Expert Systems With Applications 181 (2021) 115079

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method

Iman Ahmadianfar [a,*], Ali Asghar Heidari [b,c,1], Amir H. Gandomi [d], Xuefeng Chu [e], Huiling Chen [f,*]

[a] Department of Civil Engineering, Behbahan Khatam Alanbia University of Technology, Behbahan, Iran
[b] School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran 1439957131, Iran
[c] Department of Computer Science, School of Computing, National University of Singapore, Singapore 117417, Singapore
[d] University of Technology Sydney, Ultimo, NSW 2007, Australia
[e] Department of Civil & Environmental Engineering, North Dakota State University, Department 2470, Fargo, ND, USA
[f] College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou, Zhejiang 325035, China

ARTICLE INFO

Keywords:
Genetic algorithms
Evolutionary algorithm
Runge Kutta optimization
Optimization
Swarm intelligence
Performance

ABSTRACT

The optimization field suffers from the metaphor-based "*pseudo-novel*" or "*fancy*" optimizers. Most of these cliché methods mimic animals' searching trends and possess a small contribution to the optimization process itself. Most of these cliché methods suffer from the locally efficient performance, biased verification methods on easy problems, and high similarity between their components' interactions. This study attempts to go beyond the traps of metaphors and introduce a novel metaphor-free population-based optimization method based on the mathematical foundations and ideas of the Runge Kutta (RK) method widely well-known in mathematics. The proposed RUNge Kutta optimizer (RUN) was developed to deal with various types of optimization problems in the future. The RUN utilizes the logic of slope variations computed by the RK method as a promising and logical searching mechanism for global optimization. This search mechanism benefits from two active exploration and exploitation phases for exploring the promising regions in the feature space and constructive movement toward the global best solution. Furthermore, an enhanced solution quality (ESQ) mechanism is employed to avoid the local optimal solutions and increase convergence speed. The RUN algorithm's efficiency was evaluated by comparing with other metaheuristic algorithms in 50 mathematical test functions and four real-world engineering problems. The RUN provided very promising and competitive results, showing superior exploration and exploitation tendencies, fast convergence rate, and local optima avoidance. In optimizing the constrained engineering problems, the metaphor-free RUN demonstrated its suitable performance as well. The authors invite the community for extensive evaluations of this deep-rooted optimizer as a promising tool for real-world optimization. The source codes, supplementary materials, and guidance for the developed method will be publicly available at different hubs at http://imanahmadianfar.com and http://aliasgharheidari.com/RUN.html.

1. Introduction

Most real-world problems are complicated and present difficulties in being optimized (Xue et al., 2021). These problems are often characterized by nonlinearity, multimodality, non-differentiability, and high dimensionality (Niu et al., 2020; Zhang et al., 2019, 2020). Because of these properties, the conventional gradient-based optimization methods, such as quasi-Newton, conjugate gradient, and sequential quadratic programming methods, are unable to optimize such problems virtually (Nocedal & Wright, 2006; Wu, 2016). Therefore, existing literature suggests that other optimization techniques need to be developed for more efficient and effective optimization. An optimization problem can be in many-objective forms (Cao et al., 2020). One another problem can be multi-objective (Cao, Zhao, Yang, et al., 2019), memetic, fuzzy, robust, large scale (Cao et al., 2020), and single-objective. Real-world problems are faced every day (Qu, Xu, Zhao, & Zhang,

* Corresponding authors.
E-mail addresses: i.ahmadianfar@bkatu.ac.ir (I. Ahmadianfar), as_heidari@ut.ac.ir, aliasgha@comp.nus.edu.sg, t0917038@u.nus.edu (A.A. Heidari), gandomi@uts.edu.au (A.H. Gandomi), xuefeng.chu@ndsu.edu (X. Chu), chenhuiling_jsj@wzu.edu.cn, chenhuiling.jlu@gmail.com (H. Chen).
[1] https://aliasgharheidari.com

https://doi.org/10.1016/j.eswa.2021.115079
Received 29 April 2020; Received in revised form 22 January 2021; Accepted 17 April 2021
Available online 21 April 2021

2021), and we need to develop solvers for deep learning applications (Li et al., 2019; Shi, Lu, & Zhang, 2019; Zhang et al., 2021), decision-making procedures (Wu et al., 2020; Liu et al., 2021), image improvement optimization (Zenggang et al., 2021), imaging array (Liu et al., 2020), water-energy optimization (Chen, He, Guan, Lu, & Li, 2017), training systems and methods in artificial neural networks (Chen, Zheng, Li, & Huang, 2021; Mousavi, Zhang, Masri, & Gholipour, 2020), and optimization of the parameters (Zhang, Ou, & Zhang, 2006). Numerous metaheuristic optimization algorithms (MOAs) have been developed and widely employed as suitable alternative optimizers to solve various problems due to their flexibility and straightforward implementation procedure (Ma et al., 2021). MOAs can be categorized into three groups (Kaveh & Bakhshpoori, 2016): evolutionary algorithms (EAs), physics-based algorithms (PBAs), and swarm-based algorithms (SBAs). Nevertheless, they present some drawbacks, including high sensitivity and their control parameter settings. Also, they do not always converge toward the globally optimal solution (Wu, Pedrycz, Suganthan, & Mallipeddi, 2015). As they utilize some randomly generated components within the procedure, an appropriate balance between exploration and exploitation cannot be ensured. This limit is one of the fundamental challenges within all kinds of methods in this area.

The methods under the class of EAs are based on the principles of evolution in nature, such as selection, recombination, and mutation. The genetic algorithm (GA), another widely-used EA, was inspired by Darwin's theory of evolution (Holland, 1975). Other EAs include genetic programming (GP) (Koza, 1994), differential evolution (DE) (Storn & Price, 1995), and evolution strategy (Beyer & Schwefel, 2002). The methods in this category have the deepest roots in their foundation theory compared to other approaches, as Darwin's theory reshaped our vision of the tree of life. Later, the development of physics-based algorithms (PBAs) emerged as a trend in the field inspired by physics laws governing the surrounding world. For instance, among these emerging PBA algorithms, simulated annealing (SA) is the most popular one (Kirkpatrick, Gelatt, & Vecchi, 1983). Other PBAs include gravitational search algorithm (GSA) (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009), central force optimization (Formato, 2007), vortex search algorithm (VSA) (Doğan & Ölmez, 2015), and gradient-based optimizer (GBO) (Ahmadianfar, Bozorg-Haddad, & Chu, 2020). Researchers tried to simulate organisms' cooperative behaviors in flocks after years passed, which are natural or artificial (Baykasoğlu & Ozsoydan, 2017). For example, the main inspiration in particle swarm optimization (PSO) (Eberhart & Kennedy, 1995) is a flock of birds' social behaviors. Other SBA examples include the Bat algorithm (BA) (Yang, 2010b), ant colony optimization (ACO) (Dorigo & Di Caro, 1999), artificial bee colony (ABC) (Karaboga & Basturk, 2007), firefly algorithm (FA) (Yang, 2010a), hunger games search (HGS)[2] (Yang, Chen, Heidari, & Gandomi, 2021), slime mould algorithm (SMA)[3] (Li, Chen, Wang, Heidari, & Mirjalili, 2020), and Harris hawks optimization (HHO)[4] (Heidari et al., 2019).

On the other hand, evolution served as the core idea of swarm-based methods that evolved the algorithms themselves. Two large influences of these evolving and algorithms included the searching trend for an "unused" biologic source of inspiration and utilizing it as a dress for a set of equations. Despite the weaknesses, metaphors, and structural differences of various optimization algorithms, they all employ two typical phases, exploration and exploitation, to search the solution space regions (Salcedo-Sanz, 2016). Exploring is an optimization algorithm's ability to sincerely search the entire solution space and explore the promising areas. At the same time, exploitation is the capability of an optimization algorithm to search around near-optimal solutions. Generally, the exploration phase of an optimizer should randomly produce solutions in various regions of the solution space during early iterations of the optimization process (Heidari, Aljarah, et al., 2019). In contrast, the exploitation phase of an optimization algorithm should create a robust local search. Thus, a well-designed idea should be able of creating a suitable balance between the exploration and exploitation phases.

Generally, creating an appropriate trade-off between exploration and exploitation is an essential task for any optimization algorithm (Ahmadianfar, Kheyrandish, Jamei, & Gharabaghi, 2020). In this regard, many researchers have attempted to improve the optimizers' performance by selecting appropriate control parameters or hybridizing with other optimizers (Ahmadianfar, Khajeh, Asghari-Pari, & Chu, 2019; Zhang, Zhou, & Luo, 2018; Luo et al., 2017). Nevertheless, creating a robust algorithm that can balance exploration and exploitation is a complex and challenging issue. Moreover, as there are many real-world problems, more accurate and more consistent optimizers are needed. To fill such a gap, a well-designed population-based optimization procedure is proposed in this research. The proposed algorithm, Runge Kutta optimizer (RUN), was designed according to the foundations of the Runge Kutta method[5] (Kutta, 1901; Runge, 1895). RUN uses a specific slope calculation concept based on the Runge Kutta method as an effective search engine for global optimization. The proposed algorithm consists of two main parts: a search mechanism based on the Runge Kutta method and an enhanced solution quality (ESQ) mechanism to increase solutions' quality. RUN's performance was evaluated by using 50 mathematical test functions, and the results were compared with those of other state-of-the-art optimizers. Furthermore, the proposed RUN was employed to solve four engineering design problems to test its ability and efficiency in solving a number of real-world optimization problems.

This paper is organized as follows. Section 2 presents a summarized review of the Runge Kutta method. Section 3 provides the mathematical formulation and optimization procedures of the RUN algorithm. Section 4 evaluates the efficiency of the RUN to optimize different benchmark test functions. Section 5 assesses the ability of the proposed RUN in solving engineering design problems. Section 6 presents the main conclusions and some useful suggestions for future studies.

2. Related works

Generally, stochastic optimization algorithms can be categorized into two classes: single-based and population-based algorithms. Some algorithms begin the optimization procedure with a single random position in the first class and updates it during each iteration (Mirjalili, 2015a). Simulated annealing (SA) (Kirkpatrick et al., 1983), tabu search (TS) (Glover & Laguna, 1998), and hill-climbing (HC) (Tsamardinos, Brown, & Aliferis, 2006) belong to this class. The primary benefits of the single-based optimizers include easy implementation and a low number of function evaluations, while their main drawback is the high possibility of getting caught up in local solutions. In contrast, the population-based methods start the optimization procedure with a set of random solutions and update their positions at each iteration. The well-known GA, PSO, DE, ACO, ABC, and biogeography-based optimization (BBO) (Simon, 2008) belong to this category. Population-based optimization algorithms also have a relatively acceptable ability to avoid the local optimal solutions because they employ a set of solutions at each iteration instead of only evolving on a single agent.

Accordingly, the population-based algorithms can handle the sceneries of feature space and increase the convergence speed. Furthermore, they can share information between solutions, making a more convenient search in complex and challenging feature spaces (Yang, Chen, Heidari, & Gandomi, 2021). Notwithstanding these advantages, these optimizers require many function evaluations during the optimization

[2] https://aliasgharheidari.com/HGS.html
[3] https://aliasgharheidari.com/SMA.html
[4] https://aliasgharheidari.com/HHO.html

[5] For a better presentation of the term, we use the term Runge Kutta in this paper

process and a relatively complicated/difficult implementation. Another unavoidable issue is that these methods apply a random-based vision for understanding the problem's topographies, making them unbalanced, inaccurate, or unsuccessful in finding any best solution. However, sometimes a locally-accurate solution can satisfy the practitioners and requirements of real-world problems. Many studies indicate that the population-based optimizers are regarded as more reliable and accurate than the single-based algorithms because of the advantages mentioned above. Their applications in a broad range of fields have demonstrated their worthiness and high capability. Generally, these optimization algorithms have been largely inspired physics's laws, social behaviors of creatures, and natural phenomena.

Of pertinent mention, a study by Sörensen on the low-quality contributions in the optimization methods opened the eyes of many researchers (Sörensen, 2015). As per this research, shallow mathematic models supplied with metaphor-based outfits must be avoided to make improvements in the field (Lones, 2020). These metaphors are often perplexing and irrelevant to experts, decision-makers, algorithm designers, and those who utilize these methods for real-world cases. It has also been discovered that some methods, such as popular harmony search, are not original, in which the core mathematic models are a version of $(\mu + 1)$-evolutionary search (Saka, Hasançebi, & Geem, 2016). Regardless of these shortcomings, optimization algorithms consist of exploration and exploitation phases, as previously mentioned. Since establishing a reasonable balance between these two phases is a challenge for any optimization technique, designing a powerful and accurate optimization algorithm to achieve this goal is necessary. Hence, a novel population-based metaheuristic optimization algorithm based on the Runge Kutta method was developed in this study. The following two sections focus on the formulation of this new RUN algorithm.

3. Overview of Runge Kutta method in differential equations

The Runge Kutta method (RKM) is broadly used to solve ordinary differential equations (Kutta, 1901; Runge, 1895). RKM can be applied to create a high-precision numerical method by using functions without requiring their high-order derivatives (Zheng & Zhang, 2017). The primary formulation of the RKM is described as follows.

Consider the following first-order ordinary differential equation for an initial value problem:

$$\frac{dy}{dt} = f(x,y), y(x_0) = y_0 \tag{1}$$

In RKM, the main idea is to define $f(x,y)$ as the slope (S) of the best straight line fitted to the graph at the point (x,y). Using the slope at point (x_0,y_0), another point can be obtained by using the best fitted straight line: $(x_1,y_1) = (x_0 + \Delta x,\ y_0 + S_0\Delta x)$, where $S_0 = f(x_0,y_0)$. Similarly, $(x_2,y_2) = (x_1 + \Delta x,\ y_1 + S_1\Delta x)$. This process can be repeated m times, which yields an approximate solution in the range of $[x_0,\ x_0 + m\Delta x]$.

The derivation of RKM is based on the Taylor series, which is given by:

$$y(x+\Delta x) = y(x) + y'(x)\Delta x + y''(x)\frac{(\Delta x)^2}{2!} + \cdots \tag{2}$$

By dropping the higher-order terms, the following approximate equation can be obtained.

$$y(x+\Delta x) \approx y(x) + y'(x)\Delta x \tag{3}$$

According to Eq. (3), the formula for the first-order Runge Kutta method (or Euler method) can be expressed as:

$$y(x+\Delta x) = y(x) + k_1\Delta x \tag{4}$$

where $k_1 = y'(x) = f(x,y)$; and $\Delta x = x_{n+1}\text{-}x_n$.

The first-order derivative ($y'(x)$) can be approximated by using the following central differencing formula (Patil & Verma, 2006):

$$y'(x) = \frac{y(x+\Delta x) - y(x-\Delta x)}{2\Delta x} \tag{5}$$

Thus, the rule in Eq. (4) can be rewritten as:

$$y(x+\Delta x) = y(x) + \frac{y(x+\Delta x) - y(x-\Delta x)}{2} \tag{6}$$

In this study, the fourth-order Runge Kutta (RK4) (England, 1969) derived from Eq. (2) was used to develop the proposed optimization method. The formula for the RK4 method, which is based on the weighted average of four increments (as shown in Fig. 1), can be expressed as:

$$y(x+\Delta x) = y(x) + \frac{1}{6}(k_1 + 2\times k_2 + 2\times k_3 + k_4)\Delta x \tag{7}$$

in which the four weighted factors (k_1, k_2, k_3, and k_4) are respectively given by:

$$\begin{aligned} k_1 &= y'(x) = f(x,y) \\ k_2 &= f\left(x + \frac{\Delta x}{2},\ y + \frac{\Delta x}{2}\times k_1\right) \\ k_3 &= f\left(x + \frac{\Delta x}{2},\ y + \frac{\Delta x}{2}\times k_2\right) \\ k_4 &= f(x + \Delta x,\ y + \Delta x \times k_3) \end{aligned} \tag{8}$$

where $k_1$is the first increment and determines the slope at the beginning of the interval $[x,\ x + \Delta x]$ using y. $k_2$is the second increment and specifies based on the slope at the midpoint, using y and k_1; $k_3$is the third increment and defines regarding the slope at the midpoint, using y and k_2; and $k_4$is the fourth increment and is determined based on the slope at the end of the interval, using y and k_3. According to RK4, the next value $y(x+\Delta x)$ is specified by the current value $y(x)$ plus the weighted average of four increments.

4. Introducing the Runge Kutta optimizer (RUN)

In this study, a new swarm-based model with stochastic components is developed for optimization purposes. This model eliminates the cliché inspiration attachment with itself the proposed RUN method is represented by using a metaphor-free language with emphasis on the mathematical core as some sets of activated rules at the proper time. Using metaphors in a population-based model is rejected since the only benefit of such a way is to hide the real nature of the equations utilized within

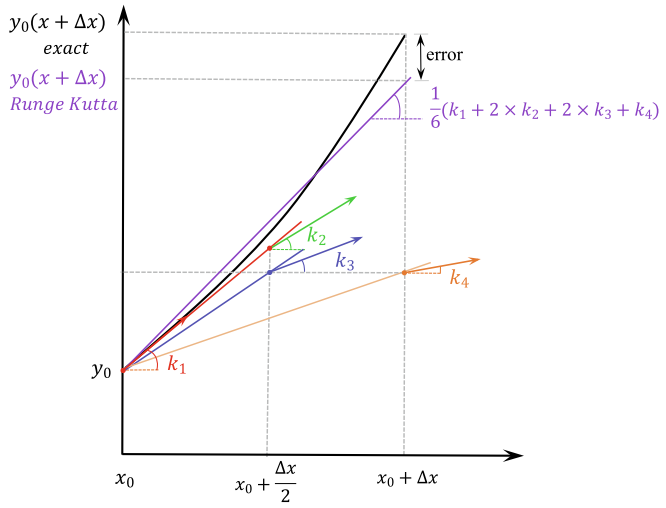

Fig. 1. Slopes utilized in the RK method.

the optimizers. Therefore, RUN accounts for the main logic of the RK technique and the population-based evolution of a crowd of agents. In fact, the RK uses a specific formulation (i.e., RK4 method) to calculate the slope and solve the ordinary differential equations (Kutta, 1901; Runge, 1895). RUN's main idea is based on the concept of the proposed calculated slope in the RK method. The RUN uses the calculated slope as a searching logic to explore the promising area in the search space and build a set of rules for the evolution of a population set according to the swarm-based optimization algorithm's logic. The mathematical formulation of RUN is detailed in the following subsections.

4.1. Initialization step

In this step, the logic is to set an initial swarm to be evolved within the allowed number of iterations. In RUN, N positions are randomly generated for a population with a size of N. Each member of the population, x_n ($n = 1, 2, \ldots, N$), is a solution with a dimension of D for an optimization problem. In general, the initial positions are randomly created by the following idea:

$$x_{n,l} = L_l + rand.(U_l - L_l) \tag{9}$$

where L_l and U_l are the lower and upper bounds of the lth variable of the problem ($l = 1,2,\ldots,D$), and *rand* is a random number in the range of [0, 1]. This rule simply generates some solutions within limits.

4.2. Root of search mechanism

The power of any optimizer is dependent on its iterative cores for generating the exploration and exploitation patterns. In the exploration core, an optimization algorithm uses a set of random solutions with a high randomness rate to explore the promising areas of the feasible space. Small and gradual variations in the exploitation of core solutions and random behaviors are remarkably lower than those in the exploration mechanism (Hu et al., 2021). In this study, RUN's leading search mechanism is based on the RK method to search the decision space using a set of random solutions and implement a proper global and local search.

The RK4 method was employed to determine the search mechanism in the proposed RUN. The first-order derivative was utilized to define the coefficient k_1, which is calculated by Eq. (5). Moreover, the proposed optimization algorithm uses position x_n instead of its fitness ($y(x_n)$), because applying the objective function of a position needs considerable time in computing. According to Eq. (5), $x_n + \Delta x$ and $x_n - \Delta x$ are two neighboring positions of x_n. By considering $y(x)$ as a minimization problem, positions $x_n - \Delta x$ and $x_n + \Delta x$ have best and worst positions, respectively. Therefore, to create a population-based algorithm, position $x_n - \Delta x$ is equal to x_b (i.e., x_b is the best position around x_n), while the position $x_n + \Delta x$ is equal to x_w (i.e., x_w is the worst position around x_k). Therefore, k_1 is defined as:

$$k_1 = \frac{x_w - x_b}{2\Delta x} \tag{10}$$

where x_w and x_b are the worst and best solutions obtained at each iteration, which are determined based on the three random solutions selected from the members of the population (x_{r1}, x_{r2}, x_{r3}), and $r1 \neq r2 \neq r3 \neq n$.

In order to enhance the exploration search and create a randomness behavior, Eq. (10) can be rewritten as follows:

$$k_1 = \frac{1}{2\Delta x}(rand \times x_w - u \times x_b) \tag{10-1}$$

$$u = round(1 + rand) \times (1 - rand) \tag{10-2}$$

where *rand* is a random number in the range of [0, 1]. Overall, the best solution (x_b) plays a crucial role in finding promising areas and moving toward the global best solution. Therefore, in this study, a random parameter (u) is used to increase the importance of the best solution (x_b) during the optimization process. In Eq. (10), Δx can be specified by:

$$\Delta x = 2 \times rand \times |Stp| \tag{11-1}$$

$$Stp = rand \times \left(\left(x_b - rand \times x_{avg}\right) + \gamma\right) \tag{11-2}$$

$$\gamma = rand \times (x_n - rand \times (u - l)) \times \exp(-4 \times \frac{i}{Maxi}) \tag{11-3}$$

where Δx is the position increment, which depends on parameter *Stp*. *Stp* is the step size determined by the difference between x_b and x_{avg}. Parameter γ is a scale factor determined by the solution space's size, decreasing exponentially during the optimization process. x_{avg} is the average all solutions at each iteration. Using the random numbers (*rand*) in Eqs. (11–1) to (11–3), the method can produce more diversification trends and find various search space areas.

Accordingly, the three other coefficients (i.e., k_2, k_3, and k_4) can be respectively written as:

$$k_2 = \frac{1}{2\Delta x}(rand.(x_w + rand_1.k_1.\Delta x) - (u.x_b + rand_2.k_1.\Delta x)) \tag{12}$$

$$k_3 = \frac{1}{2\Delta x}(rand.(x_w + rand_1.\left(\frac{1}{2}k_2\right).\Delta x) - (u.x_b + rand_2.\left(\frac{1}{2}k_2\right).\Delta x)) \tag{13}$$

$$k_4 = \frac{1}{2\Delta x}(rand.(x_w + rand_1.k_3.\Delta x) - (u.x_b + rand_2.k_3.\Delta x)) \tag{14}$$

where $rand_1$ and $rand_2$ are two random numbers in the range of [0, 1]. In this study, x_w and x_b are determined by the following:

$$\textbf{\textit{if}} f(x_n) < f(x_{bi}) \tag{15}$$

$$x_b = x_n$$

$$x_w = x_{bi}$$

else

$$x_b = x_{bi}$$

$$x_w = x_n$$

end

where x_{bi} is the best random solution, which is selected from the three random solutions (x_{r1}, x_{r2}, and x_{r3}). According to Eq. (15), if the fitness of the current solution ($f(x_n)$) is better than that of x_{bi}, the best and worst solutions (x_b and x_w) are equal to x_n and x_{bi}, respectively. Otherwise, they are equal to x_{bi} and x_n, respectively. Therefore, the leading search mechanism in RUN can be defined as:

$$SM = \frac{1}{6}(x_{RK})\Delta x \tag{16}$$

in which

$$x_{RK} = k_1 + 2 \times k_2 + 2 \times k_3 + k_4 \tag{16-1}$$

4.3. Updating solutions

The RUN algorithm begins the optimization process with a set of random individuals (solutions). At each iteration, solutions update their positions using the RK method. To do this, RUN uses a solution and the search mechanism obtained by the RK method. Fig. 2 depicts how a position updates its position by using the RK method. In this study, to provide the global (exploration) and local (exploitation) search, the following scheme is implemented to create the position at the next

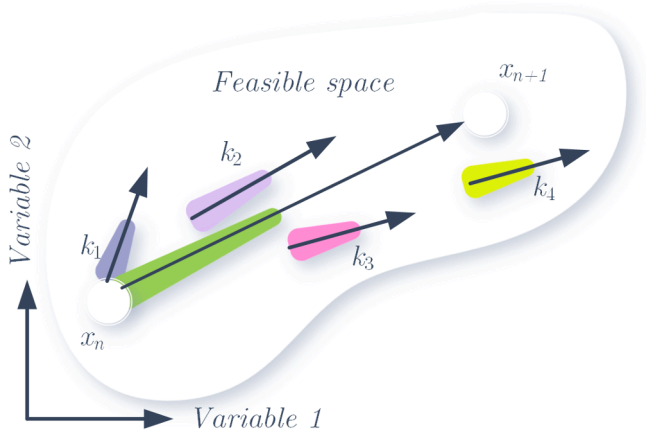

Fig. 2. Slopes employed by the RK to obtain the next position (x_{k+1}) in the RUN algorithm.

iteration:

if $rand < 0.5$ (17)

(exploration phase)

$$x_{n+1} = (x_c) + SF \times SM + \mu \times x_s$$

else

(exploitation phase)

$$x_{n+1} = (x_m) + SF \times SM + \mu \times x_{s'}$$

end

in which

$$\mu = 0.5 + 0.1 \times randn$$

where μ is a random number, *randn* is a random number with a normal distribution.

The formulas of x_s and $x_{s'}$ are expressed as

$$x_s = randn.(x_m - x_c) \quad (17\text{-}1)$$

$$x_{s'} = randn.(x_{r1} - x_{r2}) \quad (17\text{-}2)$$

x_m and x_c can be calculated as follows:

$$x_c = \varphi \times x_n + (1 - \varphi) \times x_{r1} \quad (17\text{-}3)$$

$$x_m = \varphi \times x_{best} + (1 - \varphi) \times x_{lbest} \quad (17\text{-}4)$$

whereφ is a random number in the range of (0,1). x_{best} is the best-so-far solution. x_{lbest} is the best position obtained at each iteration. *SF* is an adaptive factor, which is given by:

$$SF = 2.(0.5 - rand) \times f \quad (17\text{-}5)$$

in which

$$f = a \times exp\left(-b \times rand \times \left(\frac{i}{Maxi}\right)\right) \quad (17\text{-}6)$$

where *a* and *b* are two constant numbers. *i* is the number of iterations. *Maxi* is the maximum number of iterations. In this study, *SF* was employed to provide a suitable balance between exploration and exploitation. Based on Eq. (17–5), a large value of *SF* is specified in the early iterations to increase the diversity and enhance the exploration search; then, its value reduces to promote the exploitation search capability by increasing the number of iterations. The main control parameters of RUN include two parameters employed in the (*SF*), which are *a* and *b*.

The rule in Eq. (17) shows that the proposed RUN selects the exploration and exploitation phases based on the condition $rand < 0.5$. This novel procedure used for optimization in RUN ensures that if $rand < 0.5$, a global search is applied in the solution space and a local search around solution x_c is performed simultaneously. By implementing a novel global search (exploration), the RUN can explore the search space's superior promising regions. On the other hand, if $rand \geq 0.5$, RUN uses a local search around solution x_m. By applying this local search phase, the proposed algorithm can effectively increase the convergence speed and focus on high-quality solutions.

To perform the local search around the solutions x_c and $x_m$and explore the promising regions in the search space, Eq. (17) is rewritten as follows:

if $rand < 0.5$ (18)

(exploration phase)

$$x_{n+1} = (x_c + r \times SF \times g \times x_c) + SF \times SM + \mu \times x_s$$

else

(exploitation phase)

$$x_{n+1} = (x_m + r \times SF \times g \times x_m) + SF \times SM + \mu \times x_{s'}$$

end

where *r* is an integer number, which is 1 or −1. This parameter changes the search direction and increases diversity. *g* is a random number in the range [0, 2]. According to Eq. (18), the local search around x_c decreases as the number of iterations increases. Fig. 3 displays the search mechanism of RUN, indicating how to generate position x_{n+1} at the next iteration.

4.4. Enhanced solution quality

In the RUN algorithm, enhanced solution quality (ESQ) is employed to increase the quality of solutions and avoid local optima in each iteration. By applying ESQ, the RUN algorithm ensures that each solution moves toward a better position. In the proposed ESQ, the average of three random solutions (x_{avg}) is calculated and combined with the best position (x_b) to generate a new solution (x_{new1}). The following scheme is executed to create the solution (x_{new2}) by using the ESQ:

if $rand < 0.5$

if $w < 1$ (19)

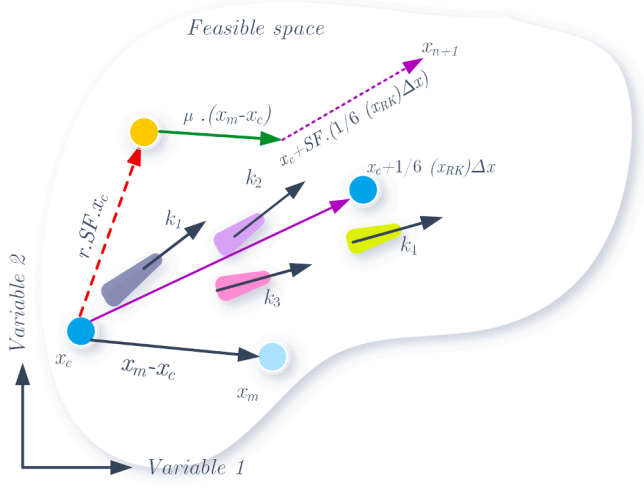

Fig. 3. Search mechanism of the RUN.

$$x_{new2} = x_{new1} + r.w.|(x_{new1} - x_{avg}) + randn|$$

else

$$x_{new2} = (x_{new1} - x_{avg}) + r.w.|(u.x_{new1} - x_{avg}) + randn|$$

end

end

in which

$$w = rand(0,2).exp\left(-c\left(\frac{i}{Maxi}\right)\right)$$

$$x_{avg} = \frac{x_{r1} + x_{r2} + x_{r3}}{3} \tag{19-1}$$

$$x_{new1} = \beta \times x_{avg} + (1-\beta) \times x_{best} \tag{19-2}$$

where β is a random number in the range of [0, 1]. c is a random number, which is equal to 5×*rand* in this study. w is a random number, which decreases with the increasing number of iterations. r is an integer number, which is 1, 0, or −1. x_{best} is the best solution explored so far. According to the above scheme, for $w < 1$ (i.e., the later iterations), solution x_{new2} trends to create an exploitation search, while for $w \geq 1$ (i.e., the early iterations), solution x_{new2} trends to make an exploration search. Note that in the latter condition, to increase the diversity, parameter u is defined. It is noteworthy that ESQ is applied when the condition $rand < 0.5$ is met.

The solution calculated in this part (x_{new2}) may not have better fitness than that of the current solution (i.e., $f(x_{new2}) > f(x_n)$). To have another chance for creating a good solution, another new solution (x_{new3}) is generated, which is defined as follows:

$$\begin{aligned} &\text{if rand} < \boldsymbol{w} \\ &x_{new3} = (x_{new2} - rand.x_{new2}) + SF.(rand.x_{RK} + (v.x_b - x_{new2})) \\ &\text{end} \end{aligned} \tag{20}$$

where v is a random number with a value of $2 \times rand$. In fact, the new solution (x_{new3}) is implemented when the condition rand < $w$is met. The main objective of Eq. (20) is to move the solution x_{new2} towards a better position. In the first rule of this equation, a local search around x_{new2} is generated, and in the second rule, RUN attempts to explore the promising regions with the movement towards the best solution. Hence, to emphasize the importance of the best solution, coefficient v is used. It should be noted that to calculate x_{RK}, solutions x_b and x_w become x_k and x_{new2}, respectively, because the fitness value of x_n is less than that of x_{new2} (*i.e.*, $f(x_{new2}) > f(x_n)$). The pseudo-code of and flowchart of RUN are presented in **Algorithm 1.** and Fig. 4, respectively.

Algorithm 1. The pseudo-code of RUN

Stage 1. Initialization
Initializea,b
Generate the RUN population $X_n(n = 1, 2, \ldots, N)$
Calculate the objective function of each member of population
Determine the solutions x_w, x_b, andx_{best}
Stage 2. RUN operators
  for $i = 1$: *Maxi*
    for $n = 1 : N$
      for $l = 1 : D$
        Calculate position $x_{n+1,l}$ using Eq. (18)
      end for
    Enhance the solution quality
      if*rand* < 0.5
        Calculate position x_{new2} using Eq. (19)
          if $f(x_n) < f(x_{new2})$
            if rand<w
              Calculate position x_{new3} using Eq. (20)
            end
          end
        end
      Update positions x_w andx_b
    end for
    Update positionx_{best}
    $i = i + 1$
  end
Stage 3. returnx_{best}

As shown in Fig. 5, three paths are considered for optimization in RUN. The proposed algorithm first uses the RK search mechanism to generate position x_{n+1} and then employs the ESQ mechanism to explore the promising regions in the search space. According to this mechanism, RUN follows three paths to reach a better solution. In the first and second paths, position x_{new2} calculated by the ESQ is compared with the position x_{k+1}. If the fitness of x_{new2} is worse than that of x_{k+1} (i.e.,$f(x_{new2}) > f(x_{n+1})$), another position (x_{new3}) is generated. If $f(x_{new3}) < f(x_{n+1})$, the best solution is x_{new3} (second path). Otherwise, it is x_{n+1} (first path). In the third path, if $f(x_{new2}) < f(x_{n+1})$, the best solution is x_{new2}.

The following characteristics theoretically demonstrate the proficiency of RUN in solving various complex optimization problems:

- Scale factor (*SF*) has a randomized adaptation nature, which assists RUN in further improving the exploration and exploitation steps. This parameter ensures a smooth transition from exploration to exploitation.
- Using the average position of solutions can promote RUN's exploration tendency in the early iterations.
- RUN employs a search mechanism based on the RK method to boost both exploration and exploitation abilities.

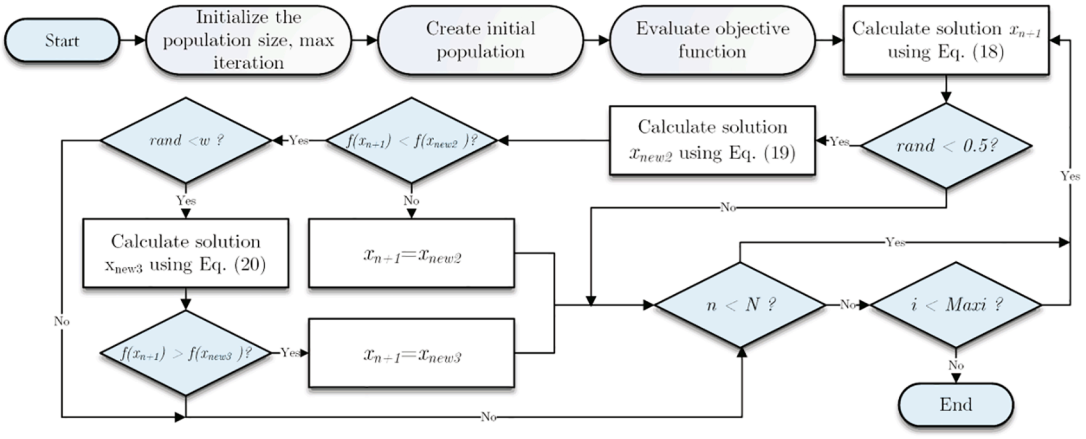

Fig. 4. Flowchart of the RUN algorithm.

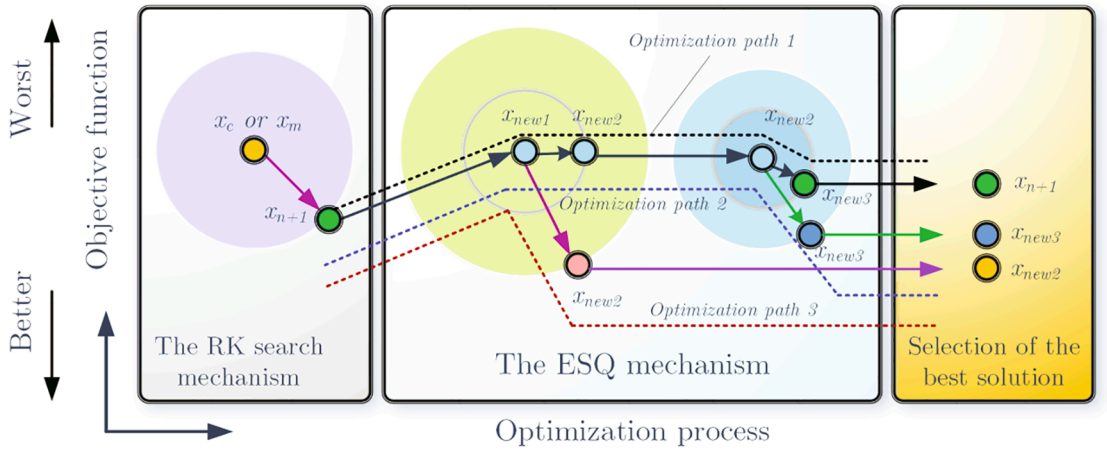

Fig. 5. Optimization process in the RUN.

- The enhanced solution quality (ESQ) in the RUN algorithm utilizes the thus-far best solution to promote the quality of solutions and improve the convergence speed.
- In the RUN algorithm, it is possible that if the new solution is not in a better position than the current solution, it can identify a new different position in the search space to reach a better position. This process can enhance the quality of solutions and improve the convergence rate.
- The search mechanism and ESQ use two randomized variables to emphasize the importance of the best solution and move toward the global best solution, which can effectively balance the exploration and exploitation steps.

4.5. Computational complexity

RUN algorithm mainly includes the following parts: initialization, getting the maximum and minimum fitness, getting the minimum in three random individuals, exploration of the search space, parameter updating, and fitness evaluation. Among them, N indicates the number of individuals in the population, D is the problem's dimension, and $MaxFES$ indicates the maximum number of iterations. The computational complexity of initialization, fitness evaluation, parameter updating, and exploration of the search space is $O(N)$, getting the minimum in three random individuals is $O(3^2)$ and the getting the maximum and minimum fitness is $O(N^2)$. From this, we can get the complexity of the whole algorithm: $O(2N + 2N^2 + MaxFES \times (3N + N^2 + 3^2))$.

5. Results and discussion

The new RUN algorithm's ability was verified using 20 benchmark functions, which have been used by many researchers (Ahmadianfar et al., 2019; Huang, Zhang, Song, Zhang, & Shi, 2019; Tian & Gao, 2017; Zhao, Wang, & Zhang, 2019). The set of benchmark problems employed in this study involves three families of mathematical functions: unimodal functions (UFs) (f_1-f_6), multimodal functions (MFs) (f_7-f_{14}), and hybrid functions (HFs) (f_{15}-f_{20}) The details on these test functions are shown in Tables 1–3.

The unimodal test functions with a global best position can evaluate different optimization algorithms' exploitative behavior, while the multimodal test functions can assess their exploration and local optima avoidance capabilities. It should be noted that the hybrid test functions are more challenging and complicated than the unimodal and multimodal test functions. Therefore, they are incredibly suitable to validate the optimizers' ability to solve complicated real-world optimization problems. The proposed RUN results and efficiency were compared with those of other well-known algorithms, including the GWO (Mirjalili, Mirjalili, & Lewis, 2014), WOA (Mirjalili & Lewis, 2016), WCA (Eskandar, Sadollah, Bahreininejad, & Hamdi, 2012), IWO (Hosseini, 2007), and cuckoo search (CS) (Yang & Deb, 2010) algorithms, based on the average and standard deviation of the results. Six different test functions were selected to assess the effects of the RUN algorithm qualitatively. Fig. 6 depicts the qualitative results of test functions f1, f2, f4, f7, f10, and f12. RUN was employed for minimizing these functions by using five solutions over 200 iterations.

Table 1
Unimodal test functions.

Function	D	Range	f_{min}
$f_1(x) = x_1^2 + 10^6\sum_{i=2}^{D} x_i^2$	30	[−100, 100]	0
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	30	[−100, 100]	0
$f_3(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4$	30	[−100, 100]	0
$f_4(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[−100, 100]	0
$f_5(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$	30	[−100, 100]	0
$f_6(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	30	[−100, 100]	0

5.1. Experimental setup

The population size and the total number of iterations were set respectively equal to 50 and 500 for the UFs and MFs and 50 and 1000 for the HFs. All results were presented and compared in terms of the optimization algorithms' average efficiencies over 30 independent runs. For GWO, WOA, CS, IWO, and WCA, the control parameters were the same as those suggested in the original work. Table 4 lists the parameters used in this study

5.2. Qualitative results of RUN

Three well-known qualitative metrics used to demonstrate RUN's performance were search history, trajectory graph, and convergence curve. The search history graph discloses the history of the RUN algorithm's positions during the optimization process. The trajectory curve displays how the first dimension of a solution changed during the iterations. The convergence curve demonstrates how the fitness value of the best solution changed during the optimization process.

Fig. 6 shows that RUN yielded a similar pattern to solve different problems regarding the history of positions. This indicates that an attempt was made to initially increase the exploration and find the promising regions of the search space and then exploit the neighborhood of the best solutions. From the trajectory curves in Fig. 6, it can be

Table 2
Multimodal test functions.

Function	D	Range	f_{min}
$f_7(x) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_{D-1}, x_D) + g(x_D, x_1)$ $g(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}) - 0.5)}{(1+0.001(x^2+y^2))^2}$	30	[-100, 100]	0
$f_8(x) = sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i-1)^2[1+10sin^2(\pi w_i+1)] + (w_D-1)^2[1+sin^2(2\pi w_D)]$ where $w_i = 1 + \frac{x_i-1}{4}$	30	[-100, 100]	0
$f_9(x) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i), z_i = x_i + 4.209687462275036e+002 g(z_i) =$ $\begin{cases} z_i \sin(\lvert z_i \rvert^{\frac{1}{2}}) & if \lvert z_i \rvert \le 500 \\ (500 - mod(z_i, 500))\sin(\sqrt{\lvert 500 - mod(z_i, 500) \rvert}) - \frac{(z_i-500)^2}{10000D} & if z_i > 500 \\ (mod(\lvert z_i \rvert, 500) - 500)\sin(\sqrt{\lvert mod(\lvert z_i \rvert, 500) - 500 \rvert}) - \frac{(z_i+500)^2}{10000D} & if z_i < -500 \end{cases}$	30	[-100, 100]	0
$f_{10}(x) = 20 - 20 \times \exp(-0.2\sqrt{\frac{1}{D}(\sum_{i=1}^{D} x_i^2)}) - \exp(\frac{1}{D}\sqrt{\sum_{i=1}^{D} cox(2\pi x_i)}) + e$	30	[–32, 32]	0
$f_{11}(x) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k(x_i+0.5))]) - D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k .0.5)]$ $a = 0.5,\ b = 3,\ kmax = 20$	30	[-100, 100]	0
$f_{12}(x) = \lvert \sum_{i=1}^{D} x_i^2 - D \rvert^{\frac{1}{4}} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	30	[-100, 100]	0
$f_{13}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600, 600]	0
$f_{14}(x) = \frac{\pi}{D}\{10\sin(\pi z_1) + \sum_{i=1}^{D-1}(z_i-1)^2[1+10sin^2(\pi z_{i+1})] + (z_D-1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $z_i = 1 + \frac{x_i+1}{4}$	30	[-50, 50]	0

Table 3
Hybrid benchmark functions.

Test function	Name	D	Search space	f_{min}
$f_{15}(x)$	HF 1 ($N = 3$)	30	[-100, 100]	1700
$f_{16}(x)$	HF 2 ($N = 3$)	30	[-100, 100]	1800
$f_{17}(x)$	HF 3 ($N = 4$)	30	[-100, 100]	1900
$f_{18}(x)$	HF 4 ($N = 4$)	30	[-100, 100]	2000
$f_{19}(x)$	HF 5 ($N = 5$)	30	[-100, 100]	2100
$f_{20}(x)$	HF 6 ($N = 5$)	30	[-100, 100]	2200

observed that RUN began the searching process with sudden fluctuations, which involved about 100% of the search space. This behavior reveals the exploration tendency of the RUN algorithm. As the number of iterations increased, the amplitude of these variations reduced. This procedure ensured the transition of RUN from the exploratory search towards exploitative trends. Therefore, it is concluded from the trajectory graphs that the RUN algorithm first provided the exploration trend and then shifted to the exploitation stage.

The convergence graph is usually employed to assess the convergence performance of optimizers. Fig. 6 displays an accelerated reducing pattern in all convergence curves, especially in the early iterations. It also shows the approximate timing when RUN transferred from the exploration to the exploitation phase. These results demonstrate the suitable accelerated convergence behavior of RUN.

5.3. Assessment of the exploitative behavior

Typically, UFs are used to test the exploitability of the optimization algorithms. Since UFs (f_1-f_6) have only one global best solution, they can be used to evaluate the exploitation ability of the optimization algorithms. Table 5 shows the results of the RUN, GWO, WOA, CS, IWO, and WCA algorithms for the UFs, including the average, best, and standard deviation values of the fitness function for 30 different runs. The comparisons of RUN with the five other meta-heuristic optimization algorithms demonstrated that RUN was the best optimizer to solve the UFs and provide competitive results. Particularly, the proposed RUN algorithm exhibited an excellent exploitation behavior.

5.4. Assessment of the exploratory behavior

The multimodal functions (f_7-f_{14}) were used to validate all optimizers' exploratory behaviors since they had many local optimal solutions. Table 5 shows the results of MFs obtained by the RUN, GWO, WOA, CS, IWO, and WCA algorithms, indicating the superior performance of RUN to the other optimizers, except for f_{11}. For function f_{11}, RUN was inferior to the WOA algorithm and superior to GWO and WCA. The results presented in Table 5 for test functions f_7-f_{14} demonstrate that RUN also has a superior exploration ability due to the use of the exploration mechanism that ensures the search process towards the global best solution.

5.5. Ability to avoid local optima

The RUN's ability to avoid the local optima was evaluated by using hybrid functions (f_{15} - f_{20}). These test functions are regarded as the most complicated benchmark test functions, and only an optimizer with an appropriate balance between global and local optima can avoid the local solutions. Table 6 presents the results of RUN and the five other optimizers on the HFs.

For the results of the HFs in Table 6, it can be clearly observed that RUN was the best optimizer among the six optimization algorithms on functions f_{15}- f_{19} according to their average fitness values. For function

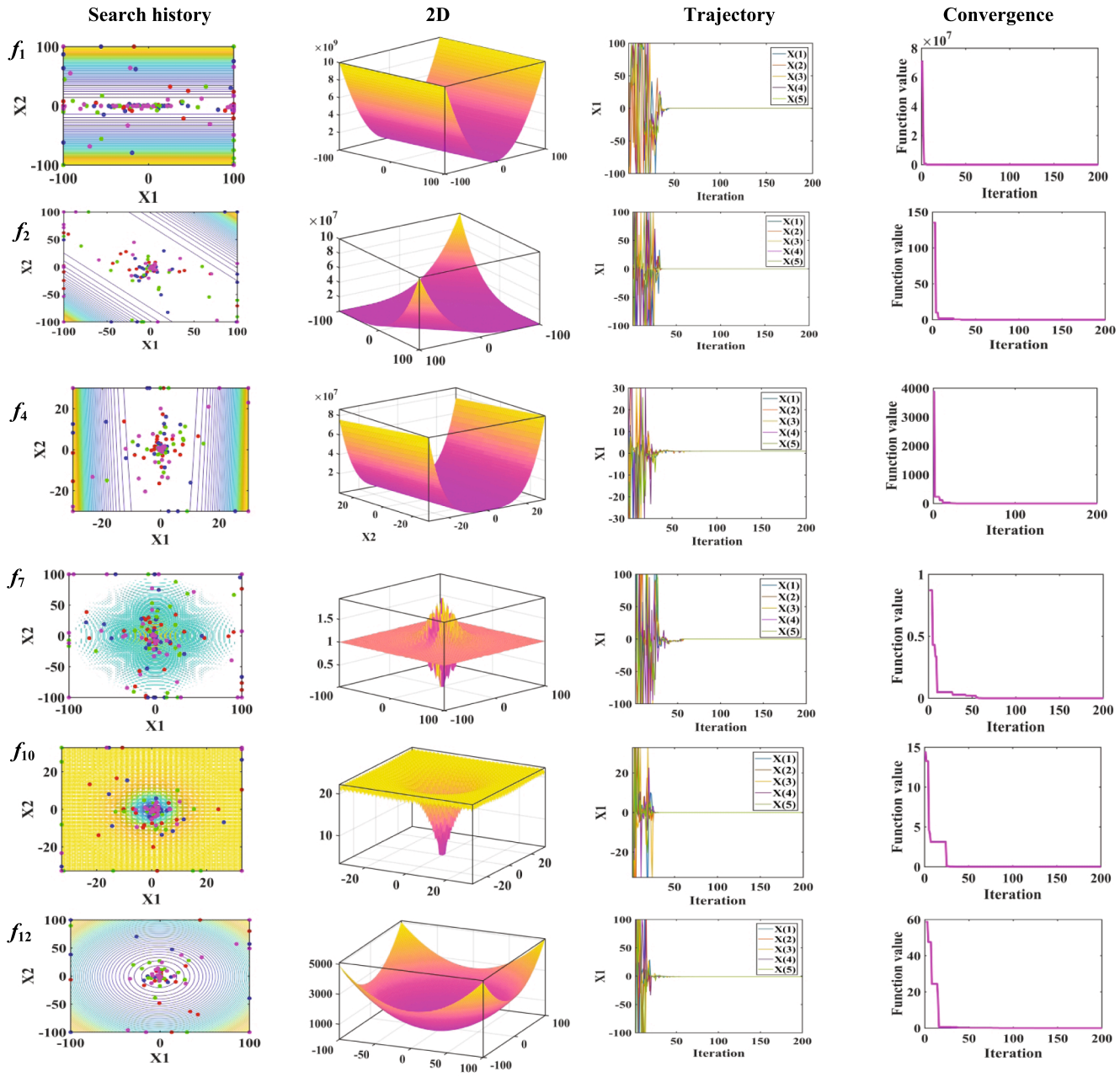

Fig. 6. Qualitative results of six benchmark test functions.

Table 4
Parameter settings of optimization algorithms.

Optimizers	Parameters
RUN	$a = 20$ and $b = 12$
GWO	$a = 2 - 2 \times i/Maxi$
WOA	$a = 2 - 2 \times i/Maxi$
	$a_2 = -1 - 1 \times i/Maxi$
CS	Rate of discovery = 0.25
WCA	number of rivers + sea (Nsr) = 10
	a controlling parameter (d_{max}) = 0.1
IWO	maximum number of seeds (S_{max}) = 15
	minimum number of seeds (S_{min}) = 0
	initial value of standard deviation $\sigma_{initial} = 5$
	final value of standard deviation $\sigma_{final} = 0.01$

f_{20}, RUN was surpassed by GWO but superior to the WOA, CS, IWO, and WCA algorithms. Indeed, the proposed optimizer was the second-best effective optimizer for this test functions. This capability is due to the adaptive mechanism employed to update the *SF* parameter and the ESQ mechanism in the proposed RUN, which assures a good transition from exploration to exploitation.

5.6. Assessment of the convergence ability

Notwithstanding, the results presented in Tables 5 and 6 demonstrate the RUN algorithm's superior efficiency compared with the other optimizers. However, the convergence behavior analysis must also be performed to further assess the proposed RUN 's performance in solving optimization problems. The convergence curves of RUN, GWO, WOA, CS, IWO, and WCA are depicted in Fig. 7, revealing the relationships of the best-so-far fitness value explored (y-axis) and the number of functional evaluations (NFE) (x-axis).

According to the convergence curves (Fig. 7), the following

Table 5
Results of the UFs and MFs from RUN and five other meta-heuristic optimization algorithms.

Optimizer		UFs							
		f_1	f_2	f_3	f_4	f_5	f_6		
RUN	Average	**1.75E-132**	**6.68E-267**	**2.16E-129**	**2.45E + 01**	**1.26E-137**	**2.35E-130**		
	Best	5.31E-145	3.55E-278	1.81E-145	2.29E + 01	6.74E-147	1.20E-145		
	SD	9.04E-132	0.00E + 00	1.18E-128	1.04E + 00	5.31E-137	1.29E-129		
GWO	Average	3.87E-27	4.17E-97	5.78E-29	2.68E + 01	5.60E-33	5.14E-30		
	Best	4.33E-29	2.8E-108	2.25E-31	2.52E + 01	1.61E-34	1.12E-31		
	SD	7.73E-27	1.87E-96	1.48E-28	7.53E-01	5.84E-33	8.14E-30		
CS	Average	2.52E-02	1.81E + 01	9.00E-01	1.39E + 02	5.16E-04	1.88E-01		
	Best	4.44E-05	1.46E-06	5.38E-03	2.96E + 01	6.67E-06	1.22E-02		
	SD	1.17E-01	8.44E + 01	1.70E + 00	2.37E + 02	7.63E-04	3.04E-01		
WCA	Average	2.31E-05	6.77E-07	5.02E-09	7.38E + 01	6.27E-07	2.86E + 03		
	Best	2.22E-07	4.05E-25	1.11E-10	8.80E-01	3.13E-12	7.39E-08		
	SD	7.01E-05	3.70E-06	9.07E-09	6.54E + 01	3.00E-06	7.78E + 03		
WOA	Average	6.75E-80	1.56E-110	5.52E + 03	2.75E + 01	2.86E-84	1.30E-81		
	Best	9.43E-89	9.17E-141	2.88E + 01	2.69E + 01	2.63E-94	2.90E-89		
	SD	2.45E-79	7.86E-110	3.85E + 03	4.12E-01	1.11E-83	5.59E-81		
IWO	Average	3.18E + 03	1.53E + 03	4.24E + 02	4.10E + 04	5.69E + 04	5.01E + 06		
	Best	8.84E + 01	1.06E-05	6.12E-05	2.37E + 01	4.21E + 04	1.25E + 06		
	SD	3.14E + 03	1.96E + 03	6.40E + 02	9.02E + 04	1.23E + 04	2.57E + 06		
		MFs							
		f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
RUN	Average	**0.00E + 00**	**2.04E-01**	**3.82E-04**	**8.88E-16**	1.04E-13	**3.42E-01**	**0.00E + 00**	**6.59E-08**
	Best	0.00E + 00	4.21E-07	3.82E-04	8.88E-16	6.39E-14	2.33E-01	0.00E + 00	3.33E-08
	SD	0.00E + 00	1.13E-01	0.00E + 00	0.00E + 00	1.63E-14	7.53E-02	0.00E + 00	1.95E-08
GWO	Average	5.91E + 00	1.01E + 00	3.82E-04	4.46E-14	2.91E + 01	6.39E-01	6.13E-03	3.20E-02
	Best	2.11E + 00	6.36E-01	3.82E-04	3.64E-14	2.27E + 01	4.41E-01	0.00E + 00	6.40E-03
	SD	2.20E + 00	1.59E-01	8.72E-13	4.19E-15	3.34E + 00	9.60E-02	1.20E-02	2.33E-02
CS	Average	9.86E + 00	2.41E + 00	4.12E-04	3.73E-03	6.23E-02	5.93E-01	1.47E-02	1.69E-01
	Best	7.74E + 00	6.28E-01	3.82E-04	4.69E-04	8.53E-14	4.42E-01	4.35E-10	5.29E-08
	SD	8.36E-01	2.27E + 00	4.54E-05	3.44E-03	9.52E-02	8.40E-02	1.80E-02	2.68E-01
WCA	Average	1.20E + 01	2.92E + 03	5.19E-03	3.40E + 00	1.20E-01	5.30E-01	3.13E-02	3.64E-01
	Best	1.03E + 01	1.10E + 03	3.82E-04	2.19E-02	8.53E-14	2.53E-01	5.08E-12	1.53E-12
	SD	6.12E-01	1.36E + 03	2.63E-02	2.28E + 00	5.12E-01	1.50E-01	3.86E-02	7.04E-01
WOA	Average	3.00E + 00	5.12E-01	3.82E-04	3.73E-15	**1.92E-14**	5.24E-01	3.05E-03	1.03E-02
	Best	0.00E + 00	6.99E-02	3.82E-04	8.88E-16	7.11E-15	2.60E-01	0.00E + 00	1.30E-03
	SD	4.43E + 00	3.58E-01	5.55E-13	2.70E-15	6.62E-14	1.88E-01	1.67E-02	1.59E-02
IWO	Average	1.30E + 01	4.59E + 03	6.89E + 02	1.24E + 00	5.29E + 00	3.58E-01	1.67E + 02	1.27E-01
	Best	1.21E + 01	3.25E + 03	3.90E-04	5.07E-03	3.00E + 00	2.21E-01	9.25E + 01	4.80E-02
	SD	4.09E-01	6.11E + 02	3.84E + 02	4.71E + 00	1.57E + 00	8.82E-02	3.99E + 01	8.93E-02

Table 6
Statistical results of the HFs from RUN and five other optimizers.

Optimizer		HFs					
		f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
RUN	Average	**104191.21**	**3435.33**	**1919.53**	**3519.30**	**48127.89**	2674.29
	Best	26504.80	2149.82	1911.91	2345.66	10865.46	2229.29
	SD	42897.96	801.49	5.01	2215.65	22065.81	227.33
GWO	Average	2017606.11	9419404.67	1945.42	23438.34	865855.49	**2581.81**
	Best	243778.74	4056.06	1912.26	11065.71	66706.84	2250.33
	SD	2197530.17	22146302.91	26.45	12065.16	1222558.84	145.41
CS	Average	1638591.37	8614.09	1931.73	94953.78	405641.76	3114.17
	Best	168986.27	2070.91	1909.39	3577.19	16508.82	2364.87
	SD	1608329.34	8165.00	30.62	309592.19	577986.74	364.57
WCA	Average	1096464.13	5561515.91	1927.69	24082.37	339962.26	2832.20
	Best	177,033	2413.67	1910.27	5378.61	23640.99	2579.874
	SD	742290.81	30411215.42	29.31	15291.10	223453.44	136.44
WOA	Average	11178976.28	93612.11	1964.90	76381.26	3876550.62	3084.20
	Best	2520022.97	9512.03	1919.07	28141.42	189834.25	2476.51
	SD	7349962.08	94864.91	34.80	48244.50	4182086.86	252.11
IWO	Average	110385.61	5178.86	1922.03	30483.82	53137.20	3263.82
	Best	15620.9	2229.473	1907.79	3739.462	11885.51	2729.88
	SD	73296.20	3721.69	21.40	13771.33	31510.29	283.44

conclusions can be obtained:

- Concerning the convergence rate, the IWO, WCA, and CS algorithms displayed weak performances in optimizing the UFs and MFs, followed by the WOA and GWO algorithms.
- The RUN optimizer had a faster convergence curve than the other algorithms for the unimodal and multimodal test functions due to the proper balance between exploration and exploitation.

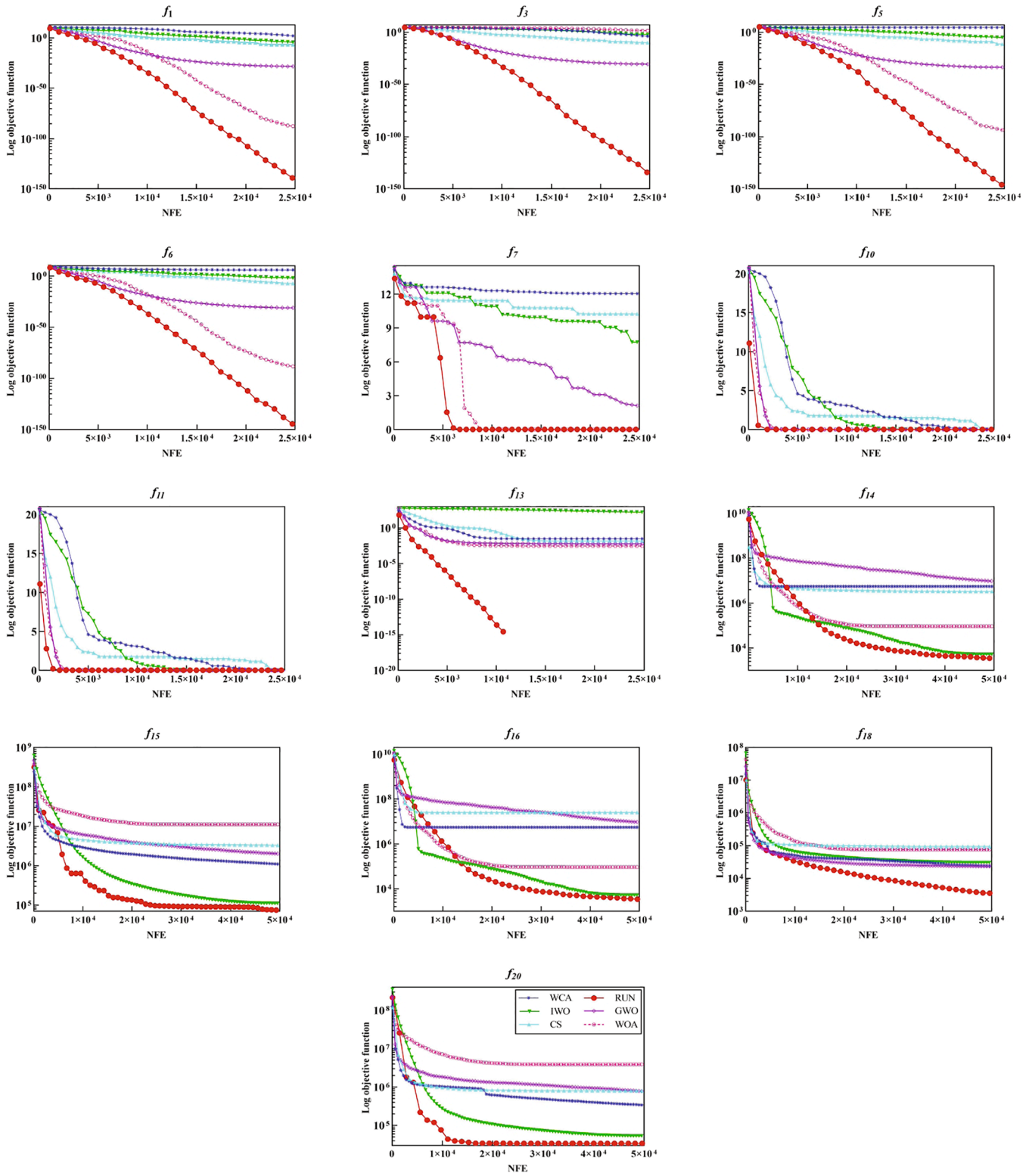

Fig. 7. Convergence graphs of the RUN and five other optimizers for the selected UFs, MFs, and HFs.

- For the HFs, the convergence rate of RUN tended to be accelerated by increasing the number of functional evaluations due to the ESQ and adaptive mechanism, which helped it to explore the promising areas of the solution space in the early iterations and more quickly converge towards the optimal solution after spending about 15% of the total number of function evaluations.
- The convergence curves revealed that RUN provides a more suitable convergence speed to optimize the test functions than the other optimizers.

5.7. Ranking analysis

The Friedman and Quade tests (Derrac, García, Molina, & Herrera,

2011) were conducted to determine the six optimizers' influential performances. These tests employ a nonparametric two-way analysis of variance, which allows the comparison of several samples. Based on the Friedman test, all samples are equal in terms of importance. In contrast, the Quade test considers the fact that some samples are more difficult or complicated than others and, thus, provides a weighted ranking analysis of the samples (Derrac et al., 2011).

Tables 7 and 8 show the Friedman and Quade test ranks, including the individual, average, and final ranks for the average fitness values from RUN and the five other optimizers on all UF, MF, and HF test functions. The Friedman and Quade test results indicated that the RUN algorithm performed the best among the six algorithms on all test functions.

Table 9 displays the statistics and *p*-values of the Friedman and Quade tests for all test functions. As per the *p*-values calculated for the two tests, significant differences can be seen among all optimizers.

5.8. Comparison of RUN with advanced optimizers

In order to further evaluate the efficiency of RUN, it was compared with eight advanced optimizers including CGSCA (Kumar, Hussain, Singh, & Panigrahi, 2017), SCADE (Nenavath & Jatoth, 2018), BMWOA (Heidari, Aljarah, et al., 2019), BWOA (Chen, Xu, Wang, & Zhao, 2019), OBLGWO (Heidari, Abbaspour, Chen, 2019), CAMES (Hansen, Müller, & Koumoutsakos, 2003), GL25 (García-Martínez, Lozano, Herrera, Molina, & Sánchez, 2008), and CLPSO (Liang, Qin, Suganthan, & Baskar, 2006) in solving the CEC-BC-2017 benchmark functions. The population size, maximum number of iterations, and dimension were set to 30, 500, and 30, respectively. All the optimization algorithms were also performed in 30 different runs for each mathematical test function.

The best, average, and standard deviation of the results calculated by RUN and the eight advanced optimizers are summarized in Table 10. As shown in Table 10, RUN presented promising results on the CEC-BC-2017 functions compared with the other optimizers. Moreover, the proposed RUN displayed the best performance in the 20 test functions (f_2, f_4, f_5, f_7, f_8, f_9, f_{10}, f_{11}, f_{13}, f_{15}-f_{24}, and f_{26}) and the second-best efficiency in the remaining 10 test functions (f_1, f_3, f_6, f_{12}, f_{14}, f_{25}, and f_{27}-f_{30}). In this study, to compute the average ranks of the optimization algorithms and specify their differences, the Friedman test was performed. Table 11 displays the average ranks of all the optimizers, where RUN achieved the best rank (1.33). Therefore, RUN had the best efficiency compared with the eight advanced optimizers. To investigate the convergence speed of RUN, the convergence curves were obtained for all the optimizers on the CEC-BC-2017 functions (Fig. 8). It can be observed from Fig. 8 that RUN achieved accurate solutions with a faster convergence rate than the eight advanced optimizers.

5.9. Sensitivity analysis of RUN

The sensitivity analysis of the control parameters of RUN (i.e., a and b) was performed, which demonstrated that RUN had a very low sensitivity to the parameter changes. This research evaluated different combinations of the control parameters on 34 mathematical test functions for designing RUN, including two groups, 14 unimodal and multimodal test functions (group 1) and 20 test functions of CEC-BC-2017 (group 2). In this regard, the values of each parameter were defined as $a = [5, 10, 20, 30, 40]$ and $b = [4, 8, 12, 16, 20]$. Since each parameter had 5 values, there were 25 combinations of the design parameters. Each combination was evaluated by the average fitness values obtained from 30 different runs. Fig. 9(a) illustrates the mean rank values of the two groups, and Fig. 9(b) presents the average rank values of the two groups. Accordingly, the best rank belongs to C13 ($a = 20$ and $b = 12$), and the rank of C19 is very close to C13. Also, the ranks for most combinations are very close, indicating that the proposed algorithm is not very sensitive to the parameter changes.

6. Engineering benchmark problems

Four engineering benchmark problems were selected in this study to evaluate the performance of the proposed RUN algorithm. Solving such engineering design problems by utilizing specific optimization algorithms is a suitable way to test their capabilities (Heidari et al., 2019). The results obtained by RUN were compared with those of different well-known optimizers suggested in previous studies. It is worth noting that the population size and the maximum number of iterations were 30 and 500, respectively, for all problems.

Table 7
Friedman ranks for the UFs, MFs, and HFs for RUN and five other optimizers.

Optimizers	UFs								Average Rank	Rank
	f_1	f_2	f_3	f_4	f_5	f_6				
RUN	1	1	1	1	1	1			1.00	**1**
GWO	3	3	2	2	3	3			2.67	2
CS	5	6	4	6	5	4			5.00	5
WCA	4	4	5	4	4	5			4.33	4
WOA	2	2	6	3	2	2			2.83	3
IWO	6	5	3	5	6	6			5.17	6
	MFs									
	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}		
RUN	1	1	2	1	2	1	1	1	1.25	**1**
GWO	3	3	2	3	6	6	3	3	3.63	4
CS	4	4	4	4	3	5	4	5	4.13	3
WCA	5	5	5	6	4	4	5	6	5.00	5
WOA	2	2	2	2	1	3	2	2	2.00	2
IWO	6	6	6	5	5	2	6	4	5.00	5
	HFs									
	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}				
RUN	1	1	1	1	1	2			1.17	**1**
GWO	5	6	5	3	5	1			4.17	4
CS	4	3	4	6	4	5			4.33	5
WCA	3	5	3	2	3	3			3.17	3
WOA	6	4	6	5	6	4			5.17	6
IWO	2	2	2	4	2	6			3.00	2

Table 8
Quade ranks for the UFs, MFs, and HFs for RUN and five other optimizers.

Optimizers	UFs								Average Rank	Rank
	f_1	f_2	f_3	f_4	f_5	f_6				
RUN	5	1	2	6	3	4			1.00	**1**
GWO	10	2	8	12	4	6			2.67	2
CS	6	15	12	18	3	9			4.57	5
WCA	16	12	4	20	8	24			4.14	4
WOA	20	5	30	25	10	15			2.76	3
IWO	18	12	6	24	30	36			5.86	6
	MFs									
	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}		
RUN	1.5	7	6	3	4	8	1.5	5	1.33	**1**
GWO	28	24	8	4	32	20	12	16	3.31	3
CS	24	21	3	6	12	18	9	15	3.94	4
WCA	35	40	5	30	15	25	10	20	4.97	5
WOA	16	12	6	2	4	14	8	10	1.89	2
IWO	30	48	42	18	24	12	36	6	5.56	6
	HFs									
	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}				
RUN	6	3	1	4	5	2			1.10	**1**
GWO	25	30	5	15	20	10			4.57	5
CS	18	9	3	12	15	6			4.14	4
WCA	20	24	4	12	16	8			3.33	3
WOA	36	24	6	18	30	12			5.19	6
IWO	12	6	2	8	10	4			2.67	2

Table 9
Statistic and *p*-value computed by the Friedman and Quade tests for the UFs, MFs, and HFs.

Average ranking		
Quade	Friedman	
UFs		
10.3445	24.7619	Statistic
1.83e-05	1.55e-04	*p*-value
MFs		
12.9663	28.3333	Statistic
3.61E-07	3.13e-05	*p*-value
HFs		
5.0844	16.6667	Statistic
2.40E-03	5.20E-03	*p*-value

6.1. Rolling element bearing design problem

The primary goal of this problem is to maximize the fatigue life, which is a function of the dynamic load-carrying capacity. It has ten variables and nine constraints for modeling and geometric-based limitations. The problem is described in detail by Gupta, Tiwari, and Nair (2007). The problem is described in detail in (Gupta et al., 2007). The related mathematical formulation is detailed in Appendix A.

Fig. 10 displays the schematic view of the rolling element bearing design problem.

The results of RUN were compared with those of the GA (Gupta et al., 2007), teaching–learning-based optimization (TLBO) (Rao, Savsani, & Vakharia, 2011), passing vehicle search (PVS) (Savsani & Savsani, 2016), and HHO (Heidari et al., 2019) algorithms. Table 12 presents the statistical results from RUN, GA, TLBO, PVS, and HHO optimizers, indicating that RUN achieved the best fitness value with significant progress. The optimal variables of the problem for the five optimizers are shown in Table 13.

6.2. Speed reducer design problem

In this problem, the weight of speed reducer is maximized (Mezura-Montes & Coello, 2005). The mathematical formulation of this problem is detailed in Appendix A. The numbers of variables and constraints of this problem were 7 and 11, respectively, and the schematic is depicted in Fig. 11.

RUN's optimal results were compared with the CS results (Gandomi et al., 2013), HGSO (Hashim, Houssein, Mabrouk, Al-Atabany, & Mirjalili, 2019), GWO, and WOA optimizers. Table 14 gives the results of these optimization algorithms for this problem. It can be observed that RUN achieved the best solution and outperformed the compared optimizers. In addition, the optimal variables of the problem are tabulated in Table 15.

6.3. Three-bar truss problem

The objective of this problem is to minimize the weight of a three-bar truss (Cheng & Prayogo, 2014; Gandomi et al., 2013), which is one of the widely-used engineering problems in previous studies. Fig. 12 displays this problem's shape, in which the main variables include the areas of bars 1, 2, and 3. The mathematical formulation (i.e., objective function and constraints) of the problem is detailed in Appendix A.

The results of RUN were compared with those of MVO (Mirjalili et al., 2016), grasshopper optimization algorithm (GOA) (Mirjalili, Mirjalili, Saremi, Faris, & Aljarah, 2018), moth-flame optimization (MFO) (Mirjalili, 2015b), mine blast algorithm (MBA) (Sadollah, Bahreininejad, Eskandar, & Hamdi, 2013), CS (Gandomi et al., 2013), and HHO (Heidari et al., 2019). Table 16 displays the results acquired from RUN and the six other optimizers, revealing that the proposed RUN yielded better results than the other optimizers. Furthermore, the optimized variables obtained by the seven optimization algorithms are shown in Table 17.

6.4. Cantilever beam problem

Fig. 13 depicts the five-stepped cantilever beam problem, for which the main variables are the height and width of the beam [63]. The main goal of the problem is to minimize the beam weight. The main formulation of the problem is defined in Appendix A.

The RUN optimized the problem, and its results were compared with those of CS (Gandomi et al., 2013), method of moving asymptotes

Table 10
Statistical results of the RUN and eight advanced optimizers on CEC-BC-2017.

		RUN	CGSCA	SCADE	BMWOA	BWOA	OBLGWO	CMAES	GL25	CLPSO
f_1	Best	1.44E + 04	1.53E + 10	1.87E + 10	5.20E + 08	1.94E + 09	4.44E + 07	1.04E + 02	6.83E + 09	7.65E + 09
	Average	3.75E + 04	2.51E + 10	2.97E + 10	1.10E + 09	5.58E + 09	1.57E + 08	**5.45E + 03**	1.69E + 10	1.16E + 10
	SD	1.40E + 04	5.37E + 09	4.86E + 09	3.73E + 08	2.05E + 09	8.59E + 07	5.75E + 03	5.28E + 09	2.59E + 09
	Best	2.92E + 14	9.54E + 33	6.98E + 34	6.58E + 22	1.25E + 27	2.68E + 17	2.02E + 10	2.93E + 30	4.62E + 32
f_2	Average	**4.17E + 17**	8.96E + 38	1.13E + 40	1.86E + 30	4.23E + 35	3.80E + 22	2.59E + 31	4.01E + 38	1.29E + 43
	SD	1.15E + 18	2.88E + 39	3.27E + 40	1.01E + 31	1.58E + 36	9.92E + 22	1.42E + 32	1.32E + 39	7.05E + 43
f_3	Best	3.59E + 04	5.40E + 04	5.72E + 04	5.00E + 04	5.78E + 04	3.27E + 04	1.23E + 05	1.22E + 05	1.09E + 05
	Average	5.05E + 04	7.16E + 04	7.68E + 04	7.99E + 04	7.51E + 04	**4.97E + 04**	1.94E + 05	1.72E + 05	1.56E + 05
	SD	8.29E + 03	1.03E + 04	7.59E + 03	1.03E + 04	7.58E + 03	8.31E + 03	5.92E + 04	3.46E + 04	2.38E + 04
f_4	Best	4.71E + 02	1.45E + 03	4.93E + 03	6.09E + 02	8.77E + 02	5.19E + 02	5.02E + 02	1.58E + 03	1.97E + 03
	Average	**5.13E + 02**	3.57E + 03	6.99E + 03	7.31E + 02	1.41E + 03	5.57E + 02	9.98E + 02	3.22E + 03	3.08E + 03
	SD	1.81E + 01	9.87E + 02	1.29E + 03	1.11E + 02	3.98E + 02	3.05E + 01	3.64E + 02	1.07E + 03	8.66E + 02
f_5	Best	5.92E + 02	7.79E + 02	8.19E + 02	7.10E + 02	7.23E + 02	6.10E + 02	5.79E + 02	7.44E + 02	7.54E + 02
	Average	**6.53E + 02**	8.52E + 02	8.74E + 02	8.09E + 02	8.20E + 02	6.84E + 02	1.22E + 03	8.46E + 02	8.08E + 02
	SD	2.91E + 01	3.21E + 01	2.41E + 01	4.46E + 01	3.44E + 01	5.05E + 01	1.92E + 02	3.96E + 01	2.60E + 01
f_6	Best	6.23E + 02	6.54E + 02	6.58E + 02	6.53E + 02	6.55E + 02	6.07E + 02	6.74E + 02	6.44E + 02	6.41E + 02
	Average	6.40E + 02	6.70E + 02	6.74E + 02	6.68E + 02	6.74E + 02	**6.25E + 02**	6.97E + 02	6.66E + 02	6.58E + 02
	SD	8.22E + 00	7.50E + 00	9.15E + 00	8.50E + 00	9.29E + 00	1.28E + 01	1.30E + 01	9.60E + 00	7.81E + 00
f_7	Best	8.02E + 02	1.16E + 03	1.17E + 03	1.10E + 03	1.10E + 03	8.78E + 02	7.71E + 02	1.18E + 03	1.13E + 03
	Average	**9.36E + 02**	1.26E + 03	1.26E + 03	1.25E + 03	1.30E + 03	1.01E + 03	4.29E + 03	1.33E + 03	1.23E + 03
	SD	5.73E + 01	4.96E + 01	5.32E + 01	8.54E + 01	7.04E + 01	6.22E + 01	1.19E + 03	8.26E + 01	4.37E + 01
f_8	Best	8.75E + 02	1.07E + 03	1.06E + 03	9.74E + 02	9.59E + 02	8.99E + 02	8.59E + 02	1.07E + 03	1.04E + 03
	Average	**9.21E + 02**	1.11E + 03	1.12E + 03	1.03E + 03	1.02E + 03	9.61E + 02	1.37E + 03	1.12E + 03	1.10E + 03
	SD	2.62E + 01	1.99E + 01	2.12E + 01	2.46E + 01	2.51E + 01	4.16E + 01	1.68E + 02	2.65E + 01	2.67E + 01
f_9	Best	2.09E + 03	5.18E + 03	7.68E + 03	5.69E + 03	6.20E + 03	1.34E + 03	1.00E + 04	5.04E + 03	5.13E + 03
	Average	**3.52E + 03**	8.85E + 03	1.06E + 04	8.59E + 03	7.51E + 03	4.46E + 03	1.50E + 04	9.25E + 03	1.14E + 04
	SD	8.96E + 02	1.65E + 03	1.05E + 03	1.27E + 03	1.11E + 03	2.28E + 03	2.48E + 03	2.41E + 03	2.35E + 03
f_{10}	Best	3.85E + 03	7.34E + 03	7.69E + 03	6.28E + 03	5.96E + 03	4.44E + 03	4.93E + 03	8.60E + 03	7.27E + 03
	Average	**5.14E + 03**	8.92E + 03	8.70E + 03	7.80E + 03	7.37E + 03	6.96E + 03	6.21E + 03	9.51E + 03	8.12E + 03
	SD	7.73E + 02	3.96E + 02	3.50E + 02	5.80E + 02	8.46E + 02	1.44E + 03	6.32E + 02	5.11E + 02	3.77E + 02
f_{11}	Best	1.19E + 03	2.57E + 03	3.56E + 03	1.38E + 03	2.27E + 03	1.28E + 03	1.35E + 03	4.48E + 03	3.24E + 03
	Average	**1.26E + 03**	4.22E + 03	5.28E + 03	2.19E + 03	3.82E + 03	1.38E + 03	1.91E + 03	1.18E + 04	6.44E + 03
	SD	3.23E + 01	9.46E + 02	1.11E + 03	5.28E + 02	9.31E + 02	5.50E + 01	9.17E + 02	3.70E + 03	2.16E + 03
f_{12}	Best	2.65E + 06	8.42E + 08	1.26E + 09	2.31E + 07	5.69E + 07	5.42E + 06	3.41E + 05	3.42E + 08	8.45E + 08
	Average	1.38E + 07	2.67E + 09	3.88E + 09	1.44E + 08	4.57E + 08	4.21E + 07	**4.20E + 06**	1.16E + 09	1.47E + 09
	SD	9.36E + 06	1.02E + 09	1.14E + 09	7.19E + 07	2.57E + 08	3.56E + 07	6.29E + 06	6.36E + 08	5.55E + 08
f_{13}	Best	1.23E + 04	5.76E + 08	5.87E + 08	2.37E + 05	1.68E + 06	2.06E + 05	1.98E + 04	1.07E + 07	1.64E + 08
	Average	**2.63E + 04**	1.37E + 09	1.51E + 09	2.17E + 06	1.30E + 07	2.08E + 06	1.63E + 07	3.46E + 08	9.58E + 08
	SD	1.45E + 04	5.20E + 08	6.76E + 08	2.97E + 06	1.02E + 07	3.41E + 06	3.52E + 07	3.12E + 08	4.91E + 08
f_{14}	Best	1.22E + 04	1.44E + 05	4.44E + 05	6.09E + 04	1.55E + 05	7.65E + 03	1.16E + 04	9.30E + 04	9.18E + 03
	Average	2.27E + 05	1.04E + 06	1.25E + 06	1.13E + 06	2.21E + 06	2.57E + 05	**2.11E + 05**	2.31E + 06	7.30E + 05
	SD	1.87E + 05	7.10E + 05	7.27E + 05	9.47E + 05	2.27E + 06	2.61E + 05	1.74E + 05	1.78E + 06	6.25E + 05
f_{15}	Best	7.28E + 03	5.83E + 06	6.36E + 06	3.22E + 04	3.87E + 04	3.59E + 04	2.71E + 04	1.71E + 05	2.51E + 05
	Average	**1.42E + 04**	4.39E + 07	2.91E + 07	2.87E + 05	6.37E + 06	2.18E + 05	2.52E + 05	1.12E + 07	7.93E + 07
	SD	3.59E + 03	4.01E + 07	2.41E + 07	2.64E + 05	7.07E + 06	2.22E + 05	3.27E + 05	1.91E + 07	5.37E + 07
f_{16}	Best	2.04E + 03	3.93E + 03	3.74E + 03	2.70E + 03	3.19E + 03	2.14E + 03	2.03E + 03	3.95E + 03	3.41E + 03
	Average	**2.84E + 03**	4.44E + 03	4.23E + 03	3.59E + 03	4.33E + 03	2.97E + 03	3.25E + 03	4.49E + 03	4.03E + 03
	SD	3.28E + 02	2.10E + 02	2.45E + 02	4.91E + 02	5.50E + 02	3.48E + 02	6.90E + 02	2.90E + 02	3.11E + 02
f_{17}	Best	1.83E + 03	2.37E + 03	2.29E + 03	1.99E + 03	2.24E + 03	1.89E + 03	1.79E + 03	2.65E + 03	2.40E + 03
	Average	**2.24E + 03**	2.91E + 03	2.84E + 03	2.47E + 03	2.71E + 03	2.33E + 03	2.35E + 03	3.00E + 03	2.80E + 03
	SD	2.22E + 02	1.92E + 02	1.61E + 02	2.68E + 02	3.23E + 02	2.04E + 02	3.89E + 02	2.23E + 02	2.05E + 02
f_{18}	Best	5.21E + 04	3.98E + 06	1.40E + 06	4.82E + 05	2.07E + 05	1.49E + 05	2.03E + 05	5.71E + 05	9.28E + 05
	Average	**6.11E + 05**	1.53E + 07	1.25E + 07	5.32E + 06	1.03E + 07	3.17E + 06	2.24E + 06	2.52E + 07	8.03E + 06
	SD	7.60E + 05	7.94E + 06	8.79E + 06	5.31E + 06	1.23E + 07	2.56E + 06	1.87E + 06	1.52E + 07	4.97E + 06
f_{19}	Best	1.53E + 04	3.44E + 07	1.28E + 07	2.22E + 05	4.04E + 05	4.41E + 04	2.97E + 05	4.11E + 05	2.52E + 06
	Average	**4.43E + 05**	1.12E + 08	7.79E + 07	1.64E + 06	1.21E + 07	1.01E + 06	1.24E + 06	2.28E + 07	9.84E + 07
	SD	3.45E + 05	6.03E + 07	5.14E + 07	1.44E + 06	1.41E + 07	8.83E + 05	1.00E + 06	4.25E + 07	8.57E + 07
f_{20}	Best	2.27E+03	2.71E+03	2.65E+03	2.40E+03	2.44E+03	2.27E+03	2.53E+03	2.96E+03	2.63E+03
	Average	**2.56E+03**	2.95E+03	2.99E+03	2.76E+03	2.81E+03	2.62E+03	3.15E+03	3.26E+03	2.87E+03
	SD	1.70E+02	1.36E+02	1.52E+02	1.85E+02	1.94E+02	1.86E+02	3.46E+02	1.64E+02	9.18E+01
f_{21}	Best	2.40E+03	2.57E+03	2.57E+03	2.49E+03	2.56E+03	2.42E+03	2.33E+03	2.57E+03	2.53E+03
	Average	**2.44E+03**	2.62E+03	2.62E+03	2.56E+03	2.64E+03	2.49E+03	2.59E+03	2.62E+03	2.60E+03
	SD	2.52E+01	2.45E+01	2.80E+01	4.40E+01	5.41E+01	5.34E+01	2.67E+02	2.59E+01	2.39E+01
f_{22}	Best	2.30E+03	4.08E+03	4.96E+03	2.55E+03	3.49E+03	2.33E+03	6.23E+03	3.31E+03	4.30E+03
	Average	**3.31E+03**	5.39E+03	6.48E+03	5.68E+03	7.74E+03	3.33E+03	8.15E+03	5.31E+03	7.40E+03
	SD	1.86E+03	1.23E+03	1.08E+03	3.15E+03	1.86E+03	1.97E+03	1.32E+03	2.11E+03	1.83E+03
f_{23}	Best	2.74E+03	3.02E+03	3.01E+03	2.87E+03	2.95E+03	2.76E+03	2.94E+03	2.99E+03	2.96E+03
	Average	**2.80E+03**	3.09E+03	3.09E+03	2.98E+03	3.19E+03	2.85E+03	4.22E+03	3.10E+03	3.09E+03
	SD	2.95E+01	3.74E+01	4.62E+01	7.05E+01	1.16E+02	5.76E+01	5.82E+02	6.85E+01	4.95E+01
f_{24}	Best	2.90E+03	3.19E+03	3.18E+03	3.04E+03	3.07E+03	2.94E+03	3.07E+03	3.12E+03	3.09E+03
	Average	**2.98E+03**	3.25E+03	3.25E+03	3.13E+03	3.28E+03	2.99E+03	3.12E+03	3.24E+03	3.25E+03
	SD	4.61E+01	4.25E+01	3.36E+01	6.49E+01	9.54E+01	3.23E+01	2.04E+01	6.14E+01	4.78E+01
f_{25}	Best	2.89E+03	3.30E+03	3.35E+03	2.99E+03	3.10E+03	2.90E+03	2.88E+03	3.34E+03	3.44E+03
	Average	2.93E+03	3.70E+03	3.81E+03	3.08E+03	3.20E+03	2.95E+03	**2.89E+03**	3.72E+03	3.77E+03

(*continued on next page*)

Table 10 (*continued*)

		RUN	CGSCA	SCADE	BMWOA	BWOA	OBLGWO	CMAES	GL25	CLPSO
	SD	2.67E+01	2.36E+02	2.47E+02	5.70E+01	7.47E+01	2.82E+01	6.37E+00	2.51E+02	2.22E+02
f_{26}	Best	2.80E+03	6.36E+03	7.36E+03	3.74E+03	4.71E+03	3.56E+03	2.80E+03	7.35E+03	6.52E+03
	Average	**4.50E+03**	8.02E+03	8.21E+03	6.82E+03	8.33E+03	5.73E+03	5.39E+03	8.47E+03	7.92E+03
	SD	1.27E+03	5.81E+02	3.96E+02	1.22E+03	1.12E+03	7.41E+02	1.84E+03	5.37E+02	5.68E+02
f_{27}	Best	3.25E+03	3.41E+03	3.39E+03	3.25E+03	3.33E+03	3.22E+03	3.35E+03	3.51E+03	3.43E+03
	Average	3.31E+03	3.52E+03	3.57E+03	3.33E+03	3.47E+03	**3.25E+03**	3.51E+03	3.66E+03	3.58E+03
	SD	3.57E+01	6.53E+01	8.54E+01	6.37E+01	1.52E+02	1.57E+01	3.47E+02	1.01E+02	7.67E+01
f_{28}	Best	3.23E+03	4.08E+03	4.48E+03	3.39E+03	3.50E+03	3.27E+03	3.19E+03	3.95E+03	4.25E+03
	Average	3.28E+03	4.76E+03	5.03E+03	3.50E+03	3.82E+03	3.35E+03	**3.23E+03**	4.88E+03	4.95E+03
	SD	2.06E+01	4.47E+02	3.53E+02	7.26E+01	2.00E+02	3.69E+01	3.00E+01	4.09E+02	4.03E+02
f_{29}	Best	3.69E+03	4.67E+03	5.18E+03	4.25E+03	4.31E+03	3.84E+03	3.42E+03	4.91E+03	4.54E+03
	Average	4.24E+03	5.29E+03	5.67E+03	5.00E+03	5.45E+03	4.28E+03	**3.76E+03**	5.56E+03	5.13E+03
	SD	2.74E+02	3.17E+02	3.15E+02	5.16E+02	6.21E+02	3.41E+02	2.50E+02	3.28E+02	3.12E+02
f_{30}	Best	3.55E+05	6.81E+07	6.71E+07	1.00E+06	6.87E+06	7.09E+05	7.94E+05	1.76E+07	1.74E+07
	Average	3.99E+06	2.19E+08	2.01E+08	8.83E+06	5.03E+07	6.50E+06	**3.18E+06**	5.03E+07	7.24E+07
	SD	2.71E+06	8.81E+07	8.13E+07	4.83E+06	4.07E+07	4.35E+06	2.42E+06	3.73E+07	4.27E+07

Table 11
Average ranks of RUN and eight advanced optimizers based on the Friedman test.

Algorithm	Friedman ranking	Rank
RUN	**1.33**	**1**
CGSCA	6.53	7
SCADE	7.40	9
BMWOA	4.00	3
BWOA	5.70	5
OBLGWO	2.23	2
CMAES	4.43	4
GL25	7.17	8
CLPSO	6.20	6

(MMA) (Chickermane & Gea, 1996), generalized convex approximation (GCA I) (Chickermane & Gea, 1996), GCA II (Chickermane & Gea, 1996), and SOS (Cheng & Prayogo, 2014). As shown in Table 18, RUN provided more promising results than the five other optimizers, which confirmed the RUN algorithm's high efficiency in approximating the global best solution for this problem. Also, the optimal variables calculated by all the six optimizers are listed in Table 19.

7. Conclusions and future directions

In this study, a novel optimization algorithm, RUN, was developed to solve various optimization problems. The RUN algorithm was developed based on the RK method used as a search engine to explore the best solution in the search space. The RUN algorithm's search mechanism was formulated to effectively implement and balance the exploration and exploitation phases. Also, the enhanced solution quality (ESQ) was proposed and incorporated into RUN to improve the quality of solutions, escape from local optima, and increase the convergence speed. 20 test functions, including unimodal, multimodal, and hybrid, were utilized to assess the efficiency of the RUN algorithm.

The RUN's superior efficiency on the unimodal and multimodal test functions demonstrated its excellent exploitation and exploration abilities, which can be attributed to utilizing the local and global terms in the RUN search mechanism and the ESQ operator. Moreover, the optimal results for the hybrid and composite test functions showed that RUN effectively facilitated the transition from global search (i.e., exploration) to local search (i.e., exploitation) by utilizing the adaptive parameters. The results from this study indicated that RUN was able to explore wondrous solutions compared with other state-of-the-art optimizers.

To evaluate the efficiencies of the RUN algorithm, comparisons were made with five other optimizers using two well-known ranking tests (i.e., the Friedman and Quade tests). The findings revealed that RUN provided very competitive results and outperformed the other optimizers for most test functions.

In addition, the efficiency of RUN algorithm was assessed by utilizing the IEEE CEC2017 test functions and it was compared with eight advanced algorithms. The results demonstrate that RUN can guarantee the efficiency of global search while obtaining superior local search, thus retaining an excellent balance between local and global search capabilities, which indicates the superior efficiency of the proposed optimizer in comparison with the advanced optimizers. Moreover, RUN was compared with other existing optimizers in solving four engineering design problems, showing that RUN presented a better performance in optimizing these complex real-world problems than the other optimization algorithms.

This study developed a new optimizer RUN to be implemented and formulated with specific exploration and exploitation strategies. Despite the promising findings, it is recommended for future studies to use other well-known operators, such as the crossover operator, mutation operator, opposite-based learning method, and levy-flight walks. A chaotic map (CMs) may also be considered when the EQS is used in each iteration. In addition, further improvement can be made by developing the multiobjective and binary versions of RUN for solving multiobjective and discrete optimization problems. Finally, other RK methods, such as the fourth-order RK contraharmonic mean method, can be considered in the RUN algorithm to enhance its efficiency.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

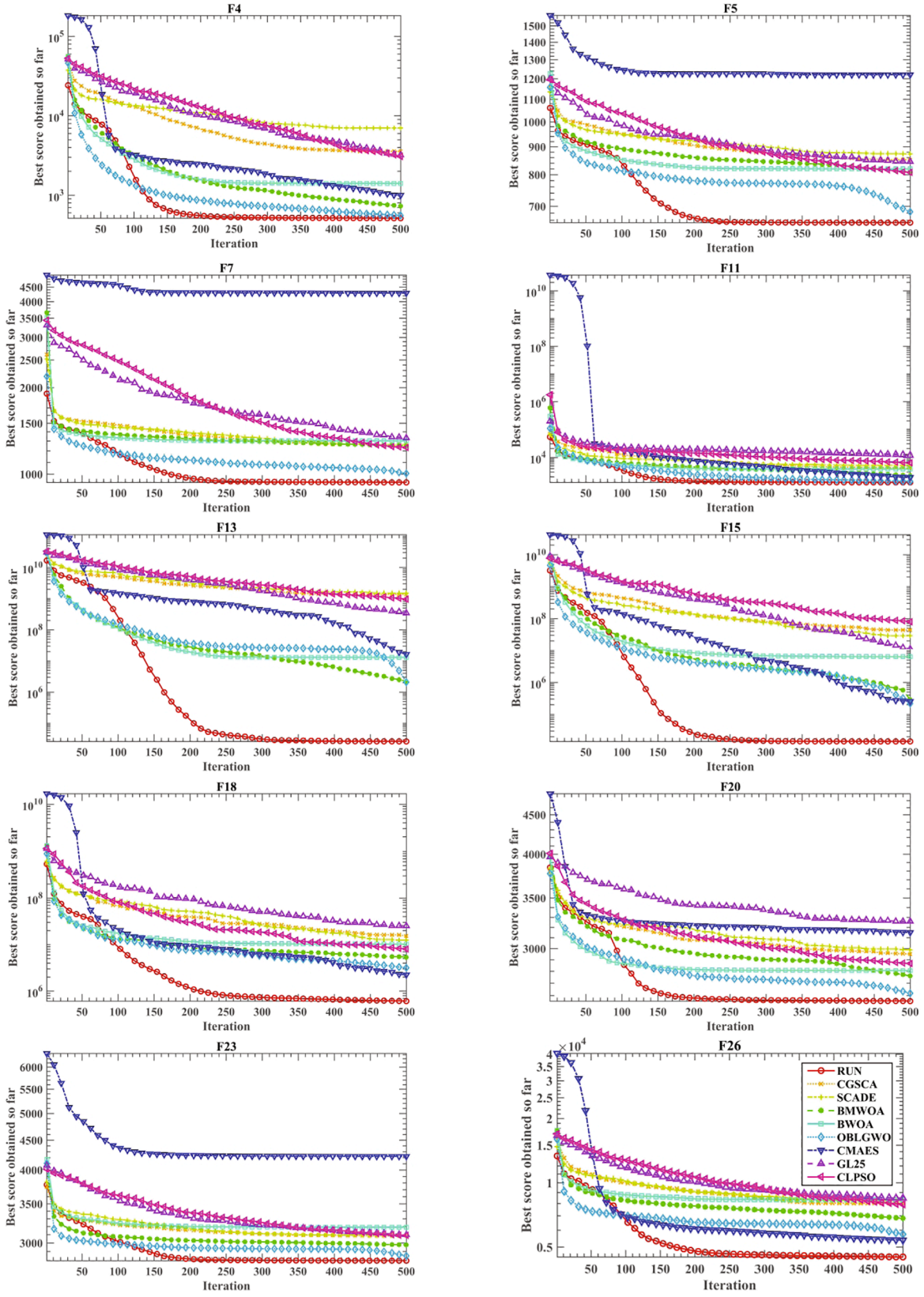

Fig. 8. Convergence graphs of RUN and eight other algorithms for the selected CEC 2017 benchmark functions.

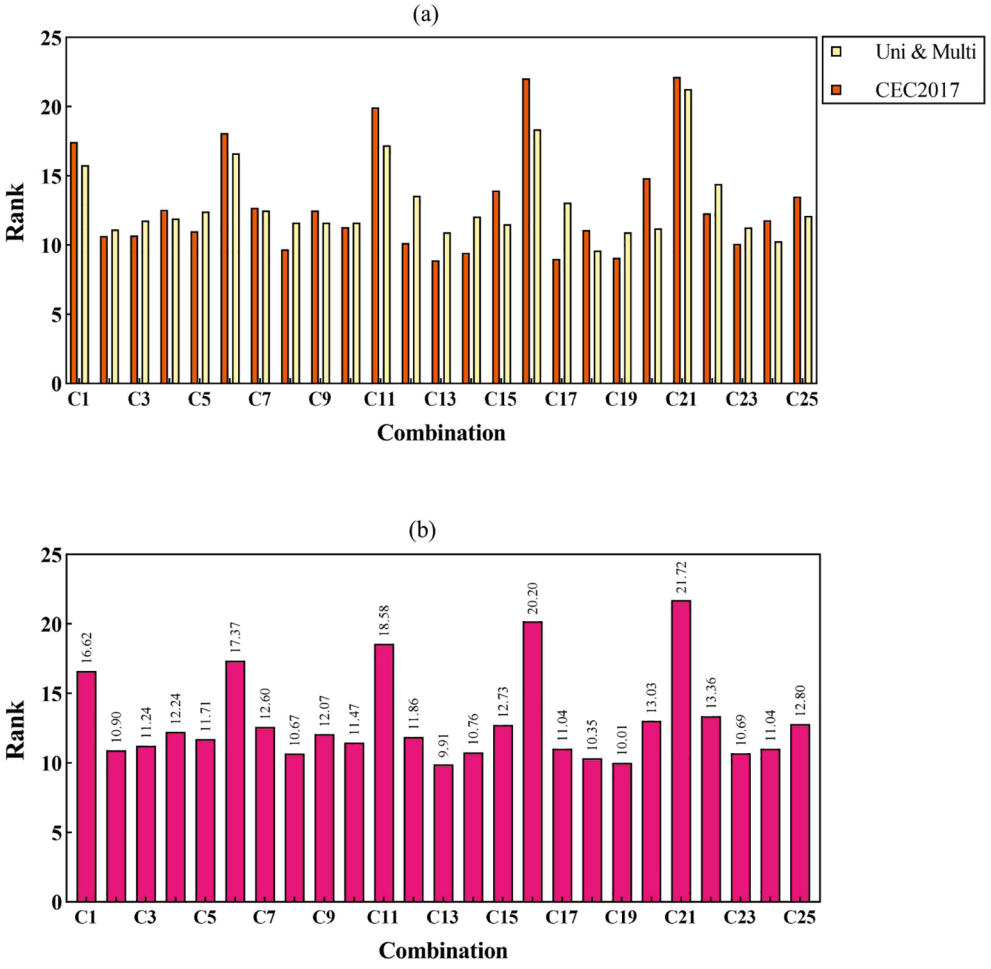

Fig. 9. Sensitivity analysis of RUN, (a) ranks of uni- and multi-modal test functions and CEC-2017 (b) average ranks of all combinations.

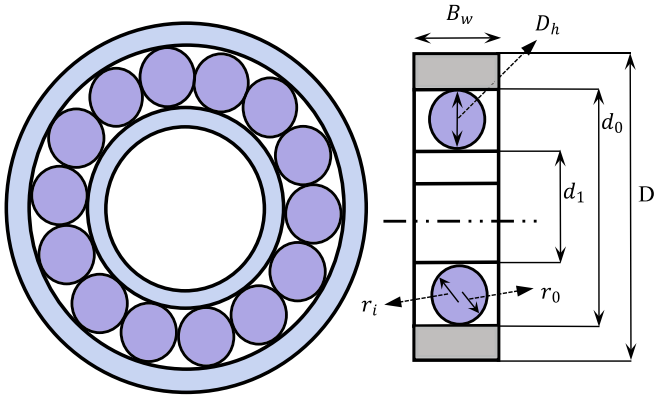

Fig. 10. Rolling element bearing design problem.

Table 12
Statistical results from RUN, TLBO, GA, PVS, and HHO for the rolling element bearing design problem.

	RUN	GA (Gupta et al., 2007)	TLBO (Rao et al., 2011)	PVS (Savsani & Savsani, 2016)	HHO (Heidari et al., 2019)
Best	83680.47	81843.30	81859.74	81859.59	83011.88
Mean	82025.24	NA*	81438.99	80803.57	NA
SD	977.95	NA	NA	NA	NA

*NA: Not Available.

Table 13
Comparison of the results from RUN, TLBO, GA, PVS, and HHO for the rolling element bearing design problem.

Variables	RUN	TLBO (Rao et al., 2011)	GA (Gupta et al., 2007)	PVS (Savsani & Savsani, 2016)	HHO (Heidari et al., 2019)
D_b	21.59796	21.42559	21.42300	21.42559	21.0000
D_m	125.2142	125.7191	125.71710	125.71906	125.0000
f_i	0.51500	0.51500	0.51500	0.51500	0.51500
f_0	0.51500	0.51500	0.51500	0.51500	0.51500
Z	11.4024	11.0000	11.0000	11.0000	11.0920
K_{Dmin}	0.40059	0.42426	0.41590	0.40043	0.4000
K_{Dmax}	0.61467	0.63394	0.65100	0.68016	0.6000
ε	0.30530	0.30000	0.30004	0.30000	0.3000
e	0.02000	0.06885	0.02230	0.07999	0.0504
ζ	0.63665	0.79994	0.75100	0.70000	0.6000

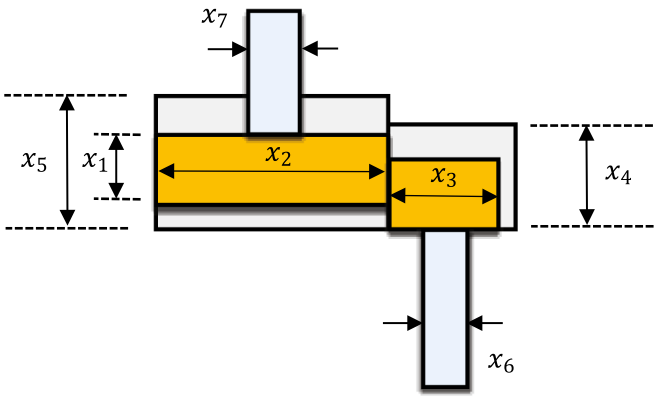

Fig. 11. Speed reducer design problem.

Table 14
Statistical results from RUN, CS, HGSO, GWO, and WOA for the speed reducer design problem.

	RUN	CS (Gandomi et al., 2013)	HGSO (Hashim et al., 2019)	GWO (Hashim et al., 2019)	WOA (Hashim et al., 2019)
Best	2996.348	NA	2996.4	2998.545	2998.134
Mean	2996.348	3007.2	2996.9	2998.832	2998.445
SD	7.63E-09	4.96E + 00	4.39E-05	1.86E-06	1.94E-06

Table 15
Comparison of the results from RUN, CS, HGSO, GWO, and WOA for the speed reducer design problem.

Variables	RUN	CS (Gandomi et al., 2013)	HGSO (Hashim et al., 2019)	GWO (Hashim et al., 2019)	WOA (Hashim et al., 2019)
x_1	3.5001	3.5015	3.4970	3.5000	3.4210
x_2	0.7000	0.7000	0.7100	0.7000	0.7000
x_3	17.000	17.000	17.020	17.000	17.000
x_4	7.0000	7.6050	7.6700	7.3000	7.3000
x_5	7.8000	7.8181	7.8100	7.8000	7.8000
x_6	3.3500	3.3520	3.3600	2.9000	2.9000
x_7	5.2900	5.2875	5.2850	2.9000	5.0000
Fitness	2996.73	3000.98	2997.10	2998.83	2998.40

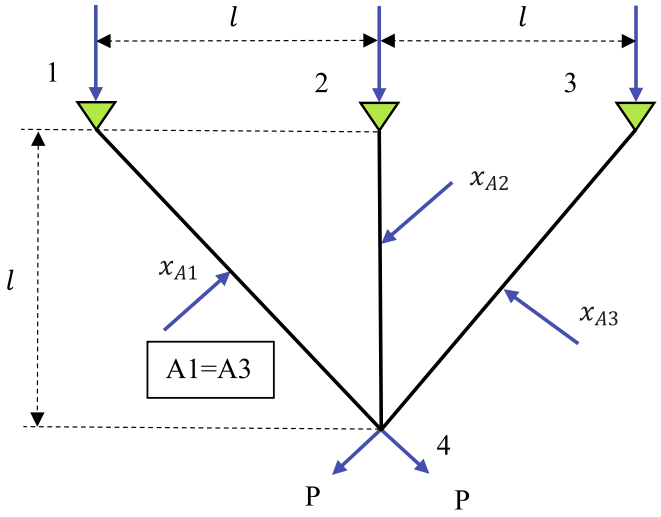

Fig. 12. Three-bar truss problem.

Table 16
Comparison of statistical results of RUN with literature for the three-bar truss problem.

	RUN	MVO (S. Mirjalili et al., 2016)	GOA (S. Z. Mirjalili et al., 2018)	MFO (S. Mirjalili, 2015b)	MBA (Sadollah et al., 2013)	CS (Gandomi et al., 2013)	HHO (Heidari et al., 2019)
Best	263.8958	263.8958	263.8958	263.8955	263.8958	263.9715	263.8958
Mean	263.89768	NA	NA	NA	263.897996	264.0669	NA
SD	2.30E-03	NA	NA	NA	3.93E-03	9.00E-05	NA

Table 17
Best solutions achieved by the seven algorithms for the three-bar truss problem.

Algorithm	x_1	x_2
RUN	0.788679110	0.408237045
MVO (S. Mirjalili et al., 2016)	0.78860276	0.408453070
GOA (S. Z. Mirjalili et al., 2018)	0.78889755	0.40761957
MFO (S. Mirjalili, 2015b)	0.78824477	0.40946690
MBA (Sadollah et al., 2013)	0.7885650	0.4085597
CS (Gandomi et al., 2013)	0.78867	0.40902
HHO (Heidari et al., 2019)	0.7886628	0.4082313

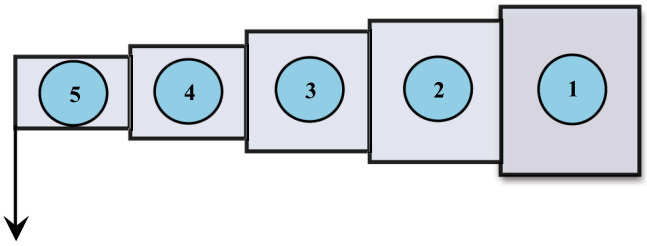

Fig. 13. Cantilever beam problem.

Table 18
Statistical results of RUN, CS, SOS, MMA, GCA I, and GCA II for the cantilever beam problem.

	RUN	CS (Gandomi et al., 2013)	SOS (Cheng & Prayogo, 2014)	MMA (Chickermane & Gea, 1996)	GCA I (Chickermane & Gea, 1996)	GCA II (Chickermane & Gea, 1996)
Best	1.3399563	1.33999	1.33996	1.34000	1.34000	1.34000
Mean	1.3399604	NA	1.33997	NA	NA	NA
SD	4.68E-06	NA	1.10E-05	NA	NA	NA

Table 19
Optimal variables obtained by the RUN, CS, SOS, MMA, GCA I, and GCA II algorithms for the cantilever beam problem.

Algorithm	x_1	x_2	x_3	x_4	x_5
RUN	6.0049	5.3190	4.4868	3.5033	2.1595
CS (Gandomi et al., 2013)	6.0089	5.3049	4.5023	3.5077	2.1504
SOS (Cheng & Prayogo, 2014)	6.0187	5.3034	4.4958	3.4989	2.1556
MMA (Chickermane & Gea, 1996)	6.0100	5.3000	4.4900	3.4900	2.1500
GCA I (Chickermane & Gea, 1996)	6.0100	5.3040	4.4900	3.4980	2.1500
GCA II (Chickermane & Gea, 1996)	6.0100	5.3000	4.4900	3.4900	2.1500

Appendix A

I- Rolling element bearing design problem

$$Maximize Z = \begin{cases} f_c \times Z^{2/3} \times {D_b}^{1.8} if D_b \le 25.4 \\ 3.647 \times f_c \times Z^{2/3} \times {D_b}^{1.4} otherwise \end{cases}$$

Subject to:

$$g_1(\vec{x}) = \frac{\varphi_0}{2sin^{-1}\left(\frac{D_b}{D_m}\right)} - Z + 1 \le 1,$$

$$g_2(\vec{x}) = 2D_b - K_{Dmin}(D - d) \ge 0,$$

$$g_3(\vec{x}) = K_{Dmax}(D-d) - 2D_b \geq 0,$$

$$g_4(\vec{x}) = \zeta B_w - D_b \leq 0,$$

$$g_5(\vec{x}) = D_m - 0.5 \times (D+d) \geq 0,$$

$$g_6(\vec{x}) = (0.5+e) \times (D+d) - D_m \geq 0,$$

$$g_7(\vec{x}) = 0.5(D - D_m - D_b) - \varepsilon D_b \geq 0,$$

$$g_8(\vec{x}) = f_i \geq 0.515,$$

$$g_9(\vec{x}) = f_0 \geq 0.515,$$

in which

$$f_c = 37.91\left[1 + \left(1.04\left(\frac{1+\gamma}{1-\gamma}\right)^{1.72}\left(\frac{f_i(2f_0-1)}{f_0(2f_i-1)}\right)^{0.41}\right)^{\frac{10}{3}}\right]^{-0.3}$$

$$\times\left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{\frac{1}{3}}}\right] \times \left[\frac{2f_i}{2f_i-1}\right]^{0.41}$$

$$x = \left[\left\{\frac{(D-d)}{2} - 3\left(\frac{T}{4}\right)\right\}^2 + \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}^2 - \left\{\frac{d}{2} + \frac{T}{4}\right\}^2\right]$$

$$y = 2\left\{\frac{(D-d)}{2} - 3\left(\frac{T}{4}\right)\right\} \times \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}$$

$$\varphi_0 = 2\pi - cos^{-1}\left(\frac{x}{y}\right)$$

$$B_w = 30,\ D = 160,\ d = 90,\ r_i\ = r_0 = 11.033$$

$$\gamma = \frac{D_b}{D_m},\ f_i = \frac{r_i}{D_b},\ f_i = \frac{r_0}{D_b}, T = D - d - 2D_b$$

$$0.15(D-d) \leq D_b \leq 0.45(D-d),\ 4 \leq Z \leq 50,\ 0.515 \leq f_i,\ f_0 \leq 0.60$$

$$0.4 \leq K_{Dmin} \leq 0.5,\ 0.6 \leq K_{Dmax} \leq 0.7,\ 0.3 \leq \varepsilon \leq 0.4,\ 0.02 \leq e \leq 0.1$$

$$0.6 \leq \zeta \leq 0.85$$

II- Speed reducer problem

Minimize

$$Fitness = 0.7854x_1x_2^2 \times \left(3.3333 \times x_3^2 + 14.9334 \times x_3 - 43.0934\right) - 1.508 \times x_1 \times \left(x_6^2 + x_7^2\right) + 7.4777 \times \left(x_6^3 + x_7^3\right) + 0.7854 \times \left(x_4x_6^2 + x_5x_7^2\right)$$

Subject to

$$g_1(x) = \frac{27}{(x_1x_2^2 \times x_3)} - 1 \leqslant 0$$

$$g_2(x) = \frac{397.5}{(x_1x_2^2 \times x_3^2)} - 1 \leqslant 0$$

$$g_3(x) = \frac{1.93x_4^3}{(x_2x_3 \times x_6^4)} - 1 \leqslant 0$$

$$g_4(x) = \frac{1.93x_5^3}{(x_2x_3 \times x_7^4)} - 1 \leqslant 0$$

$$g_5(x) = \frac{1}{(110 \times x_6^3)} \times \sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} - 1 \leqslant 0$$

$$g_6(x) = \frac{1}{(85 \times x_7^3)} \times \sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \leqslant 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \leqslant 0$$

$$g_8(x) = 5\frac{x_2}{x_1} - 1 \leqslant 0$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \leqslant 0$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leqslant 0$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leqslant 0$$

III- Three-bar truss problem

Minimize $Fitness(\vec{x}) = (2\sqrt{2}x_1 + x_2) \times l$,

Subject to: $g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0$,

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

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

$$0 \leq x_1, x_2 \leq 1, l = 100cm,\ P = 2\frac{kN}{cm^2},\ \sigma = 2\frac{kN}{cm^2}$$

IV- Cantilever beam problem

$$MinimizeFitness = 0.0624 \times (x_1 + x_2 + x_3 + x_4 + x_5)$$

Subject to:

$g(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$

Variable ranges

$$0.01 \leq x_1, x_2, x_3,\ x_4,\ x_5 \leq 100$$

References

Ahmadianfar, I., Bozorg-Haddad, O., & Chu, X. (2020). Gradient-based optimizer: A new Metaheuristic optimization algorithm. *Information Sciences, 540*, 131–159.

Ahmadianfar, I., Khajeh, Z., Asghari-Pari, S.-A., & Chu, X. (2019). Developing optimal policies for reservoir systems using a multi-strategy optimization algorithm. *Applied Soft Computing, 80*, 888–903.

Ahmadianfar, I., Kheyrandish, A., Jamei, M., & Gharabaghi, B. (2020). Optimizing operating rules for multi-reservoir hydropower generation systems: An adaptive hybrid differential evolution algorithm. *Renewable Energy*.

Baykasoğlu, A., & Ozsoydan, F. B. (2017). Evolutionary and population-based methods versus constructive search strategies in dynamic combinatorial optimization. *Information Sciences, 420*, 159–183.

Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution strategies–A comprehensive introduction. *Natural Computing, 1*, 3–52.

Cao, B., Fan, S., Zhao, J., Yang, P., Muhammad, K., & Tanveer, M. (2020). Quantum-enhanced multiobjective large-scale optimization via parallelism. *Swarm and Evolutionary Computation, 57*, Article 100697.

Cao, B., Wang, X., Zhang, W., Song, H., & Lv, Z. (2020). A many-objective optimization model of industrial internet of things based on private Blockchain. *IEEE Network, 34*, 78–83.

Cao, B., Zhao, J., Yang, P., Gu, Y., Muhammad, K., Rodrigues, J. J., & de Albuquerque, V. H. C. (2019). Multiobjective 3-D topology optimization of next-generation wireless data center network. *IEEE Transactions on Industrial Informatics, 16*, 3597–3605.

Chen, Y., He, L., Guan, Y., Lu, H., & Li, J. (2017). Life cycle assessment of greenhouse gas emissions and water-energy optimization for shale gas supply chain planning based on multi-level approach: Case study in Barnett, Marcellus, Fayetteville, and Haynesville shales. *Energy Conversion and Management, 134*, 382–398.

Chen, H., Xu, Y., Wang, M., & Zhao, X. (2019). A balanced whale optimization algorithm for constrained engineering design problems. *Applied Mathematical Modelling, 71*, 45–59.

Chen, Y., Zheng, W., Li, W., & Huang, Y. (2021). Large group Activity security risk assessment and risk early warning based on random forest algorithm. *Pattern Recognition Letters, 144*, 1–5.

Cheng, M.-Y., & Prayogo, D. (2014). Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers & Structures, 139*, 98–112.

Chickermane, H., & Gea, H. (1996). Structural optimization using a new local approximation method. *International Journal for Numerical Methods in Engineering, 39*, 829–846.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1*, 3–18.

Doğan, B., & Ölmez, T. (2015). A new metaheuristic for numerical function optimization: Vortex Search algorithm. *Information Sciences, 293*, 125–145.

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) (Vol. 2, pp. 1470-1477): IEEE.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science (Vol. 1, pp. 39-43): New York, NY.

England, R. (1969). Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations. *The Computer Journal, 12*, 166–170.

Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures, 110*, 151–166.

Formato, R. A. (2007). Central force optimization. *Progress in Electromagnetics Research, 77*, 425–491.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*, 17–35.

García-Martínez, C., Lozano, M., Herrera, F., Molina, D., & Sánchez, A. M. (2008). Global and local real-coded genetic algorithms based on parent-centric crossover operators. *European Journal of Operational Research, 185*, 1088–1113.

Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093–2229). Springer.

Gupta, S., Tiwari, R., & Nair, S. B. (2007). Multi-objective design optimisation of rolling bearings using genetic algorithms. *Mechanism and Machine Theory, 42*, 1418–1443.

Hansen, N., Müller, S. D., & Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation, 11*, 1–18.

Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W., & Mirjalili, S. (2019). Henry gas solubility optimization: A novel physics-based algorithm. *Future Generation Computer Systems, 101*, 646–667.

Heidari, A. A., Abbaspour, R. A., & Chen, H. (2019). Efficient boosted grey wolf optimizers for global search and kernel extreme learning machine training. *Applied Soft Computing, 81*, Article 105521.

Heidari, A. A., Aljarah, I., Faris, H., Chen, H., Luo, J., & Mirjalili, S. (2019). An enhanced associative learning-based exploratory whale optimizer for global optimization. *Neural Computing and Applications*, 1–27.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97*, 849–872.
Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. U Michigan Press.
Hosseini, H. S. (2007). Problem solving by intelligent water drops. In *2007 IEEE congress on evolutionary computation* (pp. 3226–3231). IEEE.
Hu, J., Chen, H., Heidari, A. A., Wang, M., Zhang, X., Chen, Y., & Pan, Z. (2021). Orthogonal learning covariance matrix for defects of grey wolf optimizer: insights, balance, diversity, and feature selection. *Knowledge-Based Systems, 213*, 106684.
Huang, Q., Zhang, K., Song, J., Zhang, Y., & Shi, J. (2019). Adaptive differential evolution with a Lagrange interpolation argument algorithm. *Information Sciences, 472*, 180–202.
Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*, 459–471.
Kaveh, A., & Bakhshpoori, T. (2016). Water evaporation optimization: A novel physically inspired optimization algorithm. *Computers & Structures, 167*, 69–85.
Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.
Koza, J. R. (1994). Genetic programming II: Automatic discovery of reusable subprograms. Cambridge, MA, USA, 13, 32.
Kumar, N., Hussain, I., Singh, B., & Panigrahi, B. K. (2017). Single sensor-based MPPT of partially shaded PV system for battery charging by using cauchy and gaussian sine cosine optimization. *IEEE Transactions on Energy Conversion, 32*, 983–992.
Kutta, W. (1901). Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys., 46*, 435–453.
Li, S., Chen, H., Wang, M., Heidari, A. A., & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems, 111*, 300–323.
Li, T., Xu, M., Zhu, C., Yang, R., Wang, Z., & Guan, Z. (2019). A deep learning approach for multi-frame in-loop filter of HEVC. *IEEE Transactions on Image Processing, 28*, 5663–5678.
Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, 10*, 281–295.
Liu, S., Yu, W., Chan, F. T. S., & Niu, B. A variable weight-based hybrid approach for multi-attribute group decision making under interval-valued intuitionistic fuzzy sets. International Journal of Intelligent Systems, n/a.
Liu, Y., Zhang, B., Feng, Y., Lv, X., Ji, D., Niu, Z., & Fan, Y. (2020). Development of 340-GHz Transceiver Front End Based on GaAs Monolithic Integration Technology for THz Active Imaging Array. *Applied Sciences, 10*(21), 7924.
Lones, M. A. (2020). Mitigating metaphors: A comprehensible guide to recent nature-inspired algorithms. *SN Computer Science, 1*, 49.
Luo, Q., Zhang, S., & Zhou, Y. (2017). Stochastic Fractal Search Algorithm for Template Matching with Lateral Inhibition. Scientific Programming, 2017.
Ma, X., Zhang, K., Zhang, L., Yao, C., Yao, J., Wang, H., & Yan, Y. (2021). Data-driven niching differential evolution with adaptive parameters control for history matching and uncertainty quantification. *SPE Journal, 26*(02), 993–1010.
Mezura-Montes, E., & Coello, C. A. C. (2005). Useful infeasible solutions in engineering optimization with evolutionary algorithms. In *Mexican International Conference on Artificial Intelligence* (pp. 652–662). Springer.
Mirjalili, S. (2015a). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.
Mirjalili, S. (2015b). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.
Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software, 95*, 51–67.
Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*, 495–513.
Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
Mirjalili, S. Z., Mirjalili, S., Saremi, S., Faris, H., & Aljarah, I. (2018). Grasshopper optimization algorithm for multi-objective optimization problems. *Applied Intelligence, 48*, 805–820.
Mousavi, A. A., Zhang, C., Masri, S. F., & Gholipour, G. (2020). Structural Damage Localization and Quantification Based on a CEEMDAN Hilbert Transform Neural Network Approach: A Model Steel Truss Bridge Case Study. *Sensors, 20*, 1271.
Nenavath, H., & Jatoth, R. K. (2018). Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking. *Applied Soft Computing, 62*, 1019–1043.
Niu, Z., Zhang, B., Wang, J., Liu, K., Chen, Z., Yang, K., & Liu, Y. (2020). The research on 220GHz multicarrier high-speed communication system. *China Communications, 17*(3), 131–139.
Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.
Patil, P., & Verma, U. (2006). *Numerical Computational Methods*. Oxford UK: Alpha Science International Ltd.
Qu, S., Xu, W., Zhao, J., & Zhang, H. (2021). Design and Implementation of a Fast Sliding-Mode Speed Controller with Disturbance Compensation for SPMSM Syste. *IEEE Transactions on Transportation Electrification*. https://doi.org/10.1109/TTE.2021.3060102
Rao, R. V., Savsani, V. J., & Vakharia, D. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43*, 303–315.
Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences, 179*, 2232–2248.
Runge, C. (1895). Über die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen, 46*, 167–178.
Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing, 13*, 2592–2612.
Saka, M. P., Hasançebi, O., & Geem, Z. W. (2016). Metaheuristics in structural optimization and discussions on harmony search algorithm. *Swarm and Evolutionary Computation, 28*, 88–97.
Salcedo-Sanz, S. (2016). Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. *Physics Reports, 655*, 1–70.
Savsani, P., & Savsani, V. (2016). Passing vehicle search (PVS): A novel metaheuristic algorithm. *Applied Mathematical Modelling, 40*, 3951–3978.
Shi, J., Lu, Y., & Zhang, J. (2019). Approximation attacks on strong PUFs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39*(10), 2138–2151.
Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*, 702–713.
Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research, 22*, 3–18.
Storn, R., & Price, K. (1995). *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. Berkeley: ICSI.
Tian, M., & Gao, X. (2017). An improved differential evolution with information intercrossing and sharing mechanism for numerical optimization. *Swarm and Evolutionary Computation*.
Tsamardinos, I., Brown, L. E., & Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning, 65*, 31–78.
Wu, G. (2016). Across neighborhood search for numerical optimization. *Information Sciences, 329*, 597–618.
Wu, G., Pedrycz, W., Suganthan, P. N., & Mallipeddi, R. (2015). A variable reduction strategy for evolutionary algorithms handling equality constraints. *Applied Soft Computing, 37*, 774–786.
Wu, C., Wu, P., Wang, J., Jiang, R., Chen, M., & Wang, X. (2020). Critical review of data-driven decision-making in bridge operation and maintenance. *Structure and Infrastructure Engineering*, 1–24.
Xue, X., Zhang, K., Tan, K. C., Feng, L., Wang, J., Chen, G., … Yao, J. (2021). Affine Transformation-Enhanced Multifactorial Optimization for Heterogeneous Problems. *IEEE Transactions on Cybernetics*. https://doi.org/10.1109/TCYB.2020.3036393
Yang, X.-S. (2010a). Firefly algorithm, stochastic test functions and design optimisation. arXiv preprint arXiv:1003.1409.
Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65-74): Springer.
Yang, X.-S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation, 1*, 330–343.
Yang, Y., Chen, H., Heidari, A. A., & Gandomi, A. H. (2021). Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Systems with Applications, 177*, 114864.
Zenggang, X., Zhiwen, T., Xiaowen, C., Xue-min, Z., Kaibin, Z., & Conghuan, Y. (2021). Research on Image Retrieval Algorithm Based on Combination of Color and Shape Features. *J Sign Process Syst, 93*, 139–146. https://doi.org/10.1007/s11265-019-01508-y
Zhang, B., Ji, D., Fang, D., Liang, S., Fan, Y., & Chen, X. (2019). A Novel 220-GHz GaN Diode On-Chip Tripler With High Driven Power. *IEEE Electron Device Letters, 40*(5), 780–783. https://doi.org/10.1109/LED.2019.2903430
Zhang, B., Niu, Z., Wang, J., Ji, D., Zhou, T., Liu, Y., & Fan, Y. (2020). Four-hundred gigahertz broadband multi-branch waveguide coupler. *IET Microwaves, Antennas & Propagation, 14*(11), 1175–1179.
Zhang, C. W., Ou, J. P., & Zhang, J. Q. (2006). Parameter optimization and analysis of a vehicle suspension system controlled by magnetorheological fluid dampers. *Structural Control and Health Monitoring, 13*, 885–896.
Zhao, W., Wang, L., & Zhang, Z. (2019). Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowledge-Based Systems, 163*, 283–304.
Zhang, K., Zhang, J., Ma, X., Yao, C., Zhang, L., Yang, Y., … Zhao, H. (2021). History matching of naturally fractured reservoirs using a deep sparse autoencoder. *SPE Journal*, 1–22.
Zhang, J., Zhou, Y., & Luo, Q. (2018). An improved sine cosine water wave optimization algorithm for global optimization. *Journal of Intelligent & Fuzzy Systems, 34*, 2129–2141.
Zheng, L., & Zhang, X. (2017). *Modeling and analysis of modern fluid problems*. Academic Press.