Computers, Materials & Continua
DOI:10.32604/cmc.2022.020847
Article

Tech Science Press

IRKO: An Improved Runge-Kutta Optimization Algorithm for Global Optimization Problems

R. Manjula Devi[1], M. Premkumar[2], Pradeep Jangir[3], Mohamed Abdelghany Elkotb[4,5], Rajvikram Madurai Elavarasan[6] and Kottakkaran Sooppy Nisar[7,*]

[1]Department of Computer Science and Engineering, Kongu Engineering College, Perundurai, 638060, Tamil Nadu, India
[2]Department of Electrical & Electronics Engineering, Dayananda Sagar College of Engineering, Bengaluru, 560078, Karnataka, India
[3]Rajasthan Rajya Vidyut Prasaran Nigam, Sikar, 332025, Rajasthan, India
[4]Mechanical Engineering Department, College of Engineering, King Khalid University, Abha, 61421, Saudi Arabia
[5]Mechanical Engineering Department, Faculty of Engineering, Kafrelsheikh University, Egypt
[6]Clean and Resilient Energy Systems (CARES) Laboratory, Texas A&M University, Galveston, 77553, TX, USA
[7]College of Arts and Sciences, Prince Sattam bin Abdulaziz University, Wadi Aldawaser, 11991, Saudi Arabia
[*]Corresponding Author: Kottakkaran Sooppy Nisar. Email: n.sooppy@psau.edu.sa

Received: 11 June 2021; Accepted: 22 July 2021

Abstract: Optimization is a key technique for maximizing or minimizing functions and achieving optimal cost, gains, energy, mass, and so on. In order to solve optimization problems, metaheuristic algorithms are essential. Most of these techniques are influenced by collective knowledge and natural foraging. There is no such thing as the best or worst algorithm; instead, there are more effective algorithms for certain problems. Therefore, in this paper, a new improved variant of a recently proposed metaphorless Runge-Kutta Optimization (RKO) algorithm, called Improved Runge-Kutta Optimization (IRKO) algorithm, is suggested for solving optimization problems. The IRKO is formulated using the basic RKO and local escaping operator to enhance the diversification and intensification capability of the basic RKO version. The performance of the proposed IRKO algorithm is validated on 23 standard benchmark functions and three engineering constrained optimization problems. The outcomes of IRKO are compared with seven state-of-the-art algorithms, including the basic RKO algorithm. Compared to other algorithms, the recommended IRKO algorithm is superior in discovering the optimal results for all selected optimization problems. The runtime of IRKO is less than 0.5 s for most of the 23 benchmark problems and stands first for most of the selected problems, including real-world optimization problems.

Keywords: Engineering design; global optimization; local escaping operator; metaheuristics; Runge-Kutta optimization algorithm

1 Introduction

The term "optimization" relates to determining the best values of different system components to complete the systems engineering at the lowest possible price. In machine learning and artificial intelligence, practical applications and problems are typically constrained, unconstrained, or discrete [1,2]. As a result, finding the best solutions using standard numerical programming approaches is difficult. Numerous optimization methods have been established in recent years to enhance the performance of various systems and reduce computing costs. Conventional optimization techniques have several flaws and restrictions, such as convergence to a local minimum and an undefined search space. Many novel optimization approaches have been presented in recent years to address such flaws. Most systems are nonlinear and require advanced optimization techniques to be evaluated. Mathematical optimization techniques are often inspired by findings in nature. Such algorithms produce random solutions inside the search space and keep making improvements until these results converge to probable global optimums. Every metaheuristic algorithm does have its settings that influence the optimization process [3–6]. In general, each metaheuristic algorithms share common characteristics, such as the search strategy, which itself is divided into two sections, one of which is known as diversification and the latter of which has been known as intensification. The first phase of the algorithm creates random variables in the initial phase to explore potential search space locations. The optimization approach attempts to discover the best result from the search space in the second phase. An excellent optimization algorithm must balance the exploitation and exploration phase to prevent entrapment at the local minima. The highest complications of such algorithms are in having an exact balance between exploitation and exploration phases to avoid trap from local minima and still enhance the solution accuracy previously obtained [7–9]. As per the discussion by [10], there are no bad or good among the various optimization techniques, but one more suitable for a certain problem, that is, still has a gap for developing new optimization techniques.

The authors want to provide a more efficient and successful method, and hence this paper proposes an Improved Runge-Kutta Optimization (IRKO) algorithm, a revolutionary metaphor-free optimization method. The suggested IRKO is formulated from the basic version of the RKO algorithm [11], and it employs a particular slope computation ideology based on the Runge-Kutta technique as a search engine for optimization problems. To improve the solution quality of the IRKO, the Local Escaping Operator (LEO) [12] is integrated with the RKO. To thoroughly demonstrate the robustness and efficacy of the IRKO, a set of twenty-three classical benchmarks and three constrained engineering problems is employed.

The paper has been organized as follows: Section 2 confers the backgrounds of metaheuristic algorithms related works, and Section 3 discusses the formulation of the IRKO algorithm employing RKO and LEO concepts. Section 4 demonstrates the numerical simulation results on classical benchmark functions and real-world problems. Section 5 concludes the paper.

2 Literature Review

Metaheuristic algorithms are divided into four categories relying on the motivation for its development: (i) swarm behaviors, (ii) natural behaviors, (iii) human behaviors, and (iv) physics concepts.

2.1 Swarm Intelligence Algorithms

Swarms can inspire natural and social phenomena. Several algorithms have been developed and presented by many researchers. Particle Swarm Optimization (PSO) is a common approach derived from the natural behavior of swarming particles [13]. Each particle can be updated as per the local positions and its global best position. The PSO addresses various challenges, such as solving global optimization problems, power systems, power electronics, feature selection, image segmentation, electric vehicles, data clustering, and much more. Ant foraging behavior inspires the Ant Colony Optimization (ACO) algorithm [14]. The ants release pheromone on the earth naturally to ensure the colony members choose the ideal path. It has been employed in several optimization problems, as stated earlier. Firefly Algorithm (FA) is based on the flashing strobe of fireflies in the sea [15]. It was well-received and implemented in many applications, such as feature selection, data clustering, mechanical systems, power systems, and other optimization challenges. The Artificial Bee Colony (ABC) algorithm is designed to mimic the activity of a honey bee [16]. It contains three groups: bees who are working, onlookers that are observing, and Scouters who are searching for food. The ABC algorithm has numerous applications, such as global optimization, job shop scheduling, wireless sensor network, image segmentation, etc. In addition, various algorithms proved to be effective for optimization problems, such as Gray Wolf Optimizer (GWO) [17], Harris-Hawks Optimizer (HHO) [18], Whale Optimization Algorithm (WOA) [19], Aquila optimizer [20], slime mold algorithm [21], marine predator algorithm [22], equilibrium optimizer [23], salp swarm optimizer [24], etc.

2.2 *Evolutionary Algorithms*

Numerous techniques presented in the literature use natural evolution's inherent characteristics to address optimization challenges. Examples of several evolutionary algorithms are as follows. The Genetic Algorithm (GA) is the most often used evolutionary technique [25]. The GA was invented after Darwinian evolution. It has gained much interest and has been widely applied in various applications, such as power systems, power electronics, electric vehicles, facial recognition, network anomaly detection, and scheduling problems. The authors of [26] demonstrate differential evolution algorithm in 1997, and it is considered to be an effective algorithm for many real-world applications. For instance, text classification, global optimization, image classification, parallel machine scheduling, power systems, smart grid, microgrid, wireless sensor networks, etc. Other famous algorithms, called biogeography-based optimizer [27], sunflower optimization [28], invasive tumor growth [29], etc., have been proven in numerous optimization tasks.

2.3 *Human-Based Algorithms*

Researchers presented numerous metaheuristic algorithms by mimicking real human actions. Teacher Learning-Based Optimizer (TLBO) is inspired by teachers' impact on their student's achievement [30]. Various problems have been solved by applying TLBO, including problems with constraints. Socio-evolution learning optimizer is developed by mimicking the human's social learning grouped as families in a social environment. Additional human-based algorithms, such as Hunger Games Search Optimizer (HGSO) [31], political optimizer [32], harmony search [33], Jaya algorithm [34], Rao algorithm [35], etc., are some of the best algorithms for solving optimization problems.

2.4 *Physics-Based Algorithms*

To provide alternatives to optimization problems, physics-based algorithms rely on physical laws. Big Bang-Big Crunch is a prominent MH algorithm inspired by the universe's development, and it has been applied in various applications, such as data clustering, global optimization, and various engineering design problems. The law of gravity and mass relations inspired the Gravitational Search Algorithm (GSA) [36]. It has also gotten much press and has been utilized to enhance and address various applications, such as global optimization, feature selection, image segmentation, and so on, which are only a few examples. The multi-verse optimizer is based on physics' multi-verses hypothesis. It has handled various problems, including global optimization, forecasting, power systems, feature selection, etc. Other physics-based algorithms, such as the Sine-Cosine Algorithm (SCA) [37], Ion-Motion Optimizer (IMO) [38], and equilibrium optimizer [23], are few best algorithms for solving optimization problems.

In general, population-based optimization algorithms begin procedures by randomly selecting candidate solutions. Such solutions are improved significantly by the algorithms and iteratively assessed against a specified fitness function, which is the basis of any algorithms. Due to the stochastic nature, obtaining an optimal or near-optimal solution in a single run is not sure. However, many random solutions and evolutionary rounds increase the possibility of discovering the global optima for the given problem. Regardless of the differences in algorithms used in metaheuristic approaches, the optimization process may be separated into two distinct phases: exploration and exploitation. This refers to the broad scope of searching by employing several solutions provided by the algorithms to circumvent search difficulties.

3 Improved Runge-Kutta Optimization (IRKO) Algorithm

Firstly, this section discusses the concepts of the RKO algorithm and LEO and then extends the discussion to the development of the IRKO algorithm using RKO and LEO.

3.1 Runge-Kutta Optimization (RKO) Algorithm

The RKO algorithm is characterized by the absence of metaphors and strict attention to the underlying mathematical structures [11]. It is inappropriate to describe the population-based technique's framework using metaphors, as doing so conceals the basis of the mathematics that power the optimizers. Thus, RKO was established considering the logic of the Runge-Kutta (RK) approach and the growth of a population of individuals. The RK employs a special formulation to handle ordinary differential equations. The fundamental premise of RKO is based on the RK method's suggested estimated gradient. The RKO employs that as a method of exploring the search space to form a set of rules. The initialization phase is the first step in algorithms, and this phase of the RKO is similar to other population-based algorithms. This section discusses only the main mathematical formulation RKO algorithm.

3.1.1 Search Mechanism

The solution space is divided into regions, with random solutions placed in each region. A set of search solutions are then placed into the various regions, and a balance between exploitation and exploration is established. To identify the search strategy in the RKO algorithm, the RK4 approach was applied. Thus, the search mechanism (SM) in RKO is represented in Eq. (1).

$$SM = \frac{1}{6}(x_{RK})\Delta x \tag{1}$$

$$x_{RK} = k_1 + 2 \times k_2 + 2 \times k_3 + k_4 \tag{2}$$

where k_1, k_2, k_3, k_4 are coefficients of the first-order derivative by the RK and Δx is position increment.

3.1.2 Solution Update

The RKO algorithm starts the procedure with a random population (solutions). Every time around, the solutions' positions are updated by the RK method. RKO employs a solution and the search mechanism generated using the Runge-Kutta approach. To offer exploration and exploitation search, Eq. (3) is utilized.

$$\left.\begin{array}{l} \textbf{\textit{if}}\ rand < 0.5 \\ x_{n+1} = (x_c + r.SF.g.x_c) + SF.SM + \mu.randn.(x_m - x_c)\ \textbf{(exploration)} \\ \textbf{\textit{else}} \\ x_{n+1} = (x_m + r.SF.g.x_m) + SF.SM + \mu.\,randn.(x_{r1} - x_{r2})\ \textbf{(exploitation)} \\ \textbf{\textit{end}} \end{array}\right\} \tag{3}$$

where μ denote random numbers between [0, 1], r and g denote random numbers between [1, −1] and [0, 2], respectively, and SF denotes adaptive factor and is given in Eq. (4).

$$SF = 2.(0.5 - rand) \times f \tag{4}$$

$$f = a \times exp\left(-b \times rand \times \left(\frac{IT}{MaxIT}\right)\right) \tag{5}$$

where a and b are constant numbers, IT denotes current iteration, and $MaxIT$ denotes the maximum number of iterations. The expressions for x_c and x_m are presented in Eq. (6).

$$\left.\begin{array}{l} x_c = \varphi \times x_n + (1-\varphi) \times x_{r1} \\ x_m = \varphi \times x_{best} + (1-\varphi) \times x_{lbest} \end{array}\right\} \tag{6}$$

where φ denotes a random number between [0, 1], x_{best} denotes the best solution, and x_{lbest} denotes the current best solution.

3.1.3 Enhanced Solution Quality (ESQ)

The RKO employs ESQ to maximize the quality of the solution across iterations while avoiding local optima. Here's how the solution (x_{new2}) is created utilizing the ESQ.

$$\left.\begin{array}{l} \textbf{\textit{if}}\ rand < 0.5 \\ \textbf{\textit{if}}\ w < 1 \\ x_{new2} = x_{new1} + r.w.|(x_{new1} - x_{avg}) + randn| \\ \textbf{\textit{else}} \\ x_{new2} = (x_{new1} - x_{avg}) + r.w.|(u.x_{new1} - x_{avg}) + randn| \\ \textbf{\textit{end}} \\ \textbf{\textit{end}} \end{array}\right\} \tag{7}$$

$$w = rand(0,2).exp\left(-c\left(\frac{IT}{MaxIT}\right)\right) \tag{8}$$

$$x_{avg} = \frac{x_{r1} + x_{r2} + x_{r3}}{3} \tag{9}$$

$$x_{new1} = \beta \times x_{avg} + (1 - \beta) \times x_{best} \tag{10}$$

where β signifies a random number between [0, 1], c signifies a random number equal to $5 \times rand$, r denotes an integer number equal to 1, 0, or -1. The solution (x_{new2}) may not have improved solution than the current solution (i.e., $f(x_{new2}) > f(x_n)$). To produce the best fitness again, alternative new solution (x_{new3}) is produced, which is represented as follows:

$$\left.\begin{array}{l} \boldsymbol{if}\ rand < \boldsymbol{w} \\ x_{new3} = (x_{new2} - rand.x_{new2}) + SF.(rand.x_{RK} + (v.x_b - x_{new2})) \\ \boldsymbol{end} \end{array}\right\} \tag{11}$$

where v denotes a random number equal to $2 \times rand$. Eq. (11) is about moving the x_{new2} toward a better location. The search around x_{new2} begins in the first criterion, and then RKO seeks to find potential directions to progress towards the right decision. The pseudocode of RKO is shown in *Algorithm 1*. For comprehensive details of RKO, the readers are encouraged to read the base paper.

Algorithm 1: Pseudocode of RKO Algorithm

```
Initialize $a, b$ and generate the population $X_n$ $(n = 1, 2, \ldots, N_p)$ and calculate the fitness of each population
Determine the solutions $x_w$, $x_b$, and $x_{best}$
      for $n = 1{:}N_p$
        for $l = 1{:}dim$
          Compute position $x_{n+1,l}$ via Eq. (3)
        end for
        if $rand < 0.5$ (ESQ)
          Calculate the position $x_{new2}$ via Eq. (7)
            if $f(x_n) < f(x_{new2})$
              if rand $< w$
                  Calculate position $x_{new3}$ via Eq. (11)
              end if
            end if
        end if
        Update the positions $x_w$ and $x_b$
      end for
      Update the position $x_{best}$
      $i = i + 1$
   end for
Return $x_{best}$
```

3.2 *Local Escaping Operator*

The authors of [3,6] presented an operator called the LEO, which is employed to boost the capabilities of a gradient-based optimizer. The LEO's updates ensure that the solutions are of superior quality. It can bypass local minima traps, and hence, it enhances convergence. LEO produces its x_{r1}^{m} and x_{r2}^{m} viable alternatives, high-performing by implementing various solutions such as the best location x_{best}, two randomly produced solutions, and a new randomly generated solution x_k^m. Therefore, X_{LEO}^m can be calculated using Eq. (12).

```
if $rand < pr$
    if $rand < 0.5$
    $X_{LEO}^m = X_x^{m+1} + a \times (a_1 \times x_b - a_2 \times x_k^m) + b \times \rho_1 \times (a_3 \times (X2_x^m - X1_x^m) + a_2 \times (x_{r1}^m - x_{r2}^m))/2$
    $X_x^{m+1} = X_{LEO}^m$
    Else                                                                     (12)
    $X_{LEO}^m = x_b + a \times (a_1 \times x_b - a_2 \times x_k^m) + b \times \rho_1 \times (a_3 \times (X2_x^m - X1_x^m) + a_2 \times (x_{r1}^m - x_{r2}^m))/2$
    End
End
```

where b is a normal distributed random number, pr represents the probability rate, and a represents a uniform random number in the range of $[-1, 1]$. The values of random numbers a_1, a_2, and a_3 are obtained using Eq. (13).

$$\left.\begin{aligned} a_1 &= Z_1 \times rand \times 2 + (1 - Z_1) \\ a_2 &= a_3 = Z_1 \times rand + (1 - Z_1) \end{aligned}\right\} \tag{13}$$

The Z_1 is equal to 1 or 0. The expression for x_k^m is given in Eq. (14).

$$\left.\begin{aligned} x_k^m &= \begin{cases} x_{rand}, & \text{if } u_2 < 0.5 \\ x_p^m, & \text{otherwise} \end{cases} \\ x_{rand} &= X_l + rand \times (X_u - X_l) \end{aligned}\right\} \tag{14}$$

where x_{rand} represents the updated solution, u_2 represents a random number between [0, 1], and x_p^m represents the random population solution. Therefore, Eq. (14) is altered, as shown in Eq. (15).

$$x_k^m = Z_2 \times x_p^m + (1 - Z_2) \times x_{rand} \tag{15}$$

The value of Z_2 is 1 or 0 based on the u_2. The readers are encouraged to read the base paper [6] for better understanding.

3.3 Development of Improved Runge-Kutta Optimization Algorithm

RKO algorithm struggles from being "trapped in the local minima situation." Optimization could not be completed since the local region confined the system. This scenario often occurs in difficult and high-dimensional optimization problems. Additionally, producing new solutions is based on the results of the last iteration. This could slow down the algorithm's convergence speed, and hence, cause early convergence of the solutions. To increase search capabilities and handle difficult real-world problems, we enhance the RKO algorithm using the LEO concept. The proposed IRKO algorithm follows the RKO algorithm procedure step-by-step. LEO improves the diversification and intensification phase of the RKO algorithm along with the RK method. The initialization phase of the IRKO is similar to the RKO algorithm.

To examine the suggested solutions and improve the new suggested solutions in the next iteration, it is necessary to analyze the solutions in each iteration. As a result, each population is assessed to acquire its solution when each new population is randomly produced, an update that permits the new fitness to be upgraded using Eq. (3). Then, the new population is improved by the LEO mechanism, as mentioned in Eq. (12). Finally, the best position is achieved after updating the population using Eqs. (7) and (11) as similar to the RKO algorithm. Iterate the suggested IRKO until it meets the terminating condition. After that, the best option so far is discovered. The flowchart of the IRKO is shown in Fig. 1.

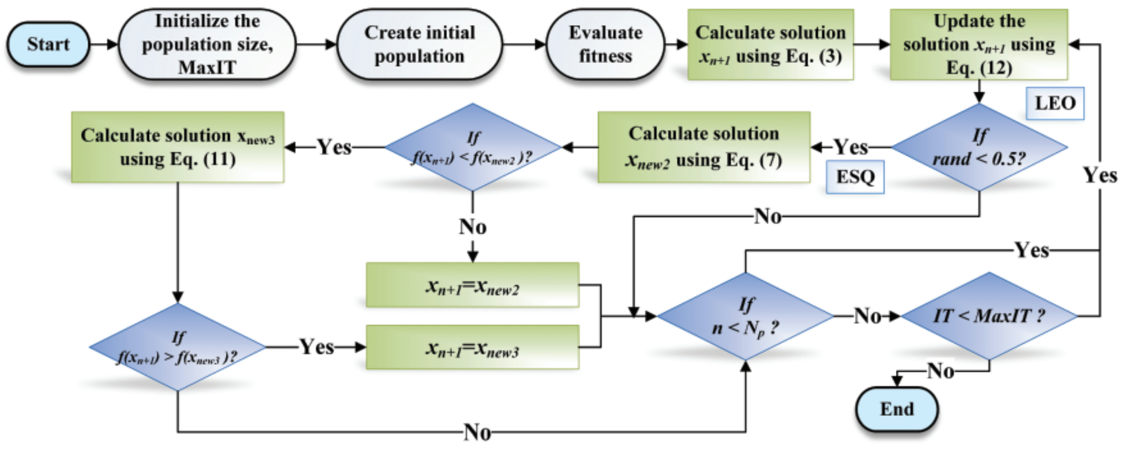

Start
Initialize the population size, MaxIT
Create initial population
Evaluate fitness
Calculate solution x_{n+1} using Eq. (3)
Update the solution x_{n+1} using Eq. (12)
LEO
If $rand < 0.5$?
Yes
ESQ
Calculate solution x_{new2} using Eq. (7)
If $f(x_{n+1}) < f(x_{new2})$?
Yes
Calculate solution x_{new3} using Eq. (11)
No
$x_{n+1}=x_{new2}$
If $f(x_{n+1}) > f(x_{new3})$?
Yes
$x_{n+1}=x_{new3}$
No
If $n < N_p$?
No
$IT < MaxIT$?
Yes
No
End

Figure 1: Flowchart of the IRKO algorithm

4 Numerical Simulation and Discussions

The performance of the IRKO is validated using twenty-three classical test benchmark functions and three real-world engineering problems. The IRKO has been simulated with a 30-population size and 500 maximum iterations to solve the benchmarks and engineering problems. The other control parameters are as follows. RKO and IRKO ($a = 20$ and $b = 12$), GWO and WOA ($a = 2 - 2 \times IT/MaxIT$), HGSO ($p_{CR} = 0.8$, $\beta_{min} = 0.2$, and $\beta_{max} = 0.8$), SCA ($a = 2$), IMO (Φ_1 and $\Phi_2 = 0.5$), and HHO ($\beta = 1.5$). The performance of the IRKO algorithm is compared with seven algorithms, namely GWO, HHO, IMO, SCA, HGSO, WOA, and the basic RKO version. To assess the steadiness and reliability, all algorithms have been run 30 times. The Min, Mean, and Standard Deviation (STD) values are reported for all chosen algorithms. To check the IRKO superiority, Friedman's rank test (FRT) is conducted, and results are reported. All algorithms have been executed with the same population and number of iterations to have a reasonable assessment.

4.1 Benchmark Test Suites

Twenty-three benchmark functions, including unimodal, multi-modal, and fixed dimension multi-modal functions, were used to examine the IRKO's ability to exploit global solutions, explore the search space, and escape from the local minima trap. The benchmark test functions are provided in the weblink [39] for the reader's reference, in which the exploitation potential is assessed using the unimodal benchmarks (F1–F7), multi-modal benchmarks (F8–F13) meant to assess the exploration ability with a dimension of 30, and fixed dimension multi-modal test benchmarks (F14–F23) demonstrate the ability to explore low-dimensional search spaces. Several well-known optimization algorithms, such as GWO, HHO, SCA, IMO, HGSO, RKO, and WOA, have been evaluated against the same functions to confirm the superiority of the suggested IRKO algorithm. The simulation was carried out using a MATLAB 2020a installed on a laptop with an Intel Core i5 processor, 2.4 GHz clock speed, and 8 GB of RAM.

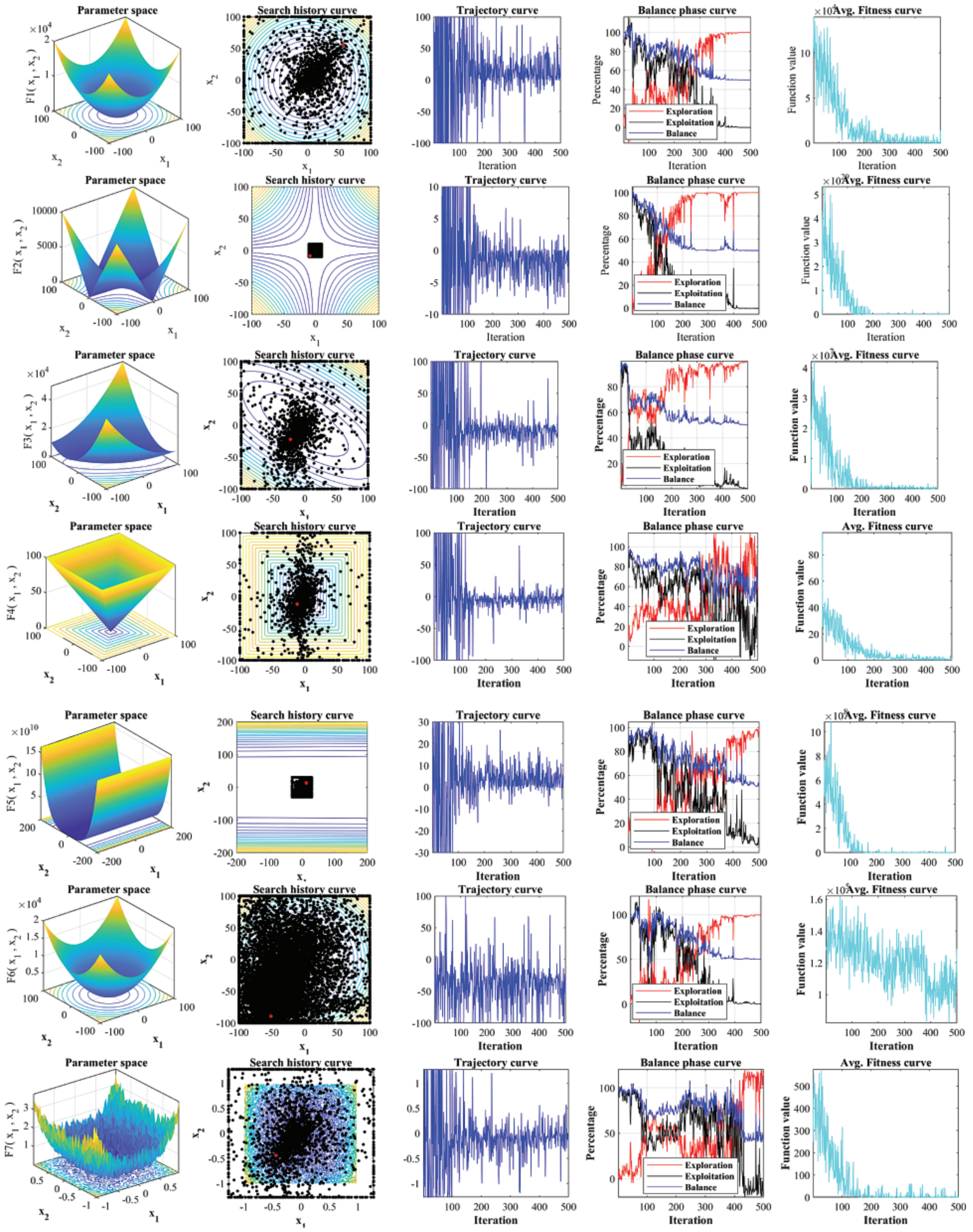
To confirm the performance of the proposed IRKO algorithm, the balance phase curve, average fitness curve, and trajectory curves are utilized and have been in Fig. 2. Fig. 2 shows the qualitative measures, such as the 2-D figure of all test functions (first column) and search history to confer the solutions in the search space (second column). The trajectory curves, balancing phase curves, and the average fitness curves are displayed in the third, fourth, and fifth columns of Fig. 2. In addition, Min, Mean, STD, and Run-Time (RT) values obtained by all selected algorithms are listed in Tab. 1. The boldface letters in the table indicate the best values. From Tab. 1, it is observed that the proposed IRKO produced the best results in terms of Min, Mean, and STD values on all 23 benchmark functions. However, the RT values of IRKO are slightly higher than the basic RKO version. This is due to the deployment of the LEO concept with the RKO algorithm. However, the proposed IRKO produces excellent performance in terms of reliability and robustness.

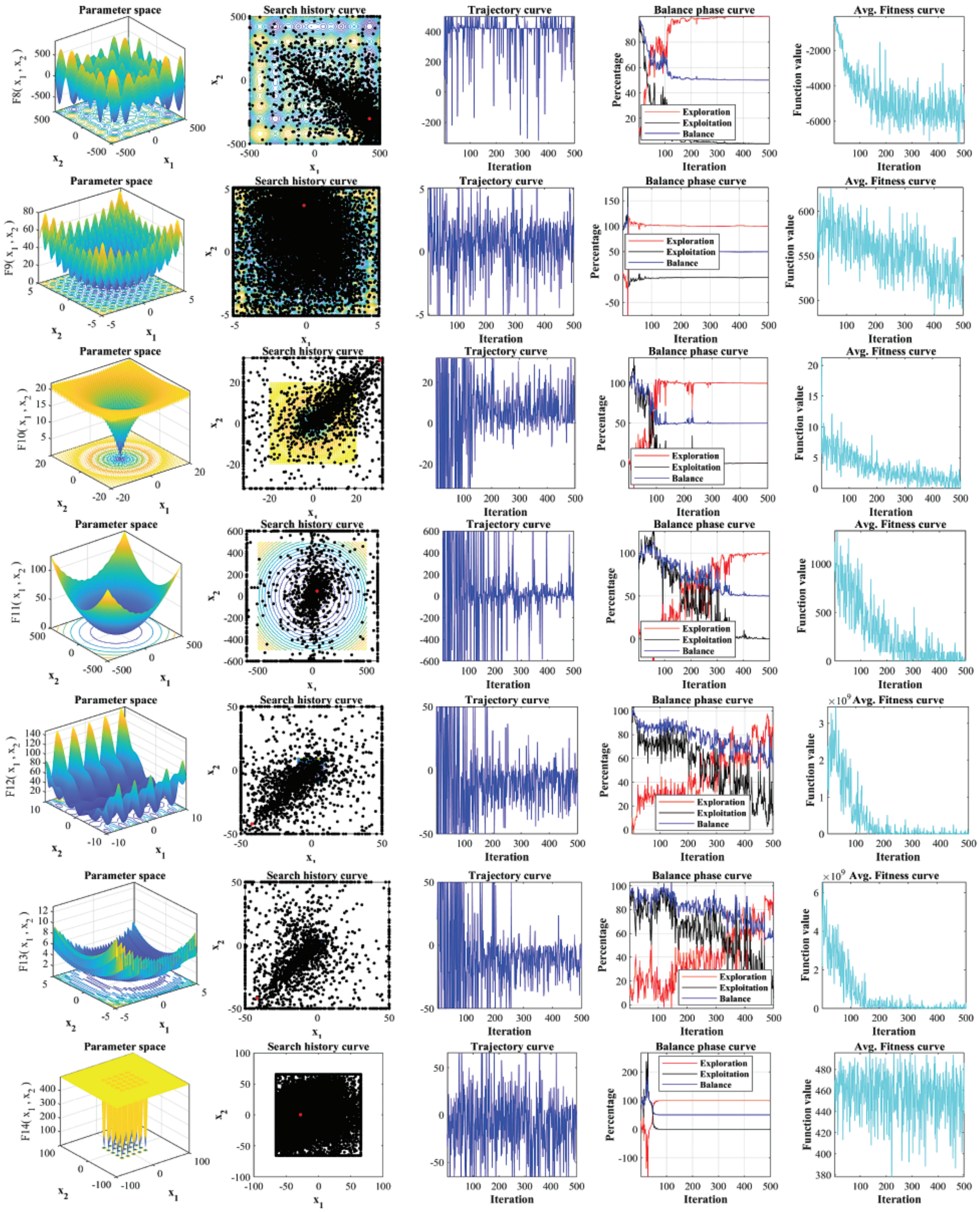
The search's history explains how the algorithm search for the optimal solution in search space. The LEO concept enhances the behavior of the IRKO. This modality depicts the movement of the population about the best position for unimodal functions. The nature of the dispersion qualities of populations corresponds to the modality. When using IRKO to address multi-modal and unimodal functions, the exploration and exploitation capabilities are all enhanced. The trajectory curve indicated a high amplitude and frequency in the initial iterations and disappeared during the final iteration. As can be seen, IRKO has a high exploration ability early on and strong exploitation later on. Based on this behavior, the ideal approach is likely for the IRKO algorithm. The LEO in IRKO algorithm promotes the search optimization process to precisely and broadly focus on the local region. Compared to other recently proven methods, the LEO aids the RKO in efficiently and accurately discovering the search space. Fig. 3 illustrates the best solutions achieved so far throughout the iteration numbers. IRKO's convergence curves show a visible decay rate for unimodal functions. Meanwhile, other algorithms have a severe standstill, making the exploitation stage of the IRKO have a reliable exploration capability. The IRKO's multi-modal functions' convergence curves indicate the seamless transition between exploitation and exploration.

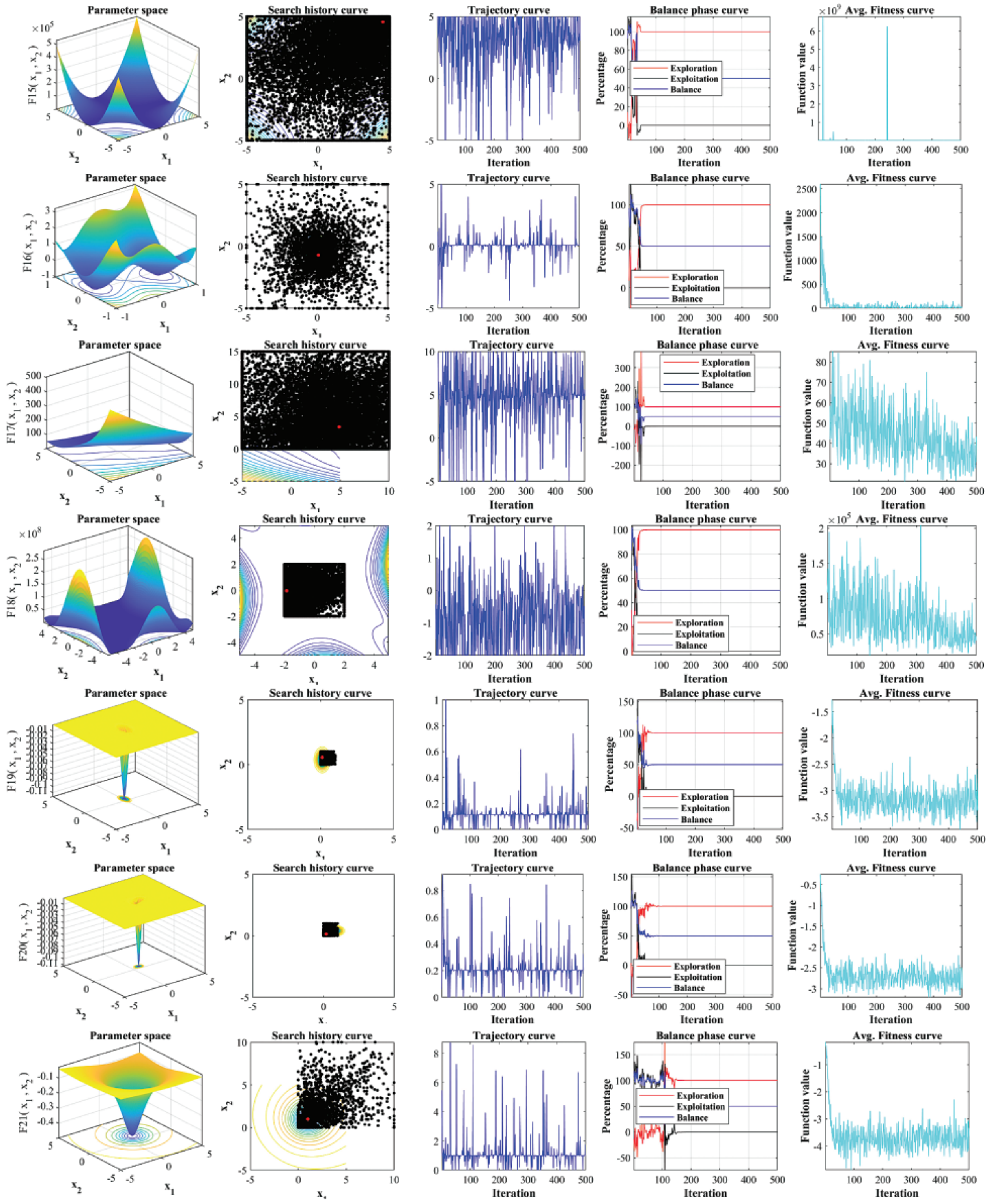
It converges quickly with better results than the other counterparts in most of the remaining test functions. This is evident when it's understood that the IRKO performs well between exploitation and exploration stages since it captures nearby values for the best solutions. These solutions are exploited proficiently throughout the iterations to offer the best solutions. In addition, the reliability of all selected algorithms is assessed by visualizing the boxplot. Fig. 4 shows the boxplot analysis of all algorithms on selected benchmarks. It is visualized that the reliability of the proposed IRKO is superior to all other selected algorithms. To rank all algorithms, Friedman's rank test is carried out, and the FRT values of all algorithms are listed in Tab. 2. It is observed from Tab. 2 that the proposed IRKO algorithm stands first in most of the benchmark functions.

4.2 Real-World Optimization Problems

In this subsection, the IRKO algorithm has been tested in addressing three real-world engineering design problems, including welded beam, tension/compression spring, and pressure vessel. All these design problems have many inequality constraints. The method gets substantial solutions if it violates any of the criteria using the death penalty function.


Parameter space
Search history curve
Trajectory curve
Balance phase curve
Avg. Fitness curve
F1(x1 , x2)
F2(x1 , x2)
F3(x1 , x2)
F4(x1 , x2)
F5(x1 , x2)
F6(x1 , x2)
F7(x1 , x2)
Percentage
Function value
Iteration
Exploration
Exploitation
Balance


Parameter space
Search history curve
Trajectory curve
Balance phase curve
Avg. Fitness curve
F8(x1 , x2)
F9(x1 , x2)
F10(x1 , x2)
F11(x1 , x2)
F12(x1 , x2)
F13(x1 , x2)
F14(x1 , x2)
x1
x2
Iteration
Percentage
Function value
Exploration
Exploitation
Balance
×10^9


Parameter space
Search history curve
Trajectory curve
Balance phase curve
Avg. Fitness curve
F15(x1, x2)
F16(x1, x2)
F17(x1, x2)
F18(x1, x2)
F19(x1, x2)
F20(x1, x2)
F21(x1, x2)
Exploration
Exploitation
Balance
Percentage
Function value
Iteration

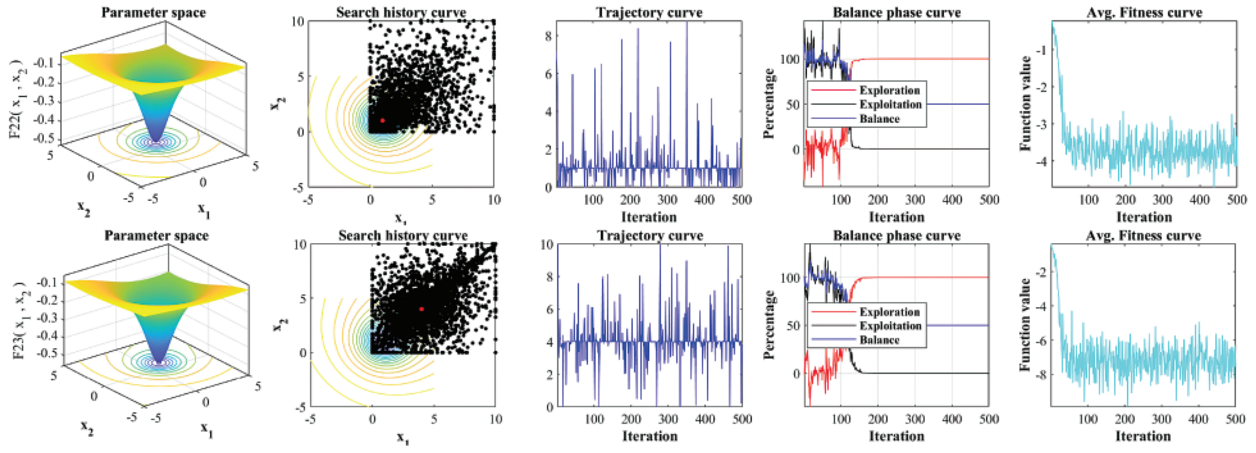


Figure 2: Qualitative results produced by IRKO algorithm on all selected benchmarks

Table 1: Results of all algorithms on classical benchmarks

	Algorithm	Min.	Mean	STD	RT		Algorithm	Min.	Mean	STD	RT
F1	**IRKO**	**9.746E − 238**	**1.319E − 229**	**0.000E + 00**	0.43	F13	**IRKO**	**3.021E − 08**	**3.662E − 03**	**6.344E − 03**	0.45
	RKO	1.433E − 175	2.212E − 170	0.000E + 00	**0.2**		RKO	3.792E − 08	1.067E − 02	1.052E − 02	**0.38**
	SCA	7.572E − 02	4.986E − 01	3.897E − 01	4.12		SCA	1.964E + 01	5.154E + 03	6.063E + 03	1.97
	WOA	3.277E − 85	4.060E − 74	7.032E − 74	6.82		WOA	1.862E − 01	2.686E − 01	8.218E − 02	6.09
	IMO	5.897E − 38	6.075E − 31	1.052E − 30	2.06		IMO	1.001E − 02	4.227E − 02	5.318E − 02	1.45
	HGSO	5.607E + 00	7.458E + 00	1.868E + 00	4.1		HGSO	2.325E + 00	3.937E + 00	1.396E + 00	4.54
	GWO	5.557E − 29	9.856E − 28	1.419E − 27	0.36		GWO	4.168E − 01	6.979E − 01	3.739E − 01	0.49
	HHO	7.745E − 107	7.672E − 100	1.326E − 99	0.45		**HHO**	**8.529E − 06**	**4.219E − 05**	**5.806E − 05**	0.98
F2	**IRKO**	**4.099E − 125**	**2.697E − 122**	**3.737E − 122**	0.14	F14	**IRKO**	**9.980E − 01**	**9.980E − 01**	**1.570E − 16**	**0.69**
	RKO	7.275E − 106	3.578E − 93	6.197E − 93	**0.1**		RKO	2.982E + 00	2.982E + 00	3.608E − 15	0.72
	SCA	2.631E − 03	2.779E − 02	4.247E − 02	1.49		SCA	1.002E + 00	2.322E + 00	1.143E + 00	2.65
	WOA	4.006E − 54	1.063E − 51	1.397E − 51	5.83		WOA	1.992E + 00	5.246E + 00	4.804E + 00	6.35
	IMO	3.394E − 36	1.212E − 24	2.098E − 24	1.12		IMO	9.980E − 01	1.661E + 00	5.739E − 01	2.06
	HGSO	7.212E + 00	1.490E + 01	7.526E + 00	3.06		HGSO	9.980E − 01	1.329E + 00	5.739E − 01	2.48
	GWO	6.323E − 17	1.085E − 16	5.997E − 17	0.19		GWO	9.980E − 01	4.914E + 00	5.161E + 00	0.92
	HHO	1.861E − 56	2.972E − 55	3.742E − 55	0.17		HHO	9.980E − 01	9.980E − 01	3.847E − 10	1.78
F3	**IRKO**	**8.149E − 222**	**1.329E − 213**	**0.000E + 00**	0.38	F15	IRKO	1.223E − 03	1.223E − 03	2.713E − 18	0.07
	RKO	6.149E − 149	1.051E − 144	1.817E − 144	**0.3**		**RKO**	**3.075E − 04**	**6.127E − 04**	**5.287E − 04**	**0.07**
	SCA	3.530E + 03	1.516E + 04	1.164E + 04	1.82		SCA	7.653E − 04	8.150E − 04	6.242E − 05	1.19
	WOA	3.367E + 04	4.919E + 04	1.421E + 04	6.01		WOA	4.976E − 04	9.951E − 04	6.649E − 04	5.59
	IMO	1.687E + 04	3.508E + 04	1.638E + 04	1.23		IMO	3.111E − 04	7.898E − 04	6.009E − 04	0.7
	HGSO	5.593E + 02	1.526E + 04	2.539E + 04	4.16		HGSO	7.719E − 04	1.954E − 03	2.039E − 03	0.49
	GWO	3.861E − 07	2.248E − 06	2.028E − 06	0.43		GWO	3.699E − 04	7.046E − 03	1.153E − 02	0.07
	HHO	2.482E − 100	4.529E − 72	7.845E − 72	0.8		HHO	3.192E − 04	9.988E − 04	5.952E − 04	0.2
F4	**IRKO**	**1.366E − 118**	**4.581E − 115**	**7.684E − 115**	0.15	F16	**IRKO**	**− 1.032**	**− 1.032**	**0.000E + 00**	**0.05**
	RKO	4.044E − 86	9.464E − 80	1.637E − 79	**0.07**		RKO	− 1.032	− 1.032	4.313E − 14	0.05
	SCA	1.654E + 01	3.611E + 01	2.026E + 01	1.3		SCA	− 1.032	− 1.032	3.138E − 05	1.13
	WOA	7.001E + 01	7.316E + 01	3.535E + 00	5.96		WOA	− 1.032	− 1.032	1.075E − 10	5.62
	IMO	8.269E − 03	7.117E + 00	1.128E + 01	0.8		IMO	− 1.032	− 1.032	4.027E − 07	0.65
	HGSO	3.707E + 00	5.463E + 00	2.069E + 00	2.72		**HGSO**	**− 1.032**	**− 1.032**	**0.000E + 00**	0.47
	GWO	5.385E − 07	5.911E − 07	7.161E − 08	0.19		GWO	− 1.032	− 1.032	5.791E − 08	0.05
	HHO	5.767E − 52	4.996E − 50	4.563E − 50	0.17		HHO	− 1.032	− 1.032	8.135E − 12	0.16

(Continued)

Table 1: Continued

	Algorithm	Min.	Mean	STD	RT		Algorithm	Min.	Mean	STD	RT
F5	**IRKO**	**3.840E − 03**	**1.643E − 02**	**1.488E − 02**	0.17	F17	**IRKO**	**3.979E − 01**	**3.979E − 01**	**0.000E + 00**	**0.08**
	RKO	2.442E + 01	2.500E + 01	5.879E − 01	**0.1**		RKO	3.979E − 01	3.979E − 01	5.612E − 13	0.08
	SCA	7.996E + 02	2.979E + 03	3.455E + 03	1.4		SCA	3.979E − 01	4.001E − 01	2.650E − 03	3
	WOA	2.748E + 01	2.790E + 01	4.165E − 01	5.79		WOA	3.979E − 01	3.979E − 01	2.315E − 07	5.29
	IMO	2.775E + 01	2.830E + 01	4.966E − 01	0.85		IMO	3.979E − 01	3.979E − 01	1.440E − 05	1.38
	HGSO	1.498E + 03	2.581E + 03	1.547E + 03	3.04		**HGSO**	**3.979E − 01**	**3.979E − 01**	**0.000E + 00**	1.08
	GWO	2.708E + 01	2.712E + 01	3.469E − 02	0.21		GWO	3.979E − 01	3.979E − 01	5.550E − 06	0.1
	HHO	2.296E + 01	2.386E + 01	1.094E + 00	0.3		HHO	3.979E − 01	3.979E − 01	2.240E − 06	0.36
F6	**IRKO**	**2.308E − 09**	**2.709E − 09**	**6.428E − 10**	0.13	F18	**IRKO**	**3.0**	**3.0**	**1.601E − 15**	0.04
	RKO	2.525E − 09	3.025E − 09	6.239E − 10	**0.08**		RKO	3.0	3.0	2.057E − 12	**0.03**
	SCA	5.223E + 00	1.239E + 01	6.206E + 00	1.32		SCA	3.0	3.0	1.928E − 05	0.95
	WOA	1.649E − 01	3.970E − 01	2.562E − 01	5.74		WOA	3.0	3.0	2.719E − 04	4.4
	IMO	4.586E − 02	4.983E − 02	4.562E − 03	0.8		IMO	3.0	3.0	1.049E − 05	0.55
	HGSO	3.662E + 00	7.069E + 00	3.538E + 00	2.91		HGSO	3.0	3.0	4.283E − 15	0.33
	GWO	2.496E − 01	7.470E − 01	6.597E − 01	0.19		GWO	3.0	3.0	4.790E − 05	0.04
	HHO	1.940E − 04	4.780E − 04	3.059E − 04	0.19		HHO	3.0	3.0	1.838E − 08	0.11
F7	IRKO	1.061E − 04	3.581E − 04	3.931E − 04	0.23	F19	**IRKO**	**−3.863**	**−3.863**	**0.000E + 00**	0.06
	RKO	**2.094E − 05**	**6.540E − 05**	**7.115E − 05**	**0.19**		RKO	−3.863	−3.863	2.560E − 10	**0.05**
	SCA	2.536E − 02	9.712E − 02	1.006E − 01	1.55		SCA	− 3.861	−3.856	4.674E − 03	1.12
	WOA	2.600E − 04	4.012E − 03	4.560E − 03	6		WOA	−3.862	−3.858	4.847E − 03	4.39
	IMO	3.502E − 04	1.225E − 03	8.936E − 04	0.99		IMO	−3.863	−3.863	5.738E − 05	0.59
	HGSO	6.957E + 00	1.350E + 01	6.048E + 00	3.33		HGSO	−3.863	−3.863	7.022E − 16	0.42
	GWO	9.594E − 04	1.187E − 03	2.461E − 04	0.3		GWO	−3.863	−3.863	1.465E − 05	0.06
	HHO	1.933E − 04	3.155E − 04	1.104E − 04	0.42		HHO	−3.863	−3.860	2.938E − 03	0.15
F8	**IRKO**	**− 1.458E + 05**	**− 7.875E + 04**	**5.963E + 04**	0.16	F20	**IRKO**	**−3.322**	**−3.282**	**6.864E − 02**	0.06
	RKO	− 8.518E + 03	− 7.825E + 03	6.011E + 02	**0.09**		RKO	−3.322	−3.282	6.864E − 02	**0.05**
	SCA	− 3.986E + 03	− 3.673E + 03	3.188E + 02	1.43		SCA	−3.114	−3.040	6.420E − 02	1.12
	WOA	− 1.257E + 04	− 1.257E + 04	1.876E + 00	5.84		WOA	−3.320	−3.188	1.181E − 01	4.41
	IMO	− 1.257E + 04	− 1.256E + 04	6.233E + 00	0.85		IMO	−3.322	−3.321	1.510E − 04	0.62
	HGSO	− 1.006E + 04	− 9.126E + 03	9.683E + 02	3.04		HGSO	−3.322	−3.138	2.201E − 01	0.45
	GWO	− 6.260E + 03	− 5.994E + 03	2.308E + 02	0.2		GWO	− 3.322E + 00	− 3.238E + 00	7.309E − 02	0.06
	HHO	− 1.257E + 04	− 1.257E + 04	1.631E + 00	0.29		HHO	− 3.183E + 00	− 3.111E + 00	7.234E − 02	0.19
F9	**IRKO**	**0.000E + 00**	**0.000E + 00**	**0.000E + 00**	0.18	F21	**IRKO**	**− 1.015E + 01**	**− 1.015E + 01**	**0.000E + 00**	**0.06**
	RKO	0.000E + 00	0.000E + 00	0.000E + 00	**0.09**		RKO	− 1.015E + 01	− 1.015E + 01	2.559E − 09	0.07
	SCA	1.184E − 01	5.683E + 00	9.576E + 00	1.38		SCA	− 6.708E + 00	− 4.070E + 00	2.953E + 00	1.2
	WOA	0.000E + 00	0.000E + 00	0.000E + 00	6.15		WOA	− 1.015E + 01	− 8.452E + 00	2.942E + 00	4.47
	IMO	0.000E + 00	0.000E + 00	0.000E + 00	0.8		IMO	− 1.015E + 01	− 1.014E + 01	8.561E − 03	0.64
	HGSO	1.360E + 02	2.253E + 02	7.838E + 01	3.16		HGSO	− 1.015E + 01	− 1.015E + 01	1.130E − 14	0.48
	GWO	1.137E − 13	1.238E + 00	2.144E + 00	0.19		GWO	− 1.015E + 01	− 6.781E + 00	2.918E + 00	0.07
	HHO	0.000E + 00	0.000E + 00	0.000E + 00	0.23		HHO	− 5.055E + 00	− 5.050E + 00	4.512E − 03	0.22
F10	**IRKO**	**8.882E − 16**	**8.882E − 16**	**0.000E + 00**	0.16	F22	**IRKO**	**− 1.040E + 01**	**−1.040E + 01**	**2.413E − 14**	0.08
	RKO	**8.882E − 16**	**8.882E − 16**	**0.000E + 00**	**0.09**		RKO	− 1.040E + 01	− 8.631E + 00	3.069E + 00	**0.07**
	SCA	1.935E + 01	1.989E + 01	4.774E − 01	1.33		SCA	− 4.569E + 00	− 3.519E + 00	9.182E − 01	1.24
	WOA	4.441E − 15	5.625E − 15	2.051E − 15	5.77		WOA	− 1.040E + 01	− 7.852E + 00	4.407E + 00	4.43
	IMO	8.882E − 16	4.441E − 15	3.553E − 15	0.81		IMO	− 1.039E + 01	− 1.032E + 01	1.150E − 01	0.67
	HGSO	1.906E + 01	1.960E + 01	4.718E − 01	3.25		HGSO	− 1.040E + 01	− 7.853E + 00	4.417E + 00	0.5
	GWO	7.905E − 14	8.971E − 14	1.281E − 14	0.21		GWO	− 1.040E + 01	− 1.040E + 01	5.920E − 04	0.08
	HHO	**8.882E − 16**	**8.882E − 16**	**0.000E + 00**	0.24		HHO	− 9.730E + 00	− 6.634E + 00	2.681E + 00	0.24
F11	**IRKO**	**0.000E + 00**	**0.000E + 00**	**0.000E + 00**	0.18	F23	**IRKO**	**−10.5400**	**−8.7340**	**3.122E + 00**	**0.09**
	RKO	**0.000E + 00**	**0.000E + 00**	**0.000E + 00**	**0.11**		RKO	−10.5400	−8.7340	3.122E + 00	0.09
	SCA	7.405E − 01	9.151E − 01	1.612E − 01	1.61		SCA	−4.5300	−2.3760	1.900E + 00	1.25
	WOA	**0.000E + 00**	**0.000E + 00**	**0.000E + 00**	5.8		WOA	−10.5300	−10.5100	2.635E − 02	4.49
	IMO	0.000E + 00	3.701E − 17	6.410E − 17	0.85		IMO	−10.5000	−10.4800	2.656E − 02	0.68
	HGSO	1.890E − 01	2.930E − 01	1.254E − 01	3.04		HGSO	−10.5400	−7.5830	5.115E + 00	0.54
	GWO	**0.000E + 00**	**0.000E + 00**	**0.000E + 00**	0.21		GWO	−10.5400	−10.5400	3.509E − 04	0.1
	HHO	**0.000E + 00**	**0.000E + 00**	**0.000E + 00**	0.28		HHO	−5.1280	−5.1260	2.162E − 03	0.28

(Continued)

Table 1: Continued

	Algorithm	Min.	Mean	STD	RT	Algorithm	Min.	Mean	STD	RT
F12	IRKO	2.310E – 09	3.172E – 09	7.895E – 10	0.59					
	RKO	**2.239E – 09**	**2.445E – 09**	**2.985E – 10**	**0.46**					
	SCA	4.813E + 00	1.804E + 01	1.873E + 01	2.1					
	WOA	1.220E – 02	1.742E – 02	5.945E – 03	6.34					
	IMO	3.234E – 03	3.997E – 03	8.271E – 04	1.52					
	HGSO	5.463E + 00	7.339E + 00	1.809E + 00	4.77					
	GWO	2.555E – 02	3.729E – 02	1.371E – 02	0.53					
	HHO	5.174E – 09	6.733E – 07	6.172E – 07	1.01					

4.2.1 Welded Beam Design

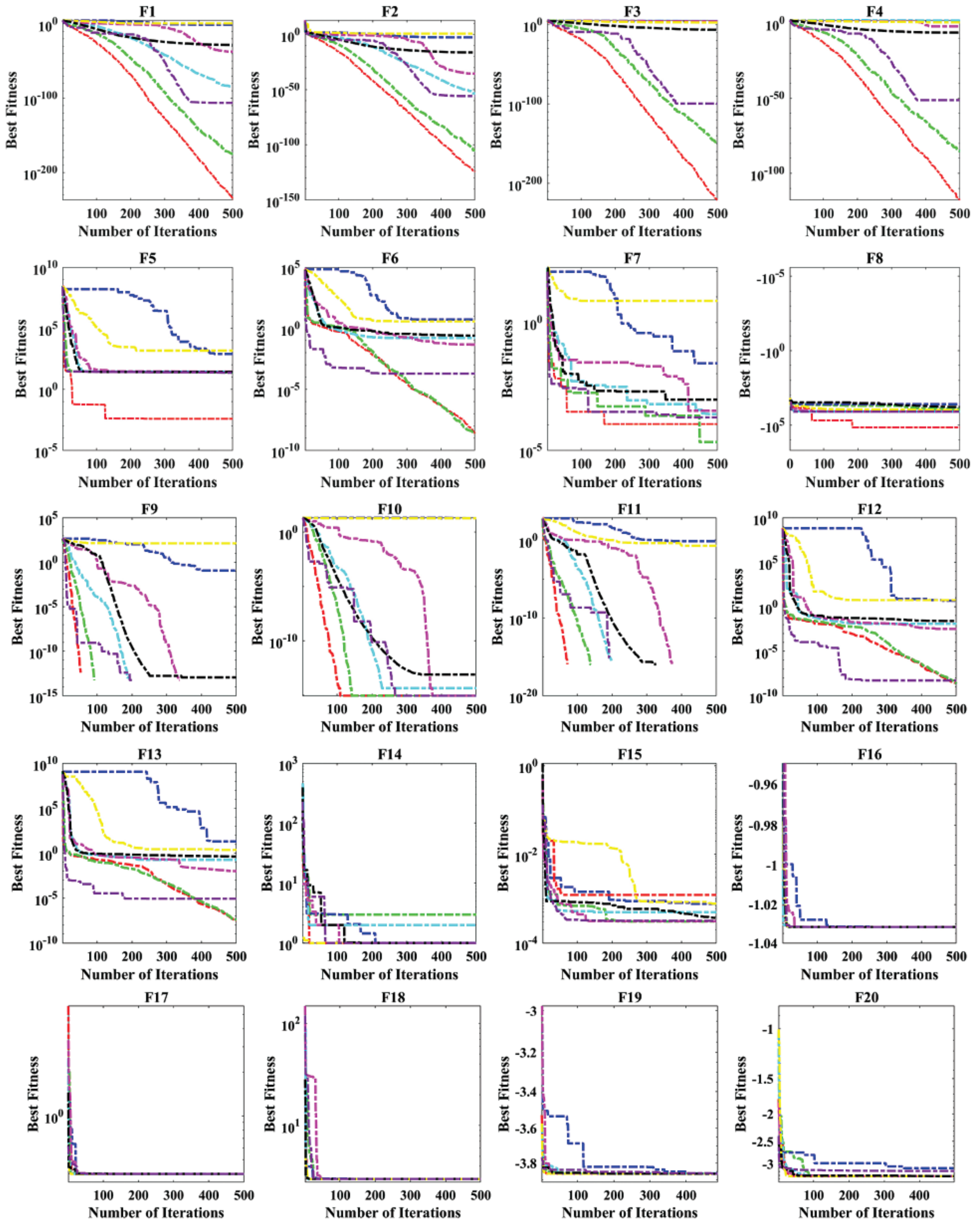
The primary objective of the problem depicted in Fig. 5 is to minimize the cost of the welded beam and find the best cost by considering the constraints. The design variables are length (l), the thickness of the bar (b), the thickness of the weld (h), and height (t). It has constraints, such as beam end deflection (δ), bar buckling load (P_c), shear stress (τ), beam blending stress (θ), and side constraints, and consider $x = [x_1,\ x_2,\ x_3,\ x_4] = [h,\ l,\ t,\ b]$. The variable ranges are $0.1 \leqslant x_1 \leqslant 2$, $0.1 \leqslant x_2 \leqslant 10$, $0.1 \leqslant x_3 \leqslant 10$, and $0.1 \leqslant x_4 \leqslant 2$, and design values are $P = 6000lb, L = 14in., \delta_{max} = 0.25in.$, $E = 30 \times 1^6 psi, G = 12 \times 10^6 psi$, and $\tau_{max} = 13600psi, \sigma_{max} = 30000psi$. The objective function is given in Eq. (16).

$$\text{Minimize, } f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \tag{16}$$

Subjected to constraints:

$$\left.\begin{aligned} &g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leqslant 0 \\ &g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leqslant 0 \\ &g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leqslant 0 \\ &g_4(\vec{x}) = x_1 - x_4 \leqslant 0 \\ &g_5(\vec{x}) = P - P_c(\vec{x}) \leqslant 0 \\ &g_6(\vec{x}) = 0.125 - x_1 \leqslant 0 \\ &g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leqslant 0 \end{aligned}\right\} \tag{17}$$

Tab. 3 lists the results obtained by the IRKO and other selected algorithms, including RKO, GWO, HHO, WOA, IMO, HGSO, and SCA. From Tab. 3, it can be seen that the IRKO performing better than all selected methods and attained the minimum cost. Tab. 3 also lists the statistical data, such as Min, Mean, STD, and RT. Therefore, it is determined that the reliability of the suggested IRKO algorithm is better for the engineering design problem. Fig. 6 depicts the convergence curves and box plots of all algorithms while handling the problem. In addition, FRT values obtained by all algorithms are also listed. The proposed IRKO algorithm stands first in solving the welded beam design problem also.


F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
F16
F17
F18
F19
F20
Best Fitness
Number of Iterations

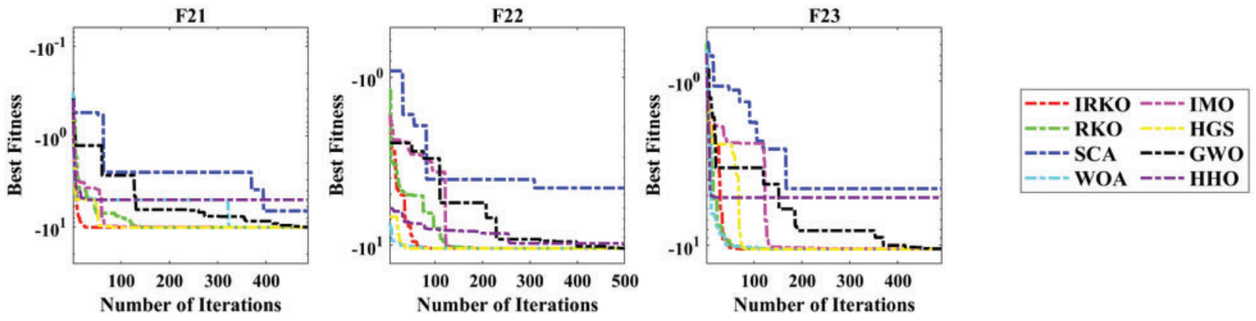


Figure 3: Convergence performance of all selected algorithms on traditional benchmarks

4.2.2 *Tension/Compression Spring Design*

Compression spring design has been considered as yet alternative traditional mechanical engineering problem. This problem is illustrated in Fig. 7, and the primary objective of this problem is to minimize the tension spring weight of the framework. The design variables of the tension/compression spring design problem are active coils (N), wire diameter (d), and mean coil diameter (D), and consider $x = [x_1, x_2, x_3] = [d, D, N]$. The variable ranges are $0.05 \le x_1 \le 2.0$, $0.25 \le x_2 \le 1.3$, and $2.0 \le x_3 \le 15.0$. The objective function is given in Eq. (18).

$$\text{Minimize}\, f(\vec{x}) = (x_3 + 2)x_2x_1^2 \tag{18}$$

Subjected to constraints:

$$\left.\begin{aligned}
g_1(\vec{x}) &= 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0 \\
g_2(\vec{x}) &= \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0 \\
g_3(\vec{x}) &= 1 - \frac{140.45x_1}{x_2^2x_3} \le 0 \\
g_4(\vec{x}) &= \frac{x_1 + x_2}{1.5} - 1 \le 0
\end{aligned}\right\} \tag{19}$$

Tab. 4 lists the results obtained by the IRKO and other selected algorithms, including RKO, GWO, HHO, WOA, IMO, HGSO, and SCA. From Tab. 4, it can be seen that the IRKO performing better than all other selected algorithms and attained the minimum tension spring weight. Tab. 4 also lists the statistical data, such as Min, Mean, STD, and RT. Therefore, it is determined that the reliability of the suggested IRKO algorithm is better for the compression spring design problem. Fig. 8 depicts the convergence curves and box plots of all algorithms while handling the problem. In addition, FRT values obtained by all algorithms are also listed. The proposed IRKO algorithm stands first in solving the tension/compression spring design problem also.

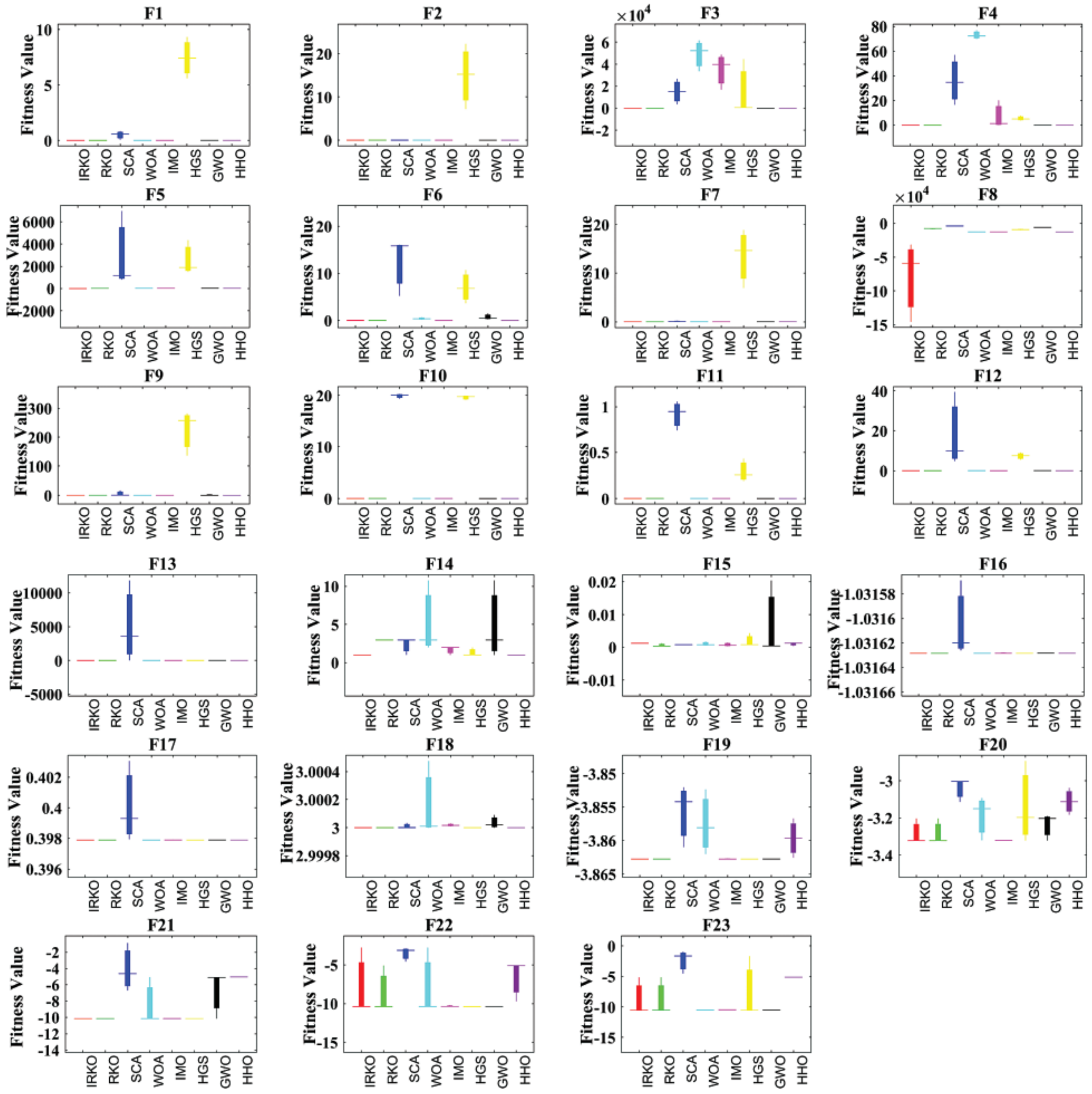


Figure 4: Boxplot analysis of all algorithms on selected benchmark functions

Table 2: FRT values of all selected algorithms

Algorithm/Function	IRKO	RKO	SCA	WOA	IMO	HGS	GWO	HHO
F1	1.000	2.000	5.667	7.667	7.000	5.667	4.000	3.000
F2	1.000	2.000	6.667	8.000	5.667	5.667	4.000	3.000
F3	1.000	3.000	7.333	5.000	6.000	7.667	4.000	2.000
F4	1.333	1.667	7.667	5.333	4.000	7.333	5.667	3.000
F5	2.333	1.000	7.000	5.000	5.000	8.000	4.667	3.000
F6	1.000	5.667	8.000	3.000	4.000	5.333	7.000	2.000
F7	3.000	3.000	6.667	3.000	3.000	8.000	6.333	3.000
F8	2.167	2.167	8.000	4.667	3.833	7.000	6.000	2.167
F9	3.333	3.333	8.000	3.333	4.333	7.000	3.333	3.333
F10	2.000	1.000	7.667	5.000	4.000	7.333	6.000	3.000
F11	2.000	2.333	8.000	5.000	3.667	7.000	6.000	2.000
F12	1.167	6.000	6.667	6.667	4.667	2.500	5.333	3.000
F13	5.667	2.667	4.333	4.667	3.667	5.667	4.000	5.333
F14	1.500	3.000	8.000	5.000	5.667	1.500	6.667	4.667
F15	1.500	3.000	8.000	4.333	6.667	1.500	5.667	5.333
F16	1.333	2.667	6.333	6.333	6.667	2.000	6.667	4.000
F17	1.167	3.000	7.667	6.667	4.333	1.833	4.667	6.667
F18	2.000	2.667	7.333	5.333	3.000	5.000	4.000	6.667
F19	1.000	3.000	7.667	5.000	5.000	2.000	5.000	7.333
F20	1.667	3.667	7.333	5.667	4.667	3.333	3.333	6.333
F21	2.667	3.667	7.667	4.333	4.667	3.333	3.000	6.667
F22	1.000	2.000	5.667	7.667	7.000	5.667	4.000	3.000
F23	1.000	2.000	6.667	8.000	5.667	5.667	4.000	3.000

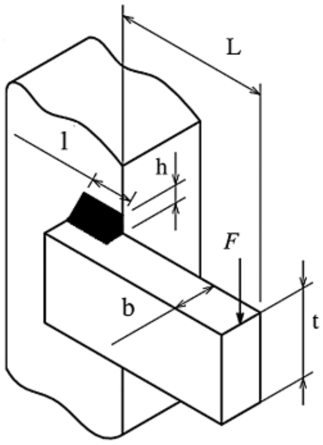
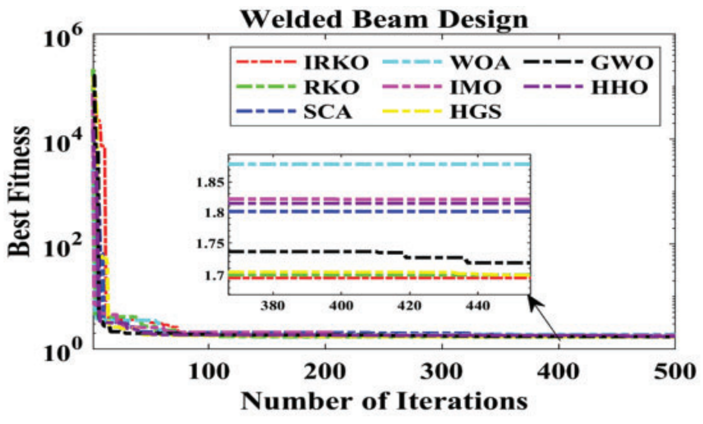


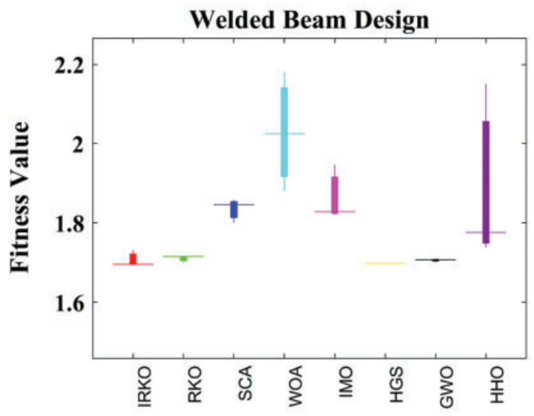
Figure 5: Welded beam design optimization problem

Table 3: Results obtained by all algorithms while solving the welded beam problem

Algorithm	h	l	t	b	Min	Mean	STD	RT	FRT
IRKO	**0.205735**	**3.253031**	**9.036624**	**0.20573**	**1.695**	**1.697**	**0.001**	**0.146**	**1.67**
RKO	0.200881	3.341663	9.036607	0.20573	1.700	1.710	0.009	0.099	3.33
SCA	0.204085	3.574402	8.771277	0.220685	1.801	1.835	0.030	3.391	6.00
WOA	0.170613	4.265315	8.470226	0.234164	1.880	2.029	0.150	6.203	7.67
IMO	0.210796	3.345326	8.462105	0.234613	1.821	1.865	0.070	1.453	6.33
HGSO	0.205291	3.26129	9.035764	0.205771	1.696	1.707	0.021	0.661	2.00
GWO	0.200872	3.347504	9.038295	0.205781	1.701	1.706	0.004	0.151	3.00
HHO	0.200804	3.43318	9.230382	0.204784	1.738	1.888	0.228	0.464	6.00



(a)



(b)

Figure 6: Curves of all algorithms for the welded beam design; (a) Convergence curve, (b) Boxplot

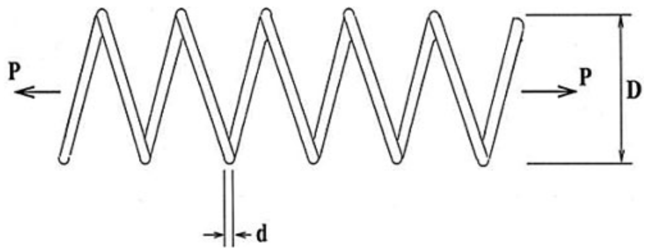


Figure 7: Tension/Compression spring design problem

4.2.3 Pressure Vessel Design

The graphic view of the pressure vessel design framework is shown in Fig. 9. The pressure vessel has hemispherical heads and capped ends. The key objective is to minimize the construction cost. It has four constraints and four parameters (i.e., the length of the cylindrical section (L), the thickness of the head (T_h), the inner radius (R), and the thickness of the shell (T_s)), and consider

$x = [x_1 x_2 x_3 x_4] = [T_s T_h R L]$. The variable ranges are $0 \le x_i \le 99$, $i = 1, 2$ and $10 \le x_i \le 200, i = 3, 4$. The objective function is given in Eq. (20).

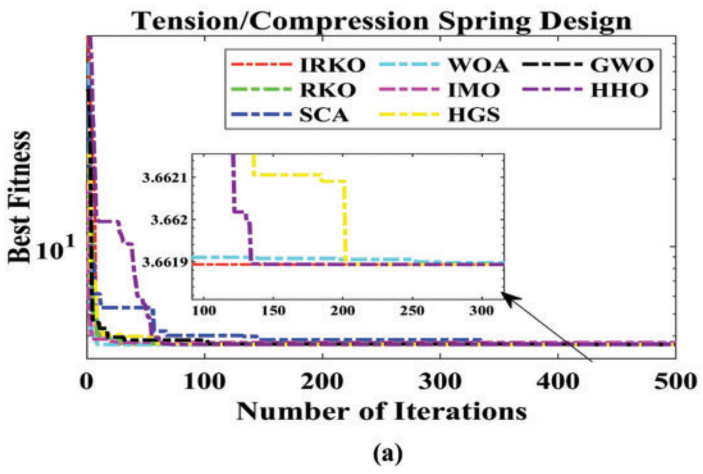
$$\text{Minimize}\, f(x) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \tag{20}$$

Subjected to constraints:

$$\left.\begin{aligned} g_1(x) &= -x_1 + 0.0193x \\ g_2(x) &= -x_2 + 0.00954 x_3 \le 0 \\ g_3(x) &= -\pi x_3^2 x_4 - (4/3)\pi x_3^3 + 1,296,000 \le 0 \\ g_4(x) &= x_4 - 240 \le 0 \end{aligned}\right\} \tag{21}$$

Table 4: Results obtained by all algorithms while solving the tension/compression spring problem

Algorithm	d	D	P	Min	Mean	STD	RT	FRT
IRKO	**0.139149732**	**1.3**	**11.89243063**	**3.662**	**3.662**	**0.000**	**0.057**	**1.00**
RKO	0.13914971	1.3	11.89242563	3.662	3.662	0.000	0.047	2.33
SCA	0.137440614	1.265	12.33009784	3.682	3.696	0.023	1.297	6.67
WOA	0.13914159	1.3	11.88950847	3.662	3.685	0.033	4.557	5.67
IMO	0.133756325	1.149	14.64262775	3.731	3.734	0.002	0.583	8.00
HGSO	0.13914701	1.3	11.89186832	3.662	3.662	0.000	0.214	3.67
GWO	0.139124009	1.3	11.88926717	3.662	3.662	0.000	0.052	5.00
HHO	0.139149956	1.3	11.89251101	3.662	3.665	0.006	0.135	3.67



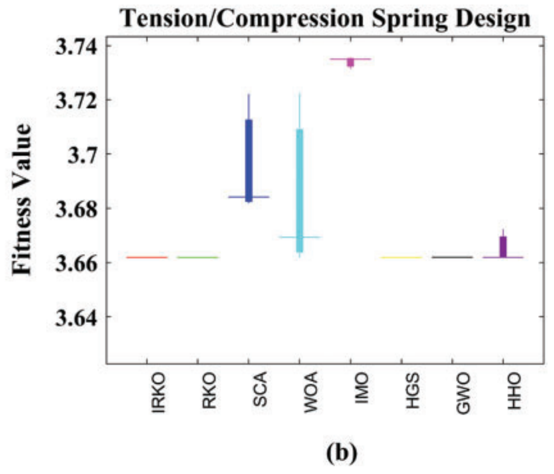


Figure 8: Curves of all algorithms for the compression spring; (a) Convergence curve, (b) Boxplot

Tab. 5 lists the results found by the IRKO and several selected algorithms, including RKO, GWO, HHO, WOA, IMO, HGSO, and SCA. From Tab. 5, it can be seen that the IRKO performing better than all other selected algorithms and attained the minimum construction cost. Tab. 5 also lists the statistical data, such as Min, Mean, STD, and RT. Therefore, it is concluded that the reliability of the proposed IRKO algorithm is better for the pressure vessel design problem. Fig. 10 depicts the convergence curves and box plots of all algorithms while handling the problem. In addition, FRT values obtained by all algorithms are also listed. The proposed IRKO algorithm stands first in solving the pressure vessel design problem.

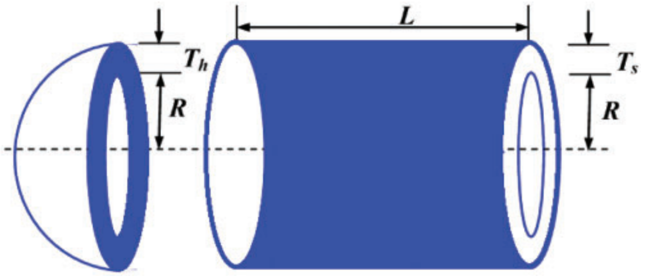


Figure 9: Pressure vessel design problem

Table 5: Results obtained by all algorithms while solving the pressure vessel problem

Algorithm	T_s	T_h	R	L	Min	Mean	STD	RT	FRT
IRKO	1.093571	0	65.22523	10	2302.55	2302.55	0.00	0.05	1.00
RKO	1.09259	1.04E – 13	65.22524	10	2302.56	2302.57	0.02	0.04	2.67
SCA	1.13138	0	65.15833	12.33304	2429.79	4856.43	2101.55	0.88	6.33
WOA	0.906674	0	57.89208	45.89891	3006.85	5122.66	1836.32	4.51	7.33
IMO	0.764525	0	51.81955	84.5345	3396.84	5173.69	1538.81	0.58	6.33
HGSO	1.09356	2.14E – 21	65.22523	10	2302.55	2683.27	659.43	0.23	3.67
GWO	1.086553	0	65.22587	10	2303.17	2309.16	10.02	0.05	3.67
HHO	0.943091	0	59.14706	39.05765	2903.52	3084.58	231.81	0.13	5.00

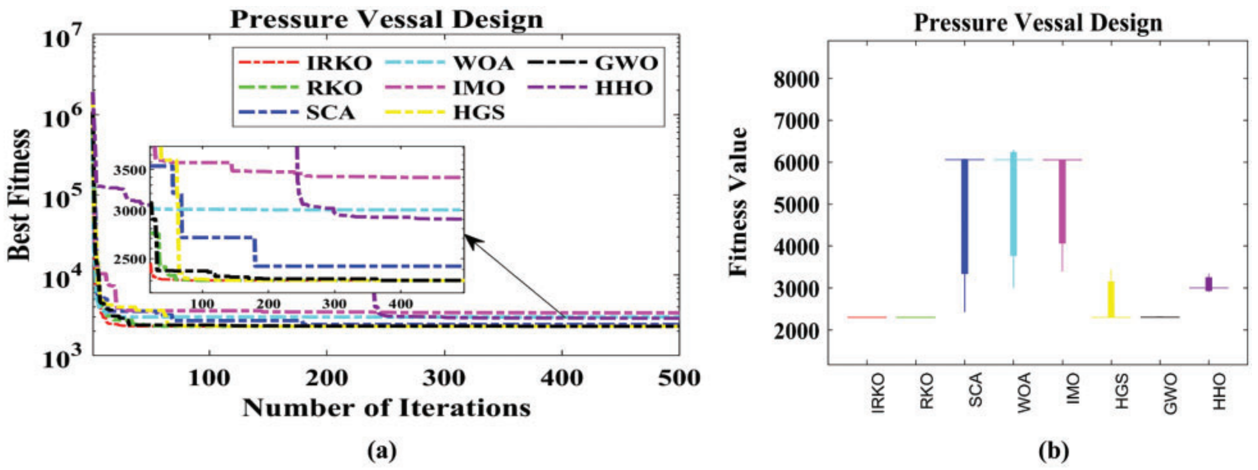


Figure 10: Curves of all algorithms for the pressure vessel problem; (a) Convergence curve, (b) Boxplot

5 Conclusion

In this paper, an enhanced variant of the RKO algorithm called IRKO algorithm is proposed. A local escaping operator concept is employed to improve the characteristics, i.e., exploration and exploitation capabilities of the original RKO algorithm. A 23 classical test suite and three engineering design problems—welded beam, tension/compression spring, and pressure vessel design problems are used to assess the performance of the IRKO algorithm. According to benchmark tests' statistical findings, the IRKO produced outcomes that were either superior or relatively close to other selected algorithms. Furthermore, it may assess the proposed IRKO's suitability to real-world applications based on actual investigations of design problems.

From the earlier analysis, the IRKO might enable a broad range of future tasks. This includes applying the IRKO algorithm in numerous applications, such as image processing, feature selection, PV parameter estimation, power systems, power electronics, smart grid, big data applications, data mining applications, signal denoising, wireless sensor networks, artificial intelligence, machine learning, and other benchmark functions. Also applicable to situations depending on binary-, multi-, and many-objective optimizations.

Acknowledgement: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University, Saudi Arabia, for funding this work through the Research Group Program under Grant No: RGP. 2/108/42.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. D. Schaffer, *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*. Nashville, USA: Vanderbilt University, pp. 115–152, 1985.

[2] W. K. Wong and C. I. Ming, "A review on metaheuristic algorithms: Trends, benchmarking and applications," in *Proc. of Int. Conf. on Smart Computing and Communications*, Malaysia, pp. 1–5, 2019.

[3] M. Premkumar, P. Jangir and R. Sowmya, "MOGBO: A new multiobjective gradient-based optimizer for real-world structural optimization problems," *Knowledge-Based Systems*, vol. 218, pp. 106856, 2021.

[4] M. Premkumar, R. Sowmya, J. Pradeep, K. S. Nisar and M. Aldhaifallah, "A new metaheuristic optimization algorithms for brushless direct current wheel motor design problem," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 2227–2242, 2021.

[5] M. Premkumar, P. Jangir, B. Santhosh Kumar, R. Sowmya, H. H. Alhelou *et al.,* "A new arithmetic optimization algorithm for solving real-world multiobjective CEC-2021 constrained optimization problems: Diversity analysis and validations," *IEEE Access*, vol. 9, pp. 84263–84295, 2021.

[6] I. Ahmadianfar, O. Bozorg-haddad and X. Chu, "Gradient-based optimizer: A new metaheuristic optimization algorithm," *Information Sciences*, vol. 540, pp. 131–159, 2020.

[7] S. Kumar, P. Jangir, G. G. Tejani, M. Premkumar and H. H. Alhelou, "MOPGO: A new physics-based multi-objective plasma generation optimizer for solving structural optimization problems," *IEEE Access*, vol. 9, pp. 84982–85016, 2021.

[8] B. A. Khateeb, K. Ahmed, M. Mahmod and D. N. Le, "Rock hyraxes swarm optimization: A new nature-inspired metaheuristic optimization algorithm," *Computers, Materials & Continua*, vol. 68, no. 1, pp. 643–654, 2021.

[9] R. Fessi1, H. Y. Rezk and S. Bouallègue, “Grey wolf optimization-based tuning of terminal sliding mode controllers for a quadrotor,” *Computers, Materials & Continua*, vol. 68, no. 2, pp. 2265–2282, 2021.

[10] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[11] A. Iman, A. H. Ali, A. H. Gandomi, X. Chu and H. Chen, “RUN beyond the metaphor: An efficient optimization algorithm based on runge kutta method,” *Expert Systems with Applications*, vol. 181, pp. 115079, 2021.

[12] M. Premkumar, P. Jangir, C. Ramakrishnan, G. Nalinipriya, H. H. Alhelou *et al.,* “Identification of solar photovoltaic model parameters using an improved gradient-based optimization algorithm with chaotic drifts,” *IEEE Access*, vol. 9, pp. 62347–62379, 2021.

[13] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proc. of the Sixth Int. Symp. on Micro Machine and Human Science*, Japan, pp. 39–43, 1995.

[14] M. Dorigo, V. Maniezzo and A. Colorni, “Ant system: Optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics: Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[15] W. C. Wang, L. Xu, K. W. Chau and D. M. Xu, “Yin-yang firefly algorithm based on dimensionally Cauchy mutation,” *Expert Systems with Applications*, vol. 150, pp. 113216, 2020.

[16] D. Oliva, E. Cuevas and G. Pajares, “Parameter identification of solar cells using artificial bee colony optimization,” *Energy*, vol. 72, pp. 93–102, 2014.

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

[18] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja *et al.,* “Harris hawks optimization: Algorithm and applications,” *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[19] N. Rana, M. S. A. Latiff, S. M. Abdulhamid and H. Chiroma, “Whale optimization algorithm: A systematic review of contemporary applications,” *Neural Computing and Applications*, vol. 32, pp. 16245–16277, 2020.

[20] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-Qaness *et al.,* “Aquila optimizer: A novel meta-heuristic optimization algorithm,” *Computers & Industrial Engineering*, vol. 157, pp. 107250, 2021.

[21] M. Premkumar, R. Sowmya, P. Jangir, H. H. Alhelou, A. A. Heidari *et al.,* “MOSMA: Multi-objective slime mould algorithm based on elitist non-dominated sorting,” *IEEE Access*, vol. 9, pp. 3229–3248, 2021.

[22] A. Faramarzi, M. Heidarinejad, S. Mirjalili and A. H. Gandomi, “Marine predators algorithm: A nature-inspired metaheuristic,” *Expert Systems with Applications*, vol. 152, pp. 113377, 2020.

[23] A. Faramarzi, M. Heidarinejad, B. Stephens and S. Mirjalili, “Equilibrium optimizer: A novel optimization algorithm,” *Knowledge-Based Systems*, vol. 191, pp. 105190, 2020.

[24] M. Premkumar, C. Kumar, R. Sowmya and J. Pradeep, “A novel salp swarm assisted hybrid maximum power point tracking algorithm for the solar PV power generation systems,” *Automatika*, vol. 62, no. 1, pp. 1–20, 2021.

[25] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Machine Learning*, vol. 3, no. 2, pp. 95–99, 1988.

[26] F. Huang, L. Wang and Q. He, “An effective co-evolutionary differential evolution for constrained optimization,” *Applied Mathematics and Computation*, vol. 186, no. 1, pp. 340–356, 2007.

[27] Q. Niu, L. Zhang and K. Li, “A biogeography-based optimization algorithm with mutation strategies for model parameter estimation of solar and fuel cells,” *Energy Conversion and Management*, vol. 86, pp. 1173–1185, 2014.

[28] M. H. Qais, H. M. Hasanien and S. Alghuwainem, “Identification of parameters for three-diode photovoltaic model using analytical and sunflower optimization algorithm,” *Applied Energy*, vol. 250, pp. 109–117, 2019.

[29] D. Tang, S. Dong, Y. Jiang, H. Li and Y. Huang, "ITGO: Invasive tumor growth optimization algorithm," *Applied Soft Computing*, vol. 36, pp. 670–698, 2015.

[30] R. V. Rao, V. J. Savsani and J. Balic, "Teaching-learning-based optimization algorithm for unconstrained and constrained optimization problems," *Engineering Optimization*, vol. 44, no. 12, pp. 1447–1462, 2012.

[31] Y. Yang, H. Chen, A. A. Heidari and A. H. Gandomi, "Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Systems with Applications*, vol. 177, pp. 114864, 2021.

[32] M. Premkumar, R. Sowmya, P. Jangir and J. S. V. Siva Kumar, "A new and reliable objective functions for extracting the unknown parameters of solar photovoltaic cell using political optimizer algorithm," in *Proc. of Int. Conf. on Data Analytics for Business and Industry*, Bahrain, pp. 1–6, 2020.

[33] A. Banerjee, V. Mukherjee and S. P. Ghoshal, "An opposition-based harmony search algorithm for engineering optimization problems," *Ain Shams Engineering Journal*, vol. 5, no. 1, pp. 85–101, 2014.

[34] M. Premkumar, P. Jangir, R. Sowmya, R. M. Elavarasan and B. S. Kumar, "Enhanced chaotic JAYA algorithm for parameter estimation of photovoltaic cell/modules," *ISA Transactions*, Article in press, pp. 1–28, 2021.

[35] M. Premkumar, T. S. Babu, S. Umashankar and R. Sowmya, "A new metaphor-less algorithms for the photovoltaic cell parameter estimation," *Optik*, vol. 208, pp. 164559, 2020.

[36] E. Rashedi, H. N. Pour and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

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

[38] B. Javidy, A. Hatamlou and S. Mirjalili, "Ions motion algorithm for solving optimization problems," *Applied Soft Computing*, vol. 32, pp. 72–79, 2015.

[39] M. Premkumar, "Classical benchmark test functions," 2021. [Online]. Available: https://premkumarmanoharan.wixsite.com/mysite/downloads [Accessed: April 25, 2021].