Journal of Scientific & Industrial Research
Vol. 84, February 2025, pp. 170-182
DOI: 10.56042/jsir.v84i02.8130

MBO: A Novel Memory based Optimizer for Continuous and Discrete Optimization Problem

Rajesh Ranjan* & Jitender Kumar Chhabra

Computer Engineering Department, National Institute of Technology, Kurukshetra 136 119, Haryana, India

Received 17 January 2024; revised 20 August 2024; accepted 13 December 2024

This study introduces a novel metaheuristic approach called Memory Based Optimizer (MBO), which emulates the problem-solving process through multiple stages by utilizing the best solution obtained in terms of memory. Rooted in principles of human psychology, MBO reflects the tendency for individuals to solve problems incrementally, using previous learning to take small steps toward an optimal solution within a limited number of attempts. MBO is first evaluated on ten CEC 2019 Benchmark Functions, and its results are compared with ten other metaheuristic algorithms under similar execution conditions. In its binary form, MBO is also applied as a wrapper for feature selection in supervised machine learning using the K-NN classifier on twelve benchmark classification datasets. The findings indicate significant improvements in average accuracy and optimal feature selection compared to other metaheuristic approaches. As per the simulation results, MBO has outperformed other metaheuristics approaches in 5 out of 10 continuous benchmark functions. Further, the MBO has achieved higher average accuracy in 11 out of 12 datasets, along with better execution times in 9 out of 12 datasets when applied as a wrapper for feature selection tasks, the average improvement in accuracy and F1 score is reported as 2.8746% and 3.1643% when compared with other metaheuristic approaches under similar execution environment further validating its robustness and efficiency across multiple optimization tasks.

Keywords: CEC2019 benchmark functions, Feature selection, Metaheuristic, Wrapper

Introduction

Real-life, complex engineering problems have various constraints that a traditional optimization technique fails to handle and thus may not yield the desired solution. These problems have a vast search space for finding the optimal solution, achieved by minimizing or maximizing some objective function with several decision variables.[1] Recently, several metaheuristic optimization techniques have been proposed and applied to solve complex problems in disciplines such as science, engineering, management, and economics.[2] The increasing popularity of meta-heuristic optimization algorithms can be attributed to their efficacy in addressing intricate problems across diverse domains due to these reasons: (i) These methods depend on elementary principles and are straightforward to execute; (ii) they do not necessitate knowledge regarding the gradient of the objective function; (iii) they can circumvent local minima; and (iv) they can be employed to address a diverse set of problems across multiple disciplines.[3,4]

Metaheuristic algorithms solve this complex optimization problem by fine-tuning the global diversified search and local intense search. Exploration systematically explores the entire search space. This is related to avoiding local optima and overcoming their traps. Contrarily, exploitation refers to the ability to examine nearby potential solutions to enhance their quality within a localized context.[5,6] Due to the inherently random behavior of metaheuristics algorithms, striking an appropriate balance between the two stages is difficult. The primary distinctions among metaheuristics lie in the specific methodologies employed to achieve a harmonious equilibrium between these two fundamental processes using different operators and methods.[7]

One of the striking features of the metaheuristic approach is that it can be applied to solve both continuous and discrete optimization problems. The binary form of the metaheuristic approach is commonly employed to address discrete optimization problems. This approach converts continuous values into discrete ones using methods such as threshold values or shape-specific functions like S-shaped or V-shaped functions.[8]

*Author for Correspondence
E-mail: rajesh_61900059@nitkkr.ac.in; iiitm.rajesh@gmail.com

Feature selection is considered a binary optimization problem. It is regarded as an NP-Hard problem, and most exhaustive and greedy approaches fail to give the optimal subset of features.[9]

Recently, researchers have successfully applied the binary metaheuristic approach for solving feature selection problems in which the algorithm acts as a wrapper to obtain the optimal feature subset.[10,11]

The No Free Lunch Theorem posits that there is no optimization procedure that can be universally applicable for the solution of all optimization problems.[12] This phenomenon has resulted in various research studies within the metaheuristic domain. These studies typically involve hybridizing multiple approaches, modifying existing algorithms, or introducing new techniques.

The present study introduces a novel metaheuristic methodology, a memory-based search technique, known as Memory Based Optimizer (MBO) which searches the solution space by balancing the global expedition and local intensification search by gradually shifting from exploratory to intensified search. The proposed work stores its best solution obtained so far during each successive iteration and updates its value if the current obtained solution is better than the best-stored solution. A novel function is also proposed, which searches the neighborhood of the given particle's position for better results than the previous one. The proposed work's significance is determined by conducting tests on CEC 2019 Benchmark Functions and comparing the results with ten other metaheuristic approaches in a similar execution environment.[13] Moreover, the proposed metaheuristic approach is implemented as a wrapper over multiple datasets, effectively selecting pertinent features and enhancing overall average accuracy level.[14]

Related Work

In the past few decades, researchers have proposed several metaheuristic approaches for solving numerous complex stochastic real-world optimization problems, which can be broadly categorized into four groups: evolutionary algorithms, swarm intelligence algorithms, physics-based methods, and human intelligence-based methods.[15]

Evolutionary algorithms are computational models that replicate the process of natural evolution. These algorithms use biologically-motivated operators like crossover and mutation to achieve their goals. This category comprises some well-known and widely-used algorithms, such as The Genetic Algorithm (GA), Evolutionary Strategy (ES), and Differential Evolution (DE).[17,18] The second category of nature-inspired techniques consists of swarm-intelligence-based methods replicating animal movement or hunting behavior.[19] This category encompasses several notable algorithms, such as Particle Swarm Optimization (PSO), Grey-Wolf Optimizer (GWO), Artificial Bee Colony (ABC), Harris Hawk Optimizer (HHO), and Crow Search Algorithm (CSA).[20–24]

Another category of optimization algorithms includes physics-based techniques. This category derives from physical laws in the actual world and often refers to sharing search answers based on governing principles rooted in physical procedures. Simulated Annealing, Gravitational Search Algorithm, and Multi-verse Optimizer are a few popular algorithms in this field.[25–27] Human-based strategies, inspired by human cooperation and social behavior, comprise the last optimization technique category. The Imperialist Competitive Algorithm, Tabu Search (TS), the Teaching-Learning-Based Optimization Algorithm, and the Student Psychology Based Optimization (SPBO) algorithm are some of the algorithms in this category that are most often utilized.[28–31]

Metaheuristic methodologies have shown their efficacy in effectively addressing optimization challenges across several areas. In the field of machine learning and data mining, feature selection is recognized as a complex undertaking. In recent times, researchers have utilized various metaheuristic approaches to achieve optimal feature subsets, which enhance the overall accuracy and other performance metrics.

In one of the prominent optimization task Binary Particle Swarm Optimizations (BPSO) was proposed by Kennedy & Eberhart to solve discrete optimization problems.[32] Engelbrecht *et al.* applied Set-based PSO (SBPSO) as a wrapper using the KNN classifier. The particle's position and velocity in SBPSO are represented as mathematical sets determined by elements from the problem space and pairs of addition or deletion operations. The optimal solution of SBPSO is represented by the best position.[33]

Emary *et al.* proposed Binary grey wolf optimizer (BGWO) in which first method involves the binarization of the solutions. The use of the stochastic crossover approach is implemented for the purpose of updating the position of the grey wolves. The second technique employs the transfer function to transform the spatial coordinates of wolves into a discretized

search space. It has enhanced search skills in comparison to PSO and GA in the identification of optimum features within a given dataset.[34]

Ouadfel & Elaziz applied the Crow Search Algorithm (CSA) to solve the premature convergence in feature selection. The researchers presented an improved iteration of CSA for feature selection in their study. An adaptive awareness probability was also developed to enhance the equilibrium between exploration and exploitation. Furthermore, an updated search method was proposed to strengthen the global search capabilities of the CSA. They demonstrated an improved convergence speed.[35]

In their research, Taradeh *et al*. proposed a unique feature selection approach that makes use of the Gravitational Search approach (GSA). This method uses cutting-edge crossover and mutation operators. KNN and Decision Tree classifier implementation was done in the experiments. It was evaluated against GA, PSO, and GWO. The experimental findings showed that the suggested work performed better than other.[36]

In one of the prominent works, Too *et al.* introduced Binary Harris Hawk Optimization (BHHO) designed explicitly for feature selection. Additionally, the proposed improvement involves the implementation of a quadratic binary HHO. The obtained results confirmed the algorithm's effectiveness.[37]

Methodology

The current study introduces a novel MBO algorithm that addresses intricate real-life optimization problems by effectively managing the exploration and exploitation stages while searching for optimal solutions. The method proposed in this study operates through four distinct phases. To enhance the efficacy of the local search process, a novel neighborhood function is suggested, which aims to explore better solutions in the vicinity of previously obtained solutions.

Memory Based Optimizer

The MBO algorithm operates through four distinct phases. The first phase involves the initialization of particles. Following this, the second phase encompasses implementing and updating particles, which occurs across four stages. The subsequent phase consists of updating the optimal solution in terms of stored memory based on the obtained solution. Ultimately, the algorithm concludes when it reaches the maximum iteration or satisfies any other predetermined criteria.

Initialization Phase

In the initialization phase, a set of particles (P) is randomly generated, as depicted in matrix 1. Apart from the particle position, another matrix is also created, which stores the best position in terms of "Memory" obtained by each particle, which is initialized to P during the initial phase. During each successive iteration, the Global Best Memory (GBM) and Global Worst Memory (GWM) are stored, which, along with the "Memory", is used for the optimal solution.

$$P = \begin{bmatrix} P_{1,1} & \cdots & \cdots & P_{1,j} & P_{1,n-1} & P_{1,n} \\ P_{2,1} & \cdots & \cdots & P_{2,j} & P_{2,n-1} & P_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{N-1,1} & \cdots & \cdots & P_{N-1,j} & P_{N-1,n-1} & P_{N-1,n} \\ P_{N,1} & \cdots & \cdots & P_{N,j} & P_{N,n-1} & P_{N-1,n} \end{bmatrix} \quad \ldots (1)$$

Two parameters, 'ctrl_para1', and 'ctrl_para2', balance the global and local search in the present work is mentioned in Eqs 2 & 3.

$$\text{ctrl\_para1} = exp\left(-(\text{iter}/\text{iter}_{\max})^{1/alpha}\right) \quad \ldots (2)$$

$$\text{ctrl\_para2} = 0.25 \quad \ldots (3)$$

Here, value of 'alpha' is taken as 3 and 'iter' and '$\text{iter}_{\max}$' represents value of current iteration and maximum iteration respectively.

Implementation and Position Updation Phase

Stage 1- In this stage, the '*ctrl_para1*' is compared with the random number '*r_0*', whose value lies between [0,1]. The '*ctrl_para1*' is defined in such a way that its value gradually decreases from 1 and tends to become smaller after each successive iteration. If the value of '*ctrl_para1*' is more significant than '*r_0*', the particle updates its position using various substages of stage 1.

Stage 1 is further divided into two separate sub-stages depending upon the value of the '*ctrl_para2*' and the current '*iteration*' value.

Stage 1.1 - If the particle is in stage-1 and '*ctrl_para2*' is less than random number '*r_1*' whose value lies between [0,1], the particle updates its position as given by Eq. 4, where the particle chases the local best particle '*Memory*' of any random particle '*k*' and simultaneously avoids the worst performing particle given as '*GWM'*. Here, '*r1*' and '*r2*' denote the random variable between [0,1]

$$P_{i,j}^{iter+1} = P_{i,j}^{iter} + r_{1,j} \times \left(MEM_{k,j} - P_{i,j}^{iter}\right) + r_{2,j} \times \left(P_{i,j}^{iter} - GWM_j\right) \quad \ldots (4)$$

In this stage, the particles are basically in their initial stage of the search, and they update their local best by randomly chasing each other and avoiding the worst-performing particle.

Stage 1.2 - If the value of '*ctrl_para2*' is greater than the random number '*r_1*', the particle updates its position as given by Eq. 5.

$$P_{i,j}^{iter+1} = f\_neighbor(MEM_i) + r\_norm_j \times w1 \quad \ldots (5)$$

Here, '*f_neighbor ()*' is a unique function that searches around the neighborhood of the particle, which is passed as an argument to the function. The proposed function calculates the Euclidean Distance between the mid-point of a particular attribute's lower and upper bound to the particles' position and utilizes the obtained distance for searching the neighborhood. A detailed description of the function and its pseudo-code are given at the end of the section. The term 'MEM_i' denotes the best position value of i^{th} particle till iteration '*iter*', '*r_norm*' denotes the random normal value having mean=0 and standard deviation=1 and '*w1*' is the random number whose value lies between [0,1]. In the case of discrete optimization, the factor (r_norm × w1) is multiplied by 0.5.

Stage 2 - In this stage, the particle enters if the value of '*ctrl_para1*' is less than '*r_0*'.

Stage 2.1 - As the value of *ctrl_para1* gradually decreases, this stage becomes more prominent during the later stage of the algorithm execution. Mathematically, it is given by Eq. 6.

$$P_{i,j}^{iter+1} = P_{i,j}^{iter} + r_{1,j} \times \left(MEM_{i,j} - P_{i,j}^{iter}\right) + r_{2,j} \times \left(GBM_j - P_{i,j}^{iter}\right) \quad \ldots (6)$$

where, 'MEM_i' represents the best position obtained by the i^{th} particle and stored as Memory, and '*GBM*' represents the Global Best Memory obtained so far.

Stage 2.1 - In this stage the '*ctrl_para2*' is again compared with another random variable 'r_2' and, if the value of '*ctrl_para2*' is greater than the random number '*r_2*' the particle updates its position as given by Eq. 7.

$$P_{i,j}^{iter+1} = P_{i,j}^{iter} + w2 \times random\_number[0,1] \times \left(f\_neighbor(GBM)_j - P_{i,j}^{iter}\right) \quad \ldots (7)$$

Here, '*w2*' is a random weight parameter whose value lies between [1.5,2].

Best Solution (Memory) Updation Phase

The updated position is checked for boundary limit, and it is ensured that the updated value must be within the lower and upper limits of the attribute. The fitness function is calculated for each updated particle, and if the updated particle has a better fitness value than the previous best, then the previous best for that particle is updated; otherwise, it remains the same. Mathematically, it is given by Eq. 8. Similarly, *GBM* and *GWM* are obtained during each successive iteration also gets updated.

$$MEM_i^{iter+1} = \begin{cases} P_i^{iter+1}, if, P_i^{iter+1} < MEM_i^{iter} \\ MEM_i^{iter}, otherwise \end{cases} \quad \ldots (8)$$

Termination Phase

The algorithm terminates when maximum iteration ($iter_{max}$) is reached.

Neighborhood Function

In this function, the mid-point of the lower limit and upper limit of each attribute of the data point is calculated, and its distance from the respective particle's position is obtained. A range of value is obtained by subtracting and adding half of the distance value to the respective particle' position, and it is updated by a random number taken from the range. For example, let us say Min_j and Max_j is the boundary limit of the j^{th} attribute. We first calculate the mid-point of Min_j and Max_j, which is Mid_j. After that, we calculate the distance of $P_{i,j}$ from Mid_j. After that, the distance D_j is subtracted and added from $P_{i,j}$ to get the lower and upper limit from which a random point is selected, which replaces $P_{i,j}$ in the updated position.

Algorithm 1*: *Neighborhood Function

$P_i: i^{th} Particle$

```
for j = 1:dim
  Mid_j = (Max_j + Min_j)/2;  D_j = Mid_j − P_{i,j};
  aux = |D_j|;
  if(D_j < 0)
    l1 = P_{i,j} − aux;   l2 = P_{i,j} + aux;
    if(l2 > Max_j)
      P'_{i,j} = l1 + (Max_j − l1) × random_number[0,1]
    else
      P'_{i,j} = l1 +  (l2 − l1) × random_number[0,1]
  else
    l3 = P_{i,j} − aux;   l4 = P_{i,j} + aux;
    if(l3 < Min_j)
      P'_{i,j} = Min_j + (l4 − Min_j) × random_number[0,1]
    else
      P'_{i,j} = l3 +  (l4 − l3) × random_number[0,1]
return P'_i
```

Algorithm 2*: *Memory Based Optimizer

$iter_{max}$: *Maximum iterations*, N: *Number of particles*
$alpha = 3$
Set $iter = 0$
Randomly initialize the particles (P)
Set Memory $= P$

while $iter < iter_{max}$
  Compute the fitness function of all the Particles
  Find Global Best Memory (GBM) *and*
    Global Worst Memory(GWM)
  $ctrl\_para1 = exp\left(-(iter/iter_{max})^{1/alpha}\right)$
  $ctrl\_para2 = 0.25$
  for $i = 1:N$
    $r\_0 = random\_number[0,1]$
    if$(ctrl\_para1 > r\_0)$
      $r\_1 = random\_number[0,1]$
      if$(ctrl\_para2 < r\_1)$
        Update the Position using Eq. 4
      else
        Update the Position using Eq. 5
    else
      $r\_2 = random\_number[0,1]$
      if$(ctrl\_para2 < r\_2)$
        Update the Position using Eq. 6
      else
        Update the Position using Eq. 7
    Check for boundary condition
    Update the Memory Matrix using Eq. 8
end while
Return the Global Best Memory(GBM)

Results and Discussion

In this section, implementation and obtained experimental results of MBO are discussed. MBO is implemented in Python 3.9 using ubuntu 22.04 operating system having i7 processor and 16 GB RAM. At first, MBO performance over CEC2019 Benchmark Functions is presented, subsequent section discusses the application of MBO for the feature selection problem.

MBO Performance over CEC2019 Benchmark Functions

The effectiveness of MBO was measured against the CEC 2019 Benchmark Functions as presented in Fig. 1. These benchmark functions comprise ten particularly difficult and intricate single-objective optimization problems. The details of these ten benchmark functions are given in Table 1.

The parameters of the algorithms used to compare the results obtained from benchmark functions are presented in Table 2. These parameters are based on prior research.[38] For F1, the number of iterations is set to 1800, while it is set to 2000 for the other functions. The mean value and standard deviation are calculated based on an average of over 30 runs. A performance comparison of MBO with other metaheuristic approaches across the CEC2019 Benchmark Functions is provided in Table 3.

As seen from Table 4, out of 10 benchmark functions, MBO has performed better in 5 functions, whereas DE and GSA have performed better in 3 and 2 benchmark functions respectively. An upward arrow sign denotes where MBO has performed better than other algorithms, and a downward arrow sign indicates where other algorithms performed better than the MBO. A value without a sign represents those comparisons where MBO is not involved.

The convergence graph of MBO and other algorithms over CEC2019 benchmark functions is illustrated in Fig. 2. It is clear from the graph that MBO has comparative convergence when compared with existing metaheuristic approaches.

Application of MBO for Feature Selection

Metaheuristic approaches have been successfully applied as a wrapper for selecting optimal features by several researchers in the past.[39] MBO is further utilized as a wrapper approach to determine optimal feature subsets, thus improving overall accuracy and other performance metrics and reducing the overall feature set.

The current study employs the k–NN (k–Nearest Neighbors) algorithm as a classifier to select optimal feature subsets. The value of k, which represents the number of nearest neighbors used in the k–NN algorithm, is set to 5.[40] The datasets are partitioned into a ratio of 8:2 for training and testing. The MBO method is evaluated alongside five other metaheuristic approaches in the context of feature selection tasks. The number of particles and maximum iteration is kept to be 20 and 100, respectively. The feature selection process utilizes a binary metaheuristic approach as a wrapper. The particles are initialized randomly within the range of 0 to 1. A fixed threshold value of 0.5 is used to select a specific feature. In the given scenario, if the value exceeds 0.5, the corresponding feature is chosen and used in the classification task. Conversely, if the value is below 0.5, the respective feature is not selected for participation. A detailed description of the metaheuristic algorithms along with their various parameters are discussed in Table 5.[41]

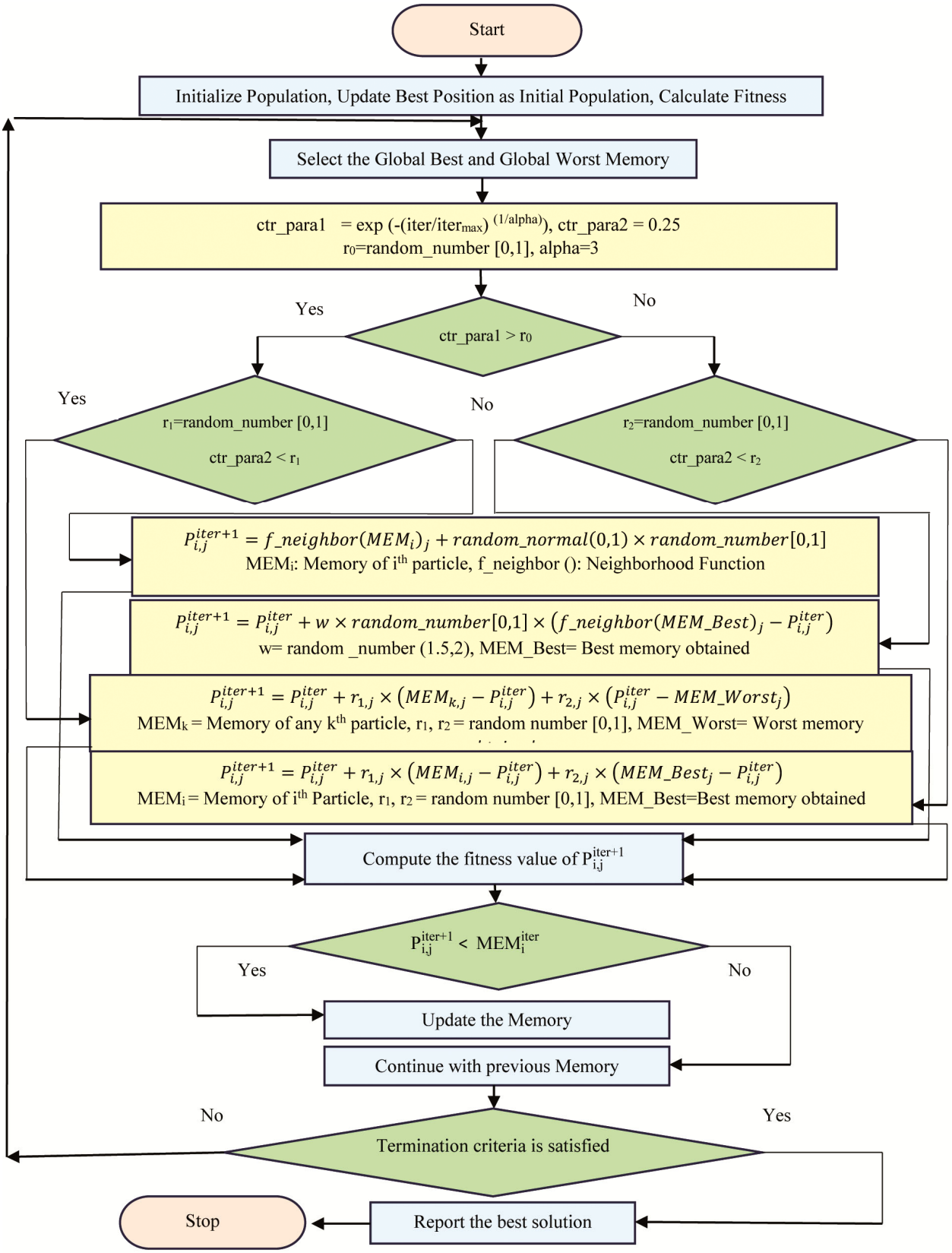

Fig. 1— Flowchart of MBO

Dataset

The efficacy of the MBO as a feature selection approach is evaluated over 12 real datasets take from UCI repository.[42] The description of the datasets is given in Table 6.

Fitness Function

In this study, we employ a fitness function that is a linear combination of the error rate and the fraction of

Table 1 — CEC2019 Benchmark functions

	Dimensions	Range	f_{min}	Nomenclature
CEC01	9	[–8192, 8192]	1	F01
CEC02	16	[– 16384, 16384]	1	F02
CEC03	18	[– 4, 4]	1	F03
CEC04	10	[– 100, 100]	1	F04
CEC05	10	[– 100, 100]	1	F05
CEC06	10	[– 100, 100]	1	F06
CEC07	10	[– 100, 100]	1	F07
CEC08	10	[– 100, 100]	1	F08
CEC09	10	[– 100, 100]	1	F09
CEC10	10	[– 100, 100]	1	F10

Table 2 — Algorithm parameters

Algorithm	Parameters Values
Memory based optimizer (MBO)	Number of particles = 50, (α, β, γ) = (3,0.1,0.35)
Crow search algorithm (CSA)	Number of particles = 50, Flight length = 2.0, Awareness probability = 0.1
Particle swarm optimization (PSO)	Number of particles = 50, Learning Coefficients = (2, 2) Inertial weight = (0.9, 0.4)
Jellyfish search optimizer (JSO)	Number of particles = 50, (η, β, γ, c_0) = (4,3,0.1,0.5)
Gravitational search algorithm (GSA)	Number of particles = 50, (Rnorm, Rpower, alpha, G_0) = (2,1,20,100)
Grey wolf optimizer (GWO)	Number of particles = 50, Control parameter(a) = (2,0)
Differential evolution (DE)	Number of particles = 50, Scheme = DE/rand/1, Scaling factor = 0.5, Crossover probability = 0.5
Arithmetic optimization algorithm (AOA)	Number of particles = 50, (μ, σ, MOA_Min, MOA_Max) = (0.499,5,0.1,1)
Harris hawk optimization (HHO)	Number of particles = 50
Jaya algorithm (JA)	Number of particles = 50
Student psychology based optimization (SPBO)	Number of particles = 50

Table 3 — Comparison of different algorithms over CEC2019 Benchmark Functions

		MBO	PSO	GWO	HHO	AOA	GSA	Jaya	JFO	DE	SPBO	CSA
F01	Best	**1.000E+00**	8.045E+05	**1.000E+00**	**1.000E+00**	1.001E+00	1.124E+08	2.262E+06	1.488E+07	**1.000E+00**	4.145E+04	2.430E+02
	Average	**5.890E+02**	2.335E+07	4.458E+03	3.003E+07	1.759E+09	3.866E+08	5.390E+06	3.755E+07	9.124E+03	2.160E+07	2.320E+04
F02	Best	**4.239E+00**	4.220E+02	1.214E+01	6.792E+00	1.352E+01	1.243E+04	2.880E+03	4.165E+03	1.594E+02	7.787E+02	3.080E+01
	Average	**4.346E+01**	5.110E+03	1.958E+02	7.346E+03	3.215E+04	2.448E+04	4.563E+03	6.890E+03	4.357E+02	4.008E+03	1.300E+02
F03	Best	**1.000E+00**	2.583E+00	**1.000E+00**	1.413E+00	**1.000E+00**	1.409E+00	6.109E+00	7.534E+00	**1.000E+00**	3.318E+00	1.030E+00
	Average	**1.502E+00**	6.994E+00	2.039E+00	2.909E+00	1.685E+00	6.804E+00	8.293E+00	8.954E+00	3.232E+00	8.421E+00	2.390E+00
F04	Best	6.970E+00	8.013E+00	4.918E+00	3.497E+01	7.661E+01	3.640E+01	2.146E+01	3.467E+01	**2.990E+00**	1.972E+01	7.960E+00
	Average	1.321E+01	3.098E+01	1.382E+01	7.072E+01	9.571E+01	4.436E+01	3.341E+01	5.173E+01	**8.486E+00**	5.982E+01	2.000E+01
F05	Best	1.030E+00	1.042E+00	1.074E+00	7.771E+00	1.490E+02	**1.000E+00**	2.227E+00	5.814E+00	1.020E+00	2.198E+00	1.050E+00
	Average	1.118E+00	1.828E+01	1.531E+00	2.838E+01	1.613E+02	**1.002E+00**	2.412E+00	1.487E+01	1.070E+00	1.660E+01	1.160E+00
F06	Best	1.012E+00	1.000E+00	1.128E+00	6.378E+00	4.383E+00	**1.000E+00**	5.570E+00	7.816E+00	**1.000E+00**	4.075E+00	1.420E+00
	Average	**1.761E+00**	4.479E+00	2.326E+00	9.659E+00	6.393E+00	2.716E+00	6.963E+00	9.081E+00	2.160E+00	8.393E+00	3.770E+00
F07	Best	4.806E+00	2.282E+02	1.253E+02	1.026E+03	8.576E+02	1.207E+03	6.680E+02	8.884E+02	**1.500E+00**	6.057E+02	3.690E+02
	Average	5.588E+02	7.832E+02	5.815E+02	1.480E+03	1.581E+03	1.656E+03	1.145E+03	1.416E+03	**3.114E+02**	1.511E+03	9.640E+02
F08	Best	2.355E+00	3.121E+00	2.551E+00	3.479E+00	4.512E+00	4.841E+00	3.957E+00	4.020E+00	**1.845E+00**	1.517E+01	2.720E+00
	Average	3.157E+00	3.939E+00	3.473E+00	4.722E+00	4.961E+00	5.307E+00	4.355E+00	4.522E+00	**2.840E+00**	5.661E+01	3.780E+00
F09	Best	1.061E+00	1.097E+00	1.067E+00	1.224E+00	1.419E+00	**1.022E+00**	1.332E+00	1.637E+00	1.074E+00	1.195E+00	1.120E+00
	Average	1.145E+00	1.410E+00	1.137E+00	1.891E+00	1.525E+00	**1.036E+00**	1.508E+00	1.996E+00	1.122E+00	1.846E+00	1.170E+00
F10	Best	**1.001E+00**	2.102E+01	2.117E+01	2.106E+01	2.101E+01	**1.000E+00**	2.118E+01	2.117E+01	2.105E+01	2.101E+01	2.150E+00
	Average	**1.531E+01**	2.119E+01	2.135E+01	2.120E+01	2.110E+01	1.564E+01	2.137E+01	2.137E+01	2.115E+01	2.132E+01	1.750E+01

Table 4 — Comparison of average results of best performing algorithms

	Best		2nd Best		3rd Best		Best vs. 2nd Best	Best vs. 3rd Best	2nd Best vs. 3rd Best
F01	MBO	588.985	GWO	4458.370	DE	9124.184	656.958 ↑	1449.137 ↑	51.137
F02	MBO	43.463	GWO	195.801	DE	435.696	350.500 ↑	902.453 ↑	55.060
F03	MBO	1.501	AOA	1.685	GWO	2.039	12.258 ↑	35.843 ↑	17.361
F04	DE	8.486	MBO	13.20	GSA	13.824	55.550 ↓	62.904	4.514 ↑
F05	GSA	1.002	DE	1.069	MBO	1.118	6.687	11.577	4.383 ↓
F06	MBO	1.760	DE	2.159	GWO	2.326	22.670 ↑	32.159 ↑	7.180
F07	DE	311.356	MBO	558.791	GWO	581.464	79.470 ↓	86.752 ↓	3.899
F08	DE	2.8404	MBO	3.156	GWO	3.473	11.111 ↓	22.272 ↓	9.128
F09	GSA	1.036	DE	1.121	GWO	1.136	8.205	9.266	1.320
F10	MBO	15.313	GSA	15.64	CSA	17.58	2.135 ↑	14.804 ↑	11.035

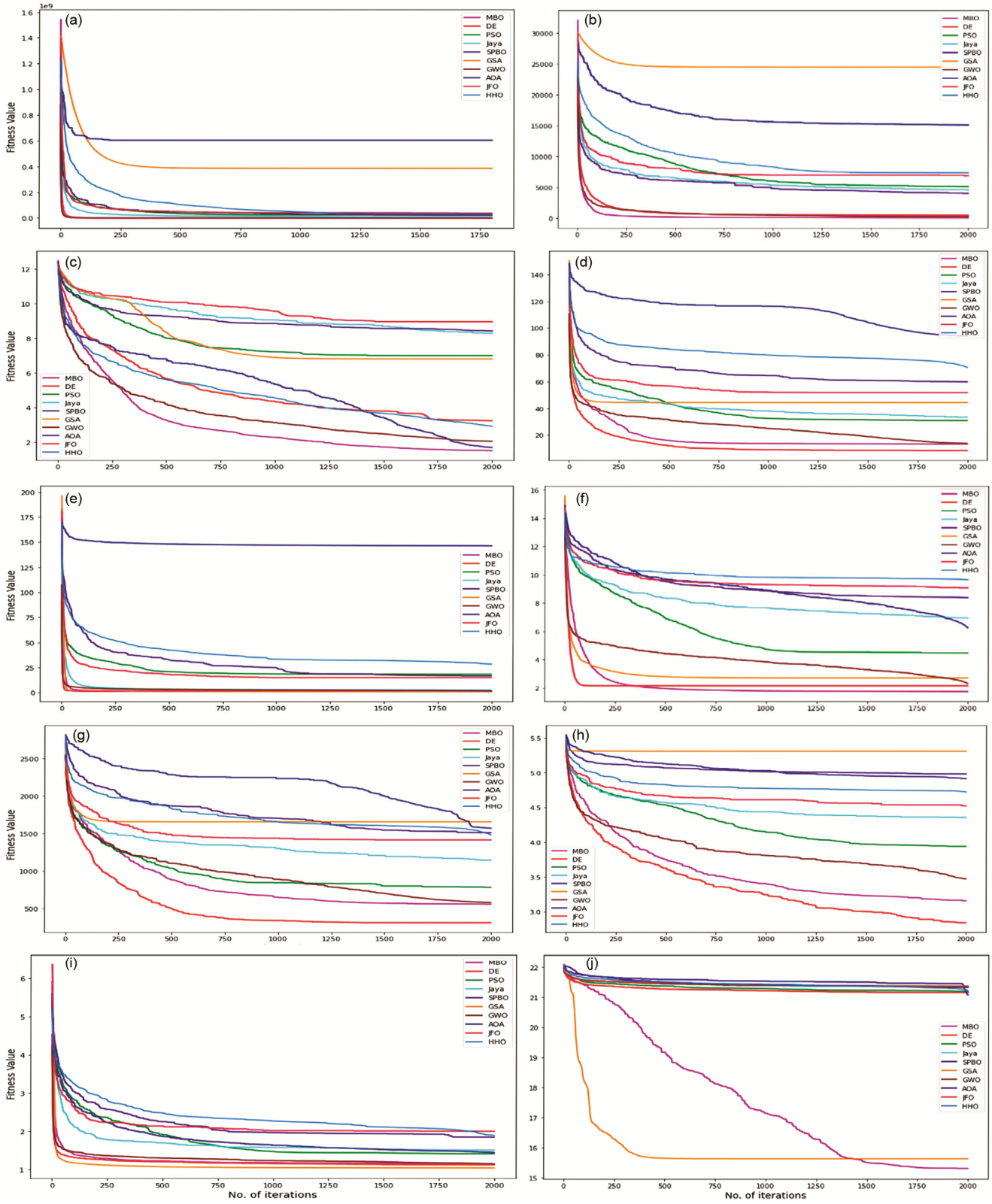

Fig. 2 — Convergence Graph for: (a) CEC01, (b) CEC02, (c) CEC03, (d) CEC04, (e) CEC05, (f) CEC06, (g) CEC07, (h) CEC08, (i) CEC09, (j) CEC10

Table 5 — Metaheuristic algorithm details for feature selection

Algorithm	Parameters Values
Memory based optimizer (MBO)	Number of particles = 20, (α, β, γ) = (3,0.1,0.35)
Particle swarm optimization (PSO)	Number of particles = 20, Learning Coefficients = (2, 2), Inertial Weight = (0.9, 0.4)
Grey wolf optimizer (GWO)	Number of particles = 20, Control parameter (a) = (2, 0)
Differential evolution (DE)	Number of particles = 20, Scheme = DE/rand/1, Scaling factor = 0.5, Crossover probability = 0.9
Harris hawk optimization (HHO)	Number of particles = 20
Student psychology based optimization (SPBO)	Number of particles = 20

Table 6 — Datasets

Datasets	Features	Class	Instances	Nomenclature
Breast cancer	10	2	699	D1
Breast EW	31	2	569	D2
Heart EW	13	2	270	D3
Ionosphere	34	2	351	D4
Libras	90	15	360	D5
Lymphography	18	4	148	D6
Parkinsons	23	2	197	D7
Sonar	60	2	208	D8
Vehicle	18	4	846	D9
Wine	13	3	178	D10
Zoo	16	7	101	D11
Segmentation	19	7	210	D12

features that were chosen. It is expressed mathematically as Eq. 9.

$$Fitness\ Function = \alpha \times (error\_rate) + (1-\alpha) \times \left(\frac{Number\ of\ Selected\ Features}{Total\ Number\ of\ Features}\right) \quad \ldots (9)$$

In this work α is taken as 0.9.

Performance Metrices

To assess the effectiveness of MBO in comparison to other metaheuristic approaches, a comparative analysis is conducted using four performance metrics. The metrics utilized in this study include average accuracy and F1 score.[43] Beside this average feature size, and average execution time is also used to measure the relative performance of MBO with other metaheuristic approaches. Average accuracy and F1 score are crucial evaluation metrics in feature selection tasks using a wrapper approach. Average accuracy measures the classifier's overall performance across different subsets of features. It reflects how well the selected features generalize across the dataset, helping to avoid overfitting. F1 score, the harmonic mean of precision and recall, is particularly relevant when dealing with imbalanced datasets. It emphasizes the trade-off between false positives and false negatives, offering insight into the quality of selected features for both minority and majority classes. Together, these metrics guide the wrapper approach in selecting the optimal feature subset, ensuring accuracy and balanced performance across all classes. All the evaluation metrics is averaged over 21 independent runs.

The average accuracy, F1 score, and the number of selected features calculated over 21 independent runs are shown in Table 7. The results indicate that MBO has achieved better accuracy in 11 out of 12 datasets. In the case of Parkinson, PSO has obtained better results than the other approaches in both average accuracy and F1 score. When considering the F1 score, MBO also outperformed others in 11 datasets. Regarding the number of selected features, MBO obtained the smallest feature set in 3 out of 12 datasets. GWO selected the minimum number of features in 7 datasets, and HHO in 2 datasets.

To further demonstrate the effectiveness of MBO in feature selection task, it is compared to 4 different works from the literature, Adaptive Black Widow Optimization with Selection strategyand Differential mutation (SDABWO) Hu *et al.*[41], New Binary version of the Grasshopper Optimization Algorithm (NBGOA) Hichem *et al.*[44], Chebyshev Binay Gaining sharing knowledge-based optimization algorithm (CBI-GSK) Agarwal *et al.*[45], and Sine Cosine Algorithm (SCA) and Genetic Algorithm (GA) (SCAGA), called SCAGA Abualigah and Dulaimi[46] over the average accuracy obtained and is given in Table 8 as it is clear from the table that the MBO has performed better in 75% of the datasets in terms of average accuracy level.

The average execution time obtained over 21 independent iterations is illustrated in Fig. 3. Among the 12 datasets, MBO achieved better results in 9 datasets, DE in 2 datasets, and GWO recorded the minimum execution time in 1 dataset. As anticipated, the vehicle dataset had the maximum execution time across all algorithms, while the zoo dataset had the minimum execution time. The overall results show that the MBO has demonstrated its effectiveness through rigorous evaluation. Initially, MBO is tested on the 2019 CEC Benchmark functions and compared with 10 established meta-heuristic approaches, outperforming them in 5 out of 10 continuous

Table 7 — Results (R1—Average accuracy, R2—Average F1 score, R3—Average number of features selected)

Datasets		MBO	PSO	GWO	DE	HHO	SPBO
	R1	**97.6530**	97.0068	97.0068	97.2449	96.5306	97.1088
D1	R2	**0.9740**	0.9670	0.9668	0.9696	0.9615	0.9681
	R3	**2.380**	2.429	2.429	2.476	2.381	3.000
	R1	**96.7000**	95.2799	95.0292	94.9039	95.4052	95.0710
D2	R2	**0.9641**	0.9488	0.9460	0.9446	0.9500	0.9467
	R3	3.047	4.905	**2.762**	3.476	4.000	10.143
	R1	**89.9470**	88.3598	87.5661	87.7425	82.2751	82.8042
D3	R2	**0.8975**	0.8818	0.8727	0.8753	0.8192	0.8242
	R3	3.952	3.810	3.667	4.857	**3.524**	5.048
	R1	**97.1830**	94.1650	95.7076	95.1710	93.0612	88.7755
D4	R2	**0.9688**	0.9338	0.9521	0.9457	0.9222	0.8705
	R3	6	9.286	4.810	6.857	**4.571**	13.381
	R1	**86.9047**	84.1270	86.3757	84.9868	81.0847	78.3730
D5	R2	**0.8648**	0.8379	0.8606	0.8487	0.8078	0.7782
	R3	26.619	35.476	**17.571**	33.143	21.524	40.476
	R1	**95.2380**	90.9524	93.0159	93.4921	88.2540	88.2540
D6	R2	**0.8277**	0.7214	0.7358	0.7699	0.6343	0.6867
	R3	6.380	6.143	**5.667**	6.429	6.333	7.619
	R1	93.7728	**93.8950**	91.3309	92.6740	92.0635	89.0110
D7	R2	0.9123	**0.9133**	0.8743	0.8932	0.8883	0.8426
	R3	2.952	3.429	**2.476**	3.143	4.238	7.524
	R1	**97.3922**	94.8980	96.1451	96.3719	91.6100	88.6621
D8	R2	**0.9737**	0.9484	0.9612	0.9635	0.9151	0.8848
	R3	16.285	19.905	**11.143**	18.810	18.000	25.381
	R1	**76.2464**	75.4902	75.8824	75.2661	72.0728	71.3445
D9	R2	**0.7563**	0.7446	0.7514	0.7449	0.7094	0.7046
	R3	**6.238**	7.143	6.429	6.333	7.429	7.333
	R1	**99.2063**	98.8095	98.9418	98.5450	95.7672	93.2540
D10	R2	**0.9926**	0.9889	0.9898	0.9860	0.9591	0.9333
	R3	**3.2857**	3.762	3.619	3.524	4.000	4.952
	R1	**99.0929**	97.2789	98.6395	98.1859	96.1451	95.9184
D11	R2	**0.9745**	0.9331	0.9565	0.9460	0.8965	0.8960
	R3	5.238	5.190	**5.048**	5.238	6.524	6.952
	R1	**94.1043**	91.9501	90.9297	90.4762	89.5692	88.4354
D12	R2	**0.9388**	0.9181	0.9076	0.9026	0.8922	0.8811
	R3	4.904	5.476	**4.524**	4.810	5.857	7.857

Table 8 — Comparative analysis of average accuracy from the literature

Datasets	MBO	SDABWO	NBGOA	CBI-GSK	SCAGA
D1	**97.6530**	97.5540	96.502	97.30	97.0143
D2	**96.7000**	95.8407	88.238	95.47	96.2105
D3	89.9470	86.9444	**94.965**	85.78	82.7407
D4	**97.1830**	95.7143	94.810	93.41	92.6705
D5	86.9047	80.4861	**91.020**	NA	NA
D6	**95.2380**	92.9310	NA	52.00	87.0816
D7	93.7728	**99.6154**	95.023	NA	NA
D8	**97.3922**	94.1463	92.987	91.83	92.8365
D9	**76.2464**	73.2840	NA	NA	NA
D10	**99.2063**	98.0000	NA	96.00	97.9213
D11	**99.0929**	98.0000	93.230	96.37	96.7461
D12	**94.1043**	91.6667	NA	NA	NA

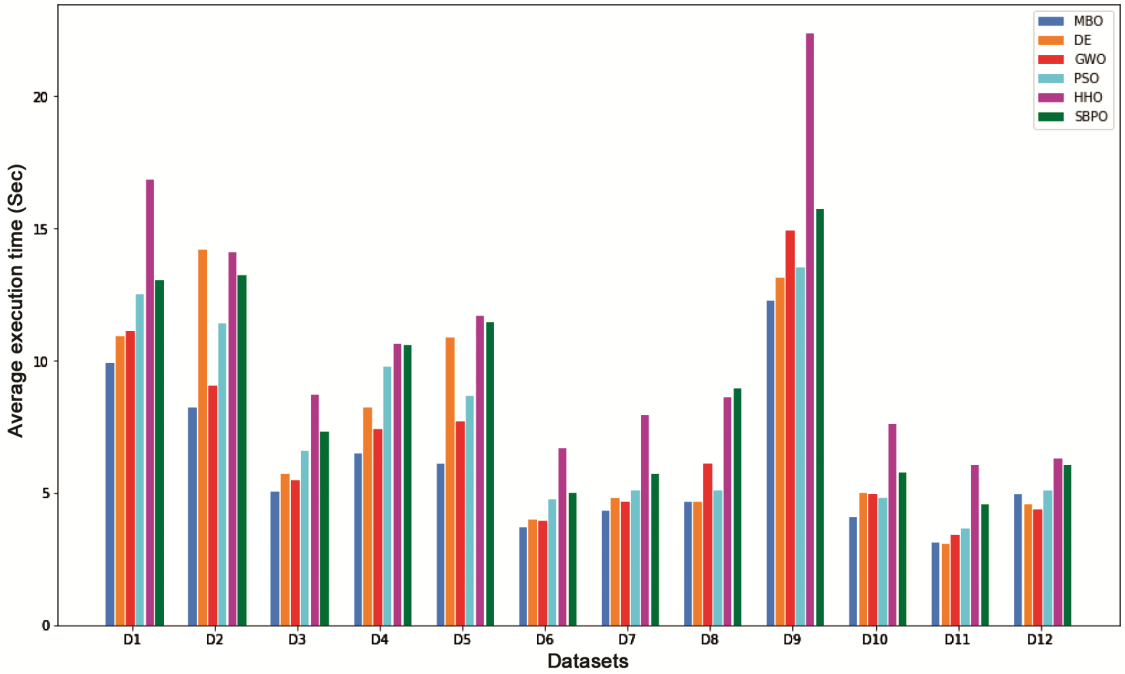

Fig. 3 — Average execution time of different datasets

benchmark functions, proving its capability in continuous optimization problems. Further, the MBO convergence curve demonstrates superior performance over the CEC2019 Benchmark functions. MBO achieves faster and more stable convergence, indicating its efficiency in navigating the solution space and quickly approaching optimal solutions. This improved convergence highlights MBO's effectiveness in balancing exploration and exploitation, leading to enhanced accuracy and consistency compared to existing algorithms. In subsequent experiments, MBO is applied as a wrapper method for feature selection, a discrete optimization task that achieves higher average accuracy in 11 out of 12 datasets and higher average execution times in 9 out of 12 datasets. These results demonstrate that MBO is effective in both continuous and discrete optimization tasks, producing competitive outcomes compared to other well-known meta-heuristic approaches, suggesting its applicability in complex optimization problems across various domains.

Conclusions

This study introduces Memory Based Optimizer (MBO), a novel metaheuristic technique that leverages the global best and worst positions and each particle's best position across iterations. MBO is first evaluated using the CEC 2019 Benchmark Functions and compared against ten well-established metaheuristic algorithms representing Evolutionary, Swarm Intelligence, Physics-based, and Human Intelligence based approaches. The results demonstrate that MBO performs competitively under equivalent conditions. Additionally, MBO is applied as a wrapper for feature selection across 12 diverse UCI datasets and is evaluated against five other metaheuristic techniques. The results show that MBO performs comparably regarding average accuracy, F1 score, number of features selected, and execution time. Furthermore, MBO is compared with four recently introduced wrapper approaches, outperforming them in 75% of the datasets used in the simulation. This highlights MBO's superior performance in both continuous and discrete optimization tasks. MBO shows significant potential for application in various optimization problems, including structural engineering, healthcare, and data analysis, with opportunities for further improvement through algorithmic modifications.

Conflict of Interest

The authors have no conflicts of interest to disclose.

References

1 Shayanfar H & Gharehchopogh F S, Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems, *Appl Soft Comput*, **71** (2018) 728–746, https://doi.org/10.1016/j.asoc.2018.07.033.

2 Kumar V, Chhabra J K & Kumar D, Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems, *J Comput Sci-Neth*, **5(2)** (2014) 144–155, https://doi.org/10.1016/j.jocs.2013.12.001.

3 Gao S & de Silva C W, Estimation distribution algorithms on constrained optimization problems, *Appl Math Comput*, **339** (2018) 323–345, https://doi.org/10.1016/ j.amc.2018.07.037.

4 Wu G, Pedrycz W, Suganthan P N & Mallipeddi R, A variable reduction strategy for evolutionary algorithms handling equality constraints, *Appl Soft Comput*, **37** (2015) 774–786, https://doi.org/10.1016/j.asoc.2015.09.007.

5 Ranjan R & Chhabra J K, Automatic feature selection using enhanced dynamic Crow Search Algorithm, *Int J Inf Tecnol*, **15** (2023) 2777–2782, https://doi.org/10.1007/s41870-023-01319-2.

6 Morales-Castaneda B, Zaldivar D, Cuevas E, Rodriguez A & Navarro M A, Population management in metaheuristic algorithms: Could less be more?, *Appl Soft Comput*, **107** (2021) 107389, https://doi.org/10.1016/j.asoc.2021.107389.

7 Swan J, Adriaensen S, Brownlee A E, Hammond K, Johnson C G, Kheiri A & White D R, Metaheuristics "in the large", *Eur J Oper Res*, **297(2)** (2022) 393–406, https://doi.org/10.1016/j.ejor.2021.05.042.

8 Banaie-Dezfouli M, Nadimi-Shahraki M H & Beheshti Z, BE-GWO: Binary extremum-based grey wolf optimizer for discrete optimization problems, *Appl Soft Comput*, **146** (2023) 110583, https://doi.org/10.1016/j.asoc.2023.110583.

9 Kohavi R & John G H, Wrappers for feature subset selection, *Artif Intell*, **97(1–2)** (1997) 273–324, https://doi.org/10.1016/S0004-3702(97)00043-X.

10 Duda R O & Hart P E, *Pattern classification* (John Wiley & Sons), 2006.

11 Dokeroglu T, Deniz A & Kiziloz H E, A comprehensive survey on recent metaheuristics for feature selection, *Neurocomputing*, **494** (2022) 269–296, https://doi.org/10.1016/ j.neucom.2022.04.083.

12 Wolpert D H & Macready W G, No free lunch theorems for optimization, *IEEE Trans Evolut Comput*, **1(1)** (1997) 67–82, doi: 10.1109/4235.585893.

13 Price K V, Awad N H, Ali M Z & Suganthan P N, The 100-digit challenge: Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization, *Nanyang Technological University*, 2018.

14 Dash M & Liu H, Feature selection for classification, *Intell Data Anal*, **1(1–4)** (1997) 131–156, https://doi.org/10.1016/S1088-467X(97)00008-5.

15 Abualigah L, Shehab M, Alshinwan M & Alabool H, Salp swarm algorithm: a comprehensive survey, *Neural Comput Appl*, **32** (2020) 11195–11215, https://doi.org/10.1007/s00521-019-04629-4.

16 Holland J H, Genetic Algorithms, *Sci Am*, **267(1)** (1992) 66–73.

17 Hansen N, Müller S D & Koumoutsakos P, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evol Comput*, **11(1)** (2003) 1–18, doi: 10.1162/106365603321828970.

18 Storn R & Price K, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *J Global Optim*, **11** (1997) 341–359, https://doi.org/10.1023/A:1008202821328.

19 Al-Sorori W & Mohsen A M, New Caledonian crow learning algorithm: A new metaheuristic algorithm for solving continuous optimization problems, *Appl Soft Comput*, **92** (2020) 106325, https://doi.org/10.1016/j.asoc.2020.106325.

20 Kennedy J & Eberhart R, Particle Swarm Optimization, *Proceedings of ICNN'95-international conference on neural networks* (IEEE), (Perth, WA, Australia), 1995, 1942–1948, doi: 10.1109/ICNN.1995.488968.

21 Mirjalili S, Mirjalili S M & Lewis A, Grey wolf optimizer, *Adv Eng Softw*, **69** (2014) 46–61, https://doi.org/10.1016/j.advengsoft.2013.12.007.

22 Karaboga D & Basturk B, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J Global Optim*, **39** (2007) 459–471, https://doi.org/10.1007/s10898-007-9149-x.

23 Heidari A A, Mirjalili S, Faris H, Aljarah I, Mafarja M & Chen H, Harris hawks optimization: Algorithm and applications, *Fut Gener Compt Syt*, **97** (2019) 849–872, https://doi.org/10.1016/j.future.2019.02.028.

24 Askarzadeh A, A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, *Comput Struct*, **169** (2016) 1–12, https://doi.org/10.1016/j.compstruc.2016.03.001.

25 Kirkpatrick S, Gelatt Jr C D & Vecchi M P, Optimization by Simulated Annealing, *Science*, **220(4598)** (1983) 671–680, doi: 10.1126/science.220.4598.671.

26 Rashedi E, Nezamabadi-Pour H & Saryazdi S, GSA: a gravitational search algorithm, *Inform Sci*, **179(13)** (2009) 2232–2248, https://doi.org/10.1016/j.ins.2009.03.004.

27 Mirjalili S, Mirjalili S M & Hatamlou A, Multi-verse optimizer: A nature-inspired algorithm for global optimization, *Neural Comput Appl*, **27** (2016) 495–513, https://doi.org/10.1007/s00521-015-1870-7.

28 Atashpaz-Gargari E & Lucas C, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. *IEEE congress on evolutionary computation* (Singapore), 2007, 4661–4667, doi: 10.1109/CEC.2007.4425083.

29 Glover F, Taillard E & Taillard E, A user's guide to tabu search, *Ann Oper Res*, **41(1)** (1993) 1–28, https://doi.org/10.1007/BF02078647.

30 Rao R V, Savsani V J & Vakharia D P, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, *Comput Aided Design*, **43(3)** (2011) 303–315, https://doi.org/10.1016/j.cad.2010.12.015.

31 Das B, Mukherjee V & Das D, Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems, *Adv Eng Softw*, **146** (2020) 102804, https://doi.org/10.1016/j.advengsoft.2020.102804.

32 Kennedy J & Eberhart R C, A discrete binary version of the particle swarm algorithm, *International conference on systems, man, and cybernetics. Computational cybernetics and simulation (IEEE)*, (Orlando, FL, USA), (1997) 4104–4108, doi: 10.1109/ICSMC.1997.637339.

33 Engelbrecht A P, Grobler J & Langeveld J, Set based particle swarm optimization for the feature selection problem, *Eng Appl Artif Intel*, **85** (2019) 324–336, https://doi.org/10.1016/j.engappai.2019.06.008.

34 Emary E, Zawbaa H M & Hassanien A E, Binary grey wolf optimization approaches for feature selection,

Neurocomputing, **172** (2016) 371–381, https://doi.org/10.1016/j.neucom.2015.06.083.

35 Ouadfel S & Abd Elaziz M, Enhanced crow search algorithm for feature selection, *Expert Syst Appl*, **159** (2020) 113572, https://doi.org/10.1016/j.eswa.2020.113572.

36 Taradeh M, Mafarja M, Heidari A A, Faris H, Aljarah I, Mirjalili S & Fujita H, An evolutionary gravitational search-based feature selection, *Inform Sci*, **497** (2019) 219–239, https://doi.org/10.1016/j.ins.2019.05.038.

37 Too J, Abdullah A R & Mohd Saad N, A new quadratic binary harris hawk optimization for feature selection, *Electronics*, **8(10)** (2019) 1130, https://doi.org/10.3390/electronics8101130.

38 Bogar E & Beyhan S, Adolescent Identity Search Algorithm (AISA): A novel metaheuristic approach for solving optimization problems, *Appl Soft Comput*, **95** (2020) 106503, https://doi.org/10.1016/j.asoc.2020.106503.

39 Tang J, Alelyani S & Liu H, Feature selection for classification: A review, *Data Classification: Algorithms and Applications*, (2014) 37.

40 Liu W & Wang J, Recursive elimination–election algorithms for wrapper feature selection, *Appl Soft Comput*, **113** (2021) 107956, https://doi.org/10.1016/j.asoc.2021.107956.

41 Hu G, Du B, Wang X & Wei G, An enhanced black widow optimization algorithm for feature selection, *Knowl-Based Syst*, **235** (2022) 107638, https://doi.org/10.1016/j.knosys.2021.107638.

42 Asuncion A & Newman D, *UCI Machine Learning Repository*, (2007).

43 Thirumoorthy K, A feature selection model for software defect prediction using binary Rao optimization algorithm, *Appl Soft Comput*, **131** (2022) 109737, https://doi.org/10.1016/ j.asoc.2022.109737.

44 Hichem H, Elkamel M, Rafik M, Mesaaoud M T & Ouahiba C, A new binary grasshopper optimization algorithm for feature selection problem, *J King Saud Univ–Comput Inf Sci*, **34(2)** (2022) 316–328, https://doi.org/10.1016/ j.jksuci.2019.11.007.

45 Agrawal P, Ganesh T & Mohamed A W, Chaotic gaining sharing knowledge-based optimization algorithm: An improved metaheuristic algorithm for feature selection, *Soft Comput*, **25(14)** (2021) 9505–9528, https://doi.org/10.1007/s00500-021-05874-3.

46 Abualigah L & Dulaimi A J, A novel feature selection method for data mining tasks using hybrid sine cosine algorithm and genetic algorithm, *ClusterComput*, **24** (2021) 2161–2176, https://doi.org/10.1007/s10586-021-03254-y.