Artificial Intelligence Review (2024) 57:313
https://doi.org/10.1007/s10462-024-10857-5

Improving the Kepler optimization algorithm with chaotic maps: comprehensive performance evaluation and engineering applications

Nawal El Ghouate[1] · Ahmed Bencherqui[1] · Hanaa Mansouri[1] · Ahmed El Maloufy[1] · Mohamed Amine Tahiri[1] · Hicham Karmouni[2] · Mhamed Sayyouri[1] · S. S. Askar[3] · Mohamed Abouhawwash[4,5]

Accepted: 3 July 2024

Abstract
The Kepler Optimisation Algorithm (KOA) is a recently proposed algorithm that is inspired by Kepler's laws to predict the positions and velocities of planets at a given time. However, although promising, KOA can encounter challenges such as convergence to sub-optimal solutions or slow convergence speed. This paper proposes an improvement to KOA by integrating chaotic maps to solve complex engineering problems. The improved algorithm, named Chaotic Kepler Optimization Algorithm (CKOA), is characterized by a better ability to avoid local minima and to reach globally optimal solutions thanks to a dynamic diversification strategy based on chaotic maps. To confirm the effectiveness of the suggested approach, in-depth statistical analyses were carried out using the CEC2020 and CEC2022 benchmarks. These analyses included mean and standard deviation of fitness, convergence curves, Wilcoxon tests, as well as population diversity assessments. The experimental results, which compare CKOA not only to the original KOA but also to eight other recent optimizers, show that the proposed algorithm performs better in terms of convergence speed and solution quality. In addition, CKOA has been successfully tested on three complex engineering problems, confirming its robustness and practical effectiveness. These results make CKOA a powerful optimisation tool in a variety of complex real-world contexts. After final acceptance, the source code will be uploaded to the Github account: nawal.elghouate@usmba.ac.ma.

Keywords Kepler optimizer · Improved Kepler optimizer · Chaotic maps · Optimization · Engineering problems

1 Introduction

Optimization is emerging as a vibrant and active field of study, extending across diverse areas such as electronic circuit design, operational analysis, and biology, while playing a vital role in supporting the growing needs of computer science, artificial intelligence (Zamani and Nadimi-Shahraki 2024), and machine learning (Modibbo et al. 2021;

Extended author information available on the last page of the article

Ahmad and Sirjani 2020; Rajput and Datta 2021). This field is primarily concerned with maximizing efficiency and reducing costs. Metaheuristics, which are general-purpose algorithms developed to tackle optimization problems of high complexity, operate by manipulating one or more potential solutions with the aim of reaching the optimum, or most advantageous, solution (Cuevas et al. 2020; Zamani et al. 2024). Their iterative approach aims to evolve from initial solutions of inferior quality towards the optimum solution (Tezel and Mert 2021), concluding the process once a certain pre-established criterion has been met, such as elapsed time, specific accuracy level, or a number of iterations.

These algorithms are distinguished by their ability to efficiently solve a multitude of real optimization problems (Tahiri et al. 2023; Bencherqui et al. 2022a, 2023), often characterized by their pronounced non-linearity and multimodality (Cao and Wu 1997). They achieve this by judiciously balancing randomization, necessary to examine the solution space exhaustively (exploration phase), and local optimization, aimed at refining solutions within specific regions (exploitation phase) (Faramarzi et al. 2020; Nikolić et al. 2020).

Metaheuristics are classified into two main categories: those based on iterative improvement of a single solution, such as Simulated Annealing (Kirkpatrick et al. 1983) and the Taboo Method (Prajapati et al. 2020), and those that explore a set of solutions in parallel, such as genetic algorithms (Sohail 2023; Chaudhary and Banati 2020), Particle Swarm Optimization (Shami et al. 2022), Duck Swarm Algorithm (Zhang and Wen 2024), Sand Cat Swarm Optimization Algorithm (Yancang et al. 2024), Artificial Locust Swarm Optimization Algorithm (Kesemen et al. 2023) and the Bee Colony Method (Bencherqui et al. 2022b). A second classification of these techniques can be made according to four main categories: evolutionary, swarm-based, human-based and physics-based. Swarm-based algorithms are inspired by the social behaviour of animals and insects, as demonstrated by the Particle Swarm Optimization (PSO) (Shami et al. 2022), Siberian Tiger Optimization (Trojovsky et al. 2022), Orca Optimization Algorithm (Golilarz et al. 2020b) and Remora optimization algorithm (Jia et al. 2021). Human-based algorithms attempt to reproduce human social behaviour, such as Running City Game Optimizer (Ma et al. 2022), Teaching–Learning based Optimization (Rao et al. 2011), Exchange Market Algorithm (Ghorbani and Babaei 2014) and Harmonic Search Optimization (Alatas 2010). Evolutionary algorithms, such as genetic algorithms (Mirjalili 2018) and Ebola Optimization Search Algorithm (Oyelade et al. 2022), model biological evolutionary behaviour through mutation and natural selection. Finally, physics-based algorithms mimic physical laws such as the gravitational force, like the Gravitational Search Algorithm (Rashedi et al. 2009), and particle interactions in an electromagnetic field, like Special Relativity Search (Goodarzimehr et al. 2022).

However, despite the increasing number of metaheuristics that have appeared in recent years, and despite their ability to efficiently solve a multitude of real optimisation problems, it should be noted that, according to the "No Free Lunch" (NFL) theorem (Wolpert and Macready 1997), no algorithm is better than all others in all optimisation problems. This is why several researchers, over the last few decades, have focused on improving the performance of existing metaheuristic algorithms (Chaudhary and Banati 2021), in terms of premature convergence, stagnation in local optima and improved exploration and exploitation of search spaces. Various concepts have been introduced for this improvement, including hybridisation between two or more metaheuristics (Tian et al. 2024; Abed-alguni 2019), the concept of quantum computing (Gharehchopogh 2022; Fatahi et al. 2024; Zamani et al. 2021, 2022), the Lévy-Flight strategy (Lan et al. 2023; Li et al. 2022) and the integration of chaotic maps (Singh et al. 2023; Cavlak et al. 2023).

Recently, an innovative physics-based metaheuristic optimization algorithm called the Kepler Optimization Algorithm (KOA) (Abdel-Basset et al. 2023) has been introduced, inspired by Kepler's laws of planetary motion. According to these principles, the four key elements that define a planet's orbit around the Sun its position, mass, gravitational force and orbital velocity are used to guide the search for optimal solutions. In this approach, candidate solutions, or "planets", establish dynamic interactions with the "Sun" (the current best solution), enabling finer exploration and exploitation of the solution space.

However, although promising, the KOA may face challenges such as convergence to suboptimal solutions or slow convergence speed. To overcome these obstacles, we propose an improvement by incorporating principles from chaos theory (Boccaletti et al. 2000; Naik and Singh 2022), a branch of mathematics dealing with dynamic systems sensitive to initial conditions. This theory has been successfully applied in several fields (Chen and W. LI et al. 2020; Talatahari and Azizi 2021, 2020; Abd Elaziz et al. 2021), including parameter optimization and feature selection (Zaimoğlu et al. 2023), to introduce unpredictable and sensitive dynamics into the optimization process (Özbay 2023; Kumar et al. 2024; Azizi et al. 2023).

As a result, we introduce the Chaotic Kepler Optimizer Algorithm (CKOA), an innovative fusion of the Kepler optimization algorithm with various chaotic maps to enhance solution diversity and search space exploration capability. This hybrid approach takes advantage of the unpredictability and sensitivity to initial conditions of chaotic systems to enhance the optimization process, aiming to overcome the limitations of conventional methods through improved exploration and exploitation.

In order to evaluate the efficiency of the ten CKOA variants, we have subjected them to the Congress on Evolutionary Computation benchmarks CEC2020 (Kumar et al. 2024) and CEC2022 (Biedrzycki et al. 2022). This evaluation began with a comparison between their performance and that of the original KOA. The comparison criteria included the mean value and standard deviation of fitness, population diversity and the convergence curve. Then, WILCOXON statistical tests were performed on the results obtained to confirm the choice of the CKOA variant, adopted later, for an external comparison with eight other recently developed metaheuristic algorithms. These are the White Shark Optimizer (WSO) (Braik et al. 2022), the Whale Optimization Algorithm (WOA) (Mirjalili and Lewis 2016), the Sin Cosine Algorithm (SCA) (Mirjalili 2016), the Seagull Optimization Algorithm (SOA) (Dhiman and Kumar 2019), the Aquila Optimization (AO) (Abualigah et al. 2021), the Archimedes Optimization Algorithm (AOA) (Hashim et al. 2021), the Coati Optimization Algorithm (COA) (Dehghani et al. 2023) and the Zebra Optimization Algorithm (ZOA) (Trojovská et al. 2022). Finally, the proposed method was tested on three concrete engineering design cases with specific constraints, in order to demonstrate its ability to solve optimisation problems in real-life situations.

The main contribution of this paper is highlighted below:

- Overview of the new Kepler Optimizer Algorithm inspired by planetary laws.
- Introduction of the CKOA algorithm, which combines 10 chaotic maps with KOA to improve the diversity of these solutions and increase their accuracy.
- Efficiency and performance test of the ten proposed CKOA algorithm variants, with the 10 test functions of the CEC2020 benchmark and the 12 test functions of the CEC2022 benchmark.
- Diversity analysis of the proposed CKOA population compared to the KOA population.

- Wilcoxon test confirmation of the statistical superiority of the proposed CKOA for CEC2020 benchmark and CEC2022 benchmark.
- Comparison between CKOA and 8 recent optimizers using CEC2020 and CEC2022 benchmarks.
- Test of CKOA on 3 engineering problems, demonstrating real-life optimization capability.

This paper is structured as follows: Sect. 2 presents, firstly, the Kepler optimization algorithm in its original state, and secondly, the ten chaotic maps we have used in our method. Section 3 provides an in-depth explanation of the proposed CKOA algorithm. Section 4 analyses the proposed algorithm's complexity. Section 5 evaluates the performance of CKOA through the CEC2020 benchmark and the CEC2022 benchmark, performs statistical test on benchmark results and compares it with other recent metaheuristic algorithms. Section 6 evaluates the effectiveness of the proposed algorithm on three different constrained engineering problems. Finally, Sect. 7 summarizes this study and proposes perspectives for future research.

2 Preliminaries

In this section, we outline the theoretical foundations and fundamental concepts that are used to develop our optimisation approach. First, we discuss Kepler's laws, which define the motion of planets around the Sun and are the main origin of the Kepler Optimizer (KOA). Next, we detail the key steps in the original KOA algorithm, highlighting its solution search and update process. Finally, we present the chaotic maps we used to enrich the exploration of the search space and improve the variety of solutions.

2.1 The original Kepler optimization algorithm

Recently, the Kepler Optimization Algorithm (KOA) (Abdel-Basset et al. 2023) has been presented as a new metaheuristic method for tackling the challenges of continuous optimization. The inspiration for the algorithm came from Kepler's theory of planetary motion, which is based on three laws. These laws can be stated as follows:

- Law 1: Every planet has an elliptical orbit. The Sun is located at one of the two focus points of this orbit. The ellipse's degree of flattening is represented by eccentricity e. The line segment ($e=1$) and the circle ($e=0$) are the limits.
- Law 2: The area swept by a line connecting a planet to the Sun is constant for equal time intervals. This law explains the variation in a planet's speed as it moves around the Sun. In other words, the planet moves at different speeds at different points in its orbit. It reaches maximum speed when close to the Sun, and slows down as it moves further away. However, the imaginary line drawn between the center of the planet and the center of the Sun sweeps an equal area in equal periods of time.
- Law 3: The square of a planet's orbital period is proportional to the cube of the semi-major axis of its orbit. This law expresses the relationship between the time required

for a planet to complete one orbit around the Sun (its orbital period) and the size of its orbit.

According to these laws, the orbit of planets around the sun is influenced by four sources: the planet's gravitational force, position, mass and orbital velocity. In the mathematical model of the KOA algorithm, the planets (candidate solutions) occupy variable positions relative to the Sun (best solution) at different times. Thus, planets far from the Sun explore the search space, while those closer to it must exploit promising areas to accelerate the speed of convergence.

During the optimization process, the KOA applies the following rules:

- The orbital period of a planet is randomly selected according to a normal distribution.
- The eccentricity of a planet's orbit is randomly determined in [0, 1].
- The Sun represents the best solution at each iteration.
- The distance between the Sun and each planet is adjusted according to the current iteration.

The main KOA steps are described below:

Step 1: Initialization

*Initialise the positions of the planets:

$$X_i^j = X_{i,low}^j + rd \times \left(X_{i,up}^j - X_{i,low}^j\right), \begin{cases} i = 1, 2, \ldots, N \\ j = 1, 2, \ldots, d \end{cases} \tag{1}$$

where X_i denotes the ith planet; N represents the number of candidate solutions; d represents the problem dimension; $X_{i,up}^j and X_{i,low}^j$ represent the upper and lower bounds, respectively, of the jth decision variable; and rd is a randomly generated number between 0 and 1.

*Initialize the orbital eccentricity (e) of each ith object:

$$e_i = r, i = 1, \ldots, N \tag{2}$$

where r is a number generated randomly between 0 and 1.

*Set the orbital periods T_i of planets:

$$T_i = \left|r_n\right|, i = 1, \ldots, N \tag{3}$$

where r_n is the randomly generated number based on the normal distribution.

Equations 1–3 randomly determine the position of each planet in the interval enclosed by its lower and upper bounds, the eccentricity of each planet in the interval [0,1] and the orbital period of each planet randomly according to a normal distribution. Thanks to this initialisation, the initial population can be diversified, offering a variety of solutions in the search space.

Step 2: Calculate the gravitational force

The orbital speed of a planet depends in particular on the Sun's gravity. It increases with increasing proximity to the Sun. The attractive force of the Sun X_S and any planet X_i can be calculated using the following formula:

$$F_{g_i}(t) = e_i \times \mu(t) \times \frac{\overline{M_s} \times \overline{m_i}}{\overline{R_i}^2 + \varepsilon} + r_1 \tag{4}$$

where ε is a small value used to avoid a division by zero error, r_1 is a random number between 0 and 1. $\overline{M_s} and \overline{m_i}$ represent the normalized values of M_s and m_i, respectively, where M_s represents the mass of X_s and m_i represents the mass of X_i. For each iteration t, M_s and m_i are calculated according to Eqs. (7) and (8); $\overline{R_i}^2$ is the normalised value of the Euclidean distance between the dimensions of the Sun X_s and those of the planet X_i, it is calculated according to Eq. (6); and μ(t) is the universal gravitational constant at iteration t.

$$\mathrm{R_i}(t) = \|X_s(t) - X_i(t)\|_2 = \sqrt{\sum_{j=1}^{d} (X_{sj}(t) - X_{ij}(t))^2} \tag{5}$$

$$\overline{R_i} = \frac{R_i(t) - \min(R(t))}{\max(R(t)) - \min(R(t))} \tag{6}$$

$$\mathrm{M}_s = r_2 \frac{fit_s(t) - worst(t)}{\sum_{k=1}^{N} (fit_k(t) - worst(t))} \tag{7}$$

$$\mathrm{m}_i = r_2 \frac{fit_i(t) - worst(t)}{\sum_{k=1}^{N} (fit_k(t) - worst(t))} \tag{8}$$

$$fit_s(t) = best(t) = \min_{k \in \{1,2,\ldots,N\}} fit_k(t) \tag{9}$$

$$worst(t) = \max_{k \in \{1,2,\ldots,N\}} fit_k(t) \tag{10}$$

where r_2 is a number randomly generated between 0 and 1 to make the mass values of the different planets diverge.

The universal gravitational constant μ(t) is calculated using the following equation:

$$\mu(t) = \mu_0 \times \exp\left(-\gamma \frac{t}{T_{max}}\right) \tag{11}$$

where γ is a constant; μ_0 is an initial value; and t and T_{max} are the current iteration number and the maximum number of iterations respectively.

Step 3: Calculate object velocities

The planet's velocity is influenced by its proximity to the Sun. In other words, as a planet moves closer to the Sun, its speed increases, while it decreases as it moves further away. This variation in velocity is due to the intense gravitational force imposed by the Sun. According to KOA (Abdel-Basset et al. 2023), the following equation can be used to determine the velocity of each planet as a function of its distance from the Sun:

$$V_i(t) = \begin{cases} l \times \left(2r_4\overrightarrow{X_i} - \overrightarrow{X_b}\right) + \ddot{\ell} \times \left(\overrightarrow{X_a} - \overrightarrow{X_b}\right) + \left(1 - \overrightarrow{R_i}(t)\right) \\ \times F \times \overrightarrow{U_1} \times \overrightarrow{r_5} \times \left(\overrightarrow{X_{i,up}} - \overrightarrow{X_{i,low}}\right), if \overrightarrow{R_i}(t) \le 0.5 \\ r_4 \times L \times \left(\overrightarrow{X_a} - \overrightarrow{X_i}\right) + \left(1 - \overrightarrow{R_i}(t)\right) \times F \times \overrightarrow{U_2} \times \overrightarrow{r_5} \\ \times \left(r_3\overrightarrow{X_{i,up}} - \overrightarrow{X_{i,low}}\right), else \end{cases} \tag{12}$$

$$\ell = \overrightarrow{\mathrm{U}} \times \mathcal{M} \times \mathcal{L} \tag{13}$$

$$\mathcal{L} = \left[\mu(t) \times (\mathrm{M_s} + \mathrm{m_i}) \left| \frac{2}{\mathrm{R_i}(t) + \varepsilon} - \frac{1}{\mathrm{a_{i(t)}} + \varepsilon} \right| \right]^{\frac{1}{2}} \tag{14}$$

$$\mathcal{M} = \left(\mathrm{r_3} \times \left(1 - \mathrm{r_4}\right) + \mathrm{r_4}\right) \tag{15}$$

$$\overrightarrow{\mathrm{U}} = \begin{cases} 0, if \ \overrightarrow{\mathrm{r_5}} \le \overrightarrow{\mathrm{r_6}} \\ 1, else \end{cases} \tag{16}$$

$$\mathcal{F} = \begin{cases} 1, if \ \mathrm{r_4} \le 0.5 \\ -1, Else \end{cases} \tag{17}$$

$$\ddot{\ell} = (1 - \overrightarrow{\mathrm{U}}) \times \overrightarrow{\mathcal{M}} \times \mathcal{L} \tag{18}$$

$$\overrightarrow{\mathcal{M}} = (r_3 \times \left(1 - \overrightarrow{r_5}\right) + \overrightarrow{r_5}) \tag{19}$$

$$\overrightarrow{U_1} = \begin{cases} 0, if \ \overrightarrow{r_5} \le r_4 \\ 1, Else \end{cases} \tag{20}$$

$$U_2 = \begin{cases} 0, if \ r_3 \le r_4 \\ 1, Else \end{cases} \tag{21}$$

where $\overrightarrow{\mathrm{V_i}}(\mathrm{t})$ represents the velocity of the ith planet at iteration t, $\overrightarrow{\mathrm{X_i}}$ represents planet i, $\mathrm{r_3}$ and $\mathrm{r_4}$ are two randomly generated numerical values between 0 and 1, and $\overrightarrow{\mathrm{r_5}}$ and $\overrightarrow{\mathrm{r_6}}$ are two random vectors including values between 0 and 1. $\overrightarrow{\mathrm{X_a}}$ and $\overrightarrow{\mathrm{X_b}}$ represent two solutions randomly selected from the population; $\mathrm{M_s}$ and $\mathrm{m_i}$ are the mass of $\mathrm{X_S}$ and $\mathrm{X_i}$, respectively; ε is a small value to avoid a division error by zero; F is a flag randomly including a value of $\{1, -1\}$ to modify the search direction to avoid getting stuck in local minima and to obtain better results; and $\mathrm{a_i}$ represents the semi-major axis of the elliptical orbit of object i at time t, and it is defined by Kepler's third law as follows:

$$a_i(t) = r_3 \times \left[T_i^2 \times \frac{\mu(t) \times (M_s + m_i)}{4\pi^2}\right]^{\frac{1}{3}} \tag{22}$$

where T_i and μ(t) are determined by Eqs. (3) and (11), respectively.

Step 4: Update object positions

Once the velocity of each planet has been estimated, its new position can be calculated using the following mathematical formula:

$$\overrightarrow{X_i}(t+1) = \overrightarrow{X_i}(t) + \mathcal{F} \times \overrightarrow{V_i}(t) + (\boldsymbol{F}_{\boldsymbol{g}_i}(t) + |r|) \times \overrightarrow{U} \times (\overrightarrow{X_s}(t) - \overrightarrow{X_i}(t)) \tag{23}$$

where $\overrightarrow{X_i}(t+1)$ is the new position of planet i at iteration t+1, $\mathcal{F}$ is a flag employed to modify the search direction, $\overrightarrow{V_i}(t)$ is the speed of planet i needed to arrive at $\overrightarrow{X_i}(t+1)$, and $\overrightarrow{X_s}(t)$ is the Sun's best position found so far.

In general, the planet velocity represents KOA's exploration operator when a planet is far from the Sun, its gravitational force serves as the exploitation operator, assisting KOA in attacking the most successful solution to date in search of even better ones.

Step 5: Update the distance to the sun

According to Abdel-Basset et al. (2023), to improve KOA exploration and exploitation operators, Eq. (23) is randomly exchanged with Eq. (24) described as follows:

$$\overrightarrow{X_i}(t+1) = \overrightarrow{X_i}(t) \times \overrightarrow{U_1} + (1 - \overrightarrow{U_1}) \times \left(\frac{\overrightarrow{X_i}(t) + \overrightarrow{X_s} + \overrightarrow{X_a}(t)}{3.0} + h \times \left(\frac{\overrightarrow{X_i}(t) + \overrightarrow{X_s} + \overrightarrow{X_a}(t)}{3.0} - \overrightarrow{X_b}(t) \right) \right) \tag{24}$$

where h, as defined below, is an adaptive factor that controls the separation between the Sun and the planet at iteration t:

$$h = \frac{1}{e^{\mathfrak{y}r}} \tag{25}$$

where r is a randomly generated number based on the normal distribution, while η is defined as follows:

$$\mathfrak{y} = \left(a_2 - 1\right) \times r_4 + 1 \tag{26}$$

where a_2 is a cyclic control parameter defined below:

$$a_2 = -1 - 1 \times \left(\frac{t\% \frac{T_{max}}{TC}}{\frac{T_{max}}{TC}} \right) \tag{27}$$

where TC is the number of cycles in the optimisation process, and % is the remainder operator.

Step 6: Elitism

In order to guarantee the optimal positions for the planets and the Sun, this step employs an elitist method. Equation (28) summarises this procedure.

$$\vec{X}_{i,new}(t+1) = \begin{cases} \vec{X}_i(t+1), iff(\vec{X}_i(t+1)) \leq f(\vec{X}_i(t)) \\ \vec{X}_i(t), Else \end{cases} \tag{28}$$

The KOA algorithm initializes the positions of the planets and the eccentricities of their orbits using random variables following a uniform distribution. However, this method frequently leads to local optima, reducing the efficiency of the search. To improve this, we use sequences generated by chaotic systems to define parameter values. Chaotic initiation provides a higher quality initial population, making it easier to find more suitable global optima. In addition, the application of chaotic maps to adjust the velocities and positions of planets contributes to a larger exploration space and finer exploration, thus optimising the search performance of the algorithm.

2.2 Chaotic maps

Recently, chaos theory has made significant progress, finding use in several scientific areas, including encryption (Bencherqui et al. 2024a) and optimisation (Bencherqui et al. 2024b; Abdelrazek et al. 2024). Chaotic maps, characterised by their sensitivity to initial conditions, their unpredictability, and their dynamic nature, have been integrated into various famous optimisation algorithms such as MFO (Gupta et al. 2022), FA (Alhadawi et al. 2020), ABC (Jin et al. 2021), BBO (Pradeep Kumar et al. 2023), PSO (Peng et al. 2021), and GWO (Lu et al. 2020). This integration has led to significant advances in problem solving and process improvement, by introducing unpredictable dynamics that help overcome local optima (Li et al. 2022b; Shinde et al. 2024). Table 1 presents the ten one-dimensional

Table 1 Chaotic maps

Map name	Equation
Tchebychev map	$x_{k+1} = \cos(k\, cos^{-1}(x_k))$
Circle map	$x_{k+1} = x_k + b - (\frac{a}{2\pi})\sin(2\pi x_k)\text{mod}(1)$
Mouse map	$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ \frac{1}{x_k}\ mod(1) & otherwise \end{cases}$
Iterative map	$x_{k+1} = \sin(\frac{a\pi}{x_k}), \quad a = 0.7$
Logistic map	$x_{k+1} = a x_k(1 - x_k), \quad a = 4$
Piecewise map	$x_{k+1} = \begin{cases} \frac{x_k}{p} & 0 \leq x_k \leq p \\ \frac{x_k - p}{0.5 - p} & P \leq x_k \leq 1/2 \\ \frac{1-p-x_k}{0.5-p} & 1/2 \leq x_k \leq 1-p \\ \frac{1-x_k}{p} & 1-p \leq x_k \leq 1 \end{cases}$
Sine map	$x_{k+1} = \frac{a}{4}\sin(\pi x_k), \quad a = 4$
Singer map	$x_{k+1} = \mu(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.302875x_k^4), \quad \mu = 1.07$
Sinusoidal map	$x_{k+1} = a x_k^2 \sin(\pi x_k), \quad a = 2.3$
Tent map	$x_{k+1} = \begin{cases} \frac{x_k}{0.7}, & x_k < 0.7 \\ \frac{10}{3}(1 - x_k), & x_k \geq 0.7 \end{cases}$

chaotic maps (Abd Elminaam et al. 2022) that we will use to improve the performance of the KOA algorithm, promising a wider exploration of solutions and a more efficient search.

3 The proposed Chaotic-Kepler optimisation algorithm (CKOA)

As part of our research, we have developed a new algorithm, which we have named the Chaotic-Kepler Optimisation Algorithm (CKOA). Our approach is to replace initialization and population search with chaotic variables, rather than conventional random variables. We exploit the ergodic properties of chaotic maps to improve search efficiency by avoiding local optima. CKOA generates its chaotic sequences with ten different chaotic maps: Chebyshev, Circle, Mouse, Iterative, Logistic, Piecewise, Sine, Singer, Sinusoidal, and Tent, as shown in Table 1. These sequences are used to replace the random sequences of the original KOA in four of its key components: initialisation of the planet's eccentricities and positions, computation of the semimajor axis of each planet's elliptical orbit and computation of the planets' velocities and positions at each iteration. This improves the ability of our algorithm to avoid local optima and increases the probability of finding the global optimum in a limited number of iterations.

Each chaotic map is evaluated independently. The pseudocode of the CKOA algorithm is shown in Algorithm 1. In this algorithm, *cc* represents a chaotic sequence of size *N,* generated by a specific chaotic map.

3.1 Population initialization

The classical method of generating initial populations, based on sequences of uniformly distributed random numbers, is used in the original KOA. However, to initialise the position and eccentricity of all the planets, our new CKOA approach uses chaotic sequences, according to the following formula:

$$X_i^j = X_{il}^j + cc_i \times \left(X_{i,up}^j - X_{i,low}^j\right), \begin{cases} i = 1, 2, \ldots, N \\ j = 1, 2, \ldots, d \end{cases} \tag{29}$$

$$e_i = cc_i, i = 1, \ldots, N \tag{30}$$

3.2 Computation of the semimajor axis values of the planet's elliptical orbits

The semi-major axis of the elliptical orbit of each planet i at time t is calculated using the chaotic maps according to the following equation:

$$a_i(t) = cc_i \times \left[T_i^2 \times \frac{\mu(t) \times (M_s + m_i)}{4\pi^2} \right]^{\frac{1}{3}}, i = 1, 2, \ldots, N \tag{31}$$

where μ(t) is the universal gravitational constant calculated by Eq. (11), T_i is the orbital period of planet i and is determined by Eq. (3), M_s is the Sun's mass and m_i is that of planet i.

3.3 Computation of planet velocities

The velocity of each planet i is calculated using the chaotic maps according to the following equation:

$$V_i(t) = \begin{cases} \ell_c \times \left(2r_4\overrightarrow{X_i} - \overrightarrow{X_b}\right) + \ddot{\ell} \times \left(\overrightarrow{X_a} - \overrightarrow{X_b}\right) + \left(1 - R_{i-norm}(t)\right) \\ \times F \times \overrightarrow{U_{c1}} \times \overrightarrow{cc} \times \left(\overrightarrow{X_{i,up}} - \overrightarrow{X_{i,low}}\right), \; if R_{i-norm}(t) \le 0.5 \\ r_4 \times L \times \left(\overrightarrow{X_a} - \overrightarrow{X_i}\right) + \left(1 - R_{i-norm}(t)\right) \times F \times \overrightarrow{U_2} \times \overrightarrow{cc} \\ \times \left(r_3\overrightarrow{X_{i,up}} - \overrightarrow{X_{i,low}}\right), else \end{cases} \tag{32}$$

where,

$$\overrightarrow{U_{c1}} = \begin{cases} 0, if \; \overrightarrow{cc} \le r_4 \\ 1, Else \end{cases} \tag{33}$$

$$\ell_c = \overrightarrow{W_c} \times \mathcal{M} \times \mathcal{L} \tag{34}$$

$$\overrightarrow{W_c} = \begin{cases} 0, \overrightarrow{cc_1} \le \overrightarrow{cc_2} \\ 1, Else \end{cases} \tag{35}$$

where $\overrightarrow{cc_1}$ *and* $\overrightarrow{cc_2}$ are two chaotic sequences.

3.4 Planet position updates

After estimating the velocity of each planet using the chaotic maps, we generate two random numbers r_1 and r_2, and depending on the result of the test ($r_1 < r_2$), the new positions of the planets can be calculated according to Eqs. (36) or (37).

$$\overrightarrow{X_i}(t+1) = \overrightarrow{X_i}(t) + \mathcal{F} \times \overrightarrow{V_i}(t) + (\boldsymbol{F}_{\boldsymbol{g_i}}(t) + |r|) \times \overrightarrow{W_c} \times (\overrightarrow{X_s}(t) - \overrightarrow{X_i}(t)) \tag{36}$$

$$\overrightarrow{X_i}(t+1) = \overrightarrow{X_i}(t) \times \overrightarrow{U_{c1}} + (1 - \overrightarrow{U_{c1}}) \times \left(\frac{\overrightarrow{X_i}(t) + \overrightarrow{X_s} + \overrightarrow{X_a}(t)}{3.0} + h \times \left(\frac{\overrightarrow{X_i}(t) + \overrightarrow{X_s} + \overrightarrow{X_a}(t)}{3.0} - \overrightarrow{X_b}(t) \right) \right) \tag{37}$$

Algorithm 1 Pseudo code of CKOA

Start

Set parameters $N, T_{max}, \mu_0, \gamma, Tc$

Initialize orbital periods on the basis of the normal distribution using Eq. (3)

Initialize objects population with chaotic position and chaotic orbital eccentricities using Eq. (29) and (30), respectively.

Evaluate fitness value for initial population.

Determine the global best (X_s) solution as the Sun.

While ($t < T_{max}$)

Update best(t), worst(t), and , $\mu(t)$ using Eqs. (9), (10) and (11), respectively.

For i = 1 : N

Calculate the Euclidian distance between the Sun and the planet i using Eq. (5).

Calculate the gravitational force between the Sun and the planet i using Eq. (4).

Calculate the semi-major axis of the orbit of the planet i using Eq. (31)

Generate two chaotic sequences $\overrightarrow{cc_1}$ et $\overrightarrow{cc_2}$

Generate a randomly binary vector by $\overrightarrow{cc_1}$ and $\overrightarrow{cc_2}$ using Eq. (33) and Eq. (35)

Calculate the velocity of the object X_i according to Eq. (32)

Generate two random numbers r, r_1 between 0 and 1.

If r> r1

Generate a randomly binary vector by chaotic sequences using Eq. (35)

Update the planet position using Eq. (36).

Else

Generate a randomly binary vector by chaotic sequences using Eq. (33)

Update the planet position using Eq. (37).

End if

Apply an elitist strategy to select the best position of the Sun and the planets, using Eq. (28).

End For

t=t+1

End While

Stop

4 Complexity analysis

The computational complexity is very important in determining the execution time of an algorithm. The complexity of the CKOA depends on the number of dimensions D, the population size N and the maximum number of iterations Tmax. It is equal to the sum of the initialization phase complexity and the algorithm's main loop complexity. The initialization phase has a time complexity of O(N × D), since it generates an initial population of N candidate solutions, each with D dimensions. The main loop runs for Tmax iterations, and the complexity of each iteration is a combination of the complexities of sorting the fitness values O(Nlog(N)), calculating the Euclidean distance O(N × D), calculating the gravitational masses and forces O(N × N), updating the positions O(N × D) and evaluating the new positions O(N). Therefore, the total time complexity of CKOA is:

$$\begin{aligned} &O(N \times D) + O(Tmax \times Nlog(N)) + O(Tmax \times N \times D) \\ &+ O(Tmax \times N \times N) + O(Tmax \times N \times D)) + O(N). \end{aligned}$$

This can be simplified according to the dominance of the terms into: O(Tmax × N × (N + D)).

5 Simulation results and discussion

Our method is initially evaluated using two recent and demanding mathematical test suites, CEC 2020 (Azizi et al. 2023) and CEC 2022 (Abdel-Basset et al. 2023; Mohamed et al. 2020). The CEC 2020 benchmark has 10 test functions, while CEC 2022 has a total of 12 functions. These functions are classified into 4 categories: unimodal, multimodal, hybrid and compositional, constituting a significant challenge for metaheuristic algorithms. After developing ten variants of the CKOA: ChebyshevKOA, CircleKOA, MouseKOA, IterativeKOA, LogisticKOA, PiecewiseKOA, SineKOA, SingerKOA, SinusoidalKOA and TentKOA, we compared their performance with that of the original KOA. The comparison criteria included the mean fitness value, the standard deviation, the diversity population and the convergence curve. We then performed the WILCOXON statistical tests on the results obtained to confirm the choice of the CKOA variant, which we will adopt later, for an external comparison with other metaheuristics, in particular: WSO (Braik et al. 2022), WOA (Mirjalili and Lewis 2016), SCA (Mirjalili 2016), SOA (Dhiman and Kumar 2019), AO (Abualigah et al. 2021), AOA (Hashim et al. 2021), COA (Dehghani et al. 2023) and ZOA (Trojovská et al. 2022).

These algorithms have been evaluated with identical control parameters: a maximum number of iterations T_{max} of 50,000 and a population N of 25, and are run independently 30 times for each test function. The tests were executed using MATLAB R2021a on a device with an Intel (R) Core (TM) i5-7300U processor at 2.60 GHz, 8 GB RAM, and a 64-bit version of Windows 10 Pro.

5.1 Tests based on the CEC'2020 and the CEC'2022 benchmark

The CEC 2020 benchmark includes 10 test functions (Azizi et al. 2023), and twelve different test functions compose the CEC2022 test suite (Abdelrazek et al. 2024; Houssein et al. 2023). The functions test for each benchmark are classified as compositional, hybrid, multimodal or unimodal. Tables 2 and 3 provide information on the nature, range and optimal value (Min) of each test function in the CEC2020 and CEC2022 benchmarks, respectively.

Tables 4 and 5 present the mean and standard deviation of KOA and the 10 variants CKOA fitness based on the test functions of the CEC2020 and CEC2022 benchmarks, respectively. With the best values highlighted in bold, in Table 4, we observe that the CKOAs provide the same values for G1 and G8 in the CEC 2020. Eight CKOA variants outperform the KOA in G10 in terms of values. Six variants overcome the KOA in G2

Table 2 Summary of CEC 2020 benchmark test functions

Function number	Function nature	Range	Min
G1	Unimodal function: Shifted and Rotated Bent Cigar function	[− 100,100]	100
G2	Basic function: Shifted and Rotated Schwefel's function	[− 100,100]	1100
G3	Basic function: Shifted and rotated Lunacek bi-Rastrigin function	[− 100,100]	700
G4	Basic function: Expanded Rosenbrock's plus Griewangk's function	[− 100,100]	1900
G5	Hybrid function 1	[− 100,100]	1700
G6	Hybrid function 2	[− 100,100]	1600
G7	Hybrid function 3	[− 100,100]	2100
G8	Composition function 1	[− 100,100]	2200
G9	Composition function 2	[− 100,100]	2400
G10	Composition function 3	[− 100,100]	2500

Table 3 Summary of CEC 2022 benchmark test functions

Function number	Function nature	Range	Min
F1	Uni-modal function: Shifted and full rotated Zakharov function	[− 100,100]	300
F2	Basic function: Full rotated and shifted Rosenbrock's	[− 100,100]	400
F3	Basic function: Full rotated and shifted Expanded Schaffer's	[− 100,100]	600
F4	Basic function: Full rotated and shifted non-Continuous Rastrigin's	[− 100,100]	800
F5	Basic function: Full rotated and shifted Levy function	[− 100,100]	900
F6	Hybrid function 1	[− 100,100]	1800
F7	Hybrid function 2	[− 100,100]	2000
F8	Hybrid function 3	[− 100,100]	2200
F9	Composition function 1	[− 100,100]	2300
F10	Composition function 2	[− 100,100]	2400
F11	Composition function 3	[− 100,100]	2600
F12	Composition function 4	[− 100,100]	2700

Table 4 Test results for the benchmark CEC 2020

	G1		G2		G3		G4		G5	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
KOA	100	2.29e−05	2688.06	310.85	758.20	8.84	1903.99	0.59	6645.87	8254.32
ChebychevKOA	**100**	0.002	2801.58	334.79	**757.22**	16.17	1905.09	1.08	**5534.26**	3815.88
CircleKOA	**100**	0.005	**2386.33**	284.86	**749.52**	9.20	**1903.14**	0.60	9533.02	13,508.80
MouseKOA	**100**	7.15e−05	**2623.91**	236.58	**755.40**	9.80	1904.06	0.63	7174.29	5104.93
IterativeKOA	**100**	6.80e−05	2711.01	318.18	**755.37**	10.11	**1903.96**	0.63	**6269.59**	4916.77
LogisticKOA	**100**	4.80e−05	2986.19	311.28	761.21	8.68	1905.06	0.73	8290.24	6805.39
PiecewiseKOA	**100**	5.20e−05	**2606.45**	349.63	**754.19**	8.83	**1903.99**	0.66	7221.42	4590.49
SineKOA	**100**	5.42e−05	2795.17	355.18	759.07	8.03	1904.87	0.72	8790.61	12,479.31
SingerKOA	48.04e + 6	2.63e + 08	**2637.39**	376.53	761.40	13.48	1907.32	7.36	**5241.14**	4186.79
SinusoidalKOA	152.18	73.31	**2376.95**	495.24	773.50	21.14	**1900.63**	1.66	13,200	7992.52
TentKOA	**100**	5.99e−05	**2407.76**	305.72	**748.90**	6.38	**1903.43**	0.69	**6316.82**	4668.18
	G6		G7		G8		G9		G10	
KOA	1608.42	7.01	3041.50	745.65	2300.71	0.87	2846.47	17.25	2951.13	32.22
ChebychevKOA	1610.06	10.77	3083.35	1189.68	**2300.67**	0.78	**2836.68**	20.98	**2948.41**	33.33
CircleKOA	1617.28	31.03	4015.11	3338.48	2301.19	1.29	**2827.92**	11.18	**2943.09**	33.94
MouseKOA	**1607.24**	4.97	**3033.49**	763.23	**2300.46**	0.71	2854.68	15.52	**2932.76**	28.63
IterativeKOA	1609.11	7.79	**2785.60**	359.93	**2300.63**	0.70	**2843.99**	16.12	**2941.17**	31.32
LogisticKOA	1610.21	7.94	3206.09	1996.64	**2300.43**	0.68	2855.44	17.93	**2949.30**	32.95
PiecewiseKOA	1609.79	6.72	**2862.21**	530.69	**2300.66**	0.82	2853.40	18.03	**2937.02**	32.12
SineKOA	**1608.34**	5.86	**2980.04**	517.97	**2300.30**	0.58	2856.17	16.96	**2933.94**	27.26
SingerKOA	1616.25	29.60	**2644.94**	412.76	**2300.66**	0.68	**2843.77**	27.56	2955.31	33.04
SinusoidalKOA	1655.96	63.60	3837.68	2857.71	2301.34	1.06	**2841.04**	16.21	2956.02	35.40
TentKOA	**1608.37**	7.05	3316.61	2564.08	**2300.50**	0.69	**2840.30**	16.05	**2937.72**	29.40

G1–G10: The CEC 2020 benchmark test functions|Avg: Average|Std: Standard deviation

Table 5 Test results for the benchmark CEC2022

	F1		F2		F3		F4		F5		F6	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
KOA	312.36	19.82	448.14	15.68	600.01	0.02	869.69	12.03	904.28	6.97	5025.46	3498.08
ChebychevKOA	**300.39**	**0.86**	**445.69**	20.25	600.03	0.05	**868.65**	24.87	908.38	27.50	**4010.21**	**3402.29**
CircleKOA	344.21	68.24	**443.99**	17.86	600.10	0.33	**839.38**	**11.87**	933.99	50.18	**2898.47**	**1392.12**
MouseKOA	437.25	228.98	**447.61**	**14.76**	**600.00**	**0.01**	**866.27**	**10.12**	**901.85**	**1.86**	**4950.09**	4022.77
IterativeKOA	432.61	125.32	**446.47**	18.13	**600.00**	**0.01**	**866.84**	16.69	904.68	11.76	**4573.71**	**3153.82**
LogisticKOA	649.11	339.34	453.60	9.17	**600.00**	0.02	875.11	14.27	**903.37**	8.74	**4783.34**	**2807.79**
PiecewiseKOA	326.91	94.96	**442.99**	20.68	**600.004**	**0.007**	**865.88**	15.09	**902.029**	**2.35**	**4311.78**	**3306.53**
SineKOA	720.03	292.51	**446.81**	17.32	**600.00**	**0.004**	875.67	11.04	**903.66**	7.36	5410.15	4397.31
SingerKOA	**300.13**	**0.27**	**434.17**	25.68	600.067	0.12	**862.83**	29.46	909.51	8.82	6608.94	19,243.94
SinusoidalKOA	**308.05**	**14.07**	**443.60**	20.62	601.21	1.10	**840.36**	14.94	1033.97	108.86	**3507.66**	**3481.64**
TentKOA	**300.22**	**0.35**	**446.58**	16.05	**600.00**	0.01	**851.71**	17.31	911.94	26.57	**3867.06**	**3442.95**
	F7		F8		F9		F10		F11		F12	
KOA	2040.31	9.31	2227.85	1.72	2480.78	4.30e− 13	2040.31	9.31	2227.85	1.72	2942.98	7.00
ChebychevKOA	**2039.51**	10.58	2229.71	1.80	**2480.78**	7.49e− 11	**2039.51**	10.58	2229.71	1.80	2947.12	9.70
CircleKOA	**2035.15**	10.11	**2226.28**	**1.27**	**2480.78**	1.01e− 10	**2035.15**	10.11	**2226.28**	**1.27**	2947.50	**6.52**
MouseKOA	**2038.14**	**5.39**	**2227.79**	**0.98**	**2480.78**	5.12e− 12	**2038.14**	**5.39**	**2227.79**	**0.98**	**2942.81**	**5.01**
IterativeKOA	**2038.91**	**7.37**	**2227.83**	**1.18**	**2480.78**	4.77e− 13	**2038.91**	**7.37**	**2227.83**	**1.18**	**2942.18**	**5.90**
LogisticKOA	**2037.91**	**6.50**	2230.64	1.20	**2480.78**	4.46e− 13	**2037.91**	**6.50**	2230.64	1.20	2944.53	7.49
PiecewiseKOA	**2038.65**	**8.59**	2227.85	1.06	**2480.78**	5.72e− 13	**2038.65**	**8.59**	2227.85	1.06	**2940.95**	**3.90**
SineKOA	**2039.78**	**6.52**	2229.36	1.50	**2480.78**	6.20e− 13	**2039.78**	**6.52**	2229.36	1.50	**2942.66**	**6.81**
SingerKOA	2044.34	32.17	2229.71	1.8606	**2480.78**	**3.27e− 13**	2044.34	32.17	2229.71	1.8606	**2942.23**	**5.91**
SinusoidalKOA	**2040.30**	13.96	2228.63	3.89	**2480.78**	4.97e− 07	**2040.30**	13.96	2228.63	3.89	2954.63	15.78
TentKOA	**2039.91**	**7.73**	**2227.23**	**1.12**	**2480.78**	6.31e− 13	**2039.91**	**7.73**	**2227.23**	**1.12**	**2942.80**	7.03

F1–F12: The CEC 2022 benchmark test functions|Avg: Average|Std: Standard deviation

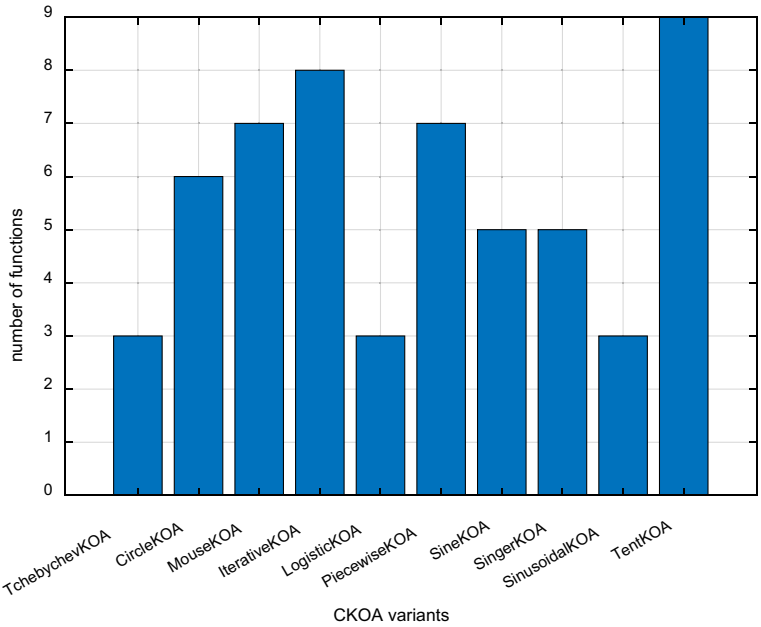

Fig. 1 Ranking of CKOA variants by the benchmark CEC2020

and G9, five surpass the KOA in G3, G4, and G7, four have better values than the KOA in G5, and three surpass the KOA in G6. This table, however, shows that in nine of the ten functions in this benchmark, the TentKOA performs better than the KOA. Table 5 of the CEC 2022 data shows that the mean for F9 is the same for KOA and the ten CKOA variants. Despite the exception of SinusoidalKOA, the same for F3. Additionally, we see that only four variants have lower mean values than KOA in F1, F5, F8, and F11, whereas nine CKOA variants exceed KOA in F2, F7, and F10, eight variants outperform KOA in F4 and F6, and six variants outperform KOA in F12. But we find that, with the exception of F5, TentKOA performs better than KOA in every test function of the CEC2022 benchmark.

As compared to the test functions of the CEC2020 and CEC2022 benchmarks, the CKOA variants are ranked in Figs. 1 and 2, respectively, based on their fitness mean performance. These numbers demonstrate that TentKOA performs better than the other variations in both benchmarks. Based on such findings, we infer that the KOA's efficiency will probably increase dramatically if chaotic maps are integrated.

Population diversity is a decisive aspect for optimisation algorithms, as it influences their ability to explore the search space and avoid local optima. The TentKOA and KOA diversity is shown in Fig. 3 for CEC2020 and Fig. 4 for CEC2022. According to the results in Fig. 3, TentKOA and KOA show different behaviours in terms of diversity over the iterations. In G1, G3, G4, G9, and G10, TentKOA maintained slightly higher diversity,

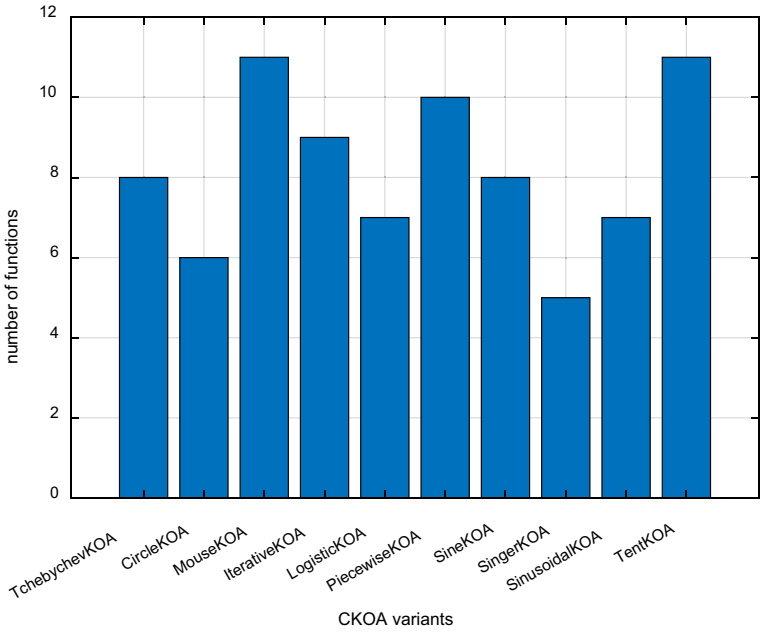

Fig. 2 Ranking of CKOA variants by the benchmark CEC2022

suggesting better exploration and stability of the search space. In contrast, KOA frequently displays higher diversity values with significant variations in the remaining test functions, which may suggest a broader but less stable exploration compared to TentKOA. In terms of convergence, there is a general reduction in diversity for both algorithms, which is expected as they approach optimal solutions. On the other hand, in Fig. 4, TentKOA tends to maintain slightly higher and more stable diversity in F2, F3, F5, F6, F8, F9 and F11, suggesting better exploration and stability of the search space. In contrast, in F1, F7, F10 and F12, KOA frequently displays higher diversity values with significant variations, suggesting a broader but potentially less stable exploration compared to TentKOA. Iterations lead to a general reduction in diversity, which corresponds to the anticipation of convergence towards optimum solutions. These observations reinforce the idea that integrating chaotic maps, such as the Tent map, into the optimisation process not only improves performance in terms of fitness, but also the algorithm's ability to maintain high population diversity.

The qualitative analysis of the convergence curves of the CKOA variants and the original KOA is visualised in Fig. 5 for CEC2020 and in Fig. 6 for CEC2022. With the exception of G1, G6, and G8, the convergence curves show that the suggested CKOA methods employing ten distinct chaotic maps exhibit a faster rate of convergence than the original KOA algorithm.

Lastly, the p-values obtained using a 95% confidence interval from the Wilcoxon test are presented in Tables 6 and 7. This test compares each variant's CKOA results to the original KOA results for every test function in the CEC 2020 and CEC 2022 benchmarks.

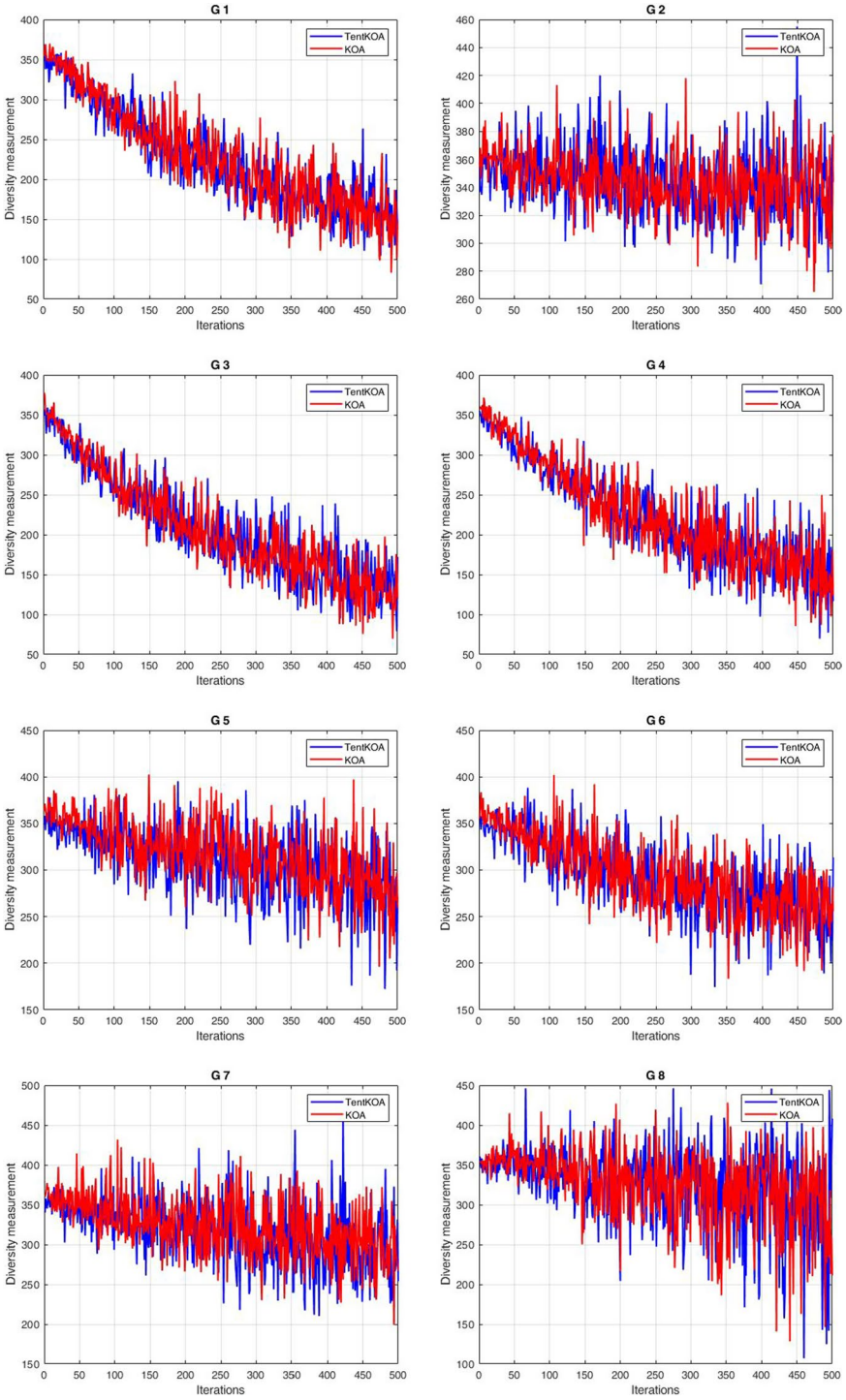

Fig. 3 Diversity analysis performed by KOA and TentKOA for CEC2020

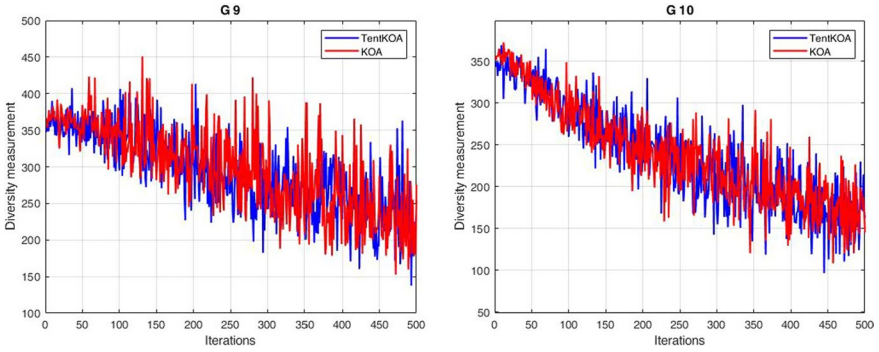

Fig. 3 (continued)

These two tables show that for the great majority of the test functions, the CKOA variants yield results that differ significantly from the original KOA. We also observe that, for the CEC2020 benchmark, the TentKOA performs better than the KOA in 7 out of 10 functions and in 8 out of 12 functions for the CEC2022 benchmark.

The graphical and statistical analyses provide a clear demonstration that the integration of chaotic systems into optimisation processes offers a significant advantage in terms of efficiency over the classic KOA approach. Among the different variants examined, the TentKOA stands out for its exceptional performance, demonstrating its superiority in terms of efficiency and its ability to provide optimal solutions compared with other CKOA variants. This is why we have chosen to compare its performance with that of other recent metaheuristic algorithms recognised in the literature.

5.2 Comparison with other optimizers

To assess the performance of the proposed TentKOA algorithm, it was compared with eight other recognised algorithms: WSO (Braik et al. 2022), WOA (Mirjalili and Lewis 2016), SCA (Mirjalili 2016), SOA (Dhiman and Kumar 2019), AO (Abualigah et al. 2021), AOA (Hashim et al. 2021), COA (Dehghani et al. 2023) and ZOA (Trojovská et al. 2022), using the parameters recommended in their respective publications.

The statistical performances of the means and standard deviations of the algorithms' fitness are summarised in Table 8 for the CEC2020 benchmark and those of the CEC2022 benchmark are summarised in Table 9. Table 8 shows that TentKOA is ranked second for G10, third for G2 and first for the 7 remaining test functions in the CEC2020 benchmark. However, in Table 9, TentKOA is ranked second for test functions F1 and F2 and outperforms the other algorithms for the remaining test functions in the CEC2022 benchmark.

The qualitative analysis of the CKOA convergence curves and the other metaheuristic algorithms for CEC2020 and CEC2022, respectively, is shown in Figs. 7 and 8. Based on these curves, we find that TentKOA converges more quickly than most other methods for the CEC 2020 and CEC 2022 benchmarks.

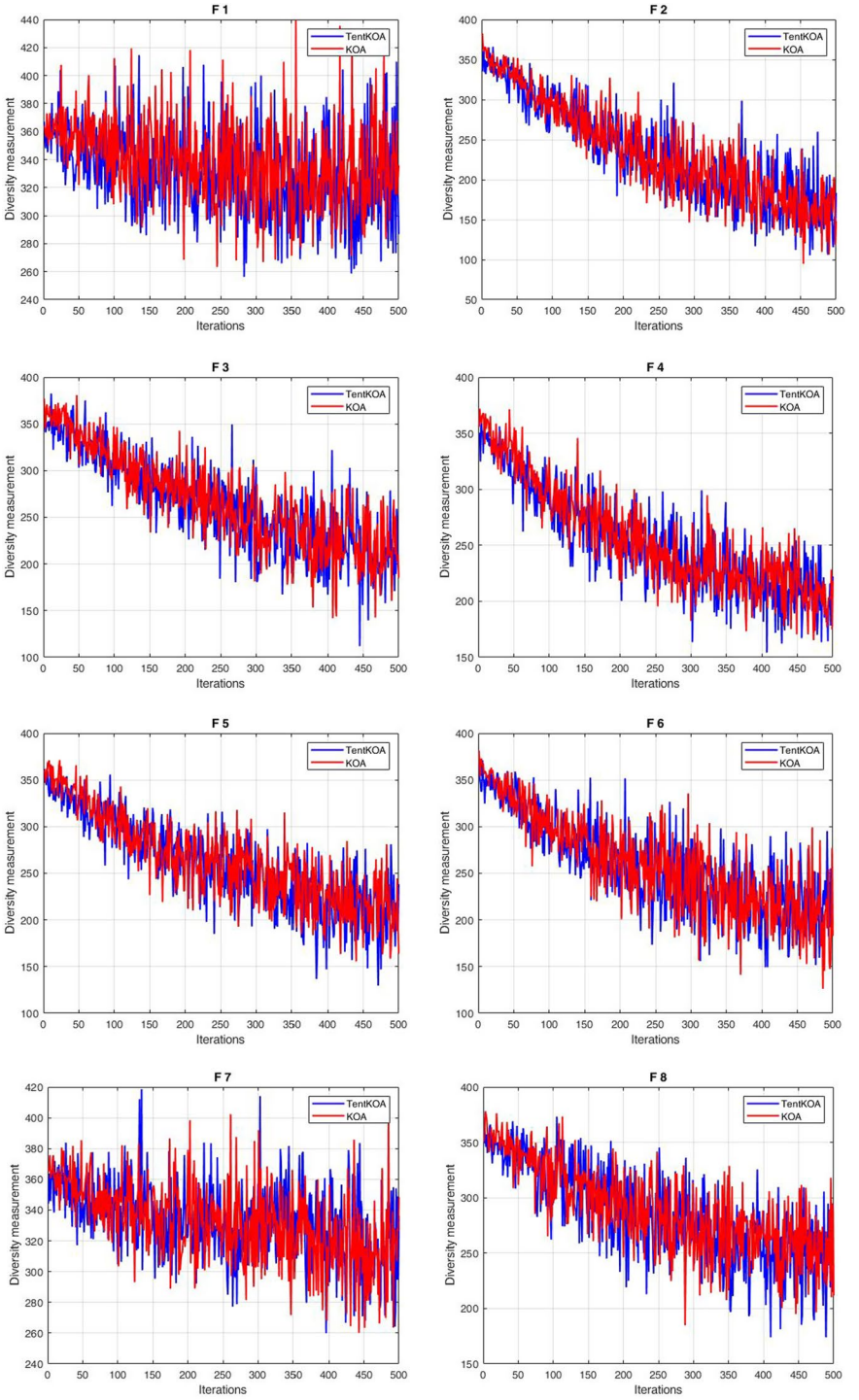

Fig. 4 Diversity analysis performed by KOA and TentKOA for CEC2022

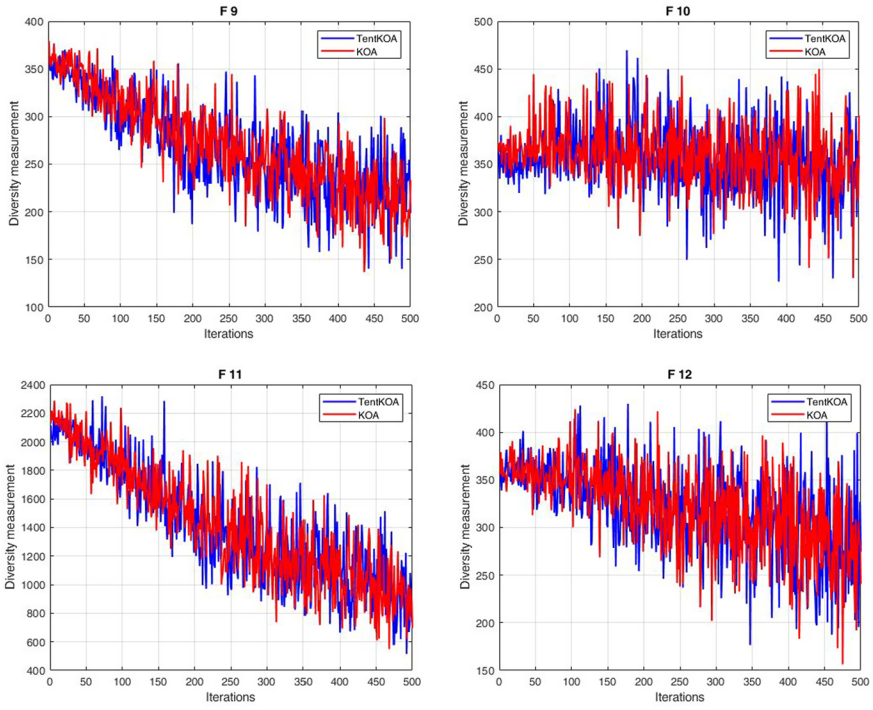

Fig. 4 (continued)

Additionally, Wilcoxon test is applied at a 95% confidence interval to statistically confirm the effectiveness of TentKOA over WSO, WOA, AO, AOA, COA, SOA, SCA and ZOA. Table 10 provides the p-value calculated for CEC2020, while Table 11 provides the p-value calculated for CEC2022. Given the number of p-values in these two tables that are less than 0.05, it is clear that TentKOA outperforms its rivals statistically.

Lastly, Tables 12 and 13 show the performance indexes for the TentKOA and its competitors on both the CEC2020 and CEC2022 benchmarks. These tables show the average fitness and the average time for 30 runs for each test function for each algorithm. TentKOA systematically obtains fitness values that are among the lowest, indicating superior efficiency in optimisation compared with the other algorithms. In addition, TentKOA stands out for its exceptional computational efficiency, generally having the shortest execution times. In terms of performance indices, TentKOA offers an optimal balance between solution quality and execution time, making it particularly well-suited to problems requiring fast, accurate optimisation.

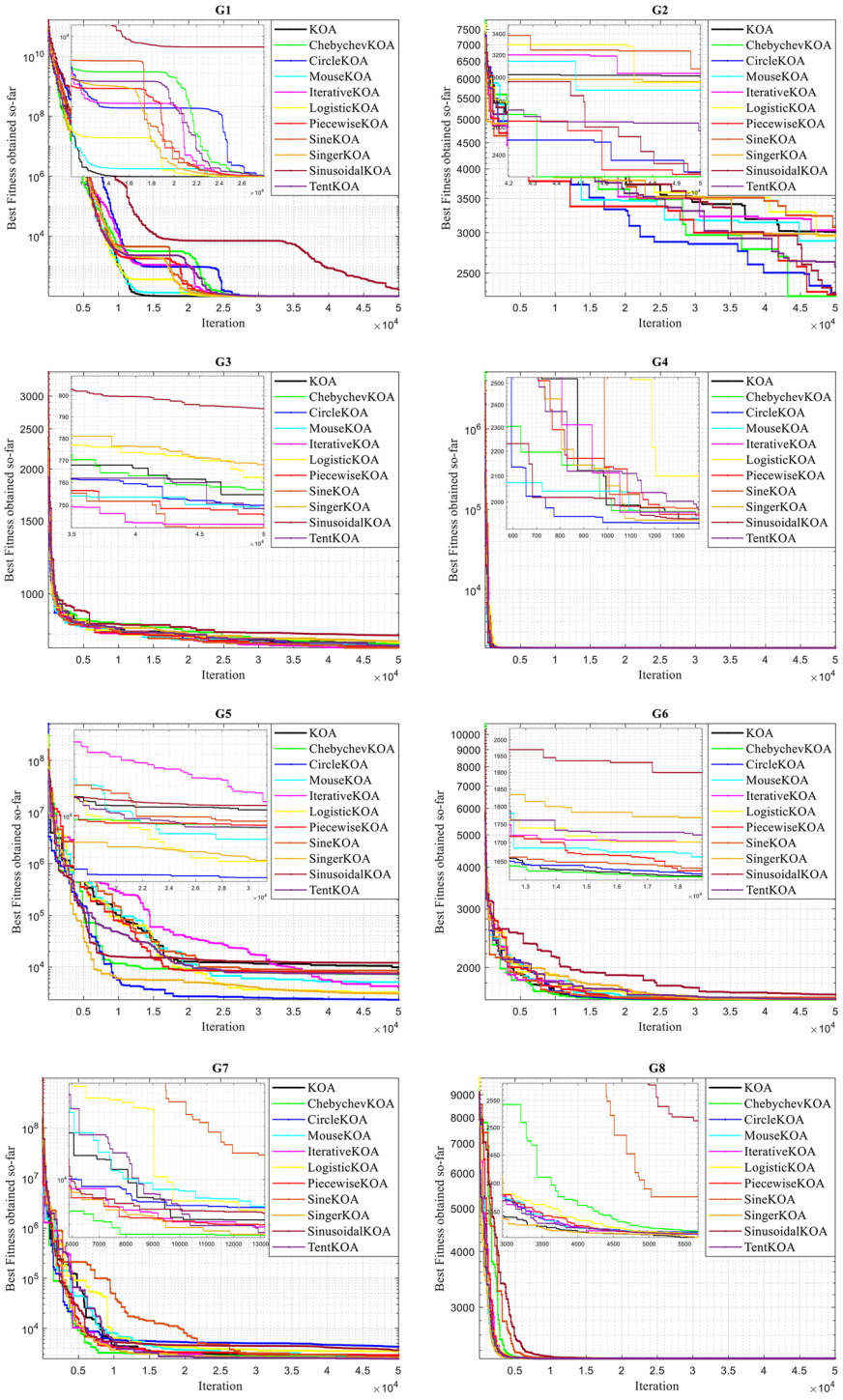

Fig. 5 Convergence curves of CEC2020

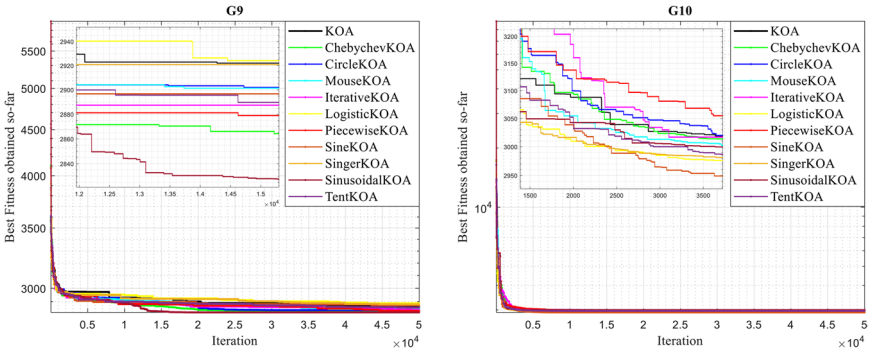

Fig. 5 (continued)

The advantage of TentKOA over the other examined algorithms is evident from statistical and graphical data, highlighting its exceptional efficiency. The TentKOA stands out in particular for its ability to optimise the initial search phases using a dynamic initialisation and updating strategy based on the Tent chaotic map, giving it a significant advantage in avoiding local minima and moving efficiently towards optimal solutions. In addition, the performance indexes underline the TentKOA's ability to deliver high-quality solutions in minimum time, consolidating its position as a powerful and effective tool for solving complex problems.

6 Comparative study of over three engineering problems

In this section, the effectiveness of TentKOA is evaluated on three different constrained engineering problems against eight other optimisers.

6.1 The welded beam design problem (WBDP)

The WBDP, illustrated in Fig. 9, is used to evaluate the effectiveness of the TentKOA. This problem is often used as a reference in metaheuristic optimisation studies. It is a non-linear, constrained optimisation problem with a combination of continuous and discrete variables.

The WBDP is a well-known optimisation problem in mechanical engineering design (RAO 2019). It consists of designing a welded steel beam to minimise cost while satisfying certain constraints related to strength, deflection and other engineering requirements (Gandomi et al. 2013). The welded beam design problem involves several variables and constraints. The design variables include beam width (b), beam thickness (t), welded joint length (l) and welded joint thickness (h). The objective is to minimise the cost function, which is a combination of the cost of the beam material and the welding cost. The constraints of the problem are multiple: the shearing in the weld and the bending in the beam

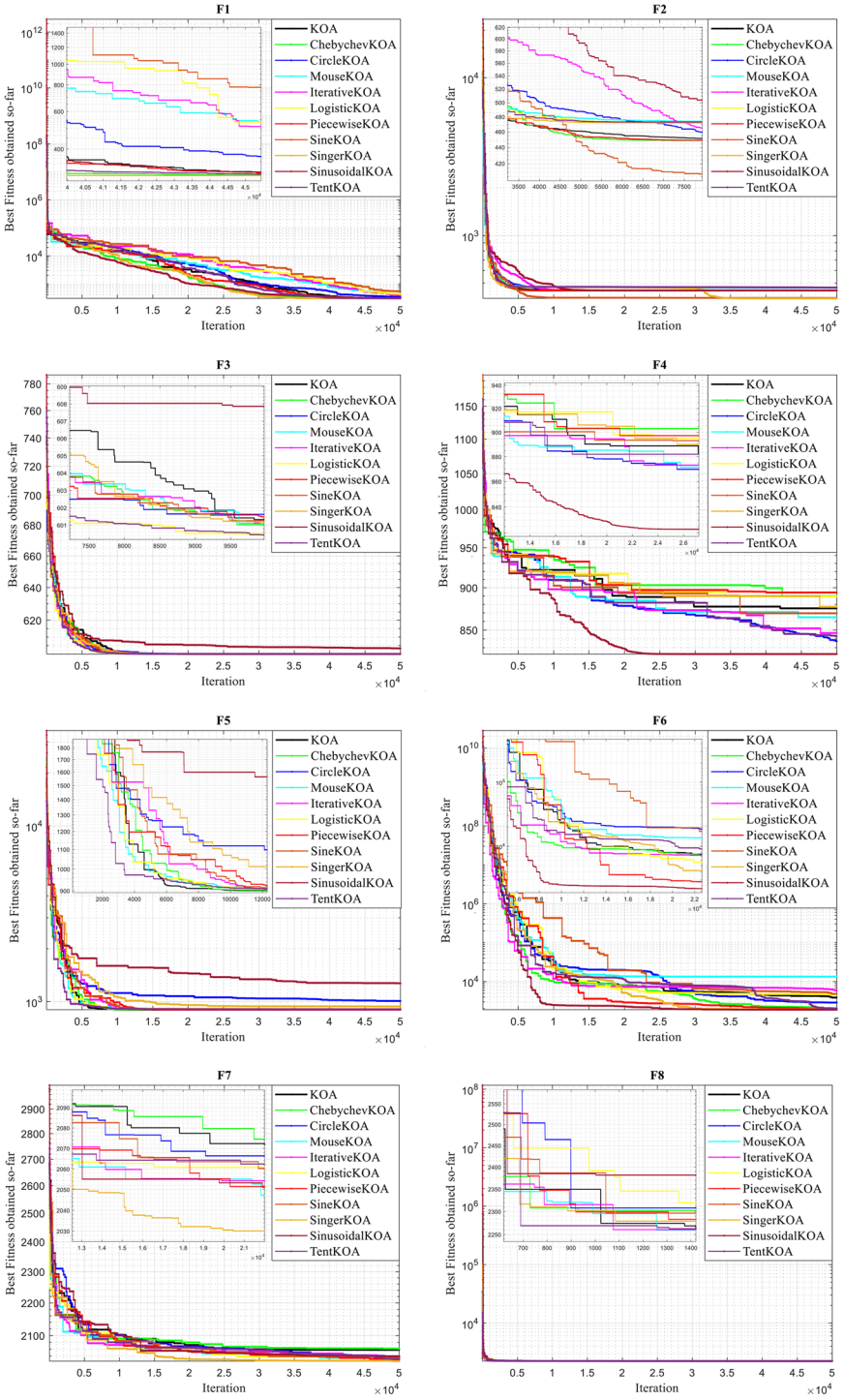

Fig. 6 Convergence curves of CEC2022

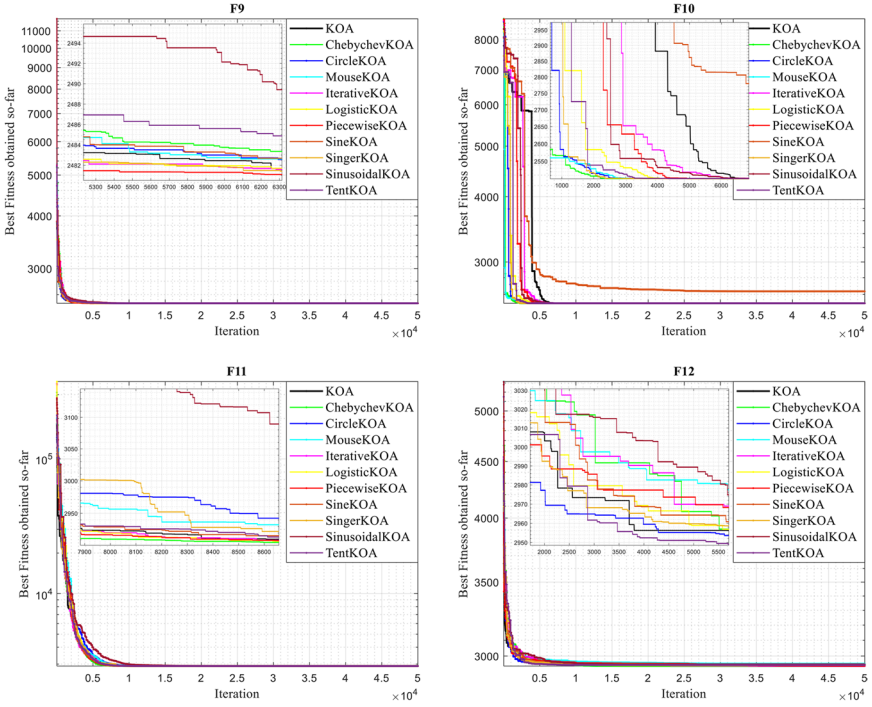

Fig. 6 (continued)

must not exceed specified maximum values, the deflection of the beam under load must be less than a specified value. Finally, the geometric constraints define the minimum and maximum limits for the design variables b, t, l and h. These constraints ensure that the design of the welded beam is both safe and economical. Several metaheuristic algorithms have been used in the literature to handle the WBDP problem and identify the optimal solution. TentKOA has been compared with several of these techniques, including: WSO, WOA, SOA, SCA, COA, AO, AOA, and ZOA. TentKOA and the other eight optimisers are compared in Tables 14 and 15, together with the design characteristics associated with the best optimisation and the statistical outcomes, which include the optimal cost's mean and standard deviation for 30 executions. According to the comparative results, TentKOA is ranked competitively for the best cost value, 1.7248, and the best mean and standard deviation for this particular design problem.

6.2 Tension/compression spring problem (TCSP)

A second classic example in mechanical engineering, illustrated in Fig. 10, is the TCSP where the objective is to design a spring that meets specific requirements while minimising production cost. The key parameters to be optimised in this problem are typically the wire diameter (d), the average coil diameter (D) and the number of active coils (N) (Bencherqui et al. 2024b). This problem is subject to several constraints (RAO 2019). Firstly, there

Table 6 Wilcoxon test for CEC2020 benchmark comparing KOA and CKOA variations

Functions	ChebychevKOA	CircleKOA	MouseKOA	IterativeKOA	LogisticKOA	PiecewiseKOA	SineKOA	SingerKOA	SinusoidalKOA	TentKOA
G1	**0.0014**	**0.0000**	0.0555	**0.0014**	0.3871	0.5895	0.1669	**0.0163**	0.5592	**0.0000**
G2	0.1624	**0.0015**	0.0724	0.1537	**0.0144**	0.7394	0.3329	0.5997	**0.0292**	**0.0012**
G3	0.2581	**0.0000**	0.3042	0.8187	0.7958	0.2398	**0.0005**	0.3555	**0.0138**	**0.0035**
G4	**0.0002**	**0.0000**	0.1154	0.9941	**0.0000**	0.1624	**0.0067**	**0.0000**	**0.0028**	**0.0000**
G5	0.7618	0.3953	0.0724	0.5395	**0.0199**	0.7394	0.0555	0.3255	0.4035	**0.0436**
G6	0.8883	0.0501	0.5997	0.2116	0.4119	0.2340	0.0773	0.7394	0.1715	**0.0000**
G7	0.5395	0.4825	0.3790	0.1154	**0.0292**	0.3042	**0.0070**	0.0877	0.4553	0.1373
G8	0.1246	**0.0467**	0.6445	0.7480	0.6535	0.2591	0.0694	0.5560	0.8289	**0.0014**
G9	**0.0010**	**0.0001**	0.2643	0.1858	0.8303	0.7845	**0.0199**	**0.0232**	**0.0009**	0.6520
G10	0.6627	0.0993	0.9470	0.8303	0.4733	0.4643	0.9823	0.6309	0.8650	0.5692

Bold font stand for the p-values smaller than 5%

Table 7 Wilcoxon test for CEC2022 benchmark comparing KOA and CKOA variations

Functions	ChebychevKOA	CircleKOA	MouseKOA	IterativeKOA	LogisticKOA	PiecewiseKOA	SineKOA	SingerKOA	SinusoidalKOA	TentKOA
F1	**0.0000**	**0.0156**	**0.0000**	**0.0000**	**0.0000**	0.1494	**0.0000**	**0.0000**	**0.0000**	0.1624
F2	0.1537	0.5298	0.8418	0.4733	0.7958	0.1984	0.6048	**0.0035**	0.1154	0.0615
F3	**0.0005**	**0.0016**	0.3871	0.9000	0.4119	0.8187	0.0963	**0.0000**	0.9705	**0.0000**
F4	0.9234	**0.0000**	0.1335	0.5011	0.1453	0.2707	0.0501	0.3042	**0.0001**	**0.0000**
F5	0.8073	**0.0000**	0.0963	0.5493	0.0537	0.0519	0.1373	**0.0014**	0.2062	**0.0000**
F6	**0.0091**	**0.0010**	0.5592	0.5493	0.9823	0.2116	0.7845	**0.0006**	**0.0150**	**0.0011**
F7	0.5793	**0.0044**	0.3711	0.6952	0.5895	0.5997	0.6309	0.7283	0.9000	0.5201
F8	**0.0001**	**0.0000**	0.4376	0.5298	**0.0000**	0.5298	**0.0008**	**0.0001**	**0.0108**	0.6414
F9	0.1289	**0.0000**	**0.0000**	0.6138	0.8742	0.0554	0.2186	0.2543	0.3770	**0.0000**
F10	0.7958	0.3953	1.0000	0.4376	0.9823	0.2707	1.0000	0.5895	0.7731	**0.0061**
F11	**0.0013**	**0.0000**	**0.0000**	0.5166	**0.0023**	0.4978	0.1429	0.0990	0.4348	**0.0000**
F12	0.0657	**0.0028**	0.7731	0.5493	0.6100	0.1120	0.9705	0.9234	0.4918	**0.0000**

Bold font stand for the p-values smaller than 5%

Table 8 Comparison with other metaheuristics according to the CEC2020 benchmark

	G1			G2			G3			G4			G5		
	Avg	Std	Rnk	Avg	Std	Rnk	Avg	Std	Rnk	Avg	Std	Rnk	Avg	Std	Rnk
TentKOA	100	6.06e−05	1	2494.93	375.72	3	751.12	8.49	1	1900.40	0.29	8	6686.40	0.48e+4	1
WSO	3842.41	4035.73	3	1978.36	437.96	1	769.20	23.11	2	1900.74	0.42	9	2.52e+4	2.93e+4	2
WOA	2915.07	2756.42	2	3429.41	600.41	7	910.31	40.27	7	1900.01	0.07	7	22.14e+4	21.84e+4	5
SCA	36.05e+8	73.83e+7	6	4432.27	329.64	8	881.84	18.33	6	1900	0	1	35.98e+4	18.22e+4	6
SOA	20.08e+8	108.11e+7	5	2667.09	511.02	5	843.99	27.10	4	1900	0	1	19.59e+4	20.84e+4	4
AO (Aquila)	12.95e+8	53.69e+3	4	2181.50	373.29	2	781.81	14.64	3	1900	0	1	6.00e+4	3.83e+4	3
AOA(Archimede)	2.54e+10	45.48e+8	9	2989.23	701.73	6	922.88	46.99	8	1900	0	1	134.53e+4	85.12e+4	8
COA	321.83e+8	48.35e+8	8	5638.26	325.25	9	1035.13	25.92	9	1900	0	1	679.61e+4	429.62e+4	9
ZOA	41.16e+8	25.62e+8	7	2549.39	492.01	4	877.28	44.05	5	1900	0	1	38.89e+4	46.60e+4	7
	G6			G7			G8			G9			G10		
TentKOA	1604.90	2.33	1	2820.08	811.96	1	2300.67	0.76	1	2836.17	16.44	1	2945.25	31.46	2
WSO	1835.61	105.82	4	9602.58	9351	2	2496.43	739.74	3	3100.25	109.77	6	2977.90	30.32	3
WOA	2054.49	198.09	6	171,275	112,376	7	4060.79	1719.59	6	2955.62	62.51	5	2996.69	34.25	5
SCA	2017.87	113.68	5	116,327	61,045	4	3350.10	1354.99	5	2952.76	11.58	4	3048.02	37.13	6
SOA	1829.15	112.64	3	20,275	22,580	3	4941.84	742.08	7	2852.07	18.70	2	2985.12	37.65	4
AO (Aquila)	1764.21	92.93	2	123,520	59,269	5	2305.05	1.08	2	2880.23	28.88	3	2942.17	37.75	1
AOA(Archimede)	2432.73	306.94	8	1,940,246	112,456	8	5282.48	930.44	8	3305.37	130.80	8	4532.89	548.42	8
COA	3317.28	397.69	9	3,430,418	484,846	9	5984.33	880.69	9	3441.85	152.20	9	5844.12	870.61	9
ZOA	2314.29	230.53	7	136,679	87,488	6	2846.08	667	4	3153.70	85.98	7	3097.83	89.43	7

G1–G10: The CEC 2020 benchmark test functions|Avg: Average|Std: Standard deviation|RNK: Rank

Table 9 Comparison with other metaheuristics according to the CEC2022 benchmark

	F1			F2			F3			F4			F5			F6		
	AVG	STD	RNK	AVG	STD	RNK	AVG	STD	RNK	AVG	STD	RNK	AVG	STD	RNK	AVG	STD	RNK
TentKOA	300.2	0.354	2	446.5	16.05	2	**600.00**	**0.01**	**1**	**851.70**	**17.311**	**1**	**911.9**	**26.57**	**1**	**3867**	**3442**	**1**
WSO	**300**	**9e-12**	**1**	**429.3**	**25.23**	**1**	602.5	2.31	2	855.98	12.297	2	1674.6	379.63	5	4626	4007	2
WOA	320.7	24.64	4	461.0	30.54	4	658.8	13.36	8	917.74	31.31	8	3147.5	949.27	8	6394	5275	3
SCA	4793.8	1021	6	576.8	30.609	6	630.8	4.265	5	915.34	10.89	7	1561.5	187.01	4	4e+7	2e+6	7
SOA	3354.6	1627	5	515.1	53.45	5	613.3	4.985	3	857.57	12.47	3	1459.8	218.60	3	9708	5063	4
AO (Aquila)	301.8	0.83	3	447.5	21.54	3	617.8	6.16	4	864.56	17.55	5	1454.2	306.69	2	1e+4	5649	5
AOA(Archimede)	25,301	7553	8	1617	465.67	8	648.7	8.562	7	888.24	18.375	6	2385.1	356.33	7	1e+9	8e+8	8
COA	68,028	24,762	9	2794	688.12	9	684.3	7.54	9	982.29	15.83	9	3472.9	386.76	9	2e+9	9e+8	9
ZOA	8297.9	3353	7	590.5	80.66	7	641.8	6.66	6	859.90	12.432	4	1710.6	215.83	6	4e+6	1e+7	6
	F7			F8			F9			F10			F11			F12		
TentKOA	**2039.9**	**7.73**	**1**	**2227**	**1.12**	**1**	**2480.8**	**6e-13**	**1**	**2509**	**34.33**	**1**	**2900**	**2e-12**	**1**	**2942**	**7.03**	**1**
WSO	2057.2	21.15	4	2278	67.61	7	**2480.8**	**3e-08**	**1**	2662	185.8	4	3071.7	936.7	4	3160	121.70	6
WOA	2146.7	53.18	8	2241	9.30	5	2481.1	0.36	4	3594	1181	8	2901.3	0.95	2	2998	44.15	5
SCA	2093.5	10.96	5	2241	3.02	4	2527	12.45	6	2527	58.62	2	4936.8	431.2	6	2994	17.33	4
SOA	2051.5	15.80	3	2231	22.33	3	2491.7	13.61	5	2667	507.3	5	5069.9	657.1	7	2946	3.79	2
AO (Aquila)	2049.7	13.32	2	2228	2.83	2	2481	0.16	3	2555	109.1	3	2905.7	107.5	3	2966	13.95	3
AOA(Archimede)	2132.1	21.46	7	2251	51.20	6	3142.2	258.25	8	3084	685.1	6	7592.8	768.8	8	3173	376.5	7
COA	2224.9	53.74	9	2533	210.14	9	3588.1	445.61	9	6811	1025	9	9114.4	614.4	9	3618	265.1	9
ZOA	2099.1	22.48	6	2297	87.10	8	2584.7	46.12	7	3369	916.6	7	4760	691.2	5	3375	190.4	8

F1-F12: The CEC 2022 benchmark test functions|Avg: Average|Std: Standard deviation| RNK: Rank

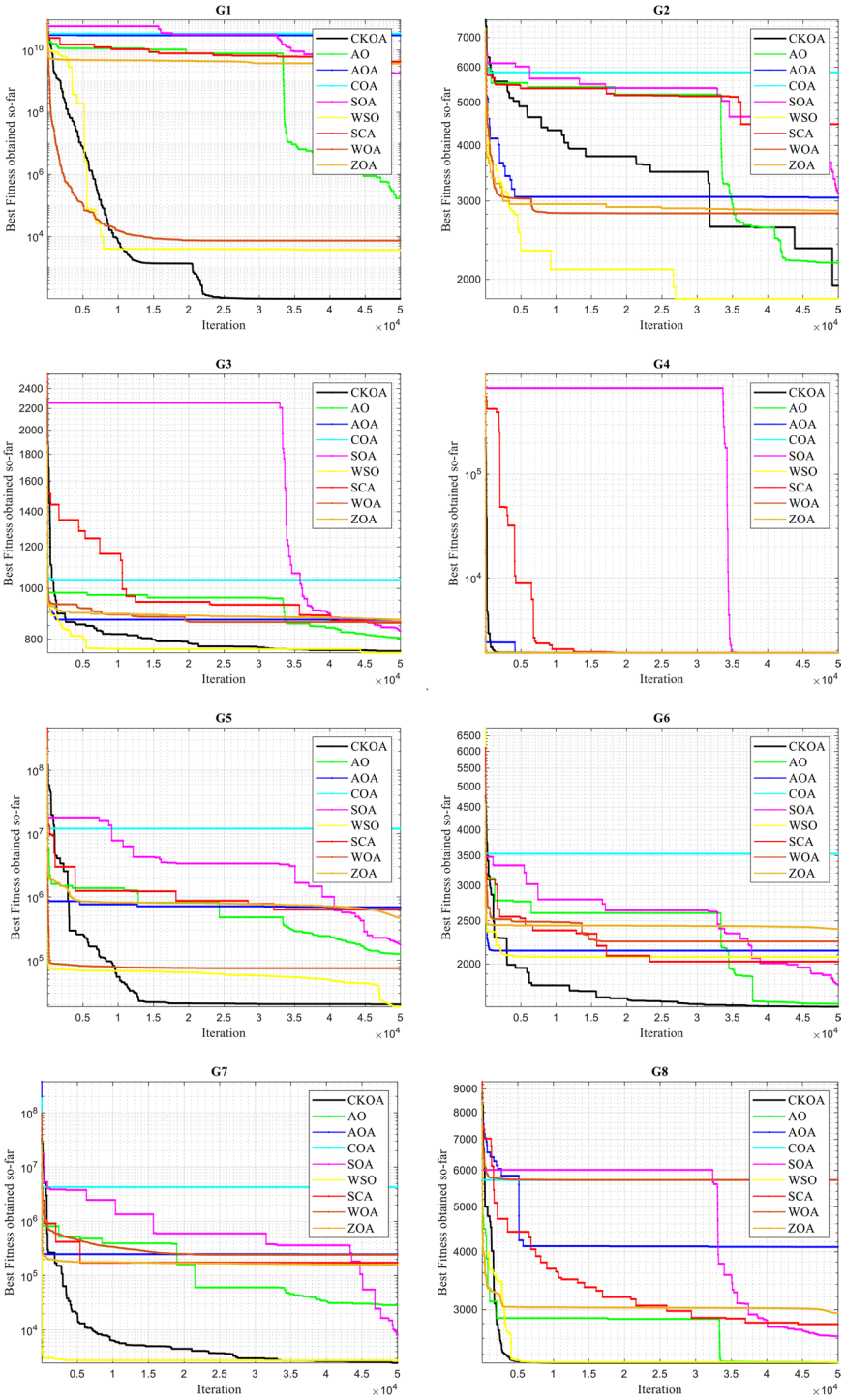

Fig. 7 TentKOA and others metaheuristic convergence curves according to CEC2020

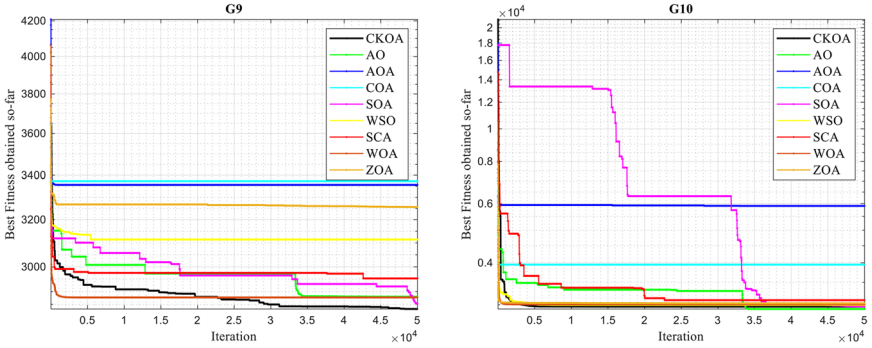

Fig. 7 (continued)

is the strength constraint, where the spring must be able to withstand a maximum load without exceeding the allowable constraint of the material used. Then there is the buckling constraint, which is crucial for compression springs to avoid buckling when compressed to their maximum load. There is also the natural frequency constraint, where the natural frequency of the spring must be above a minimum value to avoid resonance in certain applications. Finally, there are dimensional constraints, where the spring must fit into a defined space, limiting its overall dimensions.

To solve this problem, the metaheuristic algorithms aim to determine the optimal combination of parameters d, D and N that respects all the constraints while minimising the objective function. Several metaheuristic algorithms have been used in the literature to handle the TCSP problem and identify the optimal solution. TentKOA has been compared with several of these techniques, including: WSO, WOA, SOA, SCA, COA, AO, AOA, and ZOA. TentKOA and the other eight optimisers are compared in Tables 16 and 17, together with the design characteristics associated with the best optimisation and the statistical outcomes, which include the optimal cost's mean and standard deviation for 30 executions. According to the comparative results, the TentKOA is competitive with its competitors in terms of best value, mean and standard deviation of fitness, as it has managed to achieve the optimum values of the parameters and, consequently, to obtain the best value and mean of the cost function, which is 0.0126, and the best value of the standard deviation, which is 4.96e-09.

6.3 The Pressure vessel design problem (PVDP)

The PVDP problem is a common optimisation challenge in mechanical engineering, which involves minimising the cost of fabricating a cylindrical pressure vessel. The parameters to be optimised, illustrated in Fig. 11, include the cylinder wall thickness (Ts), the hemispherical head thickness (Th), the cylinder inner radius (R) and the cylinder length (L) (Mirjalili and Lewis 2016). These variables are subject to constraints such as internal pressure

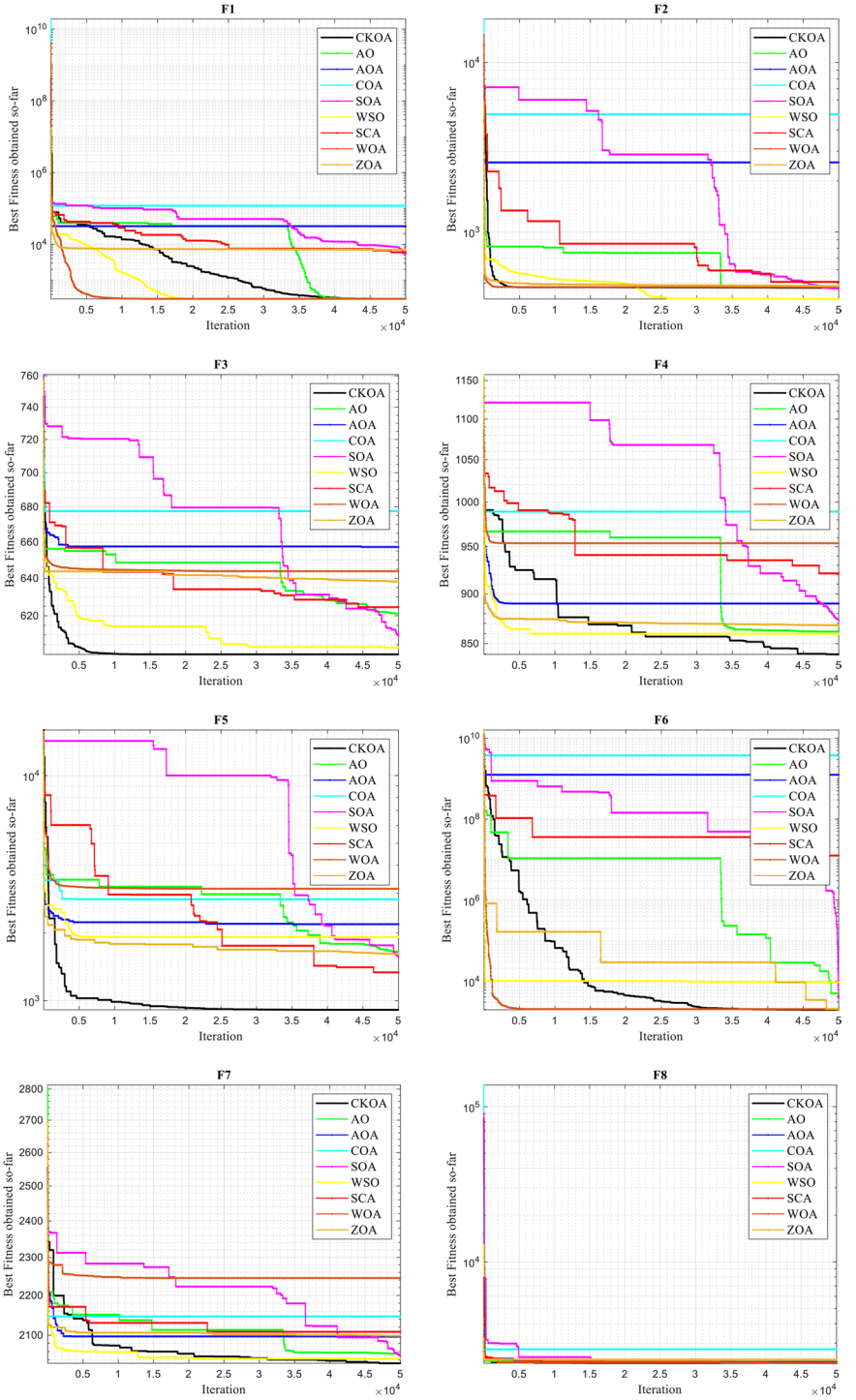

Fig. 8 TentKOA and others metaheuristic convergence curves according to CEC2022

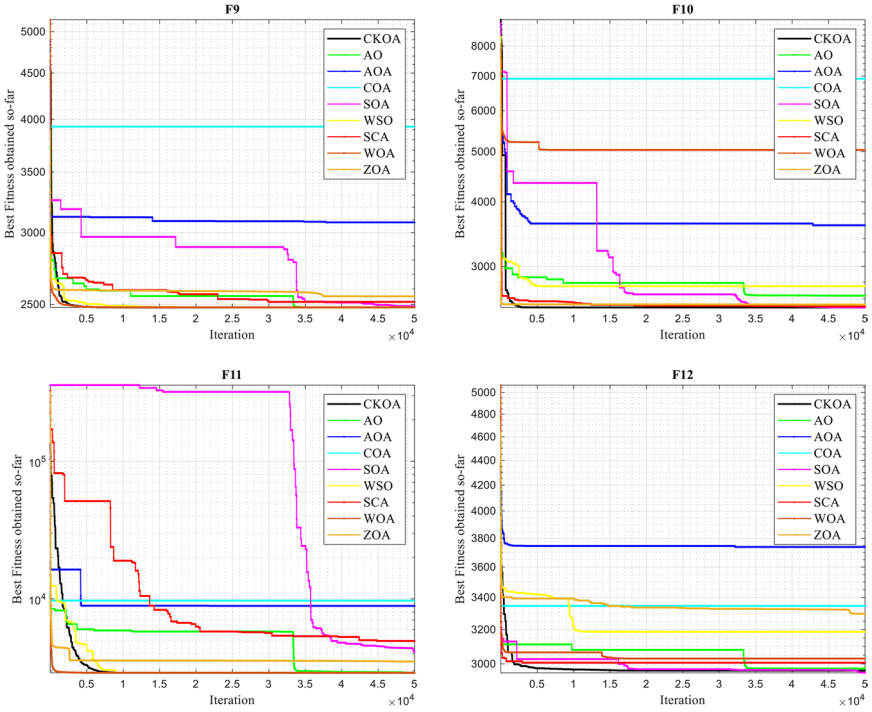

Fig. 8 (continued)

Table 10 Wilcoxon test for CEC2020 benchmark comparing TentKOA and other optimizers

Functions	AO	AOA	COA	SOA	WSO	SCA	WOA	ZOA
G1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
G2	0.0000	0.0010	0.0000	0.1188	0.0000	0.0000	0.0000	0.6520
G3	0.0000	0.0000	0.0000	0.0000	0.0004	0.0000	0.0000	0.0000
G4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
G5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
G6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
G7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
G8	0.0000	0.0000	0.0000	0.0000	0.0087	0.0000	0.0000	0.0000
G9	0.0000	0.0000	0.0000	0.0021	0.0000	0.0000	0.0000	0.0000
G10	0.9117	0.0000	0.0000	0.0004	0.0002	0.0000	0.0000	0.0000

resistance, permissible material constraints and load capacity (Garg 2019). The objective is to minimise a cost function that includes the volume of material used, welding cost, head forming cost and other relevant factors. PVDP is often used to test the effectiveness of metaheuristic methods (Hashim et al. 2021), due to its non-linear nature and the presence of multiple constraints.

Table 11 Wilcoxon test for CEC2022 benchmark comparing TentKOA and other optimizers

Functions	AO	AOA	COA	SOA	WSO	SCA	WOA	ZOA
F1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000
F3	0.0000	0.0000	0.0000	0.0000	0.0004	0.0000	0.0000	0.0000
F4	0.0112	0.0000	0.0000	0.0963	0.7283	0.0000	0.0000	0.5011
F5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F6	0.0000	0.0000	0.0000	0.0000	0.2458	0.0000	0.0224	0.0001
F7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F8	0.0156	0.0000	0.0000	0.8766	0.0080	0.0000	0.0000	0.0014
F9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F12	0.0000	0.3790	0.0000	0.2340	0.0000	0.0000	0.0000	0.0000

Table 18 presents the optimisation results obtained by TentKOA and eight recent metaheuristic optimisation algorithms. Table 19 shows their statistical results, which consist of the mean and standard deviation of 30 executions of each of these algorithms. These results show that the TentKOA stands out with optimal values for the variables and an optimal cost of 5.8853e+03, in addition to a competitive mean of 5885.3 and an equal standard deviation lower than any of its rivals. This underlines TentKOA's effectiveness in optimising design parameters for this type of problem, making it a preferable choice for similar applications.

7 Conclusion

This study presents an improved version of KOA called Chaotic Kepler Optimisation Alglorithm (CKOA). Inspired by the laws of planetary motion and enriched by the integration of chaos theory principles, CKOA has demonstrated a remarkable ability to overcome the challenges of convergence and diversity of solutions encountered by the original KOA. The results obtained on the CEC2020 and CEC2022 benchmarks and across three complex engineering problems underline the superiority of CKOA over several recent metaheuristic algorithms, confirming its effectiveness and robustness. However, despite its advantages, the CKOA also has certain limitations: its results are influenced by the choice of chaotic parameters, which requires precise calibration for each type of problem. In addition, although its convergence speed has been improved, there is still room for improvement. And finally, although the proposed method effectively combines exploration and exploitation, it can still face obstacles in extremely complex, large-scale research environments. For future work, the performance of CKOA can be improved by other concepts and adapted to other multidisciplinary optimisation problems.

Table 12 Performance index for CEC2020

Function		CKOA	AO	AOA	COA	SOA	WSO	SCA	WOA	ZOA
G1	Fitness	100	13.35e+4	2.53e+10	3.12e+10	0.19e+10	3712.41	0.35e+10	3848.08	0.51e+10
	Time	2.84	26.18	11.98	22.93	13.88	10.30	13.29	9.31	18.31
G2	Fitness	2456.90	2063.63	2986.92	5708.71	2668.31	1966.70	4401.21	3548.89	2660.47
	Time	1.67	18.80	9.08	17.73	9.75	7.6	9.60	7.25	14.35
G3	Fitness	752.14	783.09	922.45	1035.41	833.78	768.31	876.45	923.29	863.28
	Time	1.66	17.72	8.63	16.31	9.18	7.36	8.85	6.68	13.57
G4	Fitness	1903.65	1900	1900	1900	1900	1900.45	1900	1900	1900
	Time	1.59	16.21	7.89	14.47	8.56	6.57	8.26	6.03	12.21
G5	Fitness	5780.04	67,679.74	1,418,302	6,774,720	165,744	14,671.26	331,446	293,028	379,644
	Time	1.66	18.28	8.67	16.96	9.49	7.45	9.25	6.99	13.85
G6	Fitness	1607.66	1767.18	2452.42	3398.88	1855.73	1790.95	2026.57	2095.11	2201.12
	Time	1.64	17.05	7.98	15.27	8.97	6.73	8.77	6.44	12.50
G7	Fitness	2768.23	10.79e+4	265.04e+4	248.39e+4	14,236.79	8387.88	11.38e+4	16.67e+4	12.69e+4
	Time	1.69	17.901	8.25	16.13	9.17	6.90	9.23	6.75	13.21
G8	Fitness	2300.53	2304.61	5149.92	6320.69	4762.65	2303.12	3235.85	3858.78	3007.08
	Time	1.83	27.26	13.10	27.95	14.03	11.93	13.76	11.37	22.66
G9	Fitness	2832.90	2875.01	3282.77	3411.15	2849.08	3066.71	2958.10	2967.72	3154.22
	Time	2.83	46.13	22.37	49.01	22.99	19.46	22.43	20.05	40.35
G10	Fitness	2935.64	2943.03	4783.49	5975.23	2974.97	2980.33	3048.85	2990.21	3169.60
	Time	2.52	35.89	18.22	35.95	18.61	15.84	18.32	14.89	29.82

Table 13 Performance index for CEC2022

Function		CKOA	AO	AOA	COA	SOA	WSO	SCA	WOA	ZOA
F1	Fitness	300.27	301.91	23,743.62	69,385.69	3864.68	300	4558.55	314.26	8362.78
	Time	1.85	16.64	8.05	14.9	8.95	6.44	8.08	5.9	11.63
F2	Fitness	445.75	452.64	1769.52	3104.15	513.28	429.39	567.84	459.02	564.69
	Time	1.57	14.82	7.09	12.65	7.87	5.65	7.63	5.19	10.40
F3	Fitness	600.00	620.23	645.85	683.44	615.91	603.59	629.86	653.79	642.74
	Time	1.78	23.57	11.32	23.41	12.13	9.93	11.75	9.41	18.90
F4	Fitness	851.01	863.94	885.26	985.91	861.17	854.19	915.32	914.002	858.74
	Time	1.65	17.55	8.36	16.05	9.17	7.05	8.76	6.46	12.88
F5	Fitness	904.42	1547.02	2248.46	3547.72	1511.87	1812.08	1577.07	3124.21	1665.23
	Time	1.63	17.73	8.40	16.38	9.29	7.07	8.92	6.62	13.06
F6	Fitness	4018.04	9389.82	99.5e+7	243.13e+7	242.95e+7	4397.44	4.13e+7	6592.49	1.49e+7
	Time	1.59	15.61	7.41	13.64	8.23	6.12	7.82	5.54	11.21
F7	Fitness	2037.26	2051.41	2128.49	2218.13	2057.56	2050.58	2093.38	2169.73	2102.95
	Time	1.86	28.20	13.43	28.67	14.36	11.55	14.31	11.63	23.19
F8	Fitness	2228.06	2231.03	2246.18	2580.37	2231.78	2268.22	2242.01	2243.83	2341.92
	Time	1.91	31.17	14.83	32.46	15.86	13.71	15.71	13.09	26.23
F9	Fitness	2480.78	2481.02	3136.09	3703.5	2491.62	2478.08	2521.68	2481.14	2593.97
	Time	2.24	32.60	15.31	33.12	16.62	11.32	16.42	13.82	27.58
F10	Fitness	2514.13	2594.85	3097.68	6697.49	2534.71	2726.05	2528.21	3714.07	3321.91
	Time	3.76	49.73	23.82	50.34	25.02	20.80	25.01	20.52	40.25
F11	Fitness	2900	2919.52	7720.31	9146.45	4762.12	2904.04	5030.25	2900.91	4604.06
	Time	2.2	36.73	17.70	39.17	18.86	16.19	18.29	15.12	29.89
F12	Fitness	2949.02	2973.85	3195.19	3737.04	2947.15	3136.25	2992.38	3000.09	3414.92
	Time	1.99	34.91	16.87	38.32	17.77	14.48	17.53	15.02	30.09

Fig. 9 Schematic of WBDP

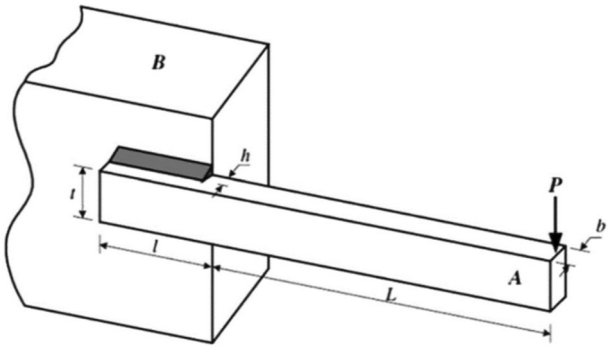

Table 14 Comparison of WBDP results

Algorithm	The optimal values of the variables				Optimal cost
	h	l	t	B	
TentKOA	0.2057	3.4705	9.0366	0.2057	1.7248
WSO	0.2057	3.4705	9.0366	0.2057	1.7249
WOA	0.1000	9.4564	9.4682	0.2036	2.2807
SCA	0.1387	6.6242	8.9746	0.2095	2.0066
SOA	0.2047	3.4772	9.2401	0.2055	1.7578
AO	0.1065	9.1985	8.9603	0.2288	2.4032
AOA	0.2057	3.4705	9.0366	0.2057	1.7249
COA	0.1268	8.0123	8.2014	0.2503	2.3167
ZOA	0.2071	3.4522	9.0073	0.2071	1.7300

Table 15 WBDP Statistical results

	TentKOA	WSO	WOA	SCA	SOA	AO	AOA	COA	ZOA
Mean	1.7248	1.7248	3.1923	1.9493	1.7666	2.1363	1.7513	2.4288	1.9568
STD	9.03e-16	3.49e–07	1.1695	0.0578	0.0186	0.1687	0.0175	0.2987	0.2631

Fig. 10 Schematic of TCSP

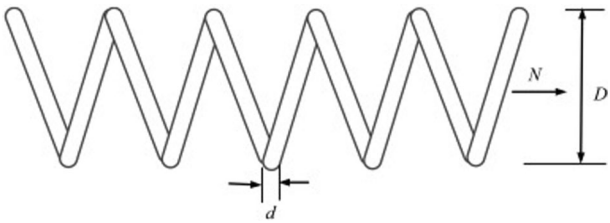

Table 16 Comparison of TCSP results

Algorithm	The optimal values of the variables			Optimal cost
	D	D	N	
TentKOA	0.0516	0.3567	11.2890	0.0126
WSO	0.0522	0.3714	10.4762	0.0127
WOA	0.0588	0.5555	5.0303	0.0127
SCA	0.0508	0.3362	12.9089	0.0129
SOA	0.0500	0.3169	14.2870	0.0129
AO	0.0686	0.8774	2.4188	0.0182
AOA	0.0529	0.3879	9.6696	0.0127
COA	0.0500	0.3161	14.2091	0.0128
ZOA	0.0528	0.3837	9.8682	0.0127

Table 17 TCSP Statistical results

	TentKOA	WSO	WOA	SCA	SOA	AO	AOA	COA	ZOA
Mean	0.0126	0.0127	0.0136	0.0132	0.0133	0.0201	0.0136	0.0144	0.0134
STD	4.96e-09	1.19e-05	0.0011	0.0006	0.0011	0.0034	0.0009	0.0035	0.0007

Fig. 11 Schematic of PVDP

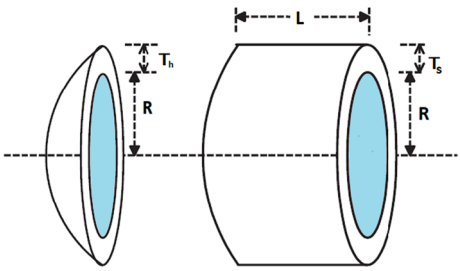

Table 18 PVDP results comparison

Algorithm	The optimal values of the variables				Optimal cost
	Ts	Th	R	L	
TentKOA	0.7781	0.3846	40.3196	1.9999e+02	5.8853e+03
WSO	0.7782	0.3847	40.3197	1.9999e+02	5.8853e+03
WOA	1.5996	0.8326	65.2252	10	1.0340e+04
SCA	1.2133	0.6793	60.6353	34.8165	7.9684e+03
SOA	1.2753	0.6044	63.2663	19.2878	7.4118e+03
AO	1.1610	0.5914	58.2913	44.2217	7.1838e+03
AOA	4.3404	0.7158	51.7242	85.2296	3.9731e+04
COA	1.2344	1.0560	61.7037	26.9025	1.0420e+04
ZOA	1.2584	0.6225	65.1853	10.1972	7.3232e+03

Table 19 PVDP Statistical results

	TentKOA	WSO	WOA	SCA	SOA	AO	AOA	COA	ZOA
Mean	5885.3	5926.9	14,136.0	7706.6	6752.9	7064.59	33,278.7	107,155	7278.2
STD	2.44e−12	87.7	12,098.3	932.1	617.5	607.38	44,063.4	71,790.1	683.99

Acknowledgements Researchers Supporting Project number (RSP2025R167), King Saud University, Riyadh, Saudi Arabia.

Author contributions NG performed the selection process and wrote the main manuscript text. All authors read and discussed the selected studies, and all authors reviewed the manuscript.

Funding This project is funded by King Saud University, Riyadh, Saudi Arabia. Researchers Supporting Project number (RSP2025R167), King Saud University, Riyadh, Saudi Arabia.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

Abd Elaziz M, Yousri D, Mirjalili S (2021) A hybrid Harris hawks-moth-flame optimization algorithm including fractional-order chaos maps and evolutionary population dynamics. Adv Eng Softw 154:102973

Abd Elminaam DS, Ibrahim SA, Houssein EH et al (2022) An efficient chaotic gradient-based optimizer for feature selection. IEEE Access 10:9271–9286

Abdel Basset M, EL-Shahat D, Jameel M et al (2023) Exponential distribution optimizer (EDO): a novel math-inspired algorithm for global optimization and engineering problems. Artif Intell Rev. https://doi.org/10.1007/s10462-023-10403-9

Abdel-Basset M, Mohamed R, Abdel Azeem SA et al (2023) Kepler optimization algorithm: a new metaheuristic algorithm inspired by Kepler's laws of planetary motion. Knowledge Based Syst. 268:110454

Abdelrazek M, Abd Elaziz M, El-Baz AH (2024) CDMO: chaotic dwarf mongoose optimization algorithm for feature selection. Sci Rep 14(1):701

Abed-alguni BH (2019) Island-based cuckoo search with highly disruptive polynomial mutation. Int J Artif Intell 17(1):57–82

Abualigah L, Yousri D, ABD Elaziz M et al (2021) Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput Indust Eng 157:107250

Ahmad A, Sirjani R (2020) Optimal placement and sizing of multi-type FACTS devices in power systems using metaheuristic optimisation techniques: an updated review. Ain Shams Eng J 11(3):611–6285

Alatas B (2010) Chaotic harmony search algorithms. Appl Math Comput 216(9):2687–2699

Alhadawi HS, Lambić D, Zolkipli MF et al (2020) Globalized firefly algorithm and chaos for designing substitution box. J Inform Sec Appl 55:102671
Azizi M, Talatahari S, Gandomi AH (2023) Fire Hawk Optimizer: a novel metaheuristic algorithm. Artif Intell Rev 56(1):287–363
Bencherqui A, Tahiri MA, Karmouni H et al (2022a) Optimization of Meixner moments by the firefly algorithm for image analysis. International conference on digital technologies and applications. Springer International Publishing, Cham, pp 439–448
Bencherqui A, Daoui A, Karmouni H et al (2022b) Optimal reconstruction and compression of signals and images by Hahn moments and artificial bee Colony (ABC) algorithm. Multimedia Tools Appl 81:29753–29783
Bencherqui A, Tamimi M, Tahiri MA et al (2023) Optimal color image watermarking based on DWT-SVD using an arithmetic optimization algorithm. International conference on digital technologies and applications. Springer Nature Switzerland, Cham, pp 441–450
Bencherqui A, Tahiri MA, Karmouni H et al (2024a) Optimal algorithm for colour medical encryption and compression images based on DNA coding and a hyperchaotic system in the moments. Eng Sci Technol Int J 50:101612
Bencherqui A, Tahiri MA, Karmouni H et al (2024b) Chaos-enhanced archimede algorithm for global optimization of real-world engineering problems and signal feature extraction. Processes 12(2):406
Biedrzycki R, Arabas J and Warchulski E (2022) A version of NL-SHADE-RSP algorithm with midpoint for CEC 2022 single objective bound constrained problems. In: 2022 IEEE Congress on Evolutionary Computation (CEC). IEEE, p. 1–8.
Boccaletti S, Grebogi C, Lai YC et al (2000) The control of chaos: theory and applications. Phys Rep 329:103–197
Braik M, Hammouri A, Atwan J et al (2022) White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowledge-Based Syst 243:108457
Cao YJ and Wu QH (1997) Evolutionary programming. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97). IEEE, p. 443–446.
Cavlak Y, Ateş A, Abualigah L, Elaziz MA (2023) Fractional-order chaotic oscillator-based Aquila optimization algorithm for maximization of the chaotic with Lorentz oscillator. Neural Comput Appl 35(29):21645–21662
Chaudhary R, Banati H (2020) Study of population partitioning techniques on efficiency of swarm algorithms. Swarm Evol Comput 55:100672
Chaudhary R, Banati H (2021) Improving convergence in swarm algorithms by controlling range of random movement. Nat Comput 20(3):513–560
Chen H, Li W, Yang X (2020) A whale optimization algorithm with chaos mechanism based on quasi-opposition for global optimization problems. Exp Syst Appl 158:113612
Cuevas E, Gálvez J, Avalos O (2020) Recent metaheuristics algorithms for parameter identification. Springer International Publishing, Cham, pp 1–8
Dehghani M, Montazeri Z, Trojovská E et al (2023) Coati optimization algorithm: a new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Syst 259:110011
Dhiman G, Kumar V (2019) Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowl-Based Syst 165:169–196
Faramarzi A, Heidarinejad M, Mirjalili S et al (2020) Marine predators algorithm: a nature-inspired metaheuristic. Exp Syst Appl 152:113377
Fatahi A, Nadimi-Shahraki MH, Zamani H (2024) An improved binary quantum-based avian navigation optimizer algorithm to select effective feature subset from medical data: a COVID-19 case study. J Bionic Eng 21(1):426–446
Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29:17–35
Garg H (2019) A hybrid GSA-GA algorithm for constrained optimization problems. Inf Sci 478:499–523
Gharehchopogh FS (2022) Quantum-inspired metaheuristic algorithms: comprehensive survey and classification. Artif Intell Rev 52:1–65
Ghorbani N, Babaei E (2014) Exchange market algorithm. Appl Soft Comput 19:177–187
Golilarz NA, Gao H, Addeh A, Pirasteh S (2020) ORCA optimization algorithm: A new meta-heuristic tool for complex optimization problems. In: 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). IEEE, p. 198–204.
Goodarzimehr V, Shojaee S, Hamzehei-Javaran S, Talatahari S (2022) Special relativity search: a novel metaheuristic method based on special relativity physics. Knowl-Based Syst 257:109484
Gupta A, Tiwari D, Kumar V et al (2022) A chaos-infused moth–flame optimizer. Arabian J Sci Eng 47(8):10769–10809

Hashim FA, Hussain K, Houssein EH et al (2021) Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl Intell 51:1531–1551
E. H. Houssein, M. K. Saeed, and M. M. AL-Sayed, "EWSO: Boosting White Shark Optimizer for solving engineering design and combinatorial problems," Mathematics and Computers in Simulation, 2023.
Jia H, Peng X, Lang C (2021) Remora optimization algorithm. Expert Syst Appl 185:115665
Jin Q, Lin N, Zhang Y (2021) K-means clustering algorithm based on chaotic adaptive artificial bee colony. Algorithms 14(2):53
Kesemen O, Özkul E, Tezel Ö, Tiryaki BK (2023) Artificial locust swarm optimization algorithm. Soft Comput 27(9):5663–5701
Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
Kumar VM, Bharatiraja C, ELrashidi A, AboRas KM (2024) Chaotic harris hawks optimization algorithm for electric vehicles charge scheduling. Energy Rep 11:4379–4396
Lan H, Xu G, Yang Y (2023) An enhanced multi-objective particle swarm optimisation with Levy flight. Int J Comput Sci Math 17(1):79–94
Li J, An Q, Lei H, Deng Q, Wang GG (2022a) Survey of lévy flight-based metaheuristics for optimization. Mathematics 10(15):2785
Li XD, Wang JS, Hao WK, Zhang M, Wang M (2022b) Chaotic arithmetic optimization algorithm. Appl Intell 52(14):16718–16757
Lu C, Gao L, Li X et al (2020) Chaotic-based grey wolf optimizer for numerical and engineering optimization problems. Memetic Comput 12:371–398
Ma B, Hu Y, Lu P, Liu Y (2022) Running city game optimizer: a game-based metaheuristic optimization algorithm for global optimization. J Comput Design Eng 10(1):65–107
Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
Mirjalili S (2018) Evolutionary algorithms and neural networks: theory and applications. Springer, Cham
Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
Modibbo UM, Singh Raghav Y, Hassan M and Mijinyawa M (2021) A critical review on the applications of optimization techniques in the un sustainable development goals. In: 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2021, pp. 572–576
Mohamed AW, Hadi AA, Mohamed AK et al (2020) Evaluating the performance of adaptive gaining sharing knowledge-based algorithm on CEC 2020 benchmark problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp. 1–8.
Naik RB, Singh U (2022) A review on applications of chaotic maps in pseudo-random number generators and encryption. Ann Data Sci. https://doi.org/10.1007/s40745-021-00364-7
Nikolić M, Šelmić M, Macura D et al (2020) Bee colony optimization metaheuristic for fuzzy membership functions tuning. Exp Syst Appl. 158:113601
Oyelade ON, Ezugwu AES, Mohamed TIA, Abualigah L (2022) Ebola optimization search algorithm: a new nature-inspired metaheuristic optimization algorithm. IEEE Access 10:16150–16177
Özbay FA (2023) A modified seahorse optimization algorithm based on chaotic maps for solving global optimization and engineering problems. Eng Sci Technol Int J 41:101408
Peng F, Hu S, Gao Z et al (2021) Chaotic particle swarm optimization algorithm with constraint handling and its application in combined bidding model. Comput Electrical Eng 95:107407
Pradeepkumar SS, Ageeskumar C, Jemilarose R (2023) An efficient SLM technique based on chaotic biogeography-based optimization algorithm for PAPR reduction in GFDM waveform. Automatika: Časopis Za Automatiku, Mjerenje, Elektroniku, Računarstvo i Komunikacije 64(1):93–103
Prajapati VK, Jain M, and Chouhan L (2020) Tabu search algorithm (TSA): a comprehensive survey. In: 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE). IEEE. p. 1–8.
Rajput SP, Datta S (2021) Application of optimization in reinforced concrete structural design-a review. Grenze Int J Eng Technol 7:27–34
Rao SS (2019) Engineering optimization: theory and practice. John Wiley & Sons, Hoboken
Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
Shami TM, El-Saleh AA, Alswaitti M et al (2022) Particle swarm optimization: a comprehensive survey. IEEE Access 10:10031–10061

Shinde V, Jha R, Mishra DK (2024) Improved Chaotic Sine cosine algorithm (ICSCA) for global optima. Int J Inf Technol 16(1):245–260
Singh AP, Kumar G, Dhillon GS, Taneja H (2023) Hybridization of chaos theory and dragonfly algorithm to maximize spatial area coverage of swarm robots. Evolut Intell. https://doi.org/10.1007/s12065-023-00823-5
Sohail A (2023) Genetic algorithms in the fields of artificial intelligence and data sciences. Ann Data Sci 10(4):1007–1018
Tahiri MA, Bencherqui A, Karmouni H et al (2023) White blood cell automatic classification using deep learning and optimized quaternion hybrid moments. Biomed Signal Proc Control 86:105128
Talatahari S, Azizi M (2020) Optimization of constrained mathematical and engineering design problems using chaos game optimization. Comput Ind Eng 145:106560
Talatahari S, Azizi M (2021) Chaos game optimization: a novel metaheuristic algorithm. Artif Intell Rev 54:917–1004
Tezel BT, Mert A (2021) A cooperative system for metaheuristic algorithms. Expert Syst Appl 165:113976
Tian Y, Zhang D, Zhang H, Zhu J, Yue X (2024) An improved cuckoo search algorithm for global optimization. Clust Comput. https://doi.org/10.1007/s10586-024-04410-w
Trojovská E, Dehghani M, Trojovský P (2022) Zebra optimization algorithm: s new bio-inspired optimization algorithm for solving optimization algorithm. IEEE Access 10:49445–49473
Trojovsky P, Dehghani M, Hanus P (2022) Siberian Tiger optimization: a new bio-inspired metaheuristic algorithm for solving engineering optimization problems. IEEE Access 10:132396–132431
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
Yancang LI, Qian YU, Zunfeng DU et al (2024) Sand cat swarm optimization algorithm and its application integrating elite decentralization and crossbar strategy. Sci Rep 14:8927
Zaimoğlu EA, Yurtay N, Demirci H, Yurtay Y (2023) A binary chaotic horse herd optimization algorithm for feature selection. Eng Sci Technol Int J 44:101453
Zamani H, Nadimi-Shahraki MH (2024) An evolutionary crow search algorithm equipped with interactive memory mechanism to optimize artificial neural network for disease diagnosis. Biomed Signal Process Control 90:105879
Zamani H, Nadimi-Shahraki MH, Qana AHG (2021) Quantum-based avian navigation optimizer algorithm. Eng Appl Artif Intell 104:104314
Zamani H, Nadimi-Shahraki MH, Qana AHG (2022) Starling murmuration optimizer: a novel bio-inspired algorithm for global and engineering optimization. Comput Methods Appl Mech Eng 392:114616
Zamani H, Nadimi-Shahraki MH, Mirjalili S, Soleimanian Gharehchopogh F, Oliva D (2024) A critical review of moth-flame optimization algorithm and its variants: structural reviewing, performance evaluation, and statistical analysis. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-023-10037-8
Zhang M, Wen G (2024) Duck swarm algorithm: theory, numerical optimization, and applications. Clust Comput. https://doi.org/10.1007/s10586-024-04293-x

Authors and Affiliations

Nawal El Ghouate[1] · Ahmed Bencherqui[1] · Hanaa Mansouri[1] · Ahmed El Maloufy[1] · Mohamed Amine Tahiri[1] · Hicham Karmouni[2] · Mhamed Sayyouri[1] · S. S. Askar[3] · Mohamed Abouhawwash[4,5]

✉ Hicham Karmouni
h.karmouni@uca.ac.ma

Nawal El Ghouate
nawal.elghouate@usmba.ac.ma

Ahmed Bencherqui
ahmed.bencherqui@usmba.ac.ma

Hanaa Mansouri
hanaa.mansouri@usmba.ac.ma

Ahmed El Maloufy
ahmed.elmaloufy@usmba.ac.ma

Mohamed Amine Tahiri
mohamedamine.tahiri@usmba.ac.ma

Mhamed Sayyouri
mhamed.sayyouri@usmba.ac.ma

S. S. Askar
saskar@ksu.edu.sa

Mohamed Abouhawwash
abouhaww@msu.edu

[1] Laboratory of Engineering, Systems and Applications, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco

[2] National School of Applied Sciences, Cadi Ayyad University, Marrakech, Morocco

[3] Department of Statistics and Operations Research, College of Science, King Saud University, P.O. Box 2455, 11451 Riyadh, Saudi Arabia

[4] Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt

[5] Department of Animal Science, Michigan State University, East Lansing, MI 48824, USA