scientific reports

Check for updates

OPEN

Considering radial basis function neural network for effective solution generation in metaheuristic algorithms

Erik Cuevas✉, Cesar Rodolfo Ascencio-Piña, Marco Pérez & Bernardo Morales-Castañeda

In many engineering optimization problems, the number of function evaluations is severely limited by the time or cost constraints. These limitations present a significant challenge in the field of global optimization, because existing metaheuristic methods typically require a substantial number of function evaluations to find optimal solutions. This paper presents a new metaheuristic optimization algorithm that considers the information obtained by a radial basis function neural network (RBFNN) in terms of the objective function for guiding the search process. Initially, the algorithm uses the maximum design approach to strategically distribute a set of solutions across the entire search space. It then enters a cycle in which the RBFNN models the objective function values from the current solutions. The algorithm identifies and uses key neurons in the hidden layer that correspond to the highest objective function values to generate new solutions. The centroids and standard deviations of these neurons guide the sampling process, which continues until the desired number of solutions is reached. By focusing on the areas of the search space that yield high objective function values, the algorithm avoids exhaustive solution evaluations and significantly reduces the number of function evaluations. The effectiveness of the method is demonstrated through a comparison with popular metaheuristic algorithms across several test functions, where it consistently outperforms existing techniques, delivers higher-quality solutions, and improves convergence rates.

Keywords Metaheuristic optimization, Radial basis function neural networks (RBFNN), Objective function analysis, Solution space exploration

Optimization[1] is the mathematical discipline focused on finding the best solution to a problem from a set of available alternatives, guided by a specific set of criteria or objectives. It involves selecting the most effective option among various possibilities with the aim of achieving the maximum efficiency, minimum cost, or other desired outcomes. This process is integral to decision-making across a wide range of fields, including engineering, economics, finance, logistics, and more. At its core, optimization seeks to identify the optimal conditions or variables that satisfy given constraints, maximizing or minimizing a target function.

In the past, classical optimization methods[2], such as gradient descent and linear programming, were sufficient to tackle the limited set of optimization challenges available, given the modest computational resources that were then available. These traditional techniques are applicable only when the problem is convex and derivable, which ensures that solutions can be found within the mathematical constraints upon which these methods are based[3]. However, with the significant advancements in computational power, the scope and complexity of problems that can be optimized have expanded. This development has led to the emergence of complex issues that fall outside the applicability of classical methods due to their inherent limitations and the specific requirements of these new challenges[4]. As a result, there is a need to develop more advanced and flexible optimization strategies that can navigate the complex landscapes of modern optimization problems.

A metaheuristic method[5] is an advanced optimization strategy designed to solve complex problems that are beyond the reach of classical optimization techniques. Unlike classical methods, which often require specific conditions like convexity and differentiability to guarantee finding an optimal solution, metaheuristics do not rely on such strict mathematical properties[6]. These methods provide flexible, heuristic solutions to a wide range of optimization problems, often using iterative processes to explore and exploit the search space. Most metaheuristic

Departamento de Computación, Universidad de Guadalajara, CUCEI, Av. Revolución, 1500, Guadalajara, Jal, México. ✉email: erik.cuevas@academicos.udg.mx

methods draw inspiration from metaphors[7], often rooted in natural phenomena or biological processes, to guide their search strategies for optimization solutions. These algorithms mimic the underlying principles of nature, such as evolution, social behavior, and physical processes, to explore complex solution spaces[8]. Some metaheuristic algorithms are inspired by Darwin's theory of evolution, aligning with the concepts of natural selection and adaptation[9,10]. These algorithms, known as evolutionary algorithms (EAs), include prominent examples like the genetic algorithm (GA) developed by Holland[11], bird mating optimizer (BMO) by Askarzadeh[12], evolution-based strategies (ES) by Schwefel[13], and improved multi-operator differential evolution (IMODE) introduced by Sallam et al.[14]. Additionally, there are algorithms based on swarm intelligence (SI), which emulate the collective intelligence behaviors observed in certain insects and animals critical for their survival[15]. Well-known SI algorithms include the artificial bee colony (ABC) by Karaboga and Basturk[16], ant colony optimization (ACO) by Dorigo and Di Caro[17], particle swarm optimization (PSO) by Eberhart and Kennedy[18], tuna swarm optimization (TSO) by Xie et al.[19], and integrations of particle filter (PF) with PSO by Pozna et al.[20]. Beyond biological inspirations, some metaheuristics draw from physical phenomena, attempting to replicate natural laws like gravity, electromagnetism, and inertia[21]. Examples of these physics-based algorithms include the gravitational search algorithm (GSA) by Rashedi, Nezamabadi-pour, and Saryazdi[22], water evaporation optimization (WEO) by Kaveh and Bakhshpoori[23], and the Black Hole (BH) algorithm by Hatamlou[24]. This diversity of inspirations showcases the breadth of approaches in metaheuristic algorithm development, from biological evolution and social behaviors to the principles governing physical phenomena.

Most metaheuristic methods operate primarily using the information provided by the best solution found in each iteration to guide their strategy[25]. This characteristic is a defining aspect of these algorithms. Although these methods produce several solutions that could provide information about the shape and structure of the objective function, their use is completely discarded[26]. The lack of information from the objective function's landscape can lead to increased search times, as these methods may need to generate and assess an exhaustive number of solutions to thoroughly explore the search space[27]. They navigate through the solution space iteratively, making decisions and adjustments based solely on the performance of the current best solution. This approach means that the algorithms generate potential solutions without any preliminary indication of their quality or effectiveness. The information of each produced solution only becomes apparent after it has been evaluated against the objective function[28]. While this strategy allows for flexibility and the ability to tackle a wide range of problems, it can also result in less efficient search processes, especially when dealing with complex or high-dimensional optimization challenges. The need to extensively sample the search space with scarce information to identify promising regions often translates into higher computational costs and longer times to find optimal or near-optimal solutions. Given the limitations of traditional metaheuristic methods in efficiently exploring the search space, there's a significant demand for algorithms that can intelligently guide the search process by utilizing information from the objective function obtained during the generation of solutions. Incorporating this information can substantially enhance the search efficiency, as it enables the algorithm to focus on more promising regions of the search space.

A radial basis function neural network (RBFNN)[29] is an artificial neural network structured into three layers: an input layer, a hidden layer equipped with radial basis functions, and an output layer. The hidden layer's activation functions are radial basis functions, which are unique mathematical functions determined by the distance between the input data points and certain center points. These center points, along with their corresponding weights, are optimized during the training process, enabling the RBFNN to accurately approximate complex functions and unravel intricate patterns within the data. A notable feature of RBFNNs is their transparency[30,31]; once trained, each neuron in the hidden layer corresponds to a specific region of the data space, indicating its sensitivity and contribution to the model's output. This attribute allows for a clear understanding of how different regions of the data space influence the output, effectively distinguishing between areas that are likely to yield higher output values from those that are not. Such insights can be invaluable for an optimization algorithm, guiding it to prioritize sampling from regions of the space that are more promising, thereby enhancing the efficiency and effectiveness of the search process.

Maximum designs[32] are experimental design techniques focused on ensuring comprehensive coverage across the full spectrum of possible experimental conditions or factors. The primary goal is to maximize the spread of experimental points throughout the design space, thus facilitating a thorough exploration of all potential interactions among the decision variables. These techniques utilize an iterative process where, in each iteration, the position of an experimental point is altered by randomly varying its decision variables[33]. This modification is evaluated against a specific criterion that measures the distribution of points. If the new configuration improves this distribution, the change is accepted; otherwise, it's rejected, and the experimental point reverts to its original position. This cycle repeats until the distribution of all points meets a predetermined threshold of desired dispersion. Consequently, this method ensures that the experimental points are optimally distributed across the design space, allowing for a comprehensive understanding of the process's responses to all variations in the decision variables. While maximum designs have been widely applied in experimental design, their use as an initialization method in metaheuristic algorithms remains almost unexplored.

Expensive optimization[34,35] is a field widely observed in engineering practice, characterized by the use of computationally intensive execution of computational models to calculate the quality of candidate solutions. Owing to significant time requirements, only a limited number of function evaluations can typically be utilized for expensive optimization. This limitation significantly restricts the application of many optimization algorithms, including popular evolutionary and metaheuristic algorithms, which generally require a large number of function evaluations to achieve satisfactory solutions[36]. Consequently, there is a need for more efficient optimization methods that can deliver high-quality results with fewer evaluations, making them suitable for expensive optimization scenarios.

This paper introduces a novel metaheuristic optimization algorithm that utilizes the capabilities of a radial basis function neural network (RBFNN) to guide the search process. The algorithm starts by employing the maximum design approach to initialize a set of solutions that are strategically positioned to cover the entire search space comprehensively. Subsequently, the algorithm enters a cyclical phase. In each cycle, the RBFNN is trained to model the objective function values based on the current set of solutions. The algorithm identifies the neurons in the hidden layer corresponding to the highest values of the objective function and utilizes the parameters of these crucial neurons, such as their centroids and standard deviations, to generate new solutions by sampling in the order of importance until the typical number of solutions is reached. As the algorithm focuses on the promising regions of the search space that generate a high value of the objective function, it avoids the exhaustive evaluation of solutions and enables a more informed exploration of the search space. The proposed method's efficiency was assessed by comparing it with several popular metaheuristic algorithms on a set of 30 test functions. The results showed that the new approach outperformed the existing techniques, achieving superior-quality solutions and better convergence rates.

The rest of this research paper is divided into several sections. Section "Radial basis functions neural network (RBFNN)" delves into the attributes and features of the Radial Basis Function Neural Network (RBFNN). Section "Maximum designs" covers the Maximum Designs Method and its attributes in exploring the search space. Section "Proposed algorithm" outlines the suggested methodology. Section "Experimental results" presents the experimental outcomes. Section "The proposed RBFNN-A method in expensive optimization" conduct a complexity time analysis from the proposed method. Lastly, Section "Conclusions" the conclusions are presented.

Radial basis functions neural network (RBFNN)

Structure and operation of a RBFNN

A Radial Basis Function Neural Network (RBFNN)[29] is a specific type of artificial neural network that stands out due to its distinct architecture. It has three layers: an input layer, a hidden layer with radial basis functions, and an output layer (as illustrated in Fig. 1). The input layer represents the input variables of the network, with each node corresponding to a specific feature or dimension of the input data. The hidden layer of an RBFNN is where the network's unique characteristics lie, comprising radial basis functions (RBFs) associated with individual neurons. Each neuron is defined by a center point and a radial basis function that quantifies its influence based on the distance from the center. The RBFs exhibit local receptive regions, meaning they are sensitive primarily to data points located near their respective centers. This characteristic enables the network to capture particular patterns or features in the input data. The output layer typically consists of a linear combination of the hidden layer outputs, transforming the information extracted by the RBFs into the network's final output.

In Fig. 1, a schematic representation of a generic Radial Basis Function Neural Network (RBFNN) is depicted. As with any neural network, the objective is to model the mapping $\mathbf{Y} = f(\mathbf{X})$ by considering a set $\mathbf{D}$ of training data $(\mathbf{X}(n), \mathbf{Y}(n))$, where n is one of the elements of $\mathbf{D}$. The network comprises of p neurons in the input layer, m neurons in the hidden layer, and an output layer with r neurons. Thus, the output k, denoted as $y_k(n)$, which is generated based on the input pattern $\mathbf{X}(n) = \{x_1(n), \ldots, x_p(n)\}$, can be expressed and modeled as follows:

$$y_k(n) = \sum_{i=1}^{m} w_{ik}\phi_i(n), \text{where } k = 1,2,\ldots,r \tag{1}$$

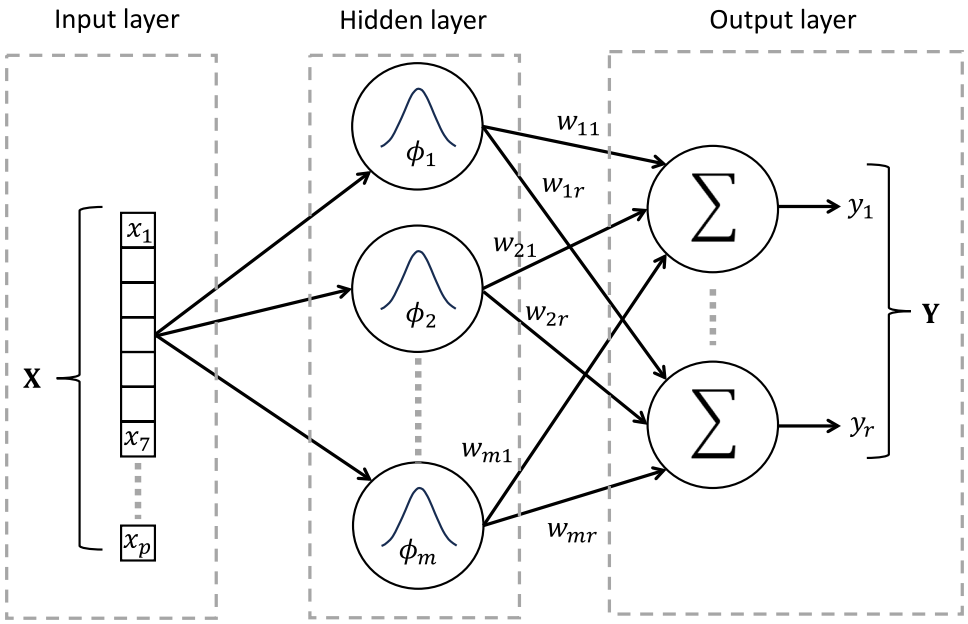

Figure 1. Architecture of a radial-based network.

In Eq. (1), the term w_{ik} signifies the weight attached to the hidden neuron i with respect to the output neuron k, while $\phi_i(n)$ denotes the activation levels of the hidden neurons ($i = 1, \ldots, m$) that correspond to the input $\mathbf{X}(n)$. The activation value of each hidden neuron i, denoted as $\phi_i(n)$, is determined by the following equation:

$$\phi_i(n) = \phi\left(\frac{\| \mathbf{X}(n) - C_i \|}{\sigma_i}\right) \tag{2}$$

In Eq. (2), $\phi(\cdot)$ is referred to as the radial basis function, while $C_i = \left(c_{i1}, \ldots, c_{ip}\right)$ represents the centers of the radial basis function. Additionally, $\sigma_i = \left(\sigma_{i1}, \ldots, \sigma_{ip}\right)$ denotes the deviation or width, and $\|\cdot\|$ represents the Euclidean distance between the input data $\mathbf{X}(n)$ and the center C_i.

The activation value of a neuron in the hidden layer is determined by the proximity of the input data $\mathbf{X}(n)$ to the center C_i. In general, a radial basis function $\phi(\cdot)$ attains a value close to its maximum when the input $\mathbf{X}(n)$ is in close proximity to the center C_i of the neuron; conversely, it assumes a minimum value when $\mathbf{X}(n)$ is far from the center. There are several functions that exhibit these characteristics. However, one of the most commonly used is the Gaussian function. Under these conditions, the activation value of a neuron i in the hidden layer can be calculated as follows:

$$\phi_i(n) = \exp^{\left(-\frac{\|\mathbf{X}(n) - C_i\|^2}{2\sigma_i^2}\right)} \tag{3}$$

A key attribute of the radial basis function neural network (RBFNN) is its inherently interpretable structure, especially evident in the hidden layer. During the training process, the RBFNN adjusts the center points and shapes of the radial basis functions, allowing it to adeptly approximate complex nonlinear relationships present in the data. This adjustability is crucial, as it enables the network to effectively model intricate patterns and dependencies between variables. The transparency of this process is a significant advantage; it allows for a clear understanding of how the network processes information. By observing the parameters in the radial basis functions and their influence on the network's output, it becomes possible to determine the specific associations and dependencies between the input and output data. This level of interpretability is particularly valuable in applications where understanding the underlying decision-making process of the model is as important as the accuracy of its predictions.

Training of a RBFNN

The training process of a Radial Basis Function Neural Network (RBFNN) consists of two stages[37] that facilitate the network's ability to approximate complex functions with precision. Initially, an unsupervised phase is employed wherein the centers C_i and widths d_i of the radial basis functions (RBFs) in the hidden layer are initialized using the k-means clustering method ($i = 1, \ldots, m$).

The supervised learning phase is the next step in the network's training process. During this phase, input data points are provided to the network, and the radial basis functions (RBFs) compute their output values. These outputs are then compared to the target values from the training dataset, and a common optimization technique, such as gradient descent, is employed to adjust the weights connecting the hidden layer ($i = 1, \ldots, m$) and output layer ($k = 1, \ldots, r$) neurons so that the error between the computed and target outputs is minimized.

The process of training the network is repeated iteratively on the training dataset to ensure that it converges to a satisfactory approximation of the target function. Properly trained, the radial basis functions of the RBFNN are positioned and shaped to capture the essential patterns and relationships within the data, thereby enabling the network to generalize well to unseen data points. Once the RBFNN has been trained, it can be applied to perform a variety of tasks, including function approximation, pattern recognition, and regression, with a high degree of accuracy and transparency. The key advantage of the RBFNN is its ability to approximate complex nonlinear relationships between inputs and outputs, making it a powerful tool for data analysis and modeling.

Maximum designs

Maximum designs[32,33] represent a class of experimental design techniques dedicated to achieving extensive coverage throughout the entire range of potential experimental conditions or factors. The central aim of these techniques is to distribute experimental points as widely as possible across the design space. This broad dispersion is crucial for enabling a comprehensive exploration of the experiment's landscape, ensuring that all potential interactions between decision variables are thoroughly investigated. By maximizing the spread of these points, maximum designs ensure that no area of the design space is overlooked, allowing for a complete and detailed understanding of how different factors interact and influence the outcomes. This approach is particularly important in experiments where capturing the full complexity of the system or process under study is essential for drawing accurate and reliable conclusions.

This section is divided into two parts. The first part explains the index used by the maximum designs method to measure data dispersion. The second part analyzes the maximum designs method in detail.

Evaluation of dispersion

To assess how well a set of N elements $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is distributed within its space, the determinant of a covariance function Co matrix serves as a critical index. A covariance function captures the variance between pairs of variables indicating their level of differentiation. The covariance matrix C is an $N \times N$ matrix, encapsulating the relationships between all N elements within the set. Each element $co_{i,j}$ within matrix Co is defined as follows:

$$co_{i,j} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{4}$$

The relation between $\mathbf{x}_i$ and $\mathbf{x}_j$ is represented by the value $co_{i,j}$. When $\mathbf{x}_i$ and $\mathbf{x}_j$ are close, the value of $co_{i,j}$ is nearly one, but it decreases to zero as they move further apart. Once the matrix Co is calculated, the value of the determinant $|Co|$ represents the degree of dispersion of $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. A high value of $|Co|$ is high indicates a higher dispersion of the data.

The method of maximum designs

In Maximum designs, the central objective is to adjust the positions of a set of N points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ within a d -dimensional space $\left(\mathbf{x}_i = \left\{x_{i,1}, \ldots, x_{i,d}\right\}, i \in 1, \ldots, N\right)$ to maximize the coverage of the design space $\mathbf{S}$, defined by the range of minimum (min_j) and maximum (max_j) values of each decision variable j where $j \in 1, .., d$. Initially, this technique involves randomly positioning the set of N points within the search space $\mathbf{S}$. Following this, an iterative process is employed. In each iteration t, the position of one of the points $\mathbf{x}_i$ is altered by randomly changing its decision variables producing the new position $\mathbf{x}_i^n$. After implementing this change, a crucial evaluation step follows. This evaluation determines whether the dispersion of the new configuration $\left|Co(\mathbf{x}_i^n)\right|$ (that includes the modified point $\mathbf{x}_i^n$) is higher than the original dispersion $|Co(\mathbf{x}_i)|$. If the new configuration improves this distribution $\left(\left|Co(\mathbf{x}_i^n)\right| > |Co(\mathbf{x}_i)|\right)$, the change $\mathbf{x}_i^n$ is accepted; otherwise, it's rejected, and the point reverts to its original position $\mathbf{x}_i$. This cycle repeats until the distribution of all points meets a predetermined threshold of desired dispersion. This iterative cycle is continued until one of two criteria is met: either a predefined number of iterations *iterini* has been completed or the set of N points has achieved a specified level of dispersion $|Co|$ in its last configuration.

Figure 2 provides a clear example of how the method of maximum designs operates in practice. Figure 2a, we observe an initial configuration of 50 points in two dimensions x_1 and x_2 considering a space defined within [0,1]. These points are deliberately clustered around the center of the space, a setup designed to highlight the impact of the method. In Fig. 2b, the outcome after 100 iterations is depicted. Here, the final arrangement of points is showcased, demonstrating how the method achieves comprehensive and efficient coverage of the space. This transformation from a centralized cluster to a well-distributed set of points across the entire space underscores the effectiveness of the maximum designs method in optimizing the spread of points for better exploration and analysis of the given area.

Proposed algorithm

This study introduces a novel metaheuristic optimization algorithm that incorporates the information derived from a radial basis function neural network (RBFNN) as a means of informing the search process. The aim of the proposed method is to identify the optimal solution for a nonlinear objective function f, which can be described as follows:

$$\begin{aligned} &\text{Maximize} f(\mathbf{x}) = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \\ &\text{Subject to } \mathbf{x} \in \mathbf{S} \end{aligned} \tag{5}$$

where the function is referred as $f : \mathbb{R}^d \to \mathbb{R}$, within a search space represented by $\mathbf{S}$ $\left(\mathbf{x} \in \mathbb{R}^d | min_i \le x_i \le max_i, i = 1, \ldots, d\right)$, which is bounded by lower (min_i) and upper (max_i) limits for each dimension. Our proposed approach consists of six essential components: (4.1) initialization, (4.2) training of the Radial Basis Function Neural Network (RBFNN) and (4.3) generation of solutions. Each of these elements plays a critical role in the effectiveness of our optimization strategy. In the subsequent subsections, we will provide a

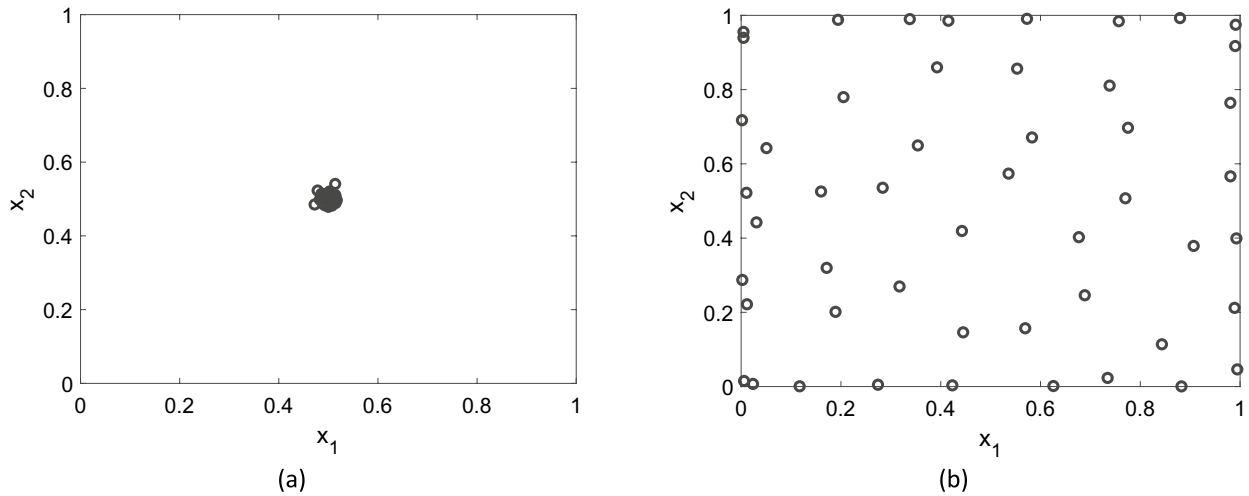

Figure 2. Example of how the method of maximum designs operates in practice. (**a**) Initial configuration of 50 points in two dimensions x_1 and x_2. (**b**) The outcome after 100 iterations.

detailed explanation of each component, highlighting their significance and how they collectively contribute to the success of our approach. At the end of this section, in subsection "Computational procedure", a summary of the computational procedure of the complete method is given.

Initialization

Our methodology incorporates the information obtained from the neural network's training to develop the search strategy. For this reason, it is essential to have an initial comprehensive and representative sample set that encompasses the entire search space. This process is achieved through the utilization of the maximum design method (as outlined in Section "Maximum designs"). The initialization stage represents the initial operation of the algorithm, with the aim of establishing an initial population consisting of N elements ($P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$). Each solution $\mathbf{x}_i$ in the population represents a specific combination of decision variables $\{x, ..., x_{i,d}\}$, which assumes an initial value for the optimization process. In our approach, the initial positions of each element $\mathbf{x}_i$ are determined by the maximum design method. The method has been configured to consider 1000 iterations ($iterini = 1000$), which guarantees that all the positions of the candidate solutions thoroughly explore the search space $\mathbf{S}$. It is important to note that the computational cost of initialization is performed only once in the optimization process.

Training the RBFNN

Given a set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N points sourced from the input space $\mathbf{S}$—either as initial configuration or as results from prior iterations—the next step involves evaluating these points against the objective function f. This evaluation yields N training pairs, each linking a position $\mathbf{x}_i$ to its respective outcome $f(\mathbf{x}_i)$ produced by the objective function. This collection of pairs forms the dataset on which the neural network is trained, aiming to closely approximate the objective function f. Through this training, the network learns to correlate input positions with their corresponding function values. This capability allows the network to grasp the intricate patterns and relationships present in the data. Consequently, this understanding significantly improves the network's prediction accuracy, thereby enhancing its role in steering the optimization process towards more effective solutions.

In the training phase, the neural network is structured to include d neurons in the input layer, correlating to each decision variable. The hidden layer contains M units. This layer is important for understanding the complex relationships within the data. A singular output neuron is assigned for generating the network's predictions of the objective function f. The training follows the procedures detailed in Section "Radial basis functions neural network (RBFNN)", employing a structured methodology to ready the network for its optimization role. This arrangement guarantees that the neural network is in harmony with the problem's specific attributes, facilitating its capacity to accurately learn and predict the objective function. Such an alignment is essential, as it equips the network to significantly influence the optimization process, steering it toward favorable results.

The performance of a Radial Basis Function Neural Network (RBFNN) is significantly influenced by the number of neurons, M, in its hidden layer. Having a higher number of neurons enhances the network's ability to precisely approximate complex functions, capturing intricate details. However, this advantage comes with the risk of overfitting to the training data, which can undermine the network's ability to perform well on unseen data. On the other hand, a smaller number of neurons simplifies the model and reduces computational demands but may compromise the quality of function approximation due to insufficient complexity. Determining the optimal number of neurons in the hidden layer is a delicate balance that depends on the scope of the search space. Achieving this balance is essential for maximizing RBFNN performance, ensuring the network can accurately model the data patterns without unnecessary complexity. Our methodology for selecting the appropriated number of neurons in the hidden layer follows the model considered in Eq. (6), tailored to align with these considerations.

$$M = \frac{10 \cdot d}{N} \tag{6}$$

In our Radial Basis Function Neural Network (RBFNN) model, the selection of the number of hidden neurons M, is influenced by both the dimensions d of the data and the quantity of training data available N. The complexity inherent in the data, which guides the determination of the optimal number of hidden neurons, is directly affected by these two factors. As the dimensionality of the data increases, a correspondingly larger number of hidden neurons may be required to effectively capture and replicate the patterns present. High dimensional data, in particular, demands a more substantial neuron count in the hidden layer to ensure the network can accurately learn and represent the underlying relationships. Utilizing the approach defined in Eq. (6), we calibrate the number of neurons in the hidden layer to achieve a balance. This balance is crucial for providing a robust approximation of the objective function, ensuring generalization while avoiding the problem of overfitting. This methodology ensures that our RBFNN model can adapt to the complexity and scale of the data it is trained on, offering an optimal level of performance.

After finalizing the network architecture, the next step is to carry out the training process as outlined in Section "Radial basis functions neural network (RBFNN)". Conducted over 100 iterations, this phase arms the neural network with the ability to closely represent the objective function. With this information, the neural network can be used to explore the search space. Upon the completion of the training, the network finds the network parameters to establish significant associations between the input space data points and their related outputs. The result of this training introduces two crucial aspects into the network's framework: A) an in-depth insight into certain areas within the input space that impact the output significantly, and B) a clear evaluation of how these areas contribute to the outcome. These integral components enable the network to uncover the complex dynamics between variables, offering a refined analysis of the data. This enhancement paves the way for devising precise and powerful optimization strategies, significantly boosting the network's functionality.

The influence of regions of the input space on particular outputs is defined by the outputs of the radial basis functions ϕ_i, where each function is associated with a neuron i in the hidden layer. These functions act as specialized filters, with each one exhibiting distinct sensitivity to certain areas of the input space. When a radial basis function's output nears one for a specific input, this indicates that the input lies within a region to which the function is highly responsive. On the other hand, a function output close to zero suggests that the input falls outside the function's sensitive region. This capability of the network to discriminate and evaluate the relevance of various parts of the input space is crucial. It enables the network to deeply grasp the intricate dynamics present in the data. As a result, the network can enhance the precision of its predictions and the efficacy of its optimization approaches, leveraging this nuanced understanding to navigate the complexities of the input space more effectively.

The weights connecting the neurons in the hidden layer to the output neuron also play a pivotal role in highlighting the importance of specific regions within the input space for determining the final output. A high and positive weight, denoted by w_i, indicates that the region of the input space associated with the radial basis function ϕ_i significantly influences the generation of high values or peaks in the network's output. Conversely, a high but negative weight w_i suggests that the region delineated by ϕ_i is instrumental in producing low values or valleys in the output. These weightings are crucial for the network's ability to prioritize different areas within the input space, effectively mapping the complex interplay between inputs and outputs. By understanding these relationships, the network enhances its ability to deliver accurate predictions and make informed optimization decisions, leveraging the nuanced insights provided by the weight assignments to navigate the optimization landscape more effectively.

Figure 3 demonstrates how the neural network's structure, post-training, captures essential details about the function it seeks to approximate. In the depicted scenario, the network aims to model a function f, utilizing a dataset composed of points $(\mathbf{x}, J(\mathbf{x}))$, where $\mathbf{x}$ encompasses two decision variables, x_1 and x_2. The figure highlights the pivotal role of the centroids of the hidden layer's neurons ($M = 3$) in determining the locations of the function's peaks and valleys after approximation. Notably, the most significant peak is at position C_3, coinciding with the highest weight $w_3 = 2$ of a positive value, whereas the valley is at position C_3, linked to the sole negative weight $w_3 = -1$. The standard deviations, σ_1, σ_2, and σ_3, are crucial for understanding how the function f is distributed around key positions C_1, C_2, and C_3 within the network. Remarkably, these standard deviations are uniform across both dimensions, x_1 and x_2, indicating a consistent spread. The larger values of σ_1 and σ_2 ($\sigma_1 = \sigma_2 = 0.8$) suggest a broader, more gentle distribution, whereas the smaller value of σ_3 ($\sigma_3 = 0.5$) indicates a sharper peak around C_3. This variance in dispersion values underlines the network's ability to differentiate and detail the function's local behaviors and characteristics, offering profound insights into the objective function's nature. Such detailed understanding is crucial for refining optimization strategies and making informed decisions.

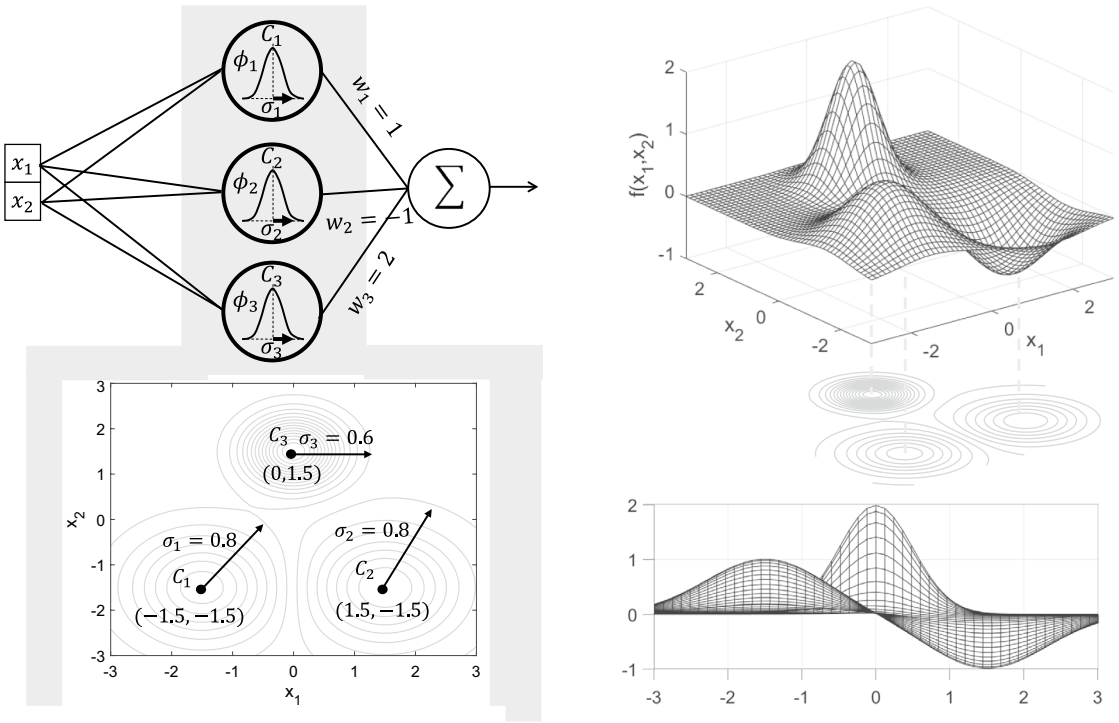

Figure 3. Post-training parameters of the neural network. It captures the essential elements about the function it seeks to approximate.

Production of the new solutions

In our methodology, at each iteration k, we create N new solutions by extracting information from the preceding N solutions used to train the neural network. To generate the N new solutions, we first identify the b most significant solutions from the current population. The knee point technique[38,39] is employed to determine the number and identity of the most significant solutions. From these b solutions, the N new solutions are generated. To perform this process, one of the b solutions is randomly selected. Assuming that solution i has been chosen, the next step is to determine which hidden layer neural network this solution belongs to. The neural network responsible for this solution will be the one that yields a higher value, as it is more closely related to its centroid. Assuming that solution i belongs to the region corresponding to neuron j, the new solution is generated by sampling a Gaussian function with the parameters of the radial basis function corresponding to neuron j. This process is repeated until N new solutions are generated. All these operations will be described in the following paragraphs.

Identification of the most significant solutions

The knee point[38,39] is used in this work to identify the number b of significant solutions among a set N of solutions. In system engineering, the knee point refers to a critical position in a performance curve or trade-off analysis in which the value of the elements changes significantly. This point is used to identify the optimal balance between different factors, such as value versus significance, in the design and development of systems. Beyond the knee point, the values of the elements strongly diminished until they were categorized as insignificant compared with first elements. Identifying this point helps decision makers select the most effective and efficient option among various alternatives.

To identify the knee point among the N solutions, we begin by organizing these solutions in descending order based on their performance against the objective function. In this arrangement, solution $\mathbf{x}_1$ is the one yielding the highest value $f(\mathbf{x}_1)$, indicating optimal performance, while solution $\mathbf{x}_N$ is at the other end, offering the least desirable outcome $f(\mathbf{x}_N)$. Following this organization, we construct two sequences. The first sequence, labeled s_1, incorporates the objective function values generated by each solution $\left(s_1 = \left\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)\right\}\right)$. The second sequence, named s_2, consists of the indices corresponding to each solution's position in the ordered list $(s_2 = \{1, \ldots, N\})$. The subsequent phase involves normalizing these sequences to ensure their values range from 0 to 1. This normalization is achieved by dividing the sequences by their respective maximum values, resulting in s_1 being scaled by $f(\mathbf{x}_1)$, the highest objective function value, and s_2 being divided by N, the total number of solutions.

Taking into account the values of s_1 and s_2, we can define the objective function J that connects the data from s_2 in terms of the significance of s_1. The structure of J is modeled as follows:

$$J = s_1 + s_2 \tag{6}$$

This objective function J, consisting of values $J_1, \ldots, J_N$, has a size of N, and one of its key features is that it possesses only one global minimum value. This minimum value signifies the knee point KP, which is determined as $KP = \min(J)$. Figure 4 shows the estimation of KP. In the example, a set of eleven points $(N = 11)$ that associates the normalized variables s_1 and s_2 is assumed. In Fig. 4, the point KP divides the graph into two parts, the first 4 points represent the most significant solutions $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ $(b = 4)$, while the other 7 $\{\mathbf{x}_5, \ldots, \mathbf{x}_{11}\}$ present the insignificant solutions. In our approach the 4 solutions will be used as the significant ones to produce the 11 new solutions.

Generation of solutions

The generation of a new set of N solutions begins by identifying the b most significant solutions. These b solutions $\{\mathbf{x}_1, \ldots, \mathbf{x}_b\}$ are then utilized as the basis for the subsequent set of N new solutions. The generation process begins with the random selection of one solution from the identified b significant solutions. If the solution i is

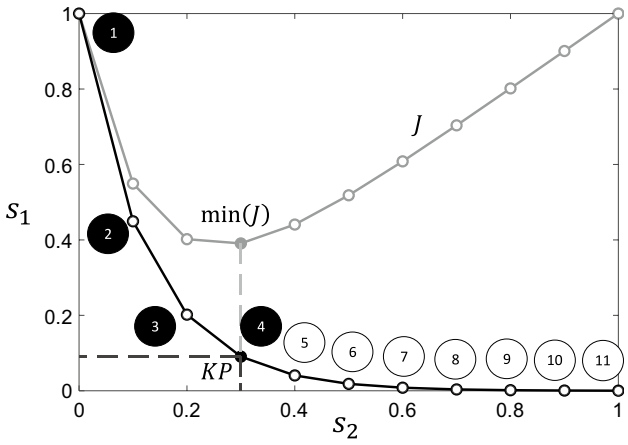

Figure 4. Estimation of KP considering a set of eleven points $(N = 11)$.

selected. The next step is to determine the specific hidden layer neural network to which this solution is most closely associated. This is determined by identifying the neuron that produces the highest output value for solution i, indicating a closer proximity to its corresponding centroid. If it is determined that solution i falls within the domain influenced by neuron j, then the new solution is created considering the parameters associates to this neuron. Under these conditions, the new solution $\mathbf{x}_{new}$ is produced by sampling from a normal distribution considering the following model:

$$\mathbf{x}_{new} = N(C_j, \sigma_j) \tag{7}$$

where C_j and σ_j ($\sigma_j = (\sigma_{j1}, \ldots, \sigma_{jd})$) represent the centroid and standard deviation of the radial basis function associated with neuron j. The process of generating a new solution is repeated until N new solutions are generated.

Figure 5 presents a visual illustration of the sampling process, showing the specific regions within the input space that each neuron in the hidden layer is responsive to. These designated areas are targeted during the sampling phase, utilizing the parameters that correspond to the radial basis functions associated with each neuron. This graphical representation effectively highlights how the neural network's structure facilitates the identification and sampling of distinct areas, based on the neurons' sensitivity to particular segments of the input space. By employing the parameters of the radial basis functions, the process ensures that samples are drawn from regions most relevant to the neurons' activation patterns, offering insight into the network's method of approximating and exploring the input space through its hidden layer dynamics.

Computational procedure

The computational methodology of our approach is encapsulated in Algorithm 1, presented as pseudocode. Initially, the algorithm involves setting up input parameters, including the population size N, the number of initialization iterations *iterini*, and the number of evolution iterations *itermax*. Following this setup, the population is initialized within the search space boundaries using the maximum designs method (line 2). The algorithm then enters a cycle (line 3) that repeats until the predefined number of iterations has been met (*itermax*). The cycle commences with training the radial basis function network with the current N elements (line 4). Subsequent to this training phase, the algorithm identifies the b most significant solutions out of the N, utilizing the knee-point method for selection (line 5). The generation of the next set of N solutions is based on these b significant solutions (line 6), necessitating the alignment of each significant solution with the neuron in the hidden layer to which it is most closely related. Leveraging this alignment, new solutions are crafted by sampling a normal distribution with parameters derived from the associated neuron. This iterative process culminates in the generation of N new solutions. Upon completion of the cycle, the algorithm deems the best-performing solution as the answer to the optimization challenge, effectively iterating through initialization, training, selection, and generation phases to arrive at an optimal solution.

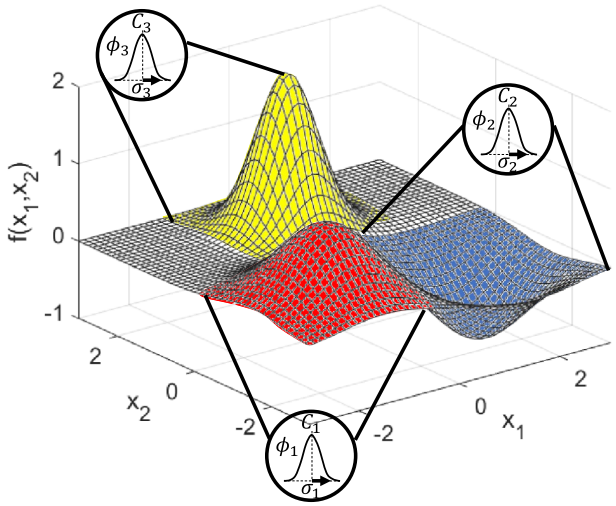

Figure 5. Illustration of the particular areas within the input space that each neuron in the hidden layer responds to during the sampling process.

Input: $N, iterini, M, maxiter, itermax, k = 0$
$\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \leftarrow$ MaximumDesigns $(N, [min_i, max_i], iterini)$
while $k \leq itermax$
  $\{\phi_1, \ldots, \phi_M\} \leftarrow$TrainingRBFNN$(\{\mathbf{x}_1, \ldots, \mathbf{x}_N\})$
  $\{\mathbf{x}_1, \ldots, \mathbf{x}_b\} \leftarrow$MostSignificantSolutions$(\{\mathbf{x}_1, \ldots, \mathbf{x}_N\})$
  $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \leftarrow$ GenerationOfSolutions$(\{\mathbf{x}_1, \ldots, \mathbf{x}_b\})$
end while
Output: $\mathbf{x}_{BEST}$

Algorithm 1. Pseudocode of the proposed method.

Experimental results

This section details a suite of experiments designed to evaluate the efficacy of our approach, herein referred to as RBFNN-A (RBFNN approach). For a comprehensive evaluation of our algorithm's performance, several elements were considered. First, a set of 29 functions traditionally used to evaluate the accuracy and efficiency of metaheuristic methods was included. These functions were chosen to maintain compatibility with other previous studies and are listed in Table A of the Appendix. Second, the CEC 2022 test function set, which includes 12 functions with varying levels of complexity, was utilized. This set allows for the consideration of composite and hybrid functions, providing a robust test for the algorithm's capabilities. These functions are detailed in Table B of the Appendix. Finally, a set of engineering problems was used in the comparisons to demonstrate the method's effectiveness on practical problems commonly encountered in practice. A description of these engineering problems is provided in Table C of the Appendix. This multi-faceted approach ensures a thorough and relevant assessment of our algorithm.

To facilitate a comprehensive evaluation of our proposed approach's effectiveness, we've incorporated a range of well-established metaheuristic algorithms into our analysis. These selected algorithms span from traditional to more contemporary optimization techniques, thereby covering a wide spectrum of methodologies. The comparative analysis includes several notable algorithms such as the Artificial Bee Colony (ABC)[16], Crow Search Algorithm (CSA)[40], Differential Evolution (DE)[41], Estimation of Distribution Algorithm (EDA)[42], Moth-Flame Optimization (MFO)[43], Harmony Search (HS)[44], Simulated Annealing (SA)[45], and State of Matter Search (SMS)[46]. To ensure a fair and consistent comparison across all methods, we have adhered to the parameter settings recommended by the original creators of each algorithm, as cited in their respective publications. By aligning each algorithm's configuration with the suggestions of its authors, we aim to present each method in its best light, ensuring that the performance metrics reflect the optimal capabilities as intended by their developers. This approach underscores the integrity of the comparison, offering a clear and unbiased assessment of how our proposed method stacks up against established benchmarks in the field.

This section is divided into four sections, each of which is carefully structured. The first section "Assessment of performance in relation to its parameters" deals with the evaluation of the performance of the proposed method in terms of its own parameters. The second section, section "Performance comparison considering the 29 functions from table A", begins with a comprehensive performance analysis that compares the proposed method to established metaheuristic algorithms considering the 29 functions of Table A. This comparison is intended to highlight the unique strengths and efficiency of the proposed approach. In subsection "Engineering design problems", the performance of all metaheuristic methods is analyzed through the set of function of CEC 2022 from Table B. Finally, in Section "Convergence analysis", we analyze the performance of metaheuristic algorithms when they attempt to solve the engineering problems described in Table C of the Appendix.

Assessment of performance in relation to its parameters

The two parameters, the number of elements N and the number of hidden neural neurons M, significantly influence the expected performance of the proposed optimization algorithm. In this sub-section, we analyze how the algorithm behaves under different settings of these parameters. To keep the analysis straightforward, we consider only a subset of functions such as f_2, f_5, f_{13}, f_{20} and f_{25}, which include both unimodal and multimodal functions. The description of these function is shown in Table A of the Appendix. During simulations, all functions are evaluated with a dimension $n = 50$. Initially, the parameters N and M are set to their default values, with $N = 50$ and $M = 10$. Our analysis then examines the impact of each parameter independently, keeping the other parameter fixed at its default value. To reduce the stochastic effects inherent to the algorithm, each benchmark function is executed independently 10 times. The termination criterion for the optimization process is set to a maximum of 1000 iterations ($itermax = 1000$). This systematic approach ensures a thorough evaluation of how N and M affect the performance of the proposed algorithm.

In the first stage, the behavior of the proposed algorithm is analyzed by considering different values for N. In this analysis, the values of N vary from 10 to 100, while the value of M remains fixed at 10. The results, recorded in Table 1, are presented in terms of the Average Best Fitness values (AB). The Average Best Fitness Value (AB) represents the lowest fitness value achieved by each method, reflecting the quality of the solutions generated. Notably, lower values indicate superior performance. From Table 1, it is evident that the proposed algorithm maintains better performance on all functions when $N \geq 50$. Conversely, when $N < 50$, the performance of the algorithm is inconsistent, generally producing suboptimal results. This analysis highlights the importance of adequately setting the parameter N to ensure the algorithm's robustness and effectiveness.

f/N		10	20	30	40	50	60	70	80	90	100
f_2	AB	2.25E+04	1.72E+03	2.21E+02	3.01E+01	2.68E+01	2.41E+01	2.30E+01	2.28E+01	2.28E+01	2.27E+01
f_5	AB	2.73E+03	2.41E+02	3.36E+01	7.03E+00	4.68E+00	4.62E+00	4.61E+00	4.60E+00	4.60E+00	4.61E+00
f_{13}	AB	6.21E+02	4.32E+02	2.01E+02	8.01E+01	3.97E+01	3.95E+01	3.93E+01	3.90E+01	3.91E+01	3.92E+01
f_{20}	AB	5.22E+04	3.34E+03	1.32E+03	12.72E+02	9.81E+02	9.78E+02	9.76E+02	9.74E+02	9.75E+02	9.73E+02
f_{25}	AB	6.62E+04	2.12E+04	5.71E+03	2.89E+02	1.63E+02	1.60E+02	1.58E+02	1.52E+02	1.55E+02	1.53E+02

Table 1. Experimental outcomes achieved by the proposed algorithm with varying values of N.

In the second stage, the performance of the proposed algorithm is evaluated by considering different number of hidden neurons M. In this experiment, the values of M are varied from 2 to 20, while the value of N remains fixed at 50. The statistical results obtained by the proposed method using different values of M are presented in Table 2. From Table 2, it is clear that our optimization algorithm performs best when $M \geq 10$, consistently obtaining superior results across all functions. Conversely, when $M < 10$, the performance of the algorithm declines significantly, resulting in suboptimal outcomes.

In general, the experimental results shown in Tables 1 and 2 suggest that a proper combination of the parameter values N and M can significantly improve the performance of the proposed method and the quality of the solutions. From this experiment, we can conclude that the optimal parameter set consists of $M \geq 10M$ and $N \geq 50N$. Consequently, to avoid unnecessary computational load while maintaining high performance, the values of M and N were set to their minimum effective values, $M = 10$ and $N = 50$. This finding not only enhances the efficiency of the computational process but also validates the practical application of Eq. (6), which has been proposed in several studies in the literature for the training of radial basis neural networks. This validation demonstrates the robustness and reliability of the proposed algorithm within the tested parameter range.

Performance comparison considering the 29 functions from table A

For a broad evaluation, we utilized a diverse set of 29 benchmark functions, encompassing types such as unimodal, multimodal, and hybrid, with their mathematical representations provided in Table A of the Appendix. These functions were chosen to maintain compatibility with other previous studies. From Table A, $f(x^*)$ represents the optimal function value obtained at the position x^*, and **S** denotes the defined lower and upper bounds of the search space. This structured approach aims to offer a detailed insight into the RBFNN-A's capabilities, highlighting its strengths and areas of robustness across a variety of challenging functions.

In our comparative analysis of various algorithms, we concentrate on analyzing the fitness values produced by each method, attempting to minimize these values throughout the optimization process. The optimization efforts are regulated by a predefined stopping criterion of a maximum of 1000 iterations ($maxiter = 1000$). Additionally, the analysis scrutinizes the algorithms' performances at different dimensional scales, notably at 50 and 100 dimensions, to evaluate the scalability of our proposed approach. For consistency, each metaheuristic algorithm is configured with a population size of 50 individuals, which dictates the number of candidate solutions explored in each iteration. Considering the inherent stochastic nature of these algorithms, we conduct 30 independent runs for each benchmark function to achieve a robust evaluation. These multiple iterations are crucial for understanding the impact of randomness and variability on the performance of the algorithms, thereby enriching the statistical analysis and insights. This meticulous methodology facilitates a detailed and nuanced comparison of the algorithms' efficiency across various operational scenarios and constraints.

The outcomes of each algorithm are thoroughly detailed in Tables 2 and 3, which provide essential numerical data for evaluating the performance of each approach for 50 and 100 dimensions, respectively. These measurements include the average best fitness values AB, the median fitness values MD, and the standard deviation SD of the fitness values. The averaged Best Fitness Value (AB) signifies the lowest fitness value achieved by each method, reflecting the quality of the solutions it generates. Notably, lower values here indicate superior performance. Furthermore, the Median Fitness values (MD) is the median fitness value computed over all runs for each method, providing an overall perspective on the algorithm's performance and consistency. This average is calculated from 30 separate runs to minimize the effect of randomness inherent in these algorithms. Lastly, the Standard Deviation (SD) quantifies the variability in the fitness values across the 30 executions for each method, with a lower standard deviation indicating more consistent and dependable performance. To aid in rapid identification

f/M		2	4	6	8	10	12	14	16	18	20
f_2	AB	4.23E+06	2.87E+02	2.42E+02	3.22E+01	2.67E+01	2.40E+01	2.38E+01	2.30E+01	2.27E+01	2.28E+01
f_5	AB	3.21E+04	4.11E+02	2.22E+01	6.23E+00	4.66E+00	4.65E+00	4.63E+00	4.62E+00	4.61E+00	4.59E+00
f_{13}	AB	3.45E+03	5.32E+02	3.38E+02	7.21E+01	3.97E+01	3.96E+01	3.95E+01	3.94E+01	3.92E+01	3.91E+01
f_{20}	AB	7.03E+06	4.15E+03	3.23E+03	10.21E+02	9.80E+02	9.79E+02	9.75E+02	9.73E+02	9.74E+02	9.70E+02
f_{25}	AB	8.71E+05	3.42E+03	2.78E+03	1.89E+02	1.65E+02	1.61E+02	1.59E+02	1.55E+02	1.53E+02	1.54E+02

Table 2. Experimental outcomes achieved by the proposed algorithm with varying values of M.

f		ABC	CAB	CSA	DE	EDA	HS	MFO	SA	SMS	RBFNN-A
	AB	8.84E+00	2.00E+01	1.81E+01	7.62E+00	2.06E+01	1.53E+01	1.66E+01	2.06E+01	2.14E+01	**3.19E+00**
f_1	MD	8.80E+00	2.00E+01	1.82E+01	6.98E+00	2.06E+01	1.55E+01	1.86E+01	2.06E+01	2.14E+01	**3.20E+00**
	SD	9.70E−01	**7.23E−15**	2.87E−01	1.42E+00	1.14E−01	6.58E−01	5.05E+00	1.91E−01	4.46E−02	7.87E−02
	AB	3.52E+05	2.72E+07	8.06E+05	4.11E+04	5.85E+06	2.59E+05	1.47E+05	9.18E+06	1.63E+07	**2.66E+01**
f_2	MD	3.43E+05	2.74E+07	8.05E+05	2.80E+04	5.76E+06	2.55E+05	3.21E+02	8.66E+06	1.62E+07	**2.68E+01**
	SD	1.15E+05	2.16E+06	9.68E+04	3.85E+04	6.44E+05	4.39E+04	3.72E+05	2.20E+06	1.02E+06	**2.16E+00**
	AB	9.01E+00	3.79E+03	3.74E+02	1.85E+01	1.11E+03	1.66E+02	3.92E+01	1.49E+03	1.99E+03	**1.88E+00**
f_3	MD	9.12E+00	3.88E+03	3.88E+02	1.59E+01	1.12E+03	1.65E+02	**7.45E−02**	1.46E+03	2.00E+03	1.91E+00
	SD	1.37E+00	2.06E+02	3.45E+01	9.97E+00	6.73E+01	1.81E+01	5.13E+01	2.25E+02	7.42E+01	**7.83E−02**
	AB	1.38E+00	3.38E+01	5.09E−01	1.23E−02	6.59E+00	1.47E−01	7.72E−02	8.63E+00	2.42E+01	**6.85E−09**
f_4	MD	1.36E+00	3.44E+01	5.14E−01	6.45E−03	6.33E+00	1.50E−01	**1.93E−14**	8.70E+00	2.40E+01	7.03E−09
	SD	4.45E−01	5.09E+00	1.37E−01	2.35E−02	1.33E+00	3.06E−02	2.94E−01	2.51E+00	2.23E+00	**1.78E−09**
	AB	1.28E+02	1.27E+03	1.31E+02	6.51E+00	4.07E+02	5.06E+01	6.30E+01	5.43E+02	9.44E+02	**4.67E+00**
f_5	MD	1.30E+02	1.29E+03	1.33E+02	6.12E+00	4.11E+02	5.21E+01	5.56E+01	5.34E+02	9.30E+02	**4.68E+00**
	SD	2.42E+01	1.51E+02	1.46E+01	3.06E+00	4.60E+01	5.34E+00	2.42E+01	7.75E+01	6.78E+01	**5.92E−02**
	AB	2.21E+08	6.83E+09	1.23E+08	**2.01E+06**	1.15E+09	5.00E+07	1.71E+07	1.80E+09	3.37E+09	4.40E+07
f_6	MD	2.00E+08	6.85E+09	1.20E+08	1.71E+06	1.20E+09	5.00E+07	**1.45E+02**	1.83E+09	3.36E+09	4.41E+07
	SD	7.70E+07	6.03E+08	1.67E+07	1.22E+06	1.53E+08	8.10E+06	6.49E+07	4.25E+08	2.39E+08	**1.16E+06**
	AB	2.88E+08	1.17E+10	2.29E+08	6.38E+06	2.14E+09	8.25E+07	**1.56E+02**	3.38E+09	5.62E+09	2.71E+06
f_7	MD	2.87E+08	1.16E+10	2.22E+08	4.73E+06	2.15E+09	8.29E+07	**1.57E+02**	3.33E+09	5.55E+09	2.74E+06
	SD	1.08E+08	9.37E+08	4.62E+07	8.17E+06	2.65E+08	1.77E+07	**7.81E+00**	6.50E+08	3.64E+08	1.25E+05
	AB	**3.10E+01**	2.30E+02	8.55E+01	3.43E+01	1.34E+02	5.10E+01	4.13E+01	1.46E+02	1.83E+02	4.43E+01
f_8	MD	**3.10E+01**	2.30E+02	8.60E+01	3.40E+01	1.35E+02	5.10E+01	4.05E+01	1.47E+02	1.83E+02	4.50E+01
	SD	6.43E−01	**0.00E+00**	2.43E+00	2.43E+00	4.74E+00	1.95E+00	8.04E+00	1.07E+01	3.13E+00	1.20E+00
	AB	6.95E+03	2.28E+04	5.66E+03	2.97E+02	3.42E+04	2.98E+03	3.91E+03	6.97E+03	2.08E+05	**2.78E+02**
f_9	MD	6.72E+03	2.26E+04	5.48E+03	**2.33E+02**	3.35E+04	2.98E+03	3.56E+03	5.19E+03	2.06E+05	2.76E+02
	SD	1.81E+03	1.46E+03	1.01E+03	2.74E+02	6.99E+03	4.48E+02	2.88E+03	6.24E+03	2.23E+04	**8.43E+00**
	AB	4.37E+10	2.28E+12	5.73E+10	2.20E+09	3.73E+11	2.20E+10	2.08E+09	5.84E+11	9.35E+11	**2.11E+05**
f_{10}	MD	3.86E+10	2.30E+12	5.81E+10	2.20E+09	3.78E+11	2.17E+10	**3.89E+00**	5.71E+11	9.19E+11	2.23E+05
	SD	1.66E+10	9.85E+10	1.02E+10	1.43E+09	3.23E+10	2.90E+09	1.14E+10	1.02E+11	6.79E+10	**4.59E+04**
	AB	5.38E+01	1.79E+03	7.66E+01	2.00E+01	3.82E+02	4.21E+01	3.38E+01	6.16E+02	1.06E+03	**1.90E+01**
f_{11}	MD	5.22E+01	1.69E+03	7.65E+01	1.97E+01	3.84E+02	4.26E+01	2.83E+01	6.16E+02	1.06E+03	**1.90E+01**
	SD	1.01E+01	2.19E+02	8.98E+00	1.79E+00	5.62E+01	3.20E+00	1.80E+01	1.52E+02	6.92E+01	**5.86E−01**
	AB	8.63E+04	2.18E+06	4.82E+04	1.17E+03	4.84E+05	1.63E+04	1.93E+04	7.35E+05	1.44E+06	**1.58E+02**
f_{12}	MD	8.94E+04	2.16E+06	4.84E+04	1.03E+03	4.97E+05	1.61E+04	**7.14E−03**	7.48E+05	1.44E+06	1.59E+02
	SD	2.84E+04	2.78E+05	1.01E+04	8.48E+02	5.53E+04	3.03E+03	4.53E+04	1.48E+05	1.27E+05	**6.34E+00**
	AB	5.05E+02	1.40E+03	4.88E+02	**3.95E+01**	7.73E+02	2.49E+02	2.80E+02	7.04E+02	1.10E+03	4.88E+01
f_{13}	MD	5.09E+02	1.45E+03	4.89E+02	**3.74E+01**	7.70E+02	2.52E+02	2.72E+02	7.21E+02	1.10E+03	4.95E+01
	SD	2.31E+01	6.57E+01	1.96E+01	7.91E+00	2.88E+01	1.48E+01	6.52E+01	8.34E+01	3.62E+01	**3.96E+00**
	AB	1.55E+05	4.39E+06	7.05E+05	**2.12E+04**	3.37E+06	1.24E+05	2.79E+05	1.13E+06	9.84E+06	3.45E+05
f_{14}	MD	1.45E+05	4.41E+06	6.78E+05	**1.83E+04**	3.40E+06	1.30E+05	2.84E+05	1.12E+06	9.79E+06	3.45E+05
	SD	5.23E+04	1.06E+05	1.29E+05	1.44E+04	4.51E+05	1.98E+04	1.85E+05	3.12E+05	6.23E+05	**5.02E+03**
	AB	8.53E+01	8.92E+01	6.02E+01	4.35E+01	9.17E+01	5.67E+01	7.85E+01	9.55E+01	1.00E+02	**3.03E+00**
f_{15}	MD	8.53E+01	8.92E+01	6.10E+01	4.33E+01	9.22E+01	5.71E+01	8.00E+01	9.60E+01	1.00E+02	**3.02E+00**
	SD	3.40E+00	1.02E+00	2.65E+00	5.97E+00	2.11E+00	1.71E+00	5.79E+00	2.16E+00	**0.00E+00**	1.45E−01
	AB	5.81E+54	9.49E+93	4.48E+53	1.15E+02	2.75E+73	3.68E+04	8.47E+02	2.52E+03	1.67E+86	**5.47E+01**
f_{16}	MD	1.40E+51	4.16E+93	5.31E+52	1.07E+02	1.54E+71	7.63E+02	8.00E+02	2.53E+03	2.10E+85	**5.51E+01**
	SD	2.87E+55	1.43E+94	8.60E+53	6.94E+01	1.13E+74	1.08E+05	3.25E+02	1.90E+02	3.55E+86	**3.22E+00**
	AB	8.71E+02	3.97E+05	4.17E+04	3.03E+03	1.19E+05	1.86E+04	5.67E+03	1.59E+05	2.21E+05	**1.08E+02**
f_{17}	MD	8.74E+02	3.96E+05	4.18E+04	2.62E+03	1.18E+05	1.85E+04	5.00E+03	1.58E+05	2.21E+05	**1.10E+02**
	SD	1.66E+02	1.33E+04	3.07E+03	1.66E+03	7.85E+03	1.63E+03	6.79E+03	1.37E+04	1.01E+04	**8.70E+00**
	AB	−9.00E+02	9.43E+02	−1.11E+03	**−1.81E+03**	−6.78E+02	−1.67E+03	−1.70E+03	−3.48E+02	5.69E+02	−1.07E+01
f_{18}	MD	−8.91E+02	9.87E+02	−1.10E+03	**−1.81E+03**	−6.70E+02	−1.67E+03	−1.70E+03	−3.75E+02	5.43E+02	−1.04E+01
	SD	7.15E+01	4.53E+02	3.85E+01	3.16E+01	8.17E+01	2.68E+01	5.23E+01	1.79E+02	1.58E+02	**1.01E+00**
	AB	1.04E+07	1.11E+07	2.15E+07	1.58E+06	6.75E+07	1.33E+07	3.10E+06	1.04E+08	1.58E+08	**5.18E+04**
f_{19}	MD	1.01E+07	5.72E+06	2.15E+07	1.59E+06	6.82E+07	1.34E+07	2.25E+06	1.04E+08	1.58E+08	**5.23E+04**

Continued

f		ABC	CAB	CSA	DE	EDA	HS	MFO	SA	SMS	RBFNN-A
	SD	2.91E+06	1.25E+07	1.89E+06	5.43E+05	7.21E+06	1.42E+06	2.53E+06	1.99E+07	1.04E+07	**7.07E+03**
	AB	7.16E+03	4.28E+06	4.28E+05	2.72E+04	1.26E+06	1.58E+05	1.27E+05	1.75E+06	2.54E+06	**9.75E+02**
f_{20}	MD	7.34E+03	4.29E+06	4.33E+05	2.57E+04	1.25E+06	1.59E+05	7.09E+04	1.71E+06	2.50E+06	**9.79E+02**
	SD	1.33E+03	1.45E+05	3.33E+04	1.17E+04	8.29E+04	1.64E+04	1.23E+05	2.32E+05	1.55E+05	**1.07E+02**
	AB	4.01E+05	3.41E+08	2.74E+07	1.58E+06	9.06E+07	8.45E+06	8.40E+06	1.31E+08	2.05E+08	**6.74E+04**
f_{21}	MD	3.63E+05	3.42E+08	2.79E+07	1.31E+06	9.04E+07	8.50E+06	6.15E+06	1.28E+08	2.05E+08	**6.71E+04**
	SD	9.86E+04	5.53E+06	3.25E+06	1.02E+06	8.06E+06	9.32E+05	7.12E+06	2.04E+07	1.21E+07	**6.94E+03**
	AB	2.21E+00	9.27E+02	1.05E+02	7.70E+00	3.17E+02	5.10E+01	1.84E+01	4.02E+02	5.83E+02	**2.51E−01**
f_{22}	MD	2.17E+00	9.19E+02	1.05E+02	7.43E+00	3.18E+02	5.24E+01	1.31E+01	4.05E+02	5.79E+02	**2.50E−01**
	SD	4.68E−01	6.09E+01	7.48E+00	3.16E+00	2.09E+01	5.10E+00	2.08E+01	4.92E+01	2.55E+01	**2.23E−02**
	AB	1.66E+02	9.40E+04	9.15E+03	6.58E+02	2.88E+04	3.77E+03	3.27E+03	3.86E+04	5.78E+04	**2.24E+01**
f_{23}	MD	1.66E+02	9.36E+04	9.15E+03	6.44E+02	2.91E+04	3.81E+03	2.75E+03	3.68E+04	5.78E+04	**2.27E+01**
	SD	3.45E+01	4.45E+03	1.08E+03	2.65E+02	2.17E+03	4.45E+02	2.92E+03	5.72E+03	2.43E+03	**2.44E+00**
	AB	2.42E−01	5.13E+00	8.05E−03	3.83E−04	9.87E−01	9.23E−05	**6.74E−21**	2.21E+00	3.29E+01	1.31E−07
f_{24}	MD	2.06E−01	4.92E+00	7.88E−03	1.39E−05	9.51E−01	9.24E−05	**5.18E−23**	1.88E+00	3.20E+01	8.40E−08
	SD	1.58E−01	1.34E+00	4.07E−03	1.68E−03	2.70E−01	3.84E−05	**2.72E−20**	1.15E+00	1.70E+00	1.52E−07
	AB	4.52E+05	4.32E+46	6.70E+04	2.64E+03	5.04E+22	1.77E+04	6.42E+04	1.74E+05	2.22E+35	**1.52E+02**
f_{25}	MD	3.29E+05	2.14E+43	5.69E+04	2.16E+03	1.31E+21	1.81E+04	6.02E+04	1.70E+05	9.40E+34	**1.52E+02**
	SD	3.35E+05	2.36E+47	3.68E+04	1.54E+03	1.05E+23	1.74E+03	4.49E+04	2.48E+04	3.28E+35	**1.36E+01**
	AB	6.74E+02	5.78E+03	1.45E+03	2.52E+02	3.65E+03	8.98E+02	4.86E+02	4.69E+03	7.01E+03	**1.01E+02**
f_{26}	MD	6.74E+02	5.83E+03	1.45E+03	2.37E+02	3.71E+03	8.96E+02	4.53E+02	4.60E+03	6.99E+03	**1.01E+02**
	SD	4.31E+01	1.72E+02	1.02E+02	6.24E+01	2.49E+02	6.17E+01	2.96E+02	5.27E+02	3.45E+02	**3.96E+00**
	AB	3.17E+08	1.26E+10	2.31E+08	4.98E+06	2.03E+09	7.79E+07	1.37E+07	3.43E+09	5.75E+09	**2.69E+02**
f_{27}	MD	3.04E+08	1.27E+10	2.43E+08	3.81E+06	2.06E+09	7.89E+07	**2.32E+02**	3.38E+09	5.57E+09	2.74E+02
	SD	1.09E+08	8.04E+08	4.68E+07	4.30E+06	3.22E+08	2.33E+07	7.49E+07	7.55E+08	4.72E+08	**4.11E+01**
	AB	4.08E+05	3.70E+46	8.83E+03	1.89E+02	2.33E+22	9.07E+02	1.81E+03	5.24E+03	6.47E+35	**1.08E+02**
f_{28}	MD	4.24E+04	2.72E+43	2.57E+03	1.81E+02	1.11E+20	9.02E+02	1.68E+03	5.23E+03	1.22E+35	**1.09E+02**
	SD	7.69E+05	2.02E+47	1.46E+04	6.52E+01	8.19E+22	4.84E+01	9.10E+02	7.77E+02	1.61E+36	**4.38E+00**
	AB	**6.74E+03**	4.26E+06	4.24E+05	2.61E+04	1.26E+06	1.58E+05	8.59E+04	1.72E+06	2.50E+06	1.98E+06
f_{29}	MD	**6.99E+03**	4.30E+06	4.35E+05	2.62E+04	1.23E+06	1.58E+05	6.44E+04	1.73E+06	2.53E+06	1.98E+06
	SD	**1.37E+03**	1.91E+05	4.86E+04	1.11E+04	1.14E+05	1.51E+04	7.54E+04	2.00E+05	1.04E+05	8.59E+03

Table 3. Results of all methods in 50 dimensions.

of the most efficient algorithms, the entries in these tables that represent the best performance are highlighted in boldface, emphasizing the methods that yield the lowest fitness values.

The optimization outcomes for all 28 functions at 50 dimensions, detailed in Table 3, highlight the RBFNN-A algorithm's superior performance across a broad spectrum of cases, specifically for functions f_1, f_2, f_3, f_4, f_5, f_9, f_{10}, f_{11}, f_{12}, f_{15}, f_{16}, f_{17}, f_{19}, f_{20}, f_{21}, f_{22}, f_{23}, f_{25}, f_{26}, f_{27}, and f_{28}. This demonstrates a notable enhancement in the proposed algorithm's efficiency over traditional metaheuristic approaches. Nevertheless, the algorithm did not achieve optimal results for functions f_6, f_7, f_8, f_{13}, f_{14}, f_{18}, f_{24}, and f_{29}. In direct comparison with the Crow Search Algorithm (CSA), the RBFNN-A notably outperforms on functions f_1, f_2, and f_4 with significant differences in the average best solution (AB) values, such as 1.49E+01, 8.05E+05, and 5.08E−01, respectively. Moreover, against functions f_3, f_5, and f_{10}, RBFNN-A still shows superior efficacy, albeit with smaller margins when compared to CSA. Similarly, against the Differential Evolution (DE) algorithm, RBFNN-A exhibits enhanced performance on the aforementioned functions, with a marked difference in AB values; for instance, a 1.66E+01 difference for function f_3. By using the information of promising regions within the search space, which are effectively delineated by the hidden neurons in the proposed algorithm, a significant enhancement in search efficiency is achieved. This advanced feature enables the algorithm to discern and prioritize areas of the search space that are most likely to yield high-quality solutions, thereby streamlining the path towards discovering optimal or near-optimal solutions. Such an approach stands in stark contrast to traditional metaheuristic strategies, which predominantly hinge on the information derived from the best-performing element within the population. Consequently, this nuanced method of leveraging insights from the broader search landscape, rather than focusing narrowly on the peak performers, facilitates a more informed and strategic exploration. This, in turn, fosters faster convergence rates by effectively bypassing less promising regions and concentrating computational efforts on areas with the highest potential for success.

The analysis of 28 functions at 100 dimensions, as detailed in Table 4, reveals that our algorithm, RBFNN-A, surpasses most competing methods, exhibiting a consistent performance in solving optimization challenges. Despite not achieving top results in eight instances, its overall performance and the potential as a robust metaheuristic solution are clearly demonstrated, especially when considering the average best outcomes. These results further establish RBFNN-A's efficacy as an instrumental approach for complex optimization issues in

f		ABC	CAB	CSA	DE	EDA	HS	MFO	SA	SMS	RBFNN-A
	AB	2.07E+01	2.00E+01	1.88E+01	1.16E+01	2.08E+01	1.83E+01	1.93E+01	2.06E+01	2.13E+01	**3.95E+00**
f_1	MD	2.07E+01	2.00E+01	1.87E+01	1.16E+01	2.08E+01	1.84E+01	1.92E+01	2.06E+01	2.13E+01	**3.92E+00**
	SD	7.84E−02	**7.23E−15**	1.56E−01	1.02E+00	6.52E−02	1.65E−01	4.19E−01	2.17E−01	5.89E−02	1.13E+00
	AB	2.58E+07	1.08E+08	5.03E+06	4.36E+05	2.68E+07	4.62E+06	1.68E+06	3.91E+07	5.75E+07	**3.00E+02**
f_2	MD	2.56E+07	1.06E+08	5.07E+06	4.09E+05	2.71E+07	4.60E+06	1.38E+06	3.95E+07	5.70E+07	**2.93E+02**
	SD	2.58E+06	1.04E+07	6.53E+05	1.70E+05	2.29E+06	3.88E+05	1.43E+06	6.70E+06	2.80E+06	**7.26E+01**
	AB	2.19E+03	7.74E+03	9.56E+02	1.48E+02	2.40E+03	8.03E+02	1.69E+02	3.01E+03	3.67E+03	**9.63E−01**
f_3	MD	2.22E+03	7.77E+03	9.59E+02	1.43E+02	2.42E+03	8.04E+02	1.82E+02	3.02E+03	3.66E+03	**1.09E+00**
	SD	1.69E+02	2.67E+02	4.53E+01	5.08E+01	1.69E+02	4.04E+01	1.43E+02	2.50E+02	1.13E+02	**3.69E−01**
	AB	1.78E+01	7.35E+01	2.02E+00	1.27E−01	1.79E+01	2.01E+00	6.32E−01	1.89E+01	4.20E+01	**1.88E−07**
f_4	MD	1.78E+01	7.39E+01	2.02E+00	1.05E−01	1.83E+01	1.97E+00	1.12E−03	1.94E+01	4.11E+01	**1.34E−07**
	SD	1.92E+00	9.72E+00	3.26E−01	7.46E−02	2.05E+00	2.76E−01	1.04E+00	4.59E+00	3.85E+00	**1.66E−07**
	AB	8.60E+02	2.63E+03	3.53E+02	3.21E+01	9.62E+02	2.59E+02	1.59E+02	1.17E+03	1.75E+03	**8.84E+00**
f_5	MD	8.63E+02	2.70E+03	3.53E+02	3.04E+01	9.64E+02	2.61E+02	1.46E+02	1.18E+03	1.73E+03	**9.88E+00**
	SD	6.42E+01	2.82E+02	2.88E+01	8.61E+00	5.50E+01	1.84E+01	4.19E+01	1.33E+02	1.16E+02	**3.00E+00**
	AB	2.89E+09	1.33E+10	4.52E+08	**2.21E+07**	2.83E+09	4.24E+08	7.27E+07	3.95E+09	5.92E+09	8.54E+07
f_6	MD	2.90E+09	1.34E+10	4.60E+08	1.95E+07	2.87E+09	4.28E+08	**3.41E+06**	4.05E+09	5.80E+09	9.96E+07
	SD	2.51E+08	1.17E+09	5.73E+07	**9.43E+06**	3.16E+08	3.86E+07	1.15E+08	4.11E+08	4.24E+08	2.60E+07
	AB	5.09E+09	2.29E+10	8.15E+08	4.97E+07	5.02E+09	8.14E+08	1.56E+08	7.30E+09	1.01E+10	**4.22E+06**
f_7	MD	5.11E+09	2.27E+10	8.16E+08	4.00E+07	4.93E+09	8.06E+08	8.19E+06	7.12E+09	1.01E+10	**3.37E+06**
	SD	3.72E+08	1.66E+09	8.78E+07	3.34E+07	4.52E+08	9.84E+07	2.23E+08	1.11E+09	5.47E+08	**3.22E+06**
	AB	1.80E+02	4.30E+02	1.58E+02	**5.53E+01**	2.54E+02	1.25E+02	6.25E+01	2.72E+02	3.25E+02	7.17E+01
f_8	MD	1.81E+02	4.30E+02	1.60E+02	**5.50E+01**	2.54E+02	1.27E+02	6.40E+01	2.74E+02	3.26E+02	7.90E+01
	SD	6.97E+00	**7.30E−01**	5.03E+00	4.22E+00	4.44E+00	3.49E+00	1.32E+01	1.56E+01	4.57E+00	1.94E+01
	AB	7.86E+04	4.86E+04	1.96E+04	1.96E+03	9.54E+04	2.23E+04	9.07E+03	1.36E+04	3.48E+05	**7.51E+02**
f_9	MD	8.33E+04	4.91E+04	1.96E+04	2.06E+03	9.89E+04	2.21E+04	9.37E+03	1.02E+04	3.46E+05	**7.67E+02**
	SD	1.17E+04	2.86E+03	2.19E+03	8.64E+02	1.54E+04	2.04E+03	5.28E+03	1.56E+04	3.20E+04	**1.17E+02**
	AB	8.73E+11	4.40E+12	1.92E+11	1.66E+10	8.80E+11	1.76E+11	2.38E+10	1.24E+12	1.67E+12	**3.07E+05**
f_{10}	MD	8.66E+11	4.55E+12	1.97E+11	1.48E+10	9.08E+11	1.74E+11	1.13E+09	1.26E+12	1.67E+12	**3.21E+05**
	SD	6.29E+10	3.19E+11	2.19E+10	5.79E+09	8.23E+10	1.25E+10	3.85E+10	1.70E+11	6.62E+10	**3.13E+04**
	AB	1.87E+03	6.79E+03	3.92E+02	7.14E+01	1.87E+03	3.55E+02	2.14E+02	2.50E+03	3.86E+03	**5.02E+01**
f_{11}	MD	1.87E+03	6.76E+03	3.92E+02	6.82E+01	1.85E+03	3.59E+02	1.92E+02	2.49E+03	3.87E+03	**4.96E+01**
	SD	1.94E+02	8.96E+02	3.42E+01	1.27E+01	1.71E+02	3.04E+01	8.84E+01	3.69E+02	1.54E+02	**3.10E+00**
	AB	1.17E+06	4.44E+06	1.88E+05	1.08E+04	1.17E+06	1.75E+05	4.36E+04	1.40E+06	2.51E+06	**3.51E+02**
f_{12}	MD	1.18E+06	4.40E+06	1.88E+05	9.45E+03	1.18E+06	1.75E+05	1.28E+04	1.42E+06	2.50E+06	**3.86E+02**
	SD	1.08E+05	5.50E+05	1.99E+04	6.26E+03	9.89E+04	1.87E+04	5.96E+04	3.01E+05	1.19E+05	**9.73E+01**
	AB	1.59E+03	2.79E+03	1.12E+03	**1.63E+02**	1.63E+03	8.32E+02	6.69E+02	1.50E+03	2.08E+03	1.76E+02
f_{13}	MD	1.59E+03	2.89E+03	1.12E+03	**1.61E+02**	1.63E+03	8.35E+02	6.70E+02	1.51E+03	2.08E+03	1.82E+02
	SD	4.93E+01	1.27E+02	2.65E+01	2.67E+01	4.84E+01	**2.55E+01**	7.87E+01	1.52E+02	3.67E+01	4.30E+01
	AB	7.36E+06	8.90E+06	2.23E+06	**1.54E+05**	8.10E+06	1.15E+06	1.04E+06	2.27E+06	1.85E+07	2.06E+05
f_{14}	MD	7.48E+06	8.91E+06	2.29E+06	1.34E+05	8.23E+06	1.12E+06	8.27E+05	2.35E+06	1.84E+07	**1.16E+05**
	SD	8.35E+05	8.73E+04	2.85E+05	**6.69E+04**	8.47E+05	1.12E+05	5.82E+05	6.12E+05	1.03E+06	2.43E+05
	AB	9.55E+01	8.88E+01	7.16E+01	5.62E+01	9.56E+01	7.66E+01	9.18E+01	9.68E+01	1.00E+02	**2.57E+00**
f_{15}	MD	9.57E+01	8.87E+01	7.20E+01	5.52E+01	9.57E+01	7.65E+01	9.15E+01	9.70E+01	1.00E+02	**2.93E+00**
	SD	1.17E+00	1.06E+00	2.23E+00	5.60E+00	1.03E+00	9.97E−01	2.17E+00	1.09E+00	**0.00E+00**	7.70E−01
	AB	2.88E+136	4.63E+185	9.14E+117	**6.38E+02**	2.67E+148	1.85E+66	1.80E+03	4.91E+03	1.31E+167	5.59E+08
f_{16}	MD	1.34E+133	8.93E+184	6.22E+115	6.15E+02	8.93E+146	6.38E+62	1.65E+03	4.89E+03	1.75E+166	**1.77E+02**
	SD	1.07E+137	6.55E+04	3.22E+118	**1.41E+02**	8.25E+148	7.76E+66	5.17E+02	3.60E+02	6.55E+04	1.97E+09
	AB	2.40E+05	8.03E+05	1.06E+05	1.55E+04	2.69E+05	8.75E+04	2.09E+04	3.21E+05	4.06E+05	**2.29E+02**
f_{17}	MD	2.38E+05	8.10E+05	1.08E+05	1.52E+04	2.71E+05	8.77E+04	2.05E+04	3.21E+05	4.05E+05	**2.29E+02**
	SD	1.88E+04	2.46E+04	6.14E+03	4.78E+03	1.37E+04	5.05E+03	1.44E+04	2.30E+04	1.12E+04	**1.97E+02**
	AB	−1.13E+03	2.13E+03	−1.98E+03	**−3.36E+03**	−1.09E+03	−2.71E+03	−3.34E+03	−5.64E+02	7.38E+02	−2.17E+03
f_{18}	MD	−1.09E+03	2.20E+03	−1.97E+03	−3.35E+03	−1.04E+03	−2.71E+03	**−3.37E+03**	−5.35E+02	7.02E+02	−2.08E+03
	SD	1.17E+02	1.04E+03	5.26E+01	1.03E+02	1.54E+02	**3.82E+01**	9.28E+01	3.35E+02	1.59E+02	9.47E+02
	AB	2.06E+09	1.19E+09	9.40E+08	1.12E+08	2.45E+09	9.36E+08	1.79E+08	3.39E+09	4.54E+09	**7.58E+01**
f_{19}	MD	2.07E+09	1.19E+09	9.52E+08	1.08E+08	2.46E+09	9.45E+08	1.46E+08	3.33E+09	4.51E+09	**9.44E+01**

Continued

f		ABC	CAB	CSA	DE	EDA	HS	MFO	SA	SMS	RBFNN-A
	SD	1.81E+08	8.10E+07	7.15E+07	3.18E+07	1.52E+08	5.39E+07	8.20E+07	5.60E+08	3.00E+08	**3.54E+02**
	AB	4.14E+06	1.67E+07	2.12E+06	2.76E+05	5.56E+06	1.62E+06	6.13E+05	7.00E+06	8.99E+06	**6.53E+03**
f_{20}	MD	4.10E+06	1.68E+07	2.15E+06	2.74E+05	5.57E+06	1.63E+06	5.81E+05	6.91E+06	9.01E+06	**7.58E+03**
	SD	3.95E+05	5.78E+05	1.41E+05	6.72E+04	2.95E+05	8.72E+04	3.43E+05	8.65E+05	2.90E+05	**3.32E+03**
	AB	4.58E+08	2.68E+09	2.97E+08	3.45E+07	8.43E+08	1.99E+08	1.02E+08	1.08E+09	1.46E+09	**7.34E+05**
f_{21}	MD	4.61E+08	2.69E+09	2.99E+08	3.15E+07	8.49E+08	2.00E+08	9.95E+07	1.07E+09	1.45E+09	**6.86E+05**
	SD	4.09E+07	5.20E+07	2.76E+07	1.11E+07	5.85E+07	9.60E+06	5.38E+07	1.53E+08	6.97E+07	**5.72E+05**
	AB	6.37E+02	1.89E+03	2.78E+02	4.30E+01	6.93E+02	2.29E+02	3.96E+01	8.34E+02	1.08E+03	**1.03E+00**
f_{22}	MD	6.38E+02	1.90E+03	2.79E+02	4.32E+01	7.03E+02	2.30E+02	2.86E+01	8.37E+02	1.08E+03	**1.03E+00**
	SD	5.18E+01	1.00E+02	1.59E+01	1.13E+01	3.90E+01	1.17E+01	2.88E+01	6.17E+01	2.82E+01	**1.77E−01**
	AB	9.87E+04	3.67E+05	4.97E+04	6.73E+03	1.32E+05	3.68E+04	1.58E+04	1.57E+05	2.11E+05	**1.80E+02**
f_{23}	MD	9.94E+04	3.69E+05	4.99E+04	6.84E+03	1.33E+05	3.72E+04	1.49E+04	1.54E+05	2.10E+05	**1.85E+02**
	SD	6.61E+03	1.93E+04	2.48E+03	1.89E+03	6.60E+03	2.05E+03	6.19E+03	1.79E+04	8.18E+03	**3.67E+01**
	AB	1.39E+00	5.29E+00	1.02E−02	1.65E−04	1.30E+00	9.74E−04	**4.02E−07**	2.55E+00	6.09E+01	5.56E−04
f_{24}	MD	1.38E+00	5.22E+00	9.17E−03	3.79E−05	1.32E+00	9.71E−04	**7.59E−10**	2.70E+00	6.00E+01	1.67E−04
	SD	2.99E−01	1.46E+00	5.99E−03	3.91E−04	3.58E−01	4.45E−04	**1.41E−06**	8.59E−01	2.47E+00	1.08E−03
	AB	6.14E+35	7.49E+85	7.11E+18	1.67E+04	3.08E+48	8.66E+04	1.32E+05	3.28E+05	1.28E+67	**4.32E+02**
f_{25}	MD	6.22E+32	2.62E+85	8.51E+15	1.70E+04	8.22E+45	8.75E+04	1.26E+05	3.36E+05	1.65E+66	**5.53E+02**
	SD	2.61E+36	1.19E+86	3.38E+19	5.47E+03	8.83E+48	6.41E+03	6.07E+04	3.22E+04	2.37E+67	**2.25E+02**
	AB	7.97E+03	1.15E+04	3.57E+03	8.98E+02	8.02E+03	3.25E+03	1.20E+03	9.30E+03	1.31E+04	**2.51E+02**
f_{26}	MD	7.98E+03	1.15E+04	3.58E+03	8.79E+02	8.04E+03	3.26E+03	1.08E+03	9.23E+03	1.30E+04	**2.76E+02**
	SD	3.73E+02	3.64E+02	1.71E+02	1.42E+02	4.58E+02	9.81E+01	3.65E+02	8.22E+02	6.35E+02	**8.02E+01**
	AB	5.05E+09	2.68E+10	8.56E+08	4.96E+07	5.08E+09	8.12E+08	1.40E+08	7.11E+09	1.02E+10	**2.34E+04**
f_{27}	MD	5.05E+09	2.69E+10	8.40E+08	4.95E+07	5.15E+09	8.06E+08	1.96E+06	7.30E+09	1.01E+10	**1.45E+04**
	SD	3.27E+08	1.86E+09	1.10E+08	2.15E+07	5.57E+08	9.66E+07	2.47E+08	8.39E+08	5.37E+08	**2.63E+04**
	AB	7.01E+36	4.23E+85	3.08E+17	8.40E+02	2.52E+48	3.32E+03	4.61E+03	1.09E+04	4.32E+68	**2.47E+02**
f_{28}	MD	3.26E+32	4.55E+84	3.74E+16	8.06E+02	1.50E+46	3.34E+03	4.13E+03	1.07E+04	1.12E+67	**2.83E+02**
	SD	2.34E+37	1.23E+86	7.85E+17	2.57E+02	6.07E+48	1.04E+02	2.07E+03	9.75E+02	1.37E+69	**9.45E+01**
	AB	4,221,693.82	14,157,709.1	2,163,788.14	**324,078.001**	5,604,717.41	1,597,070.55	542,116.463	7,269,725.62	9,176,233.14	2,105,812.77
f_{29}	MD	4,184,048.47	12,915,098.1	2,176,222.87	**303,893.571**	5,611,576.98	1,604,903.95	513,655.683	7,181,451.91	9,092,269	2,185,539.3
	SD	371,495.514	2,082,531.12	151,172.042	**92,292.3996**	266,599.4	95,447.0432	231,551.264	737,656.571	428,782.825	389,281.087

Table 4. Algorithm results with 100 dimensions.

metaheuristic research. Specifically, RBFNN-A outperformed other algorithms in 21 out of 29 functions tested, notably excelling in functions f_1, f_2, f_3, f_4, f_5, f_7, f_9, f_{10}, f_{11}, f_{12}, f_{15}, f_{17}, f_{19}, f_{20}, f_{21}, f_{22}, f_{23}, f_{25}, f_{26}, f_{27}, and f_{28}. Its performance fell short in functions f_6, f_8, f_{13}, f_{14}, f_{16}, f_{18}, f_{24}, and f_{29}. In comparison with algorithms such as ABC, CAB, CSA, DE, EDA, HS, MFO, SA, and SMS, RBFNN-A demonstrated significant advantages, especially in the average best (AB) scores for the evaluated functions. For instance, in function f_1, RBFNN-A achieved an AB of 3.95, significantly better than the ABC algorithm, which scored 20.7, highlighting RBFNN-A's superior performance. The innovative design of our proposed algorithm (RBFNN-A) plays a pivotal role in enhancing the optimization process by adeptly identifying the most promising regions within the search space, as determined by its hidden neurons. This capability is crucial for achieving an optimal balance between exploration and exploitation—two fundamental aspects of search strategies in optimization algorithms. Exploration involves venturing into new, uncharted areas of the search space to uncover potential solutions, while exploitation focuses on thoroughly investigating known, promising regions to refine existing solutions. By leveraging the predictive power of RBFNN-A to assess the potential of unexplored areas, our method intelligently decides when to embark on exploration to discover new possibilities and when to concentrate on exploitation to maximize the utility of already identified promising zones. This strategic balance ensures that computational resources are utilized efficiently, leading to a more effective and systematic approach to reaching optimal solutions.

Statistical analysis

Metaheuristic algorithms, characterized by their stochastic optimization processes, leverage randomness to navigate complex problem spaces in search of optimal solutions. Due to their inherent stochasticity, the performance outcomes of these algorithms can fluctuate across various executions or datasets. To establish the statistical significance of the results yielded by different methods, the Wilcoxon signed-rank test[47,48] is employed. This non-parametric statistical test is designed to assess whether the performance differences observed between two algorithms are statistically significant or merely the result of random variation. Unlike parametric tests, the Wilcoxon test does not assume a specific distribution for the data. Instead, it compares the relative rankings of paired observations—typically, the performance metrics of two algorithms tested against the same suite of

functions. By analyzing these ranks, the test calculates a *p*-value, reflecting the likelihood that the noted performance discrepancies are due to chance. Should this *p*-value fall below a certain threshold, such as 0.05, the differences are considered statistically significant, indicating that one algorithm consistently outperforms or underperforms the other in a meaningful way.

In our study, the Wilcoxon signed-rank test is applied to the results listed in Table 5, which features the performance of various algorithms on functions evaluated at 100 dimensions. This table facilitates a statistical comparison between our proposed RBFNN-A method and other metaheuristic approaches, offering a robust statistical validation of the conducted experiments. For clarity, Table 5 employs specific symbols to denote the outcomes of the Wilcoxon test: the symbol ▲ signifies that RBFNN-A has outperformed a competing algorithm, ▼ indicates instances where RBFNN-A was outperformed by another method, and ► denotes cases where no statistically significant difference was observed between RBFNN-A and another algorithm according to the Wilcoxon test. Analysis of Table 5 reveals that RBFNN-A demonstrates superior performance over competing algorithms across a wide array of functions, specifically f_1, f_2, f_3, f_4, f_5, f_7, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{15}, f_{16}, f_{17}, f_{19}, f_{20}, f_{21}, f_{22}, f_{23}, f_{24}, f_{25}, f_{26}, f_{27}, and f_{28}. RBFNN-A showed no significant difference when compared with the MFO algorithm at f_8 and faced challenges outperforming the ABC, CSA, EDA, HS, MFO, and SA algorithms at functions f_{18} and f_{29}, and the DE algorithm at f_6, f_8, f_{14}, f_{18} and f_{29}, highlighting the nuanced landscape of algorithmic performance across different optimization challenges. Our approach has yielded substantial advantages in terms of scalability. By using the information encoded within the neural network, our algorithm is capable of effectively addressing larger and more complex problems. This is particularly beneficial in overcoming the curse of dimensionality, a prevalent challenge in high-dimensional optimization tasks. The neural network's capacity to compactly represent and utilize intricate patterns and relationships within the data allows our method to navigate these vast search spaces more efficiently. Our algorithm can identify relevant features and dynamics of the optimization problem by leveraging the neural network's encoded knowledge, thus facilitating a more targeted and informed search process. As a result, our method is well-equipped to manage the complexities inherent in high-dimensional optimization problems, making it a reliable solution for a range of challenging scenarios.

RBFNN-A vs	ABC	CAB	CSA	DE	EDA	HS	MFO	SA	SMS
f_1	2.53E−11▲	9.62E−13▲	2.53E−11▲	1.87E−03▲	2.53E−11▲	2.53E−11▲	2.53E−11▲	2.53E−11▲	2.53E−11▲
f_2	2.53E−11▲	2.53E−11▲	2.53E−11▲	1.01E−06▲	2.53E−11▲	2.53E−11▲	2.54E−09▲	2.53E−11▲	2.53E−11▲
f_3	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	1.07E−07▲	3.02E−11▲	3.02E−11▲
f_4	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_5	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_6	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▼	3.02E−11▲	3.02E−11▲	1.95E−03▼	3.02E−11▲	3.02E−11▲
f_7	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	6.63E−01▲	3.02E−11▲	3.02E−11▲
f_8	2.64E−11▲	1.48E−12▲	2.63E−11▲	2.03E−10▼	2.63E−11▲	2.52E−11▲	5.72E−01►	2.66E−11▲	2.63E−11▲
f_9	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	6.77E−05▲	3.02E−11▲
f_{10}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{11}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{12}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	6.77E−05▲	3.02E−11▲	3.02E−11▲
f_{13}	3.02E−11▲	2.11E−11▲	3.02E−11▲	1.29E−09▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{14}	3.02E−11▲	1.72E−12▲	3.02E−11▲	3.02E−11▼	3.02E−11▲	3.02E−11▲	1.67E−01▲	5.57E−10▲	3.02E−11▲
f_{15}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	1.21E−12▲
f_{16}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{17}	3.00E−11▲	1.98E−11▲	3.00E−11▲	3.00E−11▲	3.00E−11▲	3.00E−11▲	4.11E−08▲	3.00E−11▲	3.00E−11▲
f_{18}	3.02E−11▼	3.02E−11▲	3.02E−11▼	3.02E−11▼	3.02E−11▼	3.02E−11▼	3.02E−11▼	5.57E−10▼	3.02E−11▲
f_{19}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{20}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{21}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{22}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{23}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{24}	3.02E−11▲	3.02E−11▲	3.02E−11▲	8.48E−09▲	3.02E−11▲	3.02E−11▲	6.28E−06▲	3.02E−11▲	2.84E−11▲
f_{25}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{26}	3.02E−11▲	2.92E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{27}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{28}	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲	3.02E−11▲
f_{29}	3.02E−11▼	3.02E−11▲	3.02E−11▼	3.02E−11▼	3.02E−11▼	3.02E−11▼	3.02E−11▼	3.02E−11▼	3.02E−11▲
▲	27	29	27	24	27	27	25	27	29
►	0	0	0	0	0	0	1	0	0
▼	2	0	2	5	2	2	3	2	0

Table 5. Wilcoxon *p*-values obtained over 100-dimentional results.

Convergence analysis

Convergence analysis[49] is an examination that aims to evaluate how metaheuristic algorithms approach an optimal solution to a given problem over time. The importance of convergence analysis lies in its ability to provide insights into the reliability, efficiency, and effectiveness of a metaheuristic algorithm. This analysis is crucial for algorithm selection and tuning in practical applications, as it helps to guarantee that the chosen metaheuristic will perform well on specific classes of problems, thereby saving computational resources and improving solution quality.

In our study to evaluate the convergence of metaheuristic algorithms, we conducted an experiment where all participating algorithms were tasked with optimizing the same set of 29 functions concurrently. The primary metric for assessing each algorithm's performance was the point they took to reach their best solution. However, instead of measuring this point in conventional iterations, we quantified it based on the number of function accesses required to achieve the optimal solution. This approach was chosen over counting iterations because metaheuristic algorithms vary widely in their design, with some algorithms inherently making more function accesses per iteration than others. This variance could skew the results if iterations were the sole metric. By focusing on function accesses, we ensure a more equitable and objective comparison, accurately reflecting each algorithm's efficiency in finding the best solution without bias towards their operational intricacies.

Table 6 presents a comparative analysis of the number of function accesses required by different algorithms to achieve convergence on a set of functions, specifically considering problems with 50 dimensions. Notably, the proposed algorithm stands out for its efficiency, achieving convergence with markedly fewer function accesses than its competitors. This efficiency is attributed to the algorithm's strategy of progressively narrowing down the search area within the function landscape, focusing on regions identified as most promising for locating the global solution. Through this iterative process, the algorithm systematically reduces the search space until no further promising regions remain to be explored or exploited, at which point convergence is achieved. The effectiveness of this approach is starkly illustrated in the comparison with the next best performing algorithm, the SMS, which required 81,461 function evaluations to reach convergence. In contrast, the proposed algorithm attained convergence with just approximately 1400 function accesses, highlighting its exceptional capability to navigate through complex optimization problems efficiently. This significant reduction in the number of required function evaluations not only demonstrates the proposed algorithm's superior convergence efficiency but also

Function	ABC	CAB	CSA	DE	EDA	HS	MFO	SA	SMS	RaBaNN
f_1	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	100,882	1420
f_2	99,953	100,012	100,001	100,000	100,000	100,000	100,000	100,000	86,997	1432
f_3	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	83,542	1411
f_4	99,953	100,012	100,001	100,000	100,000	100,000	100,000	100,000	87,721	1408
f_5	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	84,916	1489
f_6	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	87,800	1427
f_7	99,956	100,012	100,001	100,000	100,000	100,000	100,000	100,000	93,939	1461
f_8	99,951	100,012	100,001	100,000	100,000	100,000	100,000	100,000	93,698	1700
f_9	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,802	1421
f_{10}	99,954	100,012	100,001	100,000	100,000	100,000	100,000	100,000	84,062	1408
f_{11}	99,952	100,012	100,001	100,000	100,000	100,000	100,000	100,000	82,940	1401
f_{12}	99,952	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,604	1400
f_{13}	99,952	100,012	100,001	100,000	100,000	100,000	100,000	100,000	86,934	1429
f_{14}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,098	1439
f_{15}	99,951	100,012	100,001	100,000	100,000	100,000	100,000	100,000	81,954	1442
f_{16}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	92,599	1424
f_{17}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	90,482	1455
f_{18}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,298	1488
f_{19}	99,961	100,012	100,001	100,000	100,000	100,000	100,000	100,000	81,461	1420
f_{20}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	83,539	1413
f_{21}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	94,094	1451
f_{22}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,234	1439
f_{23}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	86,697	1485
f_{24}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	85,063	1474
f_{25}	99,956	100,012	100,001	100,000	100,000	100,000	100,000	100,000	93,761	1479
f_{26}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,642	1407
f_{27}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,039	1430
f_{28}	99,955	100,012	100,001	100,000	100,000	100,000	100,000	100,000	90,105	1477
f_{29}	99,950	100,012	100,001	100,000	100,000	100,000	100,000	100,000	91,416	1489

Table 6. Convergence results for functions in 50 dimensions.

positions it as a highly valuable tool for tackling high-dimensional optimization challenges across various scientific and engineering fields.

Performance comparison considering the 12 CEC 2022 functions from table B

The CEC 2022[50] Special Session on Single Objective Optimization involves a diverse set of benchmark functions designed to test the performance of optimization algorithms across a range of challenges. These functions are specifically chosen to represent various difficulties encountered in real-world optimization problems. The functions can be summarized as it is shown in Table B. The CEC 2022 functions include shifting and rotating basic functions, as well as combining several of them into a single function. These modifications create functions that are significantly more challenging to optimize than traditional functions. By incorporating shifts and rotations, the search space becomes more complex and less predictable, making it harder for optimization algorithms to find the global optimum. Additionally, the combination of multiple basic functions into composite and hybrid functions introduces further intricacies, as the algorithm must navigate varied landscapes within a single problem. This increased difficulty helps to more rigorously test the robustness, adaptability, and overall performance of optimization methods, ensuring they are capable of handling real-world optimization challenges.

In the optimization of the CEC 2022 functions, the standard test with the execution of the functions in 20 dimensions was considered. To mitigate the random effects inherent to the stochastic nature of the methods, each function was executed 30 times for each metaheuristic algorithm. The results, presented in Table 7, show the numerical values obtained as the average of the best values (AB) across the 30 runs, along with the dispersion in terms of standard deviation (SD). This comprehensive testing approach ensures that the performance of each algorithm is reliably assessed, providing a clear comparison of their effectiveness and consistency in optimizing the challenging CEC 2022 functions.

According to the values in Table 7, it is clear that the proposed RBFNN-A method produces the best solutions in terms of accuracy (AB). The DE and ABC methods also perform well, whereas the other methods show poor performance. A more detailed analysis of the table reveals that, generally, the methods perform better on functions F_1 to F_5, which are basic unimodal and multimodal functions. In contrast, functions F_6 to F_{12} pose greater challenges, as they represent combinations of several functions that generate higher complexity. It is in these more complex functions that the proposed algorithm's performance stands out the most, significantly differing from the other methods and highlighting its superior capability in handling difficult optimization problems.

From Table 7, it is evident that the proposed method RBFNN-A is more robust in terms of standard deviation (SD), as it produces a smaller dispersion of solutions across the set of runs. To illustrate the distribution of the solutions more clearly, Fig. 6 shows the distributions for each metaheuristic algorithm on three selected functions: (a) F_2, (b) F_8 and (c) F_{12}. These functions were chosen to represent different types of functions in the CEC 2022

		ABC	CAB	CSA	DE	EDA	HS	MFO	SA	SMS	RBFNN-A
F_1	AB	300.9	341.2	17,500.4	300	331.7	1321.2	3072.8	300.8	300.5	300
	SD	2.12E−10	2.21E−14	1.53E+04	2.45E−10	7.12E−14	1.18E+03	9.11E+02	4.71E−14	3.16E−01	0
F_2	AB	418.45	421.21	468.36	411.55	409.21	415.50	413.16	412.27	422.81	401.17
	SD	20.8	17.4	80.2	22.7	10.12	15.2	11.32	18.1	17.6	4.12
F_3	AB	616.31	610.87	634.81	616.72	632.21	652.32	630.74	672.54	647.11	600.30
	SD	7.21E+00	4.32E−01	2.27E+01	7.59E+00	3.51E−04	9.28E−01	8.25E−08	2.03E−01	4.63E+00	4.90E−01
F_4	AB	874.11	892.48	853.38	829.92	878.09	911.51	889.56	891.38	899.96	806.57
	SD	8.7	7.21	31.83	9.58	10.21	11.45	11.92	8.19	10.19	7.30
F_5	AB	926.31	970.56	1997.1	921.46	945.20	992.25	975.41	980.46	947.94	900
	SD	6.01E+01	1.85E−01	1.58E+03	6.01E+01	4.16E−01	3.14E+01	2.37E−11	3.80E+00	5.84E+01	5.11
F_6	AB	1889.7	4388.8	2521.9	1827.1	3865.6	3514.7	1932.5	2324.8	3465.2	1800.1
	SD	1.72E+01	2.55E+03	1.22E+07	1.78E+01	1.62E+03	2.35E+03	5.92E+01	1.16E+03	2.12E+03	6.3
F_7	AB	2043.6	2243.5	2100.9	2021.3	2522.1	2227.8	2405.7	2324.4	2128	2008
	SD	13.8	19.80	60.8	17.9	20.4	11.5	11.90	16.30	9.10	8.21
F_8	AB	2356.8	2524.4	2448.6	2248.6	2541.6	2322.6	2298.8	2318.1	2328.7	2205.8
	SD	40.1	25.6	42.7	24.8	26.3	15.30	23.8	26.7	19.24	13.20
F_9	AB	3002.8	2724.6	2634.4	2629.1	2733.0	2787.5	2528.2	2734.8	2638.8	2529.8
	SD	50.32	54.30	59.7	43.32	36.7	30.9	10.2	56.5	44.73	10.7
F_{10}	AB	2674.21	2759.3	2849.6	2595.3	2667.8	2756.2	2700.6	2611.4	2660.8	2500.2
	SD	61.5	56.3	90.5	69.7	59.0	52.7	143.8	20.56	59.3	54.8
F_{11}	AB	2731.5	2876.4	2990.4	2731.0	2840.4	2894.9	3321.1	2998.3	2947.3	2608.4
	SD	116.4	147.5	329.2	126.1	110.4	171.7	208.38	159.56	179.4	60.42
F_{12}	AB	2971.8	3073.3	3178.5	2891.7	2963.5	3065.4	3104.7	2964.2	3098.6	2860.6
	SD	110.9	150.8	88.5	92.9	62.04	74.72	131.0	65.5	61.15	35.19

Table 7. Numerical results for functions in 20 dimensions from the CEC 2022.

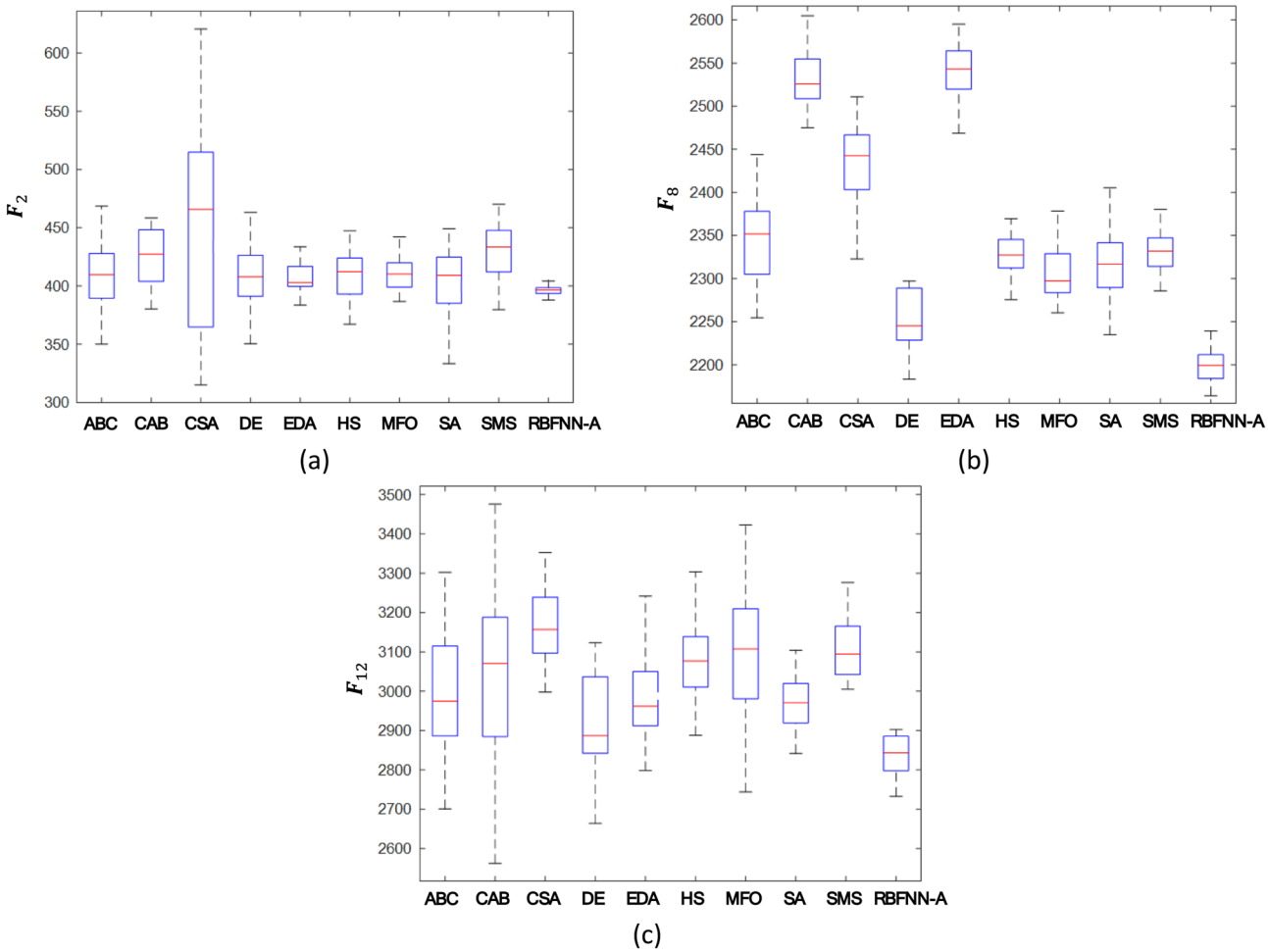

Figure 6. Distributions for each metaheuristic algorithm on three selected functions: (**a**) F_2, (**b**) F_8 and (**c**) F_{12}.

set. Specifically, F_2 is a basic multimodal function, F_8 is a hybrid function, and F_{12} is a composite function. The figure demonstrates that the proposed method has the smallest distribution for each of these functions, indicating consistent performance, whereas the other methods exhibit varying degrees of dispersion. This highlights the robustness and reliability of the proposed method in producing stable and accurate solutions across different types of optimization problems.

To assess the performance of the methodologies examined in our study, we utilized the Friedman ranking test[51]. The Friedman ranking test is a non-parametric statistical test that is suitable for evaluating and ranking multiple techniques across different datasets or scenarios, particularly when the underlying data does not meet the normality assumption. This test offers a dependable alternative to parametric tests, such as ANOVA. In the Friedman test, each method is ranked based on its performance within each dataset, with the best-performing method receiving a rank of 1, the second-best a rank of 2, and so on. If two methods perform equally, they are assigned the average of the ranks they would occupy. Table 8 presents the findings of the Friedman analysis, which considers the AB numerical values from Table 7. Based on these results, the RBFNN-A method emerges as the top performer, trailed by DE and the ABC, while the other methods receive lower rankings, indicating suboptimal performance.

Engineering design problems

Validating the performance of a new metaheuristic method in engineering design problems is crucial for several reasons. First, engineering design problems often involve complex and nonlinear optimization tasks that require robust and efficient solution strategies. By testing a new metaheuristic method in these scenarios, researchers can assess its effectiveness, reliability, and performance under realistic conditions. Second, validation helps ensure that the new method can handle the specific constraints and requirements inherent in engineering problems. This process also highlights the method's advantages over existing approaches, potentially leading to more innovative and optimized design solutions.

This subsection focuses on applying the proposed method to a range of interesting engineering design problems, including the Three-bar truss, Pressure vessel, and Welded-beam. The objective is to test the effectiveness of the proposed metaheuristic method compared to other techniques in these practical scenarios. By evaluating the performance of the methods on these diverse engineering challenges, we aim to demonstrate the robustness

Algorithm	Ranking	Mean value
RBFNN-A	1	5.8293
DE	2	4.7012
ABC	3	4.5021
MFO	4	4.0022
EDA	5	3.8032
SA	6	3.7138
SMS	7	3.5012
CAB	8	3.2019
CSA	9	2.9721
HS	10	2.7541

Table 8. Friedman Test ranking results of the AB values form Table 5.

and applicability of the proposed approach in real-world situations. For detailed mathematical descriptions of each problem, readers are referred to Table B in the appendix.

In the analysis of engineering design problems, the same metaheuristic algorithms from the previous analysis (ABC, CAB, CSA, DE, EDA, HS, MFO, SA, SMS, and RBFNN-A) were employed. Each experiment was conducted with a population of 50 individuals and a stopping criterion of 1000 iterations for every algorithm. Given the stochastic nature of metaheuristic algorithms, each engineering problem was independently executed 30 times to ensure reliable results. The numerical outcomes for each engineering challenge are detailed, considering their specific decision variables, constraints, and fitness values. Additionally, the statistical evaluation includes the averaged best fitness values (AB) and the standard deviation (SD) to provide a comprehensive understanding of each algorithm's performance. This thorough analysis ensures that the results are robust and that the effectiveness of each algorithm in solving engineering design problems is accurately assessed.

Optimization problems in engineering often have constraints on their decision variables, explicitly defined by constraint functions. To handle these constraints, the constraint separation technique has been employed in the following experiments. This technique involves treating constraints independently from the objective function during the optimization process. With this methodology, each candidate solution generated by the metaheuristic algorithm is checked for feasibility by evaluating it against all constraints. Only those solutions that satisfy all the constraints are considered feasible and are passed on to the next phase of the optimization process. Conversely, infeasible solutions are discarded and replaced by new ones. This approach ensures that the optimization focuses solely on viable solutions, enhancing efficiency and effectiveness in finding the optimal solution.

A) Three-bar truss design problem

The Three-bar truss design problem[52] involves a 2-dimensional search space, with the objective of minimizing the structure of a loaded three-bar truss. This truss structure is graphically depicted in Fig. 7. The optimization process is subject to three constraints, detailed in Table C in the appendix. The results of the optimization problem for the Three-Bar Truss are summarized in Table 9. These results demonstrate that the proposed method successfully finds the optimal solution for this engineering problem. The method's capability to effectively handle the constraints and complexities of this problem highlights its practical application and efficiency in solving tangible engineering challenges.

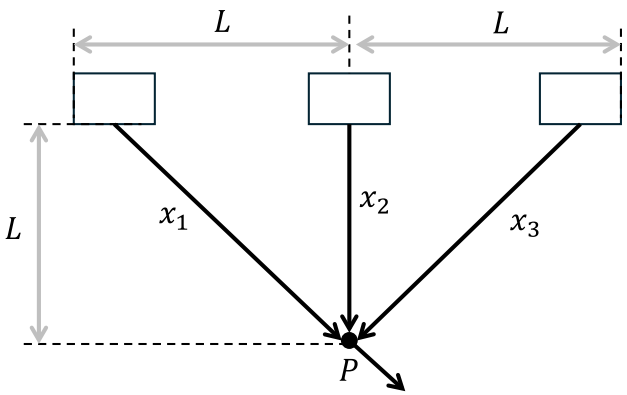

Figure 7. Three-bar truss design problem.

Algorithms	AB	SD
ABC	1.7691e+02	4.9056e−09
CAB	1.7723e+02	2.4984e−13
CSA	1.7331e+02	7.1200e−02
DE	1.7598e+02	1.5000e−03
EDA	1.7891e+02	3.9600e−02
HS	1.7782e+02	1.0000e−04
MFO	1.7599e+02	6.5500e+04
SA	1.7793e+02	7.3500e−02
SMS	1.9531e+02	6.5500e−02
RBNFF-A	1.7593e+02	7.2201e−02

Table 9. Results produced by the Three-bar truss design problem.

B) Pressure vessel design problem

The Pressure Vessel Design problem[53] is depicted in Fig. 8 and aims to minimize the total cost of material, forming, and welding, taking into account various design parameters such as the thickness of the shell ($T_s = x_1$), thickness of the head ($T_h = x_2$), inner radius ($R = x_3$), and length of the shell ($L = x_4$). A detailed mathematical description of this problem can be found in Table C of the appendix. Table 10 summarizes the results of applying the proposed method to this design problem, which show that the RBNFF-A method, along with the DE and MFO methods, obtain the best solution to the problem, while the other methods had varying levels of performance.

C) Welded-beam design problem

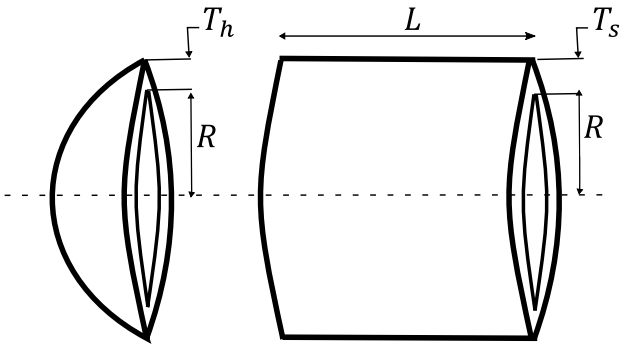

Figure 8. Pressure vessel engineering design problem.

Algorithms	AB	SD
ABC	5.8925e+03	1.0230e+01
CAB	5.8972e+03	7.7689e+01
CSA	5.8912e+03	2.2354e+01
DE	5.8854e+03	1.7046e+02
EDA	5.8890e+03	1.3423e+02
HS	5.9990e+03	3.0013e+02
MFO	5.8856e+03	3.5213e+02
SA	8.8722e+03	4.0120e+02
SMS	5.9692e+03	1.6784e+02
RBNFF-A	5.8853e+03	0.2201e+01

Table 10. Results produced by the Pressure vessel design problem.

The Welded-Beam Design problem [54] focuses on minimizing manufacturing costs by optimizing four key design variables (see Fig. 9): width ($h = x_1$), length ($l = x_2$), depth ($t = x_3$), and thickness ($b = x_4$). Governed by seven constraints, this problem presents a complex scenario typical of real-world engineering challenges. Detailed specifications for the Welded-Beam design are elaborately listed in Table C of the Appendix. The effectiveness of various algorithms in solving this problem, including the proposed method, is documented in Table 11. Analysis of the data in this table reveals that the proposed method outperforms other well-known metaheuristic methods, achieving the best solution for this specific design problem. This result highlights not only the proposed methodology's effectiveness but also confirms its practical applicability and superiority in tackling real-world engineering problems, particularly in cost-efficient design optimization.

Time complexity

In many engineering optimization problems, the cost of function evaluations is a major constraint that limits the number of evaluations available. This limitation poses a significant challenge to the field of global optimization, as conventional metaheuristic techniques typically require a substantial number of function evaluations to identify optimal solutions. Consequently, the high computational demand of these methods often exceeds what is feasible within the given resource constraints, necessitating the development of more efficient algorithms that can deliver high-quality results with fewer function evaluations. This underscores the need for innovative approaches that balance accuracy and efficiency in resource-constrained optimization scenarios.

In the proposed approach, the proposed RBFNN algorithm is trained to model the objective function values based on the current set of solutions. The algorithm identifies the neurons in the hidden layer that correspond to the highest objective function values. It then utilizes the parameters of these crucial neurons, such as their centroids and standard deviations, to generate new solutions by sampling in order of importance until the typical number of solutions is reached. By focusing on the promising regions of the search space that yield high objective function values, the algorithm avoids the need for exhaustive evaluation of all possible solutions. This strategy enables a more informed and efficient exploration of the search space. Consequently, the proposed method can find the global solution with a significantly reduced number of objective function evaluations.

To extend the analysis of the performance of all methods, a convergence test evaluating the number of function evaluations employed by the compared algorithms has been conducted. The purpose of this experiment is to assess the speed at which each method reaches the optimum in terms of function evaluations. The experiment considers the performance of each algorithm across all functions ($f_1 - f_{29}$) from Table A of the Appendix, each operated in 50 dimensions (n) with 50 individuals (N). Each algorithm is run on each function, recording the number of evaluations required to obtain the minimum function value within 1000 iterations. To mitigate

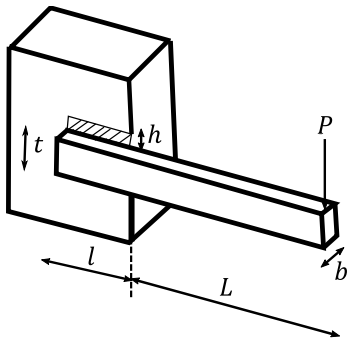

Figure 9. Welded-beam engineering design problem.

Algorithms	AB	SD
ABC	1.7345e+00	1.0210
CAB	1.7530e+00	1.3452
CSA	1.7511e+00	1.0014
DE	2.8205e+00	1.2101
EDA	1.7481e+00	1.0781
HS	17023e+00	0.8947
MFO	1.7119e+00	0.8501
SA	1.9903e+00	1.0478
SMS	1.8954e+00	1.0078
RBNFF-A	1.6509e+00	0.5427

Table 11. Results produced by the Welded-beam design problem.

random effects, each function is run 30 times. Figure 10a displays the average number of function evaluations across all functions and their 30 executions. The results in Fig. 10a show that all the metaheuristic methods maintain a number of function evaluations in the range of 35,000–50,000. In contrast, the proposed method demonstrates a significantly lower number of evaluations, ranging from 15,000 to 18,000. This indicates that the proposed method converges faster, producing the best results with fewer evaluations compared to the other methods.

All metaheuristic methods generate new solutions using a variety of heterogeneous processes, ranging from simple to complex, each with varying computational resource demands. The intrinsic stochastic components and intricate structures of metaheuristic algorithms make conventional complexity analysis impractical. To analyze the time efficiency of each method, an experiment was conducted. In this experiment, each method was executed on each function ($f_1 - f_{29}$) listed in Table A of the Appendix. The time in seconds required to obtain the minimum value with respect to the objective function was recorded. To mitigate the effect of randomness, each function's execution was repeated 30 times. Figure 10b displays the inverted time values, averaged over all functions and their 30 executions. This analysis provides a comparative understanding of the computational efficiency of each method, highlighting the differences in time performance across various algorithms.

The results in Fig. 10b show that all the metaheuristic methods maintain a processing time in the range of 20–80 s. In contrast, the proposed method exhibits significantly higher processing times, ranging from 160 to 180 s. This indicates that, although the proposed approach requires generating very few solutions to obtain the minimum value of the function, the processes used to generate these solutions consume a substantial amount of computational resources. Consequently, while the method is efficient in terms of the number of evaluations, it demands more computational time due to the complexity of its solution generation processes.

In the proposed approach, the process of training the radial basis neural network (RBFNN) is primarily responsible for the computational cost of the method. The computational complexity of training a RBFNN primarily depends on the number of training elements N and the number of neurons M in the hidden layer. Specifically, the training process involves determining the parameters of the radial basis functions, such as the centers and widths, and the weights connecting the hidden layer to the output layer. The complexity typically scales with $O(N \times M)$. As the number of training elements N increases, the computational load also increases, because more data points need to be processed to adjust the network parameters accurately. Consequently, for large values of N, the training time can grow significantly, potentially leading to longer training times and higher computational resource demands. This relationship underscores the importance of optimizing the number of hidden neurons and employing efficient training algorithms to manage the computational burden effectively.

The proposed RBFNN-A method in expensive optimization

In the industries, there is a growing emphasis on designing products using computer models. These models facilitate the exploration of alternative designs and significantly reduce the need for costly hardware prototypes. Often, these computational models require long execution times due to the complexity of the programmed codes that represent their causes and effects, such as finite element analysis and computational fluid dynamics. Improving the design of products or processes involves optimizing certain elements using these computationally intensive models. This type of optimization is known as optimization of expensive problems or expensive optimization. In this context, one simulation equates to a single function evaluation in common optimization problems. Due to the high computational cost, only a limited number of function evaluations can typically be used for expensive optimization problems. This constraint limits the application of many optimization algorithms, such as classical evolutionary or metaheuristic algorithms, which usually require a large number of function evaluations to obtain a good solution.

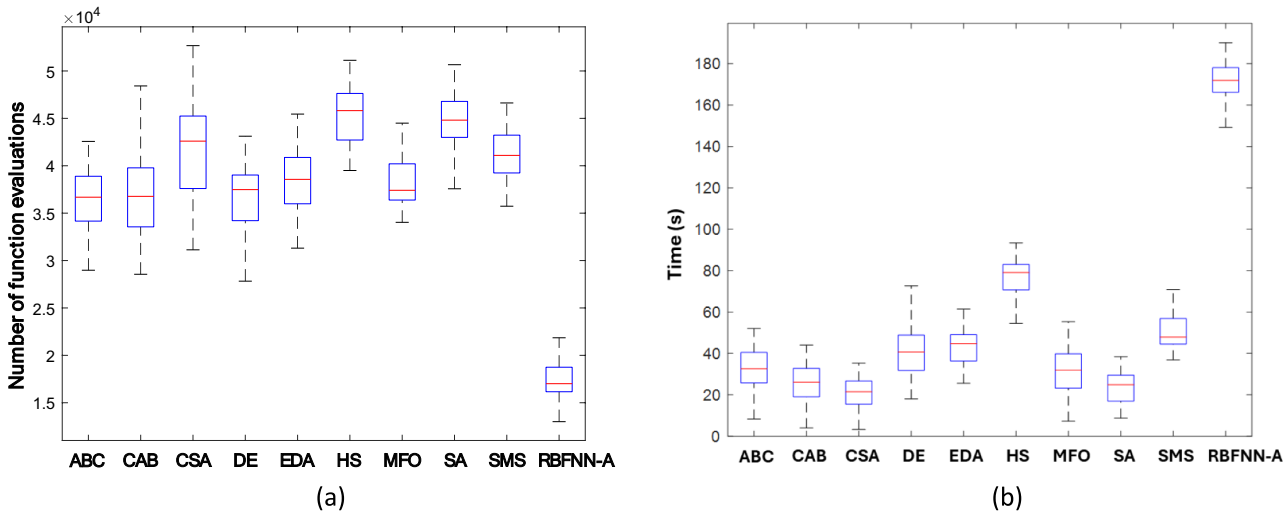

Figure 10. Convergence graphs in terms of (**a**) number of function evaluations and (**b**) time spent during the execution.

The proposed algorithm achieves optimal solutions by evaluating significantly fewer solutions compared to traditional metaheuristic algorithms. This efficiency makes our method particularly suitable for expensive optimization problems, where the time required to evaluate the quality of a solution by the computational model far exceeds the processing time of our algorithm. To demonstrate the capabilities of our algorithm in such contexts, we applied it to the problem of optimal design of bearings in an omnidirectional propeller [Rar]. This example illustrates how our method can effectively optimize complex, high-cost problems, highlighting its practical value and superior performance in scenarios where traditional methods would be prohibitively time-consuming.

The all-direction propeller has been widely used in drilling platforms and ships that require high positioning accuracy. In the design of a propeller, enhancing its vibration-resisting properties is crucial for improving service performance. Power flow serves as a dynamic index that reflects the vibration effect of the propeller, making it an important optimization objective to minimize for achieving optimal dynamic performance. However, calculating the propeller power flow necessitates computationally expensive finite element analysis simulations, rendering the acquisition of power flow a black-box problem. This complexity underscores the need for efficient optimization algorithms capable of handling such expensive evaluations to achieve the best design solutions. In general terms, the objective is to optimize the following problem:

$$\begin{aligned} &\text{Minimize } P(K) = \sqrt{\sum_{J=1}^{Z} P_j(K)^2};\ K = [k_1, k_2, k_3, k_4.k_5, c_1, c_2, c_3, c_4, c_5] \\ &P_j = \omega/2\pi \int_0^{\frac{\omega}{2\pi}} \mathrm{Re}\big(f_j(K,t)\big)\ \mathrm{Re}\big(v_j(K,t)\big)dt \\ &\text{Subject to } k_i^L \le k_i \le k_i^U;\ c_i^L \le c_i \le c_i^U;\ i = 1, \ldots, 5. \end{aligned} \tag{7}$$

where k_i and c_i represent the stiffness and damping coefficients of different bearings in the propeller, respectively. The ranges for these coefficients are denoted by k_i^L to k_i^U for stiffness and c_i^L to c_i^U for damping. P_j indicates the power flow at the selected evaluation point j in the propeller, and Z is the total number of evaluation points distributed across different bearings, set to 10 in this study. The terms $f_j(K,t)$ and $v_j(K,t)$ denote the force and velocity at the j-th evaluation point. The power flow is calculated as the sum of the product of force and speed over a cycle, as defined by Eq. (1). The finite element model (FEM) is employed to compute these elements. Evaluating completely each solution K in the computational model takes about 5 min. Further details on power flow calculation and its application in vibration control can be found in reference[34].

Figure 11 presents the execution times in hours for each optimization method during the minimization process. These times represent the average durations obtained from 10 independent runs, ensuring that the effect of randomness is minimized. Each algorithm was executed with a population of 50 individuals (N). This averaging approach provides a reliable comparison of the computational efficiency of the different methods under consistent conditions.

The results presented in Fig. 11 demonstrate that the proposed RBFNN-A method solves the problem described in Eq. (1) in less than 10 h, whereas the other methods require slightly more than a day (24 h) to achieve the same objective. This significant reduction in optimization time highlights the efficiency of the proposed method in handling expensive functions. Unlike classical metaheuristic methods, which need a larger number of candidate solutions to converge to the final solution, the RBFNN-A method achieves optimization with fewer evaluations and in a much shorter time frame. This efficiency makes it particularly advantageous for applications involving expensive computational models.

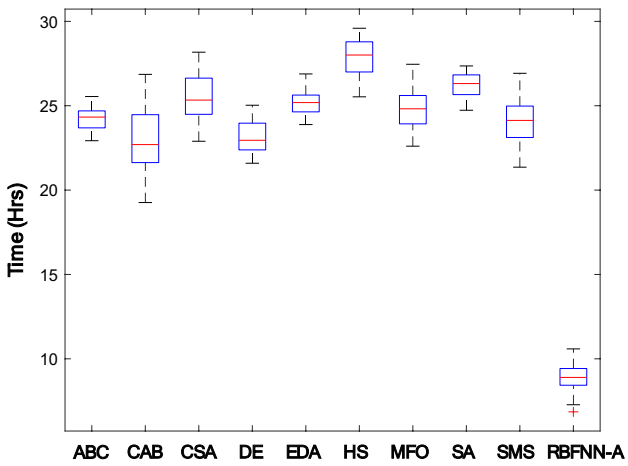

Figure 11. Execution times in hours for each optimization method during the minimization process of the problem "optimal design of bearings in an omnidirectional propeller" defined in Ec. 1.

Conclusions

This paper presents a novel optimization algorithm that employs the capabilities of a radial basis function neural network (RBFNN-A) to guide the search process. This algorithm begins by utilizing the maximum design approach to initialize a set of solutions that comprehensively cover the entire search space. Subsequently, the algorithm enters a cyclical phase, in which the RBFNN is trained to model the objective function values based on the current set of solutions. The algorithm identifies the neurons in the hidden layer corresponding to the highest values of the objective function and utilizes the parameters of these crucial neurons, such as their centroids and standard deviations, to generate new solutions by sampling in the order of importance until the typical number of solutions is reached. By focusing on the promising regions of the search space that generate a high value of the objective function, the algorithm avoids the exhaustive evaluation of solutions and enables a more informed exploration of the search space.

Our proposed algorithm has been evaluated by using a detailed array of numerical and statistical tests. To ensure a comprehensive assessment, we employed a wide-ranging collection of 29 benchmark functions that include unimodal, multimodal, and hybrid types. In assessing our method's performance, we compared its results against those from a variety of established metaheuristic algorithms. This selection spans a broad spectrum of optimization approaches, from classic to cutting-edge techniques, thereby encapsulating a diverse range of methodologies. The comparative study features several prominent algorithms, including the Artificial Bee Colony (ABC), Crow Search Algorithm (CSA), Differential Evolution (DE), Estimation of Distribution Algorithm (EDA), Moth-Flame Optimization (MFO), Harmony Search (HS), Simulated Annealing (SA), and State of Matter Search (SMS). This array of competing algorithms allows for a nuanced understanding of our proposed method's position within the broader context of metaheuristic optimization techniques.

The experimental findings reveal that our proposed algorithm exhibits superior efficiency in exploring the search space and identifying high-quality solutions. This enhanced performance is attributed to the algorithm's ability to recognize promising zones within the search space encoded in the RBFNN. By considering this knowledge, the algorithm efficiently allocates a relatively small number of iterations to identify the optimal solution. This strategic focus on areas of the search space with high potential allows for a more directed and effective search process, significantly reducing the time and computational resources required to achieve optimal outcomes. Consequently, the algorithm not only accelerates the optimization process but also improves the overall quality of the solutions found, distinguishing it from other approaches in its ability to swiftly navigate towards the most promising solutions.

The analysis of the computational time demonstrated that, although the proposed method is efficient in terms of the number of evaluations required, it demands more computational time due to the complexity of its solution generation processes. This indicates that the algorithm invests significant computational resources in generating high-quality solutions, resulting in longer processing times. Despite this, the method's efficiency in finding accurate solutions with fewer evaluations underscores its effectiveness, even though it necessitates greater computational effort. This trade-off highlights the method's robustness and precision at the cost of increased computational time.

Data availability

All experimental results have been included in the manuscript. The data can be made available upon reasonable request to the corresponding author. All experimental results have been included in the manuscript. The data can be made available upon reasonable request to the corresponding author.

Received: 12 February 2024; Accepted: 15 July 2024
Published online: 22 July 2024

References

1. Floudas, C. A. & Pardalos, P. M. *Recent advances in global optimization* (Princeton University Press, 2014).
2. Yang, Z., Dai, Y., Wang, S. & Yu, J. How to make lithium iron phosphate better: A review exploring classical modification approaches in-depth and proposing future optimization methods. *Journal of Materials Chemistry A* **4**(47), 18210–18222 (2016).
3. Kolda, T. G., Lewis, R. M. & Torczon, V. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.* **45**(3), 385–482 (2003).
4. Demidova, L. A. & Gorchakov, A. V. Research and study of the hybrid algorithms based on the collective behavior of fish schools and classical optimization methods. *Algorithms* **13**(4), 85 (2020).
5. Bandaru, S. & Deb, K. Metaheuristic techniques. In *Decision sciences* 693–750 (CRC Press, 2016).
6. Abdel-Basset, M., Abdel-Fatah, L. & Sangaiah, A. K. Metaheuristic algorithms: A comprehensive review. In *Computational intelligence for multimedia big data on the cloud with engineering applications* 185–231 (Elsevier, 2018).
7. Bennis, F. & Bhattacharjya, R. K. (eds) *Nature-Inspired Methods for Metaheuristics Optimization: Algorithms and Applications in Science and Engineering* Vol. 16 (Springer Nature, 2020).
8. Cuevas, E. *et al.* An introduction to nature-inspired metaheuristics and swarm methods. In *New Advancements in Swarm Algorithms: Operators and Applications* 1–41 (Springer, 2020).
9. Kuo, R. J. & Zulvia, F. E. The gradient evolution algorithm: A new metaheuristic. *Inf. Sci.* **316**, 246–265 (2015).
10. Malik, H. *et al.* (eds) *Metaheuristic and evolutionary computation: Algorithms and applications* Vol. 916 (Springer, 2021).
11. Booker, L. B., Goldberg, D. E. & Holland, J. H. Classifier systems and genetic algorithms. *Artificial intelligence* **40**(1–3), 235–282 (1989).
12. Askarzadeh, A. & dos Santos Coelho, L. Determination of photovoltaic modules parameters at different operating conditions using a novel bird mating optimizer approach. *Energy Conversion and Management* **89**, 608–614 (2015).
13. Schwefel, H. P. & Rudolph, G. Contemporary evolution strategies. In *European conference on artificial life* 891–907 (Springer, 1995).
14. Sallam, K. M., Elsayed, S. M., Chakrabortty, R. K., & Ryan, M. J. (2020) Improved multi-operator differential evolution algorithm for solving unconstrained problems. In 2020 IEEE congress on evolutionary computation (CEC), pp. 1–8. IEEE.
15. Chakraborty, A. & Kar, A. K. Swarm intelligence: A review of algorithms. In *Nature-inspired computing and optimization: Theory and applications* 475–494 (Springer, 2017).

16. Karaboga, D. & Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of global optimization* **39**, 459–471 (2007).
17. Dorigo, M. & Stützle, T. *Ant colony optimization: overview and recent advances* 311–351 (Springer International Publishing, 2019).
18. Kennedy, J., & Eberhart, R. (1995) Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks, Vol. 4, pp. 1942–1948. IEEE.
19. Xie, L. *et al.* Tuna swarm optimization: A novel swarm-based metaheuristic algorithm for global optimization. *Comput. Intell. Neurosci.* **2021**, 1–22 (2021).
20. Pozna, C., Precup, R. E., Horváth, E. & Petriu, E. M. Hybrid particle filter–particle swarm optimization algorithm and application to fuzzy controlled servo systems. *IEEE Trans. Fuzzy Syst.* **30**(10), 4286–4297 (2022).
21. Salcedo-Sanz, S. Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. *Phys. Rep.* **655**, 1–70 (2016).
22. Rashedi, E., Nezamabadi-Pour, H. & Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **179**(13), 2232–2248 (2009).
23. Kaveh, A. & Bakhshpoori, T. Water evaporation optimization: A novel physically inspired optimization algorithm. *Comput. Struct.* **167**, 69–85 (2016).
24. Abualigah, L. *et al.* Black hole algorithm: A comprehensive survey. *Appl. Intell.* **52**(10), 11892–11915 (2022).
25. Morales-Castañeda, B., Zaldivar, D., Cuevas, E., Fausto, F. & Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist?. *Swarm Evolut. Comput.* **54**, 100671 (2020).
26. Wang, G. G. & Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **49**(2), 542–555 (2017).
27. Santana-Quintero, L. V., Montano, A. A. & Coello, C. A. C. A review of techniques for handling expensive functions in evolutionary multi-objective optimization. In *Computational intelligence in expensive optimization problems* 29–59 (Springer, 2010).
28. Bringmann, K., Friedrich, T., Neumann, F., & Wagner, M. (2011) Approximation-guided evolutionary multi-objective optimization. In IJCAI Proceedings-International Joint Conference on Artificial Intelligence, Vol. 22, No. 1, p. 1198.
29. Bors, A. G. & Pitas, I. Median radial basis function neural network. *IEEE Trans. Neural Netw.* **7**(6), 1351–1364 (1996).
30. McGarry, K. J., Tait, J., Wermter, S., & MacIntyre, J. (1999) Rule-extraction from radial basis function networks.
31. McGarry, K. J., & MacIntyre, J. (1999) Knowledge extraction and insertion from radial basis function networks.
32. Joseph, V. R., Gul, E. & Ba, S. Maximum projection designs for computer experiments. *Biometrika* **102**(2), 371–380 (2015).
33. Gramacy, R. B. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences* (CRC Press, 2020).
34. Cai, X., Gao, L. & Li, F. Sequential approximation optimization assisted particle swarm optimization for expensive problems. *Appl. Soft Comput.* **83**, 105659 (2019).
35. Tenne, Y. A computational intelligence algorithm for expensive engineering optimization problems. *Eng. Appl. Artif. Intell.* **25**(5), 1009–1021 (2012).
36. Liu, B., Koziel, S. & Zhang, Q. A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems. *J. Comput. Sci.* **12**, 28–37 (2016).
37. Lee, C. C., Chung, P. C., Tsai, J. R. & Chang, C. I. Robust radial basis function neural networks. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **29**(6), 674–685 (1999).
38. Zhao, Q., Hautamaki, V. & Fränti, P. Knee point detection in BIC for detecting the number of clusters. In *International conference on advanced concepts for intelligent vision systems* 664–673 (Springer, 2008).
39. Zhao, Q., Xu, M., & Fränti, P. (2008) Knee point detection on bayesian information criterion. In 2008 20th IEEE international conference on tools with artificial intelligence, Vol. 2, pp. 431–438. IEEE.
40. Zolghadr-Asli, B., Bozorg-Haddad, O. & Chu, X. Crow search algorithm (CSA). In *Advanced optimization by nature-inspired algorithms* 143–149 (Springer, 2018).
41. Lampinen, J. A., Price, K. V. & Storn, R. M. *Differential evolution* (Springer-Verlag, 2005).
42. Pelikan, M., Hauschild, M. W. & Lobo, F. G. *Estimation of distribution algorithms* 899–928 (Springer handbook of computational intelligence, 2015).
43. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-based systems* **89**, 228–249 (2015).
44. Geem, Z. W. Harmony search applications in industry. In *Soft Computing Applications in Industry* 117–134 (Springer, 2008).
45. Bertsimas, D. & Tsitsiklis, J. A novel evolutionary algorithm inspired by the states of matter for template matching. *Exp. Syst. Appl.* **8**(1), 10–15 (1993).
46. Cuevas, E., Echavarría, A., Zaldívar, D. & Pérez-Cisneros, M. A novel evolutionary algorithm inspired by the states of matter for template matching. *Exp. Syst. Appl.* **40**(16), 6359–6373 (2013).
47. Oyeka, I. C. A. & Ebuh, G. U. Modified Wilcoxon signed-rank test. *Open J. Stat.* **2**(2), 172–176 (2012).
48. Taheri, S. M. & Hesamian, G. A generalization of the Wilcoxon signed-rank test and its applications. *Stat. Papers* **54**, 457–470 (2013).
49. Gutjahr, W. J. Convergence analysis of metaheuristics. In *Matheuristics: hybridizing metaheuristics and mathematical programming* 159–187 (Springer, 2009).
50. Biedrzycki, R., Arabas, J., & Warchulski, E. (2022) A version of NL-SHADE-RSP algorithm with midpoint for CEC 2022 single objective bound constrained problems. In 2022 IEEE congress on evolutionary computation (CEC), pp. 1–8. IEEE.
51. Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* **11**(1), 86–92 (1940).
52. Yildirim, A. E., & Karci, A. (2018). Application of three bar truss problem among engineering design optimization problems using artificial atom algorithm. In 2018 International conference on artificial intelligence and data processing (IDAP), pp. 1–5. IEEE.
53. Hassan, S., Kumar, K., Raj, C. D. & Sridhar, K. Design and optimisation of pressure vessel using metaheuristic approach. *Appl. Mech. Mater.* **465**, 401–406 (2014).
54. Kamil, A. T., Saleh, H. M. & Abd-Alla, I. H. A multi-swarm structure for particle swarm optimization: Solving the welded beam design problem. *J. Phys. Conf. Ser.* **1804**(1), 012012 (2021).

Author contributions

E.C. write the draft and the research.C.R. conducts the experiments.M.P. reviews the manuscriptB.M. programed all codes and routines.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-67778-0.

Correspondence and requests for materials should be addressed to E.C.