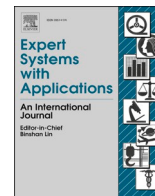




Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.



Self-adaptive moth flame optimizer combined with crossover operator and Fibonacci search strategy for COVID-19 CT image segmentation

Saroj Kumar Sahoo^a, Essam H. Houssein^{b,*}, M. Premkumar^c, Apu Kumar Saha^{a,*}, Marwa M. Emam^b

^a Department of Mathematics, National Institute of Technology Agartala, Tripura 799046, India

^b Faculty of Computers and Information, Minia University, Minia, Egypt

^c Department of Electrical and Electronics Engineering, Dayananda Sagar College of Engineering, Bengaluru, Karnataka 560078, India

ARTICLE INFO

Keywords:

Moth flame optimization algorithm
Metaheuristics
COVID-19 CT images
Multilevel thresholding
Image segmentation

ABSTRACT

The COVID-19 is one of the most significant obstacles that humanity is now facing. The use of computed tomography (CT) images is one method that can be utilized to recognize COVID-19 in early stage. In this study, an upgraded variant of Moth flame optimization algorithm (Es-MFO) is presented by considering a nonlinear self-adaptive parameter and a mathematical principle based on the Fibonacci approach method to achieve a higher level of accuracy in the classification of COVID-19 CT images. The proposed Es-MFO algorithm is evaluated using nineteen different basic benchmark functions, thirty and fifty dimensional IEEE CEC'2017 test functions, and compared the proficiency with a variety of other fundamental optimization techniques as well as MFO variants. Moreover, the suggested Es-MFO algorithm's robustness and durability has been evaluated with tests including the Friedman rank test and the Wilcoxon rank test, as well as a convergence analysis and a diversity analysis. Furthermore, the proposed Es-MFO algorithm resolves three CEC2020 engineering design problems to examine the problem-solving ability of the proposed method. The proposed Es-MFO algorithm is then used to solve the COVID-19 CT image segmentation problem using multi-level thresholding with the help of Otsu's method. Comparison results of the suggested Es-MFO with basic and MFO variants proved the superiority of the newly developed algorithm.

1. Introduction

COVID-19 is a highly contagious and severe disease that has spread rapidly across the world (Ferrer, 2020). The disease has caused numerous deaths and respiratory complications, including COVID-19 pneumonia, acute respiratory distress syndrome (ARDS), and acute respiratory failure. The World Health Organization (WHO) declared the outbreak a global pandemic on March 11, 2020 (Sohrabi et al., 2020), stressing the need for a global effort to combat the disease and reduce its impact on healthcare systems. One of the challenges in managing COVID-19 is the need for quick and accurate diagnostic technologies. The real-time polymerase chain reaction (PCR) test is commonly used to measure gene expression. However, it can produce false-negative results, is invasive, and takes a long time to diagnose. Chest computed tomography (CT) is another critical diagnostic tool for COVID-19. CT scans can guide the diagnosis and track the progression of the disease, making it a valuable technique for treating COVID-19-related lung

disease (Harmon et al., 2020). Preliminary studies have shown that chest CT is highly sensitive in detecting lung disease associated with COVID-19.

In various computer vision applications, including medical and geographical imaging, autonomous target recognition, and robotic vision, image segmentation is considered a critical and fundamental step in analyzing and interpreting captured images (Houssein et al., 2022). Medical imaging methods play a crucial role in diagnosing and treating severe illnesses and patient care. These techniques aid doctors in identifying, treating, and detecting life-threatening diseases at an early stage. Chest CT images contain a wealth of information, but manual processing is prone to errors. Therefore, numerous algorithms have been developed to assist in identifying and diagnosing COVID-19. Image segmentation is a practical approach that has been employed to improve the COVID-19 detection process (Ilhan et al., 2023; Qi et al., 2022). Image segmentation techniques have gained significant interest due to their versatile applications as a pre-processing phase in image

* Corresponding authors.

E-mail addresses: essam.halim@mu.edu.eg (E.H. Houssein), apusaha_nita@yahoo.co.in (A. Kumar Saha), marwa.khalef@mu.edu.eg (M.M. Emam).

<https://doi.org/10.1016/j.eswa.2023.120367>

Received 23 January 2023; Received in revised form 15 April 2023; Accepted 1 May 2023

Available online 6 May 2023

0957-4174/© 2023 Elsevier Ltd. All rights reserved.

processing. Particularly, in designing computer-aided diagnosis systems, image segmentation is a vital stage and is considered a crucial step in image processing. There are various ways to segment an image into distinct parts, and the most popular method is called thresholding segmentation. Thresholding segmentation, one of several popular segmentation algorithms, uses a thresholding value to separate a picture into many regions that are visually similar in terms of texture, colour, brightness, contrast, and size (s) (Houssein, Emam, & Ali, 2021a). The thresholding technique is widely used because it is simple to apply, requires minimal storage space, and is quick to execute. It includes two types of segmentation: bilevel and multilevel (Emam et al., 2023). Applying the bi-level threshold to an image produces two equally sized halves, one each for the foreground and background. Images in practical contexts unfortunately have more than two classes, hence multilevel thresholding is required. Multilevel thresholding methods are a type of image segmentation technique that divide an image into more than two regions based on the histogram of pixel intensities. This approach is useful when there are multiple objects or features in an image that need to be separated. However, choosing the right threshold values is essential to obtain accurate segmentation results. It is a critical task because the number of possible thresholds for an image is enormous, and selecting the optimal values requires careful consideration. Multilevel thresholding is a common technique used for image segmentation, especially for grayscale images like CT scans. It is particularly useful when there are distinct regions of interest within an image with different pixel intensities. COVID-19 CT images typically exhibit multiple regions of interest with varying pixel intensities, making the segmentation problem a multilevel one. In other words, it is necessary to identify multiple thresholds that can accurately distinguish between different regions of interest within the image.

There are two main methods for determining the optimal threshold value for image segmentation: Otsu's method (Otsu, 1979), which maximizes between-class variance, and Kapur's entropy (Kapur et al., 1985), which maximizes the entropy of the classes. Tsallis entropy (Tsai, 1985) is another method that can be used. These approaches are useful when only one threshold value is needed. However, these methods have significant drawbacks when dealing with multi-level thresholding, including long processing times and high complexity. Therefore, multi-level thresholding is considered a challenging optimization problem, and metaheuristic algorithms have been successfully employed to address these issues.

Overall, research has shown that metaheuristic algorithms are effective in solving difficult optimization problems in various fields, including bioinformatics, engineering, communication, drug design, and feature selection (Houssein et al., 2023). In recent years, it has been seen those real-world problems, constrained or unconstrained, linear or nonlinear, continuous or discontinuous, can be easily tackled with the help of various nature-inspired algorithms (Nama & Saha, 2019). These algorithms are prevalent due to their simplicity and user-friendly approach and play a significant role in tackling complicated optimization issues (Nama & Saha, 2018). Some of these algorithms have been developed by considering natural evolution, living and survival systems of birds, animals and insects, etc. A few of them are Genetic algorithm (GA) (Holland, 1992), Particle swarm optimization (PSO) (Kennedy & Eberhart, 1995; Cheng & Prayogo, 2014; Arora & Singh, 2015), Differential evolution (DE) (Storn & Price, 1997), Butterfly optimization algorithm (BOA; Arora & Singh, 2015), Moth-flame optimization algorithm (MFO) (Mirjalili, 2015), whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016), Jaya algorithm (Rao, 2016), Sine cosine algorithm (SCA) (Mirjalili, 2016a), Salp swarm algorithm (SSA) (Mirjalili et al., 2017). In addition to these old algorithms there are many newer ones like arithmetic optimization algorithm (AOA) (Abualigah et al., 2021; Heidari et al., 2019)

Furthermore, several excellent algorithms have been developed between 2020 and 2022, such as weighted mean of vectors (INFO) (Ahmadianfar et al., 2022) is an optimization algorithm developed in

2022, Ebola Optimization Search Algorithm (EOSA) (Oyelade et al., 2022), and Dwarf mongoose optimization algorithm (DMO) (Agushaka et al., 2022). In 2023, there are some new optimization algorithms such as, Rime optimization algorithm (RIME) (Su et al., 2023) and Nutcracker optimization algorithm (NOA) (Abdel-Basset et al., 2023).

Metaheuristic algorithms have both strengths and weaknesses. On the positive side, they are versatile and can handle non-linear, higher-dimensional, and multimodal situations. They are also straightforward to design and implement, even without knowledge of the aims' derivatives. However, some drawbacks exist, such as the need for a balance between exploration and exploitation. This can lead to issues like becoming trapped in local optima, slow convergence, and loss of diversity. Moreover, no single metaheuristic algorithm is universally effective, and researchers worldwide are developing many algorithms, hybrids, and modified variants. The development of metaheuristic algorithms depends on their ability to explore and exploit solutions effectively. The No Free Lunch theory (Wolpert & Macready, 1997) demonstrates that no algorithm can solve all problems optimally. Researchers have proposed two strategies to overcome these limitations: modifying existing algorithms or hybridizing multiple metaheuristics. Hybridization can enhance optimization performance, but it is essential to choose appropriate algorithms. Therefore, selecting algorithms is a crucial step; typically, they are chosen based on their performance. One way to enhance algorithm performance is to incorporate optimization components into the original algorithm. As a result, this encourages us to improve the MFO algorithm and apply it to solve the multilevel image segmentation problem.

An efficient algorithm named MFO, which was first developed by Mirjalili (Mirjalili, 2015) in 2015, is considered for deep study, and the transverse orientation of moths motivates authors for MFO's design. The MFO algorithm has attracted the attention of many researchers. MFO is a versatile algorithm with minimal algorithm-specific parameters, making it suitable for real-world problems. For example, it has been applied successfully to tasks such as parameter estimation for solar modules (Sharma et al., 2022), flexible operation modeling (Hou et al., 2022), intelligent route planning for multiple UAVs (Ma et al., 2022), deep learning (Khan et al., 2022), machine scheduling problems (Mohd Rose & Nik Mohamed, 2022), neural network optimization (Ramachandran et al., 2021), and many more. The MFO is a promising new population-based optimization method. However, it still has room for development in areas like accelerating convergence and expanding the scope of its search (Khalilpourazari & Khalilpourazary, 2019). In recent years, several variants of the Moth-Flame Optimization (MFO) algorithm have been proposed to overcome its weaknesses and improve its performance. The following is a summary of various modified versions of the MFO algorithm. Zhao et al. (2022a) proposed the Multiswarm Improved MFO (MIMFO) algorithm that incorporates chaotic and dynamic grouping mechanisms to enhance population diversity. They also added linear and spiral search strategies and Gaussian mutation to improve search capabilities and maintain a balance between diversification and intensification. Similarly, Sahoo, Saha, Nama, and Masdari (2022a) introduced a variant of MFO named m-DMFO, which uses a modified dynamic opposition learning (DOL) strategy to speed up convergence and prevent stagnation in local optima. Other approaches include covariance based MFO (CCMFO) by Zhao, Fang, Liu, Xu, and Li (2022b) that utilizes covariance and Cauchy mutation to upgrade location updates and enhance exploration, and multi-operator MFO (MOMFO) by Gu and Xiang (2021) that uses various strategies to balance global and local search. Moreover, Nadimi-Shahraki, Fatahi, Zamani, Mirjalili, Abualigah, and Abd Elaziz (2021a) developed the Migration-based MFO (M-MFO) to balance the exploration-exploitation characteristics of the classical MFO by using a random and guided migration operator. Li et al. (2021) created the ODSMFO method with the help of the OBL mechanism and DE, as well as a developed local search technique and the death mechanism for diversity enhancement. Shan et al. (2021) showed that the MFO algorithm might be stabilized using the Double Adaptive

Weight Mechanism (WEMFO) and tested it by utilizing it to train Kernel Extreme Learning Machines (KELMs).

Sahoo et al. (2022b) developed the upgraded MFO (EMFO) by embedding the mutualism strategy on the basic MFO for a better balance between the search processes by enhancing its searching capability. Nadimi-Shahraki, Fatahi, Zamani, Mirjalili, and Abualigah (2021b) created the improved MFO (I-MFO) that assists in locating trapped moths in local optima by defining memory for each moth. While Pelusi et al. (2020) proposed the improved MFO (IMFO) by addressing the MFO's weaknesses, such as the rate of convergence and inclusive searchability, by introducing a weight component to the suggested IMFO to maintain the search balance.

Kigsirisin and Miyauchi (2021) proposed a solution to handle the challenges in unit commitment (UC) using a modified version of the MFO algorithm called alternative binary MFO. However, this approach suffered from a flaw due to a fixed flame technique that caused the algorithm to become trapped at a local optimum. To address this issue, the authors developed four new techniques called BAMFO. These techniques were used to develop a plan to repair the UC. Sahoo and Saha (2022c) introduced an improved variant of the MFO algorithm that incorporated both global and local phases of BOA to strike a balance between diversification and intensification. Meanwhile, Nadimi-Shahraki et al. (2021c) proposed a discrete MFO for community discovery by recreating its successful tactics. This approach utilized a locus-based adjacency formulation of moths and flames to analyze node relationships and community organization during startup. An updated movement strategy was employed using single-point crossover, two-point crossover, and single-point neighbor-based mutation to balance exploration-exploitation. Dabba et al. (2021) developed the mutual information maximization-modified moth flame method (MIM-MFA) to address gene assortment in microarray data sorting with MIM. Kadry et al. (2021) used a modified version of Kapur's MFO algorithm and his threshold image segmentation to eliminate the tumor area from flair and T2 clinical MRI slices. The authors used images from the BRAINIX and TCIA-GBM benchmark sets to evaluate the suggested approach. Their experimental results revealed that the T2 modality was subpar. Sapre and Mini (2021) employed the differential MFO (DMFO) to alleviate challenges inherent in WSNs with a mobile sink. Dash et al. (2020) integrated the Jaya-based MFO and the original MFO to minimize the impact of FACTS devices on network performance in an IEEE network. The two compensators, TCSC and SVC, served as fitness functions for the JMFO and MFO algorithms in the IEEE 14 and IEEE 30 bus systems.

Apinantanakon and Sunat (2017) developed a new variant of the MFO algorithm called OMFO to address its drawbacks, such as sluggish convergence and mediocre solutions. They employed an opposition-based strategy to produce new moths in MFO and subsequently used those in the position upgrade process for rapid convergence. Li et al. (2018) developed the double-evolutionary learning MFO algorithm (DELMFO) by combining two evolutionary learning strategies to create high-performance flames and dynamically regulate the hunt for moths. Zhao et al. (2020) presented the boosted MFO algorithm, a refined version of a population-based method. Both the MFO method and the OBL strategy for creating flames were modernized using the mutation mechanism and linear quest strategy. For further details on the MFO works and their variants, one may refer to the survey work on MFO (Sahoo, Saha, & Ezugwu, 2022d).

In this paper, we suggest a modified form of the MFO algorithm (Es-MFO, for short) that uses an upgraded solution strategy to address the problems with metaheuristics techniques mentioned above. The proposed Es-MFO is used to obtain the best solution for determining the optimal thresholding that devastates the multi-level thresholding image segmentation for CT COVID-19 images. The experimental section of this paper is particularly robust, with tests of not just the diversity, effect of parameters, statistical analysis, and normal engineering test issues, but also comparisons to other popular algorithms. It has also been used to solve the segmentation problem in CT images for the COVID-19. The

following is a brief summary of the main contribution of this paper:

A crossover operator is employed in MFO algorithm to enhance the searching efficacy of the suggested Es-MFO algorithm.

A mathematical approach-based technique is employed at the end of the position update phase of the MFO, which avoids the local optimal solutions and improves the convergence speed of the proposed Es-MFO algorithm.

The proposed Es-MFO has been applied to solve IEEE CEC'2017 test suite for dimensions 30 and 50 and the results are compared with six well-known metaheuristics and three MFO variants.

Statistical tests like the Friedman rank test and the Wilcoxon signed-rank test are also used to measure how well the proposed algorithm works.

To test the proposed problem-solving Es-MFO's capability, it was used to solve three CEC 2020 limited real-world engineering problems.

At the end, Es-MFO has been applied to solve the COVID-19 CT image segmentation problem.

The article is laid out as follows: in Section 2, relevant works on the image segmentation issues are given. The MFO method is summarized in Section 3. Section 4 demonstrates the proposed Es-MFO algorithm. In Section 5, we give the experimental designs, simulation outcomes, statistical tests, and convergence analyses. Section 6 presents the engineering design challenge, Section 7 presents the COVID-19 CT image segmentation problem, and Section 8 discusses the conclusions.

2. Related works

Image segmentation methods have recently received much attention and are often used as a preprocessing step in various image processing applications. Numerous methods are available for solving the image segmentation problem, but multilevel thresholding segmentation is considered the best. However, traditional techniques need help solving the image segmentation problem as the threshold levels increase due to time complexity issues. Metaheuristic algorithms have been employed to overcome these problems and have proven efficient and useful in the relevant literature. Metaheuristic algorithms have been successfully used in medical imaging segmentation, including in the segmentation of COVID-19 medical imaging. This section provides some state-of-the-art techniques for COVID-19 segmentation. Several metaheuristic-based multilevel thresholding segmentation methods have been employed to detect COVID-19 infections in CT scans. One such approach is the Improved Manta Ray Foraging Optimization (MRFO) was created by (Houssein, Emam, & Ali, 2021b) by integrating the opposition-based learning (OBL) technique in the early phase of the MRFO algorithm to increase the population variety in the search space. The suggested algorithm is known by its abbreviation, MRFO-OBL. By using Otsu's approach to address the COVID-19 CT image segmentation problem, the effectiveness of the suggested MRFO-OBL algorithm was evaluated. The experimental results demonstrated that, in terms of performance metrics including the structural similarity index (SSIM), peak signal-to-noise ratio (PSNR), consistency, and quality, the suggested MRFO-OBL algorithm outperformed the other optimization algorithms.

Wang et al., (Wang et al., 2022) proposed a new hybrid algorithm for segmenting COVID-19 chest X-ray images by combining the PSO and firefly algorithm (FA). Multi-threshold segmentation technique based on two-dimensional reciprocal cross-entropy is suggested to solve the problem of undefined and zero values of Shannon cross entropy due to logarithm operation. The proposed algorithm was evaluated on a dataset of 300 chest X-ray images. The results showed that it outperformed several state-of-the-art algorithms regarding segmentation accuracy and computational time.

Elaziz et al. (Abd Elaziz et al., 2020) proposed a technique called MPAMFO that combines two swarm intelligence algorithms, MPA and

MFO, to achieve better results in image segmentation. MPAMFO uses MFO as a local search technique to prevent the algorithm from getting stuck in local optima. The method was tested on ten gray-scale images and thirteen CT images of COVID-19 and outperformed other swarm intelligence algorithms regarding segmentation quality. The experimental results showed that MPAMFO is a reliable and effective approach for image segmentation and has advantages over existing methods.

In another study, Houssein et al. (2022) developed a new improved equilibrium optimizer (I-EO) with the help of dimension learning hunting (DLH) technique. The efficiency of the suggested I-EO was tested by IEEE CEC 2020 test problems and COVID-19 CT image segmentation problem. The obtained experimental results outperform as compared to other metaheuristic algorithms.

Nama (2022) proposed a new population-based optimization algorithm called improved slime mould algorithm (in short QRSMA) by adding quasi-reflection-based learning (QRBL) and its jumping mechanism to improve the population diversity and accelerate the convergence speed, respectively. The author then applied the suggested QRSMA to solve the COVID-19 X-ray image segmentation problem. The experimental results demonstrated the superiority of the proposed QRSMA algorithm among other competitive metaheuristic algorithms.

Chakraborty and Mali (2021) suggested a modified version of the WOA called the Modified WOA algorithm that uses random initialization solutions in the search prey stage. A new parameter was developed to balance the exploitation and exploration of the algorithm. The algorithm was evaluated using three COVID-19 x-ray image segmentation problems and a few benchmark images. The experimental results demonstrated that the proposed algorithm outperformed other optimization algorithms in terms of several performance indicators.

Abualigah et al. (2022) proposed an upgraded version of the Ant Lion Optimizer (AOA) algorithm named DAOA by incorporating the Differential Evolution (DE) technique to enhance the local search of the algorithm. The algorithm was evaluated using a COVID-19 CT image segmentation problem, and its effectiveness was compared with other state-of-the-art algorithms in terms of SSIM and PSNR values.

Chakraborty et al. (2021) created a morphology-based radiological segmentation method for the early identification of probable COVID-19 patients. The method was evaluated using over 400 different photos, and the results showed that the proposed approach outperformed other optimization strategies in terms of SSIM, PSNR, and Mean Square Error (MSE).

Liu et al. (2021) proposed an improved version of the Ant Colony Optimization (ACO) algorithm called CLACO using Cauchy and greedy Levy mutation. The algorithm was evaluated using Kapur's entropy to solve the COVID-19 X-ray image segmentation problem after testing it on IEEE CEC 2014 benchmark functions.

Qi et al. (2022) used the directional mutation and directional crossover operator to create a fresh ACO algorithm. The COVID-19 X-ray image segmentation problem and the IEEE CEC 2017 benchmark functions were used by the author to test the suggested technique. By examining the experimental findings, the suggested algorithm displays more consistent and superior segmentation outcomes than existing models at various threshold values.

Moreover, A few additional studies have been done by scholars with the help of several unique meta-heuristic algorithms in (Chakraborty et al., 2022a; Houssein et al. 2021c, 2021d, 2021d; Sharma et al., 2021; Sharma et al., 2021) to address image segmentation issues.

The literature review mentioned some weaknesses in the segmentation of COVID-19 CT images. Metaheuristic techniques used in optimization problems have certain drawbacks, such as getting stuck in local areas, early termination, and requiring better global search capabilities. The studies reviewed on multilevel thresholding segmentation have shown that it is commonly done with metaheuristic algorithms and that the optimal thresholding set depends on the algorithm used. Therefore, using a high-performance metaheuristic algorithm can significantly improve the results of multilevel segmentation. These motivated us to

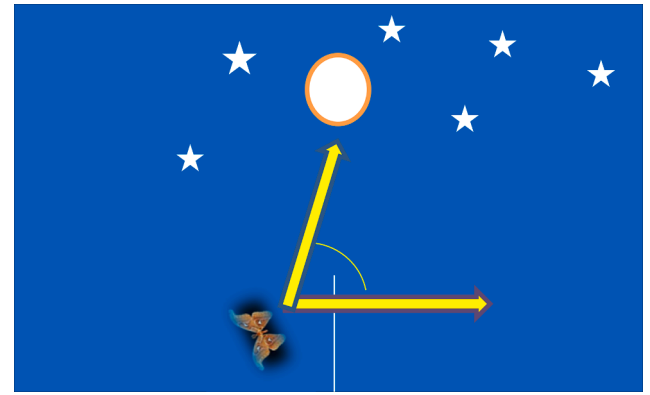


Fig. 1. Transverse orientation of Moth.

improve the diagnosis of COVID-19 by combining an effective optimization algorithm with Otsu's segmentation method for CT COVID-19 images.

3. Classical MFO algorithm

In this Section, Sections 3.1 and 3.2 explain where the MFO algorithm came from and how it works with the help of a mathematical formula.

3.1. Inspiration

Arthropoda includes moths, which are insects. Research into moths' unique navigation methods has piqued the interest of scientists. As shown in Fig. 1, moths use a transverse orientation mechanism to help them navigate at night. These moths use moonlight to fly across a horizontal inclination for a long distance in a straight line by keeping a fixed tendency toward the moon. The moth moves in a helix path around the flame as the distance between it and the flame decreases, increasing preference efficiency. As a result of his research into moth behaviour and mathematical modeling, Mirjalili created the MFO algorithm in 2015.

3.2. MFO algorithm

In MFO, the entire swarm consists of N moths; they are characterized as a group of candidate solutions to a specific problem by using a matrix as follows:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \ddots & \cdots & \cdots & x_{2,n} \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ x_{N-1,1} & \cdots & \cdots & \ddots & x_{N-1,n} \\ x_{N,1} & x_{N,2} & \cdots & x_{N,n-1} & x_{N,n} \end{bmatrix} \quad (1)$$

where the position of each moth is considered as a vector $X_i = [x_{i,1}, x_{i,2}, \dots, x_{i,n}]$, $i \in \{1, 2, \dots, N\}$.

while 'n' denotes variable numbers. The j^{th} dimension of each X_i is expressed as scalar $x_{i,j}$, $j \in \{1, 2, \dots, n\}$ in the boundary range $[x_{j,\min}, x_{j,\max}]$, where $x_{j,\min}$ and $x_{j,\max}$ are the minimum and maximum boundary of the j^{th} dimension of each X_i , respectively.

For $i \in \{1, 2, \dots, N\}$, we suppose that the corresponding fitness value of $X_i = [x_{i,1}, x_{i,2}, \dots, x_{i,n}]$ can be expressed as $Fit(X_i)$, where $Fit(*)$ represents fitness function candidate solution. Then, the corresponding fitness vector of X is represented as follows:

$$Fit[X] = \begin{bmatrix} Fit(X_1) \\ Fit(X_2) \\ \vdots \\ Fit(X_n) \end{bmatrix} \quad (2)$$

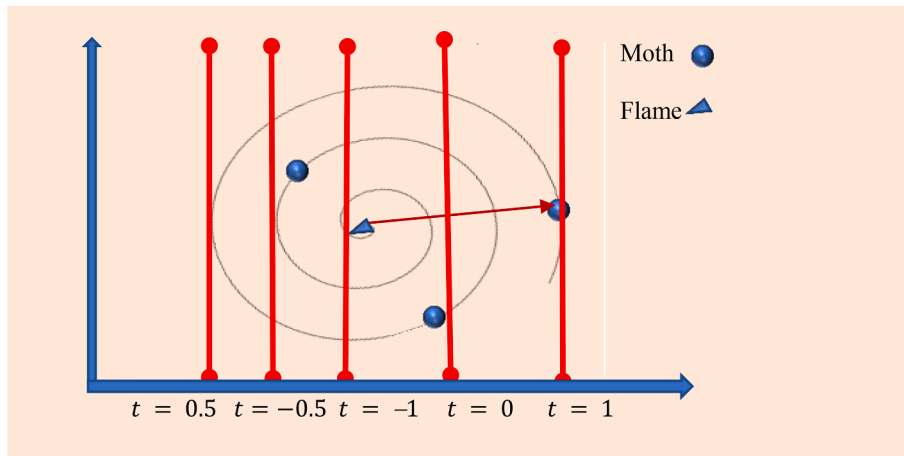


Fig. 2. Spiral movement around flame.

Each moth can fly toward its corresponding flame so that the flame matrix has the same size as the matrix X as follows:

$$FM = \begin{bmatrix} FM_1 \\ FM_2 \\ \vdots \\ FM_N \end{bmatrix} = \begin{bmatrix} Fm_{1,1} & Fm_{1,2} & \dots & Fm_{1,n-1} & Fm_{1,n} \\ Fm_{2,1} & \ddots & \dots & \dots & Fm_{2,n} \\ \vdots & \dots & \ddots & \dots & \vdots \\ Fm_{N-1,1} & \dots & \dots & \ddots & Fm_{N-1,n} \\ Fm_{N,1} & Fm_{N,2} & \dots & Fm_{N-1} & Fm_{N,n} \end{bmatrix} \quad (3)$$

where FM_i is defined as the flame corresponding to the $X_i = [x_{i,1}, x_{i,2}, \dots, x_{i,n}]$; n denotes variable number of FM_i , N is the number of flames.

Similarly, the fitness of each $FM_i = [Fm_{i,1}, Fm_{i,2}, \dots, Fm_{i,n}]$; $i = \{1, 2, \dots, N\}$, we also assume that its corresponding fitness value can be expressed as $Fit(FM_i)$, Further, the fitness vector of FMx is as follows:

$$Fit[FM] = \begin{bmatrix} Fit(FM_1) \\ Fit(FM_2) \\ \vdots \\ Fit(FM_N) \end{bmatrix} \quad (4)$$

Moth and flame are two important components of MFO algorithm. Suppose that K and k , $k \in \{1, 2, \dots, N\}$ indicate the maximum iterative number and the current iterative number, respectively. Moth moves spirally when it nearer to the flame therefore, author used a logarithmic spiral function which is as follows:

$$X_i^{k+1} = \begin{cases} \delta_i \bullet e^{bt} \bullet \cos(2\pi t) + FM_i(k), & i \leq Fl_{no} \\ \delta_i \bullet e^{bt} \bullet \cos(2\pi t) + FM_{Fl_{no}}(k), & i \geq Fl_{no} \end{cases} \quad (5)$$

where $\delta_i = |X_i^k - FM_i(k)|$, that indicates the distance between i^{th} moth X_i and its corresponding flame FM_i . Here b is a fixed constant equal one used to recognize the spiral flight shape, Fl_{no} represents flame number that has the ability to decrease the number of flames adaptively through the entire iterative process and Fig. 2 represents moths helix manner, t be any random number in $[a_1, 1]$ and is a_1 is a convergence constant which linearly decreased from (-1) to (-2) over the course of iteration and mathematically, it can be represented as follows:

$$a_1 = -1 + k \left(\frac{-1}{K} \right) \quad (6)$$

$$t = (a_1 - 1) \times r + 1 \quad (7)$$

where, r is a random number in $[0, 1]$,

Current and prior flame positions are gathered and organised by global and local fitness in every iteration. Other flames are extinguished, and only the best Fl_{no} are maintained (Li et al., 2018). The fittest flames are the first and last ones. The moths arrived to seize the flames. Moths

in the same and lower orders invariably take the final flame. The following formula is used for the determination of flame number (Fl_{no}).

$$Fl_{no} = \text{round} \left(Fl_{max} - k \frac{(Fl_{max} - 1)}{K} \right) \quad (8)$$

where, round can make the number of $(Fl_{max} - k \frac{(Fl_{max} - 1)}{K})$ be rounded to its nearest integer and Fl_{max} represents the maximum number of flames. Further k and K are represents the current iteration and maximum iterations of the population. The more detailed pseudocode of the MFO algorithm is presented in Algorithm 1.

Algorithm 1: Pseudo-code of MFO Algorithm

Input: Objective function $f(X)$, $X = (X_1 X_2 \dots X_d)$, Number of moths in the population (N), dimension (d), Current iteration (Iteration), Maximum iteration (max_{iter}), Flame number (Fl_{no}), Lower bound (LB), Upper bound (UB), $b = 1$ and other related parameters are determined.

```

for  $i = 1 : N$ 
  for  $j = 1 : d$ 
    Generate  $N$  organism solutions  $X_{ij}$  ( $i = 1, 2, \dots, N$ ) using following Equation
     $X(i, j) = LB(i) + (UB(i) - LB(i)) * rand()$ ; %  $rand()$  is a random number  $\in [0, 1]$ 
  end for
end for
Calculate fitness value  $f(X)$ 
While Iteration <  $max_{iter} + 1$ 
  if Iteration == 1
    Enter  $Fl_{no} = N$  in initial population
  else
    Apply Eq. (8)
  end if
  FM = Fitness Function  $f(X)$ 
  if Iteration == 1
    arrange the moths according to FM
    Update Flames
    Iteration = 0
  else
    Sort the moths based on FM from last iteration
  end if
  Update the Flames
end while
for  $j = 1 : N$ 
  for  $k = 1 : d$ 
    Find  $r$  and  $t$  using Eqs. (6 and 7)
    Update moths position as to their particular flame using Eq. (5)
  end for
end for
Iteration = Iteration + 1
end while

```

Output: The best solution with the minimum fitness function value in the ecosystem.

4. Proposed (Es-MFO) algorithm

This section presents the proposed Fibonacci search-based MFO algorithm (Es-MFO, in short). The adaptation of non-linear function with

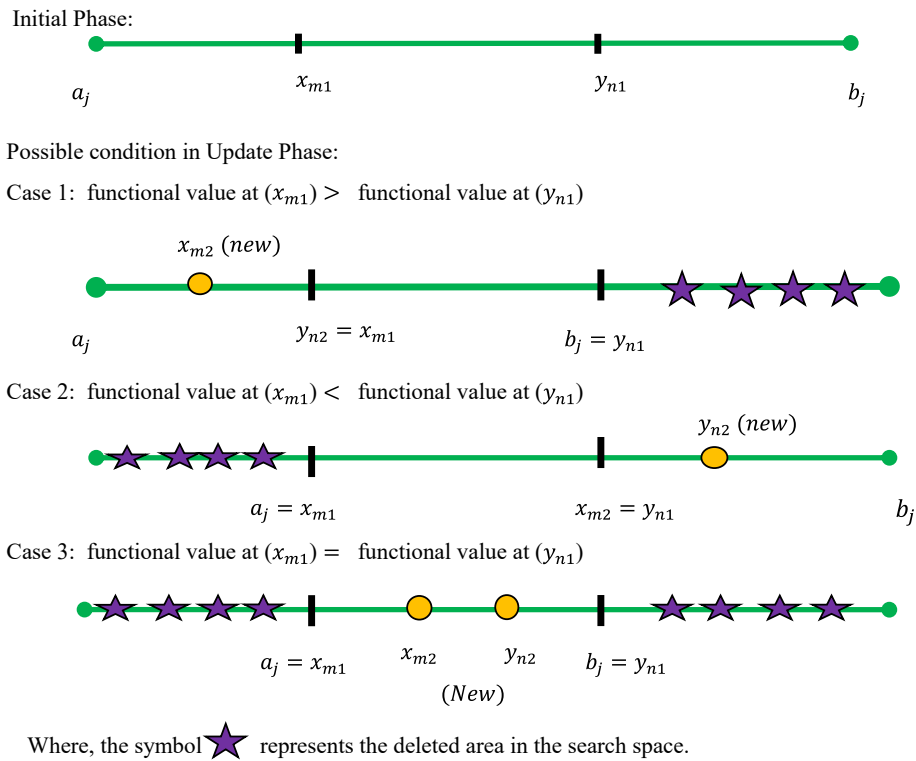


Fig. 3. The search mechanism of FSM.

self-adaptive factor is presented in Section 4.1. The improved solution quality technique is familiarized in Section 4.2. The two main characteristics of the metaheuristic algorithms include exploration and exploitation of the search space and a proper trade-off between them. Exploration involves searching the entire region, whereas exploitation is characterized as examining promising areas around a potential solution. The joint effect of these two characteristics contributes to the algorithm's ability to prevent local optima stagnation and promote convergence and solution variety. According to the literature, despite of its effectiveness, MFO suffers from low solution accuracy, sluggish convergence rate, lack of variety, and tendency to fall into local optimal solutions, i.e., the algorithm struggles to maintain a suitable balance between exploration and exploitation. To overcome the above issues, in the present work, an improved MFO, namely Es-MFO, has been proposed by integrating crossover operator and improved solution quality technique. The details are discussed in the following subsections.

4.1. Crossover operator adaption in MFO algorithm

In MFO, the spiral motion of moths around the flame results exploration and exploitation of the search space. It is easier to understand exploration and exploitation when the exponent factor 't' is used to explain it. If the value of 't' increases, then distance between moth and flame increases that means moth will cover larger distance (exploration) to reach at flame. Similarly, if 't' decreases, then distance between moth and flame decreases that means moth will cover smaller distance (exploitation) to reach at flame. Therefore, either a suitable value of 't' or an effective technique is necessary to handle the trade-off of the algorithm. In addition to the above-mentioned reason, we have added a different type of crossover (CR) operator strategy in the position update phase to provide an effective balance between the diversification and intensification. The following scheme is implemented to create the position at the next iteration:

$$X_i^{k+1} = \begin{cases} CR \cdot \delta_i \cdot e^{bt} \cdot \cos(2\pi t) + Fm_i(k), & i \leq Fl_{no} \\ \delta_i \cdot e^{bt} \cdot \cos(2\pi t) + CR \cdot Fm_{N.FM}(k), & i \geq Fl_{no} \end{cases} \quad (9)$$

where, CR is a non-linear function and defined by $CR = 2(0.5 - \mu) \cdot f$; μ is a random number uniformly distributed in $[1, N]$, 'N' is number of search agents in the population. $f = \exp(-12 \cdot (k/K))$; K and k, $k \in \{1, 2, \dots, N\}$ indicate the maximum iterative number and the current iterative number, respectively and $\delta_i = |X_i^K - Fm_i(k)|$ that indicates the distance between i^{th} moth X_i and its corresponding flame Fm_i , b is a fixed constant equal one used to recognize the spiral flight shape, Fl_{no} represents flame number that has the ability to decrease the number of flames adaptively through the entire iterative process t be any random number in $[a_1, 1]$ and is a_1 is a convergence constant which linearly decreased from -1 to -2 over the course of iteration and mathematically, it can be represented as follows:

$$t = (a_1 - 1) \cdot r + 1 \text{ and } a_1 = -1 + k \left(\frac{-1}{K} \right); r \text{ is a random number in } [0, 1].$$

4.2. Improved solution technique

An improved solution technique with the aid of the Fibonacci search method (FSM) and the average of three random solutions has been introduced in the suggested Es-MFO to improve the solution quality, avoid local optima stagnation, and accelerate the convergence speed.

4.3. Fibonacci search method

The Fibonacci search is the strategy that yields the smallest possible interval of uncertainty in which the optimal solution lies, after finite number of tests are completed (Pierre, 1986). The Fibonacci search is based on the sequence of Fibonacci numbers (Fib), which are shown by Eq. (10).

$$F_0 = 1 = F_1, F_k = F_{k-1} + F_{k-2}, k = 2, 3, 4, \dots, n \quad (10)$$

The FSM is an elimination technique and can also be said as the

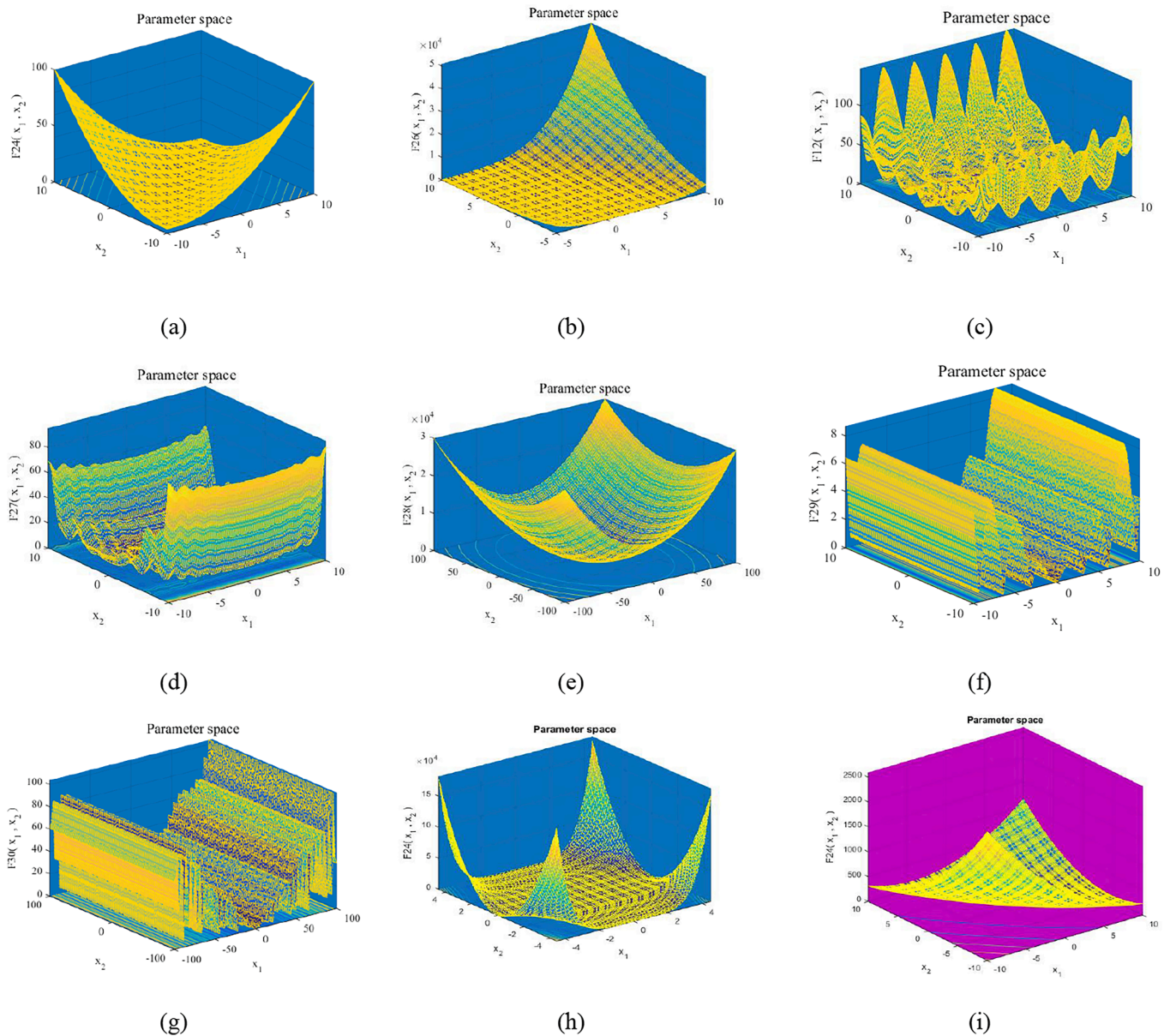


Fig. 4. 3D view of few randomly selected unimodal and multimodal functions: (a) Matyas (b) Zakhrov(c) Penalized 1.1 (d) Levy (e) Bohachevsky (f) Alpine and (g) Salomon (h)Beale (i) Booth.

interval reduction method, for solving for getting optimal for non-linear optimization problem. The FSM shifts and narrows the search range using Fibonacci numbers to obtain the extreme value of functions (Ramaprabha, 2012). To improve the balance of the search in the MFO algorithm, the proposed Es-MFO method incorporates the Fibonacci search method (FSM), which focuses on updating the current best solution obtained during the iterative process. The FSM ensures that the solutions are not trapped in local optima and promotes high diversification.

Let for any iteration T, for each moth (X_{ij}), $i = 1, 2, \dots, N$ and $j = 1, 2, \dots, D$, $X = (x_1, x_2, x_3, \dots, x_D)$ and $Y = (y_1, y_2, y_3, \dots, y_D)$ are two distinct search agents. Take two experimental members x_{m1} and y_{n1} from X and Y which are calculated as follows:

$$x_{m1} = a_j + \left(\frac{F_{k-2}}{F_k}\right)(b_j - a_j), y_{n1} = b_j - \left(\frac{F_{k-2}}{F_k}\right)(b_j - a_j) \tag{11}$$

where a_j and b_j are lower and upper bounds of i^{th} variable. The range is moved to the right if the function's value at y_{n1} is greater than that at

x_{m1} , and to the left if x_{m1} is greater than that at y_{n1} . The new value x_{m2} and y_{n2} are generated using the Fibonacci search formula as,

$$x_{m2} = a_j + \left(\frac{F_{k-j}}{F_{k-j+2}}\right)(b_j - a_j) \text{ and } y_{n2} = b_j - \left(\frac{F_{k-j}}{F_{k-j+2}}\right)(b_j - a_j) \tag{12}$$

where j represents a variable with an initial value 2 in the FSM. If there are two functional values that are not equal, then only one of them (x_{m2} or y_{n2} depending on the contracting direction) will be considered a new experimental point. If the two function values are equal, then the two new experimental points formed are x_{m2} or y_{n2} ; this process is repeated until the stopping criterion is met, and a new solution is obtained by averaging the last two experimental members in the final iteration. The various scenarios that may arise in FSM are depicted in Fig. 3.

The proposed Es- MFO's operational process is illustrated Algorithm 2 and the main Es-MFO steps are summarized below:

1. Randomly initialize parameters such as the number of populations, maximum iteration, and function evaluation.

2. Sort the moth and flame matrices based on fitness value and update the number of flames using Eq. (8).
3. Update a_1 and t by Eqs. (6) and (7).
4. Update moth positions based on the corresponding flame using Eq. (9).
5. To generate a new solution, randomly generate a number between 0 and 1. If the number is greater than 0.5, use Eq. (13). Otherwise, use the Fibonacci search method (FSM) with Eqs. (10), 11, and 12) to find the fitness value of the new solution. The best fitness value gives the optimum value.

$$X_{New} = e^{(X_{avg} / \sqrt{fit(x_1) * fit(x_2) * fit(x_3)})} \quad (13)$$

6. If the stopping criteria are not met, repeat steps 2-5 until the best fitness value is obtained.

Algorithm 2: Pseudo-code of the Es-MFO algorithm.

```

Input: Maximum iteration ( $max_{iter}$ ), Objective function  $f(X)$ ,  $X = (X_1 X_2 \dots X_d)$ ,
dimension ( $D$ ), Initial moth number ( $N$ ), Flame number ( $Fl_{no}$ ), Lower bound ( $LB$ ),
Upper bound ( $UB$ ),  $b$ , Current iteration ( $Iteration$ ), Maximum iterations, and other
related parameters are determined.
for  $i = 1 : N$ 
  for  $j = 1 : D$ 
    Generate  $N$  organism solutions  $X_{ij}$  ( $i = 1, 2, \dots, N$ ) using following Equation
     $X(i, j) = LB(i) + (UB(i) - LB(i)) * rand()$ ; %  $rand()$  is a random number  $\in [0, 1]$ 
  Find fitness
  end
end
While stopping criteria not met
if  $Iteration = 1$ 
   $Fl_{no} = N$ 
else
  Use Eq. (8)
end if
 $FM =$  Fitness Function  $f(X)$ 
if  $Iteration = 1$ 
  Place the moths in order of  $FM$ 
  Update the Flames
   $Iteration = 0$ 
else
  Sort the moths based on  $FM$  from last iteration
  Update the Flames
end if
Reduce the convergence constant
for  $j = 1 : N$ 
  for  $k = 1 : n$ 
  Update  $a_1, t$  and moths position as to their particular flame using Eq. (9)
  end for
end for
if  $p < 0.5$  %  $p$  is a random number  $\in [0, 1]$ 
  Find new solution using Eq. (13)
  Check boundary
else
  Apply Fibonacci approach using Eqs. (10)–(12) and check boundaries
end if
 $Iteration = Iteration + 1$ 
end while
Output: The best solution in the ecosystem with the lowest fitness function value.
    
```

5. Simulation study and discussions

Section 5.1 provides a high-level overview of the benchmark functions that formed the basis of this investigation. The experimental setup for our proposed method is discussed in Section 5.2. In Section 5.3, we see how Es-MFO stacks up against other common base algorithms. The Sections 5.4, 5.5, and 5.6 show results from statistical analysis, convergence performance, complexity and diversity analysis of the proposed Es-MFO algorithm, respectively. There is a presentation of experimental analysis including statistical and convergence analysis in Section 5.7.

Table 1

The parameters setup of Es-MFO and other algorithms.

Algorithm	Parameters values
DE	Scaling Factor (F) = 0.5 = Crossover probability
PSO	Inertia weight (w) = 0.9 to 0.4, Acceleration coefficient $c_1 = c_2 = 0.2$, Maximum velocity ($V_{max} = 4$)
FPA	Switching probability (p) $\in [0, 1]$, Scaling factor (γ) = 0.01
BOA	Switch probability (p) = 0.8, Sensor modality (c) = 0.01, Power exponent (a) = 0.1 to 0.3
SSA	Two random numbers (c_1 and c_2) where $0 \leq c_1, c_2 \leq 1$
SCA	Random numbers ($r_2 = rand * 2\pi, r_3 = rand * 2\pi, r_4 = rand$) and constant $a = 2$
MFO	Convergence constant $t \in [-1, 1]$ and parameter ' r ' decreases linearly from (-1) to (-2) , $b = 1$
Es-MFO	Convergence constant $t \in [-1, 1]$ and Parameter ' r ' decreases non-linearly from (-1) to (-2) , $b = 1$

5.1. Benchmark functions

When it comes to the trustworthiness, verifiability, and effectiveness of an algorithm, benchmark functions are indispensable. The selected test functions are included in Appendix-1. In order to test the effectiveness of the proposed Es-MFO method, 29 benchmark functions were chosen and classified as either unimodal or multimodal. Some chosen unimodal and multimodal functions' three-dimensional structures are displayed in Figs. 5 and 6.

Because each of the unimodal functions (F1-F7) has a unique optimal value, they can be used to test the efficiency of an algorithm in exploiting data. However, the multimodal functions (F7-F19) are rife with local optima, and so can be used to test an algorithm's ability to avoid such solutions. As more dimensions, search areas, and local optimum values are added to a multimodal function, its complexity increases. In Fig. 4, we see a three-dimensional representation of randomly chosen unimodal and multimodal benchmark functions.

5.2. Experimental setup

The proposed algorithm's source code has been written and implemented in MATLAB R2015a, which has been run on a computer with an Intel i5 processor, 8 GB of RAM, and Windows 2010. All algorithms are terminated after reaching a maximum of 1000 iterations and a population size of 30. Because of the randomness inherent in the proposed and compared algorithms, we have run each function 30 times and calculated the average and standard deviation for each. The results have been rounded up to two decimal points to provide smaller statistical errors and statistically meaningful output. Table 1 provides the parameters used by each compared algorithm. Table 2 compares Es-MFO to some of the most popular existing algorithms for data collection and organization, including DE, PSO, SCA, JAYA, BOA, and MFO. In Appendix 1, we see the precise mathematical expressions for all nineteen of our standard-setting functions, complete with their dimensions, variable ranges, and optimal values.

5.3. Discussion on classical benchmark functions

Simulation results of our proposed Es-MFO have been compared to those of 6 (six) other meta-heuristics, including DE, PSO, SCA, JAYA, and BOA, on 19 benchmark functions, both unimodal and multimodal. Table 2 includes the mean and standard deviation for optimized unimodal functions, Es-MFO, and the other six techniques. Table 2 makes clear that Es-MFO provided the fewest values when compared to other methods. The best outcomes for the functions F1, F3, F4, F6, and F7 are produced by the Es-MFO algorithm. It provides the second-best outcomes for the F2 and F5 functions. Table 2 also shows the investigation of functions F8 to F17 under multimodal function optimization. For the functions F8, F12, F13, and F17, it is obvious that Es-MFO produces better results than other techniques. It provides the second and third-

Table 2
Comparison of Es-MFO with some basic algorithms.

Sl. No.	Performance measure	Es-MFO	MFO	DE	PSO	SCA	Jaya	BOA		
F1	Avrg	3.42e-03	1.34e-02	2.65E-01	6.28E-03	1.60E-02	7.24e-07	0	4.61E-01	
	Sdev			6.28E-01	1.31E-03	0	1.22e-06	0	4.41E-01	
F2	Avrg	1.11e-04	3.44e-04	1.87	7.49E-05	1.24E + 01	5.29e-02	0	1.61E-01	
	Sdev			2.07	6.97E-05	0	1.03e-01	0	2.24E-01	
F3	Avrg	0	1.11E-11	1.14E-04	3.47E-02		1.71e-187	9.45E-56	0	
	Sdev	0	4.27E-11	1.11E-04	0		0	3.01E-55	0	
F4	Avrg	0	6.81E-11	1.16E-08	5.82E-88		5.71e-196	1.45E-218	0	
	Sdev	0	3.72E-11	2.73E-08	0		0	0	0	
F5	Avrg	-1.55e-02	8.75e-01	-3.55E-03	-3.78E-03	8.63E-03	-3.79e-03	-3.78E-03	-3.43E-03	
	Sdev			3.40E-04	6.22E-06	0	2.35e-08	2.20E-09	7.59E-04	
F6	Avrg	1.79e-07	6.95e-07	1.83E-01	2.23E-02	8.71E-02	5.88e-03	3.54E-02	1.05E-01	
	Sdev			2.74E-01	2.18E-02	0	1.42e-02	4.28E-02	1.70E-01	
F7	Avrg	0	7.46E-98	9.93E-15	6.59E-01		2.93e-03	1.14E-110	0	
	Sdev	0	4.08E-97	3.87E-14	0		1.35e-02	4.51E-112	0	
F8	Avrg	0	0	0	3.74E-07		8.16e-02	0	0	
	Sdev	0	0	0	0		1.63e-01	0	0	
F9	Avrg	8.24e-01	9.95e-01	6.35E-02	4.08E-21	2.07E-01	4.39e-03	9.30E-01	2.55E-01	
	Sdev			1.02E-01	7.56E-21	0	1.40e-02	1.01E-47	3.11E-01	
F10	Avrg	3.68e-05	2.02e-04	3.12E-51	4.28E-05	2.20E-16	1.55e-04	1.60E-04	0	
	Sdev			1.70E-50	1.14E-05	0	7.49e-04	3.18E-07	0	
F11	Avrg	-4.91e-11	3.89e-11	-1.12E-10	-1.12E-10	-5.85E-11	-1.14e-10	-1.15E-10	-9.50E-11	
	Sdev			1.32E-14	9.60E-14	0	2.53e-13	1.46E-25	2.00E-11	
F12	Avrg	0	4.17E-29	3.68E-05	2.73E-01		1.00e-08	5.07E-30	0	
	Sdev	0	2.28E-28	2.57E-05	0		2.15e-08	9.54E-30	0	
F13	Avrg	7.10	1.29e + 01	1.60E + 01	3	3.60	9.93	3	1.28E + 01	
	Sdev			2.00E + 01	1.61E-02	0	2.86	2.13E-04	9.65	
F14	Avrg	8.30e-02	2.24e-01	1.18E-01	1.92E-01	6.35E-01	1.59e-01	2.38E-02	6.05E-01	
	Sdev			1.16E-01	1.87E-01	0	2.10e-01	3.18E-02	5.90E-01	
F15	Avrg	3.57e-01	2.79e-01	2.41	1.66E + 05	4.35E + 08	4.87e-01	4.19E-01	3	
	Sdev			4.36E-01	1.32E + 05	0	4.07e-01	5.17E-01	0	
F16	Avrg	1.13e-03	5.01e-04	1.25E-03	1.70E-03	1.10E-01	5.70e-04	3.10E-02	5.13E-03	
	Sdev			1.35E-03	9.33E-04	0	4.62e-04	2.09E-05	3.33E-03	
F17	Avrg	0	0	6.44E-05	7.34E + 01		1.60e + 00	0	1.37E-04	
	Sdev	0	0	9.78E-05	0		8.20e-01	0	7.06E-04	
F18	Avrg	1.02	9.63e-01	2.37E-01	3.33E + 01	1.02E + 01	7.50E-01	2.17e-02	1.87E + 01	1.92E-08
	Sdev			2.69E-02	2.97E-01		3.70e-02	4.07E-01	1.25E-09	
F19	Avrg	2.90e-01	2.96e-01	0	5.24E + 01	2.56E + 01	1.35	4.73e-01	4.15E + 01	9.09E-01
	Sdev			0	4.52E-01		6.02e-01	5.10E-01	1.35E-02	

Table 3
Evaluation of Es- MFO’s efficiency with other approaches using 19 standard benchmark functions.

	MFO	DE	PSO	SCA	JAYA	BOA
Superior to	14	14	15	15	16	11
Similar to	2	1	0	0	2	4
Inferior to	3	4	4	4	1	4

highest outcomes for functions F9, F10, F15, F18, and F19, and it performs worse than other methods for functions F14 and F16. As a result, it can be said that Es-MFO is a better algorithm for optimizing multimodal functions than the other six algorithms.

The percentage of instances in which Es- MFO’s mean performance is superior to, comparable to, or inferior to the other six algorithms is shown in Table 3. According to Table 3, Es-MFO performs better than MFO, DE, PSO, SCA, JAYA, and BOA in 14, 14, 15, 13, and 11 benchmark functions, respectively. On 2, 1, 0, 0, 2, and 4 instances, respectively, comparable results can be observed, while poorer values are obtained in 3, 4, 4, 4, and 4 benchmark functions.

5.4. Discussion on statistical and convergence performance for basic benchmark problems

For better performance evaluation, statistical tests have become increasingly commonplace in computational methods in recent years (Sahoo & Saha, 2022c; Sharma et al., 2022). The most common application for them is in an experimental study designed to compare the

Table 4
Wilcoxon’s test results for proposed Es-MFO.

Es-MFOvs. Algorithm	P-Value	R+	R-	Winner
DE	0.000	135	36	Es-MFO
PSO	0.003	158	32	Es-MFO
JAYA	<0.001	107	46	Es-MFO
BOA	0.000	110	40	Es-MFO
SCA	0.000	140	50	Es-MFO
MFO	0.004	104	49	Es-MFO

performance of different algorithms. This potentially challenging undertaking is essential for figuring out if a newly proposed solution significantly enhances the established solutions for a certain problem. Analysis of the suggested Es-MFO algorithm’s efficacy is conducted here using Friedman and Wilcoxon signed-rank tests. The strong

Table 5
Friedman’s rank test of proposed Es-MFOwith considered algorithms.

Method	Average rank	Rank	P-value
DE	4.39	6	At the 1% level of significance, Ho is ruled out with a P-value of (0.0000.01). At a 1% level of significance, the performance of several approaches differs significantly from one another.
PSO	5.79	7	
JAYA	3.42	2	
BOA	4.00	4	
SCA	3.63	3	
MFO	4.08	5	
Es-MFO	2.68	1	

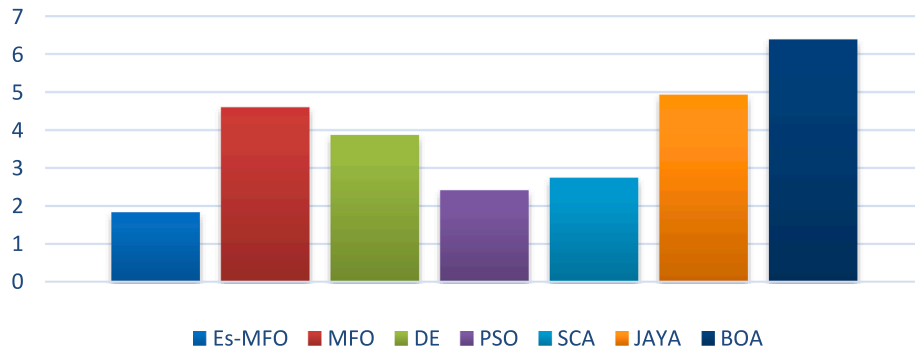


Fig. 5. Results of Friedman rank test.

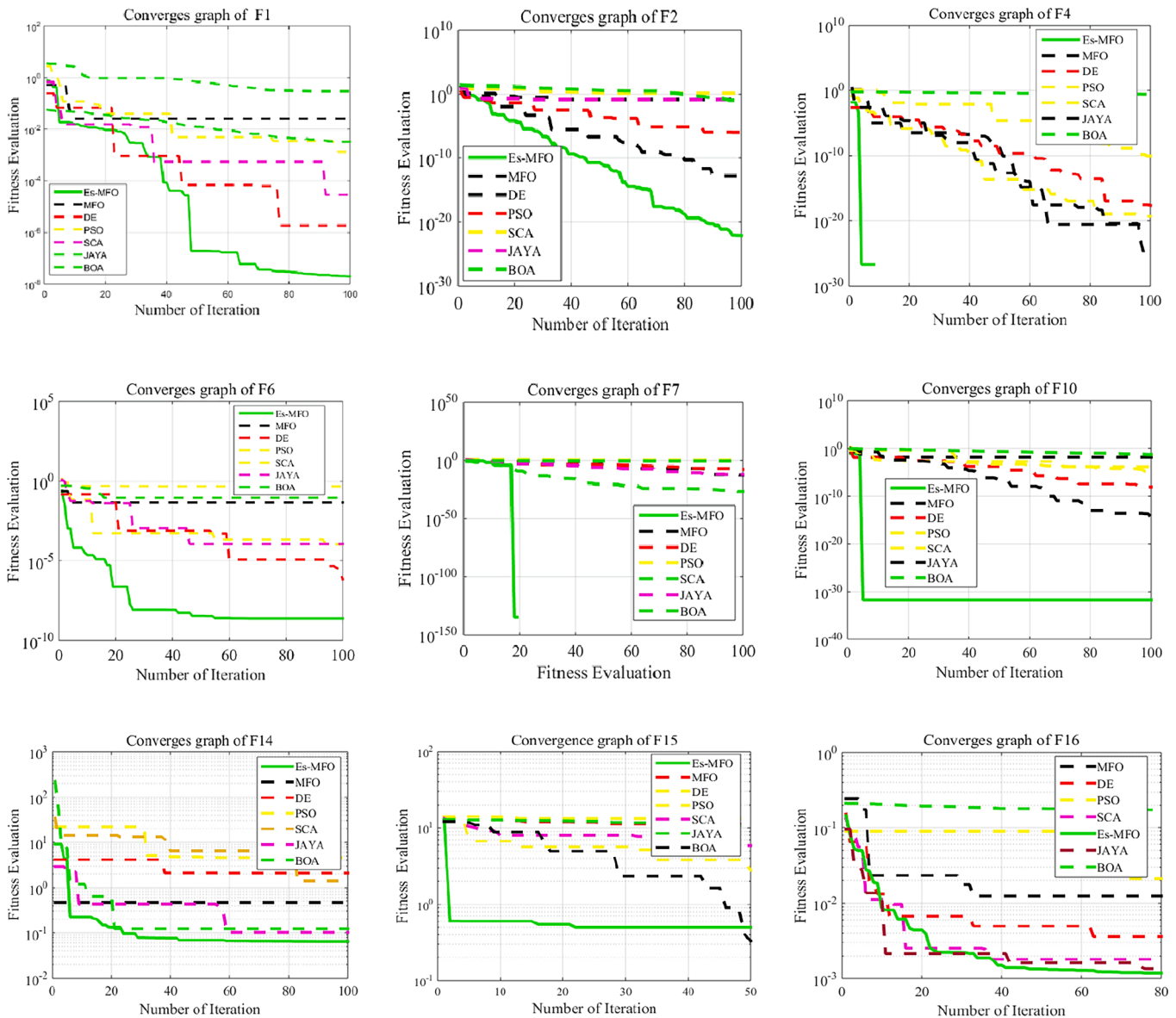


Fig. 6. Convergence graph of Basic functions.

outperformance of the Es-MFO on the test functions was assessed by running the multiple-problem Wilcoxon test and Friedman’s test in SPSS simultaneously.

Table 4 shows the Wilcoxon rank test’s significance level of 5% for comparing Es-MFO to the other algorithms studied for a set of 19

benchmark functions. Results in Table 4 show that Es-MFO is superior to its competitors because the R + values are all greater than the R- values.

The results of the Friedman-Rank test, which are shown in Table 5, have a 95% level of confidence, demonstrating that Es-performance MFO’s is both significant and compatible with conventional

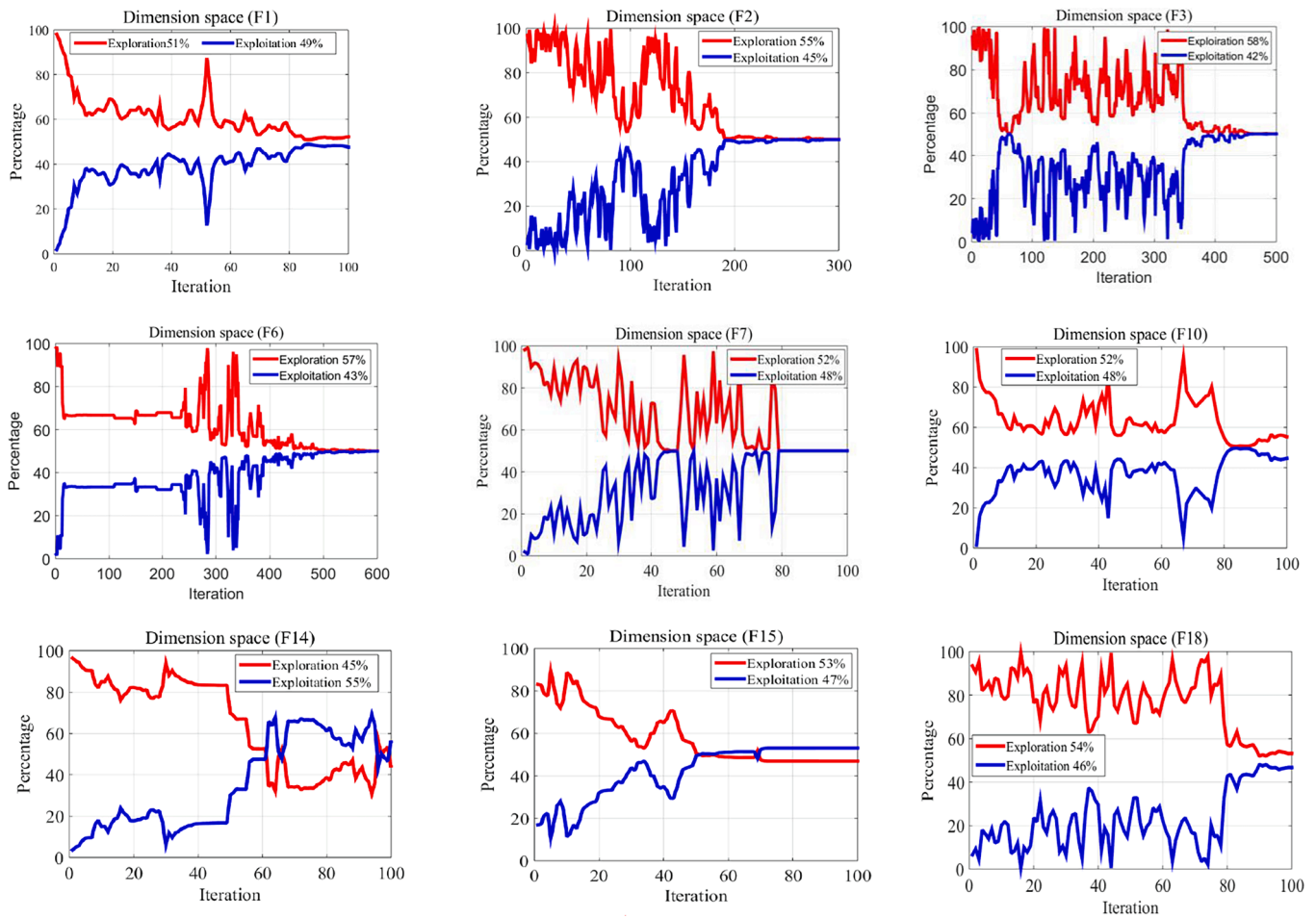


Fig. 7. Diversity analysis of Es-MFO for different benchmark functions.

algorithms like MFO, DE, PSO, SCA, JAYA, and BOA for a set of 19 benchmark functions.

According to Table 5, the proposed algorithm's mean level is the lowest of all the others, making Es-MFO's rank the lowest. Therefore, it is clear from the results on these benchmarks that the suggested Es-MFO has significantly improved space exploitation, convergence speed, and numeric performance compared to the traditional rival. In Fig. 5. A graphical comparison of mean rank to different advancement algorithms is shown.

5.5. Convergence performance on basic benchmark functions

Some of the convergence graphs compared with MFO, DE, PSO, SCA, JAYA, and BOA for a few benchmark functions, such as F1, F2, F4, F6, F7, F10, F14, F15, and F16 have been provided in Fig. 6 in order to compare the convergence rate of Es-MFO with other algorithms. These figures display the objective function value as well as the function evaluation along the horizontal and vertical axes, respectively. Dimensional size 100 was used to illustrate the curves. Comparing ES-MFO to other approaches, it converges quickly on the global optima. However, due to their tendency to get caught in local optima, other optimization methods under comparison have a moderate convergence rate. Due to its location in a narrow canyon, convergent to the global optima for F14 and F16 is challenging for optimization problems. However, this global optimum is attained within 100 dimensions by the suggested Es-MFO. As a result, the suggested Es-MFO exhibits a high rate of convergence when combined with other optimization techniques.

5.6. Complexity and diversity analysis of the Es-MFO algorithm

In this section, two efficient analyses such as complexity and diversity analysis of the suggested Es-MFO algorithm are discussed and presented in Sections 5.6.1 and 5.6.2 respectively.

5.6.1. Complexity analysis

To properly evaluate population-based algorithms, it is important to consider the quality of the solutions they produce, and the computational time required to obtain those solutions. The complexity of an algorithm can be expressed as a function of its running time or space requirements with the input size. This process of determining the formula for the total time required for a successful algorithm execution is known as time complexity analysis. The proposed Es-MFO algorithm's computational complexity is analyzed using big-O notation. The complexity of Es-MFO (T_{Es-MFO}) also depends on initialization of moth position (T_{IMP}), evaluation of moth position (T_{EMP}), sorting of moth based on fitness (T_{MF}), update of flames (T) and update phase of moth position with FSM (t_{UMFSM}). We will denote the maximum number of iterations, number of variables, and number of moths as I , D , and N , respectively. We will use time complexity for the comparison of both Es-MFO and MFO algorithm.

Complexity of initialization of moth position is $T_{IMP} = O(N * D)$.

The computation of fitness function for each organism i.e., complexity of $T_{EMP} = O(D)$.

Computational complexity of $T_{MF} = O(N \log(N))$.

Complexity of $T_{FI} = O(N * D)$.

Complexity of $T_{UMFSM} = O(N * D) + O(\log(N))$

Since the original MFO algorithm (Mirjalili, 2015) uses quicksort

Table 6a
Experimental outcomes of Es-MFO and basic algorithms on CEC'17 functions for D = 30.

Func.		Es-MFO	BOA	JAYA	SCA	MFO	GA	BA	MFO3	OMFO	SMFO
1	Avrg	6.21×10^{10}	6.70×10^{10}	6.93×10^{10}	6.39×10^{10}	6.50×10^{10}	4.69×10^{10}	1.32×10^{11}	6.92×10^{10}	6.56×10^{10}	6.21×10^{10}
	S _{dev}	1.01×10^{10}	5.88×10^9	1.99×10^9	6.80×10^9	9.04×10^9	7.50×10^9	1.47×10^{10}	8.65×10^9	8.16×10^9	6.24×10^9
3	Avrg	1.25×10^5	8.50×10^4	1.58×10^5	1.00×10^5	9.28×10^4	1.71×10^8	5.26×10^7	9.62×10^4	9.42×10^4	8.71×10^4
	S _{dev}	3.07×10^4	6.92×10^3	2.09×10^4	1.37×10^4	3.67×10^3	2.13×10^8	1.50×10^8	1.45×10^4	3.43×10^3	5.33×10^3
4	Avrg	1.66×10^4	2.31×10^4	8.47×10^2	1.69×10^4	1.80×10^4	1.88×10^4	3.78×10^4	2.29×10^4	1.88×10^4	1.68×10^4
	S _{dev}	3.93×10^3	4.26×10^3	1.53×10^2	2.87×10^3	3.90×10^3	3.59×10^3	9.33×10^3	4.67×10^3	4.48×10^3	2.80×10^3
5	Avrg	8.80×10^2	1.06×10^3	9.75×10^2	9.77×10^2	9.85×10^2	8.41×10^2	1.05×10^3	9.99×10^2	9.86×10^2	9.72×10^2
	S _{dev}	1.94×10^2	7.96×10^1	1.87×10^1	3.56×10^1	2.59×10^1	3.36×10^1	6.59×10^1	3.48×10^1	2.77×10^1	1.63×10^1
6	Avrg	6.52×10^2	7.08×10^2	6.96×10^2	6.86×10^2	7.04×10^2	6.87×10^2	6.76×10^2	7.07×10^2	7.05×10^2	7.01×10^2
	S _{dev}	2.82×10^1	1.20×10^1	2.54×10^0	5.80×10^0	6.12×10^0	5.77×10^0	7.04×10^0	7.93×10^0	8.61×10^0	6.35×10^0
7	Avrg	1.46×10^3	1.66×10^3	1.63×10^3	1.54×10^3	1.53×10^3	1.58×10^3	2.49×10^3	1.55×10^3	1.55×10^3	1.49×10^3
	S _{dev}	2.41×10^1	1.16×10^1	2.00×10^1	4.86×10^1	4.97×10^1	3.33×10^1	3.49×10^2	3.78×10^1	4.08×10^1	2.84×10^1
8	Avrg	1.08×10^3	1.26×10^3	1.05×10^3	1.23×10^3	1.19×10^3	1.29×10^3	1.38×10^3	1.21×10^3	1.20×10^3	1.18×10^3
	S _{dev}	1.50×10^2	5.59×10^1	1.45×10^1	2.47×10^1	2.45×10^1	2.69×10^1	8.85×10^1	2.98×10^1	2.51×10^1	1.80×10^1
9	Avrg	1.25×10^4	1.73×10^4	3.98×10^4	9.73×10^3	1.53×10^4	1.24×10^4	1.79×10^4	1.49×10^4	1.55×10^4	1.33×10^4
	S _{dev}	1.13×10^4	2.92×10^3	8.57×10^3	9.25×10^2	1.68×10^3	2.11×10^3	7.65×10^3	2.14×10^3	1.50×10^3	1.26×10^3
10	Avrg	6.83×10^3	7.79×10^3	9.91×10^3	9.31×10^3	9.69×10^3	8.06×10^3	1.34×10^9	9.74×10^3	9.74×10^3	9.30×10^3
	S _{dev}	7.88×10^2	6.40×10^2	2.19×10^2	6.42×10^2	4.66×10^2	3.16×10^2	3.56×10^3	4.38×10^2	4.82×10^2	3.75×10^2
11	Avrg	5.31×10^3	7.77×10^3	3.58×10^4	9.57×10^3	1.46×10^4	4.66×10^4	5.84×10^4	1.28×10^4	1.36×10^4	1.01×10^4
	S _{dev}	1.95×10^3	1.95×10^3	6.58×10^3	2.91×10^3	4.19×10^3	3.71×10^4	3.66×10^4	4.18×10^3	4.08×10^3	1.81×10^3
12	Avrg	2.31×10^9	1.81×10^{10}	1.01×10^9	1.64×10^{10}	1.51×10^{10}	1.10×10^{10}	6.63×10^{10}	1.69×10^{10}	1.68×10^{10}	1.51×10^{10}
	S _{dev}	7.40×10^9	2.58×10^9	4.32×10^8	4.07×10^9	3.20×10^9	1.14×10^9	1.70×10^{10}	4.89×10^9	3.58×10^9	2.24×10^9
13	Avrg	1.04×10^8	1.82×10^{10}	2.16×10^8	9.84×10^9	1.33×10^{10}	2.34×10^{10}	3.31×10^{10}	1.91×10^{10}	1.52×10^{10}	1.24×10^{10}
	S _{dev}	2.24×10^8	4.88×10^9	2.78×10^8	8.00×10^9	4.74×10^9	4.07×10^9	1.45×10^{10}	6.91×10^9	6.62×10^9	4.85×10^9
14	Avrg	1.39×10^6	8.18×10^6	7.05×10^5	9.70×10^5	1.02×10^7	6.95×10^7	1.30×10^8	2.85×10^7	2.53×10^7	9.31×10^6
	S _{dev}	1.99×10^6	8.85×10^6	3.77×10^5	1.31×10^6	8.01×10^6	3.95×10^7	1.23×10^8	2.82×10^7	2.59×10^7	7.40×10^6
15	Avrg	8.93×10^8	1.17×10^9	1.02×10^8	3.43×10^8	1.57×10^9	3.16×10^9	6.76×10^9	2.34×10^9	1.77×10^9	9.36×10^8
	S _{dev}	2.25×10^9	4.63×10^8	6.67×10^7	3.84×10^8	1.26×10^9	1.59×10^9	3.75×10^9	1.59×10^9	1.35×10^9	5.33×10^8
16	Avrg	3.23×10^3	7.61×10^3	5.95×10^3	5.82×10^3	6.78×10^3	8.45×10^3	8.87×10^3	7.46×10^3	7.15×10^3	6.88×10^3
	S _{dev}	3.80×10^2	1.88×10^3	2.20×10^2	6.06×10^2	9.69×10^2	2.79×10^3	2.98×10^3	1.61×10^3	1.14×10^3	9.27×10^2
17	Avrg	2.46×10^3	8.13×10^3	2.76×10^3	6.65×10^3	7.17×10^3	3.02×10^4	1.55×10^4	1.69×10^4	6.24×10^3	6.53×10^3
	S _{dev}	2.35×10^2	5.55×10^3	1.46×10^2	4.85×10^3	4.79×10^3	4.81×10^4	1.77×10^4	2.12×10^4	4.36×10^3	3.89×10^3
18	Avrg	1.02×10^7	1.37×10^8	2.12×10^7	6.83×10^6	1.43×10^8	4.68×10^8	2.37×10^8	2.38×10^8	1.72×10^8	9.26×10^7
	S _{dev}	1.54×10^7	1.03×10^8	1.06×10^7	6.05×10^6	1.05×10^8	2.64×10^8	1.97×10^8	1.69×10^8	1.35×10^8	4.95×10^7
19	Avrg	1.33×10^9	1.65×10^9	3.05×10^7	4.33×10^8	2.19×10^9	3.19×10^9	3.38×10^9	2.40×10^9	2.10×10^9	1.17×10^9
	S _{dev}	2.70×10^9	7.45×10^8	3.40×10^7	3.69×10^8	1.15×10^9	1.77×10^9	1.89×10^9	1.05×10^9	1.37×10^9	5.56×10^8
20	Avrg	2.73×10^3	3.24×10^3	3.90×10^3	3.81×10^3	3.31×10^3	3.29×10^3	4.34×10^3	3.48×10^3	3.44×10^3	3.26×10^3
	S _{dev}	2.52×10^2	2.50×10^2	1.28×10^2	2.46×10^2	1.84×10^2	1.63×10^2	6.39×10^2	2.01×10^2	1.66×10^2	1.55×10^2
21	Avrg	2.52×10^3	2.81×10^3	2.05×10^3	2.72×10^3	2.81×10^3	2.74×10^3	3.04×10^3	2.83×10^3	2.79×10^3	2.80×10^3
	S _{dev}	3.36×10^1	7.08×10^1	1.57×10^1	4.31×10^1	4.44×10^1	4.64×10^1	1.39×10^2	5.80×10^1	3.38×10^1	4.85×10^1

(continued on next page)

Table 6a (continued)

Func.		Es-MFO	BOA	JAYA	SCA	MFO	GA	BA	MFO3	OMFO	SMFO
22	Avrg	6.73×10^3	9.39×10^3	9.91×10^3	9.86×10^3	1.05×10^4	8.41×10^3	1.47×10^4	1.09×10^4	1.05×10^4	9.97×10^3
	S _{dev}	2.30×10^3	7.84×10^2	1.35×10^3	1.00×10^3	6.94×10^2	2.18×10^3	3.36×10^2	5.72×10^2	5.73×10^2	7.11×10^2
23	Avrg	2.87×10^3	3.96×10^3	3.99×10^3	3.61×10^3	3.72×10^3	3.59×10^3	1.47×10^4	3.88×10^3	3.78×10^3	3.77×10^3
	S _{dev}	3.65×10^1	2.08×10^2	4.66×10^1	1.11×10^2	1.41×10^2	1.85×10^2	3.36×10^3	2.13×10^2	1.73×10^2	1.58×10^2
24	Avrg	3.05×10^3	4.30×10^3	3.96×10^3	3.86×10^3	4.03×10^3	3.81×10^3	4.88×10^3	4.16×10^3	4.22×10^3	4.08×10^3
	S _{dev}	4.10×10^1	1.44×10^2	2.73×10^2	1.77×10^2	1.96×10^2	1.20×10^2	2.80×10^2	2.45×10^2	2.02×10^2	2.25×10^2
25	Avrg	6.39×10^3	6.16×10^3	2.96×10^3	4.96×10^3	6.30×10^3	8.40×10^3	1.80×10^4	6.84×10^3	6.29×10^3	5.94×10^3
	S _{dev}	2.92×10^3	6.24×10^2	2.46×10^1	4.86×10^2	9.06×10^2	1.78×10^3	2.91×10^3	1.03×10^3	7.23×10^2	4.78×10^2
26	Avrg	7.74×10^3	1.19×10^4	1.36×10^4	1.18×10^4	1.20×10^4	1.34×10^4	1.95×10^4	6.84×10^3	6.29×10^3	5.94×10^3
	S _{dev}	3.44×10^3	8.02×10^2	4.97×10^3	8.73×10^2	9.52×10^2	2.38×10^3	2.90×10^3	1.03×10^3	7.23×10^2	4.78×10^2
27	Avrg	3.29×10^3	5.39×10^3	4.39×10^3	3.20×10^3	4.83×10^3	4.55×10^3	3.20×10^3	1.31×10^4	1.23×10^4	1.21×10^4
	S _{dev}	2.52×10^1	5.13×10^2	7.06×10^3	1.89×10^3	4.45×10^2	3.38×10^2	5.68×10^3	1.25×10^3	8.23×10^2	7.18×10^2
28	Avrg	7.09×10^3	8.52×10^3	7.30×10^3	3.30×10^3	7.73×10^3	7.49×10^3	3.30×10^3	5.38×10^3	5.15×10^3	5.00×10^3
	S _{dev}	3.03×10^3	4.33×10^2	6.90×10^3	4.25×10^3	1.01×10^3	5.16×10^2	6.14×10^3	5.82×10^2	4.08×10^2	4.42×10^2
29	Avrg	4.36×10^3	1.10×10^4	8.33×10^3	6.58×10^3	8.66×10^3	6.98×10^4	9.27×10^4	8.43×10^3	8.23×10^3	7.74×10^3
	S _{dev}	3.06×10^2	5.37×10^3	3.60×10^2	9.67×10^2	3.13×10^3	5.34×10^4	1.01×10^3	7.84×10^2	8.18×10^2	7.03×10^2
30	Avrg	3.97×10^6	3.13×10^9	4.39×10^7	2.27×10^9	2.30×10^9	2.75×10^9	5.93×10^9	1.65×10^4	9.81×10^3	9.08×10^3
	S _{dev}	3.29×10^6	1.08×10^9	3.63×10^7	1.02×10^9	1.08×10^9	6.85×10^8	3.11×10^9	2.70×10^4	3.42×10^3	2.90×10^3

algorithm, the computational complexity is $O[NI^*(D+N)]$ for the worst case. Therefore, the overall complexity of the suggested Es-MFO algorithm for worst case can be approximated as follows:

$$T_{Es-MFO} = T_{IMP} + T_{EMP} + T_{MI} + T_{FI} + = O(I^*(2ND + D + N^2 + \text{Log}(N))) \quad (14)$$

So, the time complexity of Es-MFO is greater than MFO algorithm. It is true that the complexity of computation has increased, but the increase is negligible considering the enhancement.

5.6.2. Diversity analysis

Findings from studies of population variety can aid in the creation of an evolutionary algorithm by shedding light on how such a programme operates. Optimisation algorithms use a group of people to improve the quality of the search areas and speed up the process of locating the optimal answers. In most cases, search agents that have found the best answers will try to steer you in that direction. This pulls search agents further apart, diminishing the benefits of diversity. On the other hand, the effect of intensification grows when distance between people shrinks. A diversity assessment (Sahoo et al., 2022a) is examined and defined as follows to evaluate the shrinking and growing distances between search agents:

$$div^j = \frac{1}{N} \sum_{i=1}^N |median(x^j) - x_i^j| \quad (15)$$

$$div = \frac{1}{d} \sum_{j=1}^d div^j \quad (16)$$

where, N and d represents number of search agents and design variables respectively, x_i^j is the dimension j of the i th search agent and $median(x^j)$ is the median of dimension j in the whole population, div^j is the diversity in each dimension and mathematically, it is defined as the distance

between the j 'th dimension of every search agent and the median of that dimension. The diversity of whole population (div) is then calculated by taking average of every div^j .

Furthermore, with the help of diversity measurement, we can calculate the percentage of both exploration and exploitation during each iteration using following Equations:

$$exploration\% = \left(\frac{div}{div_{max}} \right) \times 100 \quad (17)$$

$$exploitation\% = \left(\frac{|div - div_{max}|}{div_{max}} \right) \times 100 \quad (18)$$

where div_{max} is defined as the maximum diversity value in the whole optimization process and $|div - div_{max}|$ is the absolute value between div and div_{max} . The $exploration\%$ is the link between the diversity in each iteration and the maximum diversity obtained. The $exploitation\%$ relates to the exploitation level and it is evaluated as the complementary percentage to $exploration\%$ as the difference between the maximal diversity and the current diversity of an iteration is caused by the concentration of search agents.

Our proposed Es-MFO algorithm was tested on 23 benchmark functions, allowing us to analyse the trade-offs between exploration and exploitation. The benchmark functions F1, F2, F3, F6, F7, F10, F14, F15, and F18 from Appendix 1 were chosen to ensure a level playing field and presented in Fig. 7. The X-axis shows the number of iterations, while the Y-axis displays the percentage of both exploration and exploitation.

5.6.3. Experimental comparisons on CEC 2017

To evaluate the performance of the novel hybrid Es-MFO algorithm, it is tested with high complexity CEC2017 benchmark problems consisting of four parts: (1) Unimodal (f1-f3); (2) Multimodal (f4-f10); (3) Hybrid (f11-f20) and (4) composite (f21-30) benchmark problems. Out of thirty functions, the second unimodal function (f2) has been discarded

Table 6b
Experimental outcomes of Es-MFO and basic algorithms on CEC'17 functions for D = 50.

Func.		Es-MFO	BOA	JAYA	SCA	MFO	GA	BA	MFO3	OMFO	SMFO
1	Avrg	9.29×10^{10}	1.16×10^{11}	2.89×10^{10}	5.42×10^{10}	1.29×10^{11}	4.52×10^{10}	1.52×10^{11}	1.25×10^{11}	1.24×10^{11}	1.18×10^{11}
	S _{dev}	4.73×10^{10}	6.02×10^9	4.51×10^9	9.22×10^9	8.04×10^9	8.37×10^9	1.47×10^9	6.57×10^9	6.83×10^9	6.35×10^9
3	Avrg	1.96×10^5	1.86×10^5	3.09×10^5	1.80×10^5	3.90×10^5	3.55×10^8	6.26×10^7	3.04×10^5	1.09×10^6	3.85×10^6
	S _{dev}	3.38×10^4	2.21×10^4	4.57×10^4	2.01×10^4	4.86×10^5	7.05×10^8	1.57×10^7	1.31×10^5	2.41×10^6	1.98×10^7
4	Avrg	2.83×10^4	4.29×10^4	4.38×10^4	4.13×10^4	4.39×10^4	4.83×10^4	3.68×10^4	4.56×10^4	4.41×10^4	4.12×10^4
	S _{dev}	2.58×10^4	4.79×10^3	6.46×10^3	5.40×10^3	7.35×10^3	3.06×10^3	9.53×10^3	5.98×10^3	5.71×10^3	4.74×10^3
5	Avrg	1.19×10^3	1.29×10^3	1.25×10^3	1.25×10^3	1.26×10^3	8.37×10^3	2.05×10^3	1.27×10^3	1.27×10^3	1.23×10^3
	S _{dev}	1.55×10^2	6.16×10^1	3.20×10^1	3.27×10^1	3.29×10^1	2.72×10^1	6.59×10^1	3.97×10^1	3.17×10^1	2.19×10^1
6	Avrg	6.84×10^2	7.18×10^2	7.47×10^2	7.08×10^2	7.16×10^2	6.61×10^2	7.76×10^2	7.16×10^2	7.17×10^2	7.15×10^2
	S _{dev}	3.85×10^1	1.12×10^1	6.00×10^0	6.45×10^0	4.75×10^0	7.59×10^0	7.04×10^0	6.90×10^0	5.45×10^0	4.33×10^0
7	Avrg	2.06×10^3	2.17×10^3	2.12×10^3	2.17×10^3	2.14×10^3	2.25×10^3	2.55×10^3	2.14×10^3	2.15×10^3	2.07×10^3
	S _{dev}	2.07×10^2	5.16×10^1	5.55×10^1	4.96×10^1	3.64×10^1	3.11×10^1	3.59×10^2	4.21×10^1	3.68×10^1	3.36×10^1
8	Avrg	1.51×10^3	1.63×10^3	1.58×10^3	1.59×10^3	1.58×10^3	1.55×10^3	1.58×10^2	1.60×10^3	1.59×10^3	1.55×10^3
	S _{dev}	1.82×10^2	6.08×10^1	3.23×10^1	3.12×10^1	3.98×10^1	1.91×10^1	8.85×10^1	3.29×10^1	3.56×10^1	3.10×10^1
9	Avrg	5.55×10^4	4.63×10^4	1.76×10^4	3.99×10^4	4.82×10^4	1.21×10^4	2.39×10^4	4.67×10^4	4.86×10^4	4.32×10^4
	S _{dev}	2.83×10^4	4.42×10^3	4.41×10^3	3.29×10^3	4.16×10^3	4.55×10^3	7.65×10^3	4.55×10^3	5.12×10^3	2.92×10^3
10	Avrg	1.21×10^4	1.39×10^4	1.42×10^4	1.53×10^4	1.64×10^4	2.34×10^4	1.34×10^4	1.64×10^4	1.65×10^4	1.60×10^4
	S _{dev}	1.06×10^3	7.49×10^2	3.72×10^2	4.13×10^2	4.15×10^2	6.80×10^3	3.56×10^3	5.35×10^2	4.96×10^2	3.81×10^2
11	Avrg	1.72×10^4	2.56×10^4	6.88×10^3	3.16×10^4	2.89×10^4	5.69×10^4	5.88×10^4	2.92×10^4	2.96×10^4	2.57×10^4
	S _{dev}	5.28×10^3	2.25×10^3	1.43×10^3	2.30×10^3	3.20×10^3	2.60×10^4	3.76×10^3	5.21×10^3	2.50×10^3	2.30×10^3
12	Avrg	1.47×10^{10}	8.76×10^{10}	1.09×10^{10}	7.74×10^{10}	9.32×10^{10}	1.09×10^{10}	6.69×10^{10}	9.11×10^{10}	9.72×10^{10}	9.01×10^{10}
	S _{dev}	3.53×10^{10}	8.77×10^9	2.79×10^9	1.48×10^{10}	1.47×10^{10}	1.42×10^9	1.70×10^9	1.91×10^{10}	1.49×10^{10}	1.08×10^{10}
13	Avrg	1.31×10^{10}	4.39×10^{10}	3.01×10^{10}	4.36×10^{10}	4.75×10^{10}	1.76×10^{10}	3.61×10^{10}	5.78×10^{10}	6.00×10^{10}	5.50×10^{10}
	S _{dev}	3.42×10^{10}	1.27×10^{10}	9.97×10^4	1.54×10^{10}	1.19×10^{10}	8.71×10^9	1.65×10^{10}	1.37×10^{10}	1.24×10^{10}	1.23×10^{10}
14	Avrg	4.80×10^6	1.84×10^8	1.33×10^8	5.71×10^7	1.32×10^8	6.34×10^7	1.35×10^8	2.10×10^8	1.72×10^8	1.71×10^8
	S _{dev}	4.95×10^6	1.08×10^8	7.22×10^7	5.98×10^7	6.81×10^7	2.97×10^7	2.23×10^8	1.29×10^8	7.26×10^7	9.32×10^7
15	Avrg	4.13×10^9	1.10×10^{10}	6.92×10^{10}	6.09×10^9	1.12×10^{10}	3.06×10^9	7.76×10^9	1.23×10^{10}	1.30×10^{10}	1.09×10^{10}
	S _{dev}	9.02×10^9	2.92×10^9	4.02×10^9	3.72×10^9	4.34×10^9	2.22×10^9	3.78×10^9	5.45×10^9	4.74×10^9	3.94×10^9
16	Avrg	4.84×10^3	1.15×10^4	8.31×10^3	8.51×10^3	1.05×10^4	8.20×10^3	1.85×10^4	1.12×10^4	1.13×10^4	1.06×10^4
	S _{dev}	4.92×10^2	1.84×10^3	3.08×10^2	1.64×10^3	1.74×10^3	3.39×10^3	2.98×10^3	2.19×10^3	1.66×10^3	1.18×10^3
17	Avrg	1.03×10^5	1.64×10^4	4.43×10^4	1.95×10^4	2.96×10^4	1.34×10^4	1.05×10^5	4.06×10^4	4.37×10^4	2.21×10^4
	S _{dev}	8.81×10^4	1.07×10^4	3.16×10^3	7.03×10^3	2.38×10^4	1.02×10^4	1.87×10^4	3.83×10^4	3.10×10^4	1.11×10^4
18	Avrg	3.36×10^7	2.44×10^8	8.09×10^7	8.40×10^7	3.16×10^8	9.86×10^8	2.57×10^8	4.03×10^8	3.72×10^8	2.81×10^8
	S _{dev}	2.14×10^7	9.07×10^7	3.48×10^7	6.66×10^7	1.43×10^8	1.52×10^8	1.98×10^8	2.62×10^8	1.90×10^8	1.39×10^8
19	Avrg	1.42×10^9	6.13×10^9	7.02×10^9	4.13×10^9	5.61×10^9	4.42×10^9	3.58×10^9	6.57×10^9	6.73×10^9	5.30×10^9
	S _{dev}	4.28×10^9	1.61×10^9	8.49×10^8	1.46×10^9	1.73×10^9	1.11×10^9	2.89×10^9	2.19×10^9	1.95×10^9	1.39×10^9
20	Avrg	4.00×10^3	4.37×10^3	4.35×10^3	3.96×10^3	4.61×10^3	4.36×10^3	5.36×10^3	4.68×10^3	4.76×10^3	4.50×10^3
	S _{dev}	6.95×10^2	3.78×10^2	1.51×10^2	3.59×10^2	2.41×10^2	2.51×10^2	6.39×10^2	2.52×10^2	2.01×10^2	1.68×10^2
21	Avrg	2.83×10^2	3.33×10^3	2.81×10^3	3.18×10^3	3.30×10^3	2.64×10^3	3.24×10^3	3.33×10^3	3.34×10^3	3.29×10^3
	S _{dev}	8.09×10^1	9.27×10^1	3.01×10^1	7.66×10^1	8.74×10^1	3.79×10^2	1.59×10^2	8.45×10^1	9.02×10^1	8.20×10^1

(continued on next page)

Table 6b (continued)

Func.		Es-MFO	BOA	JAYA	SCA	MFO	GA	BA	MFO3	OMFO	SMFO
22	Avrg	1.40×10^4	1.58×10^4	1.69×10^4	1.60×10^4	1.80×10^4	8.30×10^4	2.57×10^4	1.81×10^4	1.84×10^4	1.79×10^4
	S _{dev}	7.50×10^2	9.58×10^2	3.20×10^2	6.25×10^2	5.46×10^2	1.60×10^3	2.36×10^3	6.24×10^2	5.06×10^2	4.89×10^2
23	Avrg	3.32×10^3	5.03×10^3	3.63×10^3	4.24×10^3	4.79×10^3	3.35×10^3	2.46×10^4	4.90×10^3	4.83×10^3	4.71×10^3
	S _{dev}	6.26×10^1	2.40×10^2	1.54×10^2	1.65×10^2	2.68×10^2	6.28×10^1	3.66×10^3	2.81×10^2	2.97×10^2	2.56×10^2
24	Avrg	3.42×10^3	5.73×10^3	4.84×10^3	4.68×10^3	5.28×10^3	5.14×10^3	4.68	5.46×10^3	5.60×10^3	5.52×10^3
	S _{dev}	6.51×10^1	4.14×10^2	1.81×10^2	3.17×10^2	3.60×10^2	2.15×10^1	2.85×10^3	4.11×10^2	3.66×10^2	2.68×10^2
25	Avrg	1.46×10^4	1.62×10^4	4.01×10^4	1.55×10^4	1.69×10^4	3.54×10^4	2.20×10^4	1.79×10^4	1.72×10^4	1.62×10^4
	S _{dev}	6.11×10^3	1.04×10^3	2.35×10^3	1.30×10^3	1.56×10^3	1.06×10^3	2.51×10^3	1.29×10^3	1.23×10^3	1.07×10^3
26	Avrg	1.50×10^4	1.82×10^4	1.39×10^4	1.84×10^4	1.84×10^4	1.85×10^4	2.15×10^4	1.89×10^4	1.88×10^4	1.79×10^4
	S _{dev}	5.15×10^3	6.37×10^2	1.68×10^3	8.66×10^2	9.39×10^3	1.97×10^3	1.70×10^3	1.05×10^3	7.05×10^2	7.40×10^2
27	Avrg	3.87×10^3	8.52×10^3	3.20×10^3	3.20×10^3	7.79×10^3	4.04×10^3	3.25×10^3	7.96×10^3	8.17×10^3	7.94×10^3
	S _{dev}	1.10×10^2	8.67×10^2	2.99×10^3	1.27×10^3	7.94×10^2	2.32×10^2	5.58×10^3	1.03×10^3	1.05×10^3	8.54×10^2
28	Avrg	1.43×10^4	1.49×10^4	3.30×10^3	3.30×10^3	1.50×10^4	7.52×10^3	2.30×10^3	1.55×10^4	1.56×10^4	1.46×10^4
	S _{dev}	6.12×10^3	1.05×10^3	5.41×10^3	1.62×10^3	1.76×10^3	1.13×10^3	6.15×10^3	1.73×10^3	1.58×10^3	1.32×10^3
29	Avrg	2.32×10^5	2.15×10^5	9.96×10^5	7.97×10^4	1.85×10^5	6.74×10^4	1.27×10^5	2.74×10^5	2.99×10^5	1.24×10^5
	S _{dev}	1.24×10^6	1.47×10^5	1.24×10^4	1.36×10^5	2.66×10^5	3.68×10^4	1.01×10^4	2.59×10^5	4.66×10^5	1.07×10^5
30	Avrg	9.97×10^8	9.58×10^9	4.02×10^8	6.17×10^9	9.53×10^9	6.86×10^9	6.13×10^{10}	1.15×10^{10}	1.10×10^{10}	9.27×10^9
	S _{dev}	4.55×10^9	2.27×10^9	2.44×10^8	1.91×10^9	3.19×10^9	1.36×10^9	4.15×10^9	3.70×10^9	3.08×10^9	2.20×10^9

Table 7a

Simulation result for comparison of Es-MFO and basic algorithms for D = 30.

	BOA	JAYA	SCA	MFO	GA	BA	MFO3	OMFO	SMFO
Superior	22	20	21	25	23	26	24	25	26
Similar	0	0	0	2	1	0	0	0	0
Inferior	7	9	8	2	5	3	5	4	3

Table 7b

Simulation result for comparison of Es-MFO and basic algorithms for D = 50.

	BOA	JAYA	SCA	MFO	GA	BA	MFO3	OMFO	SMFO
Superior	24	23	21	27	22	22	25	26	24
Similar	0	0	0	0	0	0	0	0	0
Inferior	5	6	8	2	7	7	4	3	5

due to its unstable behavior, as presented in (Awad, Ali, Liang, Qu, & Suganthan, 2016a). All algorithms are evaluated on MATLAB 2015(a) with D*10000 function evaluation (for stopping criteria), the initial number of populations is 30 (thirty) and 30 independent runs have been executed for thirty and fifty dimensions. The average (Avrg) and standard deviation (S_{dev}) values are recorded for comparison.

5.6.4. Simulation and statistical analysis on basic and MFO variants

On CEC 2017 test suits, the Es-MFO algorithm is evaluated to nine traditional optimization algorithms: BOA, JAYA, SCA, MFO, GA, BA, MFO3 (Soliman et al. 2016), OMFO (Elsakaan et al., 2018), and SMFO (Chen et al., 2021). The comparison results of Es-MFO and with other considered algorithms are presented in Tables 6a and 6b. According to Tables 6a and 6b, our proposed Es-MFO algorithm reached more than 90% top results for all collections of CEC2017 benchmark problems on thirty and fifty dimensions when compared to other traditional optimization algorithms, but it only provides 80% best results when compared to the JAYA algorithm. Tables 7a and 7b show the number of occurrences of superiority, similarity, and inferiority for thirty and fifty

Table 8a

Friedman's rank test of Es-MFO and basic algorithms with D = 30.

Method	Mean rank	Rank	P-value
BOA	6.91	9	At the 1% level of significance, Ho is ruled out with a P-value of (0.0000.01). At a 1% level of significance, the performance of several approaches differs significantly from one another.
JAYA	5.09	4	
SCA	3.59	2	
MFO	5.66	5	
GA	6.07	7	
BA	8.73	10	
MFO3	6.73	8	
OMFO	5.89	6	
SMFO	3.68	3	
Es-MFO	2.66	1	

dimensions, respectively. From Table 7a, we revealed that Es-MFO works better than BOA, JAYA, SCA, MFO, GA, BA, MFO3, OMFO, and SMFO in 22, 22, 21, 25, 23, 26, 24, 25 and 26 benchmark functions,

Table 8b

Friedman’s rank test of Es-MFO and basic algorithms with D = 50.

Method	Average rank	Rank	P-value
BOA	4.69	5	At the 1% level of significance, Ho is ruled out with a P-value of (0.0000.01). At a 1% level of significance, the performance of several approaches differs significantly from one another.
JAYA	3.43	3	
SCA	3.18	2	
MFO	5.21	7	
GA	3.81	4	
BA	4.88	6	
MFO3	5.81	9	
OMFO	5.51	7	
SMFO	5.65	8	
Es-MFO	2.50	1	

respectively, Similar outcomes can be observed in 0, 0, 0, 2, 1, 0, 0, 0 and 0 instances, respectively, and inferior values are achieved in 7, 7, 8, 4, 6, 3, 5, 4 and 3 benchmark functions respectively. Again, From Table 7b, we observed that Es-MFO outperforms the BOA, JAYA, SCA, MFO, GA, BA, MFO3, OMFO, and SMFO in 24, 23, 21, 27, 22, 25, 26 and 24 benchmark functions, respectively.

5.6.4.1. Statistical and convergence analysis of Es-MFO on basic and MFO variants. To measure the effectiveness of the suggested Es-MFO, the Friedman rank test is conducted, and the results are displayed in Tables 8a and 8b for thirty and fifty dimensions, respectively. From Tables 8a and 8b, it can be concluded that the rank of the Es-MFO is the least. The convergence performance of Es-MFO and other considered optimization algorithms on the CEC’2017 test suite for dimensions thirty and fifty are shown in Figs. 8a and 8b, respectively. For convergence analysis, we have considered six different benchmark functions randomly out of twenty-nine as all twenty-nine functions significantly enlarge the article’s length. In Figs. 8a and 8b, the number of iterations and logarithm of best values so far are set on X-axis and Y-axis, respectively. In these figures, the suggested Es-MFO algorithm achieved fast convergence as compared to the other six basic algorithms and three MFO variants as the Es-MFO algorithm worked effectively by keeping a good balance between global and local search.

5.6.4.2. Simulation and statistical analysis on recent optimization algorithms. This subsection experiments on the proposed Es-MFO algorithm using CEC 17 functions. It is then compared to six recent algorithms, which are Fire Hawk Optimization (FHO) by Azizi et al. (2023), Arithmetic Optimization Algorithm (AOA) by Abualigah et al. (2021), Artificial Gorilla Troops Optimizer (GTO) by Abdollahzadeh et al. (2021), Multi Population-Based Adaptive Sine Cosine Algorithm (MAMSCA) by Saha (2022), Quantum Mutation Based Backtracking Search Algorithm (gQR-BSA) by Nama et al. (2022), and mLBOA by Sharma et al. (2022). The results of all algorithms are presented in Table 9, displaying the mean and standard deviation.

From Table 9, the proposed Es-MFO algorithm achieved competitive results on CEC2017 benchmark problems on thirty dimensions compared to GTO, MAMSCA, GQR-BSA, and mLBOA algorithms in most of the cases, and it provides more than 80% better results when compared to the FHO and AOA. Table 10 shows the number of superiority, similarity, and inferiority occurrences for the anticipated Es-MFO compared to FHO, AOA, GTO, MAMSCA, GQR-BSA, and mLBOA. Table 10 reveals that Es-MFO works better than FHO, AOA, GTO, MAMSCA, gQR-BSA, and mLBOA in 25, 24, 20, 21, 20, and 21 benchmark functions, respectively, and low values are achieved in 4, 5, 9, 8, 9 and 8 benchmark functions, respectively. Further, the Friedman rank test is conducted to measure the effectiveness of the suggested Es-MFO, and the results are displayed in Table 11. Table 11 shows that the rank of the Es-MFO is the least, which shows that the suggested Es-MFO is the best algorithm among those considered here.

5.6.4.3. Run time complexity of the proposed algorithm. This subsection evaluates the runtime complexity to confirm the time it takes for the algorithm to solve a problem. Using the evaluation procedure described in (Awad, Ali, Suganthan, Liang, & Qu, 2017), the complexity of the algorithm Es-MFO and its comparative algorithms are assessed in terms of run time. First, the code provided below is evaluated for each algorithm to determine the computing time T_0 .

for $i = 1$ to 100000 do

$$x = 0.55 + double(i); x = x + x; x = \frac{x}{2}; x = x \times x;$$

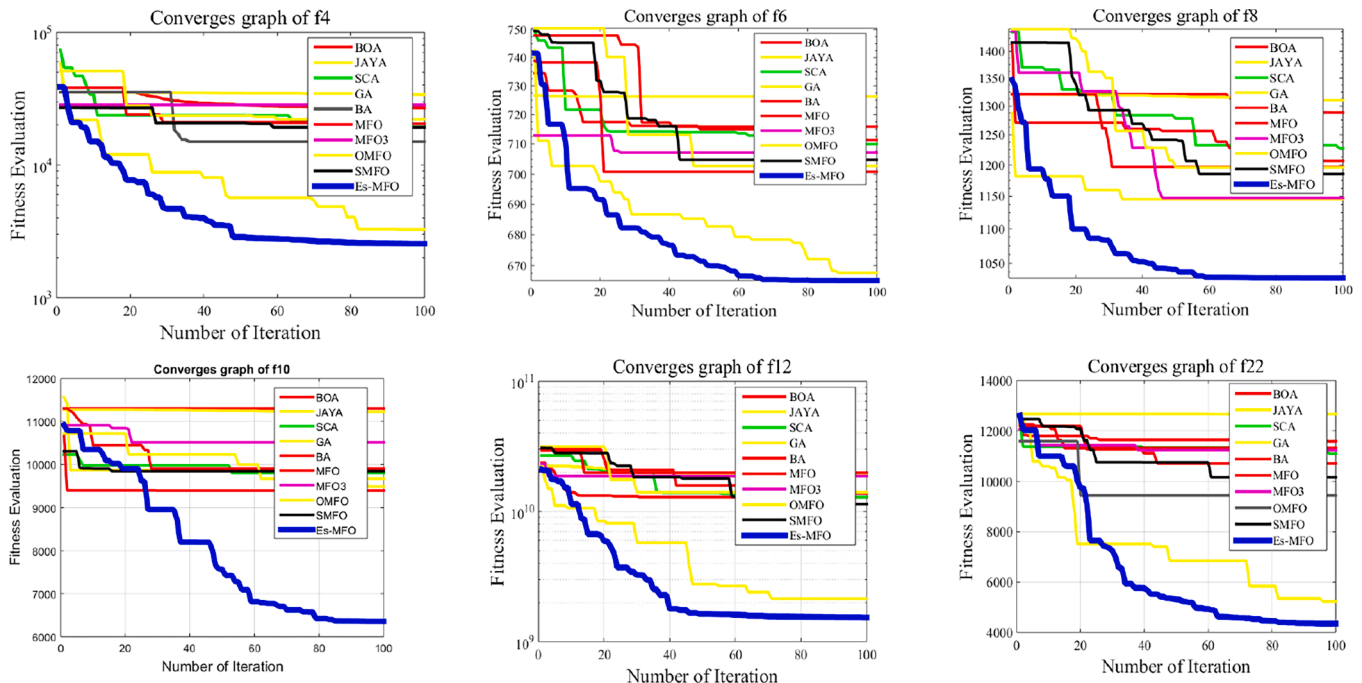


Fig. 8a. Convergence graph of Es-MFO with other algorithms on IEEE CEC 2017 problems for D = 30.

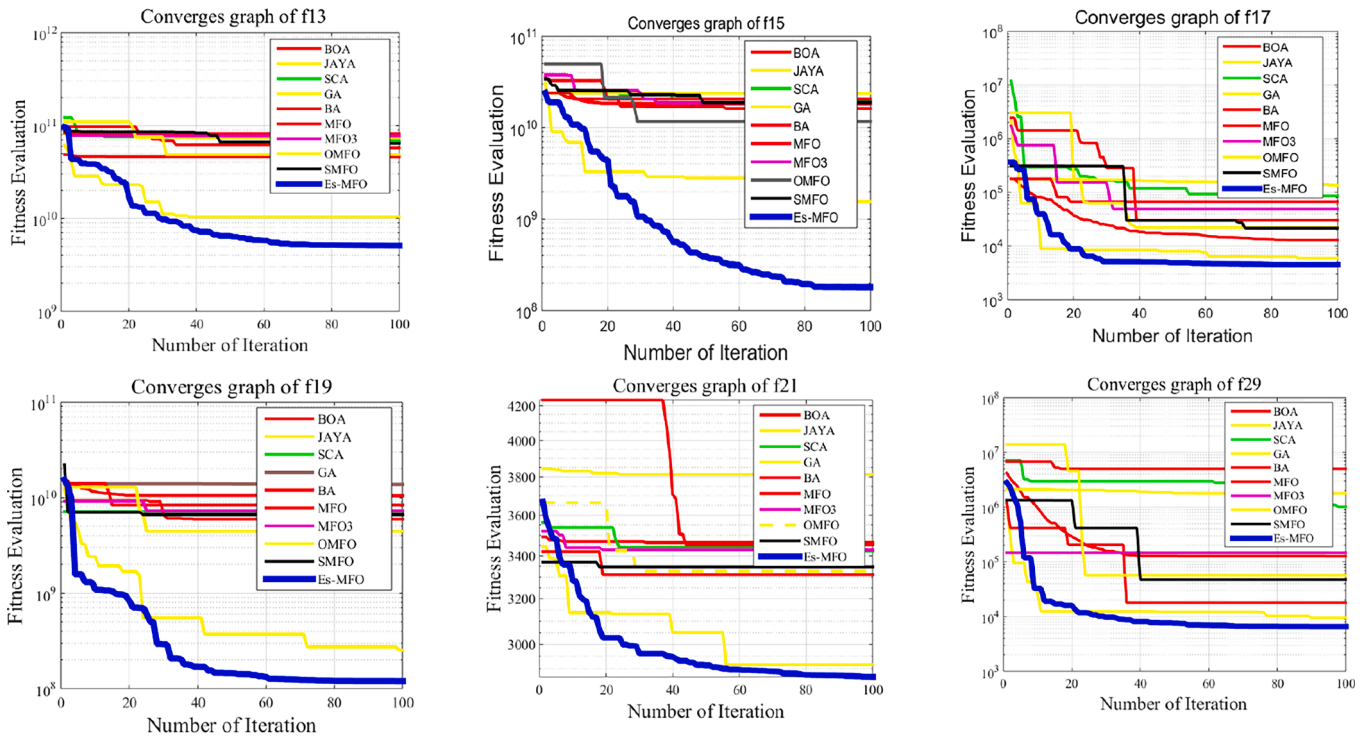


Fig. 8b. Convergence graph of Es-MFO with other algorithms on IEEE CEC 2017 problems for D = 50.

$$x = \text{sqrt}(x); x = \log(x); x = \exp(x); x = \frac{x}{(x+2)}$$

loop

By using the maximum function evaluation of 200,000 in dimension D, T_1 demonstrates the amount of time it took to compute the single function F_{18} from the IEEE CEC 2017 test problem set. T_2 is the average time it takes to run the entire algorithm five times when the same function is used, and function evaluation is 200000. Tables 12a to 12f represent the pairwise comparisons of the algorithm’s estimated runtime complexities for dimension 30. Tables 12a to 12f reveal that, except for FHO and GTO, the proposed Es-MFO has a shorter run time than all of the compared algorithms.

6. Application on CEC2020 engineering design problems

Here, we apply the proposed Es-MFO to three CEC2020 real-world constrained mechanical design optimization problems (Kumar et al., 2020a): weight minimization of a speed reducer problem, multiple disk clutch break design problem and welded beam design problem. Further, the best competitive algorithms of CEC2020 on real-life constrained single objective optimization problems such as sCMaGES (Kumar, Das, & Zelinka, 2020b), SASS (Kumar, Das, & Zelinka, 2020c) and COLSHADE (Gurrola-Ramos et al., 2020) are used for comparison. Appendices 2(a), (b) and (c) give the mathematical formulation of each engineering design problem, which typically involves many restrictions of varying types. When dealing with the restrictions that are utilized to regulate these sorts of problems, the death penalty functions (Coello, 2002) technique is a frequent and straight-forward method. For all of the comparison methods used in the evaluation, the termination conditions are specified at 100,000 function evaluations. All of the chosen algorithms’ parameters were maintained as recommended in the corresponding original study.

6.1. Application 1: Weight minimization of a speed reducer problem

A firm covering encloses the notches, which function independently to reduce or increase speed. The term “speed reducer” is used when this device is used to reduce the speed of any other device. Reducers are commonly used to reduce speed in turbines and rolling mills. The detail descriptions about speed reducer problem are presented in (Sharma et al., 2022).

The algorithm was tested against other competitive algorithms and a few MFO variants. and the comparison results of Es-MFO with other algorithms in terms of best, mean, and standard deviations (SD) are shown in Table 13. We can deduce from Table 13 that the proposed Es-MFO algorithm achieves better results than other algorithms in terms of best and mean values but in standard deviation case, COLSHADE algorithm is winner outperforms the other three contenders. However, the proposed Es-MFO algorithm provides competitive performance for speed reducer design problem in terms of best, mean, and standard deviation. This problem’s mathematical formulation can be found in Appendix 2(a).

6.2. Application 2: Multiple disk clutch break (MDCB) design problem

It’s a type of design challenge encountered in mechanical engineering. The primary goal of this problem is to reduce the total weight of the multiple disc clutch brake system. Moreover, the brief details of MDCB design problem are presented in (Chakraborty et al., 2022b).

The mathematical formulation of the MDCB design problem is presented in Appendix 2(b). The proposed Es-MFO algorithm is used to solve the MDCB problem and compared it with different competitive optimization algorithms in terms of best, mean, and standard deviation, shown in Table 14. From Table 14, we can say that the SASS algorithm obtains better performance in all three formats of statistical measures viz. best, mean, and standard deviation values as compared to all the compared algorithms. Furthermore, the proposed Es-MFO achieves more competitive results than other algorithms with slightly deviated results than SASS algorithm for MDCB problems.

Table 9
Experimental outcomes of Es-MFO and recent algorithms on CEC'17 functions for D = 30.

Func.		Es-MFO	FHO	AOA	GTO	MAMSCA	gQR-BSA	mLBOA
1	Avrg	6.21×10^{10}	5.39×10^{10}	4.63×10^{10}	3.45×10^3	1.69×10^{10}	2.04×10^9	2.55×10^{10}
	Sdev	1.01×10^{10}	1.37×10^{10}	6.37×10^{10}	4.53×10^3	5.47×10^9	1.14×10^{10}	4.88×10^9
3	Avrg	1.25×10^5	9.88×10^4	3.05×10^7	3.00×10^2	6.47×10^4	5.59×10^5	1.37×10^5
	Sdev	3.07×10^4	8.08×10^3	1.26×10^8	$5.62e-06$	6.52×10^3	8.85×10^3	2.19×10^4
4	Avrg	1.66×10^4	1.98×10^4	1.89×10^4	4.75×10^4	3.49×10^4	5.50×10^2	3.47×10^3
	Sdev	3.93×10^3	4.65×10^3	2.60×10^4	3.08×10^1	1.28×10^3	2.22×10^1	8.21×10^2
5	Avrg	8.80×10^2	9.05×10^2	8.81×10^2	9.06×10^2	7.60×10^2	9.22×10^2	7.24×10^2
	Sdev	1.94×10^2	5.29×10^1	2.11×10^2	4.45×10^1	3.87×10^1	3.16×10^1	2.15×10^1
6	Avrg	6.52×10^2	6.76×10^2	6.95×10^2	6.39×10^2	6.74×10^2	6.65×10^2	6.62×10^2
	Sdev	2.82×10^1	1.17×10^1	4.20×10^1	8.07	8.57	5.64	3.42
7	Avrg	1.46×10^3	1.56×10^3	2.27×10^3	1.78×10^3	1.54×10^3	9.26×10^3	1.54×10^3
	Sdev	2.41×10^1	5.00×10^1	7.13×10^2	6.65×10^1	6.06×10^1	5.21×10^1	4.91×10^1
8	Avrg	1.08×10^3	1.16×10^3	1.19×10^3	9.44×10^3	1.42×10^3	9.09×10^3	9.62×10^3
	Sdev	1.50×10^2	2.26×10^1	1.95×10^2	3.04×10^1	2.52×10^1	2.47×10^1	1.66×10^1
9	Avrg	1.25×10^4	1.34×10^4	1.79×10^4	4.10×10^4	4.65×10^4	2.70×10^4	8.43×10^4
	Sdev	1.13×10^4	1.59×10^3	1.40×10^4	6.91×10^2	1.13×10^3	8.44×10^2	1.13×10^3
10	Avrg	6.83×10^3	1.01×10^4	6.62×10^3	5.54×10^3	8.07×10^3	4.56×10^3	5.43×10^3
	Sdev	7.88×10^2	6.38×10^2	2.15×10^3	9.36×10^2	6.80×10^2	6.53×10^2	2.89×10^2
11	Avrg	5.31×10^3	9.45×10^3	1.33×10^4	6.22×10^3	6.46×10^3	6.40×10^3	5.35×10^3
	Sdev	1.95×10^3	1.94×10^3	3.00×10^4	5.03×10^1	1.35×10^3	2.20×10^2	1.33×10^3
12	Avrg	2.31×10^9	6.62×10^9	3.47×10^9	4.06×10^4	1.47×10^9	4.86×10^6	2.40×10^9
	Sdev	7.40×10^9	4.53×10^9	1.00×10^{10}	1.65×10^4	8.21×10^8	3.83×10^6	6.91×10^8
13	Avrg	1.04×10^8	1.47×10^9	9.72×10^9	1.83×10^4	2.52×10^8	8.88×10^8	5.71×10^8
	Sdev	2.24×10^8	1.06×10^9	1.89×10^{10}	1.91×10^4	3.57×10^8	3.67×10^5	6.25×10^7
14	Avrg	1.39×10^6	3.37×10^6	4.73×10^6	1.87×10^6	6.52×10^6	6.79×10^6	1.49×10^6
	Sdev	1.99×10^6	1.83×10^6	1.58×10^7	2.75×10^2	7.07×10^5	8.96×10^4	1.14×10^5
15	Avrg	8.93×10^8	5.10×10^8	1.89×10^9	7.07×10^3	8.77×10^6	9.19×10^3	1.42×10^5
	Sdev	2.25×10^9	6.83×10^8	4.35×10^9	8.47×10^3	2.47×10^7	9.05×10^3	4.10×10^4
16	Avrg	3.23×10^3	5.41×10^3	5.15×10^3	3.83×10^3	3.46×10^3	3.81×10^3	3.94×10^3
	Sdev	3.80×10^2	5.96×10^2	3.41×10^3	2.62×10^2	3.68×10^2	2.96×10^2	2.23×10^2
17	Avrg	2.46×10^3	3.37×10^3	7.27×10^3	2.56×10^3	2.74×10^3	2.75×10^3	2.65×10^3
	Sdev	2.35×10^2	2.16×10^2	1.39×10^4	2.92×10^2	2.44×10^2	1.74×10^2	1.39×10^2
18	Avrg	1.02×10^7	6.83×10^7	1.00×10^8	1.44×10^7	2.74×10^7	3.98×10^7	1.38×10^6
	Sdev	1.54×10^7	7.81×10^7	3.39×10^8	1.14×10^4	4.73×10^6	6.45×10^5	7.87×10^5
19	Avrg	1.33×10^9	3.00×10^8	7.45×10^8	4.56×10^3	2.42×10^7	8.03×10^3	7.89×10^5
	Sdev	2.70×10^9	1.88×10^8	2.14×10^9	2.76×10^3	4.79×10^7	1.02×10^4	4.86×10^5
20	Avrg	2.73×10^3	3.13×10^3	2.98×10^3	2.94×10^3	2.86×10^3	3.88×10^3	3.46×10^3
	Sdev	2.52×10^2	1.15×10^2	4.95×10^2	2.05×10^2	1.15×10^2	2.05×10^2	9.80×10^1
21	Avrg	2.52×10^3	2.65×10^3	2.68×10^3	2.83×10^3	2.54×10^3	2.51×10^3	2.59×10^3
	Sdev	3.36×10^1	4.52×10^1	1.92×10^2	6.03×10^1	3.65×10^1	2.79×10^1	4.73×10^1

(continued on next page)

Table 9 (continued)

Func.		Es-MFO	FHO	AOA	GTO	MAMSCA	gQR-BSA	mLBOA
22	Avrg	6.73×10^3	8.59×10^3	7.82×10^3	7.80×10^3	7.51×10^3	6.91×10^3	7.57×10^3
	Sdev	2.30×10^3	1.21×10^3	2.68×10^3	1.38×10^3	1.87×10^3	1.35×10^3	4.21×10^2
23	Avrg	2.87×10^3	3.42×10^3	3.12×10^3	2.97×10^3	3.07×10^3	3.11×10^3	3.15×10^3
	Sdev	3.65×10^1	1.53×10^2	1.97×10^2	8.95×10^1	5.64×10^1	5.47×10^1	5.25×10^1
24	Avrg	3.05×10^3	3.37×10^3	3.31×10^3	3.42×10^3	3.26×10^3	3.95×10^3	3.40×10^3
	Sdev	4.10×10^1	1.52×10^2	2.64×10^2	8.34×10^1	2.03×10^2	3.33×10^1	7.77×10^1
25	Avrg	6.39×10^3	7.10×10^3	8.77×10^3	2.90×10^3	3.51×10^3	2.95×10^3	3.48×10^3
	Sdev	2.92×10^3	6.07×10^2	8.19×10^3	1.38×10^1	1.79×10^2	2.04×10^1	1.72×10^2
26	Avrg	7.74×10^3	1.02×10^4	1.05×10^4	5.31×10^3	7.26×10^4	7.84×10^3	7.29×10^4
	Sdev	3.44×10^3	1.50×10^3	4.57×10^3	1.55×10^3	7.63×10^2	1.13×10^3	5.86×10^2
27	Avrg	3.29×10^3	3.95×10^3	4.10×10^3	3.30×10^3	3.54×10^3	4.13×10^3	3.63×10^3
	Sdev	2.52×10^1	3.52×10^2	8.67×10^2	7.90×10^1	1.78×10^2	2.64×10^1	7.74×10^1
28	Avrg	7.09×10^3	7.10×10^3	6.97×10^3	3.17×10^3	4.54×10^3	3.33×10^3	4.17×10^3
	Sdev	3.03×10^3	9.09×10^2	5.00×10^3	5.13×10^1	3.55×10^2	3.78×10^1	1.96×10^2
29	Avrg	4.36×10^3	6.60×10^3	3.01×10^4	4.19×10^3	4.60×10^3	4.77×10^3	4.44×10^3
	Sdev	3.06×10^2	1.16×10^3	6.63×10^4	3.84×10^2	3.47×10^2	2.54×10^2	2.26×10^2
30	Avrg	3.97×10^6	6.67×10^8	1.31×10^9	9.67×10^3	4.57×10^7	8.84×10^4	6.67×10^6
	Sdev	3.29×10^6	4.50×10^8	2.80×10^9	3.60×10^3	2.76×10^7	1.13×10^5	3.83×10^6

Table 10
Simulation result for comparison of Es-MFO and basic algorithms for D = 30.

	FHO	AOA	GTO	MAMSCA	gQR-BSA	mLBOA
Superior	25	24	20	21	20	21
Similar	0	0	0	0	0	0
Inferior	4	5	9	8	9	8

Table 11
Friedman's rank test of Es-MFO and basic algorithms with D = 30.

Method	Mean rank	Rank	P-value
FHO	5.28	6	At the 1% level of significance, Ho is ruled out with a P-value of (0.0000.01). At a 1% level of significance, the performance of several approaches differs significantly from one another.
AOA	5.60	7	
GTO	3.05	2	
MAMSCA	3.78	3	
gQR-BSA	3.90	5	
mLBOA	3.85	4	
Es-MFO	2.62	1	

Table 12a
Time complexity according to CEC 2017 benchmark Es-MFO vs FHO.

Dimension	T ₀	T ₁	Es-MFO		FHO	
			T ₂	(T ₂ - T ₁)/T ₀	T ₂	(T ₂ - T ₁)/T ₀
D = 30	0.06	1.64	97.99	1605.83	45.52	731.33

6.3. Application 3: Welded beam design (WBD) problem

The problem of WBD problem is a structural design issue, which has been solved by many researchers. The WB's mathematical representation is shown in Appendix 2(c) respectively. From Appendix- 2 (c), it is clear that the beam has seven constraints, and four variables and detail

Table 12b
Time complexity according to CEC 2017 benchmark Es-MFO vs AOA.

Dimension	T ₀	T ₁	Es-MFO		AOA	
			T ₂	(T ₂ - T ₁)/T ₀	T ₂	(T ₂ - T ₁)/T ₀
D = 30	0.06	1.64	97.99	1605.83	616.73	10215.5

Table 12c
Time complexity according to CEC 2017 benchmark Es-MFO vs GTO.

Dimension	T ₀	T ₁	Es-MFO		GTO	
			T ₂	(T ₂ - T ₁)/T ₀	T ₂	(T ₂ - T ₁)/T ₀
D = 30	0.06	1.64	97.99	1605.83	14.99	222.5

Table 12d
Time complexity according to CEC 2017 benchmark Es-MFO vs MAMSCA.

Dimension	T ₀	T ₁	Es-MFO		MAMSCA	
			T ₂	(T ₂ - T ₁)/T ₀	T ₂	(T ₂ - T ₁)/T ₀
D = 30	0.06	1.64	97.99	1605.83	237.21	3926.66

Table 12e
Time complexity according to CEC 2017 benchmark Es-MFO vs gQR-BSA.

Dimension	T ₀	T ₁	Es-MFO		gQR-BSA	
			T ₂	(T ₂ - T ₁)/T ₀	T ₂	(T ₂ - T ₁)/T ₀
D = 30	0.06	1.64	97.99	1605.83	103.90	1704.33

description are presented in (Sharma et al., 2022). The aim of this issue is to maximize WB's total cost with respect to the constraints of bending stress, shear stress, end deflection, and overhang load, respectively.

Table 12f
Time complexity according to CEC 2017 benchmark Es-MFO vs mLBOA.

Dimension	T_0	T_1	Es-MFO		mLBOA	
			T_2	$(T_2 - T_1)/T_0$	T_2	$(T_2 - T_1)/T_0$
D = 30	0.06	1.64	97.99	1605.83	610.67	10150.5

Table 13
Comparison performance of Es-MFO with other algorithms for speed reducer design problem.

Algorithm	Best	SD	Mean
Es-MFO	2.84376e + 03	1.2100e + 01	2.84101e + 03
MFO	2.91160e + 03	9.5184e + 01	2.91201e + 03
SaDE (Qin & Suganthan, 2005)	2.99442e + 03	3.5420e + 01	2.99442e + 03
SHADE (Tanabe & Fukunaga, 2013)	2.99442e + 03	1.78130e + 01	2.99442e + 03
LSHADE (Tanabe & Fukunaga, 2014)	2.99442e + 03	2.6940e-10	2.99442e + 03
LSHADE-EpSin (Awad et al., 2016b)	2.99442e + 03	8.92779e + 01	3.01072e + 03
COLSHADE	2.99441e + 03	4.64001e-13	2.99441e + 03
SASS	2.99442e + 03	5.91000e-09	2.99442e + 03
sCMAgES	2.99442e + 03	7.66000e-12	2.99442e + 03
EMFO (Sahoo & Saha et al., 2022b)	2.92009e + 03	1.85201e + 01	2.92201e + 03
SMFO	2.87602e + 03	1.86154e + 01	2.88041e + 03
LMFO	2.92341e + 03	1.17113e + 01	2.92102e + 03

Table 14
Comparison performance of Es-MFO with other algorithms for MDCB design problem.

Algorithm	Best	SD	Mean
Es-MFO	2.35242e-01	5.25110e-03	2.38142e-01
MFO	3.36241e-01	1.67142e-01	5.93342e-01
SaDE	23.5340	2.31150e-16	23.5242
SHADE	25.5411	4.51150e-16	23.5242
LSHADE	23.5681	2.31150e-16	23.5242
LSHADE-EpSin	23.5900	3.60150e-16	2.35242
COLSHADE	2.35242e-01	2.35242e-01	2.35242e-01
SASS	2.35241e-01	1.35014e-17	2.35241e-01
sCMAgES	2.35242e-01	1.40143e-16	2.35242e-01
EMFO	2.60451e-01	2.62412e-01	2.62452e-01
SMFO	2.87602e + 03	2.88101e + 03	2.88101e + 03
LMFO	2.60301e-01	5.05291e-03	2.67461e-01

Table 15
Comparison performance of Es-MFO with other algorithms for WB design problem.

Algorithm	Best	SD	Mean
Es-MFO	1.6609	1.88201e-01	1.69502
MFO	1.84282	1.12850e-01	1.96201
SaDE	1.77141	1.652101e-14	1.77141
SHADE	1.72213	3.12840e-12	1.72213
LSHADE	1.74415	3.45310e-13	1.74415
LSHADE-EpSin	1.74631	1.54123e-02	1.74631
SASS	1.67022	7.72000E-14	1.67021
COLSHADE	1.67010	1.69200e-02	1.69632
sCMAgES	1.67022	1.57000e-13	1.67022
EMFO	1.75315	8.43031e-02	1.82407
LMFO	1.82420	4.91302e-02	1.89784

The suggested Es-MFO algorithm is used to solve WBD problem, and its performance is measured with three best performing algorithms of CEC2020 real-world constraint single objective optimization problems with few other traditional optimization algorithms. Table 15 represents the comparison performance of the proposed Es-MFO algorithm with other algorithms. From Table 15, it can be clearly visible that the proposed algorithm provides better results than all other optimization algorithms in terms of 'best' value but in case of mean and standard deviation measures, SASS algorithm is winner. In addition to overall performance, the suggested Es-MFO algorithms achieve a very competitive result as compared to other algorithms in all formats of statistical measures for WB design problem.

7. Application of Es-MFO for COVID-19 CT image segmentation

This section examines the performance of Es-MFO in solving real-world problems by using it to determine the best thresholds for segmenting CT COVID-19 images. Image segmentation methods have recently gained a lot of attention and can be used as a preprocessing phase in various image-processing applications. Selecting suitable threshold values is crucial because image quality depends heavily on them. As a result, multi-level thresholding is considered an optimization problem, and metaheuristic algorithms are often used to solve similar problems by maximizing specific research criteria. The proposed Es-MFO algorithm uses Otsu's method as an objective function to determine the optimal thresholds in COVID-19 CT images.

The section is structured as follows: Section 7.1 presents the objective function. Section 7.2 presents the data set description used in the experiments. The development of the proposed Es-MFO for image segmentation-based Otsu objective function is discussed in Section 7.3. The evaluation criteria used in the comparison are provided in Section 7.4. Section 7.5 reported the results in terms of PSNR, SSIM, and Feature similarity index (FSIM) based on applying Otsu's method. Finally, Section 7.6 provides a Comparison the proposed Es-MFO with other state-of-the-art methods also some recent metaheuristic algorithms.

7.1. Definition of multilevel thresholding segmentation

In this subsection, we will discuss the multi-level thresholding image segmentation problem. Let's assume that the input image is represented by I , which consists of $R + 1$ groups. The main objective of any method for multi-level thresholding is to identify the R thresholds $\{th_k, R = 1, 2, R\}$ required to segment I into sub-groups $(C_R, R = 1, 2, \dots, R)$. The procedure for determining these thresholds is described in Eq. (19)

$$\begin{aligned}
 C_0 &= I'_{ij}, 0 \leq I'_{ij} \leq t_1 - 1, \\
 C_1 &= I'_{ij}, t_1 \leq I'_{ij} \leq t_2 - 1, \\
 &\dots \\
 C_R &= I'_{ij}, t_R \leq I'_{ij} \leq L' - 1
 \end{aligned}
 \tag{19}$$

where t_R denotes the threshold values. I'_{ij} is the gray level for the image. L' represents the total number of grey levels in the images.

Therefore, the challenge of multi-level thresholding can be framed as a task of maximizing the selection of optimum thresholds, stated as:


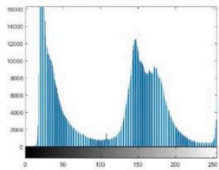

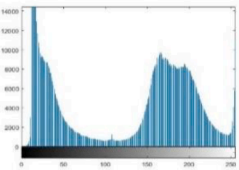
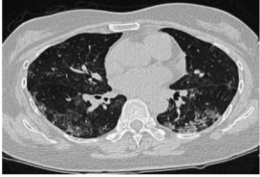
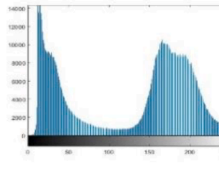

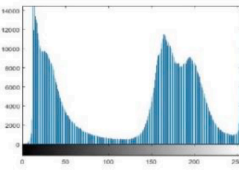

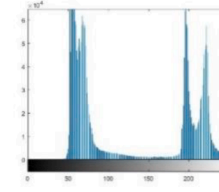

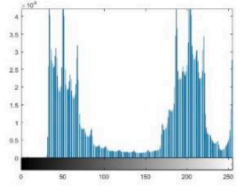

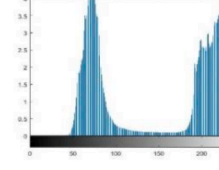

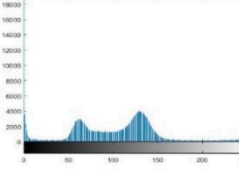
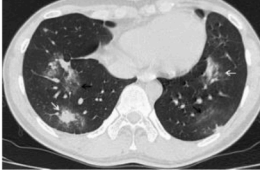
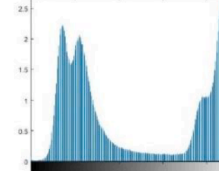

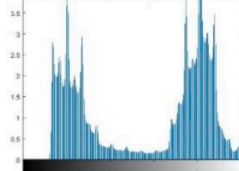
$$t_1^*, t_2^*, t_3^*, \dots, t_R^* = \text{argmax}_{t_1, \dots, t_R} Fitt(t_1, \dots, t_R)
 \tag{20}$$

We utilized Otsu's (Otsu, 1979) method as the objective function $Fitt$ to be maximized. The rationale behind selecting Otsu's method is its extensive usage in solving multi-level thresholding problems in image segmentation.

The definition of the fitness function $Fitt$ is as follows:

$$Fitt = \sum_{i=0}^R \omega_i (\mu_i - \mu_1)^2,
 \tag{21}$$

Table 16
COVID-19 CT image and their corresponding histograms.

			
(a) CT-img1	(b) Histogram of (a)	(c) CT-img2	(d) Histogram of (c)
			
(e) CT-img3	(f) Histogram of (e)	(g) CT-img4	(h) Histogram of (g)
			
(i) CT-img5	(j) Histogram of (i)	(k) CT-img6	(l) Histogram of (k)
			
(m) CT-img7	(n) Histogram of (m)	(o) CT-img8	(p) Histogram of (o)
			
(q) CT-img9	(r) Histogram of (q)	(s) CT-img10	(t) Histogram of (s)

$$\omega_i = \sum_{j=i}^{L_v-1} Ph_j, \quad (22)$$

$$\mu_i = \sum_{j=i}^{L_v-1} i \frac{Ph_j}{\omega_j}, \quad (23)$$

$$Ph_j = \frac{h_j}{N_p} \quad (24)$$

This subsection describes the mathematical model of the Otsu method (Otsu, 1979), which is a thresholding method based on the maximum variance between classes. The segmentation process depends on the image histogram (Glasbey, 1993). The histogram is passed to the Otsu's method, selecting the best thresholding values to divide the image into various classes. This technique assumes the L_v intensity levels of the image, and the probability is obtained by Eq. (25). It can be used for RGB images where Otsu is applied to each layer individually.

$$h_j = \frac{h_j}{N_p}, \sum_{j=1}^{P_n} Ph_j = 1 \quad (25)$$

where j is an intensity level in $(0 \leq j \leq L_v - 1)$ and P_n is the total number of pixels. h_j is the number of intensity frequencies j in the image denoted by the histogram. In a probability distribution Ph_j , the histogram is normalized. The classes for bi-level segmentation are computed based on the probability distribution as follows:

$$C_1 = \frac{Ph_1}{\omega_0(th)}, \dots, \frac{Ph_{th}}{\omega_0(th)} \text{ and } C_2 = \frac{Ph_{th+1}^c}{\omega_1(th)}, \dots, \frac{Ph_{L_v}}{\omega_1(th)} \quad (26)$$

where $\omega_0(th)$ and $\omega_1(th)$ are probabilities distributions for C_1 and C_2 that is defined by Eq. (27).

$$\omega_0(th) = \sum_{j=1}^{th} Ph_j \text{ and } \omega_1(th) = \sum_{th+1}^{L_v} Ph_j \quad (27)$$

It is essential to calculate the mean levels μ_0 and μ_1 that describe the classes by Eq. (28). After those operators have been determined, the

Otsu between-class σ_B^2 is computed by Eq. (29).

$$\mu_0 = \sum_{j=1}^{th} \frac{iPh_j}{\omega_0(th)} \text{ and } \mu_1 = \sum_{j=th+1}^{L_v} \frac{Ph_j}{\omega_1(th)} \quad (28)$$

$$\sigma_B^2 = \sigma_1 + \sigma_2 \quad (29)$$

Notice that σ_1 and σ_2 in Eq. (29) are the variances of C_1 and C_2 that identified as follows:

$$\sigma_1 = \omega_0(\mu_0 + \mu_y)^2 \text{ and } \sigma_2 = \omega_1(\mu_1 + \mu_y)^2 \quad (30)$$

where $\mu_y = \omega_0\mu_0 + \omega_1\mu_1$. Based on the values σ_1 and σ_2 , Eq. (30) illustrates the objective function. As a result, the optimization problem is simplified to determining the intensity level that maximizes Eq. (31).

$$F_{otsu}(th) = \text{Max}(\sigma_B^2(th)), 0 \leq th \leq L_v - 1 \quad (31)$$

where $\sigma_B^2(th)$ is the Otsu for a provided th value. Otsu's approach is used for a single layer from an image, which means that it is required to apply it for the three layers for color images. The preceding idea of a bilevel technique can be changed to accommodate multiple thresholds. The fitness function $F_{otsu}(th)$ in Eq. (31) can be changed for multiple thresholds as the following:

$$F_{otsu}(TH) = \text{Max}(\sigma_B^2(th)), 0 \leq th_i \leq L_v - 1, i = [1, 2, 3, \dots, K] \quad (32)$$

where $TH = [th_1, th_2, \dots, th_{k-1}]$, is a vector containing multiple thresholding and the variances are calculated by Eq. (33).

$$\sigma_B^2 = \sum_{i=1}^k \sigma_i = \sum_{i=1}^k \omega_i(\mu_i - \mu_y)^2 \quad (33)$$

where i denotes the specific class, ω_i and μ_j are the probability of occurrence and the mean of a class, respectively. For multiple thresholds values are obtained as:

$$\omega_{k-1}(th) = \sum_{j=th_{k-1}+1}^{L_v} Ph_j \quad (34)$$

for mean values:

$$\mu_{k-1} = \sum_{j=th_{k-1}+1}^{L_v} \frac{Ph_j}{\omega_1(th_k)} \quad (35)$$

7.2. Discussion on datasets

In this research, the proposed algorithm was evaluated using CT images from the COVID-19 dataset (Zhao et al., 2020a,2020b,2020c). The 216 patients included in the CT COVID-19 dataset are represented by 349 CT scans. Patients aged 40 to 84 of both sexes were represented in the COVID-19 images. To gauge the efficacy of the proposed algorithm, evaluation images from a variety of patients are used. The test images have names like CT-img1, CT-img2, etc., up to CT-img10. The selected test images and their corresponding histograms are shown in Table 16.

7.3. Es-MFO implementation-based image segmentation

The Es-MFO algorithm for image segmentation starts by converting the original CT image, Im_R , to a grayscale image, IO_G , and generating histograms for both images. Otsu's method is then used as the fitness function to calculate the fitness value. The process begins by setting up the parameters for ES-MFO and generating a population of N search agents with Dim dimensions. The best solution is identified from this population, and the remaining solutions are modified using ES-MFO operators outlined in Section 4. The best threshold value determines

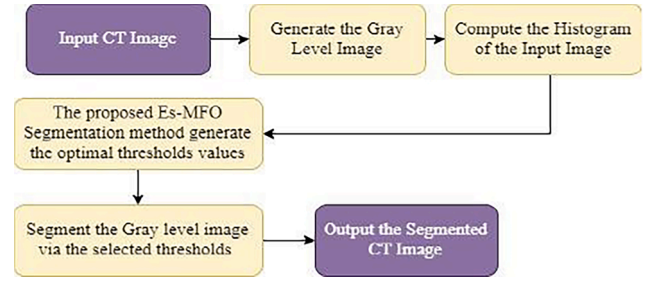


Fig. 9. Image segmentation process-based Es-MFO algorithm.

the optimal solution. Lastly, the most effective thresholds are chosen and applied to the CT images. The image segmentation process of the proposed method is illustrated in Fig. 9. The steps for developing the proposed ES-MFO algorithm for selecting the best thresholds in CT COVID-19 images using the Otsu fitness function can be summarized as follows:

1. Convert the original CT image, Im_R , to a grayscale image, IO_G .
2. Generate histograms for both the original image and the grayscale image.
3. Set the control parameters for the ES-MFO algorithm.
4. Create a set of N particles, X , each with Dim dimensions.
5. Evaluate each particle in X using Otsu's fitness function Eq. (31).
6. Sort both the moth and flame matrix based on their fitness values and update the number of flames using Eq. (8).
7. Update the position and velocity of each particle in X using Eqs. (6), and 7).
8. Update the role of the moths concerning the corresponding flames using Eq. (10).
9. To generate a new solution, randomly choose between using Eq. (13) or the FSM method Eqs. (10), 11, and 12), and then evaluate the fitness value of the new solution. The best fitness value represents the optimal solution.
10. Increase the iteration counter by 1 and check if the stop condition is met. If not, go back to Step 5.
11. Return the best solution containing the best thresholds and apply them to the CT COVID-19 images.

7.4. Performance measures

To evaluate the effectiveness of Es-MFO in image segmentation, three metrics are employed: Peak-signal-to-noise ratio (PSNR), Structural similarity index (SSIM), and Feature similarity index (FSIM). The definitions of these metrics are as follows:

- **PSNR:** This measure calculates the dissimilarity between the original image I_{org} and segmented image I_s , it is defined by Eq. (36).

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE} \right) \quad (36)$$

where $RMSE$ is the mean square error calculated using Eq. (37), M and N denote the number of raw and columns in the image.

$$RMSE = \sqrt{\frac{\sum_{i=1}^N \sum_{j=1}^M (I_{org}(i,j) - I_s(i,j))^2}{N \times M}} \quad (37)$$

- **SSIM:** Is calculated using Eq. (38) and this measure obtains the similarity between the original and the segmented image.

$$SSIM(I, I_s) = \frac{(2\mu_I\mu_{I_s} + k_1)(2\sigma_I\sigma_{I_s} + k_2)}{(\mu_I^2 + \mu_{I_s}^2 + k_1)(\sigma_I^2 + \sigma_{I_s}^2 + k_2)}, k_1 = 6.5025, k_2 = 58.52252 \quad (38)$$

- **FSIM:** This metric determines how similar two images are based on their internal characteristics (Sara et al., 2019) defined by Eq.(39)

$$FSIM = \frac{\sum_{w \in \Omega} S_L(w) PC_m(w)}{\sum_{w \in \Omega} PC_m(w)} \tag{39}$$

where ω is the entire domain of the image:

$$S_L(w) = S_{PC}(w) \times S_G(w)$$

$$S_{PC}(w) = \frac{2PC_1(w) \times PC_2(w) + T_1}{PC_1^2(w) + PC_2^2(w) + T_1} \tag{40}$$

$$S_G(w) = \frac{2G_1(w) \times G_2(w) + T_1}{G_1^2(w) + G_2^2(w) + T_1}$$

and G is the gradient magnitude (GM) of an image and is defined as:

$$G = \sqrt{G_x^2 + G_y^2} \tag{41}$$

$$PC(w) = \frac{E(w)}{(\epsilon + \sum_n A_n(w))} \tag{42}$$

The value of the vector in w on n is $E(w)$ and $A_n(w)$ is the local amplitude of scale n . ϵ is a small number and $PC_m(w) = \max(PC_1(w), PC_2(w))$.

Note that the high values of $PSNR$, $SSIM$, and $FSIM$ demonstrate the high performance of the algorithm.

7.5. Results and discussions on image segmentation problems

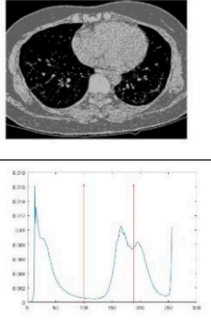
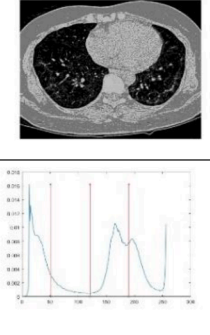
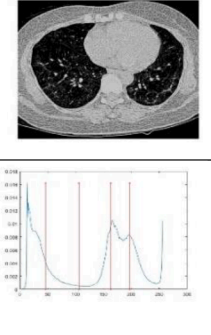
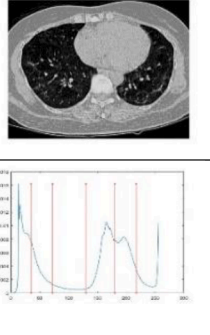
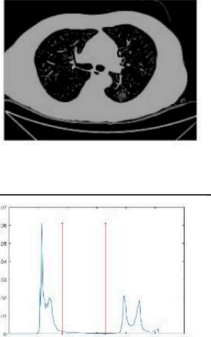
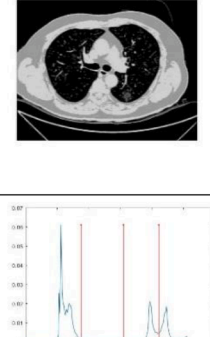
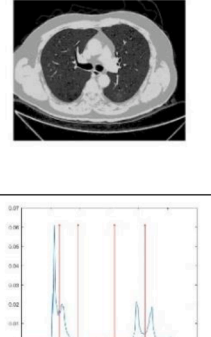
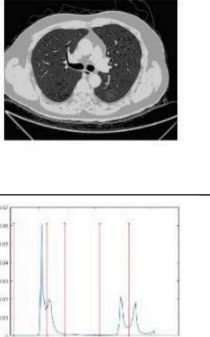
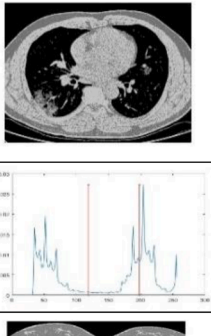
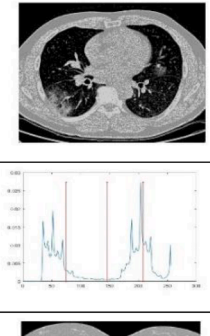
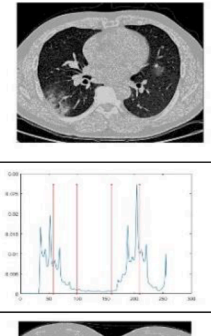
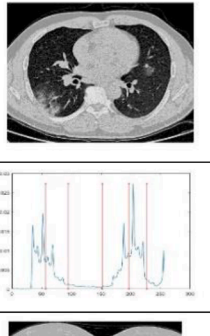
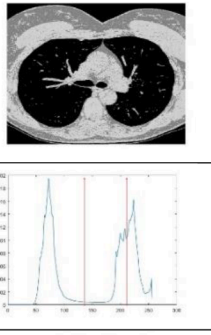
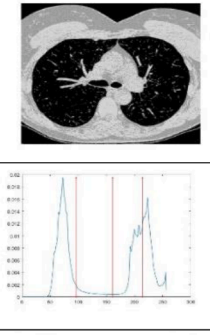
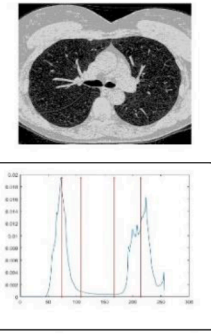
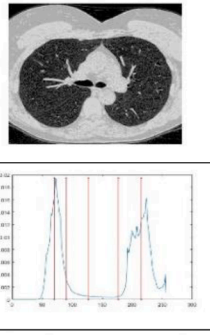
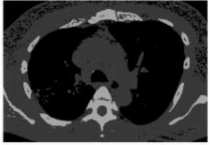
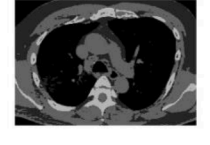


In this subsection, the effectiveness of the Es-MFO based COVID-19 image segmentation method is discussed and evaluated in comparison to eight metaheuristic algorithms. The $PSNR$, $SSIM$, and $FSIM$ are used to evaluate the outcomes obtained by Es-MFO. This evaluation is carried out across a range of threshold values ($nTh = 2, 3, 4, \text{ and } 5$). All algorithms were tested 30 times per image for a total of 350 iterations with a population size of 50. Test images of CT scans are shown in [Table 17](#)

Table 17
Segmented images and thresholding values achieved by Es-MFO algorithm over the image’s histograms.

Test-Image	nTh = 2	nTh = 3	nTh = 4	nTh = 5
CT-img1				
CT-img2				
CT-img3				

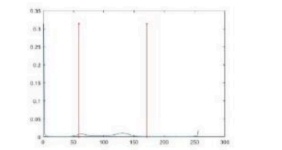
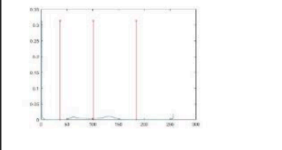
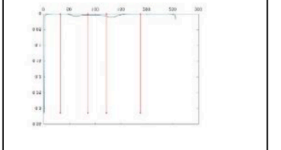
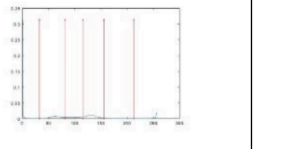
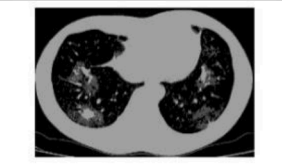



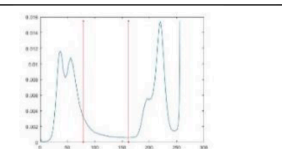
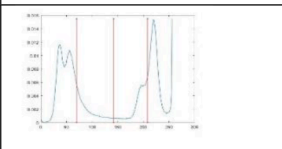
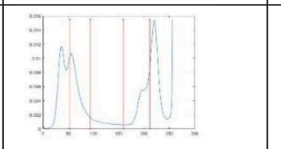
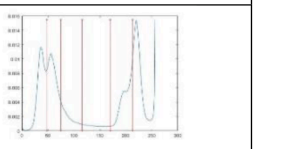
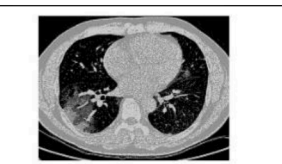
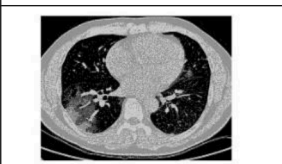
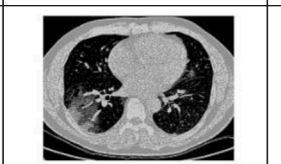
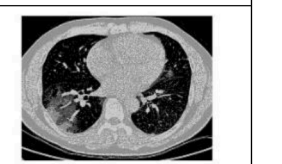

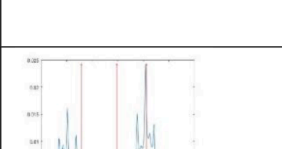
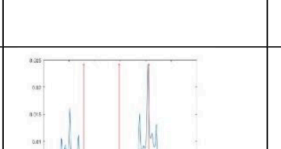
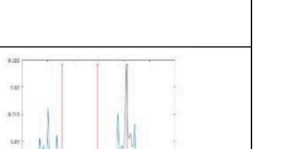
(continued on next page)

Table 17 (continued)

<p>CT-img4</p>				
<p>CT-img5</p>				
<p>CT-img6</p>				
<p>CT-img7</p>				
				

(continued on next page)

Table 17 (continued)

CT-img8				
CT-img9				
				
CT-img10				
				

after being segmented by Es-MFO using varying thresholds [nTh = 2, 3, 4, 5]. Positions of the optimally determined thresholds over the histograms of the individual images are also displayed. The average PSNR, SSIM, and FSIM results are presented in Tables 18–20, respectively. Higher algorithm values indicate greater precision and efficiency.

In conclusion, the following observations from the experiments should be noted.

- **In terms of PSNR results:** Table 18 presents the PSNR values. The higher the value, the better the quality segmentation. The best values are highlighted in bold. The proposed Es-MFO algorithm achieved high PSNR values for some test images. For more details, concerning CT-img1 and CT-img10 test images, Es-MFO has significant values for all thresholding levels. For CT-img4, CT-img6, CT-img7, and CT-img8 test images, Es-MFO has significant values for three thresholding levels. Moreover, for CT-img2 and CT-img3 Es-MFO has significant values for two thresholding levels. The SSA and MFO3 algorithms have only four significant values in all test images. Meanwhile, the PSO and the original MFO algorithms acquired only two higher PSNR values among all test images. In contrast, the OMFO and SMFO algorithms have only one higher value. Also, the SCA algorithm does not produce any best value in either image.

Overall, Es-MFO obtains higher PSNR values than other algorithms at most threshold levels in all test images.

- **In terms of SSIM results:** Es-MFO outperforms the original MFO and all other algorithms as shown in Table 19. The cultivated SSIM values in the Es-MFO are observed to be better than all algorithms, indicating relevance in all images for most thresholding levels. The other algorithms have greater SSIM values and are ranked as follows: OMFO, MFO3, MFO, SSA, SMFO, PSO, and SCA.
- **In terms of FSIM results:** The average values of the FSIM metrics are summarized in Table 20.
- This statistic measures and evaluates how well image characteristics are preserved after processing. Results with the highest segmentation quality are highlighted in bold. According to Table 20, Es-MFO performs better than MFO on all test images across the board. More than that, it outperforms every other algorithm we've seen.

7.6. Comparison the proposed Es-MFO with other state-of-the-art methods

In this subsection, we compare our proposed image segmentation method, which is based on the Es-MFO algorithm, with other state-of-the-art methods and recent metaheuristic algorithms that have been applied to the same image segmentation problem using the COVID-19

Table 18
Comparison between Es-MFO and all other algorithms according to the PSNR mean values.

Test Image	nTh	PSO	SCA	SSA	MFO3	OMFO	SMFO	MFO	Es-MFO
CT-img1	2	15.0052	15.0471	15.0675	15.0674	15.0697	15.0258	15.0675	15.0775
	3	16.7621	17.0221	16.9185	16.8067	16.7639	16.8903	16.9613	16.9613
	4	18.1970	18.2193	18.5102	18.4917	18.4377	18.3427	18.5002	18.5202
	5	20.0481	19.2041	20.0893	20.1073	20.0446	19.8716	20.3419	20.3519
CT-img2	2	14.4777	14.4336	14.4228	14.4184	14.4145	14.4229	14.4228	14.4228
	3	16.9225	16.9349	17.0347	16.9544	16.9484	16.8457	17.0347	17.0447
	4	18.3257	17.9943	18.3364	18.4511	18.3033	18.2096	18.3344	18.3344
	5	19.9959	19.3617	20.7083	20.5428	20.4410	20.6095	20.6474	20.7474
CT-img3	2	14.1268	14.3051	14.3576	14.3085	14.2889	14.2668	14.3576	14.9576
	3	16.6705	16.8777	16.8872	16.8953	16.9467	16.8370	16.8898	16.8898
	4	18.2682	18.2244	18.2690	18.3615	18.1678	18.2304	18.2664	18.2664
	5	19.9811	19.4277	20.3898	20.2233	20.3352	20.1249	20.4186	20.4396
CT-img4	2	14.5523	14.3664	14.3761	14.3734	14.3777	14.3284	14.3712	14.5523
	3	16.4688	16.6989	16.7716	16.8183	16.7512	16.7397	16.7656	16.8656
	4	18.0119	17.6707	17.9301	17.7573	18.0427	18.0493	17.9271	17.9971
	5	19.8405	19.0746	19.9558	19.9678	20.1101	19.8387	19.9691	21.9691
CT-img5	2	13.0420	13.0534	13.0621	13.0546	13.0613	13.0207	13.0621	14.0621
	3	13.8594	13.8742	13.9468	13.9396	13.9656	13.9575	13.9456	14.9456
	4	15.8259	16.1716	16.3784	16.4974	16.2324	16.0447	16.3784	16.4784
	5	16.7226	16.4266	16.7815	16.5415	16.6606	16.1151	16.6599	16.6599
CT-img6	2	13.8286	14.0259	14.0568	14.0293	14.0386	14.0273	14.0568	14.0568
	3	15.5513	15.2639	15.2676	15.3385	15.2458	15.3920	15.2676	16.2676
	4	16.9693	16.6406	16.9492	16.4845	16.6101	16.7904	16.6072	17.6772
	5	18.7863	18.0785	18.7757	18.5796	18.5945	18.8578	18.7791	18.8091
CT-img7	2	12.5174	12.6076	12.6340	12.6003	12.6050	12.6290	12.6340	12.9340
	3	13.7789	13.6920	13.8159	13.8216	13.8235	13.8074	13.8142	14.7122
	4	15.4501	15.3016	16.1851	16.4913	16.4329	15.7005	16.1115	16.3525
	5	16.6744	16.9300	16.8007	16.6797	16.5720	16.8236	16.6925	16.7725
CT-img8	2	14.5820	14.6452	14.6385	14.6486	14.6371	14.6331	14.6385	14.6999
	3	18.2476	18.4619	18.5441	18.5309	18.4965	18.4050	18.5384	18.5384
	4	20.1282	20.1620	20.5106	20.4251	20.4434	20.2280	20.5043	20.5143
	5	21.7129	20.8123	21.9502	21.7486	21.8860	21.7354	21.9239	21.9539
CT-img9	2	13.7146	13.7650	13.7472	13.7659	13.7805	13.7706	13.7472	13.7472
	3	16.1144	16.2289	16.1689	16.1939	16.1403	16.3176	16.1862	16.1962
	4	18.2480	18.1804	18.7022	18.6980	18.7497	18.6103	18.7131	19.7131
	5	19.2710	18.9611	19.6066	19.0297	19.1610	19.4551	19.4612	19.8622
CT-img10	2	14.1456	14.0788	14.0602	14.0826	14.0749	14.0691	14.0602	14.1602
	3	15.6114	15.6299	15.5026	15.6898	15.6538	15.5790	15.5176	16.5176
	4	17.2005	17.5753	17.6194	17.4701	17.6405	17.3343	17.5715	17.5753
	5	18.6476	17.9672	18.9433	18.5777	18.7826	18.8366	18.9396	19.4396

Table 19
Comparison between Es-MFO and all other algorithms according to the SSIM mean values.

Test Image	nTh	PSO	SCA	SSA	MFO3	OMFO	SMFO	MFO	Es-MFO
CT-img1	2	0.6264	0.6278	0.6279	0.6283	0.6283	0.6275	0.6279	0.6279
	3	0.7004	0.7206	0.7158	0.7096	0.7057	0.7123	0.7184	0.7418
	4	0.7727	0.7745	0.7847	0.7835	0.7786	0.7758	0.7846	0.7847
	5	0.8201	0.8106	0.8201	0.8225	0.8251	0.8143	0.8319	0.8454
CT-img2	2	0.6518	0.6514	0.6511	0.6512	0.6510	0.6510	0.6511	0.6614
	3	0.7549	0.7625	0.7623	0.7618	0.7600	0.7557	0.7623	0.7680
	4	0.8003	0.7987	0.8039	0.8012	0.8029	0.7998	0.8038	0.8039
	5	0.8469	0.8471	0.8779	0.8634	0.8680	0.8772	0.8794	0.8843
CT-img3	2	0.6476	0.6534	0.6548	0.6535	0.6531	0.6526	0.6548	0.6548
	3	0.7643	0.7690	0.7681	0.7695	0.7692	0.7702	0.7681	0.7761
	4	0.8016	0.8110	0.8108	0.8118	0.8057	0.8099	0.8107	0.8108
	5	0.8580	0.8541	0.8687	0.8612	0.8567	0.8569	0.8620	0.8839
CT-img4	2	0.6428	0.6405	0.6404	0.6409	0.6409	0.6395	0.6402	0.6404
	3	0.7530	0.7666	0.7656	0.7676	0.7669	0.7644	0.7657	0.7680
	4	0.7951	0.8041	0.8077	0.8036	0.8067	0.8051	0.8076	0.8079
	5	0.8553	0.8535	0.8677	0.8603	0.8654	0.8645	0.8755	0.8852
CT-img5	2	0.5158	0.5169	0.5168	0.5172	0.5173	0.5165	0.5168	0.5170
	3	0.5300	0.5301	0.5301	0.5304	0.5320	0.5333	0.5301	0.5405
	4	0.6573	0.6793	0.6960	0.7027	0.6875	0.6731	0.6960	0.7063
	5	0.7046	0.6959	0.7163	0.7019	0.7119	0.6778	0.7125	0.7320
CT-img6	2	0.5715	0.5724	0.5747	0.5729	0.5734	0.5744	0.5747	0.5758
	3	0.6822	0.6734	0.6744	0.6740	0.6732	0.6795	0.6744	0.6744
	4	0.7499	0.7424	0.7540	0.7221	0.7317	0.7432	0.7300	0.7975
	5	0.8318	0.8025	0.8323	0.8220	0.8213	0.8359	0.8325	0.8431
CT-img7	2	0.5031	0.5070	0.5085	0.5065	0.5067	0.5077	0.5085	0.5122
	3	0.5905	0.5842	0.5921	0.5926	0.5924	0.5921	0.5920	0.5921
	4	0.6953	0.6851	0.7386	0.7523	0.7497	0.7089	0.7341	0.7576
	5	0.7476	0.7648	0.7690	0.7626	0.7561	0.7697	0.7657	0.7873
CT-img8	2	0.7373	0.7398	0.7400	0.7398	0.7398	0.7400	0.7400	0.7400
	3	0.8539	0.8602	0.8623	0.8622	0.8612	0.8566	0.8621	0.8690
	4	0.8836	0.8879	0.8959	0.8929	0.8931	0.8875	0.8962	0.8958
	5	0.9092	0.8985	0.9157	0.9110	0.9137	0.9112	0.9153	0.9158
CT-img9	2	0.5624	0.5645	0.5640	0.5644	0.5645	0.5641	0.5640	0.5640
	3	0.6277	0.6271	0.6218	0.6242	0.6186	0.6357	0.6231	0.6357
	4	0.7423	0.7353	0.7577	0.7582	0.7613	0.7555	0.7592	0.7591
	5	0.7841	0.7754	0.7939	0.7689	0.7724	0.7900	0.7890	0.8049
CT-img10	2	0.6088	0.6037	0.6024	0.6040	0.6034	0.6048	0.6024	0.6085
	3	0.7067	0.7009	0.6971	0.7032	0.7018	0.7025	0.6976	0.7077
	4	0.7819	0.8057	0.8092	0.8005	0.8100	0.7897	0.8065	0.8204
	5	0.8391	0.8187	0.8537	0.8374	0.8443	0.8466	0.8537	0.8579

Table 20
Comparison between Es-MFO and all other algorithms according to the FSIM mean values.

Test Image	nTh	PSO	SCA	SSA	MFO3	OMFO	SMFO	MFO	Es-MFO
CT-img1	2	0.8101	0.8104	0.8100	0.8104	0.8104	0.8107	0.8100	0.8130
	3	0.8639	0.8750	0.8740	0.8712	0.8689	0.8716	0.8753	0.8867
	4	0.9131	0.9141	0.9241	0.9230	0.9204	0.9166	0.9240	0.9286
	5	0.9324	0.9271	0.9385	0.9377	0.9372	0.9343	0.9404	0.9421
CT-img2	2	0.8101	0.8109	0.8108	0.8109	0.8109	0.8109	0.8108	0.8108
	3	0.8835	0.8921	0.8919	0.8911	0.8901	0.8867	0.8919	0.8945
	4	0.9149	0.9181	0.9251	0.9211	0.9232	0.9205	0.9250	0.9351
	5	0.9383	0.9378	0.9539	0.9485	0.9492	0.9533	0.9543	0.9590
CT-img3	2	0.7989	0.8024	0.8031	0.8025	0.8024	0.8029	0.8031	0.8125
	3	0.8785	0.8823	0.8817	0.8820	0.8814	0.8827	0.8817	0.8943
	4	0.9026	0.9127	0.9134	0.9131	0.9099	0.9106	0.9134	0.9134
	5	0.9314	0.9299	0.9413	0.9364	0.9344	0.9342	0.9386	0.9474
CT-img4	2	0.7763	0.7794	0.7789	0.7797	0.7796	0.7793	0.7789	0.7795
	3	0.8613	0.8729	0.8727	0.8734	0.8736	0.8695	0.8729	0.8729
	4	0.8944	0.9025	0.9060	0.9041	0.9048	0.9025	0.9059	0.9066
	5	0.9244	0.9245	0.9345	0.9319	0.9336	0.9323	0.9372	0.9403
CT-img5	2	0.8396	0.8415	0.8416	0.8415	0.8416	0.8411	0.8416	0.8458
	3	0.8610	0.8714	0.8778	0.8763	0.8777	0.8741	0.8777	0.8778
	4	0.8428	0.8396	0.8384	0.8357	0.8325	0.8455	0.8384	0.8410
	5	0.8494	0.8419	0.8558	0.8489	0.8480	0.8551	0.8552	0.8655
CT-img6	2	0.7813	0.7897	0.7910	0.7903	0.7905	0.7909	0.7910	0.7910
	3	0.8517	0.8524	0.8536	0.8536	0.8530	0.8528	0.8536	0.8630
	4	0.8848	0.8814	0.8903	0.8874	0.8906	0.8908	0.8919	0.8920
	5	0.9082	0.9042	0.9189	0.9126	0.9130	0.9157	0.9189	0.9192
CT-img7	2	0.7771	0.7850	0.7861	0.7848	0.7850	0.7849	0.7800	0.7861
	3	0.8347	0.8362	0.8429	0.8432	0.8433	0.8407	0.8428	0.8440
	4	0.8402	0.8365	0.8266	0.8318	0.8289	0.8356	0.8282	0.8373
	5	0.8549	0.8405	0.8483	0.8499	0.8539	0.8507	0.8490	0.8489
CT-img8	2	0.6529	0.6479	0.6483	0.6476	0.6478	0.6500	0.6483	0.6493
	3	0.7648	0.7690	0.7711	0.7709	0.7700	0.7677	0.7709	0.7730
	4	0.8276	0.8306	0.8414	0.8380	0.8380	0.8332	0.8414	0.8513
	5	0.8628	0.8454	0.8689	0.8666	0.8665	0.8665	0.8687	0.8690
CT-img9	2	0.7113	0.7132	0.7128	0.7131	0.7132	0.7130	0.7128	0.7130
	3	0.7753	0.7908	0.7941	0.7928	0.7927	0.7852	0.7940	0.8021
	4	0.8183	0.8195	0.8282	0.8271	0.8283	0.8267	0.8287	0.8291
	5	0.8499	0.8366	0.8629	0.8531	0.8565	0.8605	0.8629	0.8633
CT-img10	2	0.7922	0.8005	0.8002	0.8006	0.8005	0.7960	0.8002	0.8102
	3	0.8524	0.8603	0.8591	0.8613	0.8611	0.8587	0.8592	0.8633
	4	0.8852	0.8876	0.8899	0.8871	0.8877	0.8889	0.8901	0.8999
	5	0.9093	0.8976	0.9212	0.9112	0.9149	0.9159	0.9212	0.9213

Table 21
Comparison between Es-MFO, MRFO, MRFO-OBL, MPA, RSA, and RUN methods according to the PSNR mean values for nTh = [7, 8, 9, 10].

Test Image	nTh	MRFO	MRFO-OBL	MPA	RUN	RSA	Es-MFO
CT-img1	7	22.7883	22.8366	21.4176	22.9134	22.7892	22.9892
	8	23.5724	23.6096	22.9087	23.5764	23.6032	23.8832
	9	24.4058	24.4777	23.6482	24.3107	24.4052	24.5552
	10	25.0747	25.075	24.4046	25.1588	25.0975	25.9575
CT-img2	7	23.3543	23.3691	21.9528	23.3242	23.4015	23.7615
	8	24.25	24.2945	23.3681	24.2164	24.2553	25.2553
	9	25.2425	25.2675	24.3766	24.9997	25.2257	25.2257
	10	25.9906	26.0447	25.3463	25.8697	25.9719	26.9719
CT-img3	7	23.3779	23.3835	21.9165	23.2441	23.3324	23.4024
	8	24.2597	24.292	23.3554	24.2523	24.2282	24.2585
	9	25.2401	25.2815	24.4217	24.9603	25.2046	25.2346
	10	26.0143	26.0406	25.3383	25.8116	26.0096	26.0896
CT-img4	7	23.0455	23.1315	21.5663	22.9767	23.0369	23.0169
	8	24.1247	24.1701	23.0272	24.1220	24.0187	24.0787
	9	25.2826	25.3379	24.1972	24.9683	25.3512	26.3011
	10	26.0265	26.086	25.3168	25.7838	25.0861	26.9861
CT-img5	7	17.6816	17.6923	17.3386	18.1540	17.7089	17.7789
	8	18.0973	18.1808	17.7441	19.0159	18.1785	19.1285
	9	18.4984	18.6158	17.8638	19.2159	18.6222	18.6322
	10	18.9574	19.0474	18.4492	20.4894	19.1864	19.1864
CT-img6	7	21.0786	21.1912	19.3529	21.7825	21.0613	21.0813
	8	22.2276	22.2106	20.4640	22.8256	22.2209	22.4809
	9	22.9386	23.009	22.6400	24.1204	23.0058	23.2858
	10	23.5627	23.686	23.2842	24.9278	23.6631	23.7631
CT-img7	7	19.4133	19.6895	18.4787	19.3341	19.4123	19.4123
	8	20.7551	20.702	18.8272	20.7072	20.3942	20.8042
	9	21.4188	21.8512	20.3691	21.5553	21.4879	21.5879
	10	22.6882	22.9905	20.9665	22.4265	22.6595	22.5595
CT-img8	7	24.0661	24.1512	23.5801	24.8565	24.11	24.21
	8	25.4969	25.6688	24.5144	26.1716	25.5404	26.6404
	9	27.0617	27.05	25.4224	26.8701	27.1236	27.2236
	10	27.9826	27.9897	27.5670	27.7323	27.1206	28.0906
CT-img9	7	21.7791	21.7893	20.7888	21.8192	21.5192	21.9192
	8	22.5405	22.6872	21.8236	22.8955	22.5364	22.9064
	9	23.6402	23.7957	22.3841	24.4842	23.5541	24.5541
	10	24.7814	25.0439	23.3700	25.5631	24.8784	25.8784
CT-img10	7	20.4594	20.6069	19.5189	21.0905	20.3524	20.9524
	8	22.0541	22.1712	20.4965	22.2978	22.2284	22.3259
	9	23.0701	23.085	22.8955	23.1557	23.0357	23.6357
	10	23.5813	23.6204	23.7143	24.1704	23.1276	24.4256

Table 22
Comparison between Es-MFO and I-EO according to PSNR mean values for nTh = [2, 3, 4, 5, 10].

Image	nTh	I-EO	ES-MFO	Image	Th	I-EO	ES-MFO
CT-img1	2	12.5000	15.0775	CT-img6	2	11.9000	14.0568
	3	15.7000	16.9613		3	17.5000	16.2676
	4	16.9000	18.5202		4	17.3000	17.6772
	5	19.0000	20.3519		5	18.6000	18.8091
	10	24.6000	25.9575		10	25.2000	23.7631
CT-img2	2	14.3000	14.4228	CT-img7	2	13.5000	12.934
	3	15.9000	17.0447		3	13.9000	14.7122
	4	18.9000	18.3344		4	14.5000	16.3525
	5	20.3000	20.7474		5	15.3000	16.7725
	10	23.5000	26.9719		10	22.2000	22.5595
CT-img3	2	11.1000	14.9576	CT-img8	2	12.6000	14.6999
	3	17.1000	16.8898		3	15.7000	18.5384
	4	16.8000	18.2664		4	16.7000	20.5143
	5	19.9000	20.4396		5	19.0000	21.9539
	10	24.6000	26.0896		10	25.2000	28.0906
CT-img4	2	12.0000	14.5523	CT-img9	2	12.6000	13.7472
	3	17.5000	16.8656		3	13.8000	16.1962
	4	17.0000	17.9971		4	14.9000	19.7131
	5	18.6000	21.9691		5	20.8000	19.8622
	10	25.1000	26.9861		10	22.0000	25.8784
CT-img5	2	11.8000	14.0621	CT-img10	2	10.8000	14.1602
	3	17.4000	14.9456		3	15.8000	16.5176
	4	16.9000	16.4784		4	18.4000	17.5753
	5	18.5000	16.6599		5	19.1000	19.4396
	10	25.2000	19.1864		10	26.1000	24.4256

CT images dataset. All compared methods use the same objective function as our proposed method, and we evaluate their performance using PSNR values for different thresholding numbers. Specifically, Table 21 presents a comparison between our proposed Es-MFO algorithm and several other methods, including MRFO-OBL (Houssein, Emam, & Ali, 2021b), MRFO (Zhao et al., 2020), RSA (Abualigah et al., 2022), MPA (Faramarzi et al., 2020), and RUN (Ahmadianfar et al., 2021). Meanwhile, Table 22 compares our proposed Es-MFO (Houssein et al., 2022) method with the I-EO method for different thresholding numbers. A higher PSNR value indicates a more effective and reliable algorithm. Table 22 shows that the Es-MFO algorithm achieves higher PSNR values than the I-EO algorithm for all the images and all threshold values.

8. Conclusions and future extensions

An upgraded variant of the MFO algorithm, Es-MFO, has been established and applied to solve nineteen basic benchmark functions, twenty-nine IEEE CEC 2017 test problems, two engineering design optimization problems, and COVID-19 CT image segmentation problems. Few state-of-the-art algorithms and variants of the MFO have been considered to assess the performance of the proposed Es-MFO algorithm. Moreover, to check the significance of Es-MFO, it has also been tested statistically with some of the rank analysis viz., Friedman rank tests and Wilcoxon ranks tests. Further, diversity analysis of the Es-MFO has been employed to check the balance between diversification and intensification. Moreover, the proposed Es-MFO has been applied on three CEC2020 real-world constrained engineering design problems and multilevel threshold image segmentation of COVID-19 CT images. For these real-world issues, the newly created Es-MFO algorithm outperforms than other optimization algorithms. From all the evaluation

results, comments and analysis demonstrate that the suggested Es-MFO has achieved superior results than the other considered algorithms.

In the future, this method can be used with multi- and many-objective optimization algorithms. The proposed F-MFO algorithm could also be used to study a variety of real optimization problems, such as vehicle routing, job shop planning, parameter estimation of fuel cell problem, combined economic and emission dispatch problem, image segmentation problem, workflow planning, etc.

CRedit authorship contribution statement

Saroj Kumar Sahoo: Software, Conceptualization, Formal analysis, Data curation, Writing – original draft. **Essam H. Houssein:** Supervision, Methodology, Formal analysis, Visualization, Writing – review & editing. **M. Premkumar:** Software, Methodology, Investigation, Visualization. **Apu Kumar Saha:** Supervision, Conceptualization, Visualization, Writing – review & editing. **Marwa M. Emam:** Software, Conceptualization, Methodology, Formal analysis, Data curation, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

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

Data availability

No data was used for the research described in the article.

Appendix A

Table A1
Formulation of 19 benchmark functions.

Sl. no	Functions	Formulation of objective functions	d	Fmin	Search space
Unimodal Benchmark Functions					
F1	Beale	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	0	[-100, 100]
F2	Booth	$f(x) = (2x_1 + x_2 - 5)^2 + (x_1 + 2x_2 - 7)^2$	2	0	[-10, 10]
F3	Matyas	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	0	[-10, 10]
F4	Sumsquare	$f(x) = \sum_{i=1}^D x_i^2 \times i$	30	0	[-10, 10]
F5	Zettl	$f(x) = (x - 1^2 + x - 2^2 - 2x_1)^2 + 0.25x_1$	2	-0.00379	[-1, 5]
F6	Leon	$f(x) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$	2	0	[-1.2, 1.2]
F7	Zakhrov	$f(x) = \sum_{j=1}^d x_j^2 + (0.5 \sum_{j=1}^d jx_j)^2 + (0.5 \sum_{j=1}^d jx_j)^4$	2	0	[-5, 10]
Multimodal Benchmark Functions					
F8	Bohachevsky	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.3$	2	0	[-100, 100]
F9	Bohachevsky 3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.3$	2	0	[-50, 50]
F10	Levy	$f(x) = \sin^2(\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + 10\sin^2(\pi x_i + 1)] + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]$ Where, $x_i = 1 + \frac{1}{4}(x_i - 1)$, $i = 1, 2, \dots, D$	30	0	[-10, 10]
F11	Michalewicz	$f(x) = -\sum_{i=1}^D \sin(x_i) \sin^{2m}(\frac{ix_i^2}{\pi})$, $m = 10$	10	-9.66015	[0, π]
F12	Alpine	$f(x) = \sum_{i=1}^D x_i \sin(x_i) + 0.1x_i $	30	0	[-10, 10]
F13	Schaffers	$f(x) = 0.5 + \frac{\sin^2(x_1^2 + x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$	2	0	[-100, 100]
F14	Powersum	$f(x) = \sum_{i=1}^D \left[\left(\sum_{k=1}^D (x_k^i - b_i) \right)^2 \right]$	30	0	[-10, 10]
F15	Penalized2	$f(x) = 0.1 \left\{ 10\sin^2(\pi x_i) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + 10\sin^2(3\pi x_{i+1}) + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]] \right\} + \sum_{i=1}^D u(x_i, 5, 100, 4)$	30	0	[-50, 50]
F16	Kowalik	$f(x) = \sum_{j=1}^{11} \left[a_j - \frac{x_1(b_j^2 + b_jx_2)}{(b_j^2 - b_jx_3 - x_4)} \right]^2$	4	0.0003075	[-5, 5]
F17	Foxholes	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j} + \sum_{i=1}^D (x_i - a_{ij})^6 \right]^{-1}$	2	3	[-65, 65]
F18	Inverted cosine mixture	$f(x) = 0.1 \times 30 - \left[0.1 \times \sum_{j=1}^d 5\pi x_j - \sum_{j=1}^d x_j^2 \right]$	30	[-1, 1]	0
F19	salomon	$f(x) = 1 - \cos\left(2\pi\sqrt{\sum_{j=1}^d x_j^2}\right) + 0.1\sqrt{\sum_{j=1}^d x_j^2}$	30	[-100, 100]	0

Appendix B

Speed reducer design problem

Objective function:

$$\text{Min. } f(x) = 0.7854x_2^2x_1(14.9334x_3 + 3.3333x_3^2 - 43.0934) + 0.7854(x_4x_6^2 + x_5x_7^2) + 7.477(x_7^3 + x_6^3) - 1.508(x_7^2 + x_6^2)$$

Such that

$$h_1(x) = -x_1x_2^2x_3 + 27 \leq 0,$$

$$h_2(x) = -(1)x_2^2x_3^2 + 397.5 \leq 0,$$

$$h_3(x) = -x_2x_2^2x_3x_4^{-3} + 1.93 \leq 0,$$

$$h_4(x) = -x_2x_2^2x_3x_5^{-3} + 1.93 \leq 0,$$

$$h_5(x) = 10x_6^{-3} \sqrt{16.91 \times 10^6 + (745x_4x_2^{-1}x_3^{-1})^2} - 1100 \leq 0,$$

$$h_6(x) = 10x_7^{-3} \sqrt{157.5 \times 10^6 + (745x_5x_2^{-1}x_3^{-1})^2} - 850 \leq 0,$$

$$h_7(x) = x_2x_3 - 40 \leq 0,$$

$$h_8(x) = -x_1x_2^{-1} + 5 \leq 0,$$

$$h_9(x) = x_1x_2^{-1} - 12 \leq 0,$$

$$h_{10}(x) = 1.5x_6 - x_4 + 1.9 \leq 0,$$

$$h_{11}(x) = 1.1x_7 - x_5 + 1.9 \leq 0,$$

$$2.6 \leq x_1 \leq 3.6, 0.7 \leq x_2 \leq 0.8, 17 \leq x_3 \leq 28,$$

$$2.9 \leq x_6 \leq 3.9, 5 \leq x_7 \leq 5.5, 7.3 \leq x_4, x_5 \leq 8.3.$$

Multiple disk clutch break design problem

Objective function:

$$Minf(x) = \pi(x_2^2 - x_1^2)x_3(x_5 + 1)\rho$$

Constraints:

$$h_1(x) = -p_{max} + p_{rz} \leq 0,$$

$$h_2(x) = p_{rz}V_{sr} - V_{sr,max}p_{max} \leq 0,$$

$$h_3(x) = \Delta R + x_1 - x_2 \leq 0,$$

$$h_4(x) = -L_{max} + (x_5 + 1)(x_3 + \partial) \leq 0,$$

$$h_5(x) = sM_s - M_h \leq 0,$$

$$h_6(x) = T \geq 0,$$

$$h_7(x) = -V_{sr,max} + V_{sr} \leq 0,$$

$$h_8(x) = T - T_{max} \leq 0,$$

where,

$$M_h = \frac{2}{3}\mu x_4 x_5 \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2} N.mm, \omega = \frac{\Pi n}{30} \frac{rad}{s}, A = \pi(x_2^2 - x_1^2)mm^2$$

$$p_{rz} = \frac{x_4}{A} N/mm^2, V_{sr} = \frac{\Pi R_{sr} n}{30} mm/s, R_{sr} = \frac{2}{3} \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2}, T = \frac{I_z \omega}{M_h + M_f},$$

$$\Delta R = 20mm, L_{max} = 30mm, \mu = 0.6,$$

$$V_{sr,max} = 10 \frac{m}{s}, \partial = 0.5mm, s = 1.5,$$

$$T_{max} = 15s, n = 250rpm, I_z = 55Kg.m^2$$

$$T_{max} = 15s, n = 250rpm, I_z = 55Kg.m^2$$

$$M_s = 40Nm, M_f = 3Nm, and p_{max} = 1$$

Variable range:

$$60 \leq x_1 \leq 80, 90 \leq x_2 \leq 110, 1 \leq x_3 \leq 3, 0 \leq x_4 \leq 1000, 2 \leq x_5 \leq 9.$$

Welded beam design problem

Objective function:

$$Min.f(x) = 0.04811x_3x_4\{x_2 + 14\} + 1010471x_1^2x_2$$

Subject to:

$$h_1(x) = x_1 - x_4 \leq 0,$$

$$h_2(x) = \delta(x) - \delta_{max} \leq 0,$$

$$h_3(x) = P \leq P_c(x),$$

$$h_4(x) = \tau_{max} \geq \tau(x),$$

$$h_5(x) = \sigma(x) - \sigma_{max} \leq 0,$$

where,

$$\tau = \sqrt{\tau'^2 + \tau''^2} + 2\tau'\tau''\frac{x_2}{2R}, \tau' = \frac{P}{\sqrt{2}x_2x_1}, \tau'' = \frac{RM}{J}, M = p(\frac{x_2}{2} + L)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, J = 2\frac{x_2^2}{4} + \left(\left(\frac{x_1 + x_3}{2}\right)\sqrt{2}x_1x_2\right)^2,$$

$$\sigma(x) = \frac{6PL}{x_4x_2^2}, \delta(x) = \frac{6PL^3}{Ex_3^2x_4}, P_c(x) = \frac{4.013Ex_3x_4^3}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

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

$$\sigma_{max} = 30,000psi, \delta_{max} = 0.25in, \tau_{max} = 13600psi,$$

where,

$$0.1 \leq x_3, x_2 \leq 10,$$

$$0.125 \leq x_1 \leq 2,$$

$$0.1 \leq x_4 \leq 2.$$

References

- Abd Elaziz, M., Ewees, A. A., Yousef, D., Alwerfali, H. S. N., Awad, Q. A., Lu, S., et al. (2020). An improved marine predators algorithm with fuzzy entropy for multi-level thresholding: Real world example of COVID-19 CT image segmentation. *IEEE Access*, 8, 125306–125330.
- Abdel-Basset, M., Mohamed, R., Jameel, M., & Abouhawwash, M. (2023). Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems. *Knowledge-Based Systems*, 110248.
- Abdollahzadeh, B., Soleimani Gharehchopogh, F., & Mirjalili, S. (2021). Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *International Journal of Intelligent Systems*, 36(10), 5887–5958.
- Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The arithmetic optimization algorithm. *Computer Methods in Applied Mechanics and Engineering*, 376, Article 113609.
- Abualigah, L., Abd Elaziz, M., Sumari, P., Geem, Z. W., & Gandomi, A. H. (2022). Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Systems with Applications*, 191, Article 116158.
- Agushaka, J. O., Ezugwu, A. E., & Abualigah, L. (2022). Dwarf mongoose optimization algorithm. *Computer methods in applied mechanics and engineering*, 391, Article 114570.
- Ahmadianfar, I., Heidari, A. A., Noshadian, S., Chen, H., & Gandomi, A. H. (2022). INFO: An efficient optimization algorithm based on weighted mean of vectors. *Expert Systems with Applications*, 195, Article 116516.
- Ahmadianfar, I., Heidari, A. A., Gandomi, A. H., Chu, X., & Chen, H. (2021). RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. *Expert Systems with Applications*, 181, Article 115079.
- Apinantanakon, W., & Sunat, K. (2017). Omfo: A new opposition-based moth-flame optimization algorithm for solving unconstrained optimization problems. *International Conference on Computing and Information Technology*, 22–31.
- Arora, S., & Singh, S. (2015). Butterfly algorithm with levy flights for global optimization. 2015 International Conference on Signal Processing, Computing and Control (ISPPCC), 220–224.
- Awad, N. H., Ali, M. Z., Suganthan, P. N., & Reynolds, R. G. (2016b). An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. In 2016 IEEE congress on evolutionary computation (CEC) (pp. 2958–2965). IEEE.
- Awad, N., Ali, M., Liang, J., Qu, B., & Suganthan, P. (2016a). Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization (pp. 1–34). Nanyang Technological University, Jordan University of Science and Technology and Zhengzhou University, Singapore and Zhenzhou, Tech. Rep 201611.
- Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J., & Qu, B. Y. (2017). Problem definitions and evaluation criteria for the CEC2017. *Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*.
- Azizi, M., Talatahari, S., & Gandomi, A. H. (2023). Fire hawk optimizer: A novel metaheuristic algorithm. *Artificial Intelligence Review*, 56(1), 287–363.
- Chakraborty, S., & Mali, K. (2021). A morphology-based radiological image segmentation approach for efficient screening of COVID-19. *Biomedical Signal Processing and Control*, 69, Article 102800.
- Chakraborty, S., Saha, A. K., Nama, S., & Debnath, S. (2021). COVID-19 X-ray image segmentation by modified whale optimization algorithm with population reduction. *Computers in Biology and Medicine*, 139, Article 104984.
- Chakraborty, S., Sharma, S., Saha, A. K., & Saha, A. (2022a). A novel improved whale optimization algorithm to solve numerical optimization and real-world applications. *Artificial Intelligence Review*, 1–12.
- Chakraborty, S., Saha, A. K., Sharma, S., Sahoo, S. K., & Pal, G. (2022b). Comparative performance analysis of differential evolution variants on engineering design problems. *Journal of Bionic Engineering*, 19(4), 1140–1160.
- Chen, C. L., Wang, X., Yu, H., Wang, M., & Chen, H. (2021). Dealing with multi-modality using synthesis of Moth-flame optimizer with sine cosine mechanisms. *Mathematics and Computers in Simulation*, 188, 291–318.
- Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11–12), 1245–1287.
- Dabba, A., Tari, A., Meftali, S., & Mokhtari, R. (2021). Gene selection and classification of microarray data method based on mutual information and moth flame algorithm. *Expert Systems with Applications*, 166, Article 114012.
- Dash, S. P., Subhashini, K. R., & Satapathy, J. K. (2020). Optimal location and parametric settings of FACTS devices based on JAYA blended moth flame optimization for transmission loss minimization in power systems. *Microsystem Technologies*, 26(5), 1543–1552.
- Elsakaan, A. A., El-Sehiemy, R. A., Kaddah, S. S., & Elsaid, M. I. (2018). An enhanced moth-flame optimizer for solving non-smooth economic dispatch problems with emissions. *Energy*, 157, 1063–1078.
- Emam, M. M., Houssein, E. H., & Ghoniem, R. M. (2023). A modified reptile search algorithm for global optimization and image segmentation: Case study brain MRI images. *Computers in Biology and Medicine*, 152, Article 106404.
- Faramarzi, A., Heidarinejad, M., Mirjalili, S., & Gandomi, A. H. (2020). Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert systems with applications*, 152, Article 113377.
- Ferrer, R. (2020). COVID-19 Pandemic: The greatest challenge in the history of critical care. *Medicina intensiva*, 44(6), 323.
- Glasbey, C. A. (1993). An analysis of histogram-based thresholding algorithms. *CVGIP: Graphical models and image processing*, 55, 532–537.
- Gu, W., & Xiang, G. (2021). Improved moth flame optimization with multioperator for solving real-world optimization problems. 2021 IEEE 5th Advanced Information Technology, *Electronic and Automation Conference (IAEAC)*, 5, 2459–2462. <https://doi.org/10.1109/IAEAC50856.2021.9390876>
- Gurrola-Ramos, J., Hernández-Aguirre, A., & Dalmau-Cedeño, O. (2020, July). COLSHADE for real-world single-objective constrained optimization problems. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1–8). IEEE.
- Ilhan, A., Alpan, K., Sekeroglu, B., & Abiyev, R. (2023). COVID-19 Lung CT image segmentation using localization and enhancement methods with U-Net. *Procedia Computer Science*, 218, 1660–1667.
- Harmon, S. A., Sanford, T. H., Xu, S., Turkbey, E. B., Roth, H., Xu, Z., et al. (2020). Artificial intelligence for the detection of COVID-19 pneumonia on chest CT using multinational datasets. *Nature Communications*, 11(1), 4080.
- Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future generation computer systems*, 97, 849–872.
- Holland, J. H. (1992). Genetic algorithms. *Scientific American*, 267(1), 66–73.
- Hou, G., Gong, L., Hu, B., Su, H., Huang, T., Huang, C., et al. (2022). Application of fast adaptive moth-flame optimization in flexible operation modeling for supercritical unit. *Energy*, 239, Article 121843.
- Houssein, E. H., Emam, M. M., & Ali, A. A. (2021a). An efficient multilevel thresholding segmentation method for thermography breast cancer imaging based on improved chimp optimization algorithm. *Expert Systems with Applications*, 185, Article 115651.
- Houssein, Essam H., Bahaa El-din Helmy, Diego Oliva, Ahmed A. Elngar, and Hassan Shaban. "A novel black widow optimization algorithm for multilevel thresholding image segmentation." *Expert Systems with Applications* 167 (2021e): 114159.
- Houssein, E. H., Helmy, B.-D., Oliva, D., Pradeep Jangir, M., Premkumar, A. A., & Elngar, and Hassan Shaban. (2022). An efficient multi-thresholding based COVID-19 CT images segmentation approach using an improved equilibrium optimizer. *Biomedical Signal Processing and Control*, 73, Article 103401.
- Houssein, E. H., Hussain, K., Abualigah, L., Elaziz, M. A., Alomoush, W., Dhiman, G., et al. (2021c). An improved opposition-based marine predators algorithm for global optimization and multilevel thresholding image segmentation. *Knowledge-Based Systems*, 229, Article 107348.
- Houssein, E. H., Emam, M. M., & Ali, A. A. (2021b). Improved manta ray foraging optimization for multi-level thresholding using COVID-19 CT images. *Neural Computing and Applications*, 33(24), 16899–16919.
- Houssein, E. H., Oliva, D., Çelik, E., Emam, M. M., & Ghoniem, R. M. (2023). Boosted sooty tern optimization algorithm for global optimization and feature selection. *Expert Systems with Applications*, 213, Article 119015.
- Kadry, S., Rajinikanth, V., Raja, N., Jude Hemanth, D., Hannon, N., & Raj, A. N. J. (2021). Evaluation of brain tumor using brain MRI with modified-moth-flame algorithm and Kapur's thresholding: A study. *Evolutionary Intelligence*, 14(2), 1053–1063.
- Kapur, J. N., Sahoo, P. K., & Wong, A. K. (1985). A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29, 273–285.
- Kennedy, J., Eberhart, R. (1995) Particle swarm optimization. In: Proceedings of ICNN'95—international conference on neural networks, 4:1942–1948.
- Khalilpourazari, S., & Khalilpourazary, S. (2019). An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems. *Soft Computing*, 23(5), 1699–1722.
- Khan, M. A., Arshad, H., Damaševičius, R., Alqahtani, A., Alsubai, S., Binbusayis, A., Nam, Y., & Kang, B.-G. (2022). Human Gait Analysis: A Sequential Framework of Lightweight Deep Learning and Improved Moth-Flame Optimization Algorithm. *Computational Intelligence and Neuroscience*, 2022.
- Kigsirisin, S., & Miyauchi, H. (2021). Short-term operational scheduling of unit commitment using binary alternative moth-flame optimization. *IEEE Access*, 9, 12267–12281.
- Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., & Das, S. (2020a). A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, 56, Article 100693.
- Kumar, A., Das, S., & Zelinka, I. (2020b). A modified covariance matrix adaptation evolution strategy for real-world constrained optimization problems. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (pp. 11–12).
- Kumar, A., Das, S., & Zelinka, I. (2020c). A self-adaptive spherical search algorithm for real-world constrained optimization problems. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (pp. 13–14).
- Li, C., Niu, Z., Song, Z., Li, B., Fan, J., & Liu, P. X. (2018). A double evolutionary learning moth-flame optimization for real-parameter global optimization problems. *IEEE Access*, 6, 76700–76727.
- Li, Z., Zeng, J., Chen, Y., Ma, G., & Liu, G. (2021). Death mechanism-based moth-flame optimization with improved flame generation mechanism for global optimization tasks. *Expert Systems with Applications*, 183, Article 115436. <https://doi.org/10.1016/j.eswa.2021.115436>
- Liu, L., Zhao, D., Yu, F., Heidari, A. A., Li, C., Ouyang, J., et al. (2021). Ant colony optimization with Cauchy and greedy Levy mutations for multilevel COVID 19 X-ray image segmentation. *Computers in Biology and Medicine*, 136, Article 104609.
- Cheng, M.-Y., & Prayogo, D. (2014). Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers and Structures*, 139, 98–112.
- Ma, M., Wu, J., Shi, Y., Yue, L., Yang, C., & Chen, X. (2022). Chaotic random opposition-based learning and Cauchy mutation improved moth-flame optimization algorithm for intelligent route planning of multiple UAVs. *IEEE Access*, 10, 49385–49397.
- Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, 89, 228–249.

- Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, 96, 120–133.
- Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, 95, 51–67.
- Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, 114, 163–191.
- Mohd Rose, A. N., & Nik Mohamed, N. M. Z. (2022). Hybrid Flow Shop Scheduling with Energy Consumption in Machine Shop Using Moth Flame Optimization. In *Recent Trends in Mechatronics Towards Industry 4.0* (pp. 77–86). Springer.
- Nadimi-Shahraki, M. H., Fatahi, A., Zamani, H., Mirjalili, S., & Abualigah, L. (2021b). An improved moth-flame optimization algorithm with adaptation mechanism to solve numerical and mechanical engineering problems. *Entropy*, 23(12), 1637.
- Nadimi-Shahraki, M. H., Fatahi, A., Zamani, H., Mirjalili, S., Abualigah, L., & Abd Elaziz, M. (2021a). Migration-based moth-flame optimization algorithm. *Processes*, 9(12), 2276.
- Nadimi-Shahraki, M. H., Moeini, E., Taghian, S., & Mirjalili, S. (2021c). DMFO-CD: A discrete moth-flame optimization algorithm for community detection. *Algorithms*, 14(11), 314.
- Nama, S. (2022). A novel improved SMA with quasi reflection operator: Performance analysis, application to the image segmentation problem of Covid-19 chest X-ray images. *Applied Soft Computing*, 118, Article 108483.
- Nama, S., & Saha, A. (2018). An ensemble symbiosis organisms search algorithm and its application to real world problems. *Decision Science Letters*, 7(2), 103–118.
- Nama, S., & Saha, A. (2019). A novel hybrid backtracking search optimization algorithm for continuous function optimization. *Decision Science Letters*, 8(2), 163–174.
- Nama, S., Sharma, S., Saha, A. K., & Gandomi, A. H. (2022). A quantum mutation-based backtracking search algorithm. *Artificial Intelligence Review*, 1–55.
- Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9, 62–66.
- Oyelade, O. N., Ezugwu, A. E. S., Mohamed, T. I., & Abualigah, L. (2022). Ebola optimization search algorithm: A new nature-inspired metaheuristic optimization algorithm. *IEEE Access*, 10, 16150–16177.
- Pelusi, D., Mascella, R., Tallini, L., Nayak, J., Naik, B., & Deng, Y. (2020). An Improved Moth-Flame Optimization algorithm with hybrid search phase. *Knowledge-Based Systems*, 191, Article 105277.
- Pierre, D. A. (1986). *Optimization theory with applications*. Courier Corporation.
- Qi, A., Zhao, D., Yu, F., Heidari, A. A., Wu, Z., Cai, Z., et al. (2022). Directional mutation and crossover boosted ant colony optimization with application to COVID-19 X-ray image segmentation. *Computers in Biology and Medicine*, 148, Article 105810.
- Qin, A. K., & Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *2005 IEEE congress on evolutionary computation (Vol. 2, pp. 1785-1791)*. IEEE.
- Ramachandran, R., Satheesh Kumar, J., Madasamy, B., & Veerasamy, V. (2021). A hybrid MFO-GHNN tuned self-adaptive FOPID controller for ALFC of renewable energy integrated hybrid power system. *IET Renewable Power Generation*, 15(7), 1582–1595.
- Ramaprabha, R. (2012). Maximum power point tracking of partially shaded solar PV system using modified Fibonacci search method with fuzzy controller.
- Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, 7(1), 19–34.
- Saha, A. K. (2022). Multi-population-based adaptive sine cosine algorithm with modified mutualism strategy for global optimization. *Knowledge-Based Systems*, 251, Article 109326.
- Sahoo, S. K., & Saha, A. K. (2022). A hybrid moth flame optimization algorithm for global optimization. *Journal of Bionic Engineering*, 19(5), 1522–1543.
- Sahoo, S. K., Saha, A. K., Nama, S., & Masdari, M. (2022a). An improved moth flame optimization algorithm based on modified dynamic opposite learning strategy. *Artificial Intelligence Review*.
- Sahoo, S. K., Saha, A. K., Sharma, S., Mirjalili, S., & Chakraborty, S. (2022b). An enhanced moth flame optimization with mutualism scheme for function optimization. *Soft Computing*, 26(6), 2855–2882.
- Sahoo, S. K., Saha, A. K., Ezugwu, A. E., et al. (2022c). Moth flame optimization: Theory, modifications, hybridizations, and applications. *Arch Computat Methods Eng*. <https://doi.org/10.1007/s11831-022-09801-z>
- Sapre, S., & Mini, S. (2021). A differential moth flame optimization algorithm for mobile sink trajectory. *Peer-to-Peer Networking and Applications*, 14(1), 44–57.
- Sara, U., Akter, M., & Uddin, M. S. (2019). Image quality assessment through fsm, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 7, 8–18.
- Shan, W., Qiao, Z., Heidari, A. A., Chen, H., Turabieh, H., & Teng, Y. (2021). Double adaptive weights for stabilization of moth flame optimizer: Balance analysis, engineering cases, and medical diagnosis. *Knowledge-Based Systems*, 214, Article 106728. <https://doi.org/10.1016/j.knsys.2020.106728>
- Sharma, A., Sharma, A., Averbukh, M., Rajput, S., Jatly, V., Choudhury, S., et al. (2022a). Improved moth flame optimization algorithm based on opposition-based learning and Lévy flight distribution for parameter estimation of solar module. *Energy Reports*, 8, 6576–6592.
- Sharma, S., Chakraborty, S., Saha, A. K., Nama, S., & Sahoo, S. K. (2022b). mLBOA: A modified butterfly optimization algorithm with lagrange interpolation for global optimization. *Journal of Bionic Engineering*. <https://doi.org/10.1007/s42235-022-00175-3>
- Sharma, S., Saha, A. K., Majumder, A., & Nama, S. (2021). MPBOA-A novel hybrid butterfly optimization algorithm with symbiosis organisms search for global optimization and image segmentation. *Multimedia Tools and Applications*, 80(8), 12035–12076.
- Sharma, S., Saha, A. K., Roy, S., Mirjalili, S., & Nama, S. (2022). A mixed sine cosine butterfly optimization algorithm for global optimization and its application. *Cluster Computing*, 1–28.
- Sohrabi, C., Alsafi, Z., O'neill, N., Khan, M., Kerwan, A., Al-Jabir, A., & Agha, R. (2020). World Health Organization declares global emergency: A review of the 2019 novel coronavirus (COVID 19). *International Journal of Surgery*, 76, 71–76.
- Soliman, G. M. A., Khorshid, M. M. H., & Abou-El-Enien, T. H. M. (2016). modified moth-flame optimization algorithms for terrorism prediction. 5(7), 12.
- Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4), 341–359.
- Su, H., Zhao, D., Heidari, A. A., Liu, L., Zhang, X., Mafarja, M., et al. (2023). RIME: A physics-based optimization. *Neurocomputing*.
- Tanabe, R., & Fukunaga, A. (2013, June). Success-history based parameter adaptation for differential evolution. In *2013 IEEE congress on evolutionary computation* (pp. 71–78). IEEE.
- Tanabe, R., & Fukunaga, A. S. (2014, July). Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)* (pp. 1658-1665). IEEE.
- Tsai, W.-H. (1985). Moment-preserving thresholding: A new approach. *Computer Vision, Graphics, and Image Processing*, 29, 377–393.
- Wang, G., Guo, S., Han, L., & Cekderi, A. B. (2022). Two-dimensional reciprocal cross entropy multi-threshold combined with improved firefly algorithm for lung parenchyma segmentation of COVID-19 CT image. *Biomedical Signal Processing and Control*, 78, Article 103933.
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.
- Zhao, J., Zhang, Y., He, X., & Xie, P. (2020). Covid-ct-dataset: a ct scan dataset about covid-19. arXiv preprint arXiv:2003.13865, 490.
- Zhao, X., Fang, Y., Liu, L., Li, J., & Xu, M. (2020b). An improved moth-flame optimization algorithm with orthogonal opposition-based learning and modified position updating mechanism of moths for global optimization problems. *Applied Intelligence*, 50(12), 4434–4458.
- Zhao, X., Fang, Y., Liu, L., Xu, M., & Li, Q. (2022b). A covariance-based Moth-flame optimization algorithm with Cauchy mutation for solving numerical optimization problems. *Applied Soft Computing*, 119, Article 108538.
- Zhao, X., Fang, Y., Ma, S., & Liu, Z. (2022a). Multi-swarm improved moth-flame optimization algorithm with chaotic grouping and Gaussian mutation for solving engineering optimization problems. *Expert Systems with Applications*, 204, Article 117562.
- Zhao, W., Zhang, Z., & Wang, L. (2020c). Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Engineering Applications of Artificial Intelligence*, 87, Article 103300.