism. The population is initialized using the good points set method, replacing the random initialization employed in traditional algorithms. The improved algorithm introduces a feasible sharing area, incorporating information from other cuckoos with superior qualities, supplanting the original approach that relied solely on the best cuckoo in the population. This establishes a global search strategy based on the sharing mechanism. The adoption of a dual-difference vector, in lieu of the single-difference vector used in the CS, formulates a local search strategy under the same mechanism. By transforming the problem of solving differential equations into an optimization problem, and applying the enhanced algorithm to solve this optimization problem, a novel approach to solving differential equations is provided, thereby expanding the applications of both the ICSABOSM algorithm and the CS algorithm.

Funding: This research received no external funding.

Data Availability Statement: Data are included in the article.

Acknowledgments: Thank the reviewers for their comments and suggestions.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Kumar, N.; Mahato, S.K.; Bhunia, A.K. A new QPSO based hybrid algorithm for constrained optimization problems via tournamenting process. *Soft Comput.* **2020**, *24*, 11365–11379. [CrossRef]
2. Kumar, N.; Rahman, M.S.; Duary, A.; Mahato, S.K.; Bhunla, A.K. A new QPSO based hybrid algorithm for bound-constrained optimisation problem and its application in engineering design problems. *Int. J. Comput. Sci. Math.* **2021**, *12*, 385–412. [CrossRef]
3. Storn, R.; Storn, P. Differential evolution—A simple and efficient heuristic for global optimization over continuous space. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]

4. Avijit, D.; Kumar, N.; Akhtar, M.; Shalkh, A.A.; Bhunla, A.K. Real Coded Self-Organizing Migrating Genetic Algorithm for nonlinear constrained optimization problems. *Int. J. Oper. Res.* **2022**, *45*, 29–67.
5. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Mhs95 Sixth International Symposium on Micro Machine & Human Science, Nagoya, Japan, 4 November 1995.
6. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
7. Xue, Y.; Jiang, J.M.; Zhao, B.P.; Ma, T.H. A self-adaptive artificial bee colony algorithm based on global best for global optimization. *Soft Comput.* **2018**, *22*, 2935–2952. [CrossRef]
8. Kotte, S.; Pullakura, R.K.; Injeti, S.K. Optimal Multilevel Thresholding Selection for Brain MRI Image Segmentation based on Adaptive Wind Driven Optimization. *Measurement* **2018**, *130*, 340–361. [CrossRef]
9. Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing, Coimbatore, India, 9–11 December 2009.
10. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Transient search optimization: A new meta-heuristic optimization algorithm. *Appl. Intell.* **2020**, *50*, 3926–3941. [CrossRef]
11. Yang, X.S. Flower Pollination Algorithm for Global Optimization. *Unconv. Comput. Nat. Comput.* **2012**, *7445*, 240–249.
12. Cotta, C.; Mathieson, L.; Moscato, P. Memetic Algorithms. *Springer Int. Publ.* **2016**, *72*, 607–638.
13. Biazar, J.; Ghanbary, B. A new approach for solving systems of nonlinear equations. *Int. Math. Forum* **2008**, *38*, 1885–1889.
14. Jaberipour, M.; Khorram, E.; Karimi, B. Particle swarm algorithm for solving systems of nonlinear equations. *Comput. Math. Appl.* **2011**, *62*, 566–576. [CrossRef]
15. Oliveira, H.; Petraglia, A. Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing. *Appl. Soft Comput. J.* **2013**, *13*, 4349–4357. [CrossRef]
16. Raja, M.A.Z.; Kiani, A.K.; Shehzad, A.; Zameer, A. Memetic computing through bio-inspired heuristics integration with sequential quadratic programming for nonlinear systems arising in different physical models. *Springer Plus* **2016**, *5*, 2063. [CrossRef] [PubMed]
17. Ibrahim, A.M.; Tawhid, M.A. A hybridization of cuckoo search and particle swarm optimization for solving nonlinear systems. *Evol. Intell.* **2019**, *12*, 541–561. [CrossRef]
18. Verma, P.; Parouha, R.P. Solving Systems of Nonlinear Equations Using an Innovative Hybrid Algorithm. *Iran. J. Sci. Technol. Trans. Electr. Eng.* **2022**, *46*, 1005–1027. [CrossRef]
19. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
20. Thirugnanasambandam, k.; Prakash, S.; Subramanian, V. Reinforced cuckoo search algorithm-based multimodal optimization. *Appl. Intell.* **2019**, *49*, 2059–2083. [CrossRef]
21. Civicioglu, P.; Besdok, E.; Gunen, M.A. Weighted differential evolution algorithm for numerical function optimization: A comparative study with cuckoo search, artificial bee colony, adaptive differential evolution, and backtracking search optimization algorithms. *Neural Comput. Appl.* **2018**, *26*, 3923–3937. [CrossRef]
22. Lu, Z.; Dong, L.; Zhou, J. Nonlinear Least Squares Estimation for Parameters of Mixed Weibull Distributions by Using Particle Swarm Optimization. *IEEE Access* **2019**, *7*, 60545–60554. [CrossRef]
23. Cheng, J.T.; Xiong, Y. Multi-strategy adaptive cuckoo search algorithm for numerical optimization. *Artif. Intell. Rev.* **2022**, *56*, 2031–2055. [CrossRef]
24. Wei, J.M.; Yu, Y.G. A novel cuckoo search algorithm under adaptive parameter control for global numerical optimization. *Methodol. Appl.* **2020**, *24*, 4917–4940. [CrossRef]
25. Pauline, O. Adaptive cuckoo search algorithm for unconstrained optimization. *Sci. World J.* **2014**, *2014*, 943403.
26. Wang, G.G.; Zhang, Z.J.; Deb, S. Chaotic cuckoo search. *Soft. Comput.* **2016**, *20*, 3349–3362. [CrossRef]
27. Cheng, J.T.; Wang, L.; Jiang, Q.Y.; Cao, Z.J.; Xiong, Y. Cuckoo search algorithm with dynamic feedback information. *Future Gener. Comput. Syst.* **2018**, *89*, 317–334. [CrossRef]
28. Tsipianitis, A.; Tsompanakis, Y. Improved Cuckoo Search algorithmic variants for constrained nonlinear optimization. *Adv. Eng. Softw.* **2020**, *149*, 102865. [CrossRef]
29. Salgotra, R.; Singh, U.; Saha, S. New cuckoo search algorithms with enhanced exploration and exploitation properties. *Expert Syst. Appl.* **2018**, *95*, 384–420. [CrossRef]
30. Meng, X.J.; Chang, J.X.; Wang, X.B. Multi-objective hydropower station operation using an improved cuckoo search algorithm. *Energy* **2019**, *168*, 429–439. [CrossRef]
31. Gao, S.Z.; Gao, Y.; Zhang, Y.M.; Xu, L. Multi-strategy Adaptive Cuckoo Search Algorithm. *IEEE Access* **2019**, *7*, 137642–137655. [CrossRef]
32. Salgotra, R.; Singh, U.; Saha, S. Improved Cuckoo Search with Better Search Capabilities for Solving CEC 2017 Benchmark Problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018.
33. Rajabioun, R. Cuckoo Optimization Algorithm. *Appl. Soft Comput.* **2011**, *11*, 5508–5518. [CrossRef]
34. Li, H.; Xiang, S.; Yang, Y.; Liu, C. Differential evolution particle swarm optimization algorithm based on good point set for computing Nash equilibrium of finite noncooperative game. *AIMS Math.* **2021**, *6*, 1309–1323. [CrossRef]
35. Huang, Z.Y.; Gao, Z.Z.; Qi, L.; Duan, H. A Heterogeneous Evolving Cuckoo Search Algorithm for Solving Large-scale Combined Heat and Power Economic Dispatch Problems. *IEEE Access* **2019**, *7*, 111287–111301. [CrossRef]

36. Li, J.; Li, Y.X.; Tian, S.S.; Xia, J.L. An improved cuckoo search algorithm with self-adaptive knowledge learning. *Neural Comput. Appl.* **2019**, *32*, 11967–11997. [CrossRef]
37. Li, J.; Lei, H.; Wan, G.G. Solving Logistics Distribution Center Location with Improved Cuckoo Search Algorithm. *Int. J. Comput. Intell. Syst.* **2020**, *14*, 676–692. [CrossRef]
38. Cuong-Le, T.; Minh, H.L.; Khatir, S.; Wahab, M.A.; Tran, M.T.; Mirjalili, S. A novel version of Cuckoo search algorithm for solving optimization problems. *Expert Syst. Appl.* **2021**, *186*, 115669. [CrossRef]
39. Wang, J.; Zhou, B.; Zhou, S. An Improved Cuckoo Search Optimization Algorithm for the Problem of Chaotic Systems Parameter Estimation. *Comput. Intell. Neurosci.* **2016**, *2016*, 2959370. [CrossRef]