Journal of Bionic Engineering (2023) 20:158–183
https://doi.org/10.1007/s42235-022-00255-4

RESEARCH ARTICLE

CQFFA: A Chaotic Quasi-oppositional Farmland Fertility Algorithm for Solving Engineering Optimization Problems

Farhad Soleimanian Gharehchopogh[1] · Mohammad H. Nadimi-Shahraki[2,3] · Saeid Barshandeh[4] · Benyamin Abdollahzadeh[1] · Hoda Zamani[2,3]

Received: 3 January 2022 / Revised: 20 July 2022 / Accepted: 24 July 2022 / Published online: 29 August 2022

Abstract
Farmland Fertility Algorithm (FFA) is a recent nature-inspired metaheuristic algorithm for solving optimization problems. Nevertheless, FFA has some drawbacks: slow convergence and imbalance of diversification (exploration) and intensification (exploitation). An adaptive mechanism in every algorithm can achieve a proper balance between exploration and exploitation. The literature shows that chaotic maps are incorporated into metaheuristic algorithms to eliminate these drawbacks. Therefore, in this paper, twelve chaotic maps have been embedded into FFA to find the best numbers of prospectors to increase the exploitation of the best promising solutions. Furthermore, the Quasi-Oppositional-Based Learning (QOBL) mechanism enhances the exploration speed and convergence rate; we name a CQFFA algorithm. The improvements have been made in line with the weaknesses of the FFA algorithm because the FFA algorithm has fallen into the optimal local trap in solving some complex problems or does not have sufficient ability in the intensification component. The results obtained show that the proposed CQFFA model has been significantly improved. It is applied to twenty-three widely-used test functions and compared with similar state-of-the-art algorithms statistically and visually. Also, the CQFFA algorithm has evaluated six real-world engineering problems. The experimental results showed that the CQFFA algorithm outperforms other competitor algorithms.

Keywords Nature-inspired algorithm · Farmland fertility algorithm · Chaotic maps · Quasi · Engineering optimization problems

1 Introduction

Optimization is a common problem in all fields of science and engineering problems; each problem has a wide range of issues and solutions. Optimization problems aim to find the best feasible solution(s) on the threshold of the optimal responses. Exact algorithms can find the optimal answer accurately; however, these algorithms do not hold the apt efficiency to solve challenging and complex optimization problems. The optimisation becomes more complicated with the surge in the dimensions of the problems. The execution time increases exponentially, according to the statement of the problem. Approximate algorithms are developed, finding the near-optimal solution in a considerably short time to solve challenging and complex optimization problems[1–6].

Approximate algorithms are divided into two types: heuristic and metaheuristic[7]. They are approximate methods of solving optimization problems. Heuristic algorithms often seek a near-optimal answer in a reasonable computational time. However, heuristic algorithms do not guarantee the optimal solution, get stuck in local traps, and have premature convergence. Metaheuristic algorithms have been introduced due to the weakness of heuristic algorithms. Each metaheuristic algorithm uses unique methods to get out of the local trap or prevent it from getting stuck in it. As a result, metaheuristic algorithms can find high-quality answers for all optimization problems without getting stuck in local traps[8].

✉ Farhad Soleimanian Gharehchopogh
bonab.farhad@gmail.com

[1] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia 969, Iran

[2] Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad 517, Iran

[3] Big Data Research Center, Najafabad Branch, Islamic Azad University, Najafabad 517, Iran

[4] Afagh Higher Education Institute, Urmia 969, Iran

Metaheuristic optimization algorithms have relatively simple concepts and are easy to implement. They do not need additional information and can overcome local optimums 9. These algorithms can be classified into three main categories: evolution-based methods, physics-based methods, and population-based methods. Regardless of their nature, all metaheuristic optimization algorithms have the exact mechanism. The two main phases of the search process are exploration and exploitation in population-based metaheuristic optimization algorithms. In the exploration phase, random behaviour helps search the search space. However, the primary goal is rapid exploitation in promising areas. Finding the right balance between these phases is essential since metaheuristic algorithms' nature is based on a random population. It has been shown in the literature that chaotic mappings can improve the balance between the two phases. A central and interesting query is how some chaotic mapping can improve an algorithm's performance and others do not. There is still no satisfactory theoretical framework for this analysis. This analysis presents the progress of the hybridisation of chaos and the metaheuristic algorithm. However, the hybridisation of metaheuristic algorithms with chaotic maps has been done by various researchers. It has balanced the two major and central phases of metaheuristic algorithms10.

An improved version of the Grey Wolf Optimizer (GWO) algorithm has been provided in Ref 11. to improve the exploration and exploitation. This improvement has been made through the chaotic logistic map, Opposition-Based Learning (OBL), Differential Evolution (DE), and Disruption Operator (DO). Chaotic logistics and OBL strategy are used to assign the primary candidate solutions, and these approaches address random population problems and increase algorithm convergence. The DE operators are then hybrid with the GWO algorithm. DE operators are performed as a local search mechanism to improve exploration capability by updating the GWO population. Correspondingly, after updating the solutions using hybridisation of GWO and DE, DO has been used to advance the exploration capability in which the DO has been exploited to maintain population diversity. Therefore, the hybridisation of logistic mapping, OBL, DE, and DO has offered the GWO method and apparatus to improve the search space's balance between exploration and exploitation without affecting the required computational time.

The chaos model is presented in Ref. 12 based on adaptive inertia weight to overcome the premature convergence problem in the Particle Swarm Optimization (PSO) algorithm. The initial population was generated using chaotic mapping, which improved population diversity and particle rotation, and used cube mapping sequences to assign the particles' primary position and velocity. The new inertia weight value is adjusted comparatively with the feedback parameters, including the number of iterations, the aggregation degree factor, and the evolution rate parameter. The relationship between variance, population fitness, and premature convergence threshold has prevented premature convergence. If the population fitness variance is less than the predetermined threshold, the chaos disorder surpasses the local optimal, and the optimal local problem would be solved. The improved evolution speed parameter is used as the number of feedback iterations, the aggregation factor's degree, and the adaptive inertia weight parameters. The experiments on four fitness functions show that the proposed algorithm's performance exceeds the other algorithms, improves convergence speed, and increases the ability to pass the local optimal.

Moreover, the Starling PSO algorithm, the aggregation of Starling birds, has been proposed to improve the efficiency of the global PSO search. The standard PSO performance depends on the parameter's values, and parameter adjustment is beneficial for obtaining exploration and exploitation capabilities. Nonlinear adjustment of inertial weight and chaotic search method based on logistic mapping improves the global search efficiency in the basic PSO. It helps the particles move away from the local optimal. Therefore, in Ref. 13 Chaotic Starling PSO algorithm is presented. The inertial weight is adjusted using a nonlinear decreasing approach, and the acceleration coefficients are adjusted using a chaotic logistic mapping strategy to avoid the early maturation of the search process. The speed update has used the term dynamic disruption to improve algorithm convergence14. A local search strategy based on the Starling birds' behaviour uses information from the nearest neighbours and determines a new cumulative position and rate. The two-particle selection methods maintain the Euclidean distance, and the value of the proportional function supports the particle population's diversity. These improvements aid the particles in avoiding stagnation and discovering unvisited areas in the problem space. The experiments' results on optimising the standard function and traditional clustering problems indicate this favourable efficacy method. This algorithm has better convergence features and a higher global search capability than the basic PSO and other evolutionary algorithms.

Authors in Ref. 15 present a chaos-based method for Bacterial Foraging Optimization (BFO). Adaptive Chemotaxis step settings allow fast convergence and improve convergence precision. The Chaotic perturbation operation provides the search's condition to escape local optimality and achieve better convergence precision. The proposed algorithm tested five benchmark functions, and the results show that the proposed algorithm had a better performance than the basic BFO and BFO with linear decreasing chemotaxis step.

In the present paper, first, the FFA is improved by the QOBL and chaotic-based local search mechanisms. Then,

the chaotic-based local search mechanism is utilised for twelve different chaotic functions. Improvements made using the mechanisms used complement each other. And with the FFA algorithm has shown high efficiency. Because in addition to increasing the variability using chaotic maps at all optimisation stages, the Intensification component has also been improved by using OBL. The same mechanisms have improved the FFA algorithm and have covered all the weaknesses.

The CQFFA algorithm tests 23 well-known and widely used test functions with all the chaotic functions. The results in each chaotic function are statistically compared with the other functions' results using convergence diagrams. Finally, the proposed best chaotic function algorithm is applied to several real-world engineering problems for a subtler analysis. The obtained results are compared with the results of several similar algorithms. In brief, the contribution of the paper can be summarised as follows:

- Chaotic maps have been embedded into FFA for the first time.
- The chaotic maps are used in a local search technique in the CQFFA algorithm.
- The QOBL strategy has improved the problem of trapping in local minima.
- The convergence rate is boosted using a chaotic-based local search.
- The CQFFA algorithm is tested on twenty-three different well-known test functions.
- The results of the CQFFA algorithm are compared with several similar state-of-the-art algorithms.
- The CQFFA algorithm has been applied to six real-world engineering problems.
- Experiments show the superiority of the CQFFA algorithm compared with competitor algorithms.

The present paper has been organised as follows: Sect. 2 summarises the improved algorithms using chaos maps; Sect. 3 gives a brief overview of the FFA algorithm, various chaotic functions, and mathematical models. In Sect. 4, the CQFFA algorithm is explained; Sect. 5 presents the experimental tests and findings, and finally, in the last section, conclusions and future research are discussed in detail.

2 Related Works

For scientists and researchers, there has always been the question: "How can some chaotic mappings improve the performance of an algorithm while others do not, and why the same maps do not act similarly for each algorithm?" and so far. They have not provided an excellent theoretical framework for advancing the performance of metaheuristic algorithms with chaotic maps. Therefore, by reviewing the literature, various chaos maps have been used to improve metaheuristic algorithms' versions, some of which are mentioned here.

In Ref. 16, the trapping in the optimal local problem and premature convergence in the quantum Genetic Algorithm (GA) has been solved by analysing the Piecewise Logistic Chaotic Map characteristic. Furthermore, it improves the quantum GA using the chaotic optimization approach based on piecewise logistic chaos. It optimises the quantum upgrade process by changing the angle and rotation with quantum GA's fuzzy adaptive mode. By analysing the search process of quantum GA, the optimization colony with the ergodicity of chaotic operator in the public domain, not changing the quantum GA mechanism and introducing the chaos optimization method based on Piecewise Logistic chaotic map and quantum update with fuzzy adaptation, the so-called algorithm increases the development speed of the colony and significantly increases the performance of quantum GA. The simulation results show that this method improves local and global searches, overcoming slow convergence and getting stuck in local optimization than quantum and canonical GA.

Furthermore, the solution is presented in Ref. 17 for being trapped in the local optimal, slow convergence, and inefficiency for high-dimensional problems in the chaos version of the Fruitfly optimization algorithm, whose performance is evaluated using ten chaotic maps. In the meantime, the Chebyshev map shows superiority regarding the reliability of being globally optimised and algorithm success rate. The Chaotic-based DE algorithm is presented in Ref. 18 to determine the optimal control variables for optimising the baker's yeast drying process. Instead of the random number generator, four different chaos maps, such as Lorenz, Rossler, Chua, and Mackey–Glass, have been used to determine the population's initial population in the mutation operation. The chaos-based structure for generating entities in a population is introduced instead of generating a random number. Ten optimization problems have been used to demonstrate the efficacy of Chaotic-based DE algorithms.

Additionally, Ref. 19 presents an OBL Chaotic DE algorithm. The initial population is generated according to the principles of OBL, and the dynamic matching of the F_measure factor is performed using the chaos sequence. The obtained numerical results of this algorithm on 18 benchmark functions were compared with the results of the DE algorithm and the OBL-based DE algorithm. The findings indicated that the proposed method finds superior solutions and simultaneously possesses satisfactory global search ability, stability, and convergence speed. Using chaos is among the techniques for adjusting individual parameters in metaheuristic algorithms. Therefore, in Ref. 20, a chaotic version has been provided to improve the global search in

the bat algorithm called the CBA. Four versions of the CBA have been introduced, and thirteen chaos maps have been exploited to validate each of these four versions. Comparing the various CBAs, the algorithm uses sinus mapping as pulse rate, namely CBA-IV, is considered the best type. The results show that the improvement of the new algorithms is due to the application of definite chaotic signals instead of random and constant values. Statistical results and success rates of CBAs show that adjusted algorithms can improve the reliability of general optimization. The results show that some versions of the CBA can perform better than the traditional bat one.

Ten chaos maps have been introduced in the Moth-Flame Optimization Algorithm (MSA) to surge the best prospectors to exploit the best promising solutions21. Experiments have been tested on seven benchmark functions with 30 executions. These performances' best and average modes show that chaotic maps can improve the essential MSA's efficiency in terms of convergence speed. Simultaneously, a sinusoidal map is the most preeminent map to improve the efficiency of the MSA. Four improved versions of the stochastic Fractal Search Algorithm (FSA) are presented in Ref. 22 with chaos theory to solve optimization problems and apply them in control design. The chaotic version of the Stochastic FSA performed better than the algorithm itself. It was influenced by two chaos maps (Chebyshev and Gauss/Mouse maps) that focused on the convergence speed and precision of the Stochastic FSA search results. Two standard test functions with the dimensional and landscape levels have been used to evaluate the proposed algorithm's performance with previous algorithms using the IEEE CEC2014 benchmark functions. The proposed approach has also been implemented to optimise fuzzy logic PD-type and traditional PID controllers for a Twin rotor system in hovering mode. The simulations show that the Chaotic FSA performed better with the Gauss/Mouse chaotic map in the initial upgrade and propagation process than the Chaotic FSA and Stochastic FSAs. Besides, the PD-type fuzzy logic controller in the dual-rotor system design control performed better than the PID controller.

Furthermore, in Ref. 23, ten chaotic maps are plotted on the Gravitational Search Algorithm (GSA) named Chaotic GSA (CGSA). An adaptive normalisation method is also provided for the smooth transition from the exploration phase to the exploitation phase. Twelve standard functions with bias have evaluated the efficiency of CGSA algorithms in exploration and exploitation. The results show that chaotic maps have improved GSA's exploration and exploitation phase. A statistical test called Wilcoxon has been conducted to examine the importance of results. The results show that a sinusoidal map is the best option for CGSA performance. Chaos maps improve the exploration phase since changing G's value helps the affected masses be saved from the local minimum. Chaos maps can be used to balance exploration and exploitation. Sinus map helps GSA to avoid local minimums better than other chaotic maps. The adaptive normalisation method allows the sinusoidal map to focus on exploitation instead of exploration in the final iteration.

In Ref. 24, chaos theory has been used to adjust the Crow Search Algorithm (CSA) parameters to improve the general convergence rate and exploration/exploitation trends of the CSA to solve fractional optimization problems. The CSA randomly leads to unsatisfactory solutions in some cases, which in hybridisation with the chaos method can accelerate the convergence efficiency and improve the quality of the solutions. The simulation results have been tested on 20 well-known fractional benchmark problems, enhanced by introducing chaotic maps, exploration capabilities, exploitation, and early convergence of the CSA. It works better than compared algorithms in terms of optimization. Moreover, it can find optimal global solutions for fractional problems and electrical applications. As computational speed reduces, the convergence speed reduces computational time. Also, the effectiveness of the proposed chaotic CSA has been justified using the non-parametric Wilcoxon signed-rank test. The results show that the proposed method has performed satisfactorily in terms of the quality and reliability of other algorithms.

The Grasshopper Optimization Algorithm (GOA) is an algorithm inspired by the collective behaviour of grasshoppers. In Ref. 25, the GOA optimization process introduces Chaos theory to increase its general convergence rate. Ten chaos maps have been applied to balance exploitation and exploration in the optimisation process and reduce grasshoppers' absorption/repulsion forces. The proposed methods are evaluated on thirteen test functions. The results show that chaotic maps can significantly improve GOA performance. The results also show that c_1 and c_2 cannot greatly enhance the performance of GOA. The improvement in efficiency is that the circle map provides a better balance between exploration and exploitation. The optimization process prevents it from getting stuck in the local optimal. Adjusted algorithms also improve general optimization confidence and increase the quality of the results.

Some of the other chaos-enhanced Metaheuristic algorithms are chaotic GAs26, Chaotic SA 27, Chaotic local search-based differential evolution algorithms28, Chaotic Hybrid Cognitive Optimization Algorithm29, CABC–CSA 30, Chaotic Krill Herd algorithm 31, Chaotic Salp Swarm Algorithm 32, Chaotic vortex search algorithm33, Chaotic Quantum CSA34, Chaotic Antlion Algorithm35, Chaotic PSO 36, Chaotic GWO 37. Finally, this paper presents the CQFFA algorithm to solve optimization problems, discuss the following basic concepts and proposed algorithms, and evaluates different methods and chaotic modes.

3 Fundamental Research

In this section, the basic concepts used in this paper are explained. Subsection (3.1) will briefly and usefully explain the FFA algorithm, and subsection (3.2) will describe the chaos maps using which the CQFFA is presented.

3.1 Farmland Fertility Algorithm (FFA)

FFA is a novel metaheuristic algorithm inspired by the fertility of agricultural land in nature 38. In this algorithm, the solutions are optimised based on the division of agricultural lands. With the optimal exploitation of two types of internal and external memory, the solutions of each section are optimised. According to Refs. 6, 47, the formulation and steps of this algorithm are as follows. First, the initial population is generated randomly, and the initial parameters are set. Obviously, unlike other algorithms, solutions are also divided. Then, the number of primary populations is determined by Eq. (1):

$$N = k \times n \tag{1}$$

In Eq. (1), n indicates the total number of solutions in the search space, and k indicates the number of land parts or space. n indicate the number of solutions available in each section of agricultural land. This number is a variable and an integer. The available solutions in the entire search space are evaluated according to the fitness function at this stage. FFA has a separate section for determining the value of k that specifies the optimal value for $8 \geq k \geq 2$. However, the value of k can be changed according to the optimization problem. The quality of each part of the agricultural land is obtained based on the average solutions available. Initially, the solutions are assigned to different sections by Eq. (2).

$$\begin{aligned} &\mathrm{Section_s} = \mathrm{x(aj)}, \mathrm{a} = \mathrm{n(s-1)} : \mathrm{n} \times \mathrm{s}, \\ &\mathrm{s} = \{1, 2, \ldots .\mathrm{k}\}, j = \{1, 2, \ldots .D\} \end{aligned} \tag{2}$$

In Eq. (2), the existing solutions separate each section to calculate each section's average. x is all solutions in the search space, s represents the section number, and $j = \{1,2, \ldots \mathrm{D}\}$ indicates the dimension of the variable x. After segmentation, the quality of each segment is determined by Eq. (3).

$$\begin{aligned} &Fit\_Section_s = Mean\left(all\,fit\left(x_{ji}\right) in\, Section_s\right) \\ &s = \{1.2. \ldots .k\}.i = \{1.2. \ldots .n\} \end{aligned} \tag{3}$$

In Eq. (3), *Fit_Section* indicates the value that determines the quality of the solutions in each part of the agricultural land, which in the search space is the same as the average fit or suitability of all the solutions in each part. Therefore, for each agricultural land sector, the average of all solutions within each sector is obtained and stored in $Fit\_Section_s$. Equation (4) is used to update local memory, and Eq. (5) is used for global memory.

$$M_{Global} = \mathrm{round}(t \times N);\ 0.1 < t < 1 \tag{4}$$

$$M_{local} = \mathrm{round}(t \times n);\ 0.1 < t < 1 \tag{5}$$

In Eqs. (4) and (5), M_{Global} shows the number of solutions in global memory and M_{local} indicates the number of local memory solutions, and the solutions are based on fitness and suitability in these memories. Moreover, both memories are updated at this stage, and both are updated at this point. As the work progresses, the part with the worst quality changes the most. The worst part of agricultural land in terms of quality is that all the worst parts are combined with global memory solutions according to Eqs. (6) and (7).

$$h = \alpha.rand(-1.1) \tag{6}$$

$$Xnew = h.\left(X_{ij} - X_{MGlobal}\right) + X_{ij} \tag{7}$$

In Eq. (7), $\mathrm{X_{MGlobal}}$ is a random solution that is one of the available solutions in global memory, and α is a number between zero and one, initialised in the FFA algorithm.X_{ij} is the solution in the worst part of the agricultural land selected to apply the changes, and h is a decimal number calculated based on Eq. (6). Solutions in other sections change based on Eqs. (8) and (9).

$$h = \beta.\mathrm{rand}(0.1) \tag{8}$$

$$Xnew = h \times \left(X_{ij} - X_{uj}\right) + X_{ij} \tag{9}$$

In Eqs. (8) and (9), X_{uj} is a random solution among the available solutions in the whole search space; that is, a random path is selected among all the solutions in the sections, and β is a number between intervals zero and one, which must be initialised in the proposed first algorithm. X_{ij} is a solution from the parts (except for the worst part) that have been selected to apply the changes, and h is a decimal number that can be calculated based on Eq. (9). In this algorithm, farmers in the final stage decide to combine each of the soils within the sectors of the agricultural land based on the bests available in their local memory ($Best_{Local}$). The condition for combining with the available bests in local memory is that not all solutions in all sections are combined with their local memory, and at this stage, some of the solutions in all places are combined with the best solution, which has been found so far ($Best_{Global}$) to improve the quality of the solutions in each one of them. The hybridisation of the desired resolution with $Best_{Global}$ or $Best_{Local}$ is determined by Eq. (10).

$$H = \begin{cases} X_{new} = X_{ij} + \omega_1 \left(X_{ij} - Best_{Global}(b)\right) \ if \ Q > rand \\ \\ X_{new} = X_{ij} + rand(0.1) \times \left(X_{ij} - Best_{Local}(b)\right) \ else \end{cases} \tag{10}$$

In Eq. (10), the new solution may create in two ways. Variable X_{ij} is a solution from the parts (except for the worst part) that have been selected to apply the changes. Q is a parameter initialised between zero and one. This parameter specifies the extent to which solutions are combined with$Best_{Global}$. ω_1. The FFA algorithm's parameters are integer and must first be determined in the first algorithm, gradually decreasing in value based on the algorithm's iteration (Eq. (11)).

$$\omega_1 = \omega_1 . R_v, 0 < R_v < 1 \tag{11}$$

Correspondingly, the available solutions throughout the search space are evaluated according to the objective function. This step is done regardless of the number of sections on all the available solutions in the search space. The termination conditions are checked at the end. If the end condition is met, the algorithm ends. Otherwise, the algorithm will continue to work until the termination conditions are met.

3.2 Chaos and Chaotic Map Functions

Chaos means clutter, turmoil, and disorder. The term implies the absence of any stable structure and everyday conversation. It recognises disorganisation, inefficiency, and a negative denotation. With the advancement of scientists' attitudes and the clarification of its scientific and theoretical dimensions, today's disorder and chaos are no longer considered disorganisation and disorder; instead, the disorder is unpredictable, and dynamic phenomena have their characteristics. Disorganisation is a kind of ultimate order inside disorder, and this theory studies chaotic dynamic systems. Chaotic systems are nonlinear and dynamic systems that are very sensitive to their initial conditions. A small change in such a system's initial state causes many future changes. This phenomenon is known as the butterfly effect in chaos theory. The behaviour of chaotic systems seems random on the surface. However, there is no need for an accidental element to cause chaotic behaviour, and specific dynamic systems can exhibit chaotic behaviour. Scientists initially believed that effects are linearly the result of the causes of the main body of specific categories. However, now they emphasise the creative role of disorder and chaos and see the world as a collection of systems that operate in a self-organised and unexpected method. These systems move from order to disorder. For this reason, further improvement of metaheuristic algorithms has become a scorching new research topic[33, 39, 40].

In this paper, twelve different chaotic maps like Chebyshev map [41], Circle map [42], Iterative map [43], Intermittency map [44], Liebovich map [42], Logistic map [45], Piecewise map [46], Sawtooth map [44], Sine map [47], Singer map[48], Sinusoidal map [44], and Tent map [49] has been used. Table 1 lists the formulas and specifications of the chaos maps used in the present paper.

4 CQFFA Algorithm

This section describes the details of the proposed CQFFA algorithm. Two new mechanisms have been integrated into the necessary FFA. The first mechanism is the QOBL, an extension of the opposition operator[50–53]. The QOBL improves the exploration and exploitation capabilities of the optimization algorithm. The QOBL position can be calculated as Eq. (12).

$$QX_i^j = \begin{cases} M_i^j + r.(OX_i^j - M_i^j) & X_i^j < M_i^j \\ OX_i^j + r.(M_i^j - OX_i^j) & otherwise \end{cases}$$

$$M_i^j = (lb^j + ub^j)/2$$

$$OX_i^j = lb^j + ub^j - X_i^j \tag{12}$$

where X is an n-dimensional position vector in the problem space, r is a random number between (0,1), j is the *j*th dimension, OX_i^j are the opposite of the *i*th position. lb, and ub are the lower and upper bounds of the problem space, respectively. The QOBL mechanism has been applied to both initialisation and the main loop of the CQFFA algorithm. The second mechanism is the CLS. Chaos theory is a popular way to increase the randomness and searchability of metaheuristic algorithms. The chaos theory is used as chaotic maps in optimization algorithms. In the proposed CQFFA algorithm, the chaotic maps have been used in a CLS method. The CLS method increases exploitation capability by searching nearby places. Also, the CLS method has been applied to the best solution obtained by the algorithm. The CLS method can be formulated as Eq. (13).

$$X'_{best} = X_{best} + (C - 0.5) \times (X_i - X_j) \tag{13}$$

where, X_{best} is the current best solution obtained so far, X'_{best} is the newly generated best solution, C is the chaotic value generated by the chaotic map, X_i, and X_j are two randomly selected solutions from the population. Furthermore, in the CLS, a greedy mechanism has been used. That means, if

Table 1 The mathematical details of the chaotic maps 41–49

Definition	Name	Name
$X_{i+1} = cos\left((i.cos^{-1}\left(X_i\right)\right))$	Chebyshev map	M1
$X_{i+1} = X_i + b - (a)sin(2\pi i)mod(1), a = 0.5, b = 0.2$	Circle map	M2
$X_{i+1} = \begin{cases} \varepsilon + X_i + CX_i^n 0 < X_i \le P \\ \frac{X_i - P}{1-P} P < X_i < 1 \end{cases}$	Intermittency map	M3
$P = 0.5, C = 1.5,$ and $n = 1.6$		
$X_{i+1} = sin\left(\frac{a\pi}{X_i}\right), a \in (0, 1)$	Iterative map	M4
$X_{i+1} = \begin{cases} \alpha.X_i 0 < X_i \le P \\ \frac{P - X_i}{P_2 - P_1} P_1 < X_i \le P_2 \\ 1 - \beta\left(1 - X_i\right) P_2 < X_i \le 1 \end{cases}$	Leibovich map	M5
$\alpha = \frac{P_2}{P_1}(1 - (P_2 - P_1)), \beta = \frac{1}{P_2 - 1}((P_2 - 1) - P_1(P_2 - P_1))$		
$P_1 = 0.4, P2 = 0.6, P = 0.6$		
$X_{i+1} = a.X_i(1 - X_i), a = 4$	Logistic map	M6
$X_{i+1} = \begin{cases} \frac{X_i}{P} 0 \le X_i \le P \\ \frac{X_i - P}{0.5 - P} P \le X_i < 0.5 \\ \frac{1 - P - X_i}{0.5 - P} 0.5 \le X_i < 1 - P \\ \frac{1 - X_i}{P} 1 - P \le X_i < 1 \end{cases}$	Piecewise map	M7
$P \in (0, 0.5)$ and $P \neq 0$		
$X_{i+1} = 2X_i mod(1)$	Sawtooth map	M8
$X_{i+1} = \frac{a}{4} sin\left(\pi X_i\right), 0 < a \le 4$	Sine map	M9
$X_{i+1} = \mu(7.86X_i - 23.31X_i^2 + 28.75X_i^3 - 13.302875X_i^4), \mu \in (0.9, 1.08)$	Singer map	M10
$X_{i+1} = aX_i^2 sin\left(\pi X_i\right), a = 2.3$ and $X_0 = 0.7$	Sinusoidal map	M11
$X_{i+1} = \begin{cases} \frac{X_i}{0.7} X_i < 0.7 \\ \frac{10}{3}(1 - X_i)X_i \ge 0.7 \end{cases}$	Tent map	M12

X'_{best} has better fitness, it is replaced with the X_{best}, otherwise the X'_{best} is ignored and the X_{best} is kept.

The pseudo-code of the CQFFA algorithm has shown in Fig. 1.

In Sect. (5), the performance of the proposed CQFFA algorithm is investigated, and the results are presented. For this purpose, various standard benchmark functions and several real-world engineering problems (encoded as optimization problems). The problems are optimization issues in the experiments, and the algorithms strive to reduce the cost values. Each situation has its specifications, including boundaries, optimal points, and the number of variables expressed in the corresponding subsections.

5 Experiment Evaluations and Results

This section evaluates the CQFFA algorithm's performance on well-known test functions and several real-world engineering problems. In the first subsection, twelve widely-used chaotic maps have been applied to the CQFFA, and the results have been investigated. The results of each chaotic map have been compared with others statistically. Besides, the convergence speed of the algorithms has been reached. It is worth mentioning that the simulations were conducted on a system with a Core i7 3.1 GHz processor, 8 GB RAM, and a 2 TB hard disk. The parameter value of the CQFFA algorithm is presented in Table 2.

5.1 Experiments on Test Functions

In the CQFFA algorithm, evaluations are performed to evaluate the improvements achieved. These evaluations are described using 23 standard benchmark functions in 3 separate subsections, which are fully described below.

5.1.1 Unimodal Test Functions

This subsection evaluates the CQFFA algorithm's performance on the seven widely-used unimodal test functions. These test functions only have one optimal point and consider the exploitation capability of the algorithms. Table 3 presents the unimodal test function and details the obtained results from the different chaotic maps. Also, Fig. 2 illustrates the convergence speed of the chaotic maps on these test functions.

Set number of solutions $nSol$, the maximum number of iterations $Maxiter$, number of sections $nSec$, and the values of other parameters
% Initialising solutions
For i = 1: $nSol$
    Randomly initialise *X(i)*
    Calculate $F(i)$.
    Initialise $QX(i)$ by Equation (12). % Quasi-oppositional learning
    Calculate $QF(i)$.
End
Merge X and QX in DX.
Sort DX according to the F and QF.
Store first $nSol$ of the DX as X.
Divide X into *nSec* sections. % Multi-swarming part
Find the global best solution. % Elitism solution
% Main loop
For iteration = 1 to *Maxiter*
    For s = 1: $nSec$
        Determine the soil quality of the section s
        Find the local best of the section s
    End for
    For s =1: $nSec$
        If section s is the worst section
            Update solutions of the section using equations (6) and (7).
            Calculate the fitness of new solutions.
            Apply greedy selection mechanism.
        Else
            Update solutions of the section using equations (8) and (9).
            Calculate the fitness of new solutions.
            Apply greedy selection mechanism.
        End if
    End
    For s =1: $nSec$
        For i = 1 : solutions of section s
            Update solutions of section s using Equation (10).
            Calculate the fitness of new solutions.
            Apply greedy selection mechanism.
        End for
    End
    For s =1: $nSec$
        Calculate quasi-opposite positions of the solutions of section s using Equation (12) % Quasi-Oppositional learning
        Merge basic solutions and quasi-opposite solutions
        Sort the set of merged solutions
        The store first $nSol$ solutions as the solutions of section s
        Find local best of section s % local elitism
    End
    Update the global best solution
    For i = 1 : K
        Generate a new global best-using Equation (13). % The CLS.
        Calculate the fitness of the new global best.
        Apply greedy selection mechanism.
    End for
End for
Return The global best solution.

Fig. 1 The pseudo-code of the CQFFA algorithm

According to the statistical results of Table 4, it can be seen that all chaotic versions of the FFA algorithm have found the minimum function values of F1 to F4 test functions. However, some chaotic maps have lost their functionality in other test functions. By examining Table 4 results in more detail, it can be noticed that the Sawtooth

Table 2 The parameter values of the algorithms

Algorithm	Parameter	Description	value
CQFFA	*nSol*	Number of Solutions	50
	nSec	Number of Sections	2
	Maxiter	Maximum number of Iteration	1000
	K	Agricultural land division number	8

and Tent maps outperform other maps. It also confirms this by counting the number of successes of each chaotic map. The success rate is the sum of the best performance of any map. The graphs in Fig. 2 indicate that the results of the different chaotic versions of the FFA algorithms are close, with a slight difference. Generally, according to Table 4 and Fig. 2, it can be inferred that the chaotic maps enhanced the capabilities of the basic FFA algorithm for solving unimodal test functions.

5.1.2 Multimodal Test Functions

The CQFFA algorithm's performance has been evaluated on this subsection's six well-known multimodal test functions. These test functions have more than one local optimum. Table 5 presents the details of the multimodal test function besides Table 6 and Fig. 3. Three represent the statistical results and convergence graphs on these test functions.

Considering the statistical results of Table 6, it can be seen that the results of all the chaotic versions of the FFA algorithm on functions F9 to F11 are equal. Furthermore, all chaotic algorithms achieved the most optimal function values of F9 and F10. Besides, on F12, all chaotic maps found promising positions for the test function except Circle, Liebovitch, and Tent maps. The Logistic and Sawtooth maps achieved better results for all multimodal test functions than other maps, as mentioned in Table 6. Figure 3 shows that all chaotic FFA algorithms have converged better and achieved more optimal cost values than the basic FFA algorithm. It also illustrates that the Tent map has an excellent convergence rate. However, the final result of the Logistic map is better.

5.1.3 Fix-Dimension Test Functions

This subsection provides the results of the CQFFA algorithm on the fix-dimension test functions. The dimension of these test functions cannot change. Table 7 shows the details of the fix-dimension test functions. The statistical results and the convergence graphs of the CQFFA algorithm have been provided in Table 8 and Fig. 4, respectively.

The results of the basic FFA and CQFFA algorithms express that on the F14, F16, F18, and F19. The necessary FFA performed better. However, in F17, F20, F22, and F23, the Sawtooth map obtained better results than other maps. Considering the success rates of the chaotic maps, it can be concluded that the Sawtooth maps outperform other chaotic maps in most test functions. Therefore, the Sawtooth map was selected as the victorious map. To further investigate the CQFFA algorithm's performance in the following subsection, the FFA algorithm with Sawtooth map was applied to several well-known real-world applications. The results have been compared with similar algorithms.

5.1.4 Statistical Test

In this subsection, we used the Wilcoxon signed-rank test to show the proposed algorithm's performance better. The Tables 9, 10, 11 report the results of the Wilcoxon signed-rank test of the CQOFFA algorithm versus different chaotic maps. The test is conducted with a 5% significant level on all test functions. In the tables, the one values in the R column indicate a considerable difference. The -1 values express there is no significant difference. The zero values mean the test cannot decides the critical differences.

5.2 Engineering Optimization Problems

The following section evaluates the performance of the CQFFA algorithm in solving six engineering problems. These problems using P-Metaheuristics is one of the excellent and novel research areas. The results obtained by FFA

Table 3 Details of the unimodal test functions

	Function	Dimension	Range	F_{min}
F1	$f(x) = \sum_{i=1}^{d} x_i^2$	30	$[-100, 100]^d$	0
F2	$f(x) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	30	$[-10, 10]^d$	0
F3	$f(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100, 100]^d$	0
F4	$f(x) = max_i \{ \lvert x_i \rvert, 1 \le i \le d \}$	30	$[-100, 100]^d$	0
F5	$f(x) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$	30	$[-30, 30]^d$	0
F6	$f(x) = \sum_{i=1}^{d} \left(\lvert x_i + 0.5 \rvert \right)^2$	30	$[-100, 100]^d$	0
F7	$f(x) = \sum_{i=1}^{d} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^d$	0

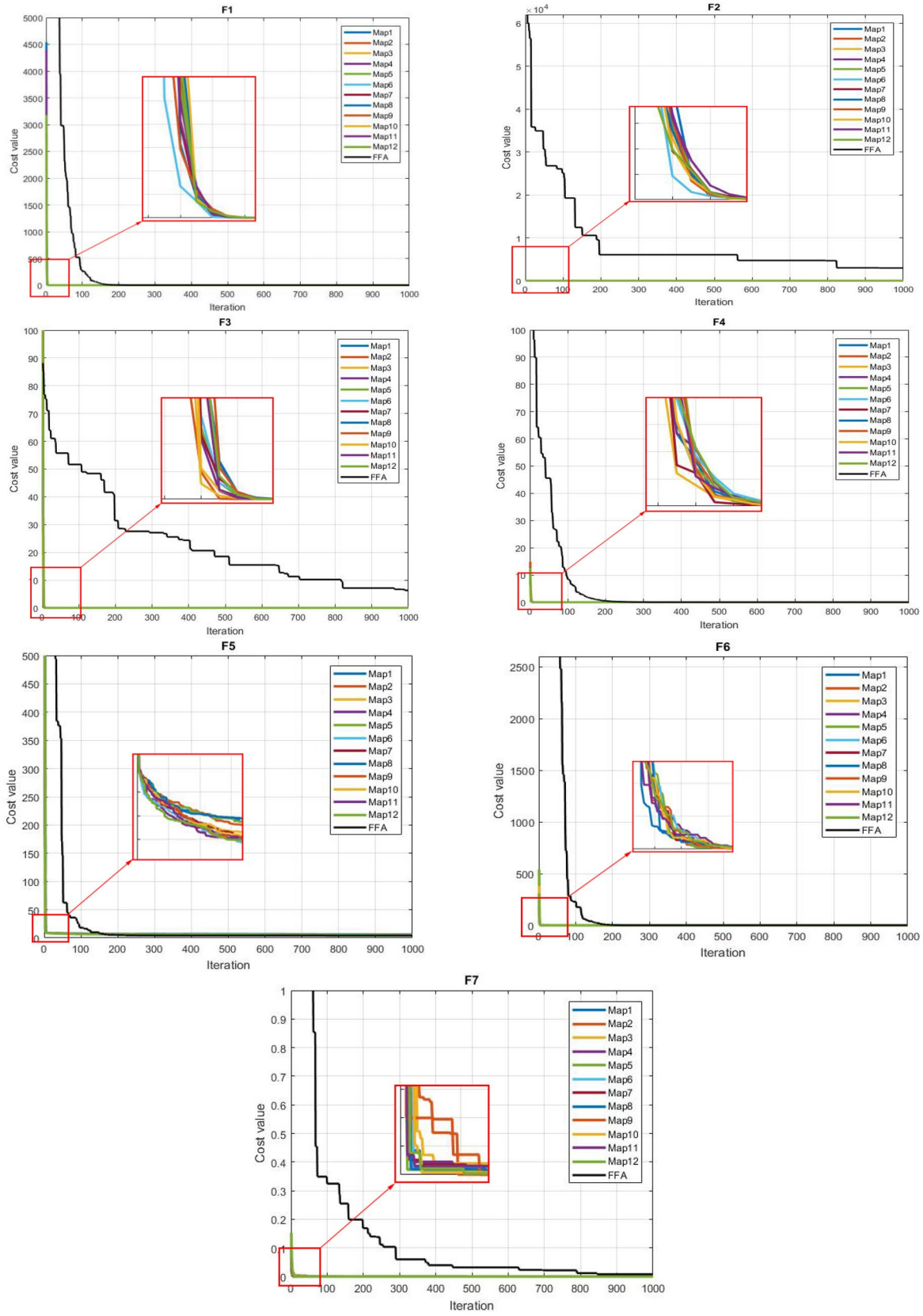

Fig. 2 Convergence graph of the different chaotic maps on the unimodal test functions

Table 4 Obtained results from different chaotic maps on unimodal test functions

		FFA	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
F1	Min	3.2718e−06	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Max	2.3104e−05	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	1.3742e−05	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	STD	5.1564e−06	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F2	Min	6.1717e−05	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Max	0.000285	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	0.000127	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	STD	5.7319e−05	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F3	Min	4992.3606	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Max	11,664.2156	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	8163.5490	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	STD	1748.7155	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F4	Min	14.7133	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Max	28.0458	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	20.9467	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	STD	3.3944	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F5	Min	28.3957	5.1485	5.73469	5.12179	5.4662	5.8457	5.2712	5.5024	5.8135	5.0897	5.2278	5.2293	**4.9348**
	Max	117.0825	6.8644	6.78604	6.52191	6.64861	6.8771	6.5684	6.8318	6.4168	6.9446	6.7980	6.5058	**6.3631**
	Mean	43.9901	6.1061	6.21265	6.07927	6.11015	6.2127	6.0743	6.1599	6.0846	6.1744	6.1368	6.0547	**6.0526**
	STD	18.1107	0.34803	0.28345	0.27106	0.3178	0.2878	0.2384	0.28086	**0.1664**	0.42292	0.3291	0.31832	0.3475
F6	Min	3.9420e−06	0	**0**	0	**0**	0	0	**0**	**0**	0	0	**0**	**0**
	Max	2.7205e−05	3.0814e−33	**0**	1.2326e−32	**0**	3.0814e−33	3.0814e−33	**0**	**0**	3.0814e−33	3.0814e−33	**0**	**0**
	Mean	1.3668e−05	1.2325e−34	**0**	6.1630e−34	**0**	2.4651e−34	1.2325e−34	**0**	**0**	1.2326e−34	1.2326e−34	**0**	**0**
	STD	5.5089e−06	6.1629e−34	**0**	2.5160e−33	**0**	8.5322e−34	6.1629e−34	**0**	**0**	6.1629e−34	6.1630e−34	**0**	**0**
F7	Min	0.02463	2.2397e−06	1.4444e−06	1.1785e−06	6.1596e−07	1.9417e−07	1.5779e−06	2.7646e−07	**1.3950e−07**	9.8043e−07	3.9128e−07	1.1465e−06	2.1790e−07
	Max	0.06063	5.0819e−05	4.1381e−05	3.9057e−05	2.9630e−05	3.1406e−05	2.7958e−05	6.3459e−05	**2.1340e−05**	4.3693e−05	5.5516e−05	5.5607e−05	4.4327e−05
	Mean	0.04093	1.4027e−05	1.7011e−05	1.3588e−05	1.3044e−05	9.4618e−06	1.1286e−05	1.1519e−05	**1.2858e−06**	1.2301e−05	1.4664e−05	1.2653e−05	1.3662e−06
	STD	0.00925	9.6081e−06	1.2683e−05	9.7042e−06	9.2083e−06	7.1261e−06	7.5363e−06	1.3716e−05	**1.1165e−06**	1.0260e−05	1.3842e−05	1.4965e−05	1.2272e−05
Sum of best result		0	4	5	4	5	4	4	5	**6**	4	4	5	**6**

Best results among all algorithms are indicated in bold

Table 5 Details of the multimodal test functions

	Function	Dimension	Range	F_{min}
F8	$f(x) = -\sum_{i=1}^{d}\left(x_i sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	30	$[-500, 500]^d$	-12,569.5
F9	$f(x) = 10d + \sum_{i=1}^{d}\left[x_i^d - 10cos\left(2\pi x_i\right)\right]$	30	$[-5.12, 5.12]^d$	0
F10	$f(x) = -20exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos 2\pi x_i\right) + 20 + e$	30	$[-32, 32]^d$	0
F11	$f(x) = \frac{1}{4000}\sum_{i=1}^{d}x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^d$	0
F12	$f(x) = \frac{\pi}{d}\left\{10\sin\left(\pi y_1\right) + \sum_{i=1}^{d-1}\left(y_i - 1\right)^2\left[1 + 10sin^2\left(\pi y_{i+1}\right)\right] + \left(y_d - 1\right)^2\right\} + \sum_{i=1}^{d}U\left(x_i, 10, 100, 4\right)$ $y_i = 1 + \frac{x_i+1}{4}$, $U\left(x_i, a, k, m\right) = \begin{cases} k\left(x_i - a\right)^m x_i > a \\ 0 - a < x_i < a \\ k\left(-x_i - a\right)^m x_i < -a \end{cases}$	30	$[-50, 50]^d$	0
F13	$f(x) = 0.1\left\{sin^2\left(3\pi x_1\right) + \sum_{i=1}^{d}\left(x_i - 1\right)^2\left[1 + sin^2\left(3\pi x_i + 1\right)\right] + \left(x_d - 1\right)^2\left[1 + sin^2\left(2\pi x_d\right)\right]\right\}$ $+\sum_{i=1}^{d}U\left(x_i, 5, 100, 4\right)$	30	$[-50, 50]^d$	0

are compared with other different optimization or modified algorithms.

5.2.1 Three-bar Truss Design Problem

The Three-bar truss engineering problems display the formulated truss and the applied forces to this structure. This problem has two design variables ($x_1 x_2$). The purpose of this problem is to minimise the total weight of the structure. Additionally, this design issue includes various limitations, such as deflection, buckling, and stress. The experiments used 50 primary populations in 500 irritations and 30 independent runs. Furthermore, since this issue has limitations, it is necessary to integrate limitation control techniques in CQFFA. The barrier penalty 54 approach has been used in CQFFA. It is expressed mathematically as Eq. (14).

$$Consider\ \vec{X} = [x_1 x_2][A_1 A_2],$$

$$Minimise\ f(\vec{X}) = \left(2\sqrt{2}X_1 + X_2\right) \times L,$$

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

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

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

$$Variable\ range\ 0 \leq x_1, x_2 \leq 1,$$

$$where L = 100cm, P = 2KN/cm^2, \sigma = 2KN/cm^2 \tag{14}$$

The results obtained by CQFFA have been compared with other optimization algorithms such as GOA55, MBA56, SSA57, PSO-DE58, DEDS59, MFO60, MVO61, Ray and Sain62, CS63, TSA64 and the results of this comparisons are shown in Table 12.

Results in Table 12 indicate that CQFFA has achieved better performance and results than other optimization algorithms. Also, according to the obtained results, it can be confirmed that CQFFA has the excellent problem-solving ability.

5.2.2 Rolling Element Bearing Design Problem

This engineering problem has ten variables and nine constraints to maximise the load-carrying capacity mathematically stated in Eq. (15).

$$Maximize\ C_d = f_c Z^{2/3} D_b^{1.8}\ if D \leq 25.4mm$$

$$C_d = 3.647 f_c Z^{2/3} D_b^{1.4}\ if D > 25.4mm$$

$$Subject\ to$$

$$g_1(\vec{z}) = \frac{\varphi_0}{2sin^{-1}(D_b/D_m)} - Z + 1 \leq 0,$$

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

Table 6 Obtained results from different maps on multimodal test functions

		FFA	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
F8	Min	−7880.481	−39,471.867	−34,507.066	−36,918.319	−35,401.525	−42,547.065	**−58,728.634**	−37,203.459	−11,907.903	−41,273.075	−55,576.752	−11,211.290	−57,027.858
	Max	−4272.748	−12,569.714	−13,798.699	−11,609.577	−13,415.789	−9805.054	**−14,762.538**	−12,008.863	−4070.713	−14,441.663	−12,954.080	−4189.828	−10,170.713
	Mean	−5730.786	−21,797.720	−21,465.799	−22,308.034	−21,877.168	−21,530.390	**−25,172.820**	−24,175.590	−8489.759	−23,481.658	−24,463.823	−8009.547	−23,618.784
	STD	**1022.6861**	6694.7384	5128.570	7250.0307	5919.359	7572.712	10,267.290	7671.290	2486.055	6884.392	8668.227	2308.316	10,203.447
F9	Min	110.3239	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Max	210.7889	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	156.1690	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	STD	28.7705	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F10	Min	0.00114	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**
	Max	0.03424	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**
	Mean	0.00426	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**	**8.8817e−16**
	STD	0.00726	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F11	Min	0.00244	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Max	0.06538	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	0.01647	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	STD	0.01343	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F12	Min	0.07350	**4.7116e−32**	4.7116e−32	**4.7116e−32**	**4.7116e−32**	4.7116e−32	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	4.7116e−32
	Max	3.1244	**4.7116e−32**	4.7116e−32	**4.7116e−32**	**4.7116e−32**	4.7358e−32	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	4.7358e−32
	Mean	0.92295	**4.7116e−32**	4.7116e−32	**4.7116e−32**	**4.7116e−32**	4.7126e−32	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7116e−32**	**4.7126e−32**	4.7126e−32
	STD	0.66970	**1.1173e−47**	4.8403e−47	**1.1173e−47**	**1.1173e−47**	4.8403e−35	**1.1173e−47**	**1.1173e−47**	**1.1173e−47**	**1.1173e−47**	**1.1173e−47**	**1.1173e−47**	4.8403e−35
F13	Min	0.09891	1.3497e−32	1.3497e−32	1.3497e−32	1.3497e−32	1.3497e−32	1.3497e−32	1.3497e−32	**1.3497e−32**	1.3497e−32	1.3497e−32	1.3497e−32	1.3497e−32
	Max	0.90751	0.01098	0.010987	0.010987	0.01098	0.0109873	0.0109874	0.0109874	**1.5842e−15**	0.0109874	0.0109874	0.0109874	0.0109874
	Mean	0.34101	0.0030764	0.001318	0.00131	0.00131	0.001318	0.00087899	0.0017579	**6.3368e−17**	0.0013184	0.0008789	0.0013185	0.0013184
	STD	0.14933	0.0050350	0.00364	0.00364	0.00364	0.003644	0.00304227	0.0041111	**3.1684e−16**	0.0036441	0.0030422	0.0036441	0.0036441
Sum of best result		0	4	3	4	4	3	**5**	4	**5**	4	4	4	3

Best results among all algorithms are indicated in bold

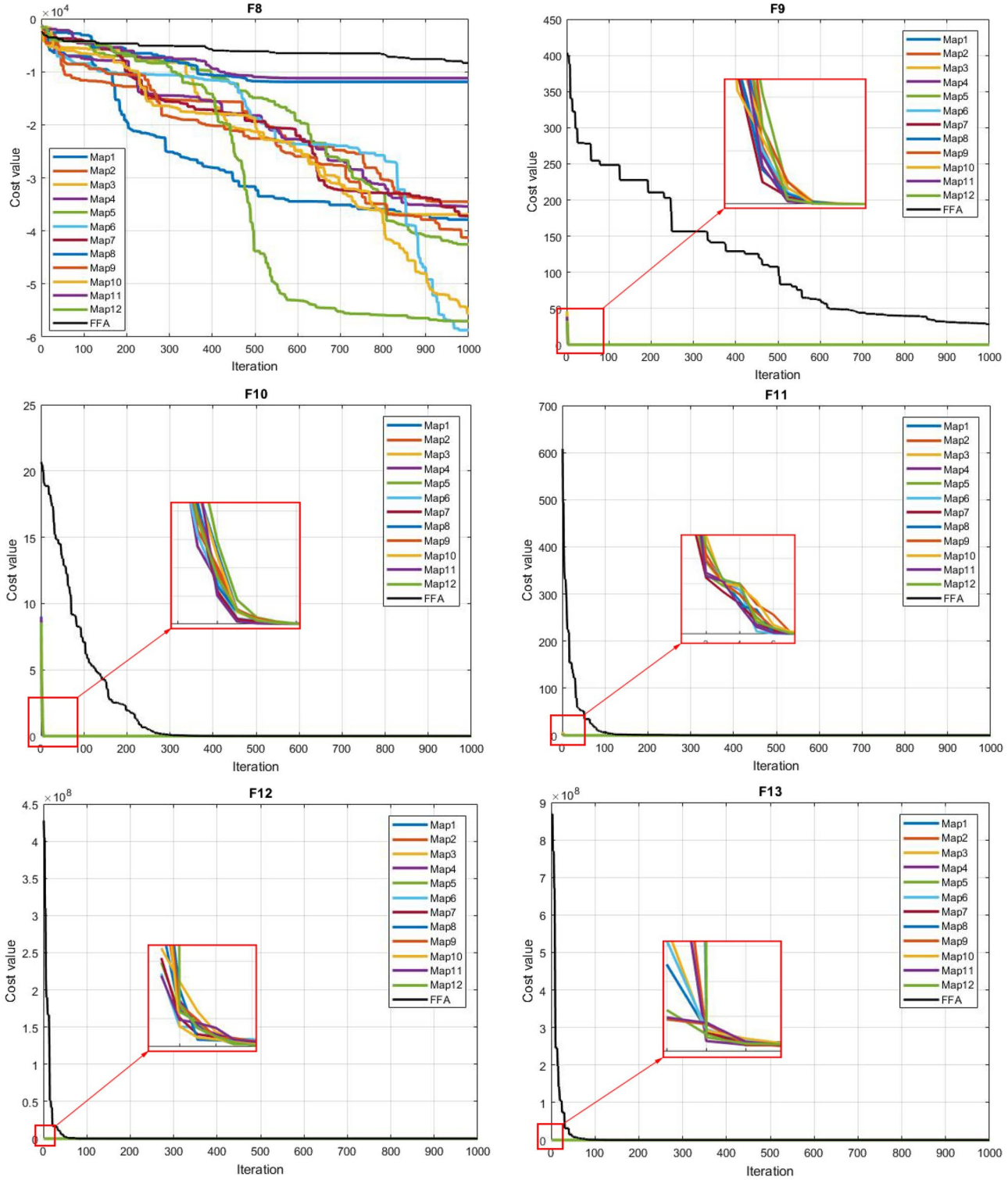

Fig. 3 Convergence graph of the different chaotic maps on the multimodal test functions

Table 7 Details of the fix-dimension test functions

	Function	Dimension	Range	F_{min}
F14	$f(x) = \left[\frac{1}{500} + \sum_{i=1}^{25} \frac{1}{i+\sum_{j=1}^{2}(x_j - a_{j,i})^6}\right]^{-1}$	2	$[-65.53, 65.53]$	0.9980
F15	$f(x) = \sum_{i=1}^{d} \left\lvert a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right\rvert^2$	4	$[-5, 5]^d$	3.0748e−04
F16	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^d$	−1.0316
F17	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	2	$[-5, 10]^d \times [0, 15]^d$	0.3979
F18	$f(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30(2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 36x_1x_2 + 27x_2^2)\right]$	2	$[-2, 2]^d$	3.0000
F19	$f(x) = -\sum_{i=1}^{4} a_i exp\left(-\sum_{j=1}^{3} b_{ij}(x_j - p_{ij})^2\right)$	3	$[0, 1]^d$	−3.86278
F20	$f(x) = -\sum_{i=1}^{4} a_i exp\left(-\sum_{j=1}^{6} b_{ij}(x_j - p_{ij})^2\right)$	6	$[0, 1]^d$	−3.322
F21	$f(x) = -\sum_{i=1}^{5} \left\lvert (x_i - a_i)(x_i - a_i)^T + c_i\right\rvert^{-1}$	4	$[0, 10]^d$	−10.1532
F22	$f(x) = -\sum_{i=1}^{7} \left\lvert (x_i - a_i)(x_i - a_i)^T + c_i\right\rvert^{-1}$	4	$[0, 10]^d$	−10.4028
F23	$f(x) = -\sum_{i=1}^{10} \left\lvert (x_i - a_i)(x_i - a_i)^T + c_i\right\rvert^{-1}$	4	$[0, 10]^d$	−10.5363

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

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

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

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

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

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

$$g_9(\vec{z}) = f_o \geq 0.515,$$

where

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

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

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

$$y = 2\{(D - d)/2 - 3(T/4)\}\{D/2 - T/4 - D_b\}$$

$$\varphi_o = 2\Pi - cos^{-1}\left(\frac{x}{y}\right), \gamma = \frac{D_b}{D_m}, f_i = \frac{r_i}{D_b}, f_o = \frac{r_o}{D_b} T = D - d - 2D_b$$

$$D = 160, d = 90, B_w = 30, r_i = r_o = 11.0330.5(D + d) \leq D_m \leq 0.6(D + d),$$

$$0.15(D - d) \leq D_b \leq 0.45(D - d), 4 \leq Z \leq 50, 0.515 \leq f_i and f_o \leq 0.6,$$

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

$$0.6 \leq \zeta \leq 0.85 \tag{15}$$

The obtained results by CQFFA compared with the optimization algorithms of PVS65, TLBO66, HHO67, and GA68, and the results are shown in Table 13.

By examining the results shown in Table 13, it was found that CQFFA has achieved much better results than the other algorithms.

Table 8 Obtained results from different maps on fix-dimension test functions

		FFA	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
F14	Min	**0.99800**	0.998003	0.998003	0.998003	0.998003	0.998003	0.998003	0.998003	0.998003	0.998003	0.998003	0.998003	0.998003
	Max	**0.99800**	10.7631	2.98210	10.76318	10.76318	2.98210	10.76318	2.982105	5.92884	2.982105	2.98210	2.98210	2.982105
	Mean	**0.99800**	1.6666	1.19649	1.42162	1.70810	1.1567	1.62742	1.315460	1.35456	1.23609	1.23609	1.3154	1.156873
	STD	**0**	2.0101	0.57287	1.95563	2.00561	0.5494	1.99273	0.742382	1.061598	0.65805	0.65805	0.74238	0.549330
F15	Min	0.000419	0.0003074	0.000307	0.0003074	**0.0003074**	0.0003074	0.0003074	0.0003074	0.0003074	0.0003074	0.0003074	0.0003074	0.0003074
	Max	0.000753	0.0003608	0.000482	0.0003842	**0.0003309**	0.0003712	0.0003354	0.00048934	0.0003341	0.0003996	0.0003961	0.0003830	0.0003621
	Mean	0.000610	0.0003185	0.0003175	0.0003163	**0.0003111**	0.0003160	0.00031279	0.00032719	0.0003112	0.0003169	0.0003144	0.0003137	0.0003126
	STD	9.7546e−05	1.7222e−05	2.8415e−05	1.9575e−05	**6.6726e−06**	1.6384e−05	7.3870e−06	4.2955e−05	6.6848e−06	2.3576e−05	1.8973e−05	1.6522e−05	1.2954e−05
F16	Min	**−1.03162**	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162
	Max	**−1.03162**	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162
	Mean	**−1.03162**	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162
	STD	**6.4098e−16**	1.0825e−15	2.7795e−11	3.1905e−12	1.4422e−12	1.0787e−13	6.7227e−16	3.1272e−12	3.2372e−12	6.4403e−10	6.3504e−13	8.8947e−13	6.5993e−16
F17	Min	0.397887	0.397887	0.397887	0.397887	0.397887	**0.397887**	0.397887	0.397887	**0.397887**	0.397887	0.397887	0.397887	0.397887
	Max	0.397887	0.397887	0.397887	0.397887	0.397887	**0.397887**	0.397887	0.397887	**0.397887**	0.397887	0.397887	0.397887	0.397887
	Mean	0.397887	0.397887	0.397887	0.397887	0.397887	**0.397887**	0.397887	0.397887	**0.397887**	0.397887	0.397887	0.397887	0.397887
	STD	7.7671e−11	1.6055e−13	1.4650e−10	2.2026e−14	1.0302e−14	**0**	1.3926e−13	9.0988e−11	**0**	1.7198e−11	2.1317e−15	3.4396e−10	2.3541e−12
F18	Min	**3.0000**	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	Max	**3.0000**	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	Mean	**3.0000**	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	STD	**6.3454e−16**	1.0493e−15	1.1546e−15	5.0111e−14	1.1662e−15	1.3627e−15	1.0912e−15	1.6616e−15	1.1727e−15	7.4199e−16	1.3353e−15	1.1749e−15	1.2429e−15
F19	Min	**−3.862782**	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782
	Max	**−3.862782**	−3.862782	−3.862782	−3.862782	−3.862782	−3.862781	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862780
	Mean	**−3.862782**	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782	−3.862782
	STD	**2.2662–15**	1.0428e−08	5.8533e−09	1.2184e−08	2.7825e−08	1.4199e−07	6.5697e−10	5.7775e−06	2.4075e−09	1.9722e−08	1.7230e−07	8.6342e−09	3.6294e−07
F20	Min	−3.3220	−3.3220	−3.3220	−3.3220	−3.3220	−3.3220	−3.3220	−3.3220	**−3.3220**	−3.3220	−3.3220	−3.3220	−3.3220
	Max	−3.32148	−3.32192	−3.32188	−3.3210	−3.3219	−3.3216	−3.3212	−3.3220	**−3.3220**	−3.3187	−3.3220	−3.3204	−3.3212
	Mean	−3.3220	−3.3220	−3.3220	−3.3219	−3.3220	−3.3220	−3.3220	−3.3220	**−3.3220**	−3.3218	−3.3220	−3.3219	−3.3219
	STD	0.000101	1.4368e−05	2.1682e−05	0.000200	1.7078e−05	7.9116e−05	1.6144e−04	6.4797e−06	**2.0039e−06**	6.5664e−04	4.9494e−06	3.1046e−04	1.6288e−04
F21	Min	−10.15319	−10.1532	−10.1532	−10.1532	**−10.1532**	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532
	Max	−10.06806	−10.1532	−10.1532	−10.1532	**−10.1532**	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532
	Mean	−10.14946	−10.1532	−10.1532	−10.1532	**−10.1532**	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532	−10.1532
	STD	0.016995	5.4389e−15	5.4390e−15	5.4389e−15	**5.4381e−15**	5.4389e−15	5.4389e−15	5.4389e−15	5.4389e−15	5.4389e−15	5.4389e−15	5.4389e−15	5.4389e−15
F22	Min	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	**−10.40294**	−10.40294	−10.40294	−10.40294	−10.4029
	Max	−10.40215	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	**−10.40294**	−10.40294	−10.40294	−10.40294	−10.4029
	Mean	−10.40286	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	−10.40294	**−10.40294**	−10.40294	−10.40294	−10.40294	−10.4029
	STD	0.000185	1.0087e−14	8.8826e−11	6.3950e−14	4.2141e−12	2.9610e−15	2.2967e−15	3.9816e−14	**1.1255e−15**	1.0902e−14	2.5121e−15	1.1919e−12	2.0188e−15
F23	Min	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	**−10.5364**	−10.5364	−10.5364	−10.5364	−10.5364
	Max	−10.35903	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	**−10.5364**	−10.5364	−10.5364	−10.5364	−10.5364
	Mean	−10.52931	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	**−10.5364**	−10.5364	−10.5364	−10.5364	−10.5364
	STD	0.035475	4.6857e−15	1.8841e−15	1.0140e−14	1.8115e−13	2.0612e−15	9.3235e−15	1.5471e−11	**1.1986e−15**	1.2376e−14	2.1755e−15	1.7389e−15	6.8606e−15
Sum of best result		**4**	0	0	0	2	1	0	0	**4**	0	0	0	0

Best results among all algorithms are indicated in bold

5.2.3 Welded Beam Design Problem

The Welded beam design problem minimises the Welded beam's cost and gets the best manufacturing cost due to the design limitations. The variables in this problem are length (l), the thickness of the weld (h), the thickness of the bar (b), and height (t). This problem is expressed mathematically as Eq. (16).

$$Consider\,\vec{z} = [z_1, z_2, z_3, z_4] = [h, l, t, b], \tag{16}$$

$$\text{Minimise} f(\vec{z}) = 1.10471 z_1^2 z_2 + 0.04811 z_3 z_4 (14.0 + z_2),$$

$$\text{Subject to } g_1(\vec{z}) = \tau(\vec{z}) - \tau_{max} \le 0,$$

$$g_2(\vec{z}) = \sigma(\vec{z}) - \sigma_{max} \le 0,$$

$$g_3(\vec{z}) = \delta(\vec{z}) - \delta_{max} \le 0,$$

$$g_4(\vec{z}) = z_1 - z_4 \le 0,$$

$$g_5(\vec{z}) = P - P_c(\vec{z}) \le 0,$$

$$g_6(\vec{z}) = 0.125 - z_1 \le 0,$$

$$g_6(\vec{z}) = 1.10471 z_1^2 + 0.04811 z_3 z_4 (14.0 + z_2) - 5.0 \le 0,$$

$$\text{Variable range } 0.05 \le z_1 \le 2.00, 0.25 \le z_2 \le 1.30, 2.00 \le z_3 \le 15.0,$$

Where

$$\tau(\vec{z}) = \sqrt{\tau'^2 2\tau'\tau'' \frac{z_2}{2R} + \tau''^2}, \tau' = \frac{P}{\sqrt{2} z_1 z_2}, \tau'' \frac{MR}{J}, M = P\left(L + \frac{z_2}{2}\right),$$

$$R = \sqrt{\frac{z_2^2}{4} + \left(\frac{z_1 + z_3}{2}\right)^2}, J = 2\left\{\sqrt{2} z_1 z_2 \left[\frac{z_2^2}{12} + \left(\frac{z_1 + z_3}{2}\right)^2\right]\right\},$$

$$\sigma(\vec{z}) = \frac{6PL}{z_4 z_3^2}$$

$$\delta(\vec{z}) = \frac{4PL^3}{E z_3^3 z_4}, P_c(\vec{z}) = \frac{4.013E\sqrt{\frac{z_3^2 z_4^6}{36}}}{L^2}\left(1 - \frac{z_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$$P = 6000lb, L = 14in, E = 30 \times 10^6 psi, G = 12 \times 10^6 psi,$$

Optimal results obtained from CQFFA have been compared with optimization algorithms GA1[69], HS[70], GSA[71], GA2[54], DAVID[72], APPROX[72], SIMPLEX[72], RANDOM[72], CDE[73], ESs[74] and are listed in Table 14.

Table 14 shows that CQFFA has obtained the best design-related settings with the least fitness compared to other optimization algorithms.

5.2.4 Pressure Vessel Design Problem

The primary purpose of this problem is to minimise the manufacturing cost. This issue has four limitations and four parameters. The parameters of this problem are (z_1—z_4): T_s (z_1, the thickness of the shell), T_h (z_2, the thickness of the head), r (z_3, inner radius), L (z_4, length of the section without the head). The mathematical formula for this problem is Eq. (17).

$$Consider\,\vec{z} = [z_1 z_2 z_3 z_4] = [T_s T_h R L],$$

$$Minimise\, f(\vec{z}) = 0.6224 z_1 z_3 z_4 + 1.7781 z_2 z_2^3 + 3.1661 z_1^2 z_4 + 19.84 z_1^2 z_3,$$

$$Subject\ to\ g_1(\vec{z}) = -z_1 + 0.0193 z_3 \le 0,$$

$$g_2(\vec{z}) = -z_3 + 0.00954 z_3 \le 0,$$

$$g_3(\vec{z}) = -\Pi z_3^2 z_4 - \frac{4}{3}\Pi z_3^3 + 1,296,000 \le 0,$$

$$g_4(\vec{z}) = z_4 - 240 \le 0, \tag{17}$$

The design space for this case is limited to $0 \le z_1, z_2 \le 99$, $0 \le z_3, z_4 \le 200$.

Results of solving this problem using CQFFA have been compared with other optimization algorithms such as WOA[71], BA[75], MDDE[76], CPSO[77], CSS[78], BIANCA[79], HPSO[80], G-QPSO[81], WEO[82], IACO[83], MFO[60], GA3[84], GWO[85], ESs[74], GA[69], DELC[86], Branch-bound (Sandgren) [71], Lagrangian multiplier (Kannan) [71]. Obtained results using CQFFA of other optimization algorithms are shown in Table 15.

By examining these results, it can be seen that CQFFA performs better than other algorithms in dealing with this problem, and the obtained results from CQFFA are much better than other methods.

5.2.5 Tension/Compression Spring Design

This problem's primary purpose is to minimise a spring's weight. The variables used to design this problem are the number of active coils (N), mean coil diameter (D), and wire diameter (d). Minimum deflection, shear stress, and surge

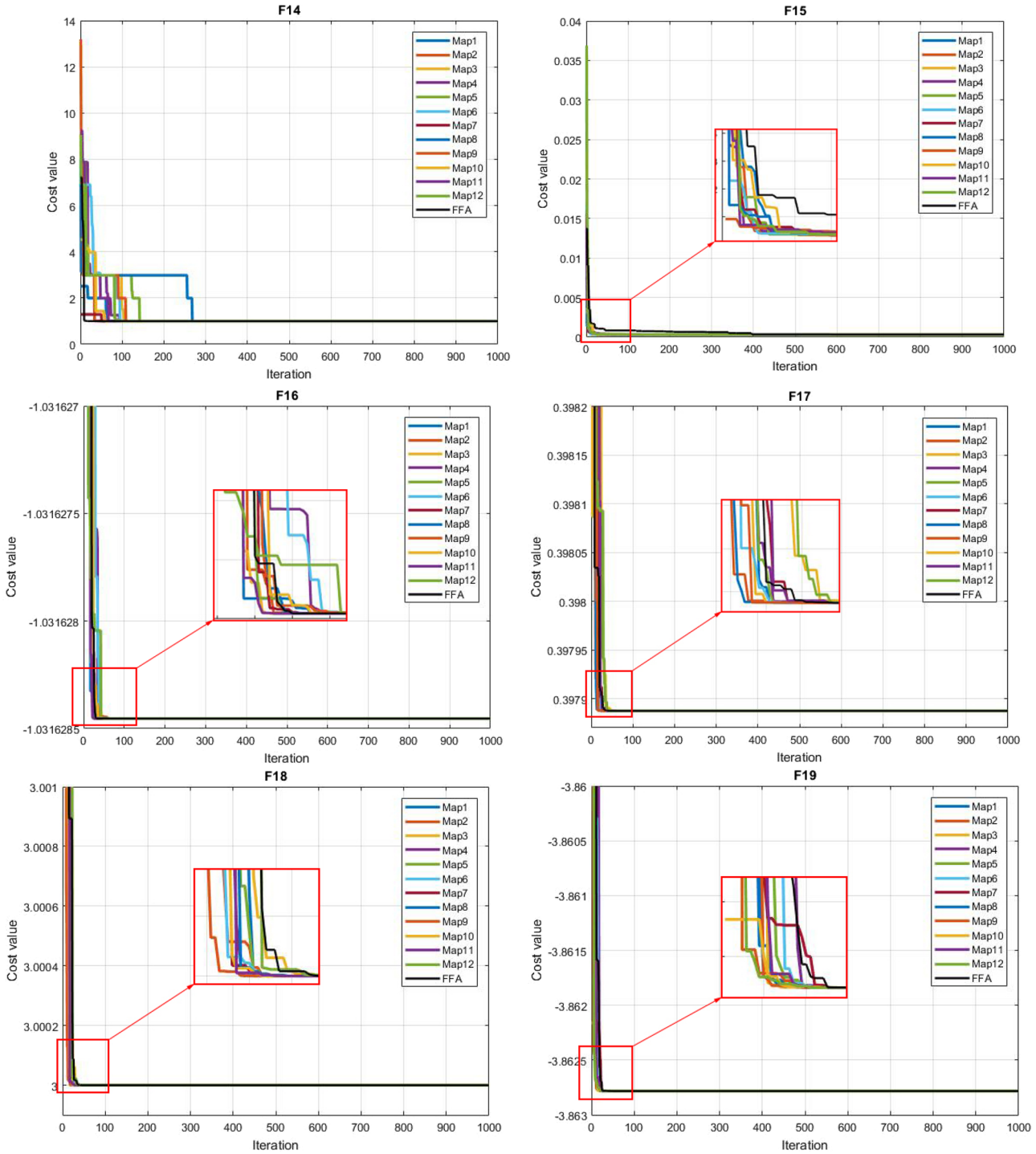

Fig. 4 Convergence graph of the different chaotic maps on the fix-dimension test functions

frequency limits should apply when solving this problem during the weight optimization process. This problem can be expressed mathematically as Eq. (18).

$$Consider \vec{z} = [z_1 z_2 z_3] = [dDN],$$

$$Minimise f(\vec{z}) = (z_3 + 2) z_2 z_1^2,$$

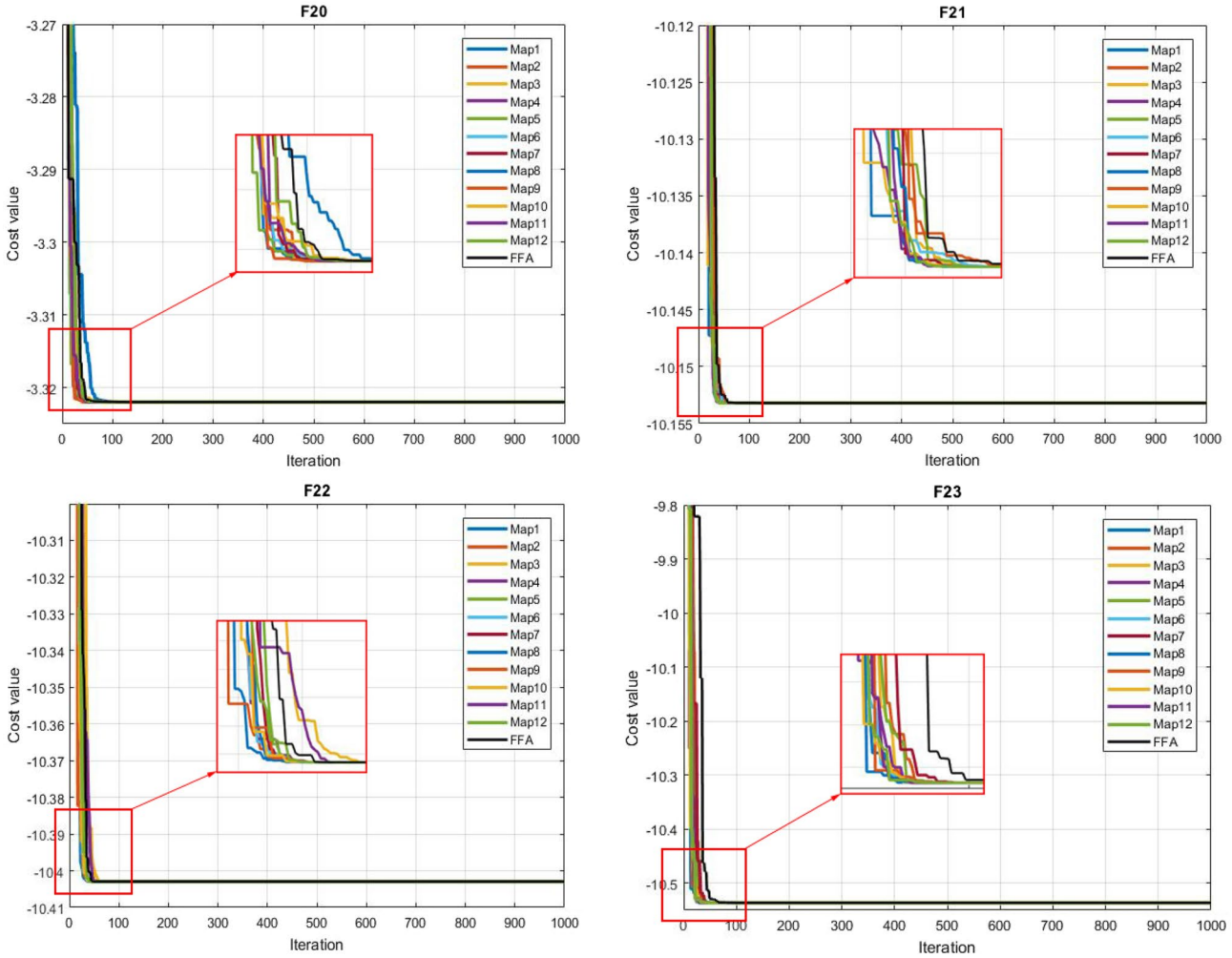

Fig. 4 (continued)

$$Subjecttog_1(\vec{z}) = 1 - \frac{z_2^3 z_3}{71785 z_1^4} \leq 0,$$

$$g_2(\vec{z}) = \frac{4z_2^2 - z_1 z_2}{12566(z_2 z_1^3 - z_1^4)} + \frac{1}{5108 z_1^2} \leq 0,$$

$$g_3(\vec{z}) = 1 - \frac{140.45 z_1}{z_2^2 z_3} \leq 0,$$

$$g_4(\vec{z}) = \frac{z_1 + z_2}{1.5} - 1 \leq 0, \tag{18}$$

To evaluate the obtained results, CQFFA has been compared with several optimization algorithms that have solved the so-called problem, such as Arora87, MFO60, GWO85, SSA57, WOA71, GSA88, ESs74, CPSO77, GA269, GA384, and WEO82. The results of this comparison are also given in Table 16.

The results in Table 16 indicate that CQFFA has a remarkable ability to produce high-quality solutions and has been able to have a good design in solving this problem. On the other hand, the results are very near and competitive compared to TEO and SFS optimization algorithms.

5.2.6 Multi-plate Disc Clutch Brake

This engineering problem's main objective is to optimise the multiple disc clutch brake's total weight for several variables, including actuating force, inner and outer radius, number of friction surfaces, and thickness of discs. It also has eight limitations. This engineering problem is expressed mathematically in Eq. (19).

$$f(x) = \Pi\left(r_o^2 - r_i^2\right) t(Z+1)\rho$$

Table 9 Wilcoxon signed-rank test results of the CQOFFA versus different chaotic maps with a 5% significance level on the unimodal test functions

	CQOFFA vs. M1	R	CQOFFA vs. M2	R	CQOFFA vs. M3	R	CQOFFA vs. M4	R	CQOFFA vs. M5	R	CQOFFA vs. M6	R
F1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F2	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F3	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F4	1.4889E−02	1	1.4303E−03	1	9.2631E−02	−1	8.2653E−02	−1	1.1547E−01	−1	1.5000E−01	−1
F5	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F6	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F7	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
	CQOFFA vs. M7	R	CQOFFA vs. M8	R	CQOFFA vs. M9	R	CQOFFA vs. M10	R	CQOFFA vs. M11	R	CQOFFA vs. M12	R
F1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F2	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F3	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F4	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F5	6.5311E−02	−1	1.8290E−01	−1	4.7967E−02	1	8.7527E−02	−1	2.8786E−01	−1	1.6584E−01	−1
F6	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F7	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1

Table 10 Wilcoxon signed-rank test results of the CQOFFA versus different chaotic maps with a 5% significance level on the multimodal test functions

	CQOFFA vs. M1	R	CQOFFA vs. M2	R	CQOFFA vs. M3	R	CQOFFA vs. M4	R	CQOFFA vs. M5	R	CQOFFA vs. M6	R
F8	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F9	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F10	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F11	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F12	6.5311E−02	−1	1.4889E−02	1	6.5311E−02	−1	1.4889E−02	1	5.1087E−02	−1	2.2586E−03	1
F13	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
	CQOFFA vs. M7	R	CQOFFA vs. M8	R	CQOFFA vs. M9	R	CQOFFA vs. M10	R	CQOFFA vs. M11	R	CQOFFA vs. M12	R
F8	1.2290E−05	1	1.3817E−02	1	1.2290E−05	1	1.2290E−05	1	2.4657E−02	1	1.2290E−05	1
F9	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F10	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F11	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F12	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F13	6.5311E−02	−1	1.2290E−05	1	1.4889E−02	1	2.2586E−03	1	4.7967E−02	1	1.4889E−02	1

Table 11 Wilcoxon signed rank test results of the CQOFFA versus different chaotic maps with a 5% significance level on the fixed-dimension test functions

	CQOFFA vs. M1	R	CQOFFA vs. M2	R	CQOFFA vs. M3	R	CQOFFA vs. M4	R	CQOFFA vs. M5	R	CQOFFA vs. M6	R
F14	3.9063E−03	1	6.2500E−02	−1	3.1250E−02	1	9.7656E−04	1	1.5625E−02	1	9.7656E−04	1
F15	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F16	1.0000E+00	0	5.0000E−01	−1	5.0000E−01	−1	1.0000E+00	0	2.5000E−01	−1	1.0000E+00	0
F14	5.0000E−01	−1	8.7500E−01	−1	5.0000E−01	−1	5.0000E−01	−1	5.0000E−01	−1	5.0000E−01	−1
F18	3.1250E−02	1	3.1250E−02	1	6.1719E−01	−1	4.8828E−04	1	3.9063E−02	1	1.5625E−02	1
F19	2.5000E−01	−1	3.1250E−02	1	7.8125E−03	1	9.7656E−04	1	3.9063E−03	1	5.0000E−01	−1
F20	3.0368E−01	−1	7.8001E−02	−1	3.6739E−01	−1	3.9137E−01	−1	1.9190E−01	−1	7.1642E−01	−1
F21	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1
F22	1.2891E−01	−1	5.4688E−01	−1	3.1250E−02	1	1.5625E−01	−1	3.1250E−02	1	3.1250E−02	1
F23	1.9531E−02	1	7.8125E−03	1	1.9531E−02	1	7.8125E−03	1	7.8125E−03	1	7.8125E−02	−1
	CQOFFA vs. M1	R	CQOFFA vs. M2	R	CQOFFA vs. M3	R	CQOFFA vs. M4	R	CQOFFA vs. M5	R	CQOFFA vs. M6	R
F14	1.9531E−03	1	4.8828E−04	1	7.8125E−03	1	3.9063E−03	1	7.8125E−03	1	7.8125E−03	1
F15	1.7735E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1	1.2290E−05	1
F16	2.5000E−01	−1	5.0000E−01	−1	1.2500E−01	−1	1.0000E+00	0	2.5000E−01	−1	1.0000E+00	0
F14	1.0000E+00	0	5.0000E−01	−1	5.0000E−01	−1	5.0000E−01	−1	1.0000E+00	0	6.2500E−01	−1
F18	3.5938E−01	−1	4.8828E−04	1	3.1250E−02	1	3.9063E−02	1	9.7656E−04	1	6.2500E−02	−1
F19	7.8125E−03	1	6.2500E−02	−1	9.7656E−04	1	1.9531E−03	1	3.9063E−03	1	4.8828E−04	1
F20	1.5000E−01	−1	4.4317E−01	−1	3.3041E−01	−1	1.5286E−01	−1	8.4007E−01	−1	5.6771E−01	−1
F21	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1	6.1035E−05	1
F22	4.6875E−02	1	1.5625E−01	−1	4.6875E−02	1	3.1250E−02	1	3.5938E−01	−1	3.1250E−02	1
F23	1.5234E−01	−1	7.8125E−03	1	7.8125E−03	1	1.5625E−02	1	7.8125E−03	1	3.5156E−02	1

Table 12 Comparison of results for the three-bar truss design problem

Algorithm	Optimal values for variables		Optimal weight
	x_1	x_2	
CQFFA	**0.7886684**	**0.4082672**	**263.8958434**
DEDS [59]	0.7886751	0.4082482	263.8958434
PSO-DE [58]	0.7886751	0.4082482	263.8958433
SSA [57]	0.7886654	0.4082757	263.8958434
MBA [56]	0.7885650	0.4085597	263.8958522
TSA [64]	0.7885416	0.4084548	263.8985569
GOA [55]	0.7888975	0.4076195	263.8958814
CS [63]	0.7886745	0.4090254	263.9716548
Ray and Sain [62]	0.7955484	0.3954792	264.3217937
MVO [61]	0.7886027	0.4084530	263.8958499
MFO [60]	0.7882447	0.4094669	263.8959797

Best results among all algorithms are indicated in bold

subject to :

$$g_1(x) = r_o - r_i - \Delta r \geq 0,$$

$$g_2(x) = l_{max} - (Z+1)(t+\delta) \geq 0,$$

$$g_3(x) = P_{max} - Prz \geq 0,$$

$$g_4(x) = P_{max} v_{srmax} - P_{rz} v_{sr} \geq 0,$$

$$g_5(x) = v_{srmax} - v_{sr} \geq 0,$$

$$g_6 = T_{max} - T \geq 0,$$

$$g_7(x) = M_h - sM_s \geq 0,$$

$$g_8(x) = T \geq 0, \tag{19}$$

Table 14 Comparison of results for the welded beam design problem

Algorithm	h	l	t	b	*Optimal cost*
CQFFA	0.20573	3.47041	9.03661	0.20573	**1.72485**
GSA [71]	0.18212	3.85698	10.0000	0.20237	1.87995
GA2 [54]	0.20880	3.42050	8.99750	0.21000	1.74831
HS [70]	0.24425	6.22316	8.29150	0.24431	2.38070
GA1 [69]	0.24890	6.17300	8.17890	0.25330	2.43311
APPROX [72]	0.24444	6.21890	8.29151	0.24440	2.38150
SIMPLEX [72]	0.27923	5.62560	7.75125	0.27960	2.53074
RANDOM [72]	0.45757	4.73134	5.08530	0.66547	4.11855
DAVID [72]	0.24340	6.25528	8.29157	0.24446	2.38410
ESs [74]	0.19974	3.61206	9.03753	0.20608	1.73730
CDE [73]	0.20313	3.54299	9.03349	0.20617	1.73346

Best results among all algorithms are indicated in bold

where,

$$M_h = \frac{2}{3}\mu FZ\frac{r_o^3 - r_i^2}{r_o^2 - r_i^3}, P_{rz} = \frac{F}{\Pi(r_o^2 - r_i^2)},$$
$$v_{rz} = \frac{2\Pi n\left(r_o^3 - r_i^3\right)}{90\left(r_o^2 - r_i^2\right)}, T = \frac{I_z \Pi n}{30(M_h + M_f)}$$

$\Delta r = 20mm$, $I_z = 55kgmm^2$, $P_{max} = 1MPa$, $F_{max} = 1000N$, $T_{max} = 15s, \mu = 0.5, s = 1.5, M_s = 40Nm$, $M_f = 3Nm, n = 250rpm$, $v_{srmax} = 10m/s, l_{max} = 30mm, r_{imin} = 60$, $r_{imax} = 80, r_{omin} = 90$, $r_{omax} = 110, t_{min} = 1.5, t_{max} = 3, F_{min} = 600$, $F_{max} = 1000, Z_{min} = 2, Z_{max} = 9.$

Table 13 Comparison of results for rolling element bearing design problem

	GA4 [68]	PVS [65]	HHO [67]	TLBO [66]	CQFFA
D_m	125.7171	125.7190	125.0000	125.7191	**1.2572E+02**
D_b	21.42300	21.42559	21.00000	21.42559	21.42330
Z	11.00000	11.00000	11.09207	11.00000	11.00115
f_i	0.515000	0.515000	0.515000	0.515000	0.515000
f_o	0.515000	0.515000	0.515000	0.515000	0.515000
K_{Dmin}	0.415900	0.400430	0.400000	0.424266	0.400155
K_{Dmax}	0.651000	0.680160	0.600000	0.633948	0.614400
ϵ	0.300043	0.300000	0.300000	0.300000	0.300000
e	0.022300	0.079990	0.050474	0.068858	0.073413
ξ	0.751000	0.700000	0.600000	0.799498	0.655940
Cost	81,843.30	81,859.74	83,011.88	81,859.74	8.5539E+04

Best results among all algorithms are indicated in bold

Table 15 Comparison of results for pressure vessel design problem

Algorithms	$T_s(x_1)$	$T_h(x_2)$	$R(x_3)$	$L(x_4)$	*Optimal cost*
CQFFA	0.778168	0.384649	40.319618	199.9900	**5.8853E + 03**
BIANCA [79]	0.812500	0.437500	42.096800	176.6580	6.0599E + 03
HPSO [80]	0.812500	0.437500	42.098400	176.6366	6.0597E + 03
CPSO [77]	0.812500	0.437500	42.091266	176.7465	6.0611E + 03
CSS [78]	0.812500	0.437500	42.103624	176.5727	6.0591E + 03
G-QPSO [81]	0.812500	0.437500	42.098415	176.6372	6.0597E + 03
WEO [82]	0.812500	0.437500	42.098444	176.6366	6.0597E + 03
IACO [83]	0.812500	0.437500	42.098353	176.6378	6.0597E + 03
GA3 [84]	0.812500	0.437500	42.097454	176.6541	6.0599E + 03
GWO [85]	0.812500	0.434500	42.089181	176.7587	6.0516E + 03
MFO [60]	0.812500	0.437500	42.098445	176.6366	6.0597E + 03
WOA [71]	0.812500	0.437500	42.098269	176.6390	6.0597E + 03
BA [75]	0.812500	0.437500	42.098445	176.6366	6.0597E + 03
MDDE [76]	0.812500	0.437500	42.098446	176.6360	6.0597E + 03
GA [69]	0.812500	0.437500	42.097398	176.6541	6.0599E + 03
DELC [86]	0.812500	0.437500	42.098445	176.6366	6.0597E + 03

Best results among all algorithms are indicated in bold

Table 16 Comparison of results for tension/compression spring problem

Algorithms	*d*	*D*	*N*	*Optimal cost*
CQFFA	0.051697	0.356909	11.277755	**0.012665**
Arora [87]	0.053396	0.399180	9.1854001	0.012730
MFO [60]	0.051994	0.364109	10.868422	0.012666
GWO [85]	0.051690	0.356737	11.288851	0.012666
SSA [57]	0.051207	0.345215	12.004032	0.012676
WOA [71]	0.051207	0.345215	12.004032	0.012676
GSA [88]	0.050276	0.323680	13.525410	0.012702
ESs [74]	0.051643	0.355360	11.397926	0.012698
CPSO [77]	0.051728	0.357644	11.244543	0.012674
GA2 [69]	0.051480	0.351661	11.632201	0.012704
GA3 [84]	0.051989	0.363965	10.890522	0.012681
WEO [82]	0.051685	0.356630	11.294103	0.012665

Best results among all algorithms are indicated in bold

Table 17 Comparison of results for multi-plate disc clutch brake

Algorithm	WCA [89]	TLBO [66]	PVS [65]	HHO [67]	CQFFA
r_i	70	70	70	70	70
r_0	90	90	90	90	90
t	1	1	1	1	1
F	910	810	980	1000	1000
Z	3	3	3	2.312781	2.312781
Optimal cost	0.313656	0.313656	0.313665	0.259768	0.259769

The optimal results obtained by CQFFA compared with the results of TLBO66, HHO67, WCA89, and PVS65 algorithms, and these results are shown in Table 17.

A review of Table 17 indicates that CQFFA performs better than TLBO, WCA, and PVS algorithms while achieving close and competitive results compared to HHO algorithms. Reviewing and evaluating the CQFFA algorithm in different benchmarks and engineering problems shows that the CQFFA has performed better than other comparative algorithms.

6 Conclusion and Future Works

In this paper, FFA's exploration and exploitation capability has improved by using twelve different chaotic maps embedded into FFA to find the best number of prospectors to increase the exploitation of the best promising solutions. Furthermore, the QOBL mechanism has enhanced the exploration and convergence rate. The CQFFA algorithm's performance has evaluated o twenty-three widely used test functions and six well-known real-world engineering problems. The results show that the CQFFA algorithm performs better and indicate that this version has a good and significant performance compared to the original version of the FFA algorithm. Because the weaknesses of FFA are well identified and well covered using the mechanisms used. Many movement search strategies can be embedded into CQFFA to improve its efficiency as much as possible in the future. Also, CQFFA can hybridise with other metaheuristics algorithms. On the other hand, the multi-objective model of the proposed algorithm can be used to solve multi-objective problems. Finally, due to the excellent potential of the proposed algorithm, it can be a perfect option for solving binary and clustering problems.

Data Availability Source code supporting this research's findings will be shared when this manuscript is accepted for publication or the Editor/reviewer is requested to share. In this paper, we used a standard benchmark (those are in Tables 3, 5, 7) to implement, evaluate and compare the proposed method with other algorithms.

Declarations

Conflict of interest The author declares that they have no conflict of interest.

References

1. Nadimi-Shahraki, M. H., Taghian, S., Mirjalili, S., Ewees, A. A., Abualigah, L., & Abd Elaziz, M. (2021). MTV-MFO: Multi-trial vector-based moth-flame optimization algorithm. *Symmetry, 13*(12), 2388.
2. Goldanloo, M. J., & Gharehchopogh, F. S. (2021). A hybrid OBL-based firefly algorithm with symbiotic organisms search algorithm for solving continuous optimization problems. *The Journal of Supercomputing, 3*, 1–34.
3. Zamani, H., Nadimi-Shahraki, M. H., & Gandomi, A. H. (2021). QANA: Quantum-based avian navigation optimizer algorithm. *Engineering Applications of Artificial Intelligence, 104*(104314), 2021.
4. Ghafori, S., & Gharehchopogh, F. S. (2021). Advances in spotted hyena optimizer: a comprehensive survey. *Archives of Computational Methods in Engineering, 29*, 1–22.
5. Banaie-Dezfouli, M., Nadimi-Shahraki, M. H., & Beheshti, Z. (2021). R-GWO: representative-based grey wolf optimizer for solving engineering problems. *Applied Soft Computing, 106*, 107328.
6. Zaman, H. R. R., & Gharehchopogh, F. S. (2021). An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems. *Engineering with Computers, 5*, 1–35.
7. Gharehchopogh, F. S. (2022). Advances in tree seed algorithm: a comprehensive survey. *Archives of Computational Methods in Engineering, 7*, 1–24.
8. Gharehchopogh, F. S. (2022). An improved tunicate swarm algorithm with best-random mutation strategy for global optimization problems. *Journal of Bionic Engineering, 48*, 1–26.
9. Gharehchopogh, F. S., & Gholizadeh, H. (2019). A comprehensive survey: whale optimization algorithm and its applications. *Swarm and Evolutionary Computation, 48*, 1–24.
10. Xu, Z., Yang, Y., Li, J., Zhang, X., Lu, B., & Gao, S. (2021). Comparative study on single and multiple chaotic maps incorporated grey wolf optimization algorithms. *IEEE Access, 9*, 77416–77437.
11. Ibrahim, R. A., Elaziz, M. A., & Lu, S. (2018). (2018) Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization. *Expert Systems with Applications, 108*, 1–27.
12. Li,J., Cheng, Y.M., & Chen,K.Z. (2014) Chaotic particle swarm optimization algorithm based on adaptive inertia weight, in *Control and Decision Conference (2014 CCDC), The 26th Chinese*, 2014: IEEE, 1310-1315.
13. Wang, L., Liu, X., Sun, M., Qu, J., & Wei, Y. (2018). A new chaotic starling particle swarm optimization algorithm for clustering problems. *Mathematical Problems in Engineering., 2018*, 1–14.
14. Gao, S., Zhou, M., Wang, Y., Cheng, J., Yachi, H., & Wang, J. (2018). Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction. *IEEE Transactions on Neural Networks and Learning Systems, 30*(2), 601–614.
15. Zhang, Y. T., Zhou, W., & Yi, J. (2016) A novel adaptive chaotic bacterial foraging optimization algorithm. *In 2016 international conference on computational modeling, simulation and applied mathematics.*
16. Teng, H., & Cao, A. (2011). 2011) An novel quantum genetic algorithm with Piecewise Logistic chaotic map, In *Natural Computation (ICNC. Seventh International Conference on, 2*, 1053–1057.
17. Mitic, M., Vukovic, N., Petrovic, M., & Miljkovic, Z. (2015). Chaotic fruit fly optimization algorithm. *Knowledge-Based Systems, 89*, 446–458.
18. Yuzgec, U., & Eser, M. (2018). Chaotic based differential evolution algorithm for optimization of baker's yeast drying process. *Egyptian Informatics Journal., 19*, 151–163.
19. Thangaraj, R., Pant, M., Chelliah, T.R. & Abraham, A. (2012) Opposition based chaotic differential evolution algorithm for solving global optimization problems, *2012 fourth world congress on nature and biologically inspired computing (NaBIC),* 1–7.
20. Gandomi, A. H., & Yang, X. S. (2014). Chaotic bat algorithm. *Journal of computational science, 5*(2), 224–232.
21. Guvenc, U., Duman, S., & Hinislioglu, Y. (2017). Chaotic Moth Swarm Algorithm. *IEEE International Conference on Inovations in Intelligent Systems and Applications (INISTA), 2017*, 90–95.
22. Rahman, T. A., Arry, A., Jalil, N. A. A., & Ahmad, R. M. K. R. (2017). Chaotic fractal search algorithm for global optimization with application to control design. *IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2017*, 111–116.
23. Mirjalili, S., & Gandomi, A. H. (2017). Chaotic gravitational constants for the gravitational search algorithm. *Applied Soft Computing, 53*, 407–419.
24. Rizk-Allah, R. M., Hassanien, A. E., & Bhattacharyya, S. (2018). Chaotic crow search algorithm for fractional optimization problems. *Applied Soft Computing, 71*, 1161–1175.
25. Arora, S., & Anand, P. (2018). Chaotic grasshopper optimization algorithm for global optimization. *Neural Computing and Applications, 31*, 1–21.
26. Yao, J. F., Mei, C., Peng, X. Q., Hu, Z. K., & Hu, J. (2001). A new optimization approach-chaos genetic algorithm. *Systems Engineering, 1*, 015.
27. Mingjun, J., & Huanwen, T. (2004). Application of chaos in simulated annealing. *Chaos, Solitons & Fractals, 21*(4), 933–941.
28. Gao, S., Yu, Y., Wang, Y., Wang, J., Cheng, J., & Zhou, M. (2019). Chaotic local search-based differential evolution algorithms for optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51*(6), 3954–3967.
29. Zhou, Y., Su, K., & Shao, L. (2018). A new chaotic hybrid cognitive optimization algorithm. *Cognitive Systems Research, 52*, 537–542.
30. Chahkandi, V., Yaghoobi, M., & Veisi, G. (2013). CABC–CSA: a new chaotic hybrid algorithm for solving optimization problems. *Nonlinear Dynamics, 73*(12), 475–484.
31. Wang, G. G., Guo, L., Gandomi, A. H., Hao, G. H., & Wang, H. (2014). Chaotic krill herd algorithm. *Information Sciences, 274*, 17–34.
32. Sayed, G. I., Khoriba, G. M., & Haggag, H. (2018). A novel chaotic salp swarm algorithm for global optimization and feature selection. *Applied Intelligence, 48*, 1–20.
33. Gharehchopogh, F. S., Maleki, I., & Dizaji, Z. A. (2021). Chaotic vortex search algorithm: metaheuristic algorithm for feature selection. *Evolutionary Intelligence, 15*, 1–32.
34. Boushaki, S. I., Kamel, N., & Bendjeghaba, O. (2018). A new quantum chaotic cuckoo search algorithm for data clustering. *Expert Systems with Applications, 96*, 358–372.
35. Tharwat, A., & Hassanien, A. E. (2018). Chaotic antlion algorithm for parameter optimization of support vector machine. *Applied Intelligence, 48*(3), 670–686.

36. Xu, X., Rong, H., Trovati, M., Liptrott, M., & Bessis, N. (2018). CS-PSO: Chaotic particle swarm optimization algorithm for solving combinatorial optimization problems. *Soft Computing, 22*(3), 783–795.
37. Kohli, M., & Arora, S. (2018). Chaotic grey wolf optimization algorithm for constrained optimization problems. *Journal of Computational Design and Engineering, 5*(4), 458–472.
38. Shayanfar, H., & Gharehchopogh, F. S. (2018). Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. *Applied Soft Computing, 71*, 728–746.
39. Yuan, X., Zhao, J., Yang, Y., & Wang, Y. (2014). Hybrid parallel chaos optimization algorithm with harmony search algorithm. *Applied Soft Computing, 17*, 12–22.
40. Asghari, K., Masdari, M., Gharehchopogh, F. S., & Saneifard, R. (2021). A chaotic and hybrid gray wolf-whale algorithm for solving continuous optimization problems. *Progress in Artificial Intelligence, 10*, 1–26.
41. Geisel, T., & Fairen, V. (1984). Statistical properties of chaos in Chebyshev maps. *Physics Letters A, 105*(6), 263–266.
42. Hilborn, R. C. (2000). *Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers*. Oxford University Press.
43. Lauwerier, H. (1989). Two-dimensional iterative maps. *Chaos, 2*, 58–95.
44. Barshandeh, S., & Haghzadeh, M. (2020). A new hybrid chaotic atom search optimization based on tree-seed algorithm and Levy flight for solving optimization problems. *Engineering with Computers, 4*, 1–44.
45. Li, Y., Deng, S., & Xiao, D. (2011). A novel Hash algorithm construction based on chaotic neural network. *Neural Computing and Applications, 20*(1), 133–141.
46. Tomida, A.G. (2008) Matlab toolbox and GUI for analyzing one-dimensional chaotic maps, *International conference on computational sciences and its application,* 321–330.
47. Devaney, R. (2008). *An Introduction to Chaotic Dynamical Systems*. Westview: Westview press.
48. Peitgen,H.O, Jurgens, H., & Saupe,D. (2006) *Chaos and fractals: new frontiers of science*. Springer science & business media.
49. Ott, E. (2002). *Chaos in Dynamical Systems*. Cambridge University Press.
50. Truong, K. H., Nallagownden, P., Baharudin, Z., & Vo, D. N. (2019). A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems. *Applied Soft Computing, 77*, 567–583.
51. Basu, M. (2016). Quasi-oppositional differential evolution for optimal reactive power dispatch. *International Journal of Electrical Power & Energy Systems, 78*, 29–40.
52. Warid, W., Hizam, H., Mariun, N., & Wahab, N. I. A. (2018). A novel quasi-oppositional modified Jaya algorithm for multi-objective optimal power flow solution. *Applied Soft Computing, 65*, 360–373.
53. Guha, D., Roy, P., & Banerjee, S. (2017). Quasi-oppositional symbiotic organism search algorithm applied to load frequency control. *Swarm and Evolutionary Computation, 33*, 46–67.
54. Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*(2), 113–127.
55. Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software, 105*, 30–47.
56. 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*(5), 2592–2612.
57. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114*, 163–191.
58. Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10*(2), 629–640.
59. Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences, 178*(15), 3043–3074.
60. Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.
61. Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*(2), 495–513.
62. Ray, T., & Saini, P. (2001). Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Engineering Optimization, 33*(6), 735–748.
63. 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*(1), 17–35.
64. Tsai, J. F. (2005). Global optimization of nonlinear fractional programming problems in engineering design. *Engineering Optimization, 37*(4), 399–409.
65. Savsani, P., & Savsani, V. (2016). Passing vehicle search (PVS): A novel metaheuristic algorithm. *Applied Mathematical Modelling, 40*(5–6), 3951–3978.
66. Rao, R. V., Savsani, V. J., & Vakharia, S. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43*(3), 303–315.
67. 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.
68. Gupta, S., Tiwari, R., & Nair, S. B. (2007). Multi-objective design optimisation of rolling bearings using genetic algorithms. *Mechanism and Machine Theory, 42*(10), 1418–1443.
69. Deb, K. (1991). Optimal design of a welded beam via genetic algorithms,". *AIAA Journal, 29*(11), 2013–2015.
70. Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers & Structures, 82*(9–10), 781–798.
71. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.
72. Ragsdell, K., & Phillips, D., (1976) Optimal design of a class of welded structures using geometric programming.
73. Huang, F. Z., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation, 186*(1), 340–356.
74. Mezura-Montes, E. N., & Coello, C. A. C. (2005). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary computation, 9*(1), 1–17.
75. Gandomi, A. H., Yang, X. S., Alavi, A. H., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications, 22*(6), 1239–1255.
76. Mezura-Montes, E. N., Coello, C. C., Velazquez-Reyes, J. S., & Muaoz-Da¡vila, L.A. (2007). Multiple trial vectors in differential evolution for engineering design. *Engineering optimization, 39*(5), 567–589.
77. He, Q., & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence, 20*(1), 89–99.

78. Kaveh, A., & Talatahari, S. (2010). A novel heuristic optimization method: charged system search. *Acta Mechanica, 213*(3–4), 267–289.
79. Montemurro, M., Vincenti, A., & Vannucci, P. (2013). The automatic dynamic penalisation method (ADP) for handling constraints with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 256*, 70–87.
80. He, Q., & Wang, L. (2007). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation, 186*(2), 1407–1422.
81. Coelho, L. D. S. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications, 37*(2), 1676–1683.
82. Kaveh, A., & Bakhshpoori, T. (2016). Water evaporation optimization: a novel physically inspired optimization algorithm. *Computers & Structures, 167*, 69–85.
83. Rosenbrock, H. (1960). An automatic method for finding the greatest or least value of a function. *The Computer Journal, 3*(3), 175–184.
84. Coello, C. A. C., & Montes, E. N. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16*(3), 193–203.
85. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
86. Wang, L., & Li, L. P. (2010). An effective differential evolution with level comparison for constrained engineering design. *Structural and Multidisciplinary Optimization, 41*(6), 947–963.
87. Arora, J.S.(2004) *Introduction to optimum design*. 945.
88. Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences, 179*(13), 2232–2248.
89. 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.