OPEN ACCESS INTERNATIONAL JOURNAL OF SCIENCE & ENGINEERING

MOP-MFO: an improved moth flame optimization algorithm for complex optimization problems

Chitaranjan Dash[1], Kanakprava Biswal[2]
Department of Mathematics, Government College of Engineering, Kalahandi Bhawanipatna-766002,Odisha, India

---

Abstract: ***The transverse-orientation process is at the heart of the new and innovative moth-flame optimization (MFO) technique, which draws inspiration from nature. This device incorporates a specialised form of navigational techniques that are symbolic of the direct flight of moths toward the moon at night. Although MFO has been successfully applied to a number of optimization problems, it still has the same issues as other evolutionary algorithms, namely a sluggish convergence rate and a lack of global search capabilities. To remedy this, we introduce the Mathematical Operator Based MFO Algorithm (or MOP-MFO for short). There are two stages to the planned MOP-MFO. After the position update phase of the original MFO algorithm, a simple quadratic interpolation approach is added to accelerate the convergence rate. The non-linear operator is introduced to keep a healthy balance between expansion and specialisation. The suggested MOP-MFO, MFO has been put through its paces by being compared to 36 traditional benchmark functions picked from the literature, and then statistically tested using a procedure called the Friedman rank test. Two engineering design challenges have also been solved using the MOP-MFO to demonstrate its problem-solving potential. These findings confirmed that the suggested MOP-MFO out performed competing optimization techniques.***

Keywords: Moth flame optimization, Evolutionary Algorithms, Quadratic interpolation. Benchmark functions.

I. INTRODUCTION

Researchers are focusing more and more on machine learning and AI techniques because they have a whole large range of practical utility and that are used to solve several problems in the real world (both constrained and unconstrained, linear and nonlinear, continuous and discontinuous) with relative ease [1-2]. Due to the foregoing heterogeneity, traditional methods involving mathematical programming or numerical method, such as the quasi-Newton method, conjugate gradient, SQP, and rapid steepest approach [3-4], present a number of challenges when applied to these situations. Experimental evidence from a wide range of studies [5] shows that none of the aforementioned approaches is adequate for dealing with the complexity of real-world multimodal problems that are neither continuous nor differentiable. Because of its straightforwardness and versatility, the meta-heuristics algorithm has proven indispensable in the face of numerous challenges. Optimization typically use resident-based algorithms to provide optimum and sub-optimal solutions that are close to but not identical to the true optimal value. Algorithms like this often start with a random collection of initial solutions and repeatedly improve those answers until they reach the global optimum.

The study of optimization techniques spans a vast expanse, and progress in this domain is rapid. Few examples are discussed in [6-18]. We examine the MFO algorithm in detail here. In 2015, Mirjalili [19] discovered MFO, an algorithm based on swarm intelligence. The transverse orientation used by moths to find their way around in the wild served as inspiration for MFOs. The creator of the MFO has demonstrated that, over 29 different benchmark functions, it outperforms the other major metaheuristic algorithms. The fundamental benefit of MFO over all other conventional algorithms is the robust capability to tackle a wide variety of difficult issues involving confined and unknown search spaces.

Due to its straightforward approach and numerous benefits, the MFO algorithm has seen widespread use in the past few years, particularly in the fields of science and engineering. Applications of MFO have been demonstrated in [20], [21], and [22]. A middle ground must be found between the two, the authors of [23] employ three novel methods: iterative division, the Cauchy distribution function, and the best flame methodology. To speed up the convergence and getting rid of local optimum stagnation, Zhao et al. [25] proposed multiswarm improved moth flame optimization (MIMFO) with the aid of chaotic and dynamic grouping mechanism to improve the population diversity. Further, they incorporated linear and spiral search strategies and Gaussian mutation to enhance the searching ability and to maintain a diversification-intensification balance. An improved MFO (I-MFO) method was created by Nadimi Sahakai et al. [26] that assists in locating captured moths at regional maxima by characterizing their individual memories. Adapted walking around search (AWAS) is a common method used by trapped

moths to escape the local optima that has trapped them. Recently, Li et al. [28] created the ODSMFO method with the help of the OBL mechanism and DE (differential evolution), as well as an enhanced local search technique and the death mechanism for diversity enhancement. Shan et al. [29] showed that the MFO algorithm may be stabilized using the double adaptive weight mechanism (WEMFO) and to test its effectiveness, the WEMFO was utilized to train kernel extreme learning machines (KELMs). An enhanced MFO (EMFO) was developed by Sahoo et al. [30] by embedding the mutualism strategy on the basic MFO for a better balance between the search processes by enhancing its searching capability.

Several strategies were offered to boost MFO performance. High-dimensional complicated optimization problems still need advancements in areas like convergence rate, exploration/exploitation balance, and finding the global optimum. To solve global optimization problems, Millie Pant et al. [31] combine quadratic interpolation with a tweaked version of particle swarm optimization; the resulting experimental performance outperforms more conventional techniques. Better balancing of intensification and diversity is required, Yongjun Sun et al. [32] recently created a quadratic interpolation method in the WOA algorithm they named QIWOA, which has yielded better results than more conventional meta-heuristic algorithms. Therefore, by motivating from above articles, we've integrated a different type of non-linear operator with quadratic interpolation into the basic MFO algorithm to achieve similar performance gains to those shown in the aforementioned studies. It has been determined that MOP-MFO outperforms the other Metaheuristic optimization techniques by comparing their results to those obtained using the proposed MOP-MFO algorithm, we have taken from the literature having a set of 36 benchmark test functions.

In the remaining portions of this article, we will do the following: In Sect. 2, In this article, we present a high-level summary of the MFO algorithm. Specifically, Sect. 3 demonstrates the suggested MOP-MFO algorithm. Sect. 4 contains the simulation results and the performance evaluation, convergence study, as well as some benchmark functions. In Section 5, we see how this method can be used to a real world scenario, and the last Section 6, last but not least, we arrive at a few conclusions.

Moth flame optimization

Moths, as an insect, are classified under the phylum Arthropoda. Researchers are intrigued by moth navigation because of its apparent uniqueness. Moths navigate via a transverse orientation mechanism and fly at night so that they can take advantage of the moon's light. They use the moon's beam to fly in a line like an arrow across the sky by maintaining a constant with regard to the Moon's inclination. The moth's inclination is most effective when the distance to the flame is small; in this case, the moth will fly in a helical pattern around the flame, bringing it closer to the source of light. Using mathematical modelling and moth behaviour, Mirjalili created the MFO algorithm in 2015.

MFO Algorithm:

For each moth, the original MFO algorithm generates a list of potential solutions. Every moth's location is encoded as a vector of alternatives. First, let's examine this moth matrix.

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

Where, $X_i = [m_{i,1}, m_{i,2}, \dots, m_{i,n}], i \in \{1, 2, \dots, N\}$.

N is the no. of moths in the original population, while 'n' denotes numbers of variable. The moth's fitness of x vector is as follows:

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

As important as the MFO method is, the flame matrix plays a secondary role. We have consider the following flame matrix ($\boldsymbol{FMm}$)

$$\boldsymbol{FMm} = \begin{bmatrix} FMm_1 \\ FMm_2 \\ \vdots \\ FMm_N \end{bmatrix} = \begin{bmatrix} Fm_{1,1} & Fm_{1,2} & \cdots & Fm_{1,n-1} & Fm_{1,n} \\ Fm_{2,1} & \ddots & \cdots & \cdots & Fm_{2,n} \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ Fm_{N-1,1} & \cdots & \cdots & \ddots & Fm_{N-1,n} \\ Fm_{N,1} & Fm_{N,2} & \cdots & Fm_{N-1} & Fm_{N,n} \end{bmatrix} \quad (3)$$

Now we can keep track of (FMx) fitness in the following matrix:

$$\boldsymbol{Fit}[\boldsymbol{FMm}] = \begin{bmatrix} Fit(FMm_1) \\ Fit(FMm_2) \\ \vdots \\ Fit(FMm_N) \end{bmatrix} \quad (4)$$

Here Fit $(*)$ represents fitness function candidate solution. Moth and flame are two important components of MFO algorithm. Moth moves spirally when it nearer to the flame therefore, author used a logarithmic spiral function which is as follows:

$$m_i^{K+1} = \begin{cases} \delta_i \cdot e^{bt} \cdot cos(2\pi t) + Fm_i(k), & i \le N.FM \\ \delta_i \cdot e^{bt} \cdot cos(2\pi t) + Fm_{N.FM}(k), & i \ge N.FM \end{cases} \quad (5)$$

Where, $\delta_i = |m_i^K - Fm_i|$

indicates the moth's flight distance and its own unique flame (Fm_i) at the i^{th} location, and the random number between -1 and 1 be t. Here b is a fixed constant equal one used to recognize the spiral flight shape.

$$a_1 = -1 + current\ iteration\left(\frac{-1}{max_{iter}}\right) \quad (6)$$

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

where, max_{iter} implies the max no. of iterations, the global and local mechanisms at work in the standard MFO method are shown by the linear decline of the convergence constant a_1 from (-1) to (-2).

Current and prior flame positions are gathered and organised by global and local fitness in every iteration. Other flames are extinguished, and only the finest $N.FM$ flames are maintained. 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. According to the amount of flames (N.FM) that have been lowered throughout and iteration can be calculated using the formula below.

$$N.FM = round\left(N.FM_{Lst\ it} - crnt.it\ \frac{(N.FM_{Lst\ it}-1)}{max\ it}\right) \quad (8)$$

More information regarding MFO may be found in reference [9].

Proposed MOP-MFO algorithm

The two main characteristics of the metaheuristic algorithms include exploration and intensification of the search space and make a proper balance between the two. Diversification involves searching the entire region, whereas exploitation is characterized as examining promising areas around a potential solution. Both of these occurrences aid the algorithm in preventing the dreaded "local minima stagnation problem" (from being exploited) and yield more optimal answers with improved convergence and diversity (from exploration). The harmony between these two phenomena is also noteworthy. State-of-the-art algorithms include those that satisfy these three criteria. The research suggests that MFO, a relatively new technique, has drawbacks related to handling difficulties with vast numbers of parameters, poor solution accuracy, a slow convergence rate, and a dearth of possible solutions, and a tendency to slip into local optimal. Also, it's possible that the algorithm won't benefit from just enhancing the exploration and not the exploitation. In order to states these problems, a new MFO algorithm that incorporates quadratic interpolation into the original MFO algorithm has been devised and put to the test on a set of some of the benchmark functions. In the following parts, we will go into greater depth about the proposed method.

Effect of best flame

An author of MFO claims that moths constantly adjust their locations in relation to their respective flames. Each iteration follows a pattern in which flames are given to specific moths. The moths will fly from the best flame to the poorest flame in reverse order. In nature, moths do not behave like this at all. Giving each and every moth, a flame increases the exploration phase but decreases the exploitation potential. As a result, in the proposed method, each moth will adjust its position so that it faces the best flame and its corresponding flame; more precisely, each moth will move so that it faces the average of the compared to others moth best flame and its corresponding flame. This improves the convergence speed, the final outcomes demonstrate that almost of the provided function have superior mean and SD figures compared to the others. What follows is the revised equation:

$$m_i^{K+1} = \begin{cases} \delta_i \cdot e^{bt} \cdot \cos(2\pi t) + Fm_i, & i \le N.FM \\ \delta_i \cdot e^{bt} \cdot \cos(2\pi t) + 0.5[c.Fm_i + (1-c).Fm_{best}(i)], & i \ge N.FM \end{cases} \quad (9)$$

In this case, the best and corresponding flame are both controlled by the parameter "c = 0.5- 0.3* (current iteration / total iteration)," a linearly decreasing function. With respect to the no. of iterations, the value of "c" is between 0.3 and 0.2. The parameter is initially set to a high value, which indicates a strong effect from the associated flame, allowing the moth to efficiently probe the search space. Next, in subsequent phases, we reduce c's value to speed up convergence and get closer to the optimal flame.

Quadratic interpolation method (QIM)

We also used a straightforward quadratic interpolation technique to optimise the MFO algorithm's equilibrium between broad and narrow searches. MOP-MFO was found to be superior to other variant algorithms in the study's results. This interpolation technique is commonly utilised [48, 49], and [50] due to its high reference value for optimization algorithms. Eq. [10] provides the equation for the adjusted update.

$$x_d = 0.5 * \frac{[(x_d{}^{r_1})^2-(x_d{}^{r_2})^2]F(X^*)-[(x_d{}^{r_1})^2-(x_d{}^{*})^2]F(X^{r_2})-[(x_d{}^{r_2})^2-(x_d{}^{*})^2]F(X^{r_1})}{[x_d{}^{r_1}-x_d{}^{r_2}]F(X^*)+[x_d{}^{r_1}-x_d{}^{*}]F(X^{r_2})+[x_d{}^{r_2}-x_d{}^{*}]F(X^{r_1})} \quad (10)$$

Where, d= 1, 2, …., D, $X^{r_1} = (x_1{}^{r_1}, x_2{}^{r_1}, \dots, x_D{}^{r_1})$, $X^* = (x_1{}^*, x_2{}^*, \dots, x_D{}^*)$ and $X^{r_2} = (x_1{}^2, x_2{}^{r_2}, \dots, x_D{}^{r_2})$ are three different solutions with respect to fitness $F(X^{r_1})$, $F(X^*)$ and $F(X^{r_2})$ respectively. These different search solutions have used to produce new position, where both X^{r_1} and X^{r_2} are random solutions and X^* be the finest solution of the current population.

The working procedure of MOP-MFO has been given in the Algorithm 1 and details are outlines as shown:

- **1st step:** Set all parameters, including the no. of populations, the no. of iterations, and the evaluation function, at random.
- **2nd Step**: Utilizing equation 8, sort the moth matrix and flame matrix according to fitness value, then recalculate the fire count to reflect the current situation.
- **3rdstep**: Using equations 6, 7 and 9 update r, t and place of the moth w.r.t. flame.
- **4th Step**: To find the most fitness value of the new outcome solutions after updating them using quadratic interpolation equation 10. The best fitness provides the greatest value.
- **5th Step**: Continue to the second step until you obtain the best fitness value if it does not meet the stopping condition.

II.RESULT AND DISCUSSIONS

This section contains a brief introduction to benchmark functions as well as various explanations of the results gained from both unimodal and multimodal benchmark functions. Functions that make up the Benchmark are broken down in further depth in section 4.1. In section 4.2, we talk about how to conduct experiments using our proposed methodology. Subsection 4.3 compared MOP-MFO to MFO and other evolutionary algorithms. Subsections 4.4 and 4.5 present the Convergence analysis of Benchmark functions

Benchmark functions

The effectiveness of a brand-new metaheuristic algorithm needs to be verified and compared to that of established metaheuristic algorithms across a reliable collection of test functions. Therefore, Benchmark functions are crucial in ensuring the robustness, veracity, and efficacy of algorithms. These test functions are supplied in Appendix-1, and they were meticulously chosen from [33]. Thirty-six benchmark functions, half of which are unimodal and half of which are multimodal, have been chosen to test our proposed MOP-MFO algorithm.

For each given unimodal function, there is only one local minimum that may be identified as the global minimum. is included in the set of chosen unimodal functions (F1 through F15). The stochastic optimization algorithm's exploitability is verified with the help of unimodal functions. The best meta-heuristic algorithms optimize these functions by taking full use of them.

Many local minimum values are connected with the selected multimodal functions (F16–F36), making them more difficult to solve than unimodal functions due to the fact that their solutions get stuck at local optima and can't be avoided. The no. of local optima values and the size of the search space both increase the difficulty of multimodal functions. These operations put the exploratory prowess of metaheuristic algorithms to the test by probing their propensity to discover previously undiscovered regions.

Experimental setup

The proposed algorithm's code is developed in MATLAB R2015a and put into practice using a PC running Windows 2010 with an Intel i5 processor and 8 GB of RAM. Our suggested algorithm is used as a base to stop after no more than 10,000 iterations. The algorithm can be stopped in a variety of ways, including the largest number of successful repetitions, a predetermined margin of error, the largest CPU time use, the largest no. of iterations with 0 improvement, etc. For each of the function, 30 runs were performed, and the results were rounded up to two decimal places in order to reduce statistical mistakes and provide results that were statistically significant.

We collate the MOP-MFO's average (AV_{rg}) and standard Deviation (S_{dev}) using additional techniques. To meet this requirement, one specific combination of elements was employed for MOP-MFO in the copies of the benchmark unimodal and multimodal functions. The size of the population is fifty, and then the powers exponent constants b and t range from 1 to 1. (50).

Results from Experiments on Benchmark Functions

In this section, we provide the simulation results obtained using our proposed MOP-MFO and compare them to the results obtained using six existing metaheuristics (DE, MFO, SOS, PSO, JAYA, and WOA) on thirty-six benchmark functions (both unimodal and multimodal).

Analysis of unimodal functions:

In Table 1, we can see the AV_{rg} and S_{dev} for the six algorithms (including MOP-MFO) and the optimized unimodal functions. The table clearly shows that MOP-MFO provided the smallest numbers when compared to the other methods. For the functions F1 to F14, the MOP-MFO algorithm yields the most optimal solutions. It produces mediocre results for functions F4, F8, F12, and F15, and second rate best outcomes for F6 and F10. The best outcomes are highlighted in bold. Because of this, it is safe to assume that our method is a more effective algorithm than the alternatives.

Discussion on Multimodal functions:

Multimodal function optimization research for functions F16 – F36 is shown in Table 2. Clear evidence exists that MOP-MFO outperforms competing algorithms on the following problems: F16 - F20, F26, F27, from F29-F34, and F35. MOP-MFO supplies the second rate best outcomes for the func. F23, F28, and F36, but it falls short when compared to the best algorithms for the other five functions. Therefore, it can be concluded that MOP-MFO is the best algorithm for optimizing multimodal functions out of the seven considered.

Algorithm1: Pseudocode of the MOP-MFO algorithm.

Table 1: Performance of MOP-MFO with other considered algorithms.

Input: Maximum iteration ($Maximum_{iter}$), Objective function, Initial moth number (N),
Flame number ($N.FM$), b, and other related parameters are determined;
for $i = 1:N$
for $j = 1:D$
Generate N organism solutions $X_{i,j}$ ($i = 1, 2, ..., N$) randomly;
Find fitness;
end
end
While stopping criteria not met **do**
if Iteration = 1
$N.FM = N$;
else
Use Eqn. (8);
end if
FM = Objective Function $f(X)$;
if Iteration = 1
Place the moths in order of FM;
Revisit the Flames;
Iteration = 0;
else
Sort the moths according to FM from the previous iteration;
Revisit the Flames;
end if
Reduce the convergence constant;
for $j = 1:N$
for $k = 1:n$
Update a_1 ,t moths' location in relation to their specific flame using Eqn. (6, 7 and 9);
end for
end for
Apply quadratic interpolation and find new solution X^* using Eqn. (10) and update according to the best value;
Current iteration = Current iteration+1
End while
Output: The best solution in the ecosystem with the lowest fitness function value;

Function		MOP-MFO	MFO	SOS	PSO	JAYA	DE	WOA
F1	AV_{rg}	0	0	6.03E−173	1.76E −51	1.39E+04	2.61E−165	0
	S_{dev}	0	0	0	1.12E −50	3.54E+03	0	0
F2	AV_{rg}	0	0	0	3.10E −53	1.86E+12	1.40E−172	0
	S_{dev}	0	0	0	2.10E −52	2.08E+11	0	0
F3	AV_{rg}	0	0	0	1.31E −31	1.47E+04	1.30E −01	0
	S_{dev}	0	0	0	6.65E −31	2.91E+03	4.85E −01	0
F4	AV_{rg}	0	2.89E+001	**−1**	2.57E +01	2.83E+07	2.35E +01	2.42E +01
	S_{dev}	0	6.44E-002	**0**	3.73E +01	1.20E+07	4.98	0
F5	AV_{rg}	0	0	2.27E−74	1.20E+03	6.54E+04	1.14E−175	0
	S_{dev}	0	0	2.41E−74	5.44E+02	1.57E+04	0	0
F6	AV_{rg}	0	0	1.24E−86	1.05E−01	6.56E+01	1.02	1.10
	S_{dev}	0	0	1.26E−86	6.53E−02	6.69	4.59E−01	0
F7	AV_{rg}	0	0	−4.19E+02	7.28E+01	4.58E+02	8.15E−91	0
	S_{dev}	0	0	0	2.27E+01	8.32E+02	1.23E−90	0
F8	AV_{rg}	6.81e-01	1.04	3	0	6.22	0	6.44E-08
	S_{dev}	7.64E-01	1.78	1.94E−15	0	6.18	0	0
F9	AV_{rg}	0	0	7.51E−13	0	2.31E-01	0	0
	S_{dev}	0	0	3.99E−12	0	1.33E-01	0	0
F10	AV_{rg}	0	0	4.62E−03	3.36E+03	1.02E+02	1.30E+02	5.38E-07
	S_{dev}	0	0	6.45E−03	9.52E+02	7.95E+01	1.85E+02	0
F11	AV_{rg}	**-3.65E-03**	-3.18E-003	1.26E−07	7.67E+05	1.53E-01	-4.59E+03	−3.79E-03
	S_{dev}	**6.90E-04**	8.43E-004	3.29E−07	4.01E+05	1.21E-01	2.71E+01	0
F12	AV_{rg}	9.13E-02	8.63E-002	9.98E−01	−3.79E−03	4.41E-01	−3.79E−03	3.55E−10
	S_{dev}	8.96E-02	1.02E-001	2.92E−16	0	4.12E-01	4.57E−19	0
F13	AV_{rg}	0	0	0	6.19E+01	3.57E+04	5.11E+04	0
	S_{dev}	0	0	0	3.44E+01	6.22E+03	0	0
F 14	AV_{rg}	0	0	0	8.40E+05	1.00E+06	1.00E+06	0
	S_{dev}	0	0	0	3.23E+05	8.60E+01	1.36E+01	0
F15	AV_{rg}	8.13	8.09	1.74	3.73	3.733	2.60E−03	**8.54**
	S_{dev}	3.07E-01	3.68	2.0E−01	1.01	2.18	1.04E−03	**0**

Table 2(a): Comparison of multimodal functions of MOP-MFO with other considered algorithms

F16	AV_{rg}	0	0	0	3.03E −02	3.04E+02	0	0
	S_{dev}	0	0	0	2.61E −17	2.69E+02	0	0
F17	AV_{rg}	**8.88E-16**	8.88E-016	1.00E−15	1.81E +01	1.96E+01	1.71	4.44E-15
	S_{dev}	**0**	0	6.38E−16	5.48	6.58E-01	0	0
F18	AV_{rg}	0	0	0	4.05E +01	1.26E+02	9.22E −17	0
	S_{dev}	0	0	0	4.13E +01	3.41E+01	2.04E −17	0
F19	AV_{rg}	0	0	0	1.44E +02	1.19E+02	1.24E +01	0
	S_{dev}	0	0	0	3.79E +01	1.26E+01	3.30	0
F20	AV_{rg}	0	0	1.82E−01	0	3.50E+02	0	0
	S_{dev}	0	0	1.20E−01	0	1.24E+02	0	0
F21	AV_{rg}	9.41E+01	-4.11E-001	**8.84E+01**	−1.00E +00	-7.24E-14	−1.00	−1
	S_{dev}	2.76	3.69E-001	**1.22E−03**	0.00E +00	4.03E-13	0	0
F22	AV_{rg}	-4.10E+02	-4.05E+02	7.04E−25	− 7.57E+03	-4.03E+02	−4.56E+12	− 4.18E+02
	S_{dev}	1.35E+01	3.58E+01	9.02E−25	8.88E+02	3.69E+01	3.19E+12	0
F23	AV_{rg}	**3.74E-01**	**1.53E-002**	**1.09E−117**	**1.98E−04**	**4.50E+01**	**6.27E−04**	**1.05**
	S_{dev}	**1.88**	**1.23E-002**	**3.94E−117**	**1.65E−04**	**7.40E+01**	**1.3E−03**	**0**
F24	AV_{rg}	1.30E-01	7.88E-01	4.71E-32	2.72E-01	6.11E+07	1.11E+03	9.74E-08
	S_{dev}	2.91E-01	3.44E-01	2.23E-47	4.71E-01	3.44E+07	1.84E+03	0
F25	AV_{rg}	5.48	6.10	9.98E-01	1.25	1.10E+02	9.98E-01	1.26E+01
	S_{dev}	3.53	3.62	2.92E-16	5.71E-01	1.30E+02	2.50E-16	0
F26	AV_{rg}	1.09E-03	5.80E-03	3.27E-04	6.05E-04	1.51E-01	1.37E-03	3.07E-04
	S_{dev}	9.80E-04	6.72E-03	7.60E-05	4.62E-04	1.64E-01	4.72E-04	0
F27	AV_{rg}	3.98E-01	5.13E-01	3.98E-01	3.98E-01	1.50	3.98E-01	2.78
	S_{dev}	1.53E-05	1.36E-01	3.17E-07	5.84E-09	1.05	4.44E-16	0
F28	AV_{rg}	-9.61E	-7.72	-9.63	-6.21	-6.11E-01	-8.02	-5.06
	S_{dev}	3.07E-01	1.03	1.53	2.17	2.77E-01	1.88	0
F29	AV_{rg}	-9.86	-7.59	-1.01E+01	-7.14	-8.02E-01	-9.73	-5.08
	S_{dev}	2.66E-01	1.13	1.33	2.85	4.01E-01	5.68E-01	0
F30	AV_{rg}	-9.92	-7.78	-1.05E+01	-6.87	-8.99E-01	-1.03E+01	-5.12
	S_{dev}	2.39E-01	1.20	2.83E-15	2.57	2.57E-01	3.12E-01	0
F31	AV_{rg}	0	0	0	6.93E+02	7.32E+03	7.50E+03	0
	S_{dev}	0	0	0	1.97E+02	1.204E+02	0	0
F32	AV_{rg}	9.51	2.19E+01	0	3.24E+04	1.95E+06	2.30E+06	2.52E-10
	S_{dev}	7.31	1.19E+01	0	2.91E+04	8.82E+04	2.14E+01	0
F33	AV_{rg}	-1.59E+03	-1.37E+03	-2.35E+03	-6.29E+02	1.22E+03	7.45E+03	-2.98
	S_{dev}	7.90E+01	1.14E+02	9.59E-13	1.17E+02	1.41E+02	2.21E+01	0
F34	AV_{rg}	0	0	0	1.57E+02	6.49E+02	2.56E+03	-8.08E+02
	S_{dev}	0	0	0	1.86E+01	3.48E+01	9.83E-02	0
F35	AV_{rg}	0	0	0	1.06E+01	1.9719E+01	3.34E+01	0
	S_{dev}	0	0	0	7.62E-01	4.1799E-01	2.98E-01	0
F36	AV_{rg}	0	0	9.98E-02	2.81E+01	4.2035E+01	5.25E+01	9.98E-02
	S_{dev}	0	0	1.99E-13	1.35	5.2080E-01	4.53E-01	0

Table 3 displays the number of times MOP-mean MFO's performance was better, equal to, or worse than the other six methods. Using the data in Table 3, we can observe that MOP-MFO outperforms MFO, SOS, PSO, DE, BOA, JAYA, DE, and WOA on 16 out of 33 benchmark functions, achieves equivalent results on 15 times, and achieves poorer values on 1 occasion, 6, 6, 5, 8, and 3. Appendix-1 displays the mathematical formulation of the thirty-six (thirty-six) reference functions together with their dimensions, variable ranges, and optimal values.

Table 3: Simulation outcomes of MOP-MFO compared with other algorithms

Proposed MOP-MFO algorithm	MFO	SOS	PSO	DE	WOA	JAYA
Better (>)	16	20	28	28	15	33
Equal (=)	19	10	2	3	13	0
Worst (<)	1	6	6	5	8	3

Convergence Analysis

Several algorithms, including PSO, DE, MFO, SOS, BOA, and JAYA, and their respective convergence graphs for a small set of benchmark functions are provided in Fig. 1 for the purpose of comparison. Both the given value function evaluation and on the other hand objective function value are displayed in these graphs, on separate horizontal and vertical axes. MOP-MFO clearly has faster convergence than the alternatives.

Our suggested MOP-MFO technique is then applied to two more engineering issues, both of which are solved in the subsequent section.

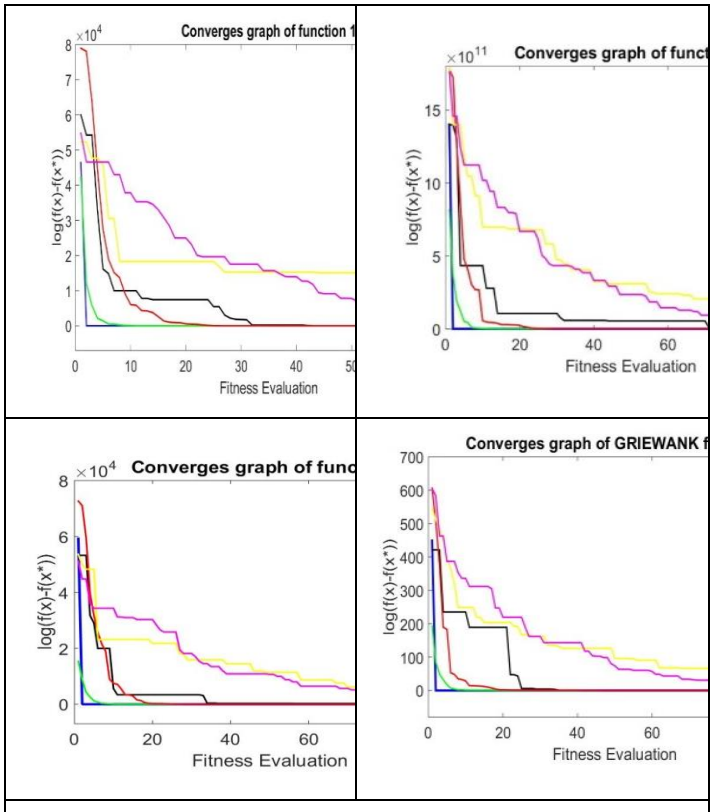

Fig. 1: Convergence graph

Description of real-life problems solved:

To demonstrate the efficacy of MFO, Specifically, it has been implemented and used to address two actual world issues (RWPs): the optimum gas output power problem and the gear train problem. The mathematical expression of the aforementioned issues is presented in Appendix-2.

The Gas Industry's Optimal Production Capacity Problem

To further complicate matters, this issue is adapted from [34]. The efficiency of MOP-MFO has been evaluated by applying it to the problem of creating gas. As shown in Table 6, we compared DE, GSA, and a hybrid DE-GSA to experimental results for this problem. Table 4 shows, in bold type, that our suggested method is more efficient than competing techniques.

Table 4: Comparison performance of MOP-MFO of the gas capacity problem

Item	DE-GSA	GSA	DE	MFO	m-MFO
x_1	17.5	17.5	17.5	17.5	17.5
x_2	600	600	600	600	600
f(x)	169.844	169.844	169.844	71.4495	**71.448**

The problem of Gear Train Design

Mechanical engineers use it to find the best possible ratio of gears for a train's four gears [35]. It's unconstrained and uses four variables. As limitations, we use the variables' ranges. Fig. 2 depicts the schematics of the aforementioned issue.

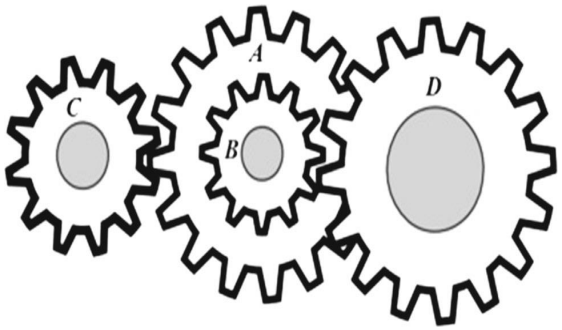

Fig. 2: gear train design problem.

Table 5 displays the results of a comparison between MOP-MFO, and eight different metaheuristic algorithms derived from [36], demonstrating the effectiveness of MOP-MFO in solving the aforementioned problem. According to Table 5, the results obtained by the MOP-MFO algorithm are superior to those obtained by the other algorithms.

ALGORITHM	x_1	x_2	x_3	x_4	$f(\vec{x})$
MOP-MFO	47	19	17	43	2.7548E-12
MFO	51.34512	21.32743	14.98321	47.43271	1.7110e-04
BA	57.4517	19.48176	18.58749	43.68715	1.5310E-11
PSO	50.30931	21.01347	14.72391	47.52131	3.0310e-04
DA	52.44017	17.00267	22.99426	51.67159	3.0210e-11
FPA	51.15054	22.45769	17.97921	55.90958	4.8310e-11
SA	51.34511	21.32742	14.98320	47.43271	1.7110e-03
GA	52.57125	23.19834	16.93254	48.24512	1.1310e-04
ACO	51.41824	21.35921	15.83209	47.34128	2.8710e-05
WWO	55.27473	24.3159	15.03104	45.82914	5.6710e-12

Table 5: Comparison results of MOP-MFO on problem of gear design.

III.CONCLUSION

In light of what has been discussed above, we may say that m-capacity MFO's for intensification is extraordinary, as seen by the outcomes of optimization of unimodal functions. Approximately 90% of the benchmark functions return results that are acceptable. MOP-MFO's high-class performance when compared to other variant algorithms is indicative of its excellent diversification capabilities. MOP-MFO is more adept at balancing global and local searches.

MOP-MFO can be modified to address multi-objective optimization, combinatorial optimization, and constrained optimization issues; so, it has vast potential for further development and improvement. QIWOA can also be used to address more intricate issues in the actual world.

Appendix-1

Table 1(a): Formalization of unimodal functions in mathematics

S/N	Func.	Mathematical formula	d	f_{min}	Search space
F1	Sphere	$f(x) = \sum_{j=1}^{d} x_j^2$	30	0	[-100, 100]
F2	Cigar	$f(x) = 10^6 \sum_{j=1}^{d} x_j^2$	30	0	[-100, 100]
F3	Step	$f(x) = \sum_{j=1}^{d} (x_j + 0.5)^2$	30	0	[-100, 100]
F4	Rosenbrock	$f(x) = \sum_{j=1}^{d} [100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2]$	30	0	[-2.048,2.048]
F5	Schwefel 1.2	$f(x) = \sum_{j=1}^{d} \sum_{k=1}^{j} x_j^2$	30	0	[-100, 100]
F6	Schwefel 2.21	$f(x) = \max\{\lvert x_i \rvert, 1 \le j \le d\}$	30	0	[-100, 100]
F7	Schwefel 2.22	$f(x) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	30	0	[-10, 10]
F8	Booth	$f(x) = (2x_1 + x_2 - 5)^2 + (x_1 + 2x_2 - 7)^2$	2	0	[-10, 10]
F9	Matyas	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	0	[-10, 10]
F10	Powell	$f(x) = \sum_{j=1}^{d/4} (x_{4j-3} + 10x_{4j-2})^2 + 5(x_{4j-3} + 10x_{4j-2})^2 + (x_{4j-2} + 2x_{4j-1})^4 + 10(x_{4j-3} - x_{4j})^4$	32	0	[-4, 5]
F11	Zettl	$f(x) = (x - 1^2 + x - 2^2 - 2x_1)^2 + 0.25x_1$	2	-0.00379	[-1, 5]
F12	Leon	$f(x) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$	2	0	[-1.2, 1.2]
F13	Zakhrov	$f(x) = \sum_{j=1}^{d} x_i^2 + \left(0.5\sum_{j=1}^{d} jx_j\right)^2 + \left(0.5\sum_{j=1}^{d} jx_j\right)^4$	2	0	[-5, 10]
F14	Tablet	$f(x) = 10^6 \times x_1^2 + \sum_{j=1}^{d} x_j^6$	30	0	[-1, 1]
F15	Quartic	$f(x) = \sum_{i=1}^{d} jx_j^4 + random(0,1)$	30	0	[-1.28, 1.28]

S/N	FUNC.	Mathematical formula	D	f_{min}	SEARCH SPACE
F16	Bohachevsky	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	0	[-100, 100]
F17	Ackley	$f(x) = 20 + e - 20e^{\frac{1}{5}\sqrt{\left(\frac{1}{d}\sum_{j=1}^{d} x_j^2\right)}} - e^{\frac{1}{d}\sum\cos(2\pi x_j)}$	30	0	[-32.768, 32.768]
F18	Griewank	$f(x) = \sum_{j=1}^{d} \frac{x_j^2}{4000} - \prod_{j=1}^{d} \cos\left(\frac{x_j}{\sqrt{j}}\right) - 1$	30	0	[-600, 600]
F19	Rastrigin	$f(x) = 10D + \sum_{j=1}^{d} [x_j^2 - 10\cos(2\pi x_j)]$	30	0	[-5.12, 5.12]
F20	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]
F21	Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp[-(x_1 - \pi)^2 \times 9(x_2 - \pi)^2]$	2	0	[-100, 100]
F22	Schwefel 2.26	$f(x) = -\sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	-418.982	[-500, 500]
F23	Powersum	$f(x) = \sum_{i=1}^{d} \left[\left(\sum_{k=1}^{d} (x_k^i) - b_i\right)^2\right]$	4	0	[0, 4]
F24	Penalized1.1	$f(x) = \frac{\pi}{d}\Big\{10\sin^2(\pi y_i) + \sum_{j=1}^{d-1} (y_j - 1)^2[1 + 10\sin^2(3\pi y_{j+1}) + (y_d - 1)^2]\Big\} + \sum_{j=1}^{d} u(x_j, 10, 100, 4)$ Where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_j - a)^m x_j > a \\ 0 \\ k(-x_j - a)^m x_j < a \end{cases}$	30	0	[-50, 50]
F25	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]

F26	Kowalik	$f(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{(b_i^2 - b_i x_3 - x_4)}\right]^2$	4	0.0003075	[-5, 5]
F27	Branin	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}(x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1\right)$	2	0	[-5, 5]
F28	Shekel 5	$f(x) = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	-10.1499	[0, 10]
F29	Shekel-7	$f(x) = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.3999
F30	Shekel-10	$f(x) = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.5319
F31	Bohachevsy 3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.3$	2	[-50, 50]	0
F32	Colville	$f(x) = 100(x_1 - x_2^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1((x_2 - 1)^2) + (x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$	4	[-10, 10]	0
F33	Himmelblau	$f(x) = 1/30\left[\sum_{j=1}^{d} x_j^4 - 16x_j^2 + 5x_j\right]$	30	[-5, 5]	-78.33236
F34	Csendes	$f(x) = \sum_{j=1}^{d} x_j^6\left(2 + \sin 1/x_j\right)$	30	[-1, 1]	0
F35	Inverted cosine mixture	$f(x) = 0.1 \times 30 - \left[0.1 \times \sum_{i=1}^{d} 5\pi x_j - \sum_{i=1}^{d} x_j^2\right]$	30	[-1, 1]	0
F36	Salomon	$f(x) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{d} x_j^2}\right) + 0.1\sqrt{\sum_{i=1}^{d} x_j^2}$	30	[-100, 100]	0

Table- 1(b): Formalization of multimodal functions in mathematics

Apendix-2

The Gas Industry's Optimal Production Capacity Problem:

$$\text{Min } f(x) = 61.8 + 5.72 \times x_1 \times 0.2623 \times \left[(40 - x_1) \times \ln\frac{x_2}{200}\right]^{-0.85} + 0.087 \times (40 - x_1) \times \ln\frac{x_2}{200} + 700.23 \times x_2^{-0.75}$$

s.t. $x_1 \geq 17.5, x_2 \geq 200,\ 17.5 \leq x_1 \leq 40, 300 \leq x_2 \leq 600;$

The problem of Gear Train Design:

Minimize $f(\vec{x}) = \left[\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right]^2$,

Subjected to $12 \leq x_1, x_2, x_3, x_4 \leq 60,$

Where $\vec{x} = [x_1 x_2 x_3 x_4] = [n\alpha\ n\beta\ n\gamma\ n\delta\ ].$

IV.REFERENCES:

[1] McCarthy JF. (1989). Block-conjugate-gradient method. Physical Review D. 40:2149.

[2] Wu, G. (2016). Across neighborhood search for numerical optimization. Information Sciences, 329: 597-618.

[3] Wu, G., Pedrycz, W., Suganthan, P. N., &Mallipeddi, R. (2015). A variable reduction strategy for evolutionary algorithms handling equality constraints. Applied Soft Computing, 37: 774-786.

[4] Holand JH (1992). Genetic algorithms. Sci Am, 267:66–72.

[5] Yi Y, He R (2014). A novel artificial bee colony algorithm. In: 2014 sixth international conference on intelligent human–machine systems and cybernetics, 1: 271-274.

[6] Kennedy J, Eberhart R (1995). Particle swarm optimization. In: Proceedings of ICNN'95—international conference on neural networks, 4:1942–1948.

[7] Storn R, Price K (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim, 11:341–359.

[8] Yang XS (2009). Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. Springer 169–178.

[9] Gandomi AH, Yang X-S, Alavi AH (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29:17–35.

[10] Cheng MY, Prayogo D (2014). Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct. 139:98–112.

[11] Wang GG, Deb S, Cui Z (2015) Monarch butterfly optimization. Neural Comput Appl31: 1995-2014.

[12] Kirkpatrick, S., Gelatt, C. D., &Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

[13] *Rashedi E, Nejamabadi-pour H, Saryajdi S (2009) GSA: A gravitation search algorithm. Information sciences 179:2232-2248.*

[14] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95:51-67.

[15] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69:46-61.

[16] Mirjalili S (2015). Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowled Based Syst 89:228–249.

[17] Muangkote N, Sunat K, Chiewchanwattana S (2016). Multilevel thresholding for satellite image segmentation with moth-_ame based optimization. In Proc. 13th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE), KhonKaen, Thailand, 1-6.

[18] S. Babar S, Mohammad I, Mohammad A, Zahoor R (2016). Novel application of moth flame optimization algorithm for solving economic load dispatch problems with emission and valve point loading effect. In: 4th International Conference on Energy, Environment and Sustainable Development.

[19] K, Singh U, Salgotra R (2018). An enhanced moth flame optimization. Neural Computing and Applications 26:1-35.

[20] Sarma, A., Bhutani, A., & Goel, L. (2017). Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality. 2017 Intelligent Systems Conference (IntelliSys), 52–60.

[21] Sarma, A., Bhutani, A., & Goel, L. (2017). Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality. 2017 Intelligent Systems Conference (IntelliSys), 52–60.

[22] 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.

[23] Pant, M., Radha, T., & Singh, V. P. (2007). A new particle swarm optimization with quadratic interpolation. In International Conference on Computational Intelligence

and Multimedia Applications (ICCIMA 2007) IEEE, 1:55-60.

[24] Jamil M, Yang XS (2013) A literature survey of benchmark functions for global optimization problems. Int. Journal of Mathematical Modelling and Numerical Optimisation 4: 150-194.

[25] Li X, Yin M, Ma Z (2011) Hybrid differential evolution and gravitation search algorithm for unconstrained optimization. International journal of the physical sciences 6:5961-5981.

[26] Gandomi AH. (2014) Interior search algorithm (ISA) a novel approach for global optimization. ISA transactions53:1168-83

[27] Połap D (2017). Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism. Symmetry 9:203.