mathematics

MDPI

Article

An Improved Golden Jackal Optimization Algorithm Based on Mixed Strategies

Yancang Li [1], Qian Yu [1], Zhao Wang [1,*], Zunfeng Du [2] and Zidong Jin [3]

[1] School of Civil Engineering, Hebei University of Engineering, Handan 056038, China; liyancang@hebeu.edu.cn (Y.L.); 15879223963@163.com (Q.Y.)
[2] School of Civil Engineering, Tianjin University, Tianjin 300354, China; dzf@tju.edu.cn
[3] Jizhong Energy Fengfeng Group Co., Ltd., Handan 056038, China; 15083818090@163.com
* Correspondence: wangzhao546@163.com

Abstract: In an effort to overcome the problems with typical optimization algorithms' slow convergence and tendency to settle on a local optimal solution, an improved golden jackal optimization technique is proposed. Initially, the development mechanism is enhanced to update the prey's location, addressing the limitation of just relying on local search in the later stages of the algorithm. This ensures a more balanced approach to both algorithmic development and exploration. Furthermore, incorporating the instinct of evading natural predators enhances both the effectiveness and precision of the optimization process. Then, cross-mutation enhances population variety and facilitates escaping from local optima. Finally, the crossbar strategy is implemented to change both the individual and global optimal solutions of the population. This technique aims to decrease blind spots, enhance population variety, improve solution accuracy, and accelerate convergence speed. A total of 20 benchmark functions are employed for the purpose of comparing different techniques. The enhanced algorithm's performance is evaluated using the CEC2017 test function, and the results are assessed using the rank-sum test. Ultimately, three conventional practical engineering simulation experiments are conducted to evaluate the suitability of IWKGJO for engineering issues. The results obtained demonstrate the beneficial effects of the altered methodology and illustrate that the expanded golden jackal optimization algorithm has superior convergence accuracy and a faster convergence rate.

Keywords: golden jackal optimization algorithm; use development mechanism; predator avoidance behavior; cross mutation; crossbar strategy

MSC: 49K35

Citation: Li, Y.; Yu, Q.; Wang, Z.; Du, Z.; Jin, Z. An Improved Golden Jackal Optimization Algorithm Based on Mixed Strategies. *Mathematics* **2024**, *12*, 1506. https://doi.org/10.3390/math12101506

Academic Editor: Fabio Raciti

Received: 26 March 2024
Revised: 5 May 2024
Accepted: 7 May 2024
Published: 11 May 2024

1. Introduction

The optimization problem often involves determining the ideal amount, which can be either the maximum or least value, given a set of interconnected constraints. Optimization challenges are prevalent throughout various engineering fields, and there is a continuous rise in the need for answers. The approach to handling optimization problems has undergone multiple updates as a means to effectively address the growing complexity of engineering challenges in contemporary times. From the previous mathematical optimization to the current heuristic optimization and meta-heuristic algorithm, it has evolved from the traditional mathematical model to be capable of solving complicated problems with multiple dimensions. In the context of problem-solving, a heuristic algorithm is employed to acquire a viable solution within a specified number of iterations. The methodology remains constant, and the distinction between a viable option and an optimal solution remains indeterminate. The meta-heuristic algorithm integrates a stochastic algorithm with a local search algorithm. In other words, when a fixed value is provided, the output value obtained is not fixed due to the influence of "random factors". The drawback of the meta-heuristic algorithm is in its susceptibility to local optima due to its inherent randomness, resulting

in an inability to identify the global optimal advantage. The inherent unpredictability of the method renders its optimization process inapplicable to a wide range of engineering problems. Consequently, an increasing number of meta-heuristic algorithms have been proposed to address the growing complexity of engineering problems.

Meta-heuristic algorithms can be broadly classified into four categories depending on various sources of inspiration: group-based algorithms, evolution-based algorithms, physics-based algorithms, and human-based algorithms. The initial approach involves simulating the foraging and social behavior exhibited by the wild creature. The beluga whale optimization algorithm (BWO) [1] draws inspiration from the locomotion patterns exhibited by beluga whales, including swimming, hunting, and falling. The gazelle optimization algorithm (GOA) [2] is designed to replicate the behavioral patterns of gazelle as they attempt to evade predators. The snake optimization algorithm (SO) [3] depends on the habits and trends of snakes while foraging and breeding. The crawfish optimization algorithm (COA) [4] draws inspiration from the observed foraging, chilling, and competing behavior exhibited by crawfish. Another objective is to replicate the process of Darwinian evolution; the liver cancer optimization algorithm (LCA) [5] emulates the biological mechanisms underlying the proliferation and invasion of liver malignancies. The human evolutionary optimization algorithm (HEOA) [6] is an algorithm that draws inspiration from the process of human evolution. The lung function optimization algorithm (LPO) [7] draws inspiration from the physiological regularity and cognitive abilities observed in the human lung. The third is the process of simulating physical phenomena; the frost optimization algorithm (RIME) [8] is derived from the growth mechanism of haze ice and is based on the motion of soft frost ice particles. The geometric mean optimizer (GMO) [9] is a software tool that emulates the distinctive characteristics of geometric mean operators in the field of mathematics. The exponential distribution optimizer (EDO) [10] draws inspiration from mathematical models that are grounded in exponential probability distributions. Finally, there are algorithms that are based on human behavior. For instance, the war strategy optimization algorithm (AOW) [11] draws inspiration from the renowned ancient military theory known as The Art of War. The gold rush optimization algorithm (GRO) [12] draws inspiration from the experience of the gold rush and aims to replicate the actions and strategies employed by gold prospectors. The chef-based optimization algorithm (CBOA) [13] emulates the cooking behavior of a cooked individual in order to identify the optimal solution. The deep sleep optimizer (DSO) [14] algorithm emulates the sleep habits of humans with the objective of addressing optimization challenges. The football team training algorithm (FTTA) [15] is an optimization algorithm that is derived from the training methodology employed in football teams.

Multiple researchers have put forward various improvement techniques for multiple algorithms due to the traditional golden jackal optimization algorithm's inadequate convergence timetable and its propensity to fall into local optimality. An upgraded version (OGJO) of the golden jackal optimization algorithm utilizing reverse learning (OBL) technology has been put forward by Sarada Mohapatra et al. [16]. The updated strategy is contrasted with alternative algorithms. Conclusions indicate that, in comparison to GJO and other comparison algorithms, the OGJO suggested in this study is far more efficient. A golden jackal optimization approach is also described by Lin et al. [17] and utilized in conjunction with a cloud computing resources allocation strategy. The optimization objective for this study was to speed up the overall task completion time by utilizing CloudSim as the simulation experiment platform. The golden jackal optimization algorithm's iteration counter was modified, experiments were conducted, and the results were compared with other algorithms. Based on the results, the golden jackal optimization algorithm executes more effectively compared to other algorithms when there are more than 1000 tasks. With the goal of enhancing the golden jackal procedure's exploration and exploration ability, Xu et al. [18] established an optimization technique that makes use of an adaptive strategy. We verified the new algorithm using a numerical example without obstacles. The augmented golden jackal algorithm demonstrates remarkable convergence, as demonstrated

by our results. A golden jackal optimization position was originated by Xie et al. [19] in an effort to address the difficulty of minimizing PID conditions. The algorithm grows accustomed to the relative positions of male and female jackals in accordance with the golden jackal hunting rules by applying the three PID parameters to put together the position coordinates. The system's short readjustment time, quick rise in speed, and least overshoot constitute some of its benefits, as evidenced by its results, which likewise illustrate that the golden jackal optimization procedure governs the PID parameters of the system.

Hence, this work presents a novel golden jackal optimization technique that utilizes reflection substitution and crossbar strategy. The initial section provides a description of the development approach applied to the meta-heuristic algorithm and the golden jackal optimization algorithm. The subsequent section provides an introduction to the methods and processes involved in the fundamental golden jackal optimization algorithm. The third chapter describes the improved algorithm. The enhanced method involves (A) revising the placements of male and female golden jackals based on changes in their developmental stage in order to enhance the correctness of the answer; (B) incorporating the strategy of evading natural predators to enhance the overall optimization capacity of the population; (C) utilizing cross mutation to enhance population diversity and enhance the capacity to escape local optima; and (D) the inclusion of a crossbar strategy that serves to rectify the global optimal solution, mitigate blind areas, and enhance the pace of optimization. Section 4 of the study examines the disparity between the enhanced algorithm and the original method through a comparative analysis of each approach using a set of 20 benchmark test functions. Section 5 of the study compares the performance of several algorithms on CEC2017 test functions of different dimensions. The Wilcoxon rank-sum test is employed to identify any significant differences in the optimization results. Section 6 of the study involves the execution of simulation experiments, wherein three conventional engineering design optimization problems are employed. The experimental findings demonstrate that the IWKGJO algorithm exhibits a noticeable enhancement in both convergence speed and accuracy, rendering it well-suited for addressing practical engineering challenges.

2. Golden Jackal Optimization Algorithm

The golden jackal optimization algorithm is an artificial intelligence algorithm that was initially proposed by Nitish Chopra et al. in 2022 [20]. The algorithm took its inspiration from the scavenging tactics utilized by golden jackals. Golden jackals are solitary creatures that live in pairs. By virtue of their abilities to roam, hunt, and forage simultaneously, they could potentially obtain more formidable game in the region. These synchronized investigations are more productive than solo investigations. When jackals capture in tandem, they predominantly employ less to win more siege tactics. They adhere to their prey right away using the relay mode; once the prey is surrounded, they will employ less to win more wheel warfare and relentlessly attack the target. Eventually, the prey will fade out of energy and evolve into being physically worn out, at which point the jackals will launch a group attack to end the dispute.

By transforming the vantage point of the prey, the golden jackal optimization process rolls off the optimization process while mimicking the cooperative foraging conduct of golden jackals. On the global search space, the prey's starting location is picked at random. Upon the update pertaining to the prey population's place of residence, the male golden jackal was in the optimal position, the female golden jackal was in the unsatisfactory position, and the victim population's location was rewritten based on the precise spot of the golden jackal couple.

Contingent on energy, the algorithm divides the prey across three distinct stages: seeking, encompassing, and shooting. The golden jackal couple searches for a quarry when the prey vitality is high; when the prey energy falls below some specific threshold measurement, each of them occupies and attacks the prey.

2.1. Search Space Model

The initialization phase of the golden jackal can be formally outlined as follows:

$$X_0 = X_{min} + rand \times (X_{max} - X_{min}) \tag{1}$$

The formula combines X_0 to symbolize the initial collection of golden jackals, *rand* to symbolize a random number in the space [0, 1], and X_{max} and X_{min} to advocate for the highest and lowest boundaries of the answer to the issue, appropriately.

2.2. Search for Prey (Exploration Phase)

Like customary canine jackals, golden jackals are competent in recognizing and monitoring their prey, and yet, on occasion, the prey will be challenged to seize and discharge. The jackal waits and seeks out other prey as its outcome. Male jackals dominate the hunting. Male and female jackals follow their companions.

$$Y_1(t) = Y_M(t) - E \times |Y_M(t) - r_1 \times Prey(t)| \tag{2}$$

$$Y_2(t) = Y_{FM}(t) - E \times |Y_{FM}(t) - r_1 \times Prey(t)| \tag{3}$$

where the current repetition count, t, is given. In the t iteration, the prey's position is labeled via $Prey(t)$; the positions of the male and female jackals are marked by $Y_M(t)$ and $Y_{FM}(t)$, respectively. The male and female jackals' updated positions that correspond to the prey in the t iteration is designated by $Y_1(t)$ and $Y_2(t)$, and so forth.

E is the prey's getaway energy, and it can perhaps be arrived at as such:

$$E = E_1 \times E_0 \tag{4}$$

E_0 symbolizes the prey's initial degree of energy, whereas E_1 indicates the prey's energy declining.

$$E_0 = 2 \times r - 1 \tag{5}$$

where r is a randomized value spanning 0 and 1.

$$E_1 = C_1 \times \left(1 - \left(\frac{t}{T}\right)\right) \tag{6}$$

where T is the largest number of repetitions; t is the number of iterations that occur during that moment; and C_1 is a continuous value with an estimate of 1.5. As the cycle evolves, E_1 drops uniformly between 1.5 to 0. A random number evolved from a Levy distribution is represented by r_1. The flight Levy operates, or $LF(y)$, is arrived at in this manner:

$$r_1 = 0.05 \times LF(y) \tag{7}$$

$$LF(y) = 0.01 \times \frac{(\mu \times \sigma)}{|v|^{\frac{1}{\beta}}} \tag{8}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\pi \times \frac{\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}}\right)^{\frac{1}{\beta}} \tag{9}$$

The quantity of arbitrary numbers in the μ and v formats is the equivalent of (0, 1). β possesses a default value of 1.5, like an integer.

Primarily, the formula for releasing updates on the Wolf's position operates using the equation below:

$$Y_{t+1}^{EP} = \frac{Y_1(t) + Y_2(t)}{2} \tag{10}$$

where the jackal's position following $t + 1$ repetitions is denoted by $Y(t + 1)$.

2.3. Surround and Attack Prey (Development Phase)

Prey that has been infected by jackals has less energy to flee, and a jackal pair will then encircle the prey that was initially identified. The jackals assault and consume their victim once they have surrounded it. The following is the mathematical model of an assortment of male and female jackals hunting together:

$$Y_1(t) = Y_M(t) - E \times |r_1 \times Y_M(t) - Prey(t)| \tag{11}$$

$$Y_2(t) = Y_{FM}(t) - E \times |r_1 \times Y_{FM}(t) - Prey(t)| \tag{12}$$

where t is the number of current iterations; $Prey(t)$ is the position of the prey in the t iteration; $Y_M(t)$ and $Y_{FM}(t)$ were the positions of male and female jackals in the t iteration, respectively; and $Y_1(t)$ and $Y_2(t)$ are the updated positions of male and female jackals corresponding to prey in the t iteration, respectively.

For Equation (12), where the current iteration count, t, is displayed, in the t iteration, the prey's position is symbolized with $Prey(t)$; the positions of the male and female jackals are denoted by $Y_M(t)$ and $Y_{FM}(t)$, respectively; and the male and female jackals' updated regions that correspond to the prey in the t iteration are designated by $Y_1(t)$ and $Y_2(t)$.

And subsequently, a whereabouts update on the jackal can be defined using

$$Y_{t+1}^{DP} = \frac{Y_1(t) + Y_2(t)}{2} \tag{13}$$

3. Improved Golden Jackal Optimization Algorithm

3.1. Improved Ways to Update Locations during Development

In this paper, a new development mechanism is proposed because when the escape energy $E < 1$, the prey cannot escape due to the jackal infestation, but, as a result, it will fall into the local optimal and cannot escape the global search for better prey. Moreover, even if the escape energy is reduced, the prey still has a chance to escape.

During the first stage of the iteration, the prey exhibits a heightened inclination to flee, resulting in a rapid emission of speed. Consequently, the elite matrix $Y_{\mathrm{E}}(t)$ was constructed based on the optimal fitness values of golden jackal individuals. Subsequently, all elite individuals commenced their exploration for global search. The prey's location is updated using the Brown random walk algorithm. Here is the mathematical model:

$$Y_1(t) = Y_M(t) - E \times D_1 \tag{14}$$

$$Y_2(t) = Y_{FM}(t) - E \times D_1 \tag{15}$$

$$D_1 = RB \times |Y_{\mathrm{E}}(t) - RB \times Prey(t)| \tag{16}$$

The vector RB comprises random numbers that are generated by the process of Brownian random walk.

As the prey is confined for an extended period, its strength diminishes and its speed also declines. During the intermediate phase of the iteration, when the velocity of the golden jackal individual and the prey individual were equivalent, the golden jackal individual implemented a division of action. Specifically, half of the golden jackal individuals engaged in development activities following Levi's flight strategy, while the remaining half pursued exploration activities based on Brownian Walk. The implementation of a division of labor not only facilitates the identification of the local ideal value, but also serves as a safeguard against the convergence of the local optimal value towards the global optimal value. Below are the mathematical models.

Development guidelines derived on Levy's flight strategy:

$$Y_1(t) = Y_M(t) - E \times D_2 \tag{17}$$

$$Y_2(t) = Y_{FM}(t) - E \times D_2 \tag{18}$$

$$D_2 = RL \times |Y_{\mathrm{E}}(t) - RL \times Prey(t)| \tag{19}$$

Exploration guidelines derived from the Brownian walk:

$$Y_1(t) = Y_2(t) = Y_{\mathrm{E}}(t) - E \times D_3 \tag{20}$$

$$D_3 = RB \times |RB \times Y_{\mathrm{E}}(t) - Prey(t)| \tag{21}$$

Upon completion of the iteration, the golden jackal's physical strength diminished and its velocity decelerated. Currently, the golden jackal employed Levi's itinerant approach to evolve. Golden jackals, due to diminished escape energy, encircle and assault their prey. Here is the mathematical model:

$$Y_1(t) = Y_2(t) = Y_{\mathrm{E}}(t) - E \times D_4 \tag{22}$$

$$D_4 = RL \times |RL \times Y_{\mathrm{E}}(t) - Prey(t)| \tag{23}$$

3.2. Strategies for Avoiding Natural Enemies

While striving for dominance, the golden jackal pair may inevitably come across their natural adversaries or rivals for food, be it due to global or local factors. Hence, it is imperative to integrate avoidance behaviors against natural predators in order to safeguard themselves when foraging. This study introduces an escape factor, denoted as EE, specifically designed for golden jackals. When the escape value exceeds 0.25, it indicates that the golden jackal couple has encountered a natural opponent or predator and successfully evaded it by moving closer to the elites. Conversely, if the escape factor is below 0.25, it signifies that they are in a secure situation. The mathematical formulas are as follows:

If $EE < 0.25$:

$$Y_1(t) = Y_M(t) - b_1 \times b_2 \times |Y_M(t) - Prey(t)| \tag{24}$$

$$Y_2(t) = Y_{FM}(t) - b_1 \times b_2 \times |Y_{FM}(t) - Prey(t)| \tag{25}$$

If $EE \geq 0.25$:

$$Y_1(t) = Y_{\mathrm{E}}(t) - b_1 \times b_2 \times |Y_M(t) - Prey(t)| \tag{26}$$

$$Y_2(t) = Y_{\mathrm{E}}(t) - b_1 \times b_2 \times |Y_{FM}(t) - Prey(t)| \tag{27}$$

$$b_1 = \tan(2 \times \pi \times r) \tag{28}$$

$$b_2 = 1 - \frac{1}{1 + e^{(\frac{t}{T^2} - \frac{1}{2T}) \times 10}} \tag{29}$$

where $Y_M(t)$ is the current position of the male golden jackal; $Y_{FM}(t)$ is the current position of the female golden jackal; and $Y_{\mathrm{E}}(t)$ is the elite matrix of optimal fitness values of the golden jackal.

3.3. Cross Mutations

A crossover mutation mechanism, akin to differential evolution, is incorporated here to achieve a local optimum after evading natural enemies. This procedure will be applied to the present location of the golden jackal and their respective new locations. Four individuals are randomly selected from the population, except the present one. A crossover operation is then undertaken utilizing crossover mutation to generate the mutation vectors. Finally, greedy selection is performed. The mathematical representation of the crossover operation is as follows:

$$Y_{i,j}^{new} = \begin{cases} Y_{i,j}^{mu}, & if\ r \le f \\ Y_{i,j}^{old}, & if\ r > f \end{cases} , j = 1, 2, \ldots, dim \tag{30}$$

where i is the number of populations; j is a random number of $[1, dim]$; $Y_{i,j}^{new}$ is the position of the golden jackal after the crossover mutation; $Y_{i,j}^{mu}$ is the mutation vector generated using four random positional mutations; and $Y_{i,j}^{old}$ is the original position of the golden jackal.

3.4. Crossbar Strategy

For supplying the portion of the golden jackal population that fits into the local perfect an opening of fleeing the iteration, the algorithm performs two sorts of crossovers, horizontal and vertical, throughout each generation of the iteration. The article will present the crossbar procedure [21] to increase the global optimization accuracy and the capacity to jump out of the local optimum, thus preventing the golden jackal individual from stepping into a state of "prematurity".

The cross protocol generates the offspring generation $M_{i,d}^{hc}$ and $M_{j,d}^{hc}$ after each crossing. These two crossing methodologies are organically merged by adding the competition operator. Adhering to each crossing operation, the competition operator enters and engages in a rivalry with the parent generation, retaining only the particles that outperform the parent generation for the next iteration of the process.

3.4.1. Horizontal Crossing

Horizontal crossover, like the GA crossover operation, is an arithmetic crossover between two distinct individual particles of the same dimension in the population. It encourages individual learning from other individuals to improve the ability of global optimization to prevent premature convergence. The formula for the formation of children of parent individual particles $X(i)$ and $X(j)$, presuming that they cross horizontally in the D-th dimension, is as follows:

$$M_{i,d}^{hc} = r_2 \times X(i,d) + (1 - r_2) \times X(j,d) + c_1 \times (X(i,d) - X(j,d)) \tag{31}$$

$$M_{j,d}^{hc} = r_3 \times X(j,d) + (1 - r_3) \times X(i,d) + c_2 \times (X(j,d) - X(i,d)) \tag{32}$$

Horizontal crossover is an arithmetic crossover between the two distinct points where r_2 and r_3 are random numbers of [0, 1]; both c_1 and c_2 are arbitrary values of $[-1, 1]$; and $X(i,d)$ and $X(j,d)$ are the parents of d dimension $X(i)$ and $X(j)$, respectively. This is analogous to the GA crossover operation. By horizontal crossing, the D-dimensional progeny of $X(i,d)$ and $X(i,d)$ are represented by $M_{i,d}^{hc}$ and $M_{j,d}^{hc}$, respectively. Following the completion of the horizontal crossing operation, the produced progeny has to be compared to the parent particle's fitness value in that proportion. The individual with the best fitness is always adopted.

3.4.2. Vertical Crossing

An arithmetic crossing between two separate dimensions of a particle inside a population is termed a horizontal crossing. Each longitudinal crossing operation only generates one child particle and updates one dimension because the elements of different dimensions have different value ranges, so the two dimensions must be normalized before crossing. Meanwhile, each longitudinal crossing operation must jump out of the regional optimal without destroying the information of the other dimension if one dimension has stalled and fallen into the local optimal. The subgeneration M_{i,d_1}^{vc} occurs using formula (3), assuming that the d_1 and d_2 dimensions of particle $X(i)$ participate in longitudinal crossing.

$$M_{i,d_1}^{vc} = r_4 \times X(i,d_1) + (1 - r_4) \times X(j,d_2) \tag{33}$$

where the random number r_4 resides on $[0,\ 1]$ and the child of parent $X(i)$ created by horizontal crossing in d_1 and d_2 dimensions is M_{i,d_1}^{vc}. Longitudinal crossing produces offspring individuals that compete with parent individuals for the custody of more fit individuals.

3.5. IWKGJO Algorithm Flow Chart

The Figure 1 algorithm flow chart is as follows:

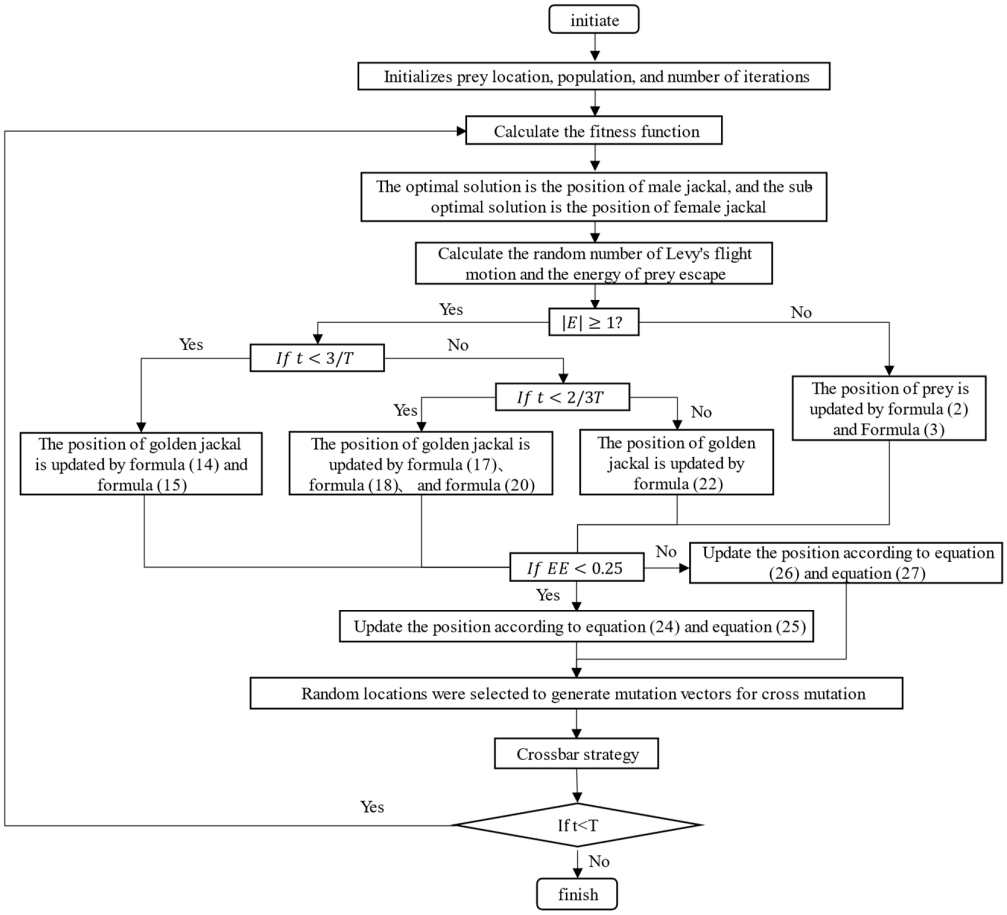


Figure 1. IWKGJO algorithm flow chart.

3.6. Upgraded Golden Jackal Optimization Algorithm Pseudo-Code

The above is a complete description of the improved Golden Jackal optimization algorithm, and its pseudo-code is explained in Algorithm 1.

Algorithm 1 Improved the pseudocode of sand cat swarm optimization algorithm

input: Initialize the population structure $\vec{x} = \left\{\vec{x_1}, \vec{x_2}, \ldots, \vec{x_n}\right\}, t, n$

output: Global optimal solution $\vec{x_k} = \left\{\vec{x_{1k}}, \vec{x_{2k}}, \ldots, \vec{x_{nk}}\right\}$

Initializes prey location, population, and number of iterations
The fitness of N individuals was calculated
The optimal solution is taken as the position of male golden jackal, and the sub-optimal solution is taken as the position of female golden jackal
Calculate the random number of Levy's flight motion and the energy of prey escape
While $t < T$
  For each $x_i, i = 1, 2, \ldots, N$
    if $(|E| < 1)$
      if $(t < \mathrm{T}/3)$
        Update the position according to Formulas (14) and (15)
      else if $(t < 2T/3)$
        Update the position according to Formulas (17) and (18)
      else
        Update the position according to Formula (20)
      end
    else
      Update the position according to Formula (22)
    end if
    if $(|E| \geq 1)$
      Update the position according to Formulas (2) and (3)
    end if
  if $(EE < 0.25)$
    Update the position according to Equations (24) and (25)
  else
    Update the position according to Equations (26) and (27)
  end if
    Random locations were selected to generate mutation vectors for cross mutation
    According to Formulas (31)–(33), the crossbar strategy is carried out
end for
$t = t + 1$
end while

3.7. Time Complexity

The temporal complexity of an algorithm is typically denoted as O. The IWKGJO algorithm primarily consists of an enhanced development phase position update, reflection instead of development, and crossbar. When the population size is N, the search space dimension is D, and the largest number of iterations is T, the time complexity analysis of this algorithm is shown in Table 1 below.

Table 1. Time complexity.

Algorithm Phase	Time Frequency T	Time Complexity O
Initial population	$T(n) = D$	$O(n) = D$
Fitness value calculation and ranking	$T(n) = N + N\log N$	$O(n) = T \times (N + N\log N)$
Change development behavior update golden jackal location	$T(n) = N \times D$	$O(n) = T \times (N \times D)$
Incorporation of predator avoidance behavior	$T(n) = N \times D$	$O(n) = T \times (N \times D)$
Crossover mutation	$T(n) = 1.5 \times D$	$O(n) = T \times (1.5 \times D)$
Crossbar strategy	$T(n) = 0.5 \times N \times D + 0.5 \times D$	$O(n) = T \times (N \times D + D)$
Total	$T(n) = 3D + T \times N \times (1 + \log N + 2.5D)$	$O(n) = D + T \times N \times (1 + \log N + D)$

3.8. Terms and Abbreviations

Table 2 shows the parameters used in this paper, and Table 3 shows the abbreviations of each strategy and comparison algorithm used in this paper.

Table 2. Parameter abbreviation table.

Basic algorithm	t	Current iterations	Improved algorithm	Y_{E}	Elite matrix
	T	Maximum iterations		RB	A random number vector generated by a Brownian random walk
	E	The escape energy of prey		RL	A random number vector that simulates Levi's motion
	r_1	Random numbers based on Levy distribution		$Y_{i,j}^{new}$	Location of the golden jackal after crossover mutation
	r	A random number from 0 to 1		$Y_{i,j}^{mu}$	Mutation vectors generated using four random positional mutations
	$LF(y)$	Levy's flight function		$Y_{i,j}^{old}$	The original location of the golden jackal
	$Prey$	Prey location		EE	The escape factor of the golden jackal
	Y_M	Location of the male jackal		M^{hc}	Horizontal cross generation
	Y_{FM}	Location of the female jackal		M^{vc}	Longitudinal crossing produces a progeny
	Y_1	The position of the male jackal corresponding to the prey		$r_2,\ r_3, r_4$	A random number from 0 to 1
	Y_2	The position of the female jackal corresponding to the prey		c_1, c_2	A random number from -1 to 1

Table 3. Algorithm abbreviation table.

Algorithm Shorthand	Algorithm Policy or Algorithm Name
KGJO	Change the development phase location update policy
QGJO	Predator avoidance behavior
CGJO	Cross mutation
ZGJO	Crossbar strategy
IWKGJO	An improved golden jackal optimization algorithm based on mixed strategies
GTO	Artificial gorilla troops optimizer
WO	Walrus optimizer
MPA	Marine predators algorithm
HMSWHO	Hybrid multi-strategy improved wild horse optimizer
IGWO	Grey wolf optimization algorithm based on elite learning for nonlinear parameters
LSHADE	Improving the search performance of SHADE using linear population size reduction
ECWOA	Whale optimization algorithm based on elite opposition-based and crisscross optimization

4. Analysis and Outcomes of the Simulation

4.1. The Impact of Multiple Approaches of Improvement on Algorithmic Performance

The simulation experiment in this paper is based on 12th Gen Intel(R) Core (TM) i7-12700H CPU @ 2.30 GHz memory 16.0 GB, Windows 11 operating system, and MATLAB R2023a simulation software.

This paper employs several improvement tactics, including the modification of the developmental stage position update strategy for selection (KGJO), incorporating natural enemy avoidance behavior (QGJO), crossover mutation (CGJO), and longitudinal and horizontal crossover strategy (ZGJO).

Twenty benchmark functions were utilized to evaluate the IWKGJO algorithm's performance. Six multi-peak test functions, ten multi-peak functions with fixed dimensions, and seven single-peak test functions make up the benchmark test functions. Since there is only one global ideal value for F1 through F7, this value is frequently used to assess the algorithm's development potential. The algorithm's capacity for both local optimal avoidance and exploration can be assessed by F8–F13. $N = 50$, $D = 50$, $T = 1000$, maximum number of iterations, repeat 30 times, and determine the ideal, average, and standard deviation values for comparison. In Table 4, the benchmark functions are displayed.

Table 4. Twelve benchmark functions.

Function	Function Name	Dimension	Radius	Optimal Value
$f_1(x) = \sum_{i=1}^{n} x_i^2$	Sphere Mode	50	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	Schwefel's Problem 2.22	50	$[-10, 10]$	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	Schwefel's Problem 1.2	50	$[-100, 100]$	0
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	Schwefel's Problem 2.21	50	$[-100, 100]$	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	Generalized Rosenbrock's Function	50	$[-30, 30]$	0
$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	Step Function	50	$[-100, 100]$	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	Quartic Function	50	$[-1.28, 1.28]$	0
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$	Generalized Schwefel's Problem 2.26	50	$[-500, 500]$	$-418.98 \times dim$
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$	Generalized Rastrigin's Function	50	$[-5.12, 5.12]$	0
$F_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	Ackley's Function	50	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	Generalized Griewank Function	50	$[-600, 600]$	0
$F_{12}(x) = \frac{\pi}{n} \left\{ 10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + 10\sin^2(\pi y_{i+1}) \right] + (y_n - 1)^2 \right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a < x_i < a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	Generalized Penalized Function	50	$[-50, 50]$	0
$F_{13}(x) = 0.1 \left\{ \sin^2(3\pi x_i) + \sum_{i=1}^{n} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_1 + 1) \right] + (x_n - 1)^2 \left[1 + \sin^2(2\pi x_n) \right] \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Generalized Penalized Function	50	$[-50, 50]$	0
$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} \left(x_i - a_{ij} \right)^6} \right)^{-1}$	Shekel's Foxholes Function	2	$[-65, 65]$	0
$F_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1 \left(b_i^2 + b_1 x_2 \right)}{b_i^2 + b_1 x_3 + x_4} \right]^2$	Kowalik's Function	4	$[-5, 5]$	0.1484
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3} x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	Six-Hump Camel-Back Function	2	$[-5,5]$	-1
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10\left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	Branin Function	2	$[-5, 5]$	0.3
$F_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2 \right) \right] \times \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2 \right) \right]$	Goldstein-Price Function	2	$[-2, 2]$	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij} \left(x_j - p_{ij} \right)^2 \right)$	Hartman's Function	3	$[1, 3]$	-3
$F_{20}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij} \left(x_j - p_{ij} \right)^2 \right)$	Hartman's Function	6	$[0, 1]$	-3

4.2. Comparison and Analysis of Experimental Results

Table 5 demonstrates that the single-peak function produces results for benchmark functions F1–F4 that achieve the theoretical optimal value. Additionally, the results of CGJO also reach the theoretical optimal value, indicating that differential crossover significantly enhances the algorithm's accuracy. In function F5, the enhancement of IWKGJO's outcomes compared to the original algorithm could be more evident. However, with numerous

iterations and testing, IWKGJO consistently yields superior results in comparison to the original method. In function F6, the IWKGJO algorithm falls short of achieving the theoretical optimal value, but it exhibits significantly higher accuracy compared to the original algorithm. In function F7, the optimal fitness value, average fitness value, and standard deviation all outperform the original algorithm and various strategies.

Table 5. Results of benchmark functions for different strategies.

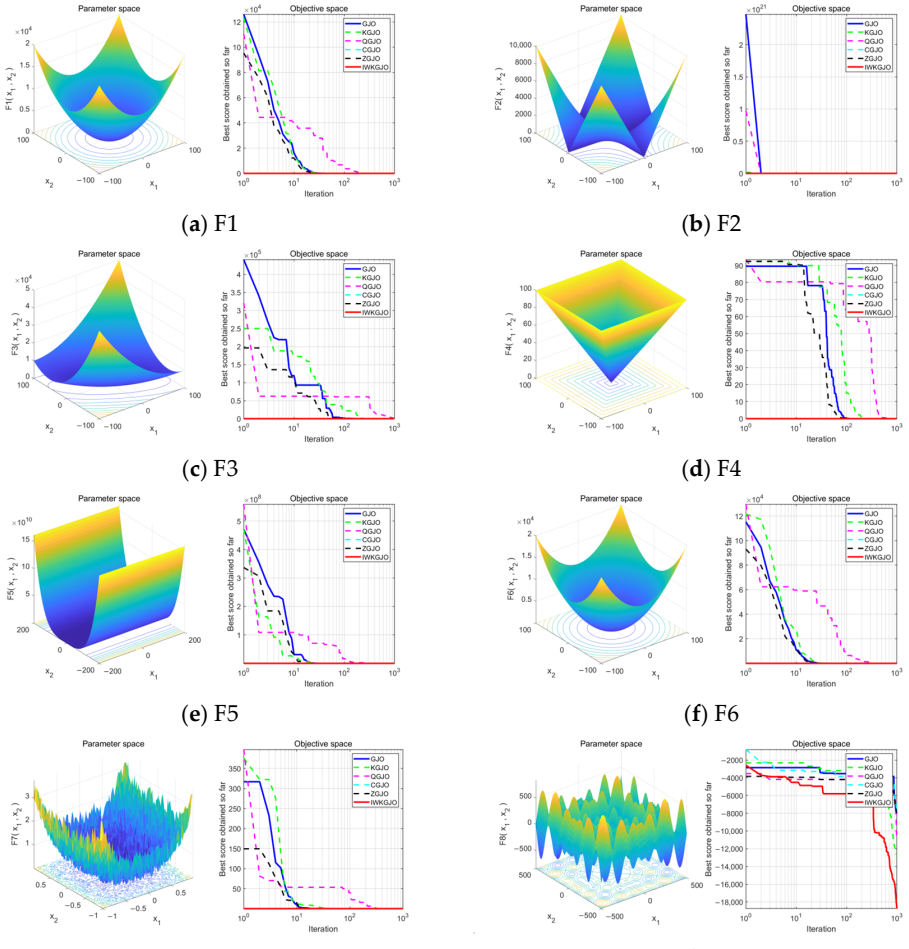
Function		Best	Mean	Std	Function		Best	Mean	Std
F1	GJO	4.2525×10^{-98}	2.1424×10^{-95}	3.7145×10^{-95}	F11	GJO	**0**	**0**	**0**
	KGJO	1.4442×10^{-50}	1.4725×10^{-46}	6.9333×10^{-46}		KGJO	**0**	**0**	**0**
	QGJO	1.9293×10^{-31}	6.6045×10^{-29}	1.4737×10^{-28}		QGJO	**0**	5.1969×10^{-3}	9.4074×10^{-3}
	CGJO	**0**	**0**	**0**		CGJO	**0**	**0**	**0**
	ZGJO	3.0416×10^{-108}	4.1617×10^{-103}	1.4710×10^{-102}		ZGJO	**0**	**0**	**0**
	IWKGJO	**0**	**0**	**0**		IWKGJO	**0**	**0**	**0**
F2	GJO	3.6518×10^{-58}	1.2592×10^{-56}	1.9164×10^{-56}	F12	GJO	1.8493×10^{-1}	3.4231×10^{-1}	9.5692×10^{-2}
	KGJO	4.0686×10^{-35}	2.2044×10^{-33}	4.0865×10^{-33}		KGJO	3.9270×10^{-3}	2.1827×10^{-2}	1.4613×10^{-2}
	QGJO	7.1171×10^{-23}	3.6323×10^{-21}	4.7920×10^{-21}		QGJO	9.1780×10^{-4}	9.0705×10^{-3}	5.9016×10^{-3}
	CGJO	**0**	**0**	**0**		CGJO	2.0848×10^{-1}	3.5093×10^{-1}	8.7961×10^{-2}
	ZGJO	7.9179×10^{-65}	2.8115×10^{-63}	8.3398×10^{-63}		ZGJO	2.9042×10^{-5}	6.5019×10^{-5}	3.8049×10^{-5}
	IWKGJO	**0**	**0**	**0**		IWKGJO	$\mathbf{4.5846 \times 10^{-7}}$	$\mathbf{8.8785 \times 10^{-7}}$	$\mathbf{2.0808 \times 10^{-7}}$
F3	GJO	3.9230×10^{-38}	4.3940×10^{-26}	2.4053×10^{-25}	F13	GJO	2.7586	3.3703	2.9969×10^{-1}
	KGJO	5.0512×10^{-4}	2.9223	5.5313		KGJO	8.4821×10^{-1}	1.4645	3.8423×10^{-1}
	QGJO	8.7224	7.0054×10^{2}	8.3814×10^{2}		QGJO	2.0702×10^{-1}	5.0356×10^{-1}	2.3235×10^{-1}
	CGJO	**0**	**0**	**0**		CGJO	3.2285	4.1204	3.2214×10^{-1}
	ZGJO	2.7449×10^{-40}	1.0937×10^{-30}	5.6667×10^{-30}		ZGJO	5.7459×10^{-1}	1.3600	7.4677×10^{-1}
	IWKGJO	**0**	**0**	**0**		IWKGJO	$\mathbf{1.1995 \times 10^{-5}}$	$\mathbf{2.3321 \times 10^{-5}}$	$\mathbf{6.3099 \times 10^{-6}}$
F4	GJO	2.5799×10^{-28}	7.6844×10^{-23}	2.8050×10^{-22}	F14	GJO	**0.998**	3.6786	3.7051
	KGJO	3.5351×10^{-6}	5.0790×10^{-1}	1.5581		KGJO	**0.998**	6.4072	4.6211
	QGJO	6.2669×10^{-4}	5.7202×10^{-3}	4.9917×10^{-3}		QGJO	**0.998**	3.4141	3.8184
	CGJO	**0**	**0**	**0**		CGJO	**0.998**	3.0224	3.6078
	ZGJO	1.4350×10^{-28}	5.3344×10^{-21}	2.9155×10^{-20}		ZGJO	**0.998**	1.3871	2.1311
	IWKGJO	**0**	**0**	**0**		IWKGJO	**0.998**	$\mathbf{9.9800 \times 10^{-1}}$	$\mathbf{4.1438 \times 10^{-16}}$
F5	GJO	4.6189×10^{1}	4.7618×10^{1}	8.6933×10^{-1}	F15	GJO	$\mathbf{3.0749 \times 10^{-4}}$	4.0393×10^{-4}	2.7900×10^{-4}
	KGJO	4.3236×10^{1}	4.6090×10^{1}	1.4070		KGJO	$\mathbf{3.0749 \times 10^{-4}}$	3.6888×10^{-4}	2.3223×10^{-4}
	QGJO	4.4327×10^{1}	5.4286×10^{1}	2.8295×10^{1}		QGJO	$\mathbf{3.0749 \times 10^{-4}}$	4.1534×10^{-4}	3.2148×10^{-4}
	CGJO	4.6234×10^{1}	4.8228×10^{1}	6.1710×10^{-1}		CGJO	$\mathbf{3.0749 \times 10^{-4}}$	3.0803×10^{-4}	2.5595×10^{-6}
	ZGJO	4.5782×10^{1}	4.6488×10^{1}	4.0979×10^{-1}		ZGJO	$\mathbf{3.0749 \times 10^{-4}}$	3.0751×10^{-4}	2.2818×10^{-8}
	IWKGJO	$\mathbf{4.1840 \times 10^{1}}$	$\mathbf{4.2198 \times 10^{1}}$	$\mathbf{1.6745 \times 10^{-1}}$		IWKGJO	$\mathbf{3.0749 \times 10^{-4}}$	$\mathbf{3.7188 \times 10^{-4}}$	$\mathbf{2.4540 \times 10^{-4}}$
F6	GJO	3.2495	5.5966	8.0268×10^{-1}	F16	GJO	**−1.0316**	**−1.0316**	2.1027×10^{-8}
	KGJO	3.1265×10^{-1}	1.0136	4.9168×10^{-1}		KGJO	**−1.0316**	**−1.0316**	6.7751×10^{-12}
	QGJO	2.0457×10^{-2}	2.0360×10^{-1}	1.7412×10^{-1}		QGJO	**−1.0316**	**−1.0316**	8.5481×10^{-10}
	CGJO	4.3640	5.9803	6.5417×10^{-1}		CGJO	**−1.0316**	**−1.0316**	2.8447×10^{-8}
	ZGJO	5.2983×10^{-4}	8.1070×10^{-4}	2.0504×10^{-4}		ZGJO	**−1.0316**	**−1.0316**	5.1632×10^{-9}
	IWKGJO	$\mathbf{2.4548 \times 10^{-5}}$	$\mathbf{4.4110 \times 10^{-5}}$	$\mathbf{1.3988 \times 10^{-5}}$		IWKGJO	**−1.0316**	**−1.0316**	$\mathbf{9.7486 \times 10^{-14}}$
F7	GJO	2.6397×10^{-6}	2.2152×10^{-4}	2.2152×10^{-4}	F17	GJO	**0.39789**	**0.39789**	2.2780×10^{-6}
	KGJO	9.0254×10^{-5}	1.6917×10^{-3}	1.5958×10^{-3}		KGJO	**0.39789**	**0.39789**	9.1299×10^{-10}
	QGJO	5.4060×10^{-3}	1.3561×10^{-2}	5.4480×10^{-3}		QGJO	**0.39789**	**0.39789**	9.2359×10^{-8}
	CGJO	7.0067×10^{-7}	9.2560×10^{-6}	6.9026×10^{-6}		CGJO	**0.39789**	**0.39789**	3.7091×10^{-5}
	ZGJO	1.1514×10^{-5}	9.6760×10^{-5}	8.9253×10^{-5}		ZGJO	**0.39789**	**0.39789**	8.4509×10^{-7}
	IWKGJO	$\mathbf{2.2378 \times 10^{-7}}$	$\mathbf{6.6218 \times 10^{-6}}$	$\mathbf{5.5602 \times 10^{-6}}$		IWKGJO	**0.39789**	**0.39789**	1.4291×10^{-11}
F8	GJO	-1.0220×10^{4}	-6.2153×10^{3}	1.9849×10^{3}	F18	GJO	**3**	**3**	4.8005×10^{-7}
	KGJO	-1.5270×10^{4}	-1.3242×10^{4}	1.0151×10^{3}		KGJO	**3**	**3**	4.5544×10^{-8}
	QGJO	-1.2357×10^{4}	-1.0717×10^{4}	1.0224×10^{3}		QGJO	**3**	**3**	1.4914×10^{-8}
	CGJO	-7.7355×10^{3}	-5.1263×10^{3}	1.4636×10^{3}		CGJO	**3**	**3**	4.0012×10^{-7}
	ZGJO	-1.1021×10^{4}	-6.9912×10^{3}	2.0483×10^{3}		ZGJO	**3**	**3**	2.3088×10^{-7}
	IWKGJO	$\mathbf{-2.0179 \times 10^{4}}$	$\mathbf{-1.9552 \times 10^{4}}$	$\mathbf{4.5937 \times 10^{2}}$		IWKGJO	**3**	**3**	$\mathbf{8.0821 \times 10^{-10}}$
F9	GJO	**0**	**0**	**0**	F19	GJO	**−3.8628**	−3.8588	3.9963×10^{-3}
	KGJO	**0**	3.7896×10^{-15}	2.0756×10^{-14}		KGJO	**−3.8628**	−3.8605	3.5558×10^{-3}
	QGJO	6.5179	2.8870×10^{1}	1.3348×10^{1}		QGJO	**−3.8628**	−3.8612	3.2064×10^{-3}
	CGJO	**0**	**0**	**0**		CGJO	**−3.8628**	−3.8599	3.8518×10^{-3}
	ZGJO	**0**	**0**	**0**		ZGJO	**−3.8628**	**−3.8628**	2.4559×10^{-5}
	IWKGJO	**0**	**0**	**0**		IWKGJO	**−3.8628**	**−3.8628**	$\mathbf{4.9232 \times 10^{-11}}$
F10	GJO	3.9968×10^{-15}	5.8916×10^{-15}	1.8027×10^{-15}	F20	GJO	**−3.3220**	−3.1125	1.7082×10^{-1}
	KGJO	7.5495×10^{-15}	1.3471×10^{-14}	4.6959×10^{-15}		KGJO	**−3.3220**	−3.1867	1.0916×10^{-1}
	QGJO	2.5313×10^{-14}	3.4313×10^{-14}	6.2401×10^{-15}		QGJO	**−3.3220**	−3.2571	6.7994×10^{-2}
	CGJO	$\mathbf{4.4409 \times 10^{-16}}$	$\mathbf{4.4409 \times 10^{-16}}$	**0**		CGJO	**−3.3220**	−3.1027	2.3208×10^{-1}
	ZGJO	3.9968×10^{-15}	5.7732×10^{-15}	1.8067×10^{-15}		ZGJO	**−3.3220**	−3.2426	5.9086×10^{-2}
	IWKGJO	$\mathbf{4.4409 \times 10^{-16}}$	$\mathbf{4.4409 \times 10^{-16}}$	**0**		IWKGJO	**−3.3220**	**−3.2467**	$\mathbf{5.8273 \times 10^{-2}}$

Significant values are in bold.

However, the multi-peak function demonstrates superior improvement with the implementation of the vertical and horizontal crossover strategy (KGJO). Thus, the integration of the three improvement methodologies has resulted in IWKGJO achieving enhanced optimization accuracy and faster convergence speed. Within the F8–F13 functions, IWKGJO demonstrates superior performance in terms of optimal value, mean, and standard deviation compared to both of the other strategy enhancements and the original method. In the F9–F11 functions, all methods except KGJO and QGJO, including the original algorithm, achieve the theoretical optimum. This suggests that the additional strategies proposed in this study maintain the algorithm's capacity to explore and escape from local optima.

The IWKGJO algorithm demonstrates proximity between its optimal and average adaptation values in the fixed dimensional function. Additionally, the convergence accuracies of the standard deviation surpass those of the improved and original algorithms for each strategy. This strongly suggests that the improved algorithm possesses the capability to escape local optima. To summarize, the enhanced algorithm exhibits superior optimization accuracy, enhanced development capability, and the capacity to escape local optima.

Figure 2 demonstrates that the IQWKGJO algorithm produces a linear image in functions F1–F7, F9–F13, and F17–F18. This suggests that the proposed approach efficiently identifies the global optimum with great precision. In the F8 function, the IWKGJO picture exhibits several points of inflection, suggesting that the suggested algorithm has a superior ability to escape local optima compared to the original approach. In the F14–F16 and F19–F20 function images, it is evident that the IWKGJO algorithm achieves the global optimum with the highest speed. Among all the function images, the IWKGJO method demonstrates the highest level of accuracy.



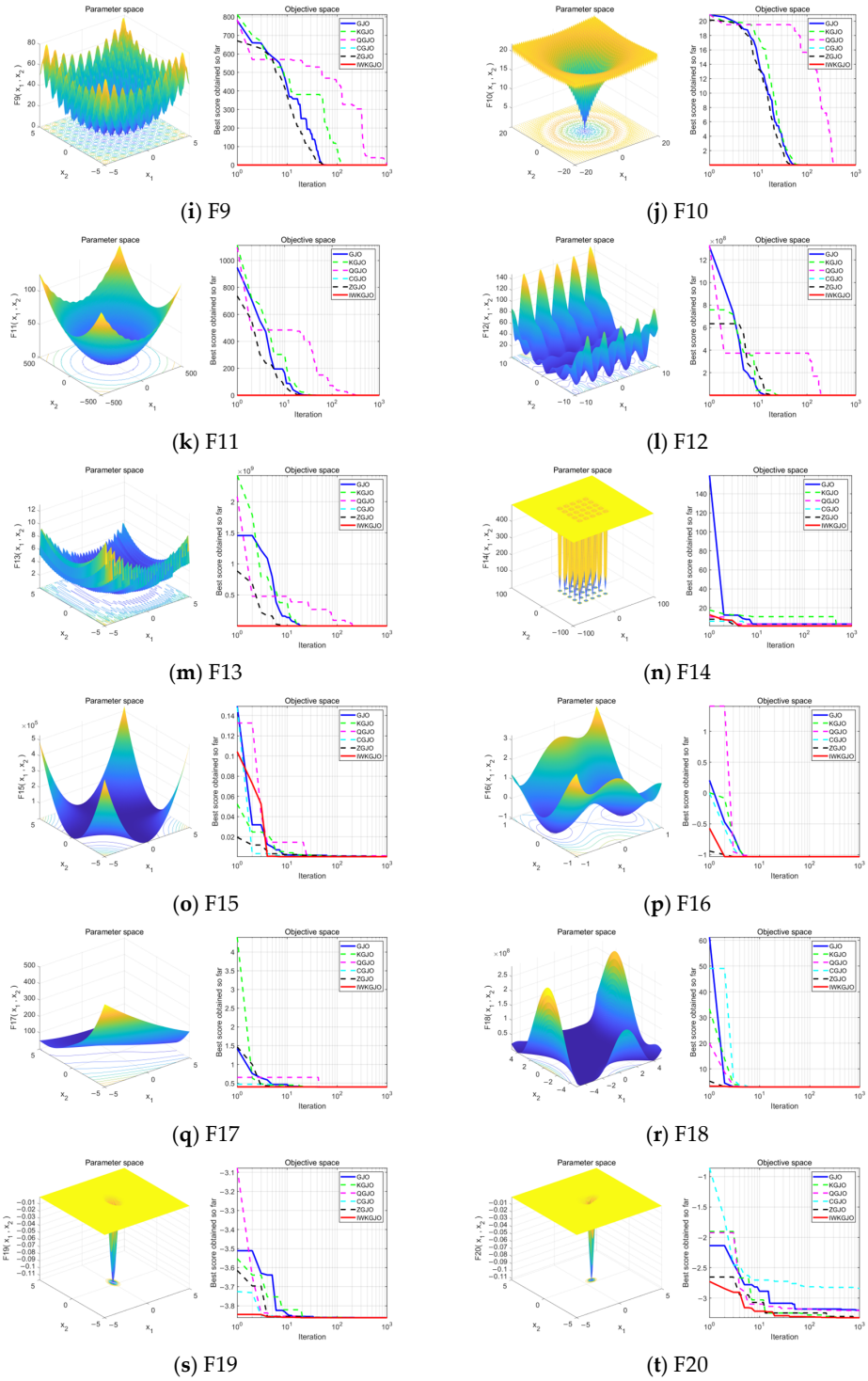
(**a**) F1 (**b**) F2

(**c**) F3 (**d**) F4

(**e**) F5 (**f**) F6

(**g**) F7 (**h**) F8

Figure 2. *Cont.*



(**i**) F9 (**j**) F10

(**k**) F11 (**l**) F12

(**m**) F13 (**n**) F14

(**o**) F15 (**p**) F16

(**q**) F17 (**r**) F18

(**s**) F19 (**t**) F20

Figure 2. Convergence curves of benchmark functions for different strategies.

To summarize, differential crossover is crucial for enhancing the algorithm's accuracy, whereas vertical crossover allows the system to escape local optimal validity. Hence, the amalgamation of the four methodologies can be productive in enhancing accuracy, thereby increasing the likelihood of discovering the global optimum.

5. Comparison of IWKGJO with Other Intelligent Algorithms

5.1. CEC2017 Test Function Basic Information

This study utilizes the CEC2017 test function to evaluate the optimization capabilities of the enhanced IWKGJO. It compares IWKGJO with three newly proposed original algorithms and four excellent improved algorithms. The comparison algorithms are respectively Gorilla Force Optimization Algorithm (GTO) [22], Walrus Optimizer (WO) [23] and Marine Predator Algorithm (MPA) [24], Hybrid Multi-Strategy Improved Wild Horse Optimizer (HMSWHO) [25], Grey Wolf Optimization Algorithm Based on Elite Learning of Nonlinear Parameters (IGWO) [26], Improving the search performance of SHADE using linear population size reduction (LSHADE) [27] and Whale Optimization Algorithm based on Elite Opposition-based and Crisscross Optimization (ECWOA) [28]. The remaining parameters remain unchanged from the aforementioned. Assign a population size of $N = 50$, and divide the dimensions into 50 and 100. The maximum number of iterations is $T = 1000$, and the process is repeated 30 times. The optimal value, average value, and standard deviation are then used for comparison. The bold represents the optimal outcome of the function. Table 6 presents the fundamental details of CEC2017.

Table 6. Basic information of CEC2017 test function.

	No.	Functions	F_i^*
Unimodal Function	1	Shifted and Rotated Zakharov Function	100
	3	Shifted and Rotated Rosenbrock's Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Rastrigin's Function	500
	6	Shifted and Rotated Expanded Schaffer's f6 Function	600
	7	Shifted and Rotated Lunacek Bi-Rastrigin's Function	700
	8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	9	Shifted and Rotated Levy Function	900
	10	Shifted and Rotated Schwefel's Function	1000
Hybrid Functions	11	Hybrid Function l (N = 3)	1100
	12	Hybrid Function 2 (N = 3)	1200
	13	Hybrid Function 3 (N = 3)	1300
	14	Hybrid Function 4 (N = 4)	1400
	15	Hybrid Function 5 (N = 4)	1500
	16	Hybrid Function 6 (N = 4)	1600
	17	Hybrid Function 6 (N = 5)	1700
	18	Hybrid Function 6 (N = 5)	1800
	19	Hybrid Function 6 (N = 5)	1900
	20	Hybrid Function 6 (N = 6)	2000
Composition Functions	21	Composition Function 1 (N = 3)	2100
	22	Composition Function 2 (N = 3)	2200
	23	Composition Function 3 (N = 4)	2300
	24	Composition Function 4 (N = 4)	2400
	25	Composition Function 5 (N = 5)	2500
	26	Composition Function 6 (N = 5)	2600
	27	Composition Function 7 (N = 6)	2700
	28	Composition Function 8 (N = 6)	2800
	29	Composition Function 9 (N = 3)	2900
	30	Composition Function 10 (N = 3)	3000

5.2. Simulation Results and Analysis

The CEC2017 test functions are divided into 29 groups. The functions in groups F1–F3 have a single peak and no local minima. These functions are suitable for evaluating the performance of optimization algorithms. On the other hand, the functions in groups F4–F10 are basic multi-peak functions. They are commonly used to challenge optimization algorithms. F11–F20 represents hybrid functions that are well-suited for complicated issues

characterized by a significant number of local minima. On the other hand, F21–F30 are combinatorial functions that are appropriate for situations exhibiting a high degree of oscillatory characteristics.

Based on the data in Table 7, it is evident that the IWKGJO algorithm is generally more suitable for simple multi-peak and combined functions. However, for mixed functions F12–F15 and F18–F19, the IWKGJO algorithm performs slightly worse than the GTO, HMSWHO, and MPA algorithms, although it still has an advantage over the other algorithms. Regarding single-peak functions, the IWKGJO algorithm offers minimal benefits compared to top-performing algorithms like MPA and HMSWHO. In terms of F1, IWKGJO is inferior to GTO, HMSWHO, and MPA, ranking 4th. Similarly, in F2, IWKGJO is likewise inferior to HMSWHO and MPA, ranking 3rd. In functions F4–F6 and F10, the IWKGJO algorithm outperforms other algorithms, such as MPA, in terms of optimal and average fitness, as well as the standard deviation. However, in functions F9, F7, and F8, the average fitness values of IWKGJO are slightly inferior to the second-place MPA algorithm. Only function F16 exhibits a higher standard deviation among the hybrid functions F11, F16, F17, and F20. However, the remaining values for the other functions are superior to both the other algorithms and the original algorithm. The combined functions F21, F23–F25, F28, and F30 exhibit advantages in both fitness value and standard deviation. However, in function F22, the IWKGJO algorithm is expected to perform somewhat worse compared to the MPA, HMSWHO, and IGWO algorithms. In functions F26, F27, and F29, the ideal fitness value exhibits an advantage, while the average fitness value and the standard deviation show a modest decline. However, the difference is not statistically significant. To summarize, the suggested approach maintains an edge in 50 dimensional functions. Although its performance may be slightly inferior to that of MPA, it excels in terms of search capability and has an advantage in highly oscillatory issues.

Table 7. Comparison of test function results of different CEC2017 algorithms ($d = 50$).

		IWKGJO	GJO	GTO	WO	MPA	HMSWHO	IGWO	LSHADE	ECWOA
F1	Best	9.0463×10^{5}	1.6669×10^{10}	$\mathbf{1.0795 \times 10^{3}}$	1.0393×10^{8}	1.7141×10^{5}	1.5621×10^{3}	9.3991×10^{10}	1.1041×10^{10}	7.0455×10^{4}
	Mean	3.0792×10^{6}	3.2808×10^{10}	2.0759×10^{4}	2.0102×10^{8}	1.4278×10^{6}	$\mathbf{1.9283 \times 10^{4}}$	1.0879×10^{11}	2.1737×10^{10}	9.2820×10^{6}
	Std	1.8456×10^{6}	9.5556×10^{9}	$\mathbf{2.0067 \times 10^{4}}$	7.4164×10^{7}	1.2571×10^{6}	2.1870×10^{4}	6.7432×10^{9}	5.9608×10^{9}	1.1285×10^{7}
	t/s	1.3191	0.5499	0.6981	1.4735	0.4227	1.0043	0.7457	0.3162	0.9686
F3	Best	1.0758×10^{4}	1.0115×10^{5}	1.5685×10^{4}	8.2172×10^{4}	$\mathbf{4.1250 \times 10^{3}}$	5.9036×10^{4}	1.7009×10^{5}	1.8569×10^{5}	4.2823×10^{4}
	Mean	1.6821×10^{4}	1.2081×10^{5}	2.5581×10^{4}	1.2506×10^{5}	$\mathbf{7.8386 \times 10^{3}}$	9.6063×10^{4}	4.0739×10^{5}	3.0116×10^{5}	7.0209×10^{4}
	Std	4.1656×10^{3}	1.2735×10^{4}	5.4517×10^{3}	1.6097×10^{4}	$\mathbf{3.0419 \times 10^{3}}$	1.8156×10^{4}	4.7125×10^{5}	7.8054×10^{4}	1.2413×10^{4}
	t/s	1.2789	0.5421	0.6841	1.4476	0.4117	1.0031	0.7484	0.3153	0.9506
F4	Best	$\mathbf{4.2519 \times 10^{2}}$	2.4904×10^{3}	4.5610×10^{2}	6.2646×10^{2}	4.3514×10^{2}	4.6919×10^{2}	2.7554×10^{4}	1.9394×10^{3}	4.9424×10^{2}
	Mean	$\mathbf{5.3706 \times 10^{2}}$	4.8272×10^{3}	5.7271×10^{2}	7.1164×10^{2}	5.4328×10^{2}	5.4176×10^{2}	3.4065×10^{4}	3.3659×10^{3}	6.0163×10^{2}
	Std	$\mathbf{5.0131 \times 10^{1}}$	1.6926×10^{3}	6.3866×10^{1}	5.4987×10^{1}	5.8681×10^{1}	5.4081×10^{1}	3.0100×10^{3}	1.0201×10^{3}	6.4254×10^{1}
	t/s	1.3102	0.5484	0.6986	1.4721	0.4144	1.0050	0.7174	0.3167	0.9721
F5	Best	$\mathbf{6.3470 \times 10^{2}}$	8.1728×10^{2}	7.8157×10^{2}	6.9640×10^{2}	6.6152×10^{2}	6.7200×10^{2}	1.1420×10^{3}	9.5044×10^{2}	7.4991×10^{2}
	Mean	$\mathbf{7.0420 \times 10^{2}}$	8.8962×10^{2}	8.3928×10^{2}	7.7338×10^{2}	7.2060×10^{2}	7.4679×10^{2}	1.2196×10^{3}	1.0318×10^{3}	8.0570×10^{2}
	Std	$\mathbf{2.4617 \times 10^{1}}$	6.2626×10^{1}	2.5906×10^{1}	7.3438×10^{1}	3.0675×10^{1}	3.9268×10^{1}	2.6447×10^{1}	3.6682×10^{1}	4.1471×10^{1}
	t/s	1.6195	0.6422	0.9172	1.8415	0.5044	1.2235	0.8332	0.4145	1.2695
F6	Best	$\mathbf{6.0001 \times 10^{2}}$	6.4045×10^{2}	6.4294×10^{2}	6.2989×10^{2}	6.0516×10^{2}	6.0292×10^{2}	7.0873×10^{2}	6.3419×10^{2}	6.0014×10^{2}
	Mean	$\mathbf{6.0007 \times 10^{2}}$	6.5069×10^{2}	6.5210×10^{2}	6.6517×10^{2}	6.0941×10^{2}	6.3001×10^{2}	7.1215×10^{2}	6.5527×10^{2}	6.0024×10^{2}
	Std	$\mathbf{1.0173 \times 10^{-1}}$	6.1580	6.5428	1.7948×10^{1}	2.5177	1.2609×10^{1}	3.2831	9.6697	1.0243×10^{-1}
	t/s	2.4045	0.9125	1.5844	2.8976	0.7799	1.7327	1.1044	0.6792	2.1200
F7	Best	**4.7088**	1.2409×10^{3}	1.2282×10^{3}	1.0506×10^{3}	9.0785×10^{2}	1.0520×10^{3}	1.9982×10^{3}	1.3965×10^{3}	1.0778×10^{3}
	Mean	1.0208×10^{3}	1.3925×10^{3}	1.4041×10^{3}	1.1450×10^{3}	$\mathbf{9.7393 \times 10^{2}}$	1.1542×10^{3}	2.0317×10^{3}	1.5786×10^{3}	1.2063×10^{3}
	Std	$\mathbf{2.1693 \times 10^{1}}$	9.5878	1.1905×10^{2}	5.7074×10^{1}	4.1350	8.2664	2.8029	9.7949	1.0471×10^{2}
	t/s	1.6413	0.6592	0.9722	1.9045	0.5224	1.2242	0.8429	0.4279	1.3043
F8	Best	$\mathbf{9.4102 \times 10^{2}}$	1.1263×10^{3}	9.9004×10^{2}	9.9615×10^{2}	9.5731×10^{2}	9.6844×10^{2}	1.4963×10^{3}	1.2370×10^{3}	1.0432×10^{3}
	Mean	1.0037×10^{3}	1.2186×10^{3}	1.1517×10^{3}	1.0597×10^{3}	$\mathbf{9.9854 \times 10^{2}}$	1.0306×10^{3}	1.5498×10^{3}	1.3396×10^{3}	1.1041×10^{3}
	Std	$\mathbf{2.4672 \times 10^{1}}$	6.7345	6.1649	7.2599	2.5994	4.8850	3.3376	5.0773	4.7526
	t/s	1.6389	0.6638	0.9669	1.9055	0.5313	1.2234	0.8334	0.4262	1.2965
F9	Best	3.9955×10^{3}	1.1299×10^{4}	6.4469×10^{3}	7.0336×10^{3}	$\mathbf{1.6926 \times 10^{3}}$	6.1440×10^{3}	3.7591×10^{4}	1.2764×10^{4}	6.0161×10^{3}
	Mean	5.9393×10^{3}	1.9390×10^{4}	1.0309×10^{4}	1.9781×10^{4}	$\mathbf{2.7057 \times 10^{3}}$	9.0932×10^{3}	4.2501×10^{4}	1.9099×10^{4}	9.5591×10^{3}
	Std	1.3458×10^{3}	4.7790×10^{3}	2.2490×10^{3}	9.8544×10^{3}	$\mathbf{9.5081 \times 10^{2}}$	2.1662×10^{3}	2.7480×10^{3}	4.9928×10^{3}	2.2255×10^{3}
	t/s	1.6325	0.6541	0.9749	2.0241	0.5948	1.3723	0.8833	0.4248	1.2922
F10	Best	$\mathbf{5.3968 \times 10^{3}}$	8.0744×10^{3}	6.4431×10^{3}	6.5572×10^{3}	5.7175×10^{3}	6.3415×10^{3}	1.5617×10^{4}	1.3458×10^{4}	6.0888×10^{3}
	Mean	$\mathbf{6.2244 \times 10^{3}}$	1.0705×10^{4}	9.1619×10^{3}	1.3237×10^{4}	6.5286×10^{3}	7.2987×10^{3}	1.6333×10^{4}	1.5465×10^{4}	6.8626×10^{3}
	Std	$\mathbf{5.0021 \times 10^{2}}$	2.4121×10^{3}	1.7483×10^{3}	3.2419×10^{3}	6.2035×10^{2}	6.0926×10^{2}	5.0176×10^{2}	6.7141×10^{2}	5.2770×10^{2}
	t/s	1.8293	0.7182	1.1184	2.1437	0.5961	1.3444	0.9210	0.4836	1.4803

Table 7. *Cont.*

		IWKGJO	GJO	GTO	WO	MPA	HMSWHO	IGWO	LSHADE	ECWOA
F11	Best	1.2038×10^{3}	3.6411×10^{3}	1.2650×10^{3}	1.5105×10^{3}	1.2166×10^{3}	$\mathbf{1.1924 \times 10^{3}}$	2.0094×10^{4}	7.9961×10^{3}	1.2681×10^{3}
	Mean	$\mathbf{1.2614 \times 10^{3}}$	7.2743×10^{3}	1.3206×10^{3}	1.8462×10^{3}	1.2624×10^{3}	1.3328×10^{3}	2.5647×10^{4}	1.6619×10^{4}	1.8272×10^{3}
	Std	$\mathbf{3.4663 \times 10^{1}}$	2.5070×10^{3}	4.2685×10^{1}	3.4050×10^{2}	3.6797×10^{1}	1.1186×10^{2}	2.0784×10^{3}	6.1089×10^{3}	7.0021×10^{2}
	t/s	1.4526	0.5900	0.8093	1.6572	0.4661	1.0954	0.7693	0.3607	1.1087
F12	Best	8.3072×10^{6}	2.0214×10^{9}	8.9842×10^{5}	2.6431×10^{7}	1.5012×10^{6}	$\mathbf{6.7615 \times 10^{5}}$	6.6204×10^{10}	1.1082×10^{9}	3.3575×10^{6}
	Mean	1.6988×10^{7}	8.5739×10^{9}	4.7225×10^{6}	9.5030×10^{7}	4.7154×10^{6}	$\mathbf{2.4882 \times 10^{6}}$	7.9638×10^{10}	2.3986×10^{9}	1.0487×10^{7}
	Std	7.7613×10^{6}	6.1587×10^{9}	4.1011×10^{6}	4.9503×10^{7}	2.4428×10^{6}	$\mathbf{1.9673 \times 10^{6}}$	7.8141×10^{9}	7.6171×10^{8}	5.2621×10^{6}
	t/s	1.6569	0.6538	0.9556	1.8999	0.5276	1.2125	0.8323	0.4209	1.2753
F13	Best	1.1970×10^{4}	2.7781×10^{8}	5.4944×10^{3}	4.5000×10^{4}	4.6831×10^{3}	$\mathbf{2.8218 \times 10^{3}}$	2.5425×10^{10}	7.3164×10^{7}	3.8136×10^{4}
	Mean	3.5157×10^{4}	1.0414×10^{9}	1.5838×10^{4}	3.8995×10^{5}	$\mathbf{9.7993 \times 10^{3}}$	1.0179×10^{4}	4.9029×10^{10}	3.5606×10^{8}	2.8642×10^{5}
	Std	1.7324×10^{4}	1.3020×10^{9}	1.1401×10^{4}	3.8474×10^{5}	$\mathbf{2.7257 \times 10^{3}}$	5.7046×10^{3}	1.2246×10^{10}	1.9336×10^{8}	4.7699×10^{5}
	t/s	1.4720	0.6004	0.8357	1.6986	0.4778	1.1158	0.8100	0.3713	1.1366
F14	Best	7.0917×10^{4}	3.8689×10^{5}	2.4182×10^{3}	1.8100×10^{5}	$\mathbf{1.5229 \times 10^{3}}$	9.9157×10^{3}	6.8478×10^{7}	1.0460×10^{5}	4.8750×10^{5}
	Mean	3.1436×10^{5}	1.5186×10^{6}	3.9588×10^{4}	1.2607×10^{6}	$\mathbf{1.5533 \times 10^{3}}$	1.4264×10^{5}	1.5211×10^{8}	2.6507×10^{6}	2.4403×10^{6}
	Std	1.9094×10^{5}	1.3568×10^{6}	3.6145×10^{4}	9.0787×10^{5}	$\mathbf{1.6202 \times 10^{1}}$	1.3549×10^{5}	6.3052×10^{7}	2.9453×10^{6}	1.5618×10^{6}
	t/s	1.7982	0.7087	1.0958	2.1291	0.5845	1.3400	0.9144	0.4773	1.4648
F15	Best	3.6090×10^{3}	3.8639×10^{5}	2.0134×10^{3}	3.2248×10^{4}	1.8178×10^{3}	$\mathbf{1.8164 \times 10^{3}}$	4.6091×10^{9}	6.8050×10^{6}	1.2698×10^{4}
	Mean	1.9176×10^{4}	1.9539×10^{8}	1.2382×10^{4}	7.8327×10^{4}	$\mathbf{2.0072 \times 10^{3}}$	1.4458×10^{4}	7.7455×10^{9}	4.1004×10^{7}	3.3390×10^{4}
	Std	5.4513×10^{3}	2.7259×10^{8}	8.8410×10^{3}	4.7077×10^{4}	$\mathbf{1.1753 \times 10^{2}}$	6.7220×10^{3}	2.6981×10^{9}	2.1198×10^{7}	3.2839×10^{4}
	t/s	1.4191	0.5802	0.7846	1.6259	0.4630	1.0890	0.7905	0.3567	1.0866
F16	Best	$\mathbf{2.2118 \times 10^{3}}$	2.9800×10^{3}	3.0829×10^{3}	2.8314×10^{3}	2.3110×10^{3}	2.4807×10^{3}	9.4060×10^{3}	5.2187×10^{3}	3.1477×10^{3}
	Mean	$\mathbf{2.1209 \times 10^{3}}$	3.7967×10^{3}	3.7032×10^{3}	4.0699×10^{3}	2.7595×10^{3}	3.0673×10^{3}	1.0534×10^{4}	5.8882×10^{3}	3.8333×10^{3}
	Std	4.7649×10^{2}	5.1805×10^{2}	4.5914×10^{2}	1.0762×10^{3}	$\mathbf{2.1864 \times 10^{2}}$	3.0671×10^{2}	8.3614×10^{2}	3.6185×10^{2}	5.7287×10^{2}
	t/s	1.5601	0.6319	0.8916	1.8001	0.5004	1.1692	0.8316	0.3942	1.2165
F17	Best	$\mathbf{2.2317 \times 10^{3}}$	2.8741×10^{3}	2.5077×10^{3}	2.8736×10^{3}	2.2389×10^{3}	2.2472×10^{3}	6.6602×10^{3}	3.5869×10^{3}	2.6908×10^{3}
	Mean	$\mathbf{2.5841 \times 10^{3}}$	3.5782×10^{3}	3.4295×10^{3}	3.3409×10^{3}	2.6537×10^{3}	2.9058×10^{3}	1.0704×10^{4}	4.6799×10^{3}	3.3250×10^{3}
	Std	$\mathbf{2.2904 \times 10^{2}}$	4.0659×10^{2}	4.2668×10^{2}	6.1293×10^{2}	2.3105×10^{2}	2.3846×10^{2}	3.5178×10^{3}	3.6121×10^{2}	2.6595×10^{2}
	t/s	2.1308	0.8227	1.3607	2.5477	0.6961	1.5470	0.9957	0.5857	1.7761
F18	Best	2.1326×10^{5}	1.7472×10^{6}	4.2110×10^{4}	9.1501×10^{5}	$\mathbf{2.1704 \times 10^{3}}$	1.0993×10^{5}	7.1931×10^{7}	2.6043×10^{6}	5.5952×10^{5}
	Mean	1.9983×10^{6}	1.3278×10^{7}	1.8078×10^{5}	3.7695×10^{6}	$\mathbf{2.4129 \times 10^{3}}$	5.6632×10^{5}	1.9728×10^{8}	1.6135×10^{7}	3.2605×10^{6}
	Std	1.2979×10^{6}	1.7160×10^{7}	1.9156×10^{5}	3.4051×10^{6}	$\mathbf{1.9042 \times 10^{2}}$	3.3810×10^{5}	6.8037×10^{7}	1.1268×10^{7}	2.5309×10^{6}
	t/s	1.5598	0.6313	0.8957	1.7794	0.5013	1.1830	0.8311	0.3926	1.2245
F19	Best	2.4754×10^{3}	1.0859×10^{5}	2.8628×10^{3}	1.0638×10^{4}	$\mathbf{1.9943 \times 10^{3}}$	1.6971×10^{4}	2.3915×10^{9}	4.8705×10^{6}	5.6209×10^{3}
	Mean	2.1151×10^{4}	1.9724×10^{8}	1.3460×10^{4}	8.4600×10^{4}	$\mathbf{2.0223 \times 10^{3}}$	2.7769×10^{4}	4.8743×10^{9}	2.3098×10^{7}	2.2200×10^{4}
	Std	1.4263×10^{4}	4.4687×10^{8}	1.1134×10^{4}	6.2239×10^{4}	$\mathbf{1.7029 \times 10^{1}}$	8.1173×10^{3}	1.4672×10^{9}	1.5489×10^{7}	1.1345×10^{4}
	t/s	5.4085	1.8981	4.1073	6.9242	1.7828	3.7253	2.1136	1.7146	5.0861
F20	Best	$\mathbf{2.3296 \times 10^{3}}$	2.9282×10^{3}	2.6224×10^{3}	2.6361×10^{3}	2.3362×10^{3}	2.5543×10^{3}	3.9464×10^{3}	3.9828×10^{3}	2.7599×10^{3}
	Mean	$\mathbf{2.6351 \times 10^{3}}$	3.3335×10^{3}	3.2302×10^{3}	3.6829×10^{3}	2.6519×10^{3}	2.9942×10^{3}	4.5547×10^{3}	4.5267×10^{3}	3.2948×10^{3}
	Std	$\mathbf{1.4464 \times 10^{2}}$	3.9500×10^{2}	3.3315×10^{2}	6.5022×10^{2}	1.5138×10^{2}	2.4232×10^{2}	2.2185×10^{2}	2.4719×10^{2}	2.6653×10^{2}
	t/s	2.2304	0.8564	1.4519	2.6785	0.7297	1.6130	1.0571	0.6237	1.8776
F21	Best	$\mathbf{2.4047 \times 10^{3}}$	2.6020×10^{3}	2.5386×10^{3}	2.4595×10^{3}	2.4093×10^{3}	2.4165×10^{3}	3.1658×10^{3}	2.7671×10^{3}	2.5250×10^{3}
	Mean	$\mathbf{2.4499 \times 10^{3}}$	2.6932×10^{3}	2.6121×10^{3}	2.5331×10^{3}	2.4525×10^{3}	2.4822×10^{3}	3.2630×10^{3}	2.8366×10^{3}	2.6182×10^{3}
	Std	$\mathbf{2.0624 \times 10^{1}}$	7.8858×10^{1}	4.2862×10^{1}	8.8153×10^{1}	2.0797×10^{1}	4.8060×10^{1}	4.7107×10^{1}	3.6335×10^{1}	6.1677×10^{1}
	t/s	3.5360	1.2913	2.5302	4.3931	1.1468	2.4650	1.4890	1.0959	3.1554
F22	Best	7.1281×10^{3}	8.7209×10^{3}	7.9191×10^{3}	2.3980×10^{3}	2.3104×10^{3}	$\mathbf{2.3032 \times 10^{3}}$	1.7567×10^{4}	1.4765×10^{4}	7.6571×10^{3}
	Mean	8.1193×10^{3}	1.2708×10^{4}	1.1057×10^{4}	1.3932×10^{4}	$\mathbf{3.1585 \times 10^{3}}$	9.0227×10^{3}	1.8221×10^{4}	1.7161×10^{4}	8.9307×10^{3}
	Std	7.2834×10^{2}	2.6874×10^{3}	1.8606×10^{3}	4.5156×10^{3}	2.2289×10^{3}	1.9243×10^{3}	$\mathbf{3.7787 \times 10^{2}}$	9.1306×10^{2}	6.1042×10^{2}
	t/s	3.8623	1.4046	2.8037	4.8571	1.2706	2.6990	1.5912	1.1719	3.4953
F23	Best	$\mathbf{2.8053 \times 10^{3}}$	3.1501×10^{3}	2.9703×10^{3}	2.9116×10^{3}	2.8145×10^{3}	2.8977×10^{3}	4.7036×10^{3}	3.2784×10^{3}	3.0192×10^{3}
	Mean	$\mathbf{2.8475 \times 10^{3}}$	3.2380×10^{3}	3.2200×10^{3}	3.0226×10^{3}	2.8682×10^{3}	2.9815×10^{3}	5.0746×10^{3}	3.3691×10^{3}	3.1096×10^{3}
	Std	$\mathbf{3.2418 \times 10^{1}}$	6.9694×10^{1}	1.3845×10^{2}	1.0913×10^{2}	3.4815×10^{1}	6.1647×10^{1}	2.1403×10^{2}	4.8520×10^{1}	5.3600×10^{1}
	t/s	4.5318	1.6227	3.3197	5.7083	1.4784	3.1076	1.7968	1.3782	4.1751
F24	Best	$\mathbf{2.9549 \times 10^{3}}$	3.3245×10^{3}	3.1932×10^{3}	3.0474×10^{3}	2.9760×10^{3}	3.0535×10^{3}	4.9138×10^{3}	3.4048×10^{3}	3.4027×10^{3}
	Mean	$\mathbf{3.0304 \times 10^{3}}$	3.4988×10^{3}	3.4016×10^{3}	3.1150×10^{3}	3.0322×10^{3}	3.1420×10^{3}	5.6254×10^{3}	3.4770×10^{3}	3.5678×10^{3}
	Std	$\mathbf{3.8042 \times 10^{1}}$	1.2888×10^{2}	1.6726×10^{2}	4.0287×10^{1}	4.4358×10^{1}	6.1225×10^{1}	3.4403×10^{2}	5.0475×10^{1}	1.0344×10^{2}
	t/s	6.7833	2.4603	5.0180	8.9431	2.3428	4.8707	2.6915	2.2053	6.4955
F25	Best	$\mathbf{3.0002 \times 10^{3}}$	4.2210×10^{3}	3.0013×10^{3}	3.1254×10^{3}	3.0313×10^{3}	3.0550×10^{3}	1.4066×10^{4}	3.9803×10^{3}	3.0131×10^{3}
	Mean	$\mathbf{3.0613 \times 10^{3}}$	5.5218×10^{3}	3.0887×10^{3}	3.1922×10^{3}	3.0689×10^{3}	3.1096×10^{3}	1.5224×10^{4}	5.2904×10^{3}	3.0935×10^{3}
	Std	$\mathbf{2.2342 \times 10^{1}}$	8.4142×10^{2}	3.5406×10^{1}	4.7673×10^{1}	2.2598×10^{1}	2.3423×10^{1}	6.2047×10^{2}	6.1220×10^{2}	3.6805×10^{1}
	t/s	4.6631	1.6733	3.4719	5.9625	1.5371	3.2315	1.8659	1.4409	4.2996
F26	Best	$\mathbf{2.9041 \times 10^{3}}$	8.1794×10^{3}	3.3578×10^{3}	3.7144×10^{3}	2.9086×10^{3}	2.9131×10^{3}	1.6325×10^{4}	8.7666×10^{3}	6.1683×10^{3}
	Mean	5.4481×10^{3}	9.3253×10^{3}	8.8962×10^{3}	4.6517×10^{3}	3.7680×10^{3}	$\mathbf{3.4607 \times 10^{3}}$	1.6964×10^{4}	1.0120×10^{4}	7.5403×10^{3}
	Std	1.4052×10^{3}	9.0897×10^{2}	2.4725×10^{3}	1.6146×10^{3}	9.8881×10^{2}	1.4826×10^{3}	$\mathbf{4.8056 \times 10^{2}}$	6.5504×10^{2}	8.8361×10^{2}
	t/s	5.5530	1.9554	4.1585	7.0937	1.8277	3.8485	2.1451	1.7258	5.1514
F27	Best	$\mathbf{3.2738 \times 10^{3}}$	3.7709×10^{3}	3.4017×10^{3}	3.4045×10^{3}	3.2809×10^{3}	3.3448×10^{3}	7.3864×10^{3}	3.8236×10^{3}	3.3767×10^{3}
	Mean	$\mathbf{3.3078 \times 10^{3}}$	4.0655×10^{3}	3.8134×10^{3}	3.5217×10^{3}	3.3544×10^{3}	3.4974×10^{3}	8.3542×10^{3}	4.0512×10^{3}	3.5453×10^{3}
	Std	7.1269×10^{1}	2.2359×10^{2}	2.6643×10^{2}	7.1674×10^{1}	$\mathbf{4.5205 \times 10^{1}}$	1.1223×10^{2}	5.4250×10^{2}	1.8233×10^{2}	1.0115×10^{2}
	t/s	6.3660	2.2542	4.9064	8.2383	2.1249	4.3831	2.3978	2.0268	5.9839
F28	Best	$\mathbf{3.2651 \times 10^{3}}$	4.8408×10^{3}	3.3118×10^{3}	3.3721×10^{3}	3.3053×10^{3}	3.2902×10^{3}	1.0960×10^{4}	4.0359×10^{3}	3.2664×10^{3}
	Mean	$\mathbf{3.3143 \times 10^{3}}$	5.5634×10^{3}	3.3618×10^{3}	3.5223×10^{3}	3.3383×10^{3}	3.3752×10^{3}	1.3068×10^{4}	6.1123×10^{3}	3.3440×10^{3}
	Std	$\mathbf{3.0613 \times 10^{1}}$	4.6943×10^{2}	4.2438×10^{1}	8.3138×10^{1}	3.9390×10^{1}	5.3383×10^{1}	7.7813×10^{2}	8.4144×10^{2}	3.5218×10^{1}
	t/s	5.7216	2.0239	4.3393	7.4185	1.8814	3.9100	2.2084	1.7891	5.3396
F29	Best	$\mathbf{3.5441 \times 10^{3}}$	4.9229×10^{3}	4.5204×10^{3}	3.8887×10^{3}	3.6547×10^{3}	3.7875×10^{3}	3.7405×10^{4}	5.9299×10^{3}	3.8408×10^{3}
	Mean	4.2481×10^{3}	5.6041×10^{3}	5.4070×10^{3}	4.4201×10^{3}	$\mathbf{4.0061 \times 10^{3}}$	4.2100×10^{3}	1.0594×10^{5}	7.0568×10^{3}	4.5777×10^{3}
	Std	$\mathbf{2.7583 \times 10^{2}}$	5.0197×10^{2}	7.1710×10^{2}	3.6797×10^{2}	3.0397×10^{2}	2.8625×10^{2}	1.0381×10^{5}	6.4330×10^{2}	3.8026×10^{2}
	t/s	3.9074	1.4061	2.7650	4.8320	1.2665	2.7087	1.5773	1.1764	3.4684
F30	Best	$\mathbf{6.2539 \times 10^{5}}$	1.5538×10^{8}	6.3196×10^{5}	6.3756×10^{6}	1.1326×10^{6}	8.3592×10^{5}	6.5318×10^{9}	8.6310×10^{7}	7.5339×10^{5}
	Mean	$\mathbf{1.0934 \times 10^{6}}$	4.3627×10^{8}	1.3104×10^{6}	1.0424×10^{7}	2.5348×10^{6}	1.2284×10^{6}	8.5402×10^{9}	2.4075×10^{8}	1.1211×10^{6}
	Std	$\mathbf{2.4110 \times 10^{5}}$	2.7906×10^{8}	3.7166×10^{5}	2.6825×10^{6}	1.1377×10^{6}	3.4177×10^{5}	1.2812×10^{9}	8.4172×10^{7}	2.9649×10^{5}
	t/s	7.2524	2.5370	5.5857	9.4007	2.4286	4.9251	2.6782	2.3090	6.8350

Significant values are in bold.

Table 8 demonstrates that the results in 100 dimensions are nearly identical to those in 50 dimensions. Specifically, in the single-peak function, the optimal fitness value of the F1 function is slightly inferior to that of the ECWOA algorithm. However, the average fitness value and standard deviation of the F1 function are superior to those of the other algorithms. The F3 function demonstrates superior performance of the teasel MPA, HMSWHO, and WO algorithms compared to other algorithms, respectively. The values of the F4, F5, F8, and F10 functions in the basic multi-peak function are higher compared to other algorithms. In function F6, the standard deviation is marginally inferior to ECWOA, respectively. In function F7, the IWKGJO algorithm has a lower average fitness value compared to the WO algorithm. Additionally, the standard deviation of the IWKGJO algorithm is slightly worse than that of the original strategy. However, the standard deviation of the original algorithm is better than that of all the comparative algorithms in this function. In the F9 function, the optimal fitness value is marginally inferior to that of the MPA, while the standard deviation is marginally inferior to that of GTO. The situation in the hybrid function is approximately identical to that achieved for the 50 dimensions. In function F12, the obtained results are inferior to those of HMSWHO and ECWOA, which rank third. The results achieved in functions F13 and F19 are inferior only to HMSWHO, which holds the second position. The results obtained in functions F14, F15, and F18 are ranked third, behind only MPA and HMSWHO. The results achieved in functions F13 and F19 are inferior only to HMSWHO, which holds the second position. The results obtained in functions F14, F15, and F18 are ranked third, behind only MPA and HMSWHO. Within function F11, the ideal fitness value surpasses all other algorithms, but the mean fitness value and standard deviation are only inferior to MPA, which ranks second. The standard deviation of F20 is marginally higher than MPA, while F16 exhibits no significant difference. In the context of the combined functions, the IWKGJO algorithm retains its superiority. Specifically, in functions F23, F25, F27, and F28, the algorithm consistently outperforms other algorithms, as seen by the superior results displayed in the table. In functions F21, F24, and F29, the only parameter that is inferior to MPA is the average fitness value, while the other parameters remain outstanding. Within the context of function F22, the optimal fitness value is marginally inferior to that of MPA. In function F26, the standard deviation is marginally inferior to MPA. In function F30, both the ideal value and the average value are somewhat worse than MPA, but still exhibit a significant advantage over other methods.

Table 8. Comparison of test function results of different CEC2017 algorithms ($d = 100$).

		IWKGJO	GJO	GTO	WO	MPA	HMSWHO	IGWO	LSHADE	ECWOA
F1	Best	1.0328×10^{6}	1.1464×10^{11}	4.0952×10^{7}	3.2592×10^{9}	1.1256×10^{8}	1.0836×10^{8}	9.0146×10^{10}	2.4457×10^{11}	$\mathbf{9.7509 \times 10^{5}}$
	Mean	$\mathbf{2.9702 \times 10^{6}}$	1.3000×10^{11}	1.1025×10^{8}	4.5338×10^{9}	2.0897×10^{8}	1.6468×10^{9}	1.1625×10^{11}	2.6093×10^{11}	7.7342×10^{6}
	Std	$\mathbf{1.3663 \times 10^{6}}$	8.8126×10^{9}	5.7354×10^{7}	6.7577×10^{8}	6.4094×10^{7}	2.0068×10^{9}	1.6145×10^{10}	7.5513×10^{9}	9.4625×10^{6}
	t/s	3.8411	1.4116	2.3915	4.4278	1.1964	2.6394	1.7949	1.0218	3.0836
F3	Best	1.5378×10^{5}	2.7706×10^{5}	1.2209×10^{5}	3.0076×10^{5}	$\mathbf{8.8854 \times 10^{4}}$	2.6820×10^{5}	3.3888×10^{5}	3.8477×10^{5}	3.3565×10^{5}
	Mean	2.0885×10^{5}	3.0246×10^{5}	1.4597×10^{5}	3.1961×10^{5}	1.1367×10^{5}	$\mathbf{3.5367 \times 10^{4}}$	3.6683×10^{5}	6.6406×10^{5}	4.7161×10^{5}
	Std	2.7372×10^{4}	1.6198×10^{4}	1.3167×10^{4}	$\mathbf{1.0559 \times 10^{4}}$	1.6173×10^{4}	3.1176×10^{5}	3.8815×10^{4}	1.7248×10^{5}	1.0493×10^{5}
	t/s	3.8386	1.4082	2.3795	4.4343	1.1968	2.6272	1.8006	1.0213	3.0976
F4	Best	$\mathbf{7.0252 \times 10^{2}}$	1.4473×10^{4}	9.5627×10^{2}	1.2104×10^{3}	7.3846×10^{2}	8.6656×10^{2}	9.0243×10^{4}	1.1703×10^{4}	7.3951×10^{2}
	Mean	$\mathbf{8.0525 \times 10^{2}}$	1.8553×10^{4}	1.0696×10^{3}	1.5557×10^{3}	9.1730×10^{2}	1.0195×10^{3}	1.0185×10^{5}	1.9306×10^{4}	8.0627×10^{2}
	Std	$\mathbf{5.0282 \times 10^{1}}$	3.9393×10^{3}	8.1954×10^{1}	1.6506×10^{2}	1.0977×10^{2}	9.3848×10^{1}	6.5493×10^{3}	3.8953×10^{3}	7.2703×10^{1}
	t/s	3.9384	1.4373	2.4573	4.5721	1.2225	2.6644	1.8167	1.0571	3.2160
F5	Best	$\mathbf{9.2271 \times 10^{2}}$	1.3492×10^{3}	1.2680×10^{3}	1.1783×10^{3}	9.6927×10^{2}	1.2054×10^{3}	2.0756×10^{3}	1.6535×10^{3}	1.1339×10^{3}
	Mean	$\mathbf{1.0597 \times 10^{3}}$	1.5300×10^{3}	1.3326×10^{3}	1.3366×10^{3}	1.1214×10^{3}	1.2766×10^{3}	2.1501×10^{3}	1.8107×10^{3}	1.2606×10^{3}
	Std	$\mathbf{4.2870 \times 10^{1}}$	1.1938×10^{2}	4.3059×10^{1}	1.5660×10^{2}	1.0502×10^{2}	7.8213×10^{1}	8.6274×10^{1}	7.1994×10^{1}	5.7453×10^{1}
	t/s	4.3417	1.6010	2.8543	5.1930	1.3911	3.0161	1.9879	1.2110	3.6721
F6	Best	$\mathbf{6.0006 \times 10^{2}}$	6.6105×10^{2}	6.5162×10^{2}	6.5025×10^{2}	6.1991×10^{2}	6.3592×10^{2}	7.1389×10^{2}	6.6255×10^{2}	6.1746×10^{2}
	Mean	$\mathbf{6.0119 \times 10^{2}}$	6.7034×10^{2}	6.6218×10^{2}	6.8432×10^{2}	6.2868×10^{2}	6.5334×10^{2}	7.1688×10^{2}	6.8207×10^{2}	6.2411×10^{2}
	Std	2.0490	9.0445	5.3307	2.3487×10^{1}	7.1319	8.6763	2.2880	7.2967	$\mathbf{5.6138 \times 10^{-1}}$
	t/s	5.9036	2.1113	4.1146	7.2466	1.8989	4.0827	2.5171	1.7076	5.2724
F7	Best	$\mathbf{1.5049 \times 10^{3}}$	2.6876×10^{3}	2.5909×10^{3}	2.2894×10^{3}	1.5904×10^{3}	1.8947×10^{3}	3.7427×10^{3}	3.0842×10^{3}	2.0339×10^{3}
	Mean	1.8614×10^{3}	2.8219×10^{3}	2.9025×10^{3}	2.4685×10^{3}	$\mathbf{1.7286 \times 10^{3}}$	2.1872×10^{3}	4.0057×10^{3}	3.5504×10^{3}	2.5420×10^{3}
	Std	1.2228×10^{2}	$\mathbf{7.9665 \times 10^{1}}$	1.8681×10^{2}	1.6785×10^{2}	1.0821×10^{2}	1.5073×10^{2}	9.6234×10^{1}	2.9631×10^{2}	3.3233×10^{2}
	t/s	4.3658	1.6206	2.8907	5.3062	1.4159	3.0623	2.0211	1.2263	3.6874
F8	Best	$\mathbf{1.2508 \times 10^{3}}$	1.6211×10^{3}	1.5819×10^{3}	1.3873×10^{3}	1.2540×10^{3}	1.4012×10^{3}	2.5291×10^{3}	2.0320×10^{3}	1.5212×10^{3}
	Mean	$\mathbf{1.3573 \times 10^{3}}$	1.8377×10^{3}	1.7497×10^{3}	1.6072×10^{3}	1.3875×10^{3}	1.5582×10^{3}	2.6178×10^{3}	2.1238×10^{3}	1.6782×10^{3}
	Std	$\mathbf{5.2782 \times 10^{1}}$	1.0394×10^{2}	8.5322×10^{1}	8.1083×10^{1}	9.3035×10^{1}	7.9106×10^{1}	7.5434×10^{1}	6.2390×10^{1}	7.7992×10^{1}
	t/s	4.4726	1.6580	2.9489	5.4159	1.4650	3.1256	2.0913	1.2723	3.8021

Table 8. *Cont.*

		IWKGJO	GJO	GTO	WO	MPA	HMSWHO	IGWO	LSHADE	ECWOA
F9	Best	1.5204×10^{4}	3.4759×10^{4}	1.9280×10^{4}	3.9495×10^{4}	$\mathbf{1.3323 \times 10^{4}}$	2.2834×10^{4}	7.9472×10^{4}	4.9151×10^{4}	1.9114×10^{4}
	Mean	$\mathbf{2.1076 \times 10^{4}}$	5.8248×10^{4}	2.3298×10^{4}	7.2546×10^{4}	2.1683×10^{4}	3.1270×10^{4}	8.5873×10^{4}	6.8740×10^{4}	2.7667×10^{4}
	Std	3.2460×10^{3}	1.2258×10^{4}	$\mathbf{2.2447 \times 10^{3}}$	1.4050×10^{4}	3.0181×10^{3}	3.6499×10^{3}	3.8457×10^{3}	9.9367×10^{3}	4.8167×10^{3}
	t/s	5.9221	2.2908	3.8096	7.0280	1.8861	4.2501	2.6134	1.6684	5.3159
F10	Best	$\mathbf{1.1836 \times 10^{4}}$	1.8562×10^{4}	1.3098×10^{4}	1.7295×10^{4}	1.3390×10^{4}	1.6220×10^{4}	3.1812×10^{4}	2.9871×10^{4}	1.2592×10^{4}
	Mean	$\mathbf{1.3618 \times 10^{4}}$	2.2968×10^{4}	1.6894×10^{4}	2.8792×10^{4}	1.5297×10^{4}	1.7901×10^{4}	3.3201×10^{4}	3.2692×10^{4}	1.5055×10^{4}
	Std	9.3870×10^{2}	4.5977×10^{3}	3.9365×10^{3}	6.3876×10^{3}	1.0574×10^{3}	7.4009×10^{2}	$\mathbf{7.2101 \times 10^{2}}$	1.5308×10^{3}	1.4631×10^{3}
	t/s	4.7258	1.7292	3.2137	5.6442	1.5165	3.2160	2.1005	1.3306	4.0181
F11	Best	$\mathbf{3.0076 \times 10^{3}}$	6.1280×10^{4}	4.2121×10^{3}	2.7487×10^{4}	3.6908×10^{3}	9.0535×10^{3}	1.7748×10^{5}	1.1418×10^{5}	1.3288×10^{4}
	Mean	1.2794×10^{4}	8.8071×10^{4}	6.3398×10^{3}	3.8824×10^{4}	$\mathbf{3.5790 \times 10^{3}}$	2.0653×10^{4}	2.3805×10^{5}	2.4576×10^{5}	3.0247×10^{4}
	Std	5.6085×10^{3}	1.8418×10^{4}	2.3601×10^{3}	7.8797×10^{3}	$\mathbf{6.6160 \times 10^{2}}$	1.1513×10^{4}	9.0793×10^{4}	7.9726×10^{4}	1.3023×10^{4}
	t/s	4.0130	1.5121	2.6438	4.8892	1.3667	2.8956	2.0016	1.3474	4.0135
F12	Best	3.4317×10^{7}	1.7612×10^{10}	2.7006×10^{7}	5.2690×10^{8}	4.4858×10^{7}	$\mathbf{1.7856 \times 10^{7}}$	1.6954×10^{11}	1.6815×10^{10}	2.0479×10^{7}
	Mean	6.9921×10^{7}	4.8776×10^{10}	8.0140×10^{7}	8.9722×10^{8}	1.8564×10^{8}	$\mathbf{4.0634 \times 10^{7}}$	1.9633×10^{11}	2.6425×10^{10}	4.9468×10^{7}
	Std	2.4252×10^{7}	1.4116×10^{10}	5.4916×10^{7}	2.1604×10^{8}	9.3404×10^{7}	$\mathbf{1.5745 \times 10^{7}}$	1.6429×10^{10}	6.7589×10^{9}	1.6486×10^{7}
	t/s	7.7358	2.6328	4.8208	9.1564	2.5862	5.5239	3.2911	2.3038	7.3141
F13	Best	1.4747×10^{4}	2.4097×10^{9}	1.0798×10^{4}	1.6671×10^{5}	3.2893×10^{4}	$\mathbf{5.4984 \times 10^{3}}$	3.4760×10^{10}	1.0984×10^{9}	3.6621×10^{4}
	Mean	2.3414×10^{5}	7.4919×10^{9}	3.2453×10^{5}	2.5089×10^{7}	5.1549×10^{4}	$\mathbf{1.2039 \times 10^{4}}$	4.6132×10^{10}	2.5579×10^{9}	3.2325×10^{5}
	Std	3.9318×10^{5}	3.2699×10^{9}	1.1746×10^{6}	6.2331×10^{7}	1.2435×10^{4}	$\mathbf{4.5773 \times 10^{3}}$	4.5208×10^{9}	8.8357×10^{8}	6.5978×10^{5}
	t/s	4.1088	1.5549	2.7092	5.0133	1.3599	2.9496	2.0093	1.1770	3.5294
F14	Best	5.8481×10^{5}	3.0752×10^{6}	2.5355×10^{5}	2.1643×10^{6}	$\mathbf{1.9806 \times 10^{3}}$	2.4004×10^{5}	5.3114×10^{7}	3.0004×10^{6}	2.7850×10^{6}
	Mean	2.6788×10^{6}	1.3635×10^{7}	4.3485×10^{5}	5.8203×10^{6}	$\mathbf{2.1809 \times 10^{3}}$	5.5254×10^{5}	1.4546×10^{8}	1.7232×10^{7}	5.4379×10^{6}
	Std	1.5504×10^{6}	7.2555×10^{6}	1.9560×10^{5}	2.6091×10^{6}	$\mathbf{1.6221 \times 10^{2}}$	2.0562×10^{5}	5.7215×10^{7}	7.5498×10^{6}	1.9429×10^{6}
	t/s	4.7755	1.7930	3.2624	5.9066	1.5747	3.3481	2.1989	1.3822	4.1669
F15	Best	7.0137×10^{3}	1.3942×10^{8}	4.1503×10^{3}	4.8203×10^{4}	1.2051×10^{4}	$\mathbf{2.2918 \times 10^{3}}$	1.7596×10^{10}	2.3759×10^{8}	1.6052×10^{4}
	Mean	1.3879×10^{4}	2.6814×10^{9}	7.7371×10^{3}	1.2856×10^{5}	2.0503×10^{4}	$\mathbf{5.0075 \times 10^{3}}$	2.4415×10^{10}	4.7543×10^{8}	1.6357×10^{5}
	Std	6.7577×10^{3}	2.6280×10^{9}	$\mathbf{3.4509 \times 10^{3}}$	1.5519×10^{5}	5.3763×10^{3}	4.6169×10^{3}	2.8800×10^{9}	2.1309×10^{8}	2.2057×10^{5}
	t/s	4.1153	1.5539	2.7209	5.0310	1.3563	2.9270	2.0072	1.1546	3.5028
F16	Best	$\mathbf{4.1718 \times 10^{3}}$	6.9747×10^{3}	4.5891×10^{3}	5.0556×10^{3}	4.3310×10^{3}	4.5722×10^{3}	2.2012×10^{4}	9.8551×10^{3}	4.9488×10^{3}
	Mean	$\mathbf{5.4367 \times 10^{3}}$	8.5797×10^{3}	5.9379×10^{3}	6.7837×10^{3}	5.8131×10^{3}	5.8502×10^{3}	2.4580×10^{4}	1.2240×10^{4}	6.3571×10^{3}
	Std	$\mathbf{5.3046 \times 10^{2}}$	1.1075×10^{3}	7.6812×10^{2}	8.7144×10^{2}	5.7407×10^{2}	5.7968×10^{2}	1.4083×10^{3}	7.3481×10^{2}	7.2669×10^{2}
	t/s	4.4036	1.6304	2.9272	5.3581	1.4481	3.1120	2.0991	1.2403	3.7715
F17	Best	3.8141×10^{3}	5.5048×10^{3}	4.8686×10^{3}	4.6060×10^{3}	$\mathbf{3.8107 \times 10^{3}}$	4.2266×10^{3}	2.6362×10^{6}	7.9828×10^{3}	4.7752×10^{3}
	Mean	$\mathbf{4.9269 \times 10^{3}}$	1.1224×10^{4}	6.2389×10^{3}	5.5932×10^{3}	5.2872×10^{3}	5.3458×10^{3}	6.4367×10^{6}	9.4597×10^{3}	5.7344×10^{3}
	Std	6.1868×10^{2}	7.5474×10^{3}	8.1725×10^{2}	5.1641×10^{2}	$\mathbf{2.5891 \times 10^{2}}$	4.3748×10^{2}	3.3589×10^{6}	1.1237×10^{3}	4.7807×10^{2}
	t/s	5.4787	1.9677	3.8705	6.8168	1.7897	3.8106	2.4384	1.6001	4.8553
F18	Best	9.4962×10^{5}	3.1495×10^{6}	3.2852×10^{5}	2.9300×10^{6}	$\mathbf{3.9328 \times 10^{4}}$	7.1780×10^{5}	1.7819×10^{8}	5.6353×10^{6}	1.0328×10^{6}
	Mean	3.5746×10^{6}	1.2378×10^{7}	7.2572×10^{5}	7.4111×10^{6}	$\mathbf{7.8123 \times 10^{4}}$	2.0247×10^{6}	3.2710×10^{8}	2.8956×10^{7}	4.4086×10^{6}
	Std	1.6770×10^{6}	8.7547×10^{6}	3.3913×10^{5}	4.6649×10^{6}	$\mathbf{2.8959 \times 10^{4}}$	1.3166×10^{6}	1.0756×10^{8}	1.1669×10^{7}	2.3185×10^{6}
	t/s	4.4907	1.6230	2.9608	5.4157	1.4489	3.0970	2.1069	1.2561	3.7853
F19	Best	5.8803×10^{3}	2.3574×10^{8}	2.7418×10^{3}	2.7621×10^{5}	8.4493×10^{3}	$\mathbf{2.2282 \times 10^{3}}$	1.8892×10^{10}	1.7025×10^{8}	2.0123×10^{4}
	Mean	2.5352×10^{4}	1.4158×10^{9}	9.9226×10^{3}	1.5670×10^{6}	4.7417×10^{4}	$\mathbf{5.3697 \times 10^{3}}$	2.3573×10^{10}	5.1937×10^{8}	1.0729×10^{5}
	Std	3.6883×10^{4}	1.4350×10^{9}	9.2389×10^{3}	1.3013×10^{6}	2.7199×10^{4}	$\mathbf{2.9430 \times 10^{3}}$	2.6930×10^{9}	2.7194×10^{8}	8.5920×10^{4}
	t/s	12.3916	4.2793	9.4724	16.1395	4.0812	8.3335	4.7699	3.9768	11.8527
F20	Best	$\mathbf{3.5102 \times 10^{3}}$	5.0013×10^{3}	4.4622×10^{3}	4.4568×10^{3}	3.9825×10^{3}	4.2830×10^{3}	7.7923×10^{3}	7.5181×10^{3}	4.4813×10^{3}
	Mean	$\mathbf{4.7368 \times 10^{3}}$	6.0160×10^{3}	5.2254×10^{3}	6.4504×10^{3}	4.7715×10^{3}	4.8884×10^{3}	8.4867×10^{3}	8.4636×10^{3}	5.5276×10^{3}
	Std	4.7171×10^{2}	9.3157×10^{2}	6.4863×10^{2}	1.3789×10^{3}	$\mathbf{2.7196 \times 10^{2}}$	4.4065×10^{2}	3.0754×10^{2}	3.4312×10^{2}	6.3391×10^{2}
	t/s	5.7108	2.0702	3.9915	7.0913	1.8721	4.0231	2.5549	1.7143	5.1034
F21	Best	$\mathbf{2.7491 \times 10^{3}}$	3.2831×10^{3}	3.0626×10^{3}	2.9508×10^{3}	2.8407×10^{3}	2.8380×10^{3}	4.7044×10^{3}	3.5064×10^{3}	2.9787×10^{3}
	Mean	2.8993×10^{3}	3.4578×10^{3}	3.3241×10^{3}	3.0630×10^{3}	$\mathbf{2.7113 \times 10^{3}}$	3.0329×10^{3}	5.2231×10^{3}	3.7000×10^{3}	3.1991×10^{3}
	Std	$\mathbf{6.7525 \times 10^{1}}$	1.1615×10^{2}	1.6406×10^{2}	1.2523×10^{2}	7.6296×10^{1}	1.1468×10^{2}	2.2509×10^{2}	6.9053×10^{1}	1.1995×10^{2}
	t/s	11.2853	3.9329	8.6258	14.5870	3.7253	7.6213	4.3612	3.5679	10.6390
F22	Best	1.4706×10^{4}	2.1747×10^{4}	1.7122×10^{4}	2.1173×10^{4}	$\mathbf{2.4077 \times 10^{3}}$	1.8214×10^{4}	3.4997×10^{4}	3.2321×10^{4}	1.5585×10^{4}
	Mean	$\mathbf{1.6709 \times 10^{4}}$	2.7760×10^{4}	2.1685×10^{4}	3.0173×10^{4}	1.7497×10^{4}	2.0738×10^{4}	3.6326×10^{4}	3.5870×10^{4}	1.7864×10^{4}
	Std	1.0502×10^{3}	5.1501×10^{3}	2.5970×10^{3}	6.2967×10^{3}	4.2990×10^{3}	1.5652×10^{3}	1.1526×10^{3}	$\mathbf{1.0500 \times 10^{3}}$	1.3444×10^{3}
	t/s	11.9621	4.1626	9.2919	15.4954	3.9517	8.0739	4.5633	3.7874	11.2156
F23	Best	$\mathbf{3.1289 \times 10^{3}}$	3.9940×10^{3}	3.8371×10^{3}	3.4007×10^{3}	3.1451×10^{3}	3.3018×10^{3}	7.3845×10^{3}	4.1557×10^{3}	3.2264×10^{3}
	Mean	$\mathbf{3.2435 \times 10^{3}}$	4.3192×10^{3}	4.0723×10^{3}	3.5958×10^{3}	3.2704×10^{3}	3.6058×10^{3}	8.3627×10^{3}	4.4292×10^{3}	3.3341×10^{3}
	Std	$\mathbf{4.1727 \times 10^{1}}$	1.9002×10^{2}	2.1326×10^{2}	1.3723×10^{2}	5.3526×10^{1}	1.4103×10^{2}	5.6612×10^{2}	1.3480×10^{2}	6.4360×10^{1}
	t/s	15.1926	5.5650	12.7935	22.2219	5.7875	11.8548	6.4597	5.6356	16.3938
F24	Best	$\mathbf{3.7121 \times 10^{3}}$	4.9288×10^{3}	4.5541×10^{3}	3.9216×10^{3}	3.7656×10^{3}	3.8146×10^{3}	1.2819×10^{4}	5.0568×10^{3}	3.9351×10^{3}
	Mean	4.0721×10^{3}	5.5242×10^{3}	4.8912×10^{3}	4.1406×10^{3}	$\mathbf{3.7686 \times 10^{3}}$	4.2280×10^{3}	1.3668×10^{4}	5.3300×10^{3}	4.1143×10^{3}
	Std	$\mathbf{9.5141 \times 10^{1}}$	2.7871×10^{2}	2.5385×10^{2}	1.0905×10^{2}	1.0704×10^{2}	2.1226×10^{2}	4.7064×10^{2}	1.5014×10^{2}	1.0862×10^{2}
	t/s	15.3368	5.2928	12.1025	20.1292	5.0871	10.4204	5.7957	4.9850	14.7719
F25	Best	$\mathbf{3.3703 \times 10^{3}}$	7.1010×10^{3}	3.5576×10^{3}	4.1541×10^{3}	3.5454×10^{3}	3.5503×10^{3}	2.5785×10^{4}	1.0844×10^{4}	3.4388×10^{3}
	Mean	$\mathbf{3.4659 \times 10^{3}}$	1.0917×10^{4}	3.7129×10^{3}	4.3814×10^{3}	3.6742×10^{3}	3.6542×10^{3}	2.7099×10^{4}	1.3963×10^{4}	3.4874×10^{3}
	Std	$\mathbf{3.8636 \times 10^{1}}$	1.3786×10^{3}	1.0013×10^{2}	1.6779×10^{2}	7.0170×10^{1}	1.0724×10^{2}	8.9908×10^{2}	1.7536×10^{3}	4.4924×10^{1}
	t/s	16.2199	5.6225	12.7534	21.5116	5.3606	10.9005	6.0473	5.2539	15.3668
F26	Best	$\mathbf{4.2482 \times 10^{3}}$	2.3675×10^{4}	1.6922×10^{4}	8.5673×10^{3}	1.0347×10^{4}	4.5873×10^{3}	5.3226×10^{4}	2.1715×10^{4}	1.3437×10^{4}
	Mean	$\mathbf{1.2894 \times 10^{4}}$	2.6205×10^{4}	2.3501×10^{4}	1.5935×10^{4}	1.3042×10^{4}	1.5902×10^{4}	5.5629×10^{4}	2.5357×10^{4}	1.7650×10^{4}
	Std	8.7666×10^{2}	1.9928×10^{3}	2.8476×10^{3}	3.3081×10^{3}	$\mathbf{5.9210 \times 10^{2}}$	6.2693×10^{3}	2.8633×10^{3}	1.7195×10^{3}	3.4457×10^{3}
	t/s	18.4209	6.3755	14.6249	24.1155	6.1177	12.4201	6.9180	6.0451	17.7302
F27	Best	$\mathbf{3.4415 \times 10^{3}}$	4.6112×10^{3}	3.6331×10^{3}	3.7372×10^{3}	3.4504×10^{3}	3.4969×10^{3}	1.4301×10^{4}	4.5779×10^{3}	3.5013×10^{3}
	Mean	$\mathbf{3.5507 \times 10^{3}}$	5.2965×10^{3}	4.0863×10^{3}	3.8545×10^{3}	3.5561×10^{3}	3.6907×10^{3}	1.5975×10^{4}	5.3025×10^{3}	3.6725×10^{3}
	Std	$\mathbf{4.4152 \times 10^{1}}$	4.8442×10^{2}	3.7591×10^{2}	1.0072×10^{2}	5.4205×10^{1}	1.1816×10^{2}	1.3819×10^{3}	3.0042×10^{2}	9.6210×10^{1}
	t/s	21.6908	7.5655	17.0821	28.2063	7.2471	14.4803	7.7225	6.9533	20.6600
F28	Best	$\mathbf{3.4311 \times 10^{3}}$	1.2511×10^{4}	3.6413×10^{3}	4.1375×10^{3}	3.5947×10^{3}	3.6602×10^{3}	3.2926×10^{4}	1.2856×10^{4}	3.4807×10^{3}
	Mean	$\mathbf{3.5304 \times 10^{3}}$	1.4895×10^{4}	3.7791×10^{3}	4.5742×10^{3}	3.7016×10^{3}	3.8729×10^{3}	3.4814×10^{4}	1.8798×10^{4}	3.5489×10^{3}
	Std	$\mathbf{2.8488 \times 10^{1}}$	1.8064×10^{3}	1.0424×10^{2}	2.3463×10^{2}	7.5750×10^{1}	1.7185×10^{2}	1.0261×10^{3}	2.3601×10^{3}	4.4406×10^{1}
	t/s	20.3126	7.0375	16.0974	26.7131	6.7057	13.6471	7.3563	6.5969	19.4686

Table 8. *Cont.*

		IWKGJO	GJO	GTO	WO	MPA	HMSWHO	IGWO	LSHADE	ECWOA
F29	Best	**5.5609 × 10^3**	1.0282 × 10^4	6.7047 × 10^3	7.0291 × 10^3	5.7472 × 10^3	6.1751 × 10^3	2.0970 × 10^5	1.2670 × 10^4	6.0744 × 10^3
	Mean	6.7735 × 10^3	1.4408 × 10^4	8.1342 × 10^3	8.0338 × 10^3	**6.7680 × 10^3**	7.0021 × 10^3	6.0719 × 10^5	1.5696 × 10^4	7.0861 × 10^3
	Std	**4.1233 × 10^2**	2.6989 × 10^3	7.8854 × 10^2	5.5954 × 10^2	4.5455 × 10^2	6.0298 × 10^2	2.2917 × 10^5	1.9861 × 10^3	6.4075 × 10^2
	t/s	11.8714	4.1179	9.0858	15.3447	3.9481	8.1091	4.5869	3.8195	11.2354
F30	Best	2.1290 × 10^5	2.0930 × 10^9	5.9159 × 10^4	6.4714 × 10^6	9.8111 × 10^5	**1.9687 × 10^4**	3.6778 × 10^{10}	6.4606 × 10^8	2.1974 × 10^5
	Mean	3.3491 × 10^5	4.9291 × 10^9	3.9761 × 10^5	3.3182 × 10^7	2.4353 × 10^6	**7.0768 × 10^4**	4.2002 × 10^{10}	1.4630 × 10^9	4.9313 × 10^5
	Std	**6.8108 × 10^4**	2.6076 × 10^9	2.4141 × 10^5	1.6641 × 10^7	8.9803 × 10^5	9.8358 × 10^4	3.5515 × 10^9	4.7415 × 10^8	2.0895 × 10^5
	t/s	18.8142	6.4678	14.8348	24.6204	6.2497	12.6758	6.8239	6.0993	18.0452

Significant values are in bold.

To summarize, there is minimal difference in the results achieved in 50 and 100 dimensions. Both dimensions have their advantages in handling simple multi-peak and combined functions. However, when it comes to single-peak and mixed functions, the MPA algorithm only performs slightly better than the other dimensions. Hence, the function suggested in this research exhibits superior search capabilities and is better suited for highly oscillatory issues.

This paper presents convergence curve images in both 50 and 100 dimensions. Figure 3 demonstrates that, in terms of convergence accuracy, the IWKGJO algorithm performs slightly worse than the MPA, LSHADE, and HMSWHO algorithms in functions F1, F8, F12–F15, F24, F26, and F30. However, in the remaining functions, the IWKGJO algorithm still maintains an advantage. The convergence speed of the IWKGJO algorithm in F4, F6, and F21–F23 function images is clearly evident. While the IWKGJO algorithm does not outperform MPA, HMSWHO, and LSHADE in terms of convergence speed, it still holds a significant edge over the WO, GTO, and ECWOA algorithms, which are considered better. Regarding stability, each function exhibits several inflection points, with the most prominent ones observed in functions F8, F10, F12, F13, F15, F16, F19, F20–F22, and the F30 image. These functions demonstrate a remarkable capacity to escape local optima.

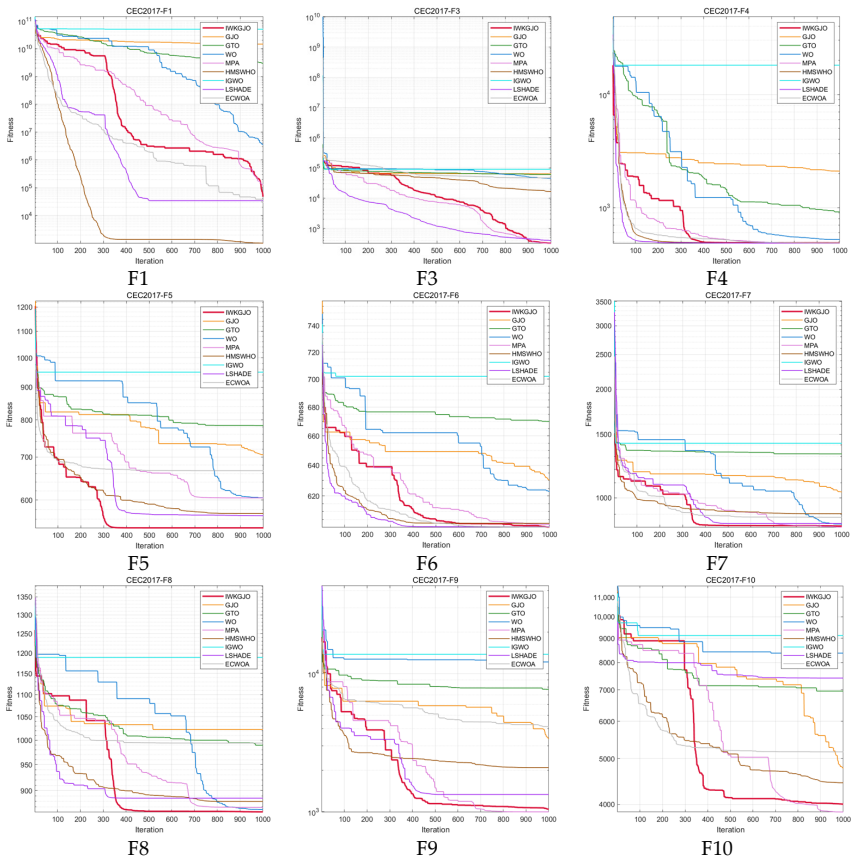


Figure 3. *Cont.*

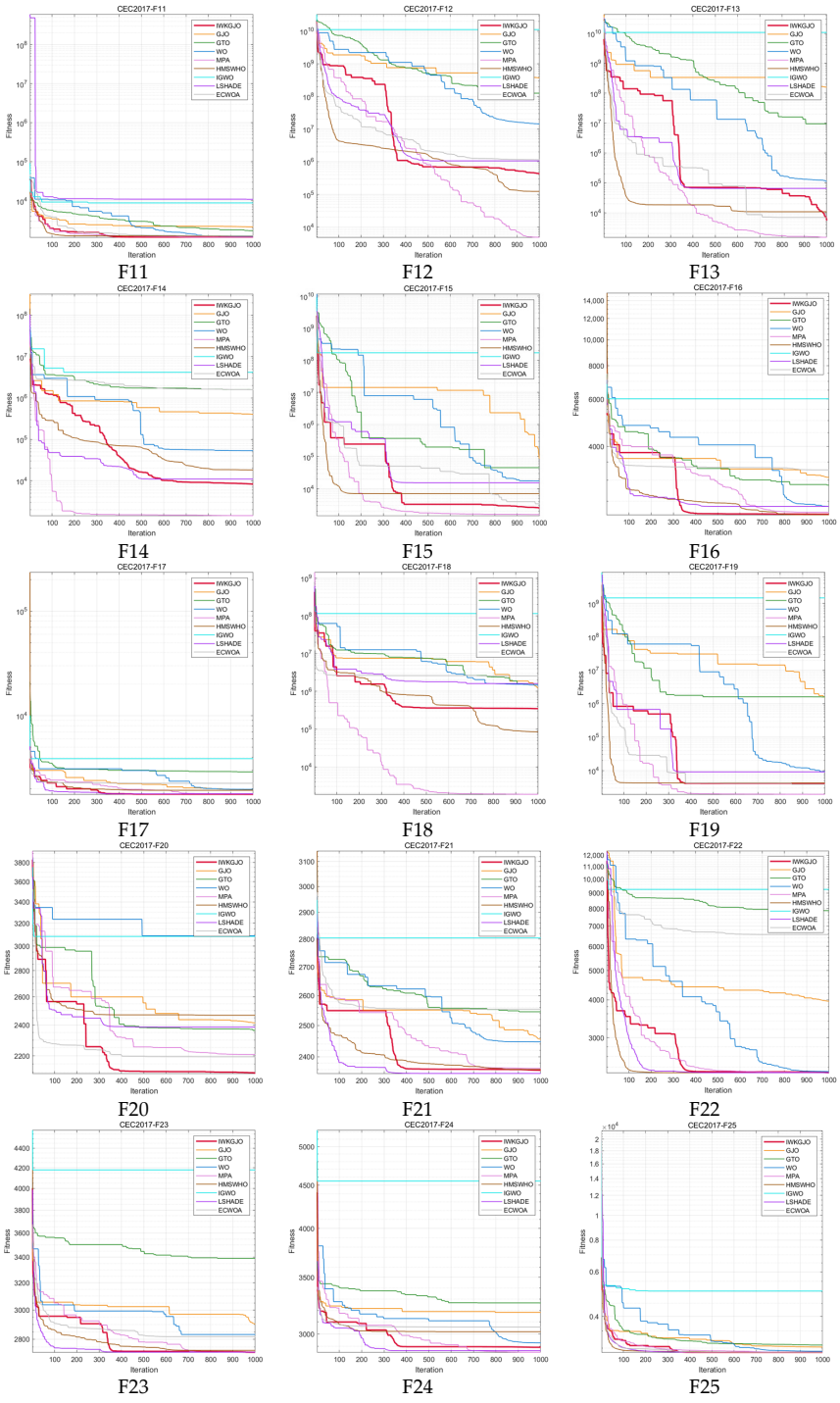


Figure 3. *Cont*.

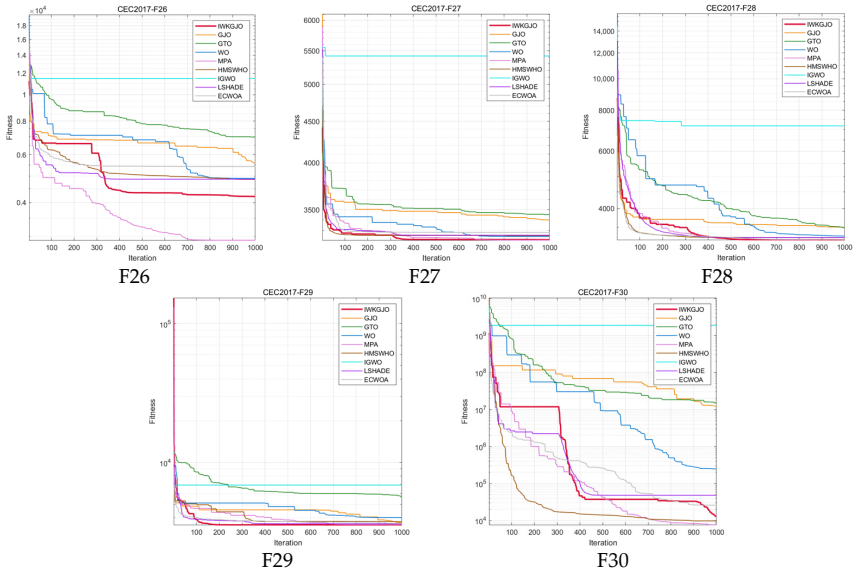


Figure 3. Convergence curve compared with other algorithms $(d = 50)$.

In Figure 4, the convergence accuracy of the MPA, LSHADE, and HMSWHO algorithms will be marginally inferior compared to the functions F1, F8, F12–F15, F21, F26, and F30 images. Nevertheless, the IWKGJO algorithm fails to reach the optimal solution in the functions F1, F3, F13, and F30, indicating the possibility of enhancing the accuracy of convergence. Regarding convergence speed, the IWKGJO algorithm is significantly slower than LSHADE. It only converges slower than MPA in functions F3, F14, F18, F19, and F25. The convergence speed of pictures in functions F1, F4, F13, F15, F19, and F30 is lower compared to that of HMSWHO. Regarding stability, the IWKGJO algorithm exhibits significant volatility in the early stages of image iteration. At approximately 300 iterations, numerous images experience a substantial leap, followed by a rapid convergence. This observation suggests that the proposed algorithm possesses a robust capability to escape local optima.

Figures 3 and 4, as well as Tables 7 and 8, provide a comprehensive analysis. The results indicate that while the data and images obtained from the F1, F12–F15, and F18–F19 functions are slightly inferior to those obtained from the MPA algorithm, the proposed algorithm still outperforms other algorithms in the remaining functions. In summary, the proposed algorithm demonstrates a significant advantage over many high-quality algorithms.

5.3. Wilcoxon Rank-Sum Test

The Wilcoxon rank-sum test is a non-parametric statistical test that takes into account both the direction and amount of the difference. Its purpose is to determine if there is a significant difference in the probability distribution of experimental data within a population. This test is suitable for analyzing the distribution of flying pillowcases, allowing the procedure to provide a more scientific representation of the algorithm's optimization performance compared to the mean value and standard deviation. This section presents a comparison and analysis of the various dimensions of CEC2017 test results. The optimization outcomes of GJO, GTO, WO, HMSWHO, IGWO, LSHADE, and ECWOA are evaluated using the Wilcoxon rank-sum test according to the IWKGJO algorithm.

The results of the Wilcoxon rank-sum test and p-value computation are presented in Tables 9 and 10. The judgment criterion for hypothesis testing is set at a significance level of $p = 5\%$. When the p-value is less than 5%, it is possible to observe a significant difference between the two sets of samples. The symbols "+", "=", and "−" are used to denote the relative performance of the IWKGJO algorithm in comparison to the comparison algorithm. The obtained statistical and analytical findings are shown below.

The data in Table 9 clearly demonstrate that IWKGJO outperforms both the original method and the IGWO algorithm. However, IWKGJO performs slightly worse than GTO in the F4 and F25 functions. Additionally, it is worse than WO in the F17, F24, and F26 functions. Furthermore, IWKGJO is also inferior to MPA in the F4, F8, F11, and F25 functions. The performance of the HMSWHO algorithm is marginally superior to that of the F4, F11, F16, F17, F24, and F29 functions. The algorithm performs less effectively than the LSHADE algorithm in the F4, F16, F20, F22, F27, and F30 functions. IWKGJO is significantly inferior to ECWOA in functions F1, F18, F19, F25, and F30. However, it still maintains an advantage over ECWOA in other functions.

The performance of the F4, F11, F16, F17, F24, and F29 functions is marginally inferior based on the results of the rank-sum test comparing various dimensions; the IWKGJO algorithm exhibits numerous advantages. However, it may somewhat underperform in certain functions compared to the reference algorithm. Consequently, the statistical outcomes typically outperform other algorithms and exhibit strong stability.

Table 10 demonstrates that in 100 dimensions, the IWKGJO method outperforms the original algorithm, as well as the WO and IGWO algorithms, with notable distinctions. IWKGJO performs slightly less effectively than GTO in functions F4 and F11. IWKGJO is less effective than MPA in functions F5, F9, F19, F20, and F27. Additionally, IWKGJO is inferior to HMSWHO in functions F9, F17, F20, and F29. Furthermore, IWKGJO performs worse than LASHADE in functions F16 and F27. Lastly, IWKGJO is also inferior to LASHADE in functions F9, F13, F18, F22, and F29. Consequently, IWKGJO is considered to be of lower quality compared to the ECWOA algorithm. In general, however, it exhibits superior performance compared to 50 dimensions, surpassing the other algorithms.

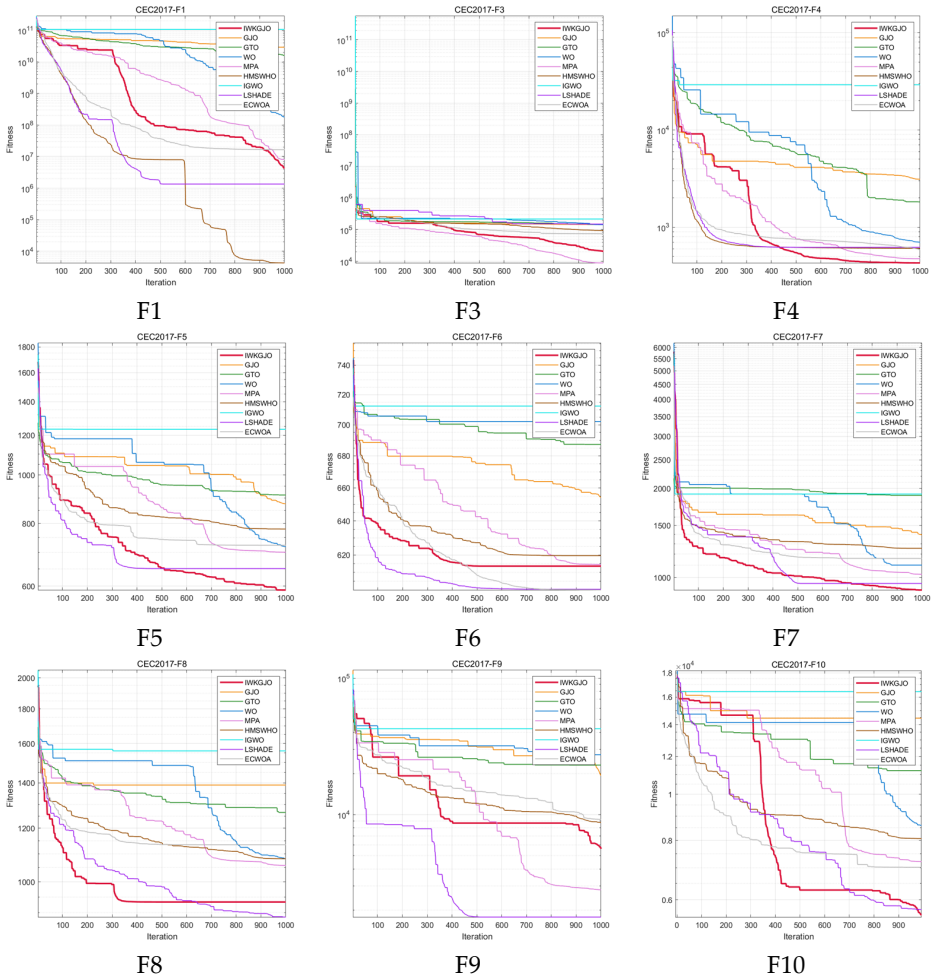


Figure 4. *Cont.*

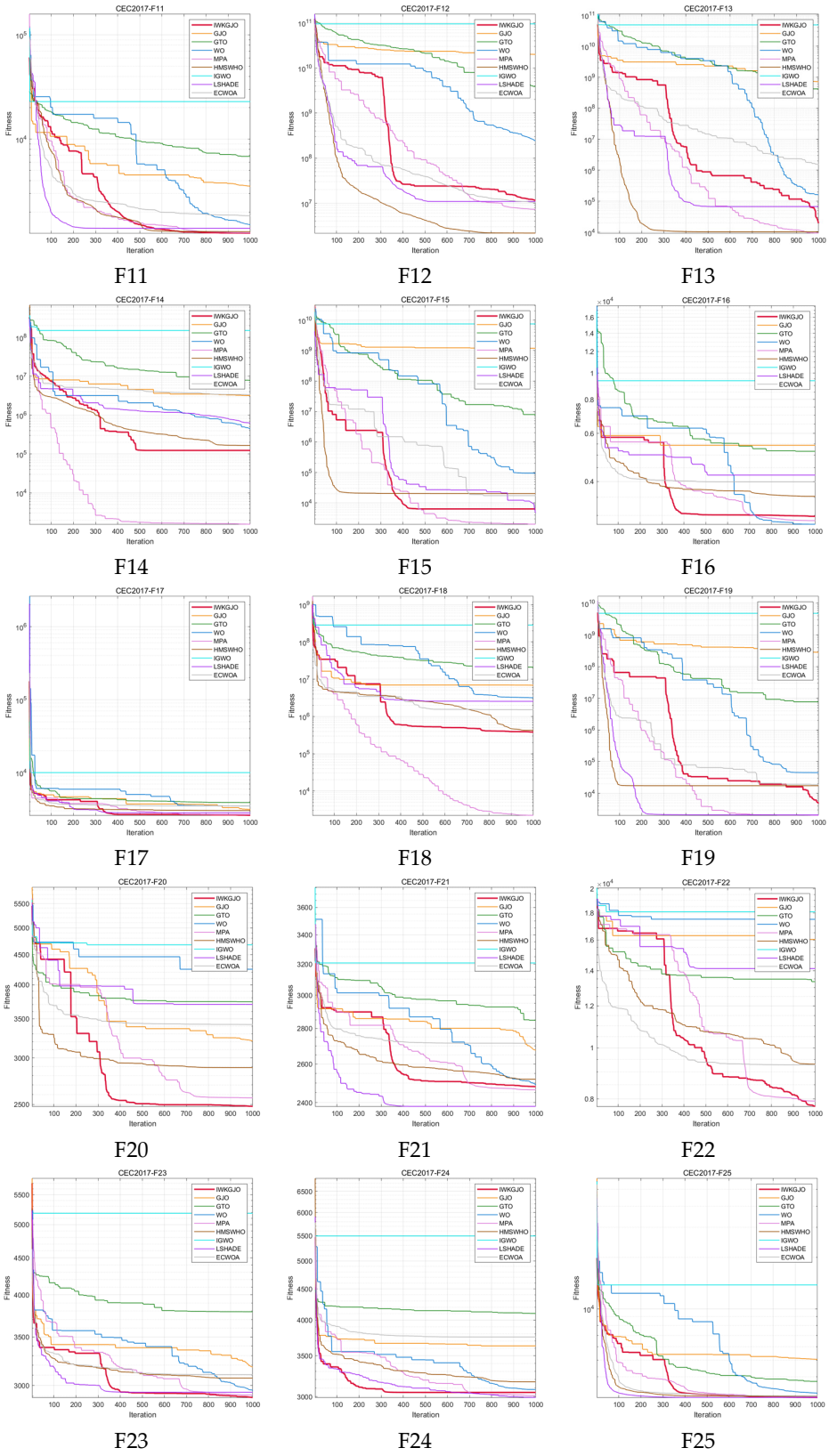


Figure 4. *Cont.*

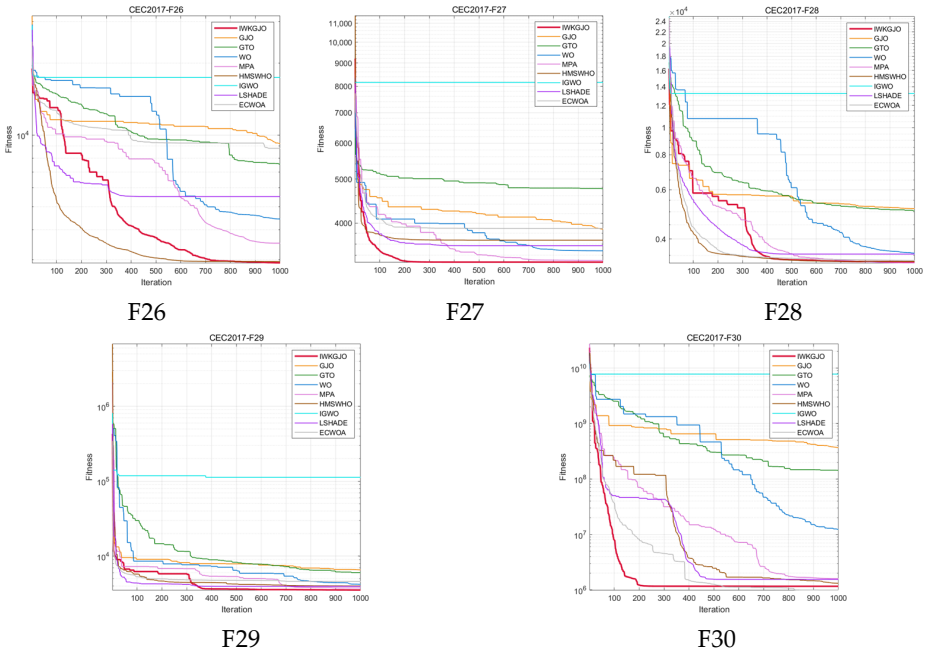


Figure 4. Convergence curve compared with other algorithms ($d = 100$).

Table 9. Results of Wilcoxon rank-sum test of CEC2017 test function ($d = 50$).

Function	D = 50							
	GJO	GTO	WO	MPA	HMSWHO	IGWO	LSHADE	ECWOA
F1	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	7.0162×10^{-3}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	$\mathbf{5.6145 \times 10^{-1}}$
F3	3.3918×10^{-6}	1.1457×10^{-4}	3.3918×10^{-6}	1.3295×10^{-5}	3.3918×10^{-6}	3.3918×10^{-6}	2.1337×10^{-2}	3.3918×10^{-6}
F4	3.3918×10^{-6}	$\mathbf{6.1970 \times 10^{-2}}$	3.3918×10^{-6}	$\mathbf{6.1867 \times 10^{-1}}$	$\mathbf{8.3571 \times 10^{-1}}$	3.3918×10^{-6}	$\mathbf{2.8084 \times 10^{-1}}$	1.0122×10^{-2}
F5	3.3918×10^{-6}	3.3918×10^{-6}	2.4626×10^{-3}	2.4549×10^{-2}	5.4521×10^{-3}	3.3918×10^{-6}	3.3918×10^{-6}	7.4772×10^{-6}
F6	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	8.1340×10^{-5}	4.0200×10^{-5}	3.3918×10^{-6}	4.1432×10^{-6}	3.3918×10^{-6}
F7	3.3918×10^{-6}	3.3918×10^{-6}	7.4772×10^{-6}	1.8067×10^{-2}	4.0200×10^{-5}	3.3918×10^{-6}	9.0734×10^{-6}	1.0992×10^{-5}
F8	3.3918×10^{-6}	2.3290×10^{-5}	2.2531×10^{-2}	$\mathbf{7.4002 \times 10^{-1}}$	1.7107×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	2.7983×10^{-5}
F9	3.3918×10^{-6}	1.6053×10^{-5}	9.0734×10^{-6}	9.0734×10^{-6}	4.2247×10^{-4}	3.3918×10^{-6}	3.3918×10^{-6}	2.6217×10^{-4}
F10	3.3918×10^{-6}	1.6053×10^{-5}	7.4772×10^{-6}	1.8441×10^{-2}	8.1340×10^{-5}	3.3918×10^{-6}	2.9976×10^{-2}	7.9403×10^{-3}
F11	3.3918×10^{-6}	1.6197×10^{-3}	3.3918×10^{-6}	$\mathbf{5.8974 \times 10^{-1}}$	$\mathbf{9.7091 \times 10^{-2}}$	3.3918×10^{-6}	1.5846×10^{-2}	1.9352×10^{-5}
F12	3.3918×10^{-6}	3.3568×10^{-5}	5.0527×10^{-6}	4.1432×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	5.4521×10^{-3}
F13	3.3918×10^{-6}	7.8021×10^{-4}	7.4772×10^{-6}	6.1516×10^{-6}	1.0992×10^{-5}	3.3918×10^{-6}	5.0527×10^{-6}	6.8368×10^{-5}
F14	4.0200×10^{-5}	1.0992×10^{-5}	1.0500×10^{-3}	3.3918×10^{-6}	7.9403×10^{-3}	3.3918×10^{-6}	3.3918×10^{-6}	1.0992×10^{-5}
F15	3.3918×10^{-6}	3.1017×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	1.2822×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	2.7925×10^{-2}
F16	1.6197×10^{-3}	3.6906×10^{-3}	4.7948×10^{-3}	1.2822×10^{-2}	$\mathbf{5.8974 \times 10^{-1}}$	3.3918×10^{-6}	$\mathbf{8.1495 \times 10^{-2}}$	1.6197×10^{-3}
F17	2.2289×10^{-4}	4.7948×10^{-3}	$\mathbf{9.7091 \times 10^{-2}}$	3.2301×10^{-3}	$\mathbf{6.7830 \times 10^{-1}}$	3.3918×10^{-6}	4.9369×10^{-4}	4.2099×10^{-3}
F18	1.1457×10^{-4}	7.4772×10^{-6}	1.2486×10^{-2}	3.3918×10^{-6}	9.6615×10^{-5}	3.3918×10^{-6}	3.3918×10^{-6}	$\mathbf{1.9851 \times 10^{-1}}$
F19	3.3918×10^{-6}	1.5846×10^{-2}	9.6615×10^{-5}	3.3918×10^{-6}	1.0574×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	$\mathbf{8.0346 \times 10^{-1}}$
F20	1.0122×10^{-2}	2.0191×10^{-2}	7.9403×10^{-3}	4.7948×10^{-3}	$\mathbf{6.1867 \times 10^{-1}}$	3.3918×10^{-6}	$\mathbf{7.4002 \times 10^{-1}}$	5.4521×10^{-3}
F21	3.3918×10^{-6}	5.0527×10^{-6}	3.6150×10^{-2}	1.3564×10^{-4}	1.7107×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	1.0992×10^{-5}
F22	7.4772×10^{-6}	4.8063×10^{-5}	8.1340×10^{-5}	1.1457×10^{-4}	2.2289×10^{-4}	3.3918×10^{-6}	$\mathbf{8.6823 \times 10^{-1}}$	2.8226×10^{-3}
F23	3.3918×10^{-6}	6.1516×10^{-6}	4.6487×10^{-2}	2.3290×10^{-5}	1.1499×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	5.0527×10^{-6}
F24	3.3918×10^{-6}	1.0992×10^{-5}	$\mathbf{5.8974 \times 10^{-1}}$	1.6053×10^{-5}	$\mathbf{8.3571 \times 10^{-1}}$	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}
F25	3.3918×10^{-6}	$\mathbf{5.6392 \times 10^{-2}}$	3.3918×10^{-6}	$\mathbf{8.3571 \times 10^{-1}}$	6.7090×10^{-4}	3.3918×10^{-6}	1.4397×10^{-2}	$\mathbf{7.4496 \times 10^{-2}}$
F26	3.3918×10^{-6}	2.6217×10^{-4}	$\mathbf{7.4496 \times 10^{-2}}$	8.9720×10^{-3}	2.2531×10^{-2}	3.3918×10^{-6}	5.4521×10^{-3}	3.3568×10^{-5}
F27	3.3918×10^{-6}	1.9352×10^{-5}	4.9369×10^{-4}	4.2111×10^{-2}	1.8067×10^{-2}	3.3918×10^{-6}	$\mathbf{9.7091 \times 10^{-2}}$	4.2247×10^{-4}
F28	3.3918×10^{-6}	1.2152×10^{-3}	4.1432×10^{-6}	2.6275×10^{-2}	3.6093×10^{-4}	3.3918×10^{-6}	1.5846×10^{-2}	1.1401×10^{-2}
F29	3.3918×10^{-6}	7.4772×10^{-6}	2.8084×10^{-2}	3.1017×10^{-2}	$\mathbf{5.0691 \times 10^{-1}}$	3.3918×10^{-6}	8.1340×10^{-5}	7.0162×10^{-3}
F30	3.3918×10^{-6}	1.0574×10^{-2}	3.3918×10^{-6}	1.6053×10^{-5}	3.4009×10^{-2}	3.3918×10^{-6}	$\mathbf{5.8974 \times 10^{-1}}$	**1.0000**
+/=/−	29/0/0	27/0/2	26/0/3	25/0/4	22/0/7	29/0/0	23/0/6	24/0/5

Significant values are in bold.

Table 10. Results of Wilcoxon rank-sum test of CEC2017 test function ($d = 100$).

Function	D = 100							
	GJO	GTO	WO	MPA	HMSWHO	IGWO	LSHADE	ECWOA
F1	3.3918×10^{-6}	$\mathbf{7.7155 \times 10^{-1}}$	3.3918×10^{-6}	1.3295×10^{-5}	1.3295×10^{-5}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}
F3	1.3564×10^{-4}	4.1432×10^{-6}	2.7983×10^{-5}	3.3918×10^{-6}	1.1457×10^{-4}	3.3918×10^{-6}	1.4041×10^{-3}	3.3918×10^{-6}
F4	3.3918×10^{-6}	5.0527×10^{-6}	3.3918×10^{-6}	2.2531×10^{-2}	4.0200×10^{-5}	3.3918×10^{-6}	1.8655×10^{-3}	1.5846×10^{-2}
F5	3.3918×10^{-6}	3.3918×10^{-6}	1.3295×10^{-5}	$\mathbf{5.8974 \times 10^{-1}}$	1.3295×10^{-5}	3.3918×10^{-6}	6.1516×10^{-6}	2.3290×10^{-5}
F6	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	1.0574×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}
F7	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	2.6217×10^{-4}	2.6217×10^{-4}	3.3918×10^{-6}	1.6053×10^{-5}	9.0734×10^{-6}
F8	3.3918×10^{-6}	3.3918×10^{-6}	2.3290×10^{-5}	$\mathbf{9.6691 \times 10^{-1}}$	3.3568×10^{-5}	3.3918×10^{-6}	7.4772×10^{-6}	4.1432×10^{-6}
F9	7.8021×10^{-4}	6.7090×10^{-4}	4.8063×10^{-5}	1.1457×10^{-4}	$\mathbf{7.4496 \times 10^{-2}}$	4.1432×10^{-6}	3.3918×10^{-6}	$\mathbf{4.5530 \times 10^{-1}}$
F10	3.3918×10^{-6}	2.0191×10^{-2}	3.3918×10^{-6}	3.4397×10^{-2}	4.1432×10^{-6}	3.3918×10^{-6}	4.1432×10^{-6}	1.2486×10^{-2}
F11	3.3918×10^{-6}	$\mathbf{3.1951 \times 10^{-1}}$	3.3918×10^{-6}	2.8226×10^{-3}	1.3295×10^{-5}	3.3918×10^{-6}	2.0191×10^{-2}	4.1432×10^{-6}
F12	3.3918×10^{-6}	1.1401×10^{-2}	3.3918×10^{-6}	4.7996×10^{-2}	1.0992×10^{-5}	3.3918×10^{-6}	3.3918×10^{-6}	2.7983×10^{-5}
F13	3.3918×10^{-6}	5.7371×10^{-5}	1.0500×10^{-3}	4.0200×10^{-5}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	$\mathbf{4.0679 \times 10^{-1}}$
F14	4.1432×10^{-6}	7.4772×10^{-6}	7.4772×10^{-6}	3.3918×10^{-6}	2.7983×10^{-5}	3.3918×10^{-6}	3.3918×10^{-6}	7.4772×10^{-6}
F15	3.3918×10^{-6}	3.3918×10^{-6}	1.8919×10^{-4}	6.7090×10^{-4}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	2.0191×10^{-2}
F16	4.1432×10^{-6}	$\mathbf{7.7155 \times 10^{-1}}$	3.6906×10^{-3}	1.3538×10^{-2}	4.0679×10^{-2}	3.3918×10^{-6}	$\mathbf{5.0691 \times 10^{-1}}$	4.1970×10^{-2}
F17	1.3295×10^{-5}	1.4041×10^{-3}	4.7091×10^{-2}	1.1457×10^{-4}	$\mathbf{7.4002 \times 10^{-1}}$	3.3918×10^{-6}	4.7996×10^{-2}	2.5103×10^{-2}
F18	3.3568×10^{-5}	3.3918×10^{-6}	2.4626×10^{-3}	3.3918×10^{-6}	2.7925×10^{-2}	3.3918×10^{-6}	3.3918×10^{-6}	$\mathbf{1.5846 \times 10^{-1}}$
F19	3.3918×10^{-6}	2.7983×10^{-5}	3.3918×10^{-6}	$\mathbf{5.6145 \times 10^{-1}}$	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	1.4397×10^{-2}
F20	2.2289×10^{-4}	3.1495×10^{-2}	1.4041×10^{-3}	$\mathbf{2.1337 \times 10^{-1}}$	$\mathbf{5.0691 \times 10^{-1}}$	3.3918×10^{-6}	7.8021×10^{-4}	2.4626×10^{-3}
F21	3.3918×10^{-6}	3.3918×10^{-6}	6.7090×10^{-4}	5.0527×10^{-6}	4.7948×10^{-3}	3.3918×10^{-6}	3.3918×10^{-6}	7.4772×10^{-6}
F22	3.3918×10^{-6}	4.8063×10^{-5}	3.3918×10^{-6}	1.4397×10^{-2}	1.6053×10^{-5}	3.3918×10^{-6}	4.1432×10^{-6}	$\mathbf{2.4549 \times 10^{-1}}$
F23	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	2.4549×10^{-2}	6.1516×10^{-6}	3.3918×10^{-6}	9.0734×10^{-6}	4.7948×10^{-3}
F24	3.3918×10^{-6}	3.3918×10^{-6}	9.6615×10^{-5}	8.1340×10^{-5}	5.7598×10^{-4}	3.3918×10^{-6}	1.1457×10^{-4}	2.6217×10^{-4}
F25	3.3918×10^{-6}	4.1432×10^{-6}	3.3918×10^{-6}	5.0527×10^{-6}	5.0527×10^{-6}	3.3918×10^{-6}	2.5103×10^{-2}	4.7948×10^{-3}
F26	3.3918×10^{-6}	3.3918×10^{-6}	1.0500×10^{-3}	5.7371×10^{-5}	2.0191×10^{-2}	3.3918×10^{-6}	5.7371×10^{-5}	1.3564×10^{-4}
F27	3.3918×10^{-6}	4.1432×10^{-6}	3.3918×10^{-6}	$\mathbf{7.7155 \times 10^{-1}}$	9.0585×10^{-4}	3.3918×10^{-6}	$\mathbf{9.0097 \times 10^{-1}}$	3.6093×10^{-4}
F28	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	3.3568×10^{-5}	3.3918×10^{-6}	3.3918×10^{-6}	3.2301×10^{-3}	2.7925×10^{-2}
F29	3.3918×10^{-6}	6.7090×10^{-4}	1.6033×10^{-4}	4.6487×10^{-2}	**1.0000**	3.3918×10^{-6}	2.8226×10^{-3}	$\mathbf{5.3383 \times 10^{-1}}$
F30	3.3918×10^{-6}	6.7090×10^{-4}	3.3918×10^{-6}	7.4772×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	3.3918×10^{-6}	2.4626×10^{-3}
+/=/−	29/0/0	26/0/3	29/0/0	24/0/5	25/0/4	29/0/0	27/0/2	24/0/5

Significant values are in bold.

Overall, as the number of dimensions increases, IWKGJO's performance in comparison to GTO, WO, MPA, HMSWHO, and LSHADE algorithms improves progressively. Nevertheless, the performance of MPA, HMSWHO, and ECWOA algorithms will improve as the dimensionality increases. Out of all the functions, IWKGJO surpasses the comparison algorithm in terms of performance, while only a small number of functions are marginally inferior to these algorithms. Regardless of the specific function used, both other comparison algorithms and the original algorithm consistently outperform them in terms of performance.

6. Practical Engineering Application

The objective function $f\left(\vec{x}\right)$ is the fitness function in the engineering optimization problem [29], where $\vec{x}$ is the search space and distinct variables are the dimensions. The objective function, constraints, and variable interval are all subject to equal or unequal restrictions. There are other methods to convert algorithms into restricted optimization problems, and the literature has extensively examined and assessed the CHT method [30]. These comprise penalty, separation of constraints and objectives, special operators, hybrid, and repair-algorithms-based CHTs. The punishment function is the most basic and often utilized. The penalty function serves to convert a constrained problem into one or multiple unconstrained problems. During the solving process, any point that violates a constraint is assigned a high value for the objective function, which is then replaced with a new value in the subsequent iteration. This forces the extreme point of the unconstrained problem to approach the feasible domain infinitely closely, until it ultimately converges to the extreme point of the original constrained problem. Due to the widespread applicability and usefulness of the penalty function, the suggested algorithm utilizes the penalty function to handle restrictions.

This work selects three classical engineering instances, namely, the pressure vessel design problem, the three-bar truss design problem, and the gear train design problem, in order to assess the engineering application abilities of the suggested algorithm. This paper

selects and compares five algorithms that are suitable for engineering applications: the butterfly optimization algorithm (BOA) [31], the snake optimization algorithm (SO) [3], the osprey optimization algorithm (OOA) [32], the sine and cosine algorithm (SCA) [33], and the Harris eagle optimization algorithm (HHO) [34]. The maximum iteration and overall size (search agent) are set to 1000 and 100, respectively.

6.1. Pressure Vessel Design Problem

The current research analyzes the actual architectural problem of the broadened algorithm's performance using an actual pressure vessel design optimization contention. We chose the variables below to minimize manufacturing expenses and assure pressure vessel function. The following four elements influence shell thickness (T_S), head thickness (T_h), inner radius (R), and cylindrical section length (L) minus a head:

$$\vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_S\ T_h\ R\ L]$$

Figure 5 is the schematic diagram of the pressure vessel.

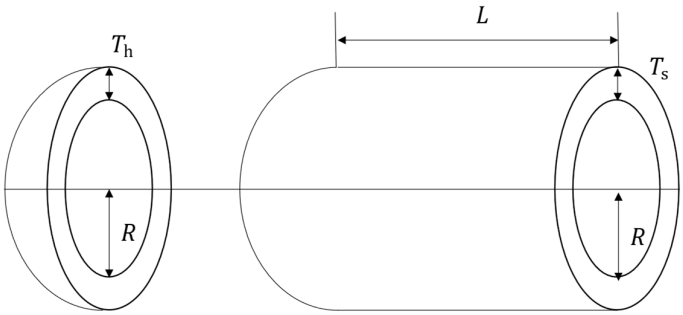


Figure 5. Pressure vessel diagram.

The aim of this exercise is to minimize the corresponding significance of the aim of the function:

$$f\left(\vec{x}\right) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

The preceding restrictions ought to be carried out:

$$g_1\left(\vec{x}\right) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2\left(\vec{x}\right) = -x_3 + 0.00954x_3 \leq 0$$

$$g_3\left(\vec{x}\right) = -\pi x_3^2 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0$$

$$g_4\left(\vec{x}\right) = x_4 - 240 \leq 0$$

The number associated with the span is where the topic is:

$$0 \leq x_1, x_2 \leq 99, \quad 10 \leq x_3, x_4 \leq 200$$

The most appropriate divergence curve for the pressure-related problem design is highlighted in Figure 6. IWKGJO converges with greater speed than the remaining algorithms, as can be discovered. Table 11's optimization outcomes regarding the pressure vessel design problems demonstrate that IWKGJO generated the lowest number of optimization results all around, indicating that the algorithm's accuracy stood out more than that of the other strategies. Because of this, it showcases the excellence of IWKGJO and its engineering optimization competence, which can be employed to address real-world problem areas in engineering.

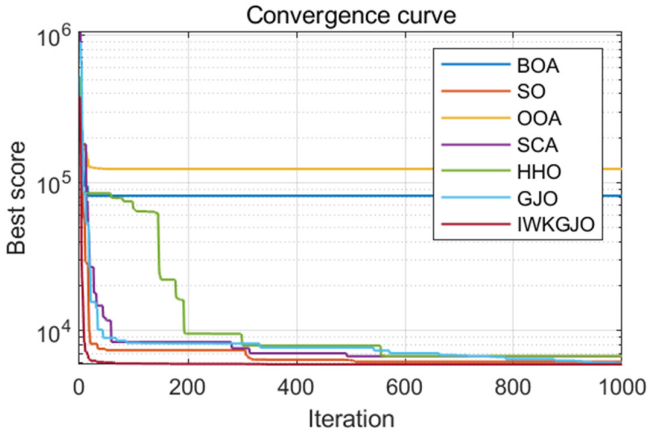


Figure 6. Optimization convergence diagram of pressure vessel design problem.

Table 11. Optimization results of pressure vessel design problems.

Algorithm	$T_S(x_1)$	$T_h(x_2)$	$R(x_3)$	$L(x_4)$	Result
BOA	2.7332	11.5585	56.7424	57.0640	8.1439×10^4
SO	0.8598	0.4324	43.9128	155.3836	6.1419×10^3
OOA	5.8318	12.9359	53.4190	73.3678	1.2381×10^5
SCA	0.8250	0.5383	40.3831	200.0000	6.6846×10^3
HHO	1.0813	0.5132	53.7796	70.9268	6.7164×10^3
GJO	0.7788	0.4274	40.3268	200.0000	6.0143×10^3
IWKGJO	**0.7799**	**0.3855**	**40.4089**	**198.7621**	$\mathbf{5.8883 \times 10^3}$

Significant values are in bold.

6.2. Three-Bar Truss Design Problem

By tweaking two parameter variables along with matching the stress restrictions on both edges of the truss member, the most important objective of the three-bar truss design issue is to mitigate the whole weight of the entire structure. The parameters of the two trusses are x_1 and the parameters of the middle trusses are x_2, as shown in the Figure 7.

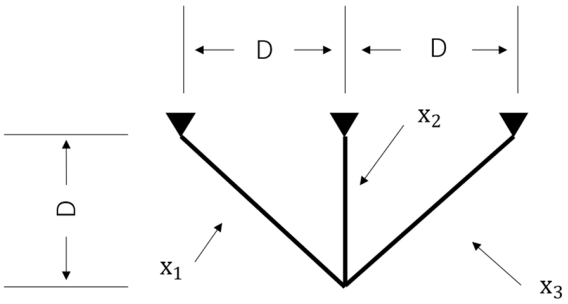


Figure 7. Schematic diagram of the three-bar truss.

Function:

$$minf(x) = \left(2\sqrt{2}x_1 + x_2\right) \times l$$

Constraint condition:

$$g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}p - \sigma \leq 0$$

$$g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}p - \sigma \leq 0$$

$$g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1} p - \sigma \leq 0$$
$$0 \leq x_i \leq 1, i = 1, 2$$

Argument:

$$l = 100 \text{ cm}, \ p = 2 \text{ KN}/\left(\text{cm}^2\right), \ \sigma = 2 \text{ KN}/\left(\text{cm}^2\right)$$

The final score of IWKGJO is 263.8959, as Table 12 demonstrates. As shown in Figure 8, IWKGJO converges faster than other algorithms. While contrasting the updated guidelines weight in general against various algorithms and the original procedure, it is the minimum, presuming that the stress boundaries on each of the sides of the truss members are maintained. The improvement's performance and adaptability to initiatives in the real world have therefore been verified.

Table 12. Optimization results of three-bar truss design problems.

Algorithm	x_1	x_2	Result
BOA	0.7901	0.4447	267.9272
SO	0.7882	0.4095	263.8960
OOA	0.7491	0.5338	265.2504
SCA	0.7961	0.3891	264.0967
HHO	0.7931	0.3958	263.9103
GJO	0.7900	0.4046	263.8994
IWKGJO	**0.7890**	**0.4074**	**263.8959**

Significant values are in bold.

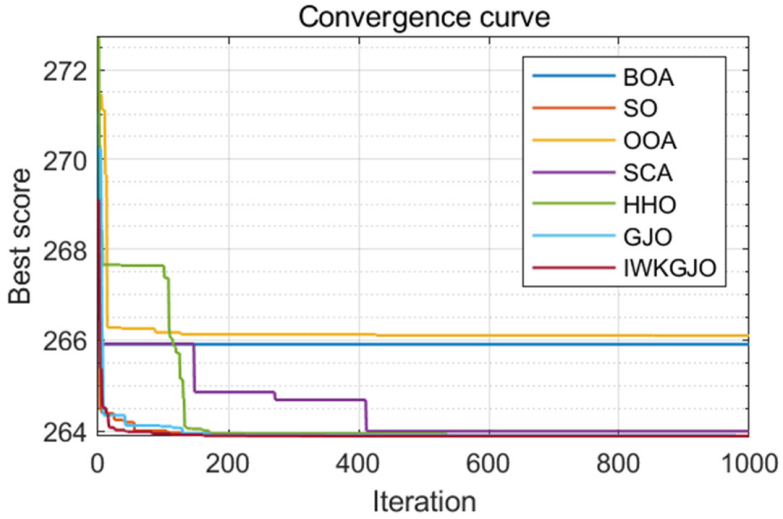


Figure 8. Optimization convergence diagram of three-bar truss design problems.

6.3. Gear Train Design Problem

In the discipline of mechanical engineering, gearing train engineering problem-solving is an unbounded combinatorial design task where the ultimate objective is to lessen the gear proportion—which is a combination of the angle of rotation of the shaft that goes into the machine to the angular speed of the resultant wheel—and thereby decrease the individual communication cost of a gear train. The variable is the number of gears $N_a(x_1)$, $N_b(x_2)$, $N_d(x_3)$, and $N_f(x_4)$ of the four gears, corresponding to the A, B, D, and F gears in Figure 9 below, respectively.

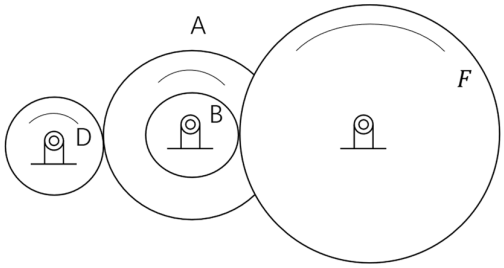


Figure 9. Gear design drawing.

Function:

$$minf(x) = \left(\frac{1}{6.931} - \frac{x_2 x_3}{x_1 x_4}\right)^2$$

Constraint condition:

$$12 \leq x_i \leq 60, i = 1,\ 2,\ 3,\ 4$$

Table 13 reveals how IWKGJO picks up an outcome of 2.7009×10^{-12}. And in Figure 10, IWKGJO reaches the optimal value first. When weighed against various alternatives and the initially proposed approach, there is already a minimal transmission cost. It also clarifies how IWKGJO can be used for architectural occasions and proves how well it works in gear train layout.

Table 13. Optimization results of gear train design problems.

Algorithm	$N_a(x_1)$	$N_b(x_2)$	$N_d(x_3)$	$N_f(x_4)$	Result
BOA	21.5239	12.0000	12.0686	41.9815	1.3375×10^{-4}
SO	53.1540	12.6016	30.4272	51.4634	2.3078×10^{-11}
OOA	31.4092	12.4288	12.0956	32.1359	7.7786×10^{-7}
SCA	42.8486	20.6894	13.3656	43.6433	9.9216×10^{-10}
HHO	46.7179	12.5606	12.0000	23.2558	2.3576×10^{-9}
GJO	34.0407	12.8914	20.3882	53.1689	2.3078×10^{-11}
IWKGJO	**48.5067**	**16.0298**	**18.7942**	**43.4418**	$\mathbf{2.7009 \times 10^{-12}}$

Significant values are in bold.

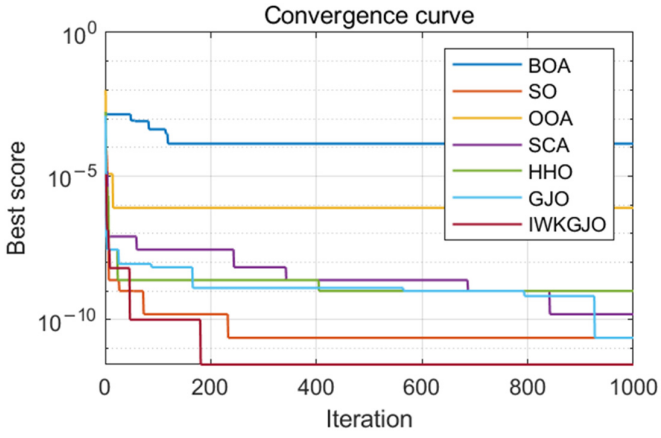


Figure 10. Optimization convergence diagram of gear train design problems.

7. Conclusions

To enhance the optimization performance of GJO, a refined golden jackal optimization algorithm is suggested, utilizing a mixed strategy approach. The location update of the enhanced development stage is employed to address the limitation of solely conducting a local search in the algorithm's later phase. Furthermore, the inclusion of the avoidance of natural adversaries serves to enhance both search efficiency and optimization accuracy.

Subsequently, the population's variety was augmented by cross mutation. The crossbar technique is implemented to enhance the global optimal solution, increase population variety, and improve the capacity to avoid local optima. To assess the practicality and resilience of the enhanced algorithm, we utilize 20 benchmark test functions to compare it with the original approach. To further assess the benefits of the enhanced method, simulation experiments are conducted using the CEC2017 test function set. Ultimately, the engineering optimization capability of IWKGJO is demonstrated by the application of pressure vessel design problems, three-bar truss design difficulties, and gear train design challenges. This confirms that IWKGJO is suitable for solving real-world problems. The results indicate that IWKGJO exhibits superior optimization performance and enhanced convergence speed. The utilization of the golden jackal optimization technique has been extensive in the field of engineering, with a future emphasis on addressing multi-objective, nonlinear, and other intricate engineering optimization issues.

Author Contributions: Conceptualization, Y.L.; software, Q.Y.; data curation, Q.Y.; writing—review and editing, Q.Y.; supervision, Z.W., Z.D. and Z.J.; project administration, Z.W.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (grant no. 52278171), and Natural Science Foundation of Hebei Province (grant no. E2020402079).

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank the anonymous reviewers for helping us improve this paper's quality. Thanks to the National Natural Science Foundation of China (grant no. 52278171), and Natural Science Foundation of Hebei Province (grant no. E2020402079) funded us to complete the experiment.

Conflicts of Interest: Author Zidong Jin was employed by the company Jizhong Energy Fengfeng Group Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The Jizhong Energy Fengfeng Group Co., Ltd. had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Zhong, C.; Li, G.; Meng, Z. Beluga Whale Optimization: A Novel Nature-Inspired Metaheuristic Algorithm. *Knowl. Based Syst.* **2022**, *251*, 109215. [CrossRef]
2. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Gazelle Optimization Algorithm: A Novel Nature-Inspired Metaheuristic Optimizer. *Neural Comput. Appl.* **2023**, *35*, 4099–4131. [CrossRef]
3. Hashim, F.A.; Hussien, A.G. Snake Optimizer: A Novel Meta-Heuristic Optimization Algorithm. *Knowl. Based Syst.* **2022**, *242*, 108320. [CrossRef]
4. Jia, H.; Rao, H.; Wen, C.; Mirjalili, S. Crayfish Optimization Algorithm. *Artif. Intell. Rev.* **2023**, *56*, 1919–1979. [CrossRef]
5. Houssein, E.H.; Oliva, D.; Samee, N.A.; Mahmoud, N.F.; Emam, M.M. Liver Cancer Algorithm: A Novel Bio-Inspired Optimizer. *Comput. Biol. Med.* **2023**, *165*, 107389. [CrossRef] [PubMed]
6. Lian, J.; Hui, G. Human Evolutionary Optimization Algorithm. *Expert Syst. Appl.* **2024**, *241*, 122638. [CrossRef]
7. Ghasemi, M.; Zare, M.; Zahedi, A.; Trojovský, P.; Abualigah, L.; Trojovská, E. Optimization Based on Performance of Lungs in Body: Lungs Performance-Based Optimization (LPO). *Comput. Methods Appl. Mech. Eng.* **2024**, *419*, 116582. [CrossRef]
8. Su, H.; Zhao, D.; Heidari, A.A.; Liu, L.; Zhang, X.; Mafarja, M.; Chen, H. RIME: A Physics-Based Optimization. *Neurocomputing* **2023**, *532*, 183–214. [CrossRef]
9. Rezaei, F.; Safavi, H.R.; Abd Elaziz, M.; Mirjalili, S. GMO: Geometric Mean Optimizer for Solving Engineering Problems. *Soft. Comput.* **2023**, *27*, 10571–10606. [CrossRef]
10. Abdel-Basset, M.; El-Shahat, D.; Jameel, M.; Abouhawwash, M. Exponential Distribution Optimizer (EDO): A Novel Math-Inspired Algorithm for Global Optimization and Engineering Problems. *Artif. Intell. Rev.* **2023**, *56*, 9329–9400. [CrossRef]
11. Ayyarao, T.S.; Ramakrishna, N.S.S.; Elavarasan, R.M.; Polumahanthi, N.; Rambabu, M.; Saini, G.; Khan, B.; Alatas, B. War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization. *IEEE Access* **2022**, *10*, 25073–25105. [CrossRef]
12. Zolfi, K. Gold Rush Optimizer: A New Population-Based Metaheuristic Algorithm. *Oper. Res. Decis.* **2023**, *33*, 1. [CrossRef]
13. Trojovská, E.; Dehghani, M. A New Human-Based Metahurestic Optimization Method Based on Mimicking Cooking Training. *Sci. Rep.* **2022**, *12*, 14861. [CrossRef] [PubMed]

14. Oladejo, S.O.; Ekwe, S.O.; Akinyemi, L.A.; Mirjalili, S.A. The Deep Sleep Optimizer: A Human-Based Metaheuristic Approach. *IEEE Access* **2023**, *11*, 83639–83665. [CrossRef]
15. Tian, Z.; Gai, M. Football Team Training Algorithm: A Novel Sport-Inspired *Meta*-Heuristic Optimization Algorithm for Global Optimization. *Expert Syst. Appl.* **2024**, *245*, 123088. [CrossRef]
16. Mohapatra, S.; Mohapatra, P. An Improved Golden Jackal Optimization Algorithm Using Opposition-Based Learning for Global Optimization and Engineering Problems. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 147. [CrossRef]
17. Li, W.Y.; Dong, B.L.; Wang, K.; Lian, L.P. Research on resource scheduling in cloud computing environment based on golden jackal optimization algorithm. *Electron. Des. Eng.* **2023**, *31*, 41–45. [CrossRef]
18. Xu, K.; Zhang, H.F. Multi-Objective Optimization of Hydrodynamic Bearing Based on Adaptive Golden Jackal Algorithm. *J. Mech. Strength* **2023**, *45*, 640–645. [CrossRef]
19. Xie, H.; Li, L.J.; Liao, K.; Gao, Z.C. Research on PID parameters optimization based on golden jackal optimization algorithm. *Mod. Manuf. Eng.* **2023**, 146–151. [CrossRef]
20. Chopra, N.; Mohsin Ansari, M. Golden Jackal Optimization: A Novel Nature-Inspired Optimizer for Engineering Applications. *Expert Syst. Appl.* **2022**, *198*, 116924. [CrossRef]
21. Meng, A.; Chen, Y.; Yin, H.; Chen, S. Crisscross Optimization Algorithm and Its Application. *Knowl. Based Syst.* **2014**, *67*, 218–229. [CrossRef]
22. Abdollahzadeh, B.; Soleimanian Gharehchopogh, F.; Mirjalili, S. Artificial Gorilla Troops Optimizer: A New Nature-Inspired Metaheuristic Algorithm for Global Optimization Problems. *Int. J. Intell. Syst.* **2021**, *36*, 5887–5958. [CrossRef]
23. Han, M.; Du, Z.; Yuen, K.F.; Zhu, H.; Li, Y.; Yuan, Q. Walrus Optimizer: A Novel Nature-Inspired Metaheuristic Algorithm. *Expert Syst. Appl.* **2024**, *239*, 122413. [CrossRef]
24. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A Nature-Inspired Metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
25. Li, Y.; Yuan, Q.; Han, M.; Cui, R. Hybrid Multi-Strategy Improved Wild Horse Optimizer. *Adv. Intell. Syst.* **2022**, *4*, 2200097. [CrossRef]
26. Lu, M.; He, D.X.; Qu, L.D. Grey Wolf Optimization Algorithm Based on Elite Learning for Nonlinear Parameters. *J. Guangxi Norm. Univ. (Nat. Sci. Ed.)* **2021**, *39*, 55–67. [CrossRef]
27. Tanabe, R.; Fukunaga, A.S. Improving the Search Performance of SHADE Using Linear Population Size Reduction. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1658–1665.
28. Liu, K.; Zhao, L.L.; Wang, H. Whale Optimization Algorithm Based on Elite Opposition-based and Crisscross Optimization. *J. Chin. Comput. Syst.* **2020**, *41*, 2092–2097.
29. Lagaros, N.D.; Kournoutos, M.; Kallioras, N.A.; Nordas, A.N. Constraint Handling Techniques for Metaheuristics: A State-of-the-Art Review and New Variants. *Optim. Eng.* **2023**, *24*, 2251–2298. [CrossRef]
30. Coello Coello, C.A.; Mezura Montes, E. Constraint-Handling in Genetic Algorithms through the Use of Dominance-Based Tournament Selection. *Adv. Eng. Inform.* **2002**, *16*, 193–203. [CrossRef]
31. Arora, S.; Singh, S. Butterfly Optimization Algorithm: A Novel Approach for Global Optimization. *Soft. Comput.* **2019**, *23*, 715–734. [CrossRef]
32. Dehghani, M.; Trojovský, P. Osprey Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Engineering Optimization Problems. *Front. Mech. Eng.* **2023**, *8*, 1126450. [CrossRef]
33. Mirjalili, S. SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
34. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris Hawks Optimization: Algorithm and Applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]