IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 December 2024, accepted 23 January 2025, date of publication 28 January 2025, date of current version 3 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3535667

RESEARCH ARTICLE

CMACGSA: Improved Gravitational Search Algorithm Based on Cerebellar Model Articulation Controller for Optimization

NAZMIYE EBRU BULUT[1,2], EMRE DANDIL[3], UGUR YUZGEC[3], AND ALPASLAN DUYSAK[4]

[1]Department of Electronics and Computer Engineering, Institute of Graduate, Bilecik Şeyh Edebali University, 11230 Bilecik, Türkiye
[2]Distance Education Application Research Center, Konya Technical University, 42250 Konya, Türkiye
[3]Department of Computer Engineering, Faculty of Engineering, Bilecik Şeyh Edebali University, 11230 Bilecik, Türkiye
[4]Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA

Corresponding author: Emre Dandil (emre.dandil@bilecik.edu.tr)

ABSTRACT Metaheuristic algorithms have gained significant attention in recent years for addressing complex and challenging optimization problems, especially in engineering. These algorithms often take inspiration from natural phenomena, systems or biological behaviour to find optimal solutions. Recent advances in the field often involve hybrid methods that combine several algorithms to improve performance. This study introduces an improved Gravitational Search Algorithm, named CMACGSA, which incorporates the Cerebellar Model Articulation Controller (CMAC)-a neural network model-to enhance the performance of Gravitational Search Algorithm (GSA). By employing the CMAC neural network, CMACGSA dynamically learns the masses of particles/agents of GSA, enabling a learning-driven approach to mass computation. Additional enhancements include Lévy mutation, boundary control methods and an error handling mechanism, which together improve the robustness and adaptability of the algorithm. The effectiveness of CMACGSA is demonstrated through extensive testing on a set of 2D CEC 2014 benchmark functions, where it significantly outperforms the original GSA. Further evaluations on multidimensional CEC 2014 test problems, including 30-dimensional cases, reveal improved performance over widely used optimization algorithms and state-of-the-art (SOTA) algorithms. Furthermore, CMACGSA consistently achieves top-tier average performance metrics when benchmarked against four well-established GSA variants. The applicability of the algorithm is further validated by engineering design problems where it demonstrates outstanding performance, confirming its value in solving complex engineering challenges.

INDEX TERMS Optimization, hybrid optimization methods, metaheuristic algorithms, gravitational search algorithm, cerebellar model articulation controller, engineering optimization.

I. INTRODUCTION

Optimization is the broad concept that encompasses deterministic, stochastic and other types of problem-solving techniques. Scenario-based stochastic optimization focuses on dealing with uncertainty through scenarios, often using mathematical programming or decomposition methods [1], while metaheuristic algorithms are general-purpose optimization algorithms inspired by natural or biological phenomena. Furthermore, stochastic optimization methods are widely used in cloud-based power systems [2], [3]. On the other hand, metaheuristic algorithms are becoming increasingly popular because they can be applied to engineering problems and provide fast and reliable results [4]. These algorithms are basically developed by taking inspiration from natural events, living behaviours, systems and simple processes to find optimal solutions. Metaheuristic algorithms consist of different subgroups such as biological, physical/chemical, swarm intelligence, evolution-inspired and population-based algorithms [5], [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

Biology-based algorithms are designed to mimic the structures and behaviours of biological systems. For example, Ant Colony Optimization (ACO) [7] is based on the most efficient route that ant colonies take to find food. There are many algorithms previously proposed based on biology. Some of these are Dragonfly Algorithm (DA) [8], Artificial Bee Colony (ABC) [9], Cuckoo Search (CS) [10], Grey Wolf Optimizer (GWO) [11], Whale Optimization Algorithm (WOA) [12], Bat-inspired Algorithm (BA) [13], Electric Eel Foraging Optimization (EEFO) [14], Hippopotamus Optimization Algorithm [15], Crayfish Optimization Algorithm [16].

Physics-based algorithms aim to find the best solution by imitating the physical rules and behaviors of systems. For example, Simulated Annealing (SA) [16] is based on the cooling of solid objects and the lowest energy crystallization process. The algorithm aims to find the best solution. Examples of physically based algorithms are Black Hole Algorithm (BH) [17], Charged System Search (CSS) [18], Gravitational Search Algorithm (GSA) [19], Electromagnetism-like Algorithm (EM) [20], Coulomb's Law Optimization (CLO) [18].

On the other hand, evolution-based algorithms are inspired by natural evolutionary rules such as mutation and crossover in biology. Examples of this category are Genetic Algorithm (GA) [21], Genetic Programming (GP) [22], Evolutionary Programming [23], Differential Evolution (DE) [24], Memetic Algorithms [25]. It also mimics algorithms based on swarm intelligence. In addition, cooperative movements and social behaviours of animal flocks where information is shared between individuals. Particle Swarm Optimization (PSO) [26] emerged by imitating the food-finding behavior of flocks of birds or fish. Examples of this category are Swarm Intelligence Algorithm (SI) [27], BA [13], Firefly Algorithm (FA) [28], Grasshopper Optimization Algorithm (GOA) [29], Snake Optimization (SO) [30], Spotted Hyena Optimization (SHO) [31]. In these algorithms, there are initially many randomly generated particles in the search space and then the iteration is started to find the best particle.

The versatility and adaptability of meta-heuristic algorithms make them suitable for a wide range of applications, from optimization problems to machine learning and beyond. Compared to classical optimization algorithms, metaheuristic algorithms have attracted attention in the scientific world due to their flexible structure, applicability to problems and understandability [31]. The main purpose of metaheuristic algorithms is to optimize problems effectively and accurately, in other words, to obtain the best solution. However, while optimization algorithms are suitable for certain types of problems, they may not be suitable for other problems [32]. Most optimization algorithms initially distribute random agents/particles in the search/solution space. Every agent/particle positioned in the solution space is the best solution candidate. The solution is a local optimum if some of the candidates are close to a region, that is, the minimum or maximum value of the solution in a certain range, but not the minimum or maximum value of the entire area [16]. This situation causes early convergence problem of the algorithm. There is a constant need to develop new meta-heuristic algorithms to overcome limitations and improve the performance of existing algorithms.

In this study, we propose CMACGSA, an improved GSA algorithm based on CMAC for optimization. GSA, proposed in 2009, is one of the metaheuristic optimization methods inspired by Newton's law of universal gravitation in space [19]. In GSA, each particle in the search space is considered an agent, and there is a gravitational force that these agents exert on each other. The gravitational force between two agents is directly proportional to their masses, while the agents are inversely proportional to the distance between them. It is assumed that objects interact with each other thanks to this force and that they attract each other as a result of this interaction. In the typical GSA algorithm, particles are randomly positioned in the solution space. During this positioning process, particles may be densely or sparsely distributed in some areas. This can have a negative effect on the solution of the problem. To overcome this problem, this study uses the quantization processing that CMAC initially applies to the input values. Thus, each particle/agent is equally partitioned in the solution space. This proposed initial criterion is one of the motivations of the study. Another motivation is the easy learning and generalization structure of CMAC and the data transfer between agents that are close to each other and have common weight values. CMAC has a different working principle from the classical neural network model, as it works according to the lookup table model [33]. In the lookup table model, vectors with similar input values produce similar output values. In addition, since CMAC has a generalization feature in its working principle, it works very fast. It also has a single-layer structure, and with this feature it is similar to a perceptron artificial neural network (ANN). As a result, CMACGSA is expected to find the best solution faster and more effectively. It is possible to come across optimization algorithms used in the training of artificial neural networks, but the contribution of this study is that the optimization algorithm has been improved using a neural network for the first time. The key contributions of this study can be summarized as follows.

i. We propose CMACGSA, a novel improvement of the GSA by integrating the CMAC, a neural network model. To the best of our knowledge, this is the first time that CMAC has been used to dynamically manage agent masses within a population-based optimization algorithm.
ii. By incorporating CMAC's quantization process, CMACGSA addresses the uneven initial distribution of agents in GSA and ensures a more uniform exploration of the search space.
iii. The CMACGSA leverages CMAC's efficient learning and generalization capabilities to enable faster convergence towards optimal solutions by adapting agent behavior during the search process.

iv. The performance of CMACGSA is rigorously evaluated on a comprehensive suite of benchmark functions, including the CEC 2014 test suite, and compared against standard GSA, other well-established metaheuristics, and state-of-the-art algorithms. In addition, the proposed CMACGSA is successfully applied to complex engineering design problems, showcasing its practical value in solving real-world optimization challenges.
v. This study provides a pioneering improvement of a metaheuristic optimization algorithm GSA using CMAC neural network, and brings a new perspective and methodology to future research in this area.

The following sections of this paper are organized as follows. Section II reviews the existing studies on metaheuristic algorithms, providing a comprehensive background on their development, applications, and the reason behind hybrid approaches. Section III provides a detailed overview of the working principle of GSA and the CMAC neural network. We also explain the design of our proposed CMACCGSA in this section, explaining in detail how the overall optimization process is developed. Section IV presents the experimental setup used to evaluate the performance of the CMACGSA algorithm, the benchmark test functions used, and the results obtained when comparing CMACGSA with traditional GSA and other SOTA algorithms. In addition, this section also includes results from engineering design problems. The final section summarizes the main contributions and findings of our work.

II. RELATED WORK

Many methods have been presented by researchers to improve GSA's performance and various methods have been proposed by researchers to improve the performance of the GSA. Some of these applications include data clustering [34], rock tensile strength estimation [35], classification [36], electrical power generation systems [37], data mining [38], civil engineering [39], and water management [40].

The hybrid and improved GSAs presented in the previous studies have been addressed in many engineering problems and applications. Mirjalili and Hashim [41] combined the exploitation feature of PSO with the exploration feature of GSA and presented the PSOGSA algorithm. In the study, they compared the performance of the proposed algorithm with typical PSO and GSA algorithms on some benchmark functions. In another study, Garg [42] proposed GSA-GA, a hybrid of GA and GSA for mixed-variable constrained nonlinear optimization problems. In the study, after adjusting the solution of the algorithm with GSA, each solution was updated with genetic operators such as selection, crossover and mutation. This increased the performance of GSA in finding optimal solutions. To verify the performance of the proposed algorithm, the study was compared with existing algorithms such as Cuckoo Search Algorithm (CSA) [43], PSO, ABC [44], [45], PSOGSA using data from nine different benchmark engineering design problems. The results showed that the performance of the proposed GSA-GA algorithm was better than other algorithms. Ozyon et al. [46] proposed incremental GSA (IGSA), which is based on the combination of GSA and the Incremental Social Learning (ISL) algorithm. In their study, they added new agents to GSA in certain steps starting from the first population and completed the process of adding agents to the maximum number of populations. They studied the process of adding agents to the population with three different approaches and obtained three different IGSAs. They compared the proposed algorithms with a typical GSA using benchmark functions and observed that the IGSAs gave better results than the GSA. In another study, Shehadeh [47] proposed a hybrid of Sperm Swarm Optimization (SSO) and GSA, called HSSOGSA. The study combined the power of both algorithms by combining the exploitation capability of SSO with the exploration capability of GSA. The performance of the proposed HSSOGSA was compared with GSA and SSO using CEC 2017 and benchmark tests, and it was observed that the performance of the hybrid method was better than typical SSO and GSA in most functions.

In order to improve the efficiency of typical GSA in optimization problems, GSA optimization approach has been developed by combining with many metaheuristic optimization algorithms. Mirjalili and Lewis [48] proposed gbest-guided GSA (GGSA) by archiving the best mass (gbest) to avoid local minima during the iteration of GSA and to speed up the exploitation phase. The effectiveness of GGSA was tested on CEC 2005 and two different engineering problems and the performance results were analyzed. In another study, Naik and Panda [49] combined CSA and GSA, called CS-GSA, for function minimization. In the proposed algorithm, they used the exploration capability of GSA in CSA, which is popular for exploitation behaviour. They used benchmark functions to investigate the performance of CS-GSA compared to classical CSA and GSA. As a result, they showed that the proposed CS-GSA gives better results in function minimization and convergence speed compared to CSA and GSA. Zhao et al. [50] proposed a hybrid algorithm based on self-adaptive GSA and DE to increase the convergence speed of GSA and maintain the balance between exploration and exploitation. In the proposed algorithm, crossover and mutation operators of DE [51] and a new perturbation based on Lévy flight theory are used to increase the exploitation capability. The performance of the proposed algorithm was evaluated with CEC 2017 benchmark functions and compared with the latest variants of GSA and it was concluded that the proposed algorithm gave better results.

Hybrid approaches using GSA have also been proposed in previous studies for feature selection. Taradeh et al. [52] proposed a hybrid algorithm, called HGSA, by applying the swarm intelligence algorithm and the features of GA to improve the detection ability of GSA in feature selection in big data. To investigate the performance of their proposed algorithm, they compared it with k-nearest neighbors and decision tree classifiers on eighteen datasets. They also

compared the results of GSA with metaheuristic algorithms such as GA, PSO and GWO. In another study, Li et al. [53] proposed a chaos embedded GSA-SVM hybrid algorithm by combining GSA with chaotic search SVM to improve parameter optimization and feature selection, which significantly affects the classification accuracy of SVM. To optimize the feature subsets, the input feature subsets and SVM parameters were optimized simultaneously, while chaotic search was embedded in the search iterations of GSA and GSA is used to optimize the parameters of SVM. The performance of the proposed algorithm was compared with fourteen UCI datasets using some hybrid algorithms such as PSO-SVM, GSA-SVM, GA-SVM. As a result, classification accuracy of the proposed algorithm was higher than other hybrid algorithms. Guha et al. [54] proposed clustering based population in binary GSA (CPBGSA) for feature selection. The purpose of this algorithm was to deal with the early convergence problem that occurs due to lack of exploration. The initial population was divided into different groups depending on the similarity of its members, and a representative was selected from each group. To compare the performance of the proposed CPBGSA, twenty datasets were used and it was shown that CPBGSA had a better classification accuracy compared to other methods.

In addition, GSA is also used for parameter estimation in various systems. Li et al. [55] proposed a piecewise function-based GSA (PFGSA) by designing a new gravity constant to verify the performance and efficiency of GSA. The work was experimentally tested for parameter identification of the automatic voltage regulator (AVR) system. They conducted an experimental study to investigate the performance of PFGSA in comparison with GSA, PSO and GA. They proved that PFGSA has better accuracy and stability rates in parameter estimation of AVR system compared to other algorithms.

There are very few studies on the hybridization or combination of GSA and ANN in previous studies. The models developed in the existing ones have been adapted to some engineering problems. The hybridization of optimization algorithms is widely used to achieve better results in the training of ANN. Most of these studies [56], [57], [58], [59] aim to improve the parameters or weaknesses of the structural features of ANN. In addition, it can be seen that there are applications in different fields, such as nuclear power plants and soil structures, where ANN and GSA are combined [60], [61]. While these strengths of CMAC allow it to be hybridized with other algorithms, there are very few studies hybridized with CMAC in previous. In one of these studies, Kong et al. [62] constructed a two-input prediction model for desulfurization and economic cost rate outputs using CMAC and GA algorithms. In another work, Lee et al. [63] proposed the CMAC algorithm with GA-based robust learning credit assignment. In the proposed algorithm, it is aimed to increase the accuracy by optimizing the learning times obtained with credit-assigned CMAC with GA. The performance of the developed algorithm was examined with the help of some mathematical functions and the proposed algorithm was compared with credit assignment CMAC, fuzzy CMAC and standard CMAC. As a result, it has been observed that the proposed algorithm gives better results compared to the other CMAC algorithm. In their study, Tu and Cao [64] proposed the PSO-CMAC algorithm based on the hybridization of PSO and CMAC for a permanent magnet synchronous motor. In the proposed algorithm, PSO simulated random learning among individuals in the population, while CMAC simulated self-learning of an individual. They conducted some experiments to examine the performance of the new algorithm. As a result, they compared their proposed algorithm with CMAC and PSO and showed that the proposed algorithm gave good results compared to other methods.

In the existing situation, the use of the GSA as a hybrid with other algorithms has been widely investigated and many studies have been conducted on combining GSA with various approaches, especially ANN. However, studies on the use of the CMAC structure with other algorithms are very limited. To the best of our knowledge, this study is the first study in which CMAC and GSA are used together and, in this context, it makes an important contribution to the literature. Moreover, the learning model introduced by the CMAC structure gives a metaheuristic algorithm the ability to learn for the first time. This innovative approach reinforces the originality of the work and provides a significant improvement by increasing the adaptability and performance of GSA.

III. THE PROPOSED CMACGSA ALGORITHM

A. GRAVITATIONAL SEARCH ALGORITHM (GSA)

The GSA is inspired by the law of gravity and the interaction of masses [65]. In the GSA algorithm, each object with mass applies a gravitational force on every other object in space. Elements of such algorithm are symbolized in Fig. 1 where F_i and F_j are the forces that M_i and M_j objects apply against each other and R_{ij} represents the distance between the masses [19].

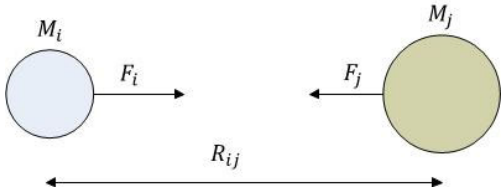

FIGURE 1. Gravitational force between two objects.

In GSA, each mass in the search space is called an agent, which has a position and experiences a gravitational force. Thanks to this force between the agents, an agent with the smaller mass tends to accelerate towards the agent with the larger mass. Initially, the positions of the cluster of N agents are randomly distributed in the search space S. Each of these agents is a member of the solution set given in Eq. (1).

$$X_i = \left(x_i^1, \ldots\ldots, x_i^d, \ldots..x_i^n\right), i = 1, 2, 3, \ldots, N \quad (1)$$

where x_i^d represents the position of i_{th} agent in the d_{th} dimension, n denotes the problem dimension, and N stands for the number of agents. The force acting on mass$i$ from mass j is given in Eq. (2) and Eq. (3).

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t)\,M_{aj}(t)}{R_{ij}(t)+\varepsilon}\left[x_j^d(t) - x_i^d(t)\right] \quad (2)$$

$$G(t) = G_0 e^{-\alpha\left(\frac{t-1}{t_m-1}\right)} \quad (3)$$

where t stands for the current iteration, $M_{pi}(t)$ and $M_{aj}(t)$ represent passive and active gravitational masses associated with agents i and j, $R_{ij}(t)$ gives the Euclidian distance between these masses,$G(t)$ expresses the gravitational constant, while $x_j^d(t) - x_i^d(t)$ presents the direction of the assumed force between the masses. G_0 represents initial gravitational value, α is a control factor that influences the rate of decrease of the gravitational constant over iteration, and t_m is the maximum number of iterations. The Euclidian distance $R_{ij}(t)$ can be calculated as given in Eq. (4):

$$R_{ij}(t) = \sqrt{\sum_{k=1}^{n}\left(X_i^k(t) - X_j^k(t)\right)^2} \quad (4)$$

Let the total force acting on agent $i$in dimension d be expressed as a randomly weighted sum of the d_{th} components of the forces exerted by other agents, as seen in Eq. (5).

$$F_i^d(t) = \sum_{j=1,i\neq j}^{N} r_j F_{ij}^d(t) \quad (5)$$

Here, r_j stands for a random number generated between *0* and *1*. Agents affected by this force move to a new position at each iteration and eventually reach a stable position representing the solution. During displacement, the agent with less mass moves faster, while the agent with more mass moves slower. An agent with a smaller mass will move towards an agent with a larger mass if they are in motion. Thanks to this force between the agents, an agent with the smaller mass tends to accelerate towards the agent with the larger mass. This acceleration behavior of the agent i with dimension d at iteration t is calculated in Eq. (6).

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \quad (6)$$

where $M_{ii}(t)$ denotes the inertial mass of agent i. Assuming that gravitational and inertial mass are equivalent in the original GSA, the masses are determined by a fitness matching process. Gravitational and inertial masses are calculated according to the Eq.'s (7), (8) and (9).

$$M_i(t) = M_{ii}(t) = M_{pi}(t) = M_{ai}(t) \quad (7)$$

$$m_i(t) = \frac{f_i(t) - f_w(t)}{f_b(t) - f_w(t)} \quad (8)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (9)$$

Here, $f_i(t)$ is the fitness value of the i_{th} agent at iteration t, $f_w(t)$ is the worst agent and $f_b(t)$ is the best agent at iteration t. If the calculation of the acceleration of agent $i$with dimension $d$at time t given in Eq. (6) is rewritten, the Eq. (10) is obtained.

$$a_i^d(t) = \sum_{j=1,i\neq j}^{N} r_j G(t)\frac{M_j(t)}{R_{ij}(t)+\varepsilon}\left[x_j^d(t) - x_i^d(t)\right] \quad (10)$$

The next velocity of i_{th} agent $v_i^d(t+1)$ in Eq. (11) is calculated by using its current velocity $v_i^d(t)$ and its current acceleration $a_i^d(t)$ at iteration t as:

$$v_i^d(t+1) = r_i \times v_i^d(t) + a_i^d(t) \quad (11)$$

where r_i denotes a randomly selected value in the range $[1, 0]$. From here, the next position of i_{th} agent can be calculated using its current position $x_i^d(t)$ and next velocity $v_i^d(t+1)$ as given in Eq. (12).

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (12)$$

The flow diagram of original GSA is presented in Fig. 2. In GSA, a random initial population is first generated. Then the fitness of each agent is calculated and the best agent, the worst agent, Euclidian distance, gravitational mass, force and acceleration of each agent are updated. In the next step, the velocity and then the position of each agent is updated. These processes continue until the expected result is obtained, i.e. until the stopping criterion is reached. In the last step the best solution is returned.

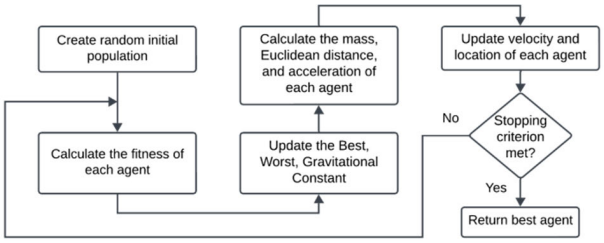

FIGURE 2. The flow diagram of a typical GSA.

B. CEREBELLAR MODEL ARTICULATION CONTROLLER (CMAC)

The CMAC is a feed forward neural network and an alternative to backpropagated multilayer networks [66], [67]. CMAC performs several consecutive mappings from input space to output space to compute its output. CMAC performs linear interpolation using the binary basis function for generalization scheme. As a result, the output is constant within each quantized state [68]. In CMAC architecture, Fig. 3 shows these mappings from the multidimensional input vector x_i to multidimensional output vector y_i.

CMAC assumes that input x_i is bounded and its range is known. The CMAC algorithm then quantizes the input x_i into one of n possible values, q_1 to q_n, which span the entire input space. This step constitutes the first mapping. Each of the quantization level q_k is defined as given in Eq. (13).

$$q_k = \frac{x_i - x_{min}}{\Delta} \quad (13)$$

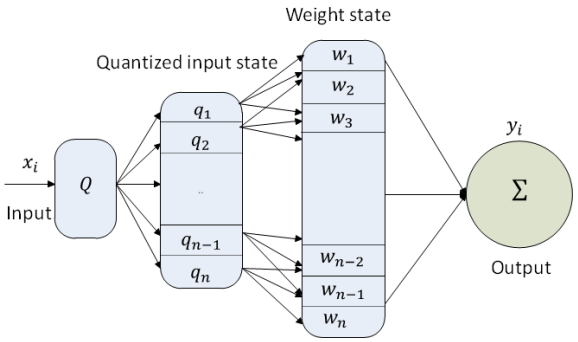

FIGURE 3. CMAC mapping from input vector x_i to output vector y_i.

where $x_i \in [x_{min}, x_{max}]$ and $\lfloor p \rfloor$ is the largest integer less than the real number p. The index k represents the quantization levels, x_{min} and x_{max} are the minimum and maximum values of the input, respectively. The quantization levels q_k range from 0 to $(q_{max} - 1)$. Here, Δ represents the quantization resolution. For fixed quantization resolution, Eq. (14) is used.

$$\Delta = \frac{x_{max} - x_{min}}{q_{max}} \tag{14}$$

The second mapping takes place from the quantization levels q_k, to weights w_n, in the weight space. Each quantization level maps to A^* weights in the weight space. A^* represents the number of weights used at any time by a particular quantization level. Any two adjacent quantization levels use weights that overlap by $A^* - 1$. If there are q_{max} quantization levels, then the weight space has N_C weights as given in Eq. (15).

$$N_C = q_{max} + A^* - 1 \tag{15}$$

The output of the CMAC is then obtained by summing the activated weights in the weight space as shown in Eq. (16).

$$y_i = \sum\nolimits_i Q_i w \tag{16}$$

where Q represents the mapping from quantization levels to weights. There are several ways to train a CMAC network, such as using the batch algorithm or using non-batch algorithm. For the batch algorithm, the weight update rule is given as Eq.'s (17), (18), and (19).

$$w_j(k+1) = w_j(k) + \Delta w_j(k+1) \tag{17}$$

$$\Delta w_j(k+1) = \eta \Delta e(k) \tag{18}$$

$$\Delta e(k) = \frac{1}{A^*} e(k) \tag{19}$$

where η is the learning rate and e(k) is the error function between the actual output and network output.

C. CMACGSA

The original GSA has a number of limitations and drawbacks. First, for complex and multimodal functions, it tends to get stuck at local minima, which can make it difficult to reach the global optimum. In addition, as the number of iterations increases, the diversity between the solutions may decrease and the agents may get too close to each other and lose the ability to explore. GSA is also known for its slow convergence speed for some optimization problems. In cases where the masses become more attracted to each other during optimization, the solution process slows down. Furthermore, the performance of the algorithm is very sensitive to its parameters. The choice of the right parameters can vary from problem to problem, and finding the right settings can be time-consuming. A loss of efficiency can also be observed for large problems, because as the size of the problem increases, the computational load increases and the performance of the algorithm is adversely affected.

In this study, a new method CMACGSA is proposed to overcome all these handicaps and improve the performance of the GSA. Basically, the learning ability of the CMAC network model is used to calculate the gravitational and inertial masses within the GSA structure. In contrast to classical GSA, the masses of the agents are calculated with weights (W) that undergo a learning process. That is, the masses of the agents are dynamically updated thanks to a learning mechanism integrated in the CMAC structure. In structure of the CMAC model in Fig. 4, each input signal corresponds to three weight values. Here, X denotes the input of the model and in this study all inputs are considered to be 1.

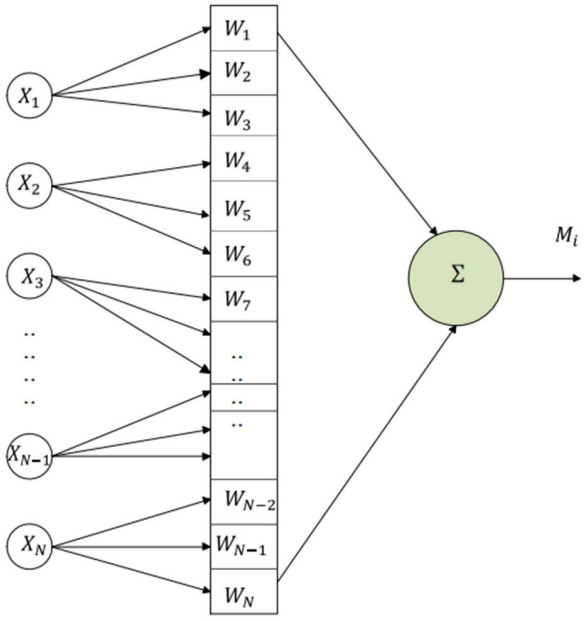

FIGURE 4. CMAC model architecture used in this study.

A^* is the number of weights selected by the user. A^* was chosen as three in order not to increase the training load of the model too much. N represents the number of search agents. For example, while the weight values for agent X_1 are expressed as W_1, W_2, W_3 the weight values for agent X_2 are expressed as W_4, W_5, W_6. The masses of the agents are then obtained by summing the corresponding weights. The mass

of an agent M_i is calculated using Eq.'s (20) and (21):

$$m_i(t) = \sum_{k=i}^{i+A^*} W_k(t) \tag{20}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{21}$$

After the mass calculation, the acceleration of agent i ($a_i^d(t)$) is given by Eq. (22). In contrast to the original GSA, the random value (r_j) is not used here, which is due to the learning process in the CMAC structure.

$$a_i^d(t) = \sum_{j=1, i\neq j}^{N} G(t) \frac{M_j(t)}{R_{ij}(t) + \varepsilon} \left[x_j^d(t) - x_i^d(t) \right] \tag{22}$$

where $G(t)$ is the gravitational constant over iteration t, R_{ij} is the distance between the masses, and ε is a very small value to eliminate ambiguity. After that, the updated velocity $v_i^d(t+1)$ and position $x_i^d(t+1)$ of an agent are calculated as in the original GSA. At the end of each iteration, the weight values used for mass calculation are updated after a training process as Eq. (23):

$$W_i(t+1) = W_i(t) + \eta \frac{f_b(t) - f_i(t)}{f_b(t) - f_w(t)} \cdot \frac{1}{A^*} \tag{23}$$

where η is the learning rate, f_w is the worst agent and f_b is the best agent. This training procedure optimizes the performance of the agents in an iterative manner by updating the weights of each agent based on their fitness values. The weight updates are organized according to the differences between the learning rate and the fitness values of the agents. For the best agent, the weights are not updated, while for the worst agent the weights are updated with the largest error rate.

In optimization algorithms, each particle or agent in the search space is a component of the solution set. The particles/agents are initially randomly positioned in the search space. One of the basic concepts in our proposed algorithm is the quantization of the input values of the CMAC neural network. This process involves dividing continuous input values into predefined intervals and allows the network to match input patterns more efficiently. In the proposed CMACGSA structure, we proposed to embed the quantization and weight matching structure of CMAC into the standard GSA for the initial positions of the agents. Fig. 5 shows the random positioning of the agents in the solution space in the original GSA. Fig. 6 shows the initialization of agents to quantization levels in CMACGSA.

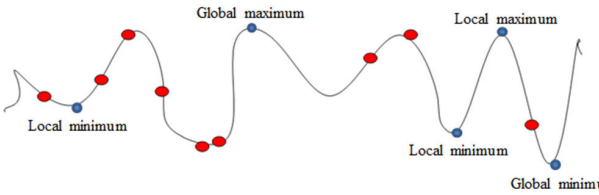

FIGURE 5. Agents with random initialized in GSA.

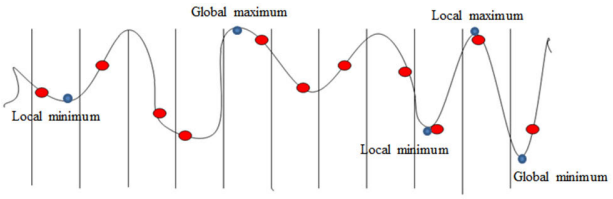

FIGURE 6. Agents assigned by quantization levels in CMACGSA.

In the proposed CMACGSA, a connection is established between quantization levels and weights. This connection is expected to make it easier for agents to reach their optimal values. In the quantization process, factors such as population number, problem size, lower and upper limits are taken into account. Thus, in the proposed algorithm, in d dimensions, within the lower and upper limits, the number of agents as many as the population number is initialized by quantizing.

During the development of the algorithms in CMACGSA, many problems are encountered, such as early convergence and local minima getting stuck. Therefore, various improvements are applied to the developed algorithms by taking advantage of some mechanisms or the strengths of other existing algorithms. In developing CMACGSA, we have added three different mechanisms, the first of which is the boundary control method. As the algorithm is updated, the agents whose new positions are determined may exceed the search boundaries and the situation of going out of the boundary is reached. A control mechanism was added to CMACGSA to overcome this problem. This mechanism is a function in the L-SHADE algorithm [69] that checks whether the agents exceed the bounds. If the new position of an agent violates the given lower and upper bounds, the agent position is updated to stay within the bounds. This is done by averaging between the agent position in the previous iteration and the boundary value, as seen in Eq.'s. (24) and (25).

$$v_{i,j} = \frac{x_{i,j} + x_{l,j}}{2}, \; if \; v_{i,j} < x_{l,j} \tag{24}$$

$$v_{i,j} = \frac{x_{i,j} + x_{u,j}}{2}, \; if \; v_{i,j} > x_{u,j} \tag{25}$$

where, $x_{l,j}$ and $x_{u,j}$ are the lower and upper boundary values of *jth*dimension in the search space, $x_{i,j}$ denotes the ith agent position in the previous iteration in the jth dimension, $v_{i,j}$ stands for ith agent current position in the *jth*dimension.

Another problem encountered is that the algorithm gets stuck in a local optimum during each update process. To overcome this problem, a mutation operator based on the Lévy flight mechanism has been added to the proposed algorithm. This mutation operator is defined by Eq.'s. (26) and (27).

$$\sigma = \left(\frac{\Gamma(1+\beta) sin\left(\frac{\pi\beta}{2}\right)}{\Gamma(\frac{1+\beta}{2}) \beta 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}} \tag{26}$$

$$x_{i,j} = x_{i,j} \left(1 + \frac{u_i}{|v_i|^{1/\beta}} \right) \tag{27}$$

where $x_{i,j}$ is the agent's position, u_i and v_i represent the random vectors, β denotes the stability index of the Lévy distribution and the parameter $\beta = 1.5$ is set. The vector u is randomly generated with a normal distribution and scaled by σ. This vector represents the direction and magnitude of the mutation. Another normally distributed random vector v is generated to set the step size based on the Lévy distribution.

The final innovation to the proposed algorithm CMACGSA is the addition of a counter mechanism. This additional mechanism uses a counter system that keeps track of each agent's failed improvement attempts. If an agent's position fails to improve for a certain number of iterations, that agent is randomly assigned to a new position in order to explore the search space more efficiently. Initially, the failure counters for all agents are set to zero. If the fitness value of the i*th* agent $f_i(t)$ is not better than the best fitness value $f_b(t)$, the failure counter is incremented by one. When the counter value reaches the threshold level (here $m = 100$), the position of the agent is randomly updated as in Eq. (28) and the search is continued with this new agent instead of the failed agent.

$$x_{i,j} = x_{l,j} + r_i(x_{u,j} - x_{l,j}) \tag{28}$$

In this updated equation, r_i represents a random number generated in the range from *0* to *1*. The pseudo-code of the proposed CMACGSA is shown in Algorithm 1. Here, the blue-highlighted lines of code indicate the innovative aspects of the proposed CMACGSA structure. Fig. 7 shows the flowchart of CMACGSA. In this flowchart, the blocks with blue outline and grey background indicate the new methods used in the proposed algorithm. From both the pseudo-code and the flowchart, the proposed algorithm incorporates several key modifications, starting with an initialization process where agent positions are determined through a quantization process, allowing for more structured placement of agents within the search space. This initialization improves the algorithm's starting point by evenly spreading agents across the problem domain.

The integration of the CMAC introduces active weights to guide learning, with masses now calculated using these weights to link agent behavior with learning dynamics, fostering more adaptive movement. A Lévy mutation function is included to prevent agents from getting stuck in local minima, enhancing exploration throughout the search process. To maintain agents within valid boundaries, a new boundary constraint mechanism averages out-of-bound positions with defined limits, keeping the search stable. A failure-handling mechanism resets agents' positions when they reach a threshold of unsuccessful attempts, encouraging exploration in new regions and improving robustness.

Additionally, CMAC weight updates are governed by fitness differences between agents, ensuring that better-performing agents influence the learning model more significantly, accelerating convergence toward optimal solutions. These modifications collectively boost the algorithm's efficiency, balancing exploration and exploitation to perform effectively on complex optimization tasks.

Algorithm 1 Pseudo-code of the Proposed CMACGSA

Input: Number of agents N, Maximum iterations T, Gravitational constant G_0, Decay factor α, Lower bound lb, Upper bound ub, Dimension d, Objective function $f(x)$, Number of unsuccessful attempts m
Output: Best solution X_{best}
Initialize:
$A^* = 3$ ▷ Number of active weights for CMAC
Randomly initialize wcights w for CMAC:
$w = \text{rand}(N \times \Lambda, d)$
Initialize by the quantization process the locations of the agents X_i in the scarch space:
Evaluate fitness of each agent: $f(X_i)$
Determine the best and worst agents: X_{best} , X_{worst} , f best $(t), f_{\text{worst}}(t)$
Initialize the velocities of the agents $V_i = 0$
Reset the counter for unsuccessful attempts: counter-zeros(N, 1)
for $t = 1$ to T **do**
  Update the gravitational constant: $G(l)$
  Calculate mass m_i of each agent using CMAC weights:
  for $i = 1$ to N **do**.
    $m_i(\ell) = \sum_{k-i}^{i+\Lambda^+} W_k(\ell)$
  end for
  Normalize the masses: $M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{m_j(t)}}$
  for each agent i **do**
    Calculate the acceleration $a_i^d(l)$ of agent i:

$$a_i^d(t) = \sum_{j-1, i \neq j}^{N} \frac{G(l) \cdot M_j(t) \cdot \left(X_j^d(t) - X_i^d(t)\right)}{\left\|X_j(l) - X_i(t)\right\| + c}$$

  end for
  for each agent i **do**
    Update the velocity of agent $i : V_i^d(t+1) = r_i \cdot V_i^d(t) + a_i^d(t)$
    Update the position of agent $i : X_i^d(\iota + 1) = X_i^d(t) + V_i^d(\iota + 1)$
    Lévy mutation function to avoid the local minimal
  end for
  if a dimension of the agent solution goes out of boundaries:
    New bound constraint mechanism: $X_i^d(\ell + 1) = \frac{X_1^d(t) + \text{Limit}}{2}$
  end if
  Evaluate fitness of each agent and selection better agent.
  if the new agent is better than the old agent
    Update agent and reset the counter
  else
    increase the counter by one
  end if
  if it reaches the number of failex attempts: counter $\stackrel{?}{=} m$
    $X_i = lb + (ub - lb) \cdot \text{rand}(1, d)$
  end if
  Update the best and worst agents: X_{best} , X_{worst} , $f_{\text{best}}(t)$, $\int_{\text{worst}}(t)$
  Update weights of the CMAC model:

$$W_i(l \mid 1) = W_i(t) \left| \eta \cdot \frac{\int_{\text{best}}(t) - f_i(t)}{f_{\text{best}}(t) - \int_{\text{worst}}(l)} \cdot \frac{1}{\Lambda^*} \right.$$

end for
Return the best solution X_{best} and the best fitness $\int_{\text{best}}$

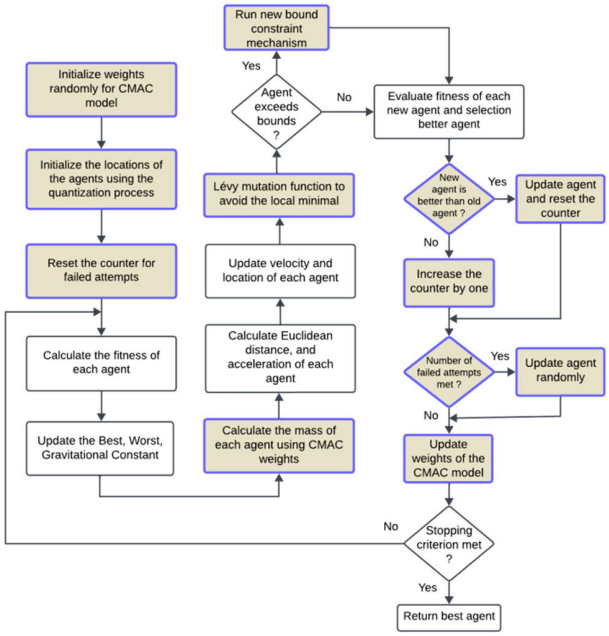

FIGURE 7. Flowchart of proposed CMACGSA method.

IV. RESULTS AND DISCUSSION

In the experimental analysis, firstly, the search behaviour of the proposed CMACGSA and the original GSA while searching for the global optimum in the search space was investigated using different analysis methods. These analysis methods include convergence, search history, elite trajectory, average distance, balance, diversity and computational complexity. To ensure a fair comparison between the original GSA and the proposed CMACGSA, the same parameters were used in all analyses. In the analyses, four test functions from the CEC 2014 benchmark suite were utilized. After the analyses, CMACGSA and the original GSA were run 51 times for different dimensions of these test problems and the statistical results were presented. Then, the 30D CEC 2014 benchmark results of the proposed algorithm were compared with other popular metaheuristic algorithms and GSA variants obtained from the previous studies. Experimental studies were conducted using MATLAB R2022A on a computer with an i7 2.3 GHz CPU, 8 GB RAM and a 64-bit Windows 11 operating system.

In experiments, the common parameters for both GSA and CMACGSA are set as follows: the population size is utilized as 10 $\times$ problem dimension, α is 20, the initial gravitational constant G_0 is 100, epsilon ε tolerance is $1e-6$, and the number of Function Evaluations (FEs) is used as problem dimension $\times$ 10000. The specific parameters associated with CMACGSA were utilized as follows: a learning rate (η) of 0.001, a learning epoch of 20, a number of weights (A^*) in quantization of 3, and a number of unsuccessful attempts (m) of 100.

A. ANALYSIS RESULTS OF CMACGSA

The analysis of the proposed CMACGSA structure considers various items, including convergence, search history, elite agent behavior, average distance of agents in the population, balance, diversity, and the algorithm's computational complexity. The CEC 2014 benchmark set [70], which is widely used in the literature, was the chosen for this. Table 1 provides a summary of the CEC 2014 benchmark functions. This benchmark set consists of four basic groups. Here, unimodal functions are shown in blue, simple multimodal functions in red, hybrid functions in green, and composition functions in orange [70]. In addition, the analyses conducted within the scope of the experimental studies are carried out for 2D.

TABLE 1. Summary of the CEC 2014 Test Suite (Blue: Unimodal, Red: Simple Multimodal, Green: Hybrid, and Orange: Composition).

No	Function name	$f(x^*)$	No	Function name	$f(x^*)$
F1	Rotated High Conditioned Elliptic	100	F16	Shifted & Rotated Expanded Schaffer	1600
F2	Rotated Bent Cigar	200	F17	(N=3) Hybrid Function 1	1700
F3	Rotated Discus	300	F18	(N=3) Hybrid Function 2	1800
F4	Shifted & Rotated Rosenbrock	400	F19	(N=4) Hybrid Function 3	1900
F5	Shifted & Rotated Ackley	500	F20	(N=4) Hybrid Function 4	2000
F6	Shifted & Rotated Weierstrass	600	F21	(N=5) Hybrid Function 5	2100
F7	Shifted & Rotated Griewank	700	F22	(N=5) Hybrid Function 6	2200
F8	Shifted Rastrigin	800	F23	(N=5) Composition Function 1	2300
F9	Shifted & Rotated Rastrigin	900	F24	(N=3) Composition Function 2	2400
F10	Shifted Schwefel	1000	F25	(N=3) Composition Function 3	2500
F11	Shifted & Rotated Schwefel	1100	F26	(N=5) Composition Function 4	2600
F12	Shifted & Rotated Katsuura	1200	F27	(N=5) Composition Function 5	2700
F13	Shifted & Rotated HappyCat	1300	F28	(N=5) Composition Function 6	2800
F14	Shifted & Rotated HGBat	1400	F29	(N=3) Composition Function 7	2900
F15	Expanded Griewank plus Rosenbrock	1500	F30	(N=3) Composition Function 8	3000

The concepts presented in this analysis serve to better understand the performance of the algorithm and its working mechanisms. Four benchmark problems of three different types are selected to be used in the proposed CMACGSA analyses. These are Rotated Bent Cigar Function (*F2*), Shifted and Rotated analyses (*F9*), Expanded Griewank's plus Rosenbrock's Function (*F15*), (N=5) Composition Function 5 (*F27*). A total of 8 different graphs are presented in the analyses of the four benchmark functions. These are the 3D and contour plots of the benchmark function, the convergence plot, the search history, the search history of the elite candidate, the position changes of the elite

candidate, the average distance change of the search agents, the balance between the exploration and exploitation phases and the diversity of candidate solutions.

Fig. 8 shows the analysis results of F2 benchmark for the proposed CMACGSA and original GSA. This function is a unimodal function, meaning it has a single global minimum. It is characterized by a highly stretched, elongated shape where most variables contribute little to the function value except for one or two key variables. The function applies a rotation to the traditional Bent Cigar function, adding complexity to its landscape.

Fig. 9 and Fig. 10 show the analysis results of F9 and F15 benchmarks for both algorithms. F9 benchmark is a multimodal function, known for its numerous local minima distributed throughout the search space. The traditional Rastrigin function has a cosine wave-like structure, but in this variant, both a shift and rotation are applied to increase complexity and make the global minimum harder to locate. The function tests an algorithm's ability to escape local optima and converge to the global minimum, as the local minima can trap poorly designed optimization techniques. The other benchmark F15 combines characteristics of both the Griewank and Rosenbrock functions. It exhibits the valley-like structure of the Rosenbrock function but with the periodic and separable nature of the Griewank function. The expansion introduces non-linearity and increases the complexity of the function. This hybrid structure challenges algorithms to balance exploration and exploitation, as the landscape has flat regions and sharp ridges that are difficult to navigate.

Fig. 11 illustrates the results of the 2D analyses performed on Composition Function 5. This benchmark is a complex function created by blending five simpler sub-functions. These sub-functions are weighted and combined to form a highly multimodal landscape with various global and local optima. The weighting and blending of the sub-functions create regions of varying difficulty across the search space, making it particularly challenging for algorithms to find the global optimum. This function tests an algorithm's adaptability and robustness across different types of landscapes within the same search space.

The convergence curves show that the proposed CMACGSA algorithm outperforms the original GSA for the four benchmarks. Especially, in benchmark F27, the original GSA gets stuck in the problem solution, whereas CMACGSA continues its convergence in the exploration phase and approaches the solution in the exploitation phase.

The search history, which shows the points in the solution space travelled by the search agents in order to find the global minima of the four benchmark functions, shows where in the solution space the searches of the candidates are concentrated and where they are neglected. In this analysis, the search agents in the original GSA are represented by squares and those in the proposed CMACGSA by circles. In order to reduce the complexity of the graphs and to obtain faster plots, the analysis results were obtained with a refresh rate of 100 (changes were reflected every 100 steps). Here, a blue square is used in GSA and a red circle in CMACGSA for the best solution point obtained at the end of the optimization. Due to the quantization mechanism in the proposed CMACGSA structure for the four benchmark problems, it is seen that the solution, which starts with equally spaced positioned search candidates to characterize the search space, finds different final solution points than the original GSA except F9. Along the solution, it is understood that the search agents of the proposed new CMACGSA especially visit the local minima but do not get stuck there.

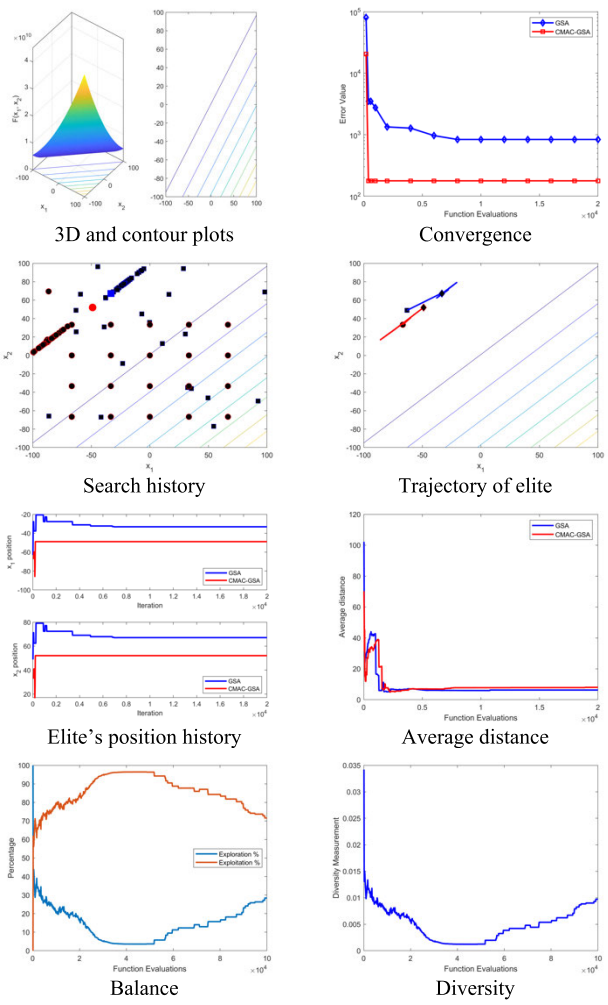

FIGURE 8. Analysis results for Rotated Bent Cigar function (F2).

When the trajectory of the elite candidate and the position history analysis of the elite candidate, which shows the behavior of the elite solution during optimization, are examined, it is seen that CMACGSA and the original GSA try to reach the solution with different paths (trajectories). Only in the F9 benchmark, both algorithms reached approximately the same solution point in the global minima. For the other benchmarks, it should not be difficult to say that the both algorithms reach different global solution points as a result of different elite trajectory behavior. By examining the changes in the average distances between search agents, it can

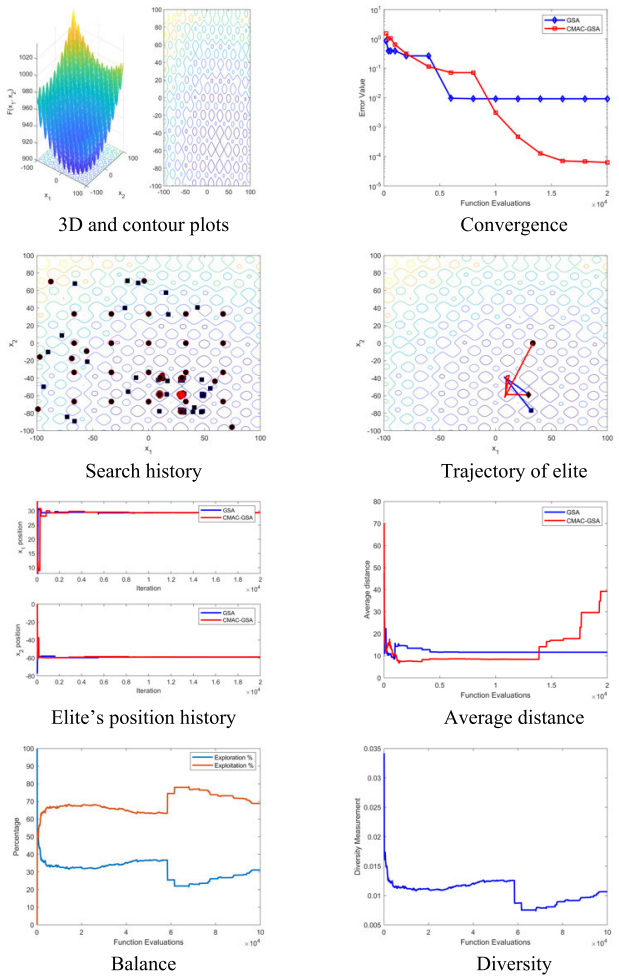

FIGURE 9. Analysis results for Shifted and Rotated Grievance (*F*9).

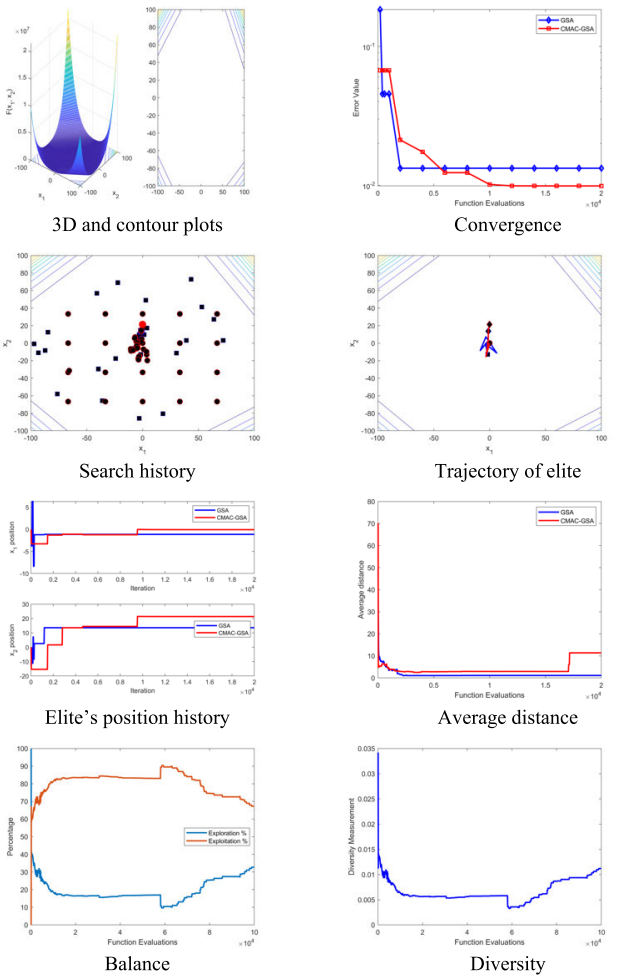

FIGURE 10. Analysis results for Shifted and Rotated Expanded Griewank's plus Rosenbrock's function (*F*15).

be observed that as the solution is approached, the overall distance decreases due to the convergence of candidates. However, in the proposed CMACGSA structure, the mutation mechanism causes the position of one or several agents to shift, leading to an increase in the average distance.

In the balance analysis, except for the F27 benchmark function, the other three benchmarks show that the proposed CMACGSA focuses on both exploration and exploitation throughout the optimization with a balance transition of 40%-60% between the exploration and exploitation phases. In F27, it is seen that extensive exploration is done at the beginning, then the focus is gradually given to exploitation. In the final stage, a more balanced approach is adopted between exploration and exploitation, indicating that the algorithm continues to explore new areas and tries to improve existing good solutions.

In the diversity analysis, a high initial diversity is generally observed in all graphs, indicating that the algorithm has explored the solution space extensively. It is understood that the diversity tends to decrease with different characteristics during the optimization process. In the F9 benchmark, the increase in diversity in the last stage is due to the mutation mechanism in the proposed CMACGSA structure.

Considering the time complexity of the proposed CMACGSA algorithm, the time complexity for each iteration consists of the following operations:

i. The calculation of the updated masses is performed for all search agents (N: number of search agents) and has a complexity of O(N).
ii. The calculation of the gravitational constant is performed in two nested cycles, with a complexity of $O(N^2)$.
iii. The complexity of the calculation of the acceleration in the gravitational field is again $O(N^2)$ due to the two cycles.
iv. The complexity of the fitness value calculation and mutation operations for each search agent is $O(N \times D)$, where D is the number of dimensions.
v. For the training phase of the CMAC structure, two loops are used, the outer loop is 20 steps and the inner loop is N steps, so the time complexity is $O(20 \times N)$.

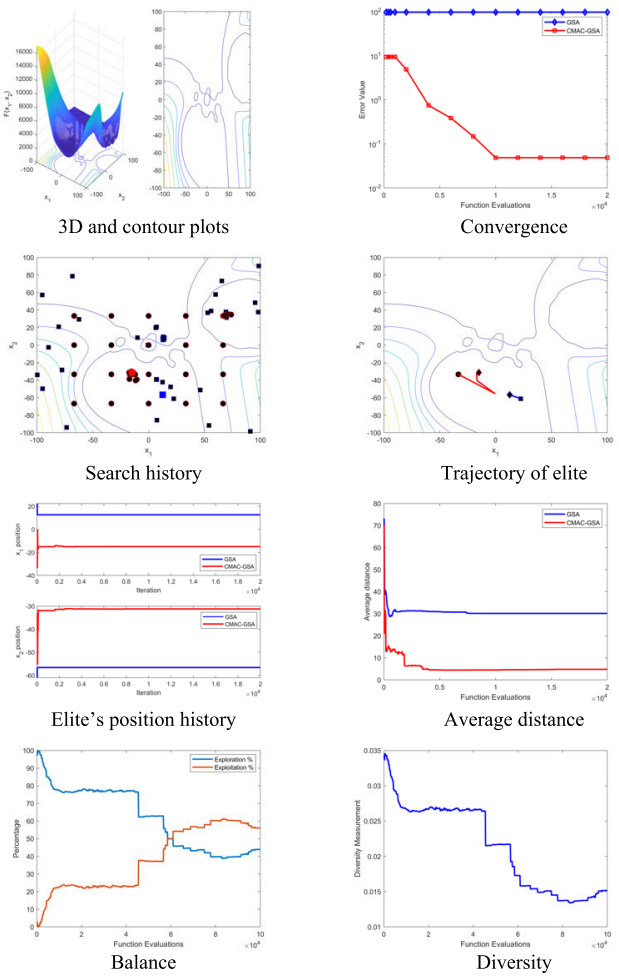

FIGURE 11. Analysis results for Composition function 5 (N=5) (*F*27).

The total time complexity is obtained by combining all these steps. The time complexity of the proposed algorithm is $O(2N^2 + N(D + 20 + 1))$. Following the simplification of the time complexity, the resulting complexity is $O(N^2)$. As a result, the time complexity of the original GSA algorithm and the proposed CMACGSA algorithm is the same. This shows that the CMACGSA algorithm is not more costly in terms of time than the original GSA, although it additionally includes a complex structure such as the CMAC structure.

B. CEC 2014 COMPARISON RESULTS

In order to better understand the performance comparison between CMACGSA and original GSA, CEC 2014 test functions were run fifty-one times in different dimensions (10, 30, and 50). In this comparison, the common parameters of both algorithms are chosen in the same way. The results for 10D benchmarks are given in Table 2, the results for 30D benchmarks are given in Table 3 and the results for 50D benchmarks are shown in Table 4. These tables present the best, worst, median, mean, and standard deviation statistical metrics for both algorithms.

In the 10D benchmarks, CMACGSA outperforms GSA in 20 out of 30 problems for mean metric results, demonstrating superior optimization capability in this dimension. In the 30D benchmarks, CMACGSA continues to perform better, achieving outperformance in 18 out of 30 problems based on both the mean values. At the 50D level, GSA shows improvement by achieving better results in 18 out of 30 problems for the best values. However, CMACGSA remains more consistent overall, with higher median and mean values across most tests.

The standard deviation is similar between the two algorithms in all dimensions, although it increases slightly for CMACGSA in higher dimensions, indicating that although it generally performs better, the variability of its results increases as the problem complexity increases. These results reflect CMACGSA's strong ability to solve both low and high dimensional problems, maintaining a competitive edge over GSA in terms of overall consistency and average performance.

To prove the performance superiority of CMACGSA, SOTA algorithms such as WOA [12], PSO [26], CCS [71], Coyote optimization Algorithm (COA) [72], DA [8], Sine Cosine Algorithm (SCA) [73] were selected for comparison. The fact that these algorithms are well known and their results are available in the literature with CEC 2014 30D is the main reason for choosing these algorithms.

Table 5 shows the used parameters and their values for other existing metaheuristic algorithms in comparison with CMACGSA. The comparison was performed with the same parameter values as in the original papers. The results of the comparison are given in Table 6. The mean values of each algorithm are shown in the table. In the bottom row of each function, a ranking was added according to the mean values. The second row from the bottom of the table shows the average of these rankings across 30 benchmarks. The bottom row shows the overall rank according to these average rankings. Looking at the comparison result, we can see that the proposed CMACGSA gives better results than the other metaheuristic algorithms.

Finally, the performance of the CMACGSA was compared to the CEC 2014 results obtained from four GSA variants in existing. The compared algorithms are Sine Chaotic Gravitational Search Algorithm (SCGSA) [74], Balance Adjustment Based Chaotic Gravitational Search Algorithm (BA-CGSA) [75], Gravitational Evaluation Algorithm (GEA) [76], Gbest-Guided Gravitational Search Algorithm (GG-GSA) [77]. Table 7 shows the used parameters and their values for four GSA variants such as SCGSA, BA-CGSA, GEA and GG-GSA in comparison with CMACGSA.

The CEC 2014 30D benchmark results for the proposed CMACGSA and GSA variants are presented in Table 8. Here, the mean metric values of these algorithms are summarized. The ranking of the algorithms according to the benchmark

TABLE 2. 10D CEC 2014 benchmark results of GSA and CMACGSA algorithms.

	GSA					CMACGSA				
No	**Best**	**Worst**	**Median**	**Mean**	**Std**	**Best**	**Worst**	**Median**	**Mean**	**Std**
1	42.622	39620	4383.1	7480.9	7462.5	1176.6	3.75e+5	27569	45917	70803
2	1331.3	7662.2	3255	3413.4	1573.9	4535.1	44912	14274	15123	6808.4
3	707.2	8714	4039.3	4297.4	1936.6	4894.9	17357	10233	9979.2	3022.6
4	0.0241	34.781	34.781	33.418	6.812	34.783	34.793	34.788	34.788	0.0022
5	20.189	20.515	20.345	20.354	0.0682	20.137	20.527	20.36	20.348	0.0751
6	3.2621	7.8805	5.6291	5.5606	0.9799	0.2247	4.9173	1.8229	2.0526	1.2785
7	0.0425	0.5025	0.0961	0.1534	0.1299	0.1101	0.4805	0.311	0.2997	0.0829
8	22.889	77.98	33.834	39.186	13.876	14.941	44.007	22.906	23.338	5.9024
9	18.912	59.88	29.852	30.468	7.3829	10.957	50.739	21.896	26.389	11.775
10	757.92	2237.8	1759.2	1663	348.97	673.07	1639.9	1265.4	1219.9	193.4
11	469.26	1941.7	1440.5	1322.9	383.25	664.7	1412	1160.2	1135.6	180.22
12	0.5727	1.556	1.1752	1.1721	0.1965	0.7823	1.5477	1.1541	1.1598	0.1750
13	0.0705	0.2836	0.1502	0.1552	0.0502	0.0888	0.2333	0.1421	0.1465	0.0359
14	0.2396	0.4698	0.4144	0.4035	0.0477	0.1194	0.4647	0.3809	0.3721	0.0557
15	0.7590	2.4871	1.6493	1.6365	0.354	1.1815	2.7349	1.9796	1.9694	0.3034
16	2.5933	3.5738	3.1317	3.1121	0.2584	1.8767	3.0469	2.5645	2.5458	0.2952
17	593	4127.2	1058.5	1349.9	716.64	578.7	4.00e+5	9419.2	8.73e+5	1.23e+5
18	2059.8	7910.2	5335.9	5336.9	1208.5	280.54	4470.4	2710.5	2556	973.95
19	1.662	4.747	3.2525	3.1975	0.6077	1.5975	4.7593	3.2144	2.9296	0.7395
20	311.57	4095.8	1759.2	1943.2	955.05	94.177	2381.4	698.82	874.25	575.82
21	65.303	2724.1	583.38	691.13	555.54	250.61	10255	1826.2	1968.2	1569.5
22	143.64	235.92	168.14	168.08	21.063	99.555	186.7	173.73	164.98	22.582
23	201.57	329.46	202.15	222.03	46.799	202.8	329.46	329.46	287.51	59.913
24	189.56	203.62	202.13	200.77	3.2101	110.93	203.98	145.41	149.11	27.996
25	191.06	200.2	199.64	199.24	1.4242	189.42	200.17	200.17	199.36	2.2838
26	100.1	178.7	104.3	106.13	10.862	100.06	103.01	100.72	101.1	0.9779
27	261.74	746.64	573.21	572.51	110.33	184.66	443.59	365.21	367.15	46.789
28	935.97	2233.8	1523.8	1582.4	317.83	112.28	439.92	116.21	231.96	147.77
29	5.66e+5	1.11e+8	3.46e+7	3.39e+7	2.50e+7	3.40e+2	1.89e+5	5.53e+2	7.90e+3	30768
30	2826.3	1.21e+5	15127	29473	30890	855.07	2886.2	2009.7	1971.9	407.67

results for each function is given at the end of the rows. The proposed CMACGSA ranks first and outperforms all other GSA variants. Specifically, CMACGSA achieves the 1st rank across the evaluated functions, indicating that it provides the most effective optimization results on the average metric. GG-GSA follows in 2nd place, suggesting strong performance but slightly less consistent results compared to CMACGSA. GEA is ranked 3rd, demonstrating moderate competitiveness, while BA-CGSA and SCGSA are ranked 4th and 5th respectively, reflecting lower overall effectiveness. These rankings highlight that the proposed CMACGSA not only introduces significant improvements over the original GSA, but also outperforms other improved variants from the literature. The superior performance of CMACGSA, particularly on 30D problems, demonstrates its ability to balance exploration and exploitation more effectively, making it the most reliable and robust algorithm among the variants compared.

C. COMPARISON WITH ENGINEERING DESIGN PROBLEMS

In this study, three widely used engineering design problems were considered to demonstrate the performance of the proposed CMACGSA in real-world problems. The common feature of these problems is to find the optimum result using the given parameters. Here, the ability of CMACGSA for real-world problems was first compared with the original GSA and then with other metaheuristics from the literature.

TABLE 3. 30D CEC 2014 benchmark results of GSA and CMACGSA algorithms.

	GSA					CMACGSA				
No	**Best**	**Worst**	**Median**	**Mean**	**Std**	**Best**	**Worst**	**Median**	**Mean**	**Std**
1	7.88e+4	1.38e+6	1.11e+6	1.02e+6	3.27e+5	5.44e+5	3.28e+6	1.02e+6	1.38e+6	9.85e+5
2	1.33e+4	3.11e+4	2.07e+4	2.11e+4	4.10e+3	3.34e+4	8.19e+4	6.37e+4	6.04e+4	1.37e+4
3	2.91e+4	6.89e+4	5.02e+4	4.98e+4	7.63e+3	2.34e+4	3.94e+4	3.14e+4	3.14e+4	5.77e+3
4	0	74.53	66.56	44.11	29.98	89.284	135.11	127.63	119.07	16.927
5	20.83	21.03	20.94	20.94	0.05	20.767	20.988	20.920	20.913	0.0621
6	30.04	44.14	40.04	38.88	3.78	15.899	31.108	27.694	25.034	5.3301
7	0.02	0.05	0.04	0.04	0.01	0.0766	0.1720	0.1417	0.1396	0.0274
8	121.4	335.49	306.49	266.25	79.55	119.84	257.91	248.81	236.12	39.786
9	140.3	330.82	159.21	165.55	33.91	114.49	135.57	130.43	130.34	6.0094
10	5979.1	8371.2	7679.2	7646.6	442.76	4805.7	5967.1	5306.6	5397.1	368.05
11	6727.9	8388.6	7961.5	7853.2	375.64	5773.8	6836.7	6605.0	6452.5	379.75
12	1.57	2.93	2.51	2.48	0.3	1.6838	2.5626	2.1982	2.2051	0.2639
13	0.37	0.56	0.47	0.47	0.04	0.2383	0.3537	0.3170	0.3129	0.0333
14	0.21	0.34	0.27	0.28	0.03	0.3182	0.4205	0.3597	0.3634	0.0306
15	1.62	9.88	2.83	3.07	1.17	11.391	13.360	12.327	12.302	0.6686
16	12.46	13.38	13.01	12.97	0.18	12.138	12.555	12.362	12.343	0.1347
17	1.28e+4	1.09e+5	5.06e+4	5.14e+4	2.61e+4	4.52e+4	2.61e+5	1.57e+5	1.46e+5	6.56e+4
18	225.98	516.18	303.62	333.87	79.8	686.80	1609.02	978.235	1058.556	303.83
19	10.28	16.07	12.48	12.43	1.12	11.678	16.448	14.039	14.047	1.3651
20	1.04e+4	6.57e+4	2.15e+4	2.41e+4	9.88e+3	7.56e+3	1.92e+4	1.16e+4	1.23e+4	3.92e+3
21	2.53e+4	1.63e+5	1.03e+5	1.01e+5	3.21e+4	1.30e+4	5.75e+4	2.36e+4	2.93e+4	1.46e+4
22	405.69	1245.89	805.24	848.63	171.51	397.61	868.95	631.18	620.36	148.57
23	203.23	204.49	204.04	204	0.24	315.25	315.25	315.25	315.25	0.0006
24	200.21	200.35	200.29	200.29	0.03	223.64	238.26	224.04	225.64	4.2914
25	200.05	200.07	200.06	200.06	0.01	206.97	213.08	211.49	210.81	1.8621
26	106.06	200.01	183.29	169.03	34.12	105.10	200.04	107.23	119.92	28.916
27	865.35	1870.8	1625.7	1536.9	260.13	509.17	566.32	549.47	543.02	18.755
28	1293.6	8559.8	7152.2	6658.2	1712.3	2688.5	3282.5	2798.3	2937.7	233.28
29	2.33e+8	5.73e+8	4.95e+8	4.89e+8	6.27e+7	6038.7	10284	8624.37	8720.35	1309.4
30	1.48e+6	8.68e+6	4.29e+6	4.21e+6	1.45e+6	1567.2	7184.3	3366.1	3389.7	1485.7

These algorithms are GA [21], PSO [26], DE [24], ABC [9], GWO [11], GSA [19], Biogeography-based Optimization (BBO) [78], SCA [73], Salp Swarm Algorithm (SSA) [79]. The number of runs for each algorithm is 10, and the number of populations is 30. The parameters of the problems were selected according to their own characteristics. Table 9 shows the used parameters and their values for other existing metaheuristic algorithms in comparison with CMACGSA in

TABLE 4. 50D CEC 2014 benchmark results of GSA and CMACGSA algorithms.

	GSA					CMACGSA				
No	Best	Worst	Median	Mean	Std	Best	Worst	Median	Mean	Std
1	5.61e+5	1.61e+6	9.19e+5	9.44e+5	2.13e+5	2.11e+6	2.77e+6	2.57e+6	2.47e+6	2.43e+5
2	2.76e+4	4.79e+4	4.01e+4	3.99e+4	4.92e+3	4.67e+5	7.46e+5	5.20e+5	5.62e+5	8.73e+4
3	8.05e+4	1.28e+5	1.15e+5	1.13e+5	9.30e+3	4.70e+4	1.09e+5	8.60e+4	8.47e+4	1.53e+4
4	18.33	143.3	89.11	92.74	28.03	98.574	108.92	102.57	103.52	2.9606
5	21	21.19	21.14	21.13	0.04	21.055	21.169	21.118	21.121	0.0322
6	68.37	76.27	73.72	73.51	1.78	26.581	34.161	31.012	30.567	2.5966
7	0.02	0.04	0.03	0.03	0.01	0.5878	0.8429	0.6274	0.6779	0.0967
8	246.76	650.91	611.19	552.53	134.29	460.52	537.41	523.06	513.96	21.520
9	337.3	702.06	366.16	478.8	154.38	266.72	427.70	336.45	334.83	46.215
10	1.31e+4	1.54e+4	1.46e+4	1.45e+4	4.84e+2	1.12e+4	1.21e+4	1.16e+4	1.16e+4	291.21
11	1.34e+4	1.52e+4	1.45e+4	1.45e+4	3.98e+2	1.24e+4	1.34e+4	1.29e+4	1.29e+4	328.19
12	2.46	4.01	3.46	3.37	0.37	2.6129	3.3039	3.0971	3.0767	0.1905
13	0.39	0.49	0.44	0.44	0.02	0.3035	0.4053	0.3244	0.3349	0.0324
14	0.28	0.39	0.33	0.33	0.02	0.2982	0.3552	0.3293	0.3256	0.0222
15	3.75	8.04	5.95	5.96	1.03	7.008	24.365	9.330	13.616	6.5679
16	22.11	22.82	22.65	22.63	0.14	21.276	21.958	21.7093	21.664	0.1945
17	4.15e+4	4.91e+5	1.45e+5	1.86e+5	1.18e+5	8.72e+4	3.87e+5	1.80e+5	2.16e+5	9.10e+4
18	1035.8	3941.3	2658.4	2643.5	745.15	1.65e+4	1.71e+5	4.31e+4	5.02e+4	4.20e+4
19	20.2	30.09	25.15	25.06	1.97	21.939	32.505	26.404	26.662	3.3169
20	1.67e+4	4.99e+4	3.05e+4	3.05e+4	8.09e+3	1.30e+4	2.34e+4	1.53e+4	1.62e+4	3.51e+3
21	1.23e+5	9.92e+5	3.39e+5	3.49e+5	1.59e+5	3.21e+5	1.57e+6	5.62e+5	7.00e+5	3.85e+5
22	1231.4	2482.3	1786.0	1786.5	284.82	1462.0	2004.5	1779.4	1757.8	179.58
23	203.9	205.36	204.86	204.82	0.28	344.44	345.24	344.88	344.79	0.2738
24	200.47	200.83	200.64	200.64	0.06	271.86	281.18	276.42	276.38	3.0000
25	200.08	200.11	200.1	200.1	0.01	228.35	238.08	234.58	234.54	3.0534
26	112.83	200.04	200.02	197.03	13.63	107.79	200.21	129.81	155.21	43.568
27	2517.6	3203.1	2869.3	2868.5	153.17	1422.8	1777.35	1690.4	1657.9	102.36
28	5761.1	16525.1	14575.4	13709.8	2514.6	7842.3	8656.8	8201.1	8256.9	283.49
29	8.17e+8	1.21e+9	1.03e+9	1.03e+9	8.65e+7	1.78e+4	2.53e+4	1.97e+4	2.07e+4	2689.2
30	1.53e+7	4.20e+7	2.96e+7	2.93e+7	5.51e+6	1.36e+4	2.18e+4	1.61e+4	1.67e+4	2514.5

TABLE 5. The Used Parameters and Their Values for Other Existing Metaheuristic Algorithms in Comparison with CMACGSA.

Algorithm	Parameters
CCS [71]	β=1.5, N=10, pa=0.025
COA [72]	Number of coyotes=10
DA [8]	w_{max}=0.9, w_{min}=0.4
PSO [26]	c_1=c_2=2
SCA [73]	a=2
WOA [12]	b=1

engineering design problems. The parameter values of the presented problems may be different and have constraints.

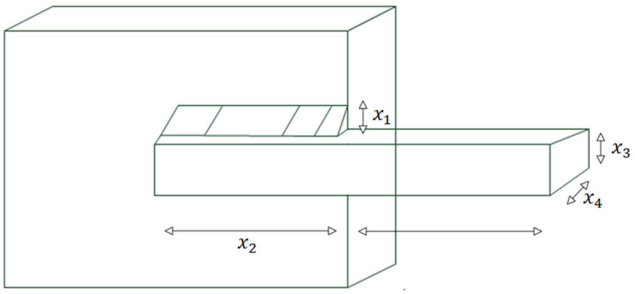

FIGURE 12. Welded beam design problem.

1) WELDED BEAM DESIGN

The Welded Beam Design (WBD) problem aims to minimize the production cost of the welded beam using the weld thickness x_1, length x_2, height x_3, and thickness x_4 variables [80], [81], [82]. The WBD problem is shown in Fig. 12. The Eq.'s (29)-(38) for the WBD problem are given below:

$$M = P(L_c + \frac{x_2}{2}) \tag{29}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{30}$$

$$J = 2\sqrt{2}\, x_1 x_2 R^2 \tag{31}$$

$$\Phi = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) \tag{32}$$

$$\sigma = \frac{6PL_c}{x_3^2 x_4} \tag{33}$$

$$\Delta = \frac{4PL_c^3}{E x_3^3 x_4} \tag{34}$$

$$P_c = 4.013E\sqrt{\frac{x_2^3 x_4^6}{36}}\frac{1 - x_3\frac{\sqrt{\frac{E}{4G}}}{2L_c}}{L_c^2} \tag{35}$$

$$\tau_p = \frac{P}{\sqrt{2}\, x_1 x_2} \tag{36}$$

$$\tau_{pp} = \frac{M \cdot R}{J} \tag{37}$$

$$\tau = \sqrt{\tau_p^2 + \frac{2\tau_p \tau_{pp} x_2}{2R} + \tau_{pp}^2} \tag{38}$$

where M denotes the bending moment, R is the distance to neutral axis, J is the polar moment of inertia, Φ is the cost function, σ is the bending stress, Δ is the tip deflection, P_c is the buckling load, τ_p, τ_{pp} are 1st and 2nd components of shear stress, and τ is the total shear stress. The parameters for the WBD problem are defined as follows: the applied tip load $P = 6000lb$, the Young's modulus of the beam $E = 30 \times 10^6 psi$, the shear modulus of the beam $G = 12 \times 10^6 psi$, and the length of the cantilever part of the beam $L_c = 14in$.

The constraints for the maximum shear and bending stress, the tip deflection, geometric, the buckling load, the minimum weld thickness and cost in the WBD problem ensure both structural integrity and cost-effectiveness. The results for the WBD problem, presented in Table 10, highlight the performance of CMACGSA compared to various metaheuristic algorithms. CMACGSA achieves competitive outcomes with a best solution cost of 1.9082, showing strong optimization performance close to that of PSO (1.6977) and GWO (1.6978), which produced the best results overall. The median and mean values for CMACGSA (2.0009 and 2.0146, respectively) demonstrate its consistency, with low variability reflected in the standard deviation (0.1001). This indicates that CMACGSA reliably produces near-optimal solutions across multiple runs, performing better than algorithms like GSA and BBO, which show higher variability and worse mean values. While CMACGSA does not outperform PSO and GWO in terms of best solution, it surpasses algorithms such as GA, DE, and SSA in terms of stability and average performance. The relatively low standard deviation further underscores the algorithm's robustness, making it a reliable choice for solving the WBD problem with a balance of cost-effectiveness and solution quality. Fig. 13 shows the convergence curves and boxplot graph of all metaheuristics used in this problem. The first graph depicts the best fitness values achieved over iterations for multiple algorithms, including the proposed CMACGSA and other metaheuristics. The second graph specifically compares the convergence behavior of GSA and CMACGSA, demonstrating that CMACGSA converges faster and more effectively than GSA, achieving lower fitness values earlier in the iteration process. The last graph shows the statistical results of the algorithms using the boxplot. It shows that algorithms like GWO and PSO reached lower fitness values, indicating better solution quality.

2) PRESSURE VESSEL DESIGN

The Pressure Vessel Design (PVD) operates with high pressure and is one of the engineering problems that transport, store or process liquids or gases from one place to another [80], [83]. The problem aims to minimize the total production cost using the variables (shell thickness) x_1, (head thickness) x_2, (inner radius) $x_3$and (length of the cylindrical vessel minus the head part) x_4. The PVD problem is shown in Fig.14.

TABLE 6. 30D CEC 2014 benchmark results for CMACGSA and other popular and SOTA algorithms.

		WOA	PSO	CCS	COA	DA	SCA	CMACGSA
F1	Mean	3.24e+7	8.19e+7	2.93e+9	1.03e+8	4.49e+8	2.24e+8	1.38e+6
	Rank	2	3	7	4	6	5	1
F2	Mean	5.05e+6	2.81e+4	9.48E+10	2.06e+9	3.67E+10	1.60E+10	60428.06
	Rank	3	1	7	4	6	5	2
F3	Mean	2.97e+4	20.9	1.01e+7	1.78e+5	1.81e+5	3.79e+4	31412.40
	Rank	2	1	7	5	6	4	3
F4	Mean	176.00	9010.00	2.43e+4	398.00	4860.00	1020.00	119.07
	Rank	2	6	7	3	5	4	1
F5	Mean	20.40	219.00	503.00	488.00	20.80	20.90	20.91
	Rank	1	5	7	6	2	3	4
F6	Mean	34.80	6780.00	604.00	437.00	37.20	34.40	25.03
	Rank	3	7	6	5	4	2	1
F7	Mean	1.03	1.61	1660.00	701.00	354.00	133.00	0.14
	Rank	2	3	7	6	5	4	1
F8	Mean	176.00	1100.00	1160.00	421.00	289.00	236.00	236.12
	Rank	5	6	7	4	3	1	2
F9	Mean	213.00	2.30e+6	1330.00	902.00	301.00	267.00	130.34
	Rank	2	7	6	5	4	3	1
F10	Mean	3770.00	44.30	8060.00	1870.00	6570.00	5880.00	5397.06
	Rank	3	1	7	2	6	5	4
F11	Mean	4500.00	6.50e+5	8830.00	4980.00	6920.00	7040.00	6452.47
	Rank	1	7	6	2	4	5	3
F12	Mean	1.67	373.00	1210.00	1210.00	2.21	2.45	2.21
	Rank	1	4	5	5	2	3	2
F13	Mean	0.50	214.00	1310.00	1310.00	5.14	2.89	0.31
	Rank	2	5	6	6	4	3	1
F14	Mean	0.28	905.00	1670.00	1410.00	131.00	41.10	0.36
	Rank	1	5	7	6	4	3	2
F15	Mean	70.00	1.04e+7	1.81e+6	3.84e+4	2.85e+5	2820.00	12.30
	Rank	2	7	6	4	5	3	1
F16	Mean	1.26e+1	2.49e+7	1610.00	1610.00	1.32e+1	1.28e+1	12.34
	Rank	2	6	5	5	4	3	1
F17	Mean	4.33e+6	1.26e+4	3.82e+8	6.41e+6	1.61e+7	6.61e+6	146397.25
	Rank	3	1	7	4	6	5	2
F18	Mean	1.55e+4	0.05	1.21E+10	2.18e+8	6.11e+8	1.85e+8	1058.56
	Rank	3	1	7	5	6	4	2
F19	Mean	44.80	29.10	2730.00	2540.00	268.00	90.80	14.05
	Rank	3	2	7	6	5	4	1
F20	Mean	2.04e+4	13.40	5.99e+6	8.81e+4	2.78e+5	1.31e+4	12315.21
	Rank	4	1	7	5	6	3	2
F21	Mean	9.48e+5	388.00	1.85e+8	2.51e+5	8.23e+6	1.48e+6	29317.72
	Rank	3	1	7	5	6	4	2
F22	Mean	749.00	0.74	3.43e+6	8.21e+4	1.12e+3	754.00	620.36
	Rank	3	1	7	6	5	4	2
F23	Mean	331.00	2940.00	3730.00	2630.00	576.00	370.00	315.25
	Rank	2	6	7	5	4	3	1
F24	Mean	206.00	1.05e+6	2530.00	2410.00	281.00	201.00	225.64
	Rank	2	7	6	5	4	1	3
F25	Mean	225.00	16.40	2660.00	5290.00	240.00	227.00	210.81
	Rank	3	1	6	7	5	4	2
F26	Mean	100.00	2.83e+5	2790.00	2620.00	104.00	102.00	119.92
	Rank	1	7	6	5	3	2	4
F27	Mean	905.00	22.80	7070.00	2720.00	1090.00	722.00	543.02
	Rank	4	1	7	6	5	3	2
F28	Mean	2150.00	2.59	1.41e+4	2820.00	2240.00	2000.00	2937.73
	Rank	3	1	7	5	4	2	6
F29	Mean	4.38e+6	271.00	1.10e+9	1.26e+7	2.15e+7	1.34e+7	8720.35
	Rank	3	1	7	4	6	5	2
F30	Mean	8.20e+4	1.18e+7	2.35e+7	3.88e+5	4.79e+5	2.48e+5	3389.66
	Rank	2	6	7	4	5	3	1
Average rank		**2.43**	**3.70**	**6.60**	**4.80**	**4.60**	**3.43**	**2.07**
Overall rank		**2**	**4**	**7**	**6**	**5**	**3**	**1**

TABLE 7. The used parameters and their values for four GSA variants in comparison with CMACGSA.

Algorithm	Parameters
SCGSA [74]	α =20, G_0=100, ε=1e-6, $k = 2 \times (1 - t/T)$
BA-CGSA [75]	α =20, G_0=100, ε=1e-6, $k = c - c \times t/T$
GEA [76]	α =20, G_0=100, ε=1e-6, $p_{CR} = 0.5$, $\beta_0 = 0.95$, $\lambda = 0.01$
GG-GSA [77]	α =20, G_0=1, ε=1e-6, $\acute{c}_1 = (-2t^3/T^3) + 2$, $\acute{c}_2 = (-2t^3/T^3)$

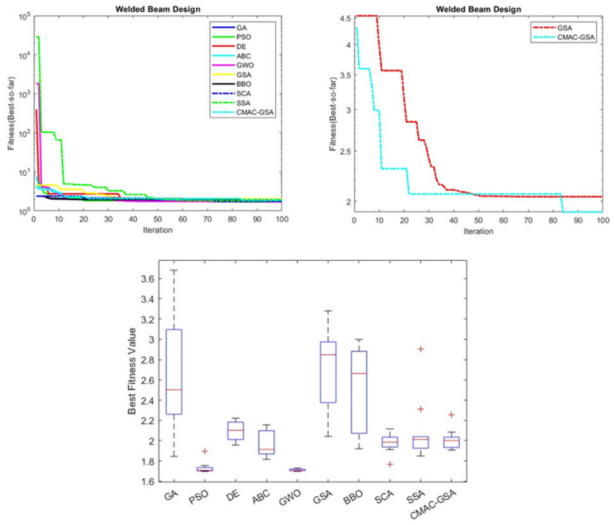

FIGURE 13. Convergence curves and boxplot graph of the results for WBD problem.

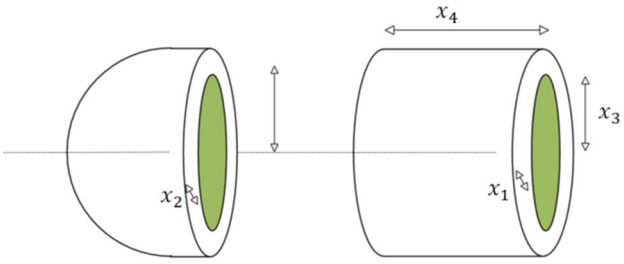

FIGURE 14. Pressure vessel design problem.

The PVD problem are given in Eq.'s. (39)-(43):

$$\min F(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.166x_1^2x_4 + 19.84x_1^2x_3 \quad (39)$$

$$G_1 = -x_1 + 0.0193x3 \quad (40)$$

$$G_2 = -x3 + 0.00954x3 \quad (41)$$

$$G_3 = -\pi x_3^2 x4 - (4/3)\pi x_3^3 + 1296000 \quad (42)$$

$$G_4 = x^4 - 240 \quad (43)$$

where $F(x)$ represents the cost function, G_1, G_2, G_3, and G_4 denote the constraint functions of this problem. Table 11 summarizes the optimization results of the PVD problem. From the results, CMACGSA demonstrates a competitive yet mixed performance compared to other algorithms. Its best solution has a cost of 2679.1, which is higher than the most efficient algorithms like DE (2302.5) and ABC (2303.1). However, CMACGSA outperforms GSA significantly in terms of both solution quality and consistency. While GSA shows extremely high variability, with a mean of 6.05e+5 and a standard deviation of 3.61e+5, CMACGSA achieves far better stability, reflected in its mean cost of 3192.8 and a much lower standard deviation (SD=477.4).

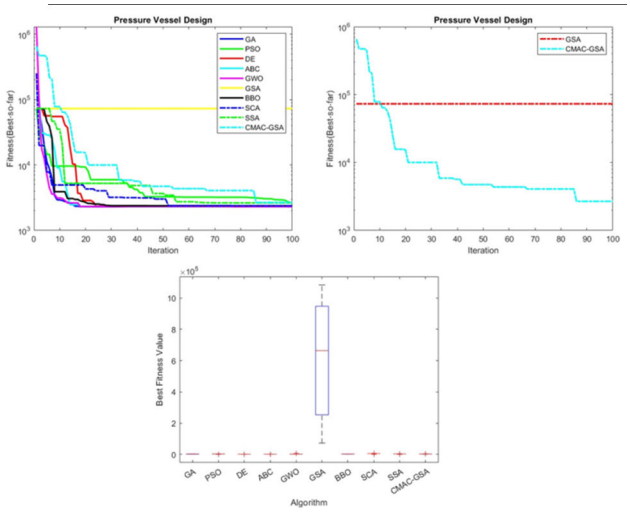

FIGURE 15. Convergence curves and boxplot graph of the results for PVD problem.

Fig. 15 denotes the convergence curves and boxplot graph of all metaheuristics used in PVD problem. The top-left graph shows the convergence curves for the PVD problem across several algorithms. CMACGSA shows rapid convergence in the initial iterations, indicating its ability to quickly approach optimal or near-optimal solutions. In contrast, the original GSA shows a slower and more static convergence trend, suggesting limited adaptability in navigating the search space. The top-right graph compares the convergence behavior of CMACGSA and the original GSA over iterations. CMACGSA consistently refines its solutions while GSA reaches an early stagnation point, highlighting the enhanced search capabilities introduced by CMACGSA's mechanisms such as boundary control, Lévy mutation and CMAC integration. The lower plot shows a statistical comparison of the optimization results. The box plot shows that CMACGSA achieves a higher quality solution set with less variability. This consistency reflects the robustness of CMACGSA in maintaining a balance between exploration and exploitation. The narrower interquartile range for CMACGSA highlights its ability to produce reliable and repeatable solutions compared to the wider distribution of GSA, which implies less stability and accuracy. The results underline the superiority of CMACGSA over GSA in terms of both convergence speed and solution quality. CMACGSA not only outperforms GSA, but also achieves results on a par with other advanced algorithms.

TABLE 8. Results of 30D CEC 2014 benchmark tests for CMACGSA and various variants of GSA.

		SCGSA	BA-CGSA	GEA	GG-GSA	CMACGSA
F1	Mean	4.74e+6	4.47e+6	6.68e+5	4.49e+5	1.38e+6
	Rank	4	4	2	1	3
F2	Mean	2.57e+3	8.22e+3	1.31E-03	1.15e+4	6.04e+4
	Rank	2	4	1	3	5
F3	Mean	1.26e+5	4.17e+4	2.52e+1	4.10e+3	3.14e+4
	Rank	7	5	2	3	4
F4	Mean	5.36e+2	4.91e+2	1.10e+2	4.63e+2	1.19e+2
	Rank	5	4	1	3	2
F5	Mean	5.20e+2	5.20e+2	2.00e+1	5.20e+2	2.09e+1
	Rank	3	3	1	3	2
F6	Mean	6.20e+2	6.16e+2	1.41e+6	6.09e+2	2.50e+1
	Rank	4	3	5	2	1
F7	Mean	7.00e+2	7.00e+2	6.40E-03	7.00e+2	1.40E-01
	Rank	3	3	1	3	2
F8	Mean	9.51e+2	9.03e+2	9.50e+6	8.73e+2	2.36e+2
	Rank	4	3	5	2	1
F9	Mean	1.04e+3	9.99e+2	1.04e+2	978.00	130.34
	Rank	5	4	1	3	2
F10	Mean	6.59e+3	4.16e+3	2.83e+6	3.30e+3	5.40e+3
	Rank	4	2	5	1	3
F11	Mean	7.26e+3	4.80e+3	3.33e+3	3.43e+3	6.45e+3
	Rank	5	3	1	2	4
F12	Mean	1.20e+3	1.20e+3	1.51E-02	1.20e+3	2.21e+0
	Rank	3	3	1	3	2
F13	Mean	1.30e+3	1.30e+3	4.47E-01	1.30e+3	3.13E-01
	Rank	4	3	1	3	2
F14	Mean	1.40e+3	1.40e+3	2.85E-01	1.40e+3	3.63E-01
	Rank	4	3	1	3	2
F15	Mean	1.52e+3	1.51e+3	3.77e+0	1.50e+3	1.23e+1
	Rank	5	3	1	4	2
F16	Mean	1.62e+3	1.61e+3	1.26e+6	1.61e+3	1.23e+1
	Rank	4	3	5	2	1
F17	Mean	1.28e+6	2.24e+5	2.14e+5	4.37e+4	1.46e+5
	Rank	5	4	3	1	2
F18	Mean	2.96e+3	2.34e+3	4.35e+4	2.32e+3	1.06e+3
	Rank	4	3	5	2	1
F19	Mean	1922.90	1.92e+3	2.71e+1	1.92e+3	1.40e+1
	Rank	5	3	2	4	1
F20	Mean	4.21e+4	2.72e+4	5.66e+6	1.41e+4	1.23e+4
	Rank	4	3	5	2	1
F21	Mean	1.32e+6	1.68e+5	3.68e+4	3.78e+4	2.93e+4
	Rank	5	4	1	2	3
F22	Mean	3.55e+3	3.21e+3	7.24e+6	3.10e+3	6.20e+2
	Rank	4	2	5	3	1
F23	Mean	2.65e+3	2.62e+3	3.16e+2	2.62e+3	3.15e+2
	Rank	5	4	1	3	2
F24	Mean	2.66e+3	2.61e+3	2.30e+6	2.60e+3	2.26e+2
	Rank	4	2	5	3	1
F25	Mean	2.70e+3	2.70e+3	2.16e+2	2.70e+3	2.11e+2
	Rank	3	3	2	3	1
F26	Mean	2.80e+3	2.79e+3	1.65e+6	2.80e+3	1.20e+2
	Rank	4	2	5	3	1
F27	Mean	3.73e+3	3.72e+3	6.54e+2	3.33e+3	5.43e+2
	Rank	5	4	2	3	1
F28	Mean	8.37e+3	6.01e+3	2.39e+6	5.45e+3	2.94e+3
	Rank	4	2	5	3	1
F29	Mean	1.18e+4	6.53e+3	5.15e+6	4.08e+3	8.72e+3
	Rank	4	2	5	1	3
F30	Mean	5.76e+4	1.14e+4	23354.46	5.24e+3	3.39e+3
	Rank	5	3	4	1	2
Average rank		**5.96**	**4.76**	**4.13**	**4.06**	**3.26**
Overall rank		**5**	**4**	**3**	**2**	**1**

TABLE 9. The used parameters and their values for other existing metaheuristic algorithms in comparison with cmacgsa in engineering design problems.

Algorithm	Parameters
ABC [9]	Trial Limit=0.6 × D × Pop
BBO [78]	Keep rate=0.2, α=0.9, p_m=0.1, sigma=0.02×(U_b-L_b)
DE [24]	β_{min}=0.2, β_{max}=0.8, p_{CR}=0.2
GA [21]	p_c = 0.95, p_m = 0.1
GSA [19]	α =20, G_0=100, ε=1e-6
GWO [11]	$a = 2 - t(2/T)$
PSO [26]	c_1=c_2=2
SCA [73]	a=2
SSA [79]	$c_1 = 2\exp(-(4t/T)^2)$

3) THREE BAR TRUSS DESIGN

The Three Bar Truss (TBT) design is one of the widely used unrestricted engineering problems. This problem has three parameters and aims to find the minimum cost weight [29], [84]. In this problem, considering the tension on each of the truss elements over the x_1 and x_2 values positioned diagonally, the volume of the truss is expected to be at the optimum value. The design of the problem is presented in Fig. 16.

$$(x_1 = x_3)$$

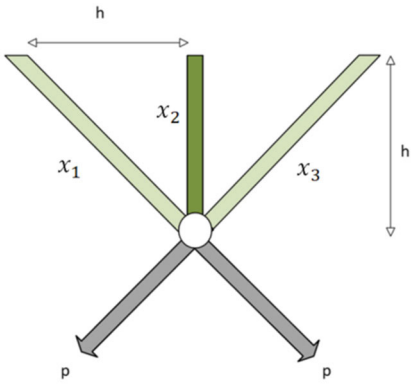

FIGURE 16. Three bar truss design problem.

The TBT problem are as follows in Eq.'s. (44)-(47):

$$\min F(x) = L * (2\sqrt{2}x_1 + x_2) \tag{44}$$

$$G_1 = \frac{\sqrt{2}\,x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - RU \tag{45}$$

$$G_2 = \frac{x_2}{\sqrt{2}\,x_1^2 + 2x_1x_2}P - RU \tag{46}$$

$$G_3 = \frac{1}{x_1 + \sqrt{2}\,x_2}P - RU \tag{47}$$

where $F(x)$ represents the cost function, G_1, G_2, and G_3 represent the constraint functions of this problem, $L = 100$ is the bar length, $P = 2$ is the load, and $RU = 2$ is the allowable stress ratio. Table 12 gives the comparison results for TBT problem. The proposed algorithm (CMACGSA) offers excellent consistency and stability, comparable to the top-performing algorithms (like PSO, DE, and SSA). However, its best cost (186.630), while very close, does not quite match the optimal value of 186.386 achieved by some other algorithms. This suggests that CMACGSA is well-suited for scenarios where stability and repeatability are more critical than achieving the absolute minimum cost in every case. Future improvements could focus on fine-tuning the algorithm's search mechanisms to enhance its ability to find the global optimum.

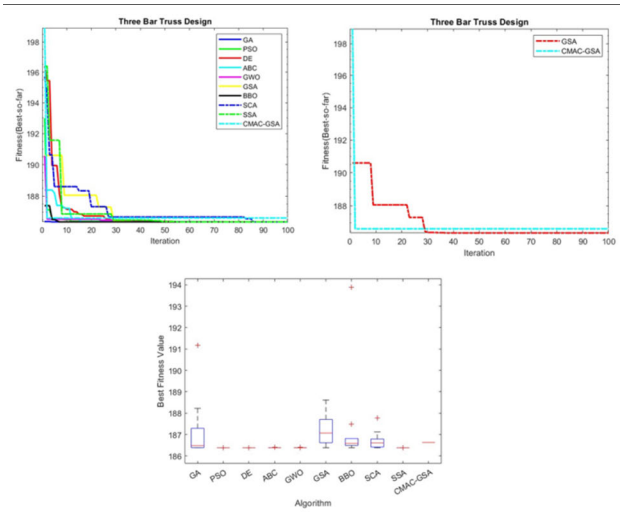

FIGURE 17. Convergence curves and boxplot graph of the results for TBT problem.

In Fig. 17, the convergence curves and boxplot results for the TBT design problem are presented, providing a detailed comparison of the performance of CMACGSA, the original GSA, and other optimization algorithms. CMACGSA shows a rapid decrease in objective values, particularly in the early iterations, demonstrating its ability to efficiently explore the solution space and converge to near-optimal solutions with minimal computational effort. In comparison, the original GSA exhibits a much slower convergence rate and struggles to make significant progress after the first few iterations. This reflects its limited ability to effectively balance exploration and exploitation for this particular problem. The results presented in Fig. 17 show that CMACGSA delivers performance comparable to other SOTA algorithms, while significantly outperforming the original GSA in terms of convergence speed, solution quality and consistency. These improvements can be attributed to the advanced features incorporated in CMACGSA, such as the intelligent mass-learning capability of the CMAC neural network.

TABLE 10. Results of CMACGSA and other Metaheuristics for WBD problem.

Algorithm	Best Solution				Cost				
	x_1	x_2	x_3	x_4	Best	Worst	Median	Mean	SD
GA	0.2022	3.5333	8.4084	0.2376	1.8450	3.6806	2.5013	2.6848	0.5971
PSO	0.2049	3.2640	9.0520	0.2057	1.6977	1.8965	1.7069	1.7317	0.0606
DE	0.2169	3.1770	9.9664	0.2176	1.9571	2.2219	2.1033	2.0949	0.0968
ABC	0.1747	4.3112	9.1855	0.2065	1.8166	2.1563	1.9144	1.9645	0.1218
GWO	0.2056	3.2595	9.0478	0.2057	1.6978	1.7296	1.7074	1.7103	0.0100
GSA	0.2966	2.4484	7.6867	0.2966	2.0424	3.2792	2.8482	2.7212	0.4398
BBO	0.1870	3.8924	8.1776	0.2515	1.9207	2.9974	2.6626	2.5302	0.4042
SCA	0.1952	3.4905	9.3547	0.2061	1.7691	2.1159	1.9859	1.9764	0.0931
SSA	0.2105	3.4400	8.3871	0.2388	1.8490	2.9062	2.0128	2.0939	0.3120
CMACGSA	0.1891	3.9672	9.2266	0.2196	1.9082	2.2548	2.0009	2.0146	0.1001

TABLE 11. Results of CMACGSA and other Metaheuristics for PVD problem.

Algorithm	Best Solution				Cost				
	x_1	x_2	x_3	x_4	Best	Worst	Median	Mean	SD
GA	1.0969	0	65.226	10.00	2302.7	3611	3089.8	2925.4	525
PSO	1.0828	0.0073	64.636	16.274	2598.3	3638.3	3509.4	3342.5	392.25
DE	1.0935	0	65.225	10.000	2302.5	2309.2	2302.8	2303.8	2.2
ABC	1.0932	0	65.231	10.000	2303.1	2310.5	2303.7	2304.8	2.43
GWO	1.1032	0	65.236	10.002	2304.7	6123.3	2356.6	3094.6	1582.42
GSA	3.0668	5.744	73.784	21.89	7.31e+4	1.08e+6	6.64e+5	6.05e+5	3.61e+5
BBO	1.0755	0	64.477	13.270	2386.3	4.e+3	3151.8	3070.9	4.e+2
SCA	1.1713	0	65.402	10.000	2383.2	6156.4	6072.1	5346.8	1561.1
SSA	1.0172	0	62.074	24.296	2635.7	3637.3	3625.6	3523.1	312.41
CMACGSA	1.1683	0	66.873	10.459	2679.1	4185	3085.7	3192.8	477.4

TABLE 12. Results of CMACGSA and other Metaheuristics for TBT problem.

Algorithm	Best Solution		Cost				
	x_1	x_2	Best	Worst	Median	Mean	SD
GA	0.7870	0.2877	186.386	191.176	186.486	187.203	1.516
PSO	0.7868	0.2880	186.386	186.386	186.386	186.386	0.000
DE	0.7868	0.2880	186.386	186.387	186.386	186.386	0.000
ABC	0.7867	0.2883	186.386	186.398	186.387	186.388	0.003
GWO	0.7870	0.2877	186.386	186.400	186.387	186.389	0.004
GSA	0.7859	0.2898	186.387	188.612	187.071	187.204	0.715
BBO	0.7866	0.2884	186.386	193.877	186.585	187.395	2.299
SCA	0.7862	0.2893	186.386	187.773	186.612	186.728	0.432
SSA	0.7868	0.2880	186.386	186.386	186.386	186.386	0.000
CMACGSA	0.7857	0.2857	186.630	186.630	186.630	186.630	0.000

V. CONCLUSION

In this study, we introduced CMACGSA, an extended version of GSA that incorporates a CMAC neural network. By exploiting the quantization and generalization capabilities of the CMAC neural network, CMACGSA improves the performance of traditional GSA by effectively addressing issues such as early convergence and local minimum entropy. Additional mechanisms, including a Lévy mutation operator,

boundary control, and error tracking, further enhance the robustness and adaptability of the algorithm, enabling it to reliably find optimal solutions to complex optimization problems. Our experimental evaluation on the CEC 2014 benchmark functions (run in 10D, 30D and 50D) has shown that CMACGSA consistently outperforms the original GSA and compares favourably with popular SOTA algorithms and GSA variants, particularly in the 30D results. Further testing on three real-world design problems confirmed the competitive advantages of CMACGSA, demonstrating its robustness and effectiveness across different optimization scenarios.

The integration of the CMAC neural network into the GSA framework introduces a novel approach to particle/agent mass optimization, allowing for more accurate and efficient convergence to the global optimum. This study represents a pioneering example of augmenting an optimization algorithm with a neural network, and sets a precedent for future research in hybrid metaheuristic optimization techniques. Future work could explore adaptive mechanisms for dynamically adjusting CMAC parameters to further enhance learning and generalization. In addition, extending CMACGSA to a wider range of real-world optimization problems may provide deeper insights into its practical scalability. Investigating other neural network models or machine learning techniques may also provide new directions for improving GSA and similar metaheuristic algorithms.

In conclusion, CMACGSA represents a promising advance in metaheuristic optimization. By successfully integrating the CMAC neural network with GSA, it provides a powerful tool for tackling complex optimization challenges, outperforming traditional methods and laying the foundation for continued innovation in hybrid optimization techniques in engineering and beyond.

ACKNOWLEDGMENT

The authors declare that no known competing financial interests and conflicts of interest in this study.

REFERENCES

[1] X. Han, Z. Li, and Y. Xu, "Quantum assisted stochastic economic dispatch for renewables rich power systems," 2024, *arXiv:2404.13073*.

[2] X. Han and Y. Zhang, "Decomposition-coordination-based voltage control for high photovoltaic-penetrated distribution networks under cloud-edge collaborative architecture," *Int. Trans. Electr. Energy Syst.*, vol. 2022, no. 1, Jan. 2022, Art. no. 7280220.

[3] T. Yang, X. Han, H. Li, W. Li, and A. Y. Zomaya, "Parallel scientific power calculations in cloud data center based on decomposition-coordination directed acyclic graph," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 2491–2502, Mar. 2022.

[4] Z. Guan, C. Ren, J. Niu, P. Wang, and Y. Shang, "Great wall construction algorithm: A novel meta-heuristic algorithm for engineer problems," *Expert Syst. Appl.*, vol. 233, Dec. 2023, Art. no. 120905.

[5] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the Metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges," *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 13187–13257, Nov. 2023.

[6] J. Li, Y. Hu, B. Ma, and D. Wang, "MBSCSO: Multi-strategy boosted sand cat swarm optimization for engineering applications," *IEEE Access*, vol. 12, pp. 153743–153782, 2024.

[7] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern., B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[8] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Dept. Computer Engineering Dept., Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.

[10] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214.

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Jan. 2014.

[12] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, Feb. 2016.

[13] X. Yang, "A new Metaheuristic bat-inspired algorithm," in *Proc. Nature Inspired Cooperat. Strategies Optim. (NICSO)*. Springer, Jan. 2010, pp. 65–74.

[14] W. Zhao, L. Wang, Z. Zhang, H. Fan, J. Zhang, S. Mirjalili, N. Khodadadi, and Q. Cao, "Electric eel foraging optimization: A new bio-inspired optimizer for engineering applications," *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122200.

[15] M. H. Amiri, N. Mehrabi Hashjin, M. Montazeri, S. Mirjalili, and N. Khodadadi, "Hippopotamus optimization algorithm: A novel nature-inspired optimization algorithm," *Sci. Rep.*, vol. 14, no. 1, p. 5032, Feb. 2024.

[16] H. Jia, H. Rao, C. Wen, and S. Mirjalili, "Crayfish optimization algorithm," *Artif. Intell. Rev.*, vol. 56, no. 2, pp. 1919–1979, Nov. 2023.

[17] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Inf. Sci.*, vol. 222, pp. 175–184, Feb. 2013.

[18] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: Charged system search," *Acta Mechanica*, vol. 213, nos. 3–4, pp. 267–289, Sep. 2010.

[19] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, Jun. 2009.

[20] I. Birbil and S.-C. Fang, "An electromagnetism-like mechanism for global optimization," *J. Global Optim.*, vol. 25, no. 3, pp. 263–282, 2003.

[21] D. E. Goldberg, *Genetic algorithms*. India: Pearson Education, 2013.

[22] J. Koza, "Genetic programming as a means for programming computers by natural selection," *Statist. Comput.*, vol. 4, no. 2, pp. 87–112, Jun. 1994.

[23] D. B. Fogel, *Artificial Intelligence Through Simulated Evolution*. Hoboken, NJ, USA: Wiley, 1998.

[24] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[25] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent Comput. Program*, vol. 826, no. 1989, p. 37, 1989.

[26] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, vol. 4, Nov. 2002, pp. 1942–1948.

[27] E. Bonabeau, *Swarm Intelligence: From Natural To Artificial Systems*, vol. 2. London, U.K.: Oxford Univ. Press, 1999, pp. 25–34.

[28] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, p. 78, 2010.

[29] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.

[30] F. A. Hashim and A. G. Hussien, "Snake optimizer: A novel meta-heuristic optimization algorithm," *Knowl.s-Based Syst.*, vol. 242, Apr. 2022, Art. no. 108320.

[31] V. Tomar, M. Bansal, and P. Singh, "Metaheuristic algorithms for optimization: A brief review," *Eng. Proc.*, vol. 59, no. 1, p. 238, Mar. 2024.

[32] W. Zhao, L. Wang, Z. Zhang, S. Mirjalili, N. Khodadadi, and Q. Ge, "Quadratic interpolation optimization (QIO): A new optimization algorithm based on generalized quadratic interpolation and its applications to real-world engineering problems," *Comput. Methods Appl. Mech. Eng.*, vol. 417, Dec. 2023, Art. no. 116446.

[33] J. S. Albus, "Data storage in the cerebellar model articulation controller (CMAC)," *J. Dyn. Syst., Meas., Control*, vol. 97, no. 3, pp. 228–233, Sep. 1975.

[34] M. B. Dowlatshahi and H. Nezamabadi-Pour, "GGSA: A grouping gravitational search algorithm for data clustering," *Eng. Appl. Artif. Intell.*, vol. 36, pp. 114–121, Nov. 2014.

[35] H. Harandizadeh, D. J. Armaghani, and E. T. Mohamad, "Development of fuzzy-GMDH model optimized by GSA to predict rock tensile strength based on experimental datasets," *Neural Comput. Appl.*, vol. 32, no. 17, pp. 14047–14067, Sep. 2020.

[36] A. Bahrololoum, H. Nezamabadi-Pour, H. Bahrololoum, and M. Saeed, "A prototype classifier based on gravitational search algorithm," *Appl. Soft Comput.*, vol. 12, no. 2, pp. 819–825, Feb. 2012.

[37] S. Özyön and C. Yasar, "Gravitational search algorithm applied to fixed head hydrothermal power system with transmission line security constraints," *Energy*, vol. 155, pp. 392–407, Jul. 2018.

[38] M. A. Mosa, "Real-time data text mining based on gravitational search algorithm," *Expert Syst. Appl.*, vol. 137, pp. 117–129, Dec. 2019.

[39] A. Singh and K. Deep, "Hybridizing gravitational search algorithm with real coded genetic algorithms for structural engineering design problem," *Opsearch*, vol. 54, no. 3, pp. 505–536, Sep. 2017.

[40] C. Li, J. Zhou, P. Lu, and C. Wang, "Short-term economic environmental hydrothermal scheduling using improved multi-objective gravitational search algorithm," *Energy Convers. Manage.*, vol. 89, pp. 127–136, Jan. 2015.

[41] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *Proc. Int. Conf. Comput. Inf. Appl.*, Dec. 2010, pp. 374–377.

[42] H. Garg, "A hybrid GSA-GA algorithm for constrained optimization problems," *Inf. Sci.*, vol. 478, pp. 499–523, Apr. 2019.

[43] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A Metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, Jan. 2013.

[44] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, Jun. 2014.

[45] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Jan. 2008.

[46] S. Özyön, C. Yasar, and H. Temurtas, "Incremental gravitational search algorithm for high-dimensional benchmark functions," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 3779–3803, Aug. 2019.

[47] H. A. Shehadeh, "A hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) for global optimization," *Neural Comput. Appl.*, vol. 33, no. 18, pp. 11739–11752, Sep. 2021.

[48] S. Mirjalili and A. Lewis, "Adaptive gbest-guided gravitational search algorithm," *Neural Comput. Appl.*, vol. 25, nos. 7–8, pp. 1569–1584, Dec. 2014.

[49] M. K. Naik and R. Panda, "A new hybrid CS-GSA algorithm for function optimization," *Positions*, vol. 1, no. 1, p. 12, Jan. 2015.

[50] F. Zhao, F. Xue, Y. Zhang, W. Ma, C. Zhang, and H. Song, "A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution," *Expert Syst. Appl.*, vol. 113, pp. 515–530, Dec. 2018.

[51] K. Price, *Differential Evolution: A Practical Approach To Global Optimization*. Springer, 2006.

[52] M. Taradeh, M. Mafarja, A. A. Heidari, H. Faris, I. Aljarah, S. Mirjalili, and H. Fujita, "An evolutionary gravitational search-based feature selection," *Inf. Sci.*, vol. 497, pp. 219–239, Sep. 2019.

[53] C. Li, X. An, and R. Li, "A chaos embedded GSA-SVM hybrid system for classification," *Neural Comput. Appl.*, vol. 26, no. 3, pp. 713–721, Apr. 2015.

[54] R. Guha, M. Ghosh, A. Chakrabarti, R. Sarkar, and S. Mirjalili, "Introducing clustering based population in binary gravitational search algorithm for feature selection," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106341.

[55] C. Li, H. Li, and P. Kou, "Piecewise function based gravitational search algorithm and its application on parameter identification of AVR system," *Neurocomputing*, vol. 124, pp. 139–148, Jan. 2014.

[56] A. Özden and I. Iseri, "COOT optimization algorithm on training artificial neural networks," *Knowl. Inf. Syst.*, vol. 65, no. 8, pp. 3353–3383, Aug. 2023.

[57] N. Nezamoddini and A. Gholami, "Integrated genetic algorithm and artificial neural network," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Aug. 2019, pp. 260–262.

[58] E. Bas, E. Egrioglu, and O. Karahasan, "A pi-sigma artificial neural network based on sine cosine optimization algorithm," *Granular Comput.*, vol. 7, no. 4, pp. 813–820, Oct. 2022.

[59] M.-L. Huang and Y.-C. Chou, "Combining a gravitational search algorithm, particle swarm optimization, and fuzzy rules to improve the classification performance of a feed-forward neural network," *Comput. Methods Programs Biomed.*, vol. 180, Oct. 2019, Art. no. 105016.

[60] A. Naserbegi and M. Aghaie, "Multi-objective optimization of hybrid nuclear power plant coupled with multiple effect distillation using gravitational search algorithm based on artificial neural network," *Thermal Sci. Eng. Prog.*, vol. 19, Oct. 2020, Art. no. 100645.

[61] E. Momeni, A. Yarivand, M. B. Dowlatshahi, and D. J. Armaghani, "An efficient optimal neural network based on gravitational search algorithm in predicting the deformation of geogrid-reinforced soil structures," *Transp. Geotechnics*, vol. 26, Jan. 2021, Art. no. 100446.

[62] Z. Kong, Y. Zhang, X. Wang, Y. Xu, and B. Jin, "Prediction and optimization of a desulphurization system using CMAC neural network and genetic algorithm," *J. Environ. Eng. Landscape Manage.*, vol. 28, no. 2, pp. 74–87, Apr. 2020.

[63] Z.-J. Lee, Y.-P. Wang, and S.-F. Su, "A genetic algorithm based robust learning credit assignment cerebellar model articulation controller," *Appl. Soft Comput.*, vol. 4, no. 4, pp. 357–367, Sep. 2004.

[64] J. Tu and S. Cao, "PMSM driver based on hybrid particle swarm optimization and CMAC," *Phys. Proc.*, vol. 33, pp. 983–990, Jan. 2012.

[65] S. Han and X. Sun, "Optimizing product design using genetic algorithms and artificial intelligence techniques," *IEEE Access*, vol. 12, pp. 151460–151475, 2024.

[66] W. T. Miller, F. H. Glanz, and L. G. Kraft, "CMAC: An associative neural network alternative to backpropagation," *Proc. IEEE*, vol. 78, no. 10, pp. 1561–1567, Oct. 1990.

[67] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *J. Dyn. Syst., Meas., Control*, vol. 97, no. 3, pp. 220–227, Sep. 1975.

[68] J. C. Jan and S.-L. Hung, "High-order MS CMAC neural network," *IEEE Trans. Neural Netw.*, vol. 12, no. 3, pp. 598–603, May 2001.

[69] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.

[70] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Dept. Computational Intelligence Laboratory, Zhengzhou Univ., Nanyang Technological Univ., Zhengzhou, China, Tech. Rep. 201311, 2013, vol. 635, no. 2, p. 2014.

[71] G.-G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "Chaotic cuckoo search," *Soft Comput.*, vol. 20, no. 9, pp. 3349–3362, Sep. 2016.

[72] J. Pierezan and L. D. S. Coelho, "Coyote optimization algorithm: A new Metaheuristic for global optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.

[73] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.

[74] J. Jiang, R. Jiang, X. Meng, and K. Li, "SCGSA: A sine chaotic gravitational search algorithm for continuous optimization problems," *Expert Syst. Appl.*, vol. 144, Apr. 2020, Art. no. 113118.

[75] J. Jiang, X. Yang, X. Meng, and K. Li, "Enhance chaotic gravitational search algorithm (CGSA) by balance adjustment mechanism and sine randomness function for continuous optimization problems," *Phys. A, Stat. Mech. Appl.*, vol. 537, Jan. 2020, Art. no. 122621.

[76] H. J. Neamah, A. M. Almobarqaa, and Z. A. Abdulhusien, "Gravitational evaluation algorithm for global optimization problem," *Int. J. Nonlinear Anal. Appl.*, vol. 13, no. 2, pp. 345–359, 2022.

[77] V. K. Bohat and K. V. Arya, "An effective gbest-guided gravitational search algorithm for real-parameter optimization and its application in training of feedforward neural networks," *Knowl.-Based Syst.*, vol. 143, pp. 192–207, Mar. 2018.

[78] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Aug. 2008.

[79] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[80] D. Zou, H. Liu, L. Gao, and S. Li, "A novel modified differential evolution algorithm for constrained optimization problems," *Comput. Math. Appl.*, vol. 61, no. 6, pp. 1608–1623, Mar. 2011.

[81] A. Askarzadeh, "A novel Metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, Jun. 2016.

[82] T. Ray and K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 386–396, Aug. 2003.
[83] E.-S.-M. El-Kenawy, N. Khodadadi, S. Mirjalili, A. A. Abdelhamid, M. M. Eid, and A. Ibrahim, "Greylag goose optimization: Nature-inspired optimization algorithm," *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122147.
[84] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.

NAZMIYE EBRU BULUT was born in Afyon, Türkiye, in 1992. She received the B.S. degree in computer engineering from Dumlupinar University, in 2011, and the M.S. degree from Bilecik Şeyh Edebali University, where she is currently pursuing the Ph.D. degree in electronics and computer engineering. She was a Lecturer with the Department of Information Processing, Afyon Kocatepe University, from 2019 to 2021. Since 2021, she has been a Lecturer with the Distance Education Application Research Center, Konya Technical University. Her working areas are computer graphics, optimization, and heuristic algorithms.

EMRE DANDIL received the bachelor's degree in electronic and computer sciences, in 2007, the M.Sc. degree from Süleyman Demirel University, Türkiye, and the Ph.D. degree in computer engineering from Sakarya University, Türkiye, in 2015. He joined Bilecik Şeyh Edebali University, as a Lecturer, in 2010, having previously taught at the Ministry of Education for three years. He also visited the Center for Secure Information Technologies (CSIT), Institute of Electronics, Communication and Information Technology (ECIT), Queen's University Belfast, U.K., to research postdoctoral study as a Visiting Researcher, from 2016 to 2017. He has studied there in a project relating to fetal movement detection on US scans. He is currently an Associate Professor with the Department of Computer Engineering, Bilecik Şeyh Edebali University. His research interests include artificial intelligence, image processing, pattern recognition, biomedical, machine learning, computer-aided detection, medical informatics, data mining, and computer vision.

UGUR YUZGEC was born in Adilcevaz, Bitlis, Türkiye, in May 1974. He received the B.Sc. degree in electronics and communication engineering from Yıldız Technical University, Istanbul, Türkiye, in 1995, and the M.Sc. and Ph.D. degrees in electronics and communication engineering from Kocaeli University, Kocaeli, Türkiye, in 1999 and 2005, respectively. From 1998 to 2010, he was a Research Assistant with the Department of Electronics and Communication Engineering, Kocaeli University. Since 2020, he has been a Professor with the Department of Computer Engineering, Faculty of Engineering, Bilecik Şeyh Edebali University, Türkiye. His research interests include intelligent systems and control, fuzzy and neuro-fuzzy systems, metaheuristic algorithms, unmanned aerial vehicles, and numerical techniques for optimization problems.

ALPASLAN DUYSAK received the master's degree in electrical engineering from Pennsylvania State University, USA, in 1997, and the Ph.D. degree from Bournemouth University, U.K., in 2004. He is currently an Instructional Associate Professor with the Computer Science and Engineering Department, Texas A&M University. His research interests include deformation algorithms, real-time rendering, optimization algorithms, and machine learning.