ESKİŞEHİR TECHNICAL UNIVERSITY JOURNAL OF SCIENCE AND TECHNOLOGY
A- APPLIED SCIENCES AND ENGINEERING

2024, 25(1), pp. 78-98, DOI: 10.18038/estubtda.1348497

RESEARCH ARTICLE

PERFORMANCE EVALUATIONS OF THE MANTA RAY FORAGING OPTIMIZATION ALGORITHM IN REAL-WORLD CONSTRAINED OPTIMIZATION PROBLEMS

Gülnur YILDIZDAN [1, *]

[1] Kulu Vocational School, Selcuk University, Konya, Turkey

ABSTRACT

Metaheuristic algorithms are often preferred for solving constrained engineering design optimization problems. The most important reason for choosing these algorithms is that they guarantee a satisfactory response within a reasonable time. The swarm intelligence-based manta ray foraging optimization algorithm (MRFO) is a metaheuristic algorithm proposed to solve engineering applications. In this study, the performance of MRFO is evaluated on 19 mechanical engineering optimization problems in the CEC2020 real-world constrained optimization problem suite. In order to increase the MRFO performance, three modifications are made to the algorithm; in this way, the enhanced manta ray foraging optimization (EMRFO) algorithm is proposed. The effects of the modifications made are analyzed and interpreted separately. Its performance has been compared with the algorithms in the literature, and it has been shown that EMRFO is a successful and preferable algorithm for this problem suite.

Keywords: Constrained optimization problems, Manta ray foraging optimization algorithm, Real-world optimization problems

1. INTRODUCTION

Exact and metaheuristic methods can be used to solve constrained engineering design optimization problems. However, exact methods are not preferred because of local optima, computational complexity, and time requirements. Metaheuristic algorithms generally use random number search techniques to obtain a satisfactory answer within a reasonable period of time [1, 2]. In recent years, meta-heuristic algorithms have become increasingly popular for tackling challenging optimization problems in all engineering fields due to their cheap, efficient, and easy implementation. Figure 1 categorizes meta-heuristics into five groups based on their natural inspirations. These categories include evolution-based (Genetic algorithm [3] (GA), differential evolution [4] (DE), and biogeography-based optimization [5] (BBO), etc.), swarm-based, physics/chemistry-based (Gravitational search algorithm [6] (GSA), Artificial chemical reaction optimization algorithm [7] (ACROA), atom search optimization [8] (ASO), etc.), human-based algorithms (teaching-learning-based optimization [9] (TLBO), poor and rich optimization [10] (PRO), society and civilization algorithm [11] (SCA), etc.), and others (supply-demand-based optimization [12] (SDO), backtracking optimization search algorithm [13] (BSA), differential search algorithm [14] (DSA), etc.) [15].

*Corresponding Author: gavsar@selcuk.edu.tr
Received: 23.08.2023 Published: 28.03.2024

Figure 1. Meta-heuristic method classification[15]

Swarm-based algorithms, one of these classification categories, are algorithms inspired by the herd consciousness of living things in nature. The most well-known of these are algorithms such as the particle swarm optimization algorithm (PSO) [16], the artificial bee colony algorithm (ABC) [17], and the ant colony algorithm (ACO) [18], and their numbers are increasing rapidly day by day. One of the recently proposed swarm intelligence-based algorithms is manta ray foraging optimization (MRFO) [19]. This algorithm models three different strategies that manta rays use in foraging. Compared to other traditional optimization algorithms, it has a clearer mechanism, does not require additional parameter tuning, has a better balance between exploration and exploitation search ability, and has improved solution performance [20]. These features are the reason for the preference for MRFO. In this study, the performance of MRFO, which has been used in different optimization problems, on CEC2020 real-world constrained engineering problems has been evaluated. An enhanced algorithm (EMRFO) has been proposed by making three modifications to increase the performance of the algorithm. The performance increase provided by EMRFO is presented and discussed. The rest of the study is organized as follows: The literature review for MRFO is presented in the second section. The third section explains the steps of the MRFO algorithm and introduces the proposed EMRFO algorithm. The experimental study results are showcased in the fourth section. The fifth section covers the conclusions drawn from the study and suggests future work.

2. LITERATURE REVIEW

MRFO is a preferred algorithm in many different research fields. Houssein et al. developed a novel feature selection and electrocardiogram arrhythmia classification approach based on MRFO and support vector machines [21]. Houssein et al. used their proposed improved MRFO using opposition-based learning to solve the image segmentation problem in COVID-19 computed tomography images [22]. Hemeida et al. implemented the MRFO to reduce power loss by determining the optimal size and placement of distributed generators within the radial distribution network [23]. Tang et al. developed MRFO using adaptive control parameter strategies, an elite search pool, and a distribution estimation strategy to overcome the shortcomings of MRFO. They tested it on different test suites and three engineering design problems [24]. Gokulkumari developed a method for classifying brain tumors using the MRFO-based deep convolutional neural network algorithm [25]. Hassan et al. proposed the MRFO algorithm integrated with the gradient-based optimizer to solve the single and multi-purpose economic emission distribution problem [26]. Micev et al. proposed a new method based on the hybrid use of MRFO and a simulated annealing algorithm to solve the problem of setting the proportional-integral-derivative controller type for an automatic voltage regulator system [27]. Kahraman et al. used the MRFO algorithm, which they developed with the crowd-distance-based Pareto archiving strategy, to solve the CEC 2020 benchmarking functions and the multi-objective optimal power flow problem [28]. Got et al. proposed MRFO, which they developed with external archive and grid mechanisms, for multi-

purpose problems [29]. Elaziz et al. developed the algorithm by integrating the triangle mutation operator and orthogonal learning strategies into MRFO and tested it on CEC functions and engineering problems [30]. Zouache and Abdelaziz extended MRFO so that it can be applied to multi-objective problems [31]. The algorithm was created by integrating strategies such as population archive, crowding distance, and ϵ-dominance, which contribute to diversity and convergence. The proposed algorithm was applied to structural design problems such as four-bar truss design,speed-reduced design, welded beam design, and disk brake design, and compared with the literature. Ekinci et al. proposed the MRFO algorithm with improved diversification and intensification features [32]. The MRFO algorithm was developed with a generalized opposition-based learning technique and the Nelder-Mead simplex search method. The proposed algorithm was tested for solving unimodal and multimodal benchmark functions. It was also used to find the optimum values of a real PID plus a second-order derivative controller used in the magnetic object suspension system. Yousri et al. proposed an improved MRFO that adopts the Caputo fractional differ-sum operator to increase the utilization of past optimal solutions in MRFO and adaptively uses the somersault factor to avoid premature convergence [33]. The proposed algorithm was used for global optimization problems, engineering design optimization problems, and multi-threshold segmentation. Daqaq et al. presented an improved MRFO algorithm based on an elitist non-dominated sorting strategy to solve multi-objective optimization problems [34]. The algorithm's performance was tested on multimodal optimization problems, four engineering optimization problems, the CEC2020 test suite, and the modified real-world issue of IEEE 30-bus optimal power flow involving the wind/solar/small-hydro power generations. Liu et al. proposed an effective MRFO algorithm by integrating a nonlinear adjustment parameter based on the cosine factor, random individuals' information interaction, and fractional derivative mutation strategy into the standard algorithm [20]. The proposed algorithm was used to solve CEC2017 benchmark functions and seven engineering design optimization problems. Zhu et al. proposed a new MRFO based on variable spiral factors, matching games, and progressive learning that enables the dynamic adjustment of internal parameters [35]. The performance of the algorithm was validated on classical benchmark functions, the CEC2022 test suite, and three engineering optimization problems. Yang et al. proposed a new elite chaotic MRFO in which the population is initialized chaotic and integrated with an opposition-based learning strategy [36]. This algorithm was tested on the classical benchmark function, CEC2020 test functions, and three engineering optimization problems. Ghosh et al. obtained a binary version of the MRFO by applying the transfer functions, which are S-shaped and V-shaped. The resulting binary algorithm was applied to eighteen feature selection problems [37]. Yıldızdan proposed MRFO's binary version with the help of transfer functions and tested the binary algorithm on classical and CEC2005 benchmark functions [38]. Wang et al. discretized the MRFO algorithm with the sigmoid function, improved it with the XOR operator and velocity adjustment factor, and applied it to the spectrum allocation problem [39].

When the literature review was examined, it was seen that there were many improved or hybrid versions of MRFO proposed for many different continuous or discrete problems. The results obtained in the literature are also quite successful. The motivation for this study is that MRFO has become a frequently preferred algorithm recently, and there has been no significant study on its performance on CEC2020 real-world engineering problems. The contributions of the study to the literature are as follows:

- To preserve diversity and ensure the balance between foraging processes, three modifications are made to MRFO, and a new manta ray foraging optimizer (EMRFO) is proposed.
- The effectiveness of EMRFO was demonstrated on 19 mechanical engineering optimization problems in the CEC2020 real-world constrained optimization problem suite.
- Although the performance of MRFO has been examined on a few well-known engineering optimization problems, thanks to this study, its performance on a larger number of engineering

problems has been examined. A guiding resource has been created for researchers who are considering working in this field.

- The results obtained within the scope of this study revealed that EMRFO was more successful compared to the literature. Thus, a new version of MRFO that is promising for different problems was introduced to the literature.

3. MATERIAL AND METHOD

3.1. Manta Ray Foraging Optimization Algorithm (MRFO)

MRFO is a bio-inspired optimization algorithm proposed by Zhao et al. [19] in 2020, inspired by the feeding behavior of manta rays. The algorithm is based on three foraging techniques: chain, spiral, and somersault. The steps of the algorithm are as follows:

3.1.1. Initialization

The MRFO algorithm, similar to other meta-heuristic algorithms, begins by creating a random population using Equation 1.

$$X_i^d = Lb_i^d + rand \times \left(Ub_i^d - Lb_i^d\right) \qquad i = 1, \ldots., N \qquad d = 1, \ldots., D \tag{1}$$

In the equation, D is the number of dimensions and N is the population size (the number of individuals in the population). Lb and Ub are the lower and upper limits of the dimensions.

3.1.2. Chain foraging

Manta rays that feed on plankton can sense the position of food. Manta rays tend to gravitate toward places with high nutrient density. For this, they align and form a foraging chain. Except for the first manta ray at the beginning of the chain, they search according to the food source and the individual in front of it. That is, each individual is updated after each iteration using both the solution before it and the best solution so far. Chain foraging behavior in MRFO is formulated as in Equation 2.

$$X_{i,d}^{t+1} = \begin{cases} X_{i,d}^t + r \times \left(X_{best,d}^t - X_{i,d}^t\right) + \alpha \times \left(X_{best,d}^t - X_{i,d}^t\right) & i = 1 \\ X_{i,d}^t + r \times \left(X_{i-1,\mathrm{d}}^t - X_{i,d}^t\right) + \alpha \times \left(X_{best,d}^t - X_{i,d}^t\right) & i = 2, \ldots., N \end{cases} \tag{2}$$

$$\alpha = 2 \times r \times \sqrt{|\log(r)|} \tag{3}$$

In Equation 2, r is a vector of random numbers in the range [0,1]. α is the weight coefficient whose formula is given in Equation 3. $X_{i,d}^t$ is the position of the ith individual at time t of the dth dimension, and $X_{best,d}^t$ is the the position with the highest nutrient density.

3.1.3. Cyclone foraging

When manta rays see a piece of plankton in deep water, they both spiral toward the plankton piece and swim toward the manta rays in front of them. This behavior, called cyclone foraging, is formulated in the MRFO as given in Equation 4. β given in Equation 5 is the weight coefficient, T is the maximum number of iterations, and r is also a random number between 0 and 1.

$$X_{i,d}^{t+1} = \begin{cases} X_{best,d} + r \times \left(X_{best,d}^t - X_{i,d}^t\right) + \beta \times \left(X_{best,d}^t - X_{i,d}^t\right) & i = 1 \\ X_{best,d} + r \times \left(X_{i-1,\mathrm{d}}^t - X_{i,d}^t\right) + \beta \times \left(X_{best,d}^t - X_{i,d}^t\right) & i = 2, \ldots., N \end{cases} \tag{4}$$

$$\beta = 2e^{r\frac{T-t+1}{T}} \times \sin(2\pi r) \tag{5}$$

MRFO is an algorithm with comprehensive global search capability. To achieve this, a random position is created in the optimization process, and a spiral search is performed around this position. In the algorithm, this behavior is formulated as in Equations 6 and 7. In Equation 6, $X_{r,d}^{t}$ denotes a random location in the search space. Ub^{d} and Lb^{d} are also the upper and lower limit values for the dth dimension, respectively.

$$X_{r,d}^{t} = Lb^{d} + r \times (Ub^{d} - Lb^{d}) \tag{6}$$

$$X_{i,d}^{t+1} = \begin{cases} X_{r,d}^{t} + r \times \left(X_{r,d}^{t} - X_{i,d}^{t}\right) + \beta \times \left(X_{r,d}^{t} - X_{i,d}^{t}\right) & i = 1 \\ X_{r,d}^{t} + r \times \left(X_{i-1,\mathrm{d}}^{t} - X_{i,d}^{t}\right) + \beta \times \left(X_{r,d}^{t} - X_{i,d}^{t}\right) & i = 2, \ldots., N \end{cases} \tag{7}$$

3.1.4. Somersault foraging

Each manta ray swims back and forth around a food location, seen as a pivot, and somersaults to a new position. Thus, individuals update their location according to the best location found so far. In MRFO, this behavior is formulated according to Equation 8. In Equation 8, $r1$ and $r2$ are two random numbers between 0 and 1. The value of S is the somersault factor, which determines the distance covered during a somersault. The pseudocode and flowchart of MRFO are given in Figure 2 and Figure 3, respectively.

$$X_{i,d}^{t+1} = X_{i,d}^{t} + S \times \left(r1 \times X_{best,d}^{t} - r2 \times X_{i,d}^{t}\right), \qquad i = 1, \ldots, N \tag{8}$$

```
Determine T_max, N, Ub, Lb, t = 1
Initialize population according to Equation 1.
Compute the fitness of each individual f(X_i)
Obtain best individual (X_best)
 While stop criterion is not satisfied  do
    For  i=1 to N
      If  rand < 0.5 then          // Cyclone foraging
         If   t / Tmax < rand    then
            Generate a random individual(X_r(t)) according to Equation 6.
            Create a new candidate individual (X_i(t + 1)) according to Equation 7.
         Else
            Create a new candidate individual (X_i(t + 1)) according to Equation 4.
         End If
       Else          //Chain foraging
          Create a new candidate individual (X_i(t + 1)) according to Equation 2.
      End If
      Compute the fitness of each individual f(X_i(t + 1))
       If  f(X_i(t + 1))< f(X_best) then
         f(X_best) = f(X_i(t + 1))
       End If
      //Somersault foraging
       Create a new candidate individual (X_i(t + 1))according to Equation 8.
       Compute the fitness of the individual f(X_i(t + 1))
        If  f(X_i(t + 1))< f(X_best) then
           f(X_best) = f(X_i(t + 1))
        End If
     End For
  End While
```

Figure 2. The pseudocode of MRFO

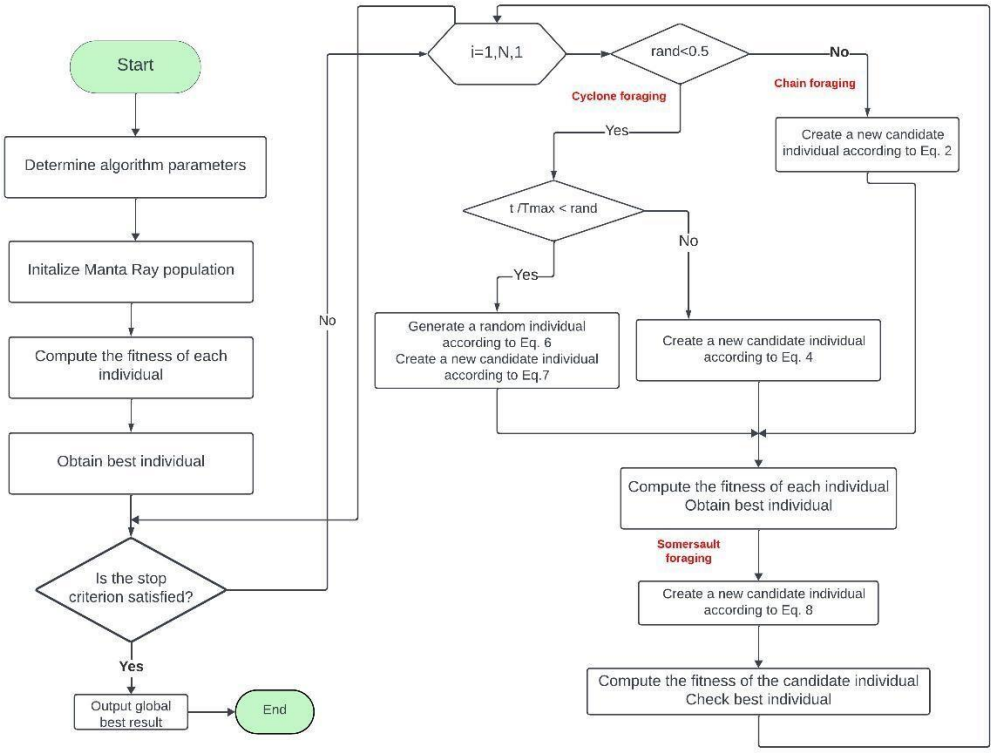

Figure 3. The flowchart of MRFO

3.2. Enhanced Manta Ray Foraging Optimization Algorithm (EMRFO)

MRFO is a swarm intelligence-based metaheuristic algorithm proposed for engineering applications. In general, MRFO, which has a good performance, has some drawbacks when analyzed in detail. These can be listed as follows: especially in the early iterations, the selection of random reference points weakens the exploitation ability, the chain foraging tends to bring the solutions to the local optimum, and the population diversity decreases in the late iterations [24]. This study proposes an enhanced algorithm called EMRFO to address the drawbacks of MRFO. Three modifications have been made to the MRFO to achieve this goal.

- As can be seen from Figure 2, the balance between exploration and exploitation in MRFO is determined by whether the result (C) of the linearly increasing iteration/maximum iteration (i.e., $t/Tmax$) operation is less than a random number. Here, choosing the C parameter with nonlinearly increasing characteristics is beneficial for the search process to be more balanced. Thus, one of the disadvantages of MRFOA, which is the weakening of the ability to benefit due to random reference location selection in early iterations, is prevented. For this reason, different studies have been carried out on the nonlinear use of this parameter in the literature [24, 40]. This study proposes the use of the simulated annealing inertia weight strategy [41] for parameter C. The mathematical formulation of this strategy is given in Equation 9. In Equation 9, t represents the number of iterations, and w_{start} and w_{end} represent the start and end values for the parameter, respectively.

$$C_t = w_{start} + (w_{end} - w_{start}) \times 0.95^{t-1} \quad (9)$$

Figure 4 shows the change of the C parameter in MRFO and EMRFO. When the graphic is examined, when the nonlinear C parameter is used as in EMRFO, foraging behavior in random

locations is dominant in the first 20% of the total iteration. In the remaining part, searches are made around locations with high nutritional value. Thus, the disadvantage of weakening the ability of MRFO to provide benefits due to random reference location selection in early iterations is eliminated.

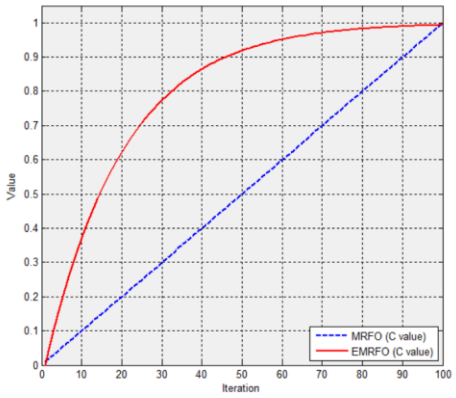

Figure 4. Change of C values

- The second modification was made to the S parameter used in somersault foraging (Equation 8). The value of S is the somersault factor that determines the distance covered during a somersault around the food location chosen as the pivot. In MRFO, the S parameter is a fixed value ($S = 2$). There are different studies in the literature on the selection of the S value [24, 42]. In this study, the S value is used by decreasing it within a certain interval. To determine these interval values, tests are performed for different values. At the end of this testing process, the interval [5, 2] is chosen, which yields the most successful results. In this way, exploration is supported by using a larger somersault distance initially. Over time, decreasing the S value supports exploitation by conducting a more focused search with smaller somersault distances. Thus, by changing the distance during the somersault foraging, diversity is contributed and premature convergence is prevented. In order to achieve this, the S value is used by decreasing in a certain interval nonlinearly according to Equation 10, which is obtained by editing Equation 9.

$$S_t = w_{end} + (w_{start} - w_{end}) \times 0.95^{t-1} \tag{10}$$

Figure 5 shows the change of the S parameter in MRFO and EMRFO. The graphic shows that the non-linear change of the S parameter in EMRFO supports the ability to exploration first, then exploitation, using different somersault distances.

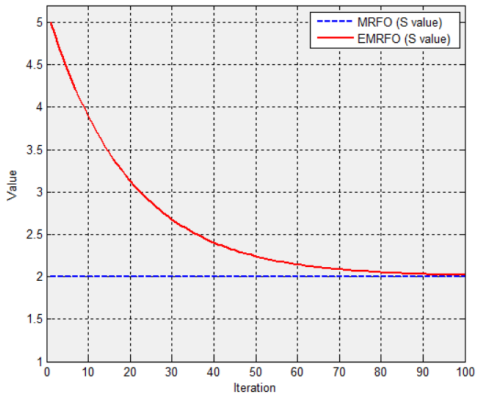

Figure 5. Change of S values

- Finally, an update strategy was integrated into somersault foraging to prevent the drawback of MRFO losing diversity towards the last iterations and to contribute to avoiding local minima by preserving diversity for longer periods of time. For this, an update rate (UR) is first determined. Then, as shown in Figure 6, each individual in the population and the candidate individual obtained after applying the algorithm steps to that individual are exchanged at the determined UR rate [43].

UR=0.3	**Selected dimensions={1, 5, 9}**									
ith individual	0.21	0.45	0.11	0.54	0.43	0.7	0.98	0.23	0.87	0.12
candiadate individual for ith individual	0.27	0.22	0.65	0.57	0.32	0.8	0.73	0.13	0.91	0.44
new candidate individual after the update strategy applied	**0.21**	0.22	0.65	0.57	**0.43**	0.8	0.73	0.13	**0.87**	0.44

Figure 6. Updating strategy

The pseudocode of the EMRFO algorithm obtained after the three modifications mentioned above in the MRFO algorithm is given in Figure 7. The flowchart of EMRFO is also presented in Figure 8.

```
Determine T_max, N, Ub, Lb, t = 1
Initialize population according to Equation 1.
Compute the fitness of each individual f(X_i)
Obtain best individual (X_best)
 While stop criterion is not satisfied  do
    For  i=1 to N
      If  rand < 0.5 then          // Cyclone foraging
        Calculate C_t value according to Equation 9.
         If  C_t < rand    then
            Generate a random individual(X_r(t)) according to Equation 6.
            Create a new candidate individual (X_i(t + 1)) according to Equation 7.
         Else
            Create a new candidate individual (X_i(t + 1)) according to Equation 4.
         End If
      Else          //Chain foraging
          Create a new candidate individual (X_i(t + 1)) according to Equation 2.
      End If
      Compute the fitness of each individual f(X_i(t + 1))
       If  f(X_i(t + 1)) < f(X_best) then
         f(X_best) = f(X_i(t + 1))
       End If
      //Somersault foraging
       Calculate S_t value according to Equation 10.
       Create a new candidate individual (X_i(t + 1))according to Equation 8.
       Apply updating strategy.
       Compute the fitness of the individual f(X_i(t + 1))
        If  f(X_i(t + 1)) < f(X_best) then
           f(X_best) = f(X_i(t + 1))
         End If
    End For
 End While
```

Figure 7. The pseudocode of EMRFO

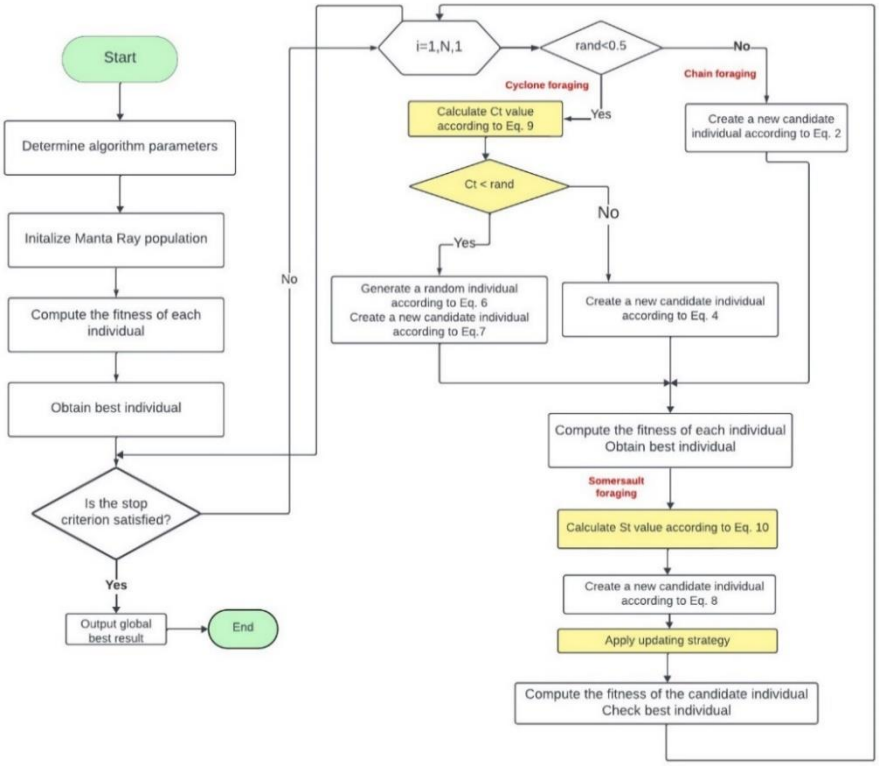

Figure 8. The flowchart of EMRFO

3.3. Nonparametric Friedman Test

In this study, the non-parametric Friedman test is used to reveal significant differences between metaheuristic algorithms. The steps of the test can be summarized as follows [44]:
The hypotheses are defined first.

Hypothesis 0 (HO): The results of the compared algorithms do not differ statistically significantly.
Hypothesis 1 (H1): The results of the compared algorithms differ statistically significantly.

- During R_N runs, gather evaluation criteria for every metaheuristic algorithm (the run number is represented by R_N).
- Sort the metaheuristic algorithms that have been tested in order of best to worst, from 1 to k, which is denoted as rij.
- Calculate the mean of the obtained ranks over R_N runs for the jth algorithm according to Equation 11.

$$R_j = \frac{1}{R\_N}\sum_i^j r_{ij} \tag{11}$$

- Using Equation 12, write the nonparametric Friedman statistic Ff.

$$Ff = \frac{12n}{(k+1)k}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{12}$$

To calculate the probability of rejecting the null hypothesis in this test, a p-value is used. The null hypothesis should be rejected if the p-value is less than 0.05. In other words, there is a significant difference in the results of the compared algorithms [45].

4. EXPERIMENTAL RESULTS

Solving real-world optimization problems can be challenging due to their complex objective functions and numerous constraints. To overcome these challenges, various metaheuristics and constraint-handling approaches have been proposed. In this section, the performance of the proposed EMRFO algorithm is evaluated on CEC2020 real-world constrained optimization problems [46]. The real-world constrained optimization problems are formulated as in Equation 13 [46].

$$\begin{aligned} Minimize: f(\bar{x}), \bar{x} = (x_1, x_2, \dots, x_D) \\ Subject\ to: \ g_i(\bar{x}) \le 0, \quad i = 1,2, \dots, p \\ h_j(\bar{x}) = 0, \quad j = p+1, \dots, m \end{aligned} \tag{13}$$

where $f(\bar{x})$ is the objective function, $\bar{x}$ is the D-dimensional solution vector, the $g_i(\bar{x})$ function is the ith inequality constraint, and $h_j(\bar{x})$ is the jth equality constraint. These constraints may or may not be linear. The CEC2020 real-world constrained optimization problem suite consists of 57 problems in total and six groups: industrial chemical processes, process synthesis and design problems, mechanical engineering problems, power system problems, power electronics problems, and livestock feed ration optimization. In this study, the performance of EMRFO is tested on mechanical engineering problems (RC15-RC33). This group consists of 19 problems, and their features are given in Table 1. In the table, D is the dimension of the problem, g is the number of inequality constraints, h is the number of equality constraints, and $f(\bar{x}^*)$ is the best-known feasible objective function value.

As it is known, in optimization, a fixed amount of function evaluations called maximum function evaluation numbers ($MaxFEs$) are allocated, and when the algorithm reaches this number, the optimization process is stopped. The MaxFEs for the problems in Table 1 are determined by Equation 14. The population size is also chosen as 100. In comparisons, the best (best fitness value), mean (mean of fitness values), and standard deviation (standard deviation of fitness values) values obtained from 25 independent studies are used. Also, in comparison tables, bold font shows a better value, italic font shows equal values, and gray background shows a better standard deviation value for equal mean values.

$$\text{MaxFEs} = \begin{cases} 1 \times 10^5 & if\ D \le 10 \\ 2 \times 10^5 & if\ 10 \le D \le 30 \end{cases} \tag{14}$$

In this section, firstly, the effect of the modifications made in $MRFO$ on performance is analyzed. Let $MRFO_C$ be the improved algorithm obtained by adding the first modification that suggests the use of the parameter C, which exhibits nonlinearly increasing characteristics in $MRFO$, to the algorithm. Accordingly, Table 2 presents the comparison results of $MRFO$ and $MRFO_C$. When the results are analyzed, it is seen that $MRFO_C$ achieved a better mean in 11 out of 19 problems. In two of the remaining problems, $MRFO$ finds a better mean value, while in the other five, the algorithms find the same mean value. But in these five problems, $MRFO_C$ has a smaller standard deviation value. In the RC31 problem, both algorithms find $0.00E+00$. These results show that the first modification positively affects the MRFO's performance.

Table 1. Features of the mechanical engineering problems

Problem	Name	D	g	h	$f(\bar{x}^*)$
RC15	Weight Minimization of a Speed Reducer	7	11	0	2,9944244658E+03
RC16	Optimal Design of Industrial refrigeration System	14	15	0	3,2213000814E-02
RC17	Tension/compression spring design (case 1)	3	3	0	1,2665232788E-02
RC18	Pressure vessel design	4	4	0	5,8853327736E+03
RC19	Welded beam design	4	5	0	1,6702177263E+00
RC20	Three-bar truss design problem	2	3	0	2,6389584338E+02
RC21	Multiple disk clutch brake design problem	5	7	0	2,3524245790E-01
RC22	Planetary gear train design optimization problem	9	10	1	5,2576870748E-01
RC23	Step-cone pulley problem	5	8	3	1,6069868725E+01
RC24	Robot gripper problem	7	7	0	2,5287918415E+00
RC25	Hydro-static thrust bearing design problem	4	7	0	1,6254428092E+03
RC26	Four-stage gear box problem	22	86	0	3,5359231973E+01
RC27	10-bar truss design	10	3	0	5,2445076066E+02
RC28	Rolling element bearing	10	9	0	1,4614135715E+04
RC29	Gas Transmission Compressor Design (GTCD)	4	1	0	2,9648954173E+06
RC30	Tension/compression spring design (case 2)	3	8	0	2,6138840583E+00
RC31	Gear train design Problem	4	1	1	0,0000000000E+00
RC32	Himmelblau's Function	5	6	0	-3,0665538672E+04
RC33	Topology Optimization	30	30	0	2,6393464970E+00

Secondly, the effect of the second modification made in $MRFO$ is analyzed. Let $MRFO_S$ be the improved algorithm obtained by adding the second modification that suggests the use of the parameter S, which exhibits nonlinearly decreasing characteristics in $MRFO$, to the algorithm. Accordingly, Table 3 presents the comparison results of $MRFO$ and $MRFO_S$. When the results are analyzed, it is found that $MRFO_S$ finds a better mean value in 7 problems, while $MRFO$ finds a better mean value in 5 problems. In six of the remaining problems, the algorithms found the same mean value. In two of these problems, $MRFO$, and in the remaining four, the $MRFO_S$ algorithm obtained a better standard deviation value. In the RC31 problem, both algorithms find $0.00E+00$. These results show that the second modification improves the MRFO's performance.

Table 4 shows the comparison results of $MRFO_{CS}$, which is obtained by integrating the first and second modifications into the standard algorithm, and the $MRFO$ algorithm. When the results in the table are analyzed, $MRFO_{CS}$ finds the best mean value in 12 out of 19 problems, while $MRFO$ finds the best mean value in 1 problem. In five of the remaining problems, the algorithms find the same mean value, but in these problems, $MRFO_{CS}$ performs better by finding better standard deviation values. In the RC31 problem, both algorithms find $0.00E+00$. These results show that integrating both modifications into $MRFO$ further improves MRFO's performance. To see this effect more clearly, the results of the $MRFO_C$, $MRFO_S$, and $MRFO_{CS}$ algorithms are compared in Table 5. When the table is examined, the $MRFO_C$ algorithm finds a better mean value in 3 of the problems, while the $MRFO_S$ algorithm finds a better mean value in 2 of the problems. The $MRFO_{CS}$ algorithm achieves a better mean value in 8 of the remaining problems. Of the five problems where the algorithms find the same mean, $MRFO_C$ obtains a better standard deviation value in three of them and $MRFO_{CS}$ in the remaining two. Additionally, in the RC31 problem, it finds the value $0.00E+00$ in three algorithms. According to the results of this comparison, it is proven that the use of the two modifications together contributes more to the performance.

Table 2. Comparison results of the $MRFO$ and $MRFO_C$ algorithms

	MRFO			***$MRFO_C$***		
	Best	Mean	Std	Best	Mean	Std
RC15	2,994424E+03	2,994425E+03	8,933243E-05	2,994424E+03	**2,994424E+03**	4,781112E-07
RC16	3,254210E-02	4,194690E-02	6,952061E-03	3,221305E-02	**4,038194E-02**	8,051717E-03
RC17	1,266675E-02	1,267320E-02	5,055811E-06	1,266576E-02	**1,266873E-02**	2,743722E-06
RC18	6,059714E+03	6,324653E+03	2,404949E+02	6,059714E+03	**6,127095E+03**	1,267901E+02
RC19	1,670218E+00	*1,670218E+00*	1,234202E-12	1,670218E+00	*1,670218E+00*	2,703950E-15
RC20	2,638958E+02	*2,638959E+02*	8,129818E-06	2,638958E+02	*2,638959E+02*	6,355131E-06
RC21	2,352425E-01	*2,352425E-01*	3,271315E-12	2,352425E-01	*2,352425E-01*	5,870076E-14
RC22	5,257687E-01	5,287543E-01	3,035527E-03	5,257687E-01	**5,278581E-01**	2,496295E-03
RC23	1,714135E+01	2,619215E+01	1,118297E+01	1,624213E+01	**2,307006E+01**	6,292782E+00
RC24	2,697086E+00	2,954188E+00	2,301767E-01	2,614242E+00	**2,839908E+00**	1,684963E-01
RC25	1,640789E+03	1,766357E+03	9,996301E+01	1,623517E+03	**1,708607E+03**	6,220753E+01
RC26	3,736598E+01	**8,869061E+15**	1,611897E+16	3,646089E+01	1,288893E+16	4,760657E+16
RC27	5,245301E+02	**5,275083E+02**	3,028573E+00	5,245136E+02	5,279593E+02	3,156006E+00
RC28	1,695820E+04	*1,695820E+04*	2,184191E-03	1,695820E+04	*1,695820E+04*	9,737234E-05
RC29	2,964896E+06	2,964902E+06	4,628019E+00	2,964895E+06	**2,964899E+06**	2,284177E+00
RC30	2,658559E+00	2,671626E+00	4,955796E-02	2,658559E+00	**2,660319E+00**	8,184043E-03
RC31	0,000000E+00	*0,000000E+00*	0,000000E+00	0,000000E+00	*0,000000E+00*	0,000000E+00
RC32	-3,066554E+04	*-3,066554E+04*	2,027150E-03	-3,066554E+04	*-3,066554E+04*	2,448015E-04
RC33	2,639352E+00	2,639400E+00	8,820392E-05	2,639347E+00	**2,639348E+00**	5,596236E-06

Table 3. Comparison results of the $MRFO$ and $MRFO_S$ algorithms

	MRFO			***$MRFO_S$***		
	Best	Mean	Std	Best	Mean	Std
RC15	2,994424E+03	*2,994425E+03*	8,933243E-05	2,994424E+03	*2,994425E+03*	4,928646E-05
RC16	3,254210E-02	**4,194690E-02**	6,952061E-03	3,249612E-02	4,267801E-02	7,982063E-03
RC17	1,266675E-02	**1,267320E-02**	5,055811E-06	1,266574E-02	1,268067E-02	1,033643E-05
RC18	6,059714E+03	6,324653E+03	2,404949E+02	6,059720E+03	**6,244510E+03**	2,431724E+02
RC19	1,670218E+00	*1,670218E+00*	1,234202E-12	1,670218E+00	*1,670218E+00*	9,856824E-13
RC20	2,638958E+02	2,638959E+02	8,129818E-06	2,638958E+02	**2,638958E+02**	7,175023E-06
RC21	2,352425E-01	*2,352425E-01*	3,271315E-12	2,352425E-01	*2,352425E-01*	7,432136E-12
RC22	5,257687E-01	5,287543E-01	3,035527E-03	5,257687E-01	**5,283521E-01**	2,244001E-03
RC23	1,714135E+01	**2,619215E+01**	1,118297E+01	1,632368E+01	4,634495E+01	6,675387E+01
RC24	2,697086E+00	2,954188E+00	2,301767E-01	2,705441E+00	**2,915821E+00**	2,119451E-01
RC25	1,640789E+03	1,766357E+03	9,996301E+01	1,647601E+03	**1,747498E+03**	8,257355E+01
RC26	3,736598E+01	8,869061E+15	1,611897E+16	3,647122E+01	**4,404589E+14**	1,081002E+15
RC27	5,245301E+02	**5,275083E+02**	3,028573E+00	5,245299E+02	5,297212E+02	2,717610E+00
RC28	1,695820E+04	*1,695820E+04*	2,184191E-03	1,695820E+04	*1,695820E+04*	5,237155E-04
RC29	2,964896E+06	*2,964902E+06*	4,628019E+00	2,964896E+06	*2,964902E+06*	5,143730E+00
RC30	2,658559E+00	2,671626E+00	4,955796E-02	2,658559E+00	**2,662654E+00**	1,294429E-02
RC31	0,000000E+00	*0,000000E+00*	0,000000E+00	0,000000E+00	*0,000000E+00*	0,000000E+00
RC32	-3,066554E+04	*-3,066554E+04*	2,027150E-03	-3,066554E+04	*-3,066554E+04*	1,383139E-03
RC33	2,639352E+00	**2,639400E+00**	8,820392E-05	2,639356E+00	2,639488E+00	3,840580E-04

Table 4. Comparison results of the $MRFO$ and $MRFO_{CS}$ algorithms

	MRFO			***$MRFO_{CS}$***		
	Best	Mean	Std	Best	Mean	Std
RC15	2,994424E+03	2,994425E+03	8,933243E-05	2,994424E+03	**2,994424E+03**	5,672839E-07
RC16	3,254210E-02	4,194690E-02	6,952061E-03	3,222404E-02	**3,859848E-02**	6,765000E-03
RC17	1,266675E-02	1,267320E-02	5,055811E-06	1,266555E-02	**1,266943E-02**	4,402280E-06
RC18	6,059714E+03	6,324653E+03	2,404949E+02	6,059714E+03	**6,110158E+03**	1,064905E+02
RC19	1,670218E+00	*1,670218E+00*	1,234202E-12	1,670218E+00	*1,670218E+00*	7,921320E-15
RC20	2,638958E+02	*2,638959E+02*	8,129818E-06	2,638958E+02	*2,638959E+02*	6,802541E-06
RC21	2,352425E-01	*2,352425E-01*	3,271315E-12	2,352425E-01	*2,352425E-01*	1,070512E-13
RC22	5,257687E-01	5,287543E-01	3,035527E-03	5,259674E-01	**5,276695E-01**	1,210525E-03
RC23	1,714135E+01	2,619215E+01	1,118297E+01	1,698765E+01	**1,854255E+01**	1,514401E+00
RC24	2,697086E+00	2,954188E+00	2,301767E-01	2,757983E+00	**2,897183E+00**	1,471321E-01
RC25	1,640789E+03	1,766357E+03	9,996301E+01	1,621391E+03	**1,706032E+03**	9,037795E+01
RC26	3,736598E+01	8,869061E+15	1,611897E+16	3,625376E+01	**2,349845E+15**	6,911838E+15
RC27	5,245301E+02	**5,275083E+02**	3,028573E+00	5,244839E+02	5,284012E+02	3,188696E+00
RC28	1,695820E+04	*1,695820E+04*	2,184191E-03	1,695820E+04	1,695820E+04	4,729283E-05
RC29	2,964896E+06	2,964902E+06	4,628019E+00	2,964896E+06	**2,964899E+06**	3,753715E+00
RC30	2,658559E+00	2,671626E+00	4,955796E-02	2,658559E+00	**2,658587E+00**	8,867289E-05
RC31	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
RC32	-3,066554E+04	*-3,066554E+04*	2,027150E-03	-3,066554E+04	*-3,066554E+04*	1,895641E-04
RC33	2,639352E+00	2,639400E+00	8,820392E-05	2,639347E+00	**2,639347E+00**	6,932692E-07

Table 5. Comparison results of the $MRFO_C$, $MRFO_S$ and $MRFO_{CS}$ algorithms

	$MRFO_c$		***$MRFO_S$***		***$MRFO_{CS}$***	
	Mean	Std	Mean	Std	Mean	Std
RC15	2,994424E+03	4,781112E-07	2,994425E+03	4,928646E-05	**2,994424E+03**	5,672839E-07
RC16	4,038194E-02	8,051717E-03	4,267801E-02	7,982063E-03	**3,859848E-02**	6,765000E-03
RC17	**1,266873E-02**	2,743722E-06	1,268067E-02	1,033643E-05	1,266943E-02	4,402280E-06
RC18	6,127095E+03	1,267901E+02	6,244510E+03	2,431724E+02	**6,110158E+03**	1,064905E+02
RC19	1,*670218E+00*	2,703950E-15	*1,670218E+00*	9,856824E-13	*1,670218E+00*	7,921320E-15
RC20	2,638959E+02	6,355131E-06	**2,638958E+02**	7,175023E-06	2,638959E+02	6,802541E-06
RC21	*2,352425E-01*	5,870076E-14	*2,352425E-01*	7,432136E-12	*2,352425E-01*	1,070512E-13
RC22	5,278581E-01	2,496295E-03	5,283521E-01	2,244001E-03	**5,276695E-01**	1,210525E-03
RC23	2,307006E+01	6,292782E+00	4,634495E+01	6,675387E+01	**1,854255E+01**	1,514401E+00
RC24	**2,839908E+00**	1,684963E-01	2,915821E+00	2,119451E-01	2,897183E+00	1,471321E-01
RC25	1,708607E+03	6,220753E+01	1,747498E+03	8,257355E+01	**1,706032E+03**	9,037795E+01
RC26	1,288893E+16	4,760657E+16	**4,404589E+14**	1,081002E+15	2,349845E+15	6,911838E+15
RC27	**5,279593E+02**	3,156006E+00	5,297212E+02	2,717610E+00	5,284012E+02	3,188696E+00
RC28	*1,695820E+04*	9,737234E-05	*1,695820E+04*	5,237155E-04	*1,695820E+04*	4,729283E-05
RC29	*2,964899E+06*	2,284177E+00	2,964902E+06	5,143730E+00	*2,964899E+06*	3,753715E+00
RC30	2,660319E+00	8,184043E-03	2,662654E+00	1,294429E-02	**2,658587E+00**	8,867289E-05
RC31	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
RC32	*-3,066554E+04*	2,448015E-04	*-3,066554E+04*	1,383139E-03	*-3,066554E+04*	1,895641E-04
RC33	2,639348E+00	5,596236E-06	2,639488E+00	3,840580E-04	**2,639347E+00**	6,932692E-07

Thirdly, the performance impact of the update strategy integrated into the somersault phase of the algorithm in order to preserve diversity for longer is analyzed. The optimal value for the UR value, indicating the update rate, is also determined. The results are given in Table 6 in comparison according to the mean value. The UR values are selected from a range between 0.1 and 0.5, with increments of 0.1. When the UR value is taken as 0.5, the best mean value is found in 4 problems. In two of the problems where the algorithms obtained the same mean value, it found the best standard deviation for $UR = 0.5$. According to these results, although 0.5 seems to be the most appropriate value, the nonparametric Friedman test [47] is applied for a more detailed evaluation. As mentioned before, in this study, the Friedman test is used to determine whether there is a significant difference between the algorithms, and the mean rank values obtained are given in Figure 9. The problems we study are minimization problems, and therefore, when evaluating Friedman test results, the algorithm with a smaller mean rank is considered to perform better. Accordingly, when Figure 9 is analyzed, it is seen that the algorithm with a UR value of 0.2 ranks first with a mean rank value of 2.89. Therefore, the UR value is chosen as 0.2 in the proposed EMRFO algorithm. Moreover, all EMRFO sub-versions with the update strategy obtained better mean rank values than the standard MRFO algorithm and ranked higher in the ranking. Figure 10 shows the convergence graphics of MRFO and EMRFO for four selected problems. Graphics are drawn according to the best values obtained by the algorithms. The convergence ability has been examined by selecting problems in which the algorithms find close values. In the RC15 problem, the algorithms find the same value. In the RC15 graphic, it can be seen that although EMRFO starts from a solution that is farther from the optimal, it converges faster than MRFO. The algorithms find the same value in the RC19 problem. According to the graphic, EMRFO starts to search from a solution closer to the optimal, but the two algorithms exhibit a similar convergence speed. Similar to RC19, in RC28, although the EMRFO starts searching from a better solution, EMRFO and MRFO converge at the same speed. When the RC33 graphic is examined, it is seen that EMRFO converges to a value closer to the optimal faster than MRFO.

Table 6. Comparison results of the $EMRFO$ with different UR values

	UR	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
RC15	Mean	*2,994424E+03*	*2,994424E+03*	*2,994424E+03*	*2,994424E+03*	*2,994424E+03*
	Std	4,071313E-07	6,178244E-07	9,357967E-07	5,032104E-07	3,901230E-07
RC16	Mean	3,932953E-02	3,503066E-02	3,731769E-02	3,483386E-02	**3,462827E-02**
	Std	6,280985E-03	2,998453E-03	5,382151E-03	2,028029E-03	2,311009E-03
RC17	Mean	1,267058E-02	1,267062E-02	1,267238E-02	1,267158E-02	**1,266951E-02**
	Std	6,530173E-06	2,743085E-06	4,531782E-06	8,685000E-06	3,154004E-06
RC18	Mean	6,113323E+03	6,097922E+03	6,107582E+03	6,107154E+03	**6,065886E+03**
	Std	1,052608E+02	1,101071E+02	1,073265E+02	1,075247E+02	1,300511E+01
RC19	Mean	*1,670218E+00*	*1,670218E+00*	*1,670218E+00*	*1,670218E+00*	*1,670218E+00*
	Std	6,938080E-15	1,280029E-14	4,246029E-15	2,607383E-15	8,639472E-15
RC20	Mean	*2,638959E+02*	*2,638959E+02*	*2,638959E+02*	*2,638959E+02*	*2,638959E+02*
	Std	9,255367E-06	1,117208E-05	1,285150E-05	7,508691E-06	8,537156E-06
RC21	Mean	*2,352425E-01*	*2,352425E-01*	*2,352425E-01*	*2,352425E-01*	*2,352425E-01*
	Std	1,835236E-14	3,668458E-14	7,047372E-14	1,163729E-13	8,960553E-14
RC22	Mean	5,282837E-01	**5,265442E-01**	5,267592E-01	5,275840E-01	5,265863E-01
	Std	2,952660E-03	1,341955E-03	7,461250E-04	1,364557E-03	1,070154E-03
RC23	Mean	1,476827E+02	4,552499E+01	**1,667919E+01**	5,648618E+02	1,736887E+01
	Std	1,181553E+02	5,342818E+01	2,640131E-01	1,732807E+03	1,947398E+00
RC24	Mean	2,846105E+00	**2,841463E+00**	2,920189E+00	2,981002E+00	2,997555E+00
	Std	1,193459E-01	1,071138E-01	1,745038E-01	1,618122E-01	1,040719E-01
RC25	Mean	1,725546E+03	1,722699E+03	1,747194E+03	**1,697705E+03**	1,713279E+03
	Std	9,147076E+01	6,927179E+01	5,385686E+01	3,039111E+01	5,802935E+01
RC26	Mean	2,923076E+15	5,425705E+01	4,819498E+01	1,034750E+14	**4,557533E+01**
	Std	9,243577E+15	1,514599E+01	1,538641E+01	3,272166E+14	7,277022E+00
RC27	Mean	5,271598E+02	5,271363E+02	**5,271166E+02**	5,271591E+02	5,289394E+02
	Std	3,188783E+00	3,091667E+00	3,129993E+00	3,066334E+00	2,810040E+00
RC28	Mean	*1,695820E+04*	*1,695820E+04*	*1,695820E+04*	*1,695820E+04*	*1,695820E+04*
	Std	1,008211E-04	6,555526E-05	1,165918E-04	7,306640E-05	1,162785E-04
RC29	Mean	*2,964898E+06*	*2,964898E+06*	*2,964898E+06*	2,964899E+06	2,964900E+06
	Std	1,500307E+00	1,415458E+00	2,514889E+00	2,798226E+00	2,903451E+00
RC30	Mean	*2,658559E+00*	*2,658559E+00*	2,658565E+00	*2,658559E+00*	*2,658559E+00*
	Std	1,311500E-09	9,747018E-13	1,750382E-05	1,670440E-10	0,000000E+00
RC31	Mean	**0,000000E+00**	4,435135E-14	1,687493E-13	3,234578E-14	2,772378E-13
	Std	0,000000E+00	6,419820E-14	2,304124E-13	4,757450E-14	7,406385E-13
RC32	Mean	*-3,066554E+04*	*-3,066554E+04*	*-3,066554E+04*	*-3,066554E+04*	*-3,066554E+04*
	Std	1,451593E-04	3,579815E-04	8,693618E-05	7,873023E-04	9,772617E-05
RC33	Mean	*2,639347E+00*	2,639348E+00	*2,639347E+00*	*2,639347E+00*	2,639349E+00
	Std	9,732917E-08	3,714109E-06	3,301452E-07	3,685461E-07	6,480390E-06

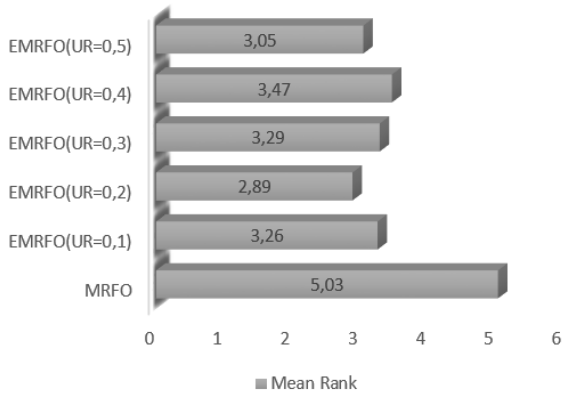

Figure 9. Friedman test results

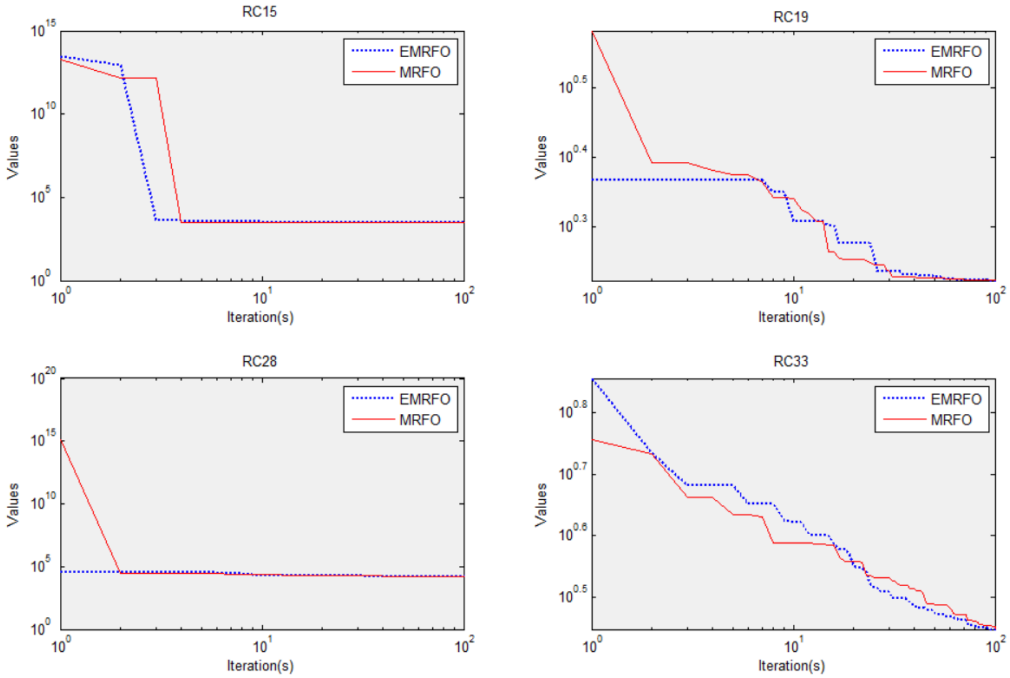

Figure 10. Convergence graphics of MRFO and EMRFO

Finally, the proposed EMRFO algorithm is compared with the algorithms in the literature, and the obtained results are given in Table 7. In this table, MVPA (Most Valuable Player Algorithm) results are taken from [48], ABC (Artificial Bee Colony Algorithm), SSA (Salp Swarm Algorithm), and GWO (Grey Wolf Optimizer) results are taken from [49]. When Table 7 is analyzed, it is observed that EMRFO finds the best mean value alone in 5 problems, and in 7 problems, it finds the same best mean value with some other algorithms. According to the pairwise comparison results with other algorithms given at the end of Table 7, the proposed algorithm obtains better mean values in more problems than the other algorithms except for ABC. In addition, Friedman test results based on the mean values given in Table 7 are presented in Figure 11. According to the ranking result, EMRFO ranks first with a mean rank value of 3.11. In addition, the ABC algorithm ranks second, while the MRFO and GWO algorithms rank third. In light of all the evaluations given in this section, it can be concluded that the modifications made in MRFO contribute to the performance of the algorithm, and the proposed EMRFO is an algorithm that produces preferable successful results compared to the literature.

Table 7. Comparison results of the EMRFO with other algorithms in the literature

Problems	**EMRFO**	**MRFO**	**MVPA**	**ABC**	**SSA**	**GWO**
RC15	*2,994E+03*	*2,994E+03*	*2,994E+03*	*2,994E+03*	3,002E+03	2,998E+03
RC16	**3,503E-02**	4,195E-02	4,936E-02	6,360E−02	4,916E+00	3,878E−02
RC17	**1,267E-02**	**1,267E-02**	1,277E-02	1,284E−02	1,272E−02	**1,268E−02**
RC18	6,098E+03	6,325E+03	6,371E+03	**5,774E+03**	6,108E+03	5,906E+03
RC19	*1,670E+00*	*1,670E+00*	*1,670E+00*	1,889E+00	1,696E+00	1,688E+00
RC20	*2,639E+02*	*2,639E+02*	*2,639E+02*	*2,639E+02*	*2,639E+02*	*2,639E+02*
RC21	*2,352E-01*	*2,352E-01*	*2,352E-01*	*2,352E−01*	*2,352E−01*	2,353E−01
RC22	**5,265E-01**	5,288E-01	7,484E-01	5,34E+00	5,358E−01	5,34E+00
RC23	4,552E+01	2,619E+01	1,628E+01	1,605E+01	**1,604E+01**	1,605E+01
RC24	2,841E+00	2,954E+00	**2,810E+00**	4,071E+00	2,945E+00	3,907E+00
RC25	1,723E+03	1,766E+03	2,284E+03	6,573E+02	4,346E+02	**3,665E+02**
RC26	**5,426E+01**	8,869E+15	1,218E+02	1,293E+06	3,500E+06	1,631E+07
RC27	5,271E+02	5,275E+02	5,281E+02	**5,231E+02**	5,247E+02	5,241E+02
RC28	1,696E+04	1,696E+04	1,696E+04	**1,695E+04**	1,698E+04	1,700E+04
RC29	*2,965E+06*	*2,965E+06*	*2,965E+06*	3,005E+06	3,059E+06	2,967E+06
RC30	**2,659E+00**	2,672E+00	2,903E+00	2,767E+00	2,675E+00	2,702E+00
RC31	4,435E-14	*0,000E+00*	*0,000E+00*	4,377E−15	9,281E−22	2,443E−14
RC32	*-3,067E+04*	*-3,067E+04*	*-3,067E+04*	−3,044E+04	*−3,067E+04*	−3,066E+04
RC33	*2,639E+00*	*2,639E+00*	*2,639E+00*	*2,639E+00*	2,713E+00	2,790E+00
Better		8	8	7	11	10
Worst		2	3	9	7	8
Equal		9	8	3	1	1

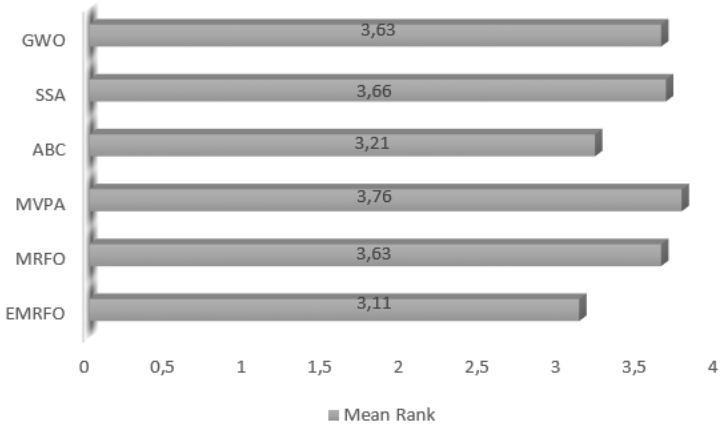

Figure 11. Friedman test results

5. CONCLUSIONS AND FUTURE WORK

In this study, the performance of the proposed MRFO algorithm for engineering applications is tested on mechanical engineering optimization problems in the CEC2020 real-world constrained optimization problems suite. In order to overcome the drawbacks of the MRFO algorithm and increase its performance, the algorithm has been developed with three modifications, and EMRFO has been

proposed. The modifications made are integrated into the algorithm separately, and their effect on performance is shown. When the results of EMRFO are analyzed, it is determined that the algorithm found more successful values and converged faster than the standard algorithm. In addition, in the comparisons made with the algorithms in the literature, EMRFO took first place and proved that it is a competitive, successful, and preferable algorithm for this problem suite. In future studies, EMRFO can be applied to different problem suites. In addition, it can be discretized for solving discrete optimization problems.

CONFLICT OF INTEREST

The author stated that there are no conflicts of interest regarding the publication of this article.

REFERENCES

[1] Ezugwu AE, Shukla AK, Nath R, Akinyelu AA, Agushaka JO, Chiroma H, Muhuri PK. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. Artificial Intelligence Review 2021; 54(6): p. 4237-4316.

[2] Pan JS, Zhang LG, Wang RB, Snášel V, Chu SC. Gannet optimization algorithm : A new metaheuristic algorithm for solving engineering optimization problems. Mathematics and Computers in Simulation 2022; 202: p. 343-373.

[3] Tang KS, Man KF, Kwong S, He Q. Genetic algorithms and their applications. IEEE signal processing magazine 1996; 13(6): p. 22-37.

[4] Price KV. Differential evolution in Handbook of optimization: From classical to modern approach. Springer, p. 187-214, 2013.

[5] Simon D. Biogeography-based optimization. IEEE transactions on evolutionary computation 2008; 12(6): p. 702-713.

[6] Rashedi E, Nezamabadi-Pour H,Saryazdi S. GSA: a gravitational search algorithm. Information sciences 2009; 179(13): p. 2232-2248.

[7] Alatas B. ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Systems with Applications 2011; 38(10): p. 13170-13180.

[8] Zhao W, Wang L, Zhang Z. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. Knowledge-Based Systems 2019; 163: p. 283-304.

[9] Rao RV,VSavsani VJ,Vakharia D. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-aided design 2011; 43(3): p. 303-315.

[10] Moosavi SHS, Bardsiri VK. Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Engineering Applications of Artificial Intelligence 2019; 86: p. 165-181.

[11] Ray T, Liew KM. Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation 2003; 7(4): p. 386-396.

[12] Zhao W, Wang L, Zhang Z. Supply-demand-based optimization: A novel economics-inspired algorithm for global optimization. IEEE Access 2019; 7: p. 73182-73206.

[13] Civicioglu P. Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and computation 2013; 219(15): p. 8121-8144.

[14] Liu J, Wu C, Wu G,Wang X. A novel differential search algorithm and applications for structure design. Applied Mathematics and Computation 2015; 268: p. 246-269.

[15] Wang L, Cao Q, Zhang Z, Mirjalili S, Zhao W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Engineering Applications of Artificial Intelligence 2022; 114: p. 105082.

[16] Kennedy J, Eberhart R. Particle swarm optimization. in Proceedings of ICNN'95-international conference on neural networks 1995: IEEE.

[17] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Applied mathematics and computation 2009: 214(1): p. 108-132.

[18] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE computational intelligence magazine 2006; 1(4): p. 28-39.

[19] Zhao W, Zhang Z, Wang L. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Engineering Applications of Artificial Intelligence 2020; 87: p. 103300.

[20] Liu J, Chen Y, Liu X, Zuo F, Zhou H. An efficient manta ray foraging optimization algorithm with individual information interaction and fractional derivative mutation for solving complex function extremum and engineering design problems. Applied Soft Computing 2024; 150: p. 111042.

[21] Houssein EH, Ibrahim IE, Neggaz N, Hassaballah M,Wazery YM. An efficient ECG arrhythmia classification method based on Manta ray foraging optimization. Expert Systems with Applications 2021; 181: p. 115131.

[22] Houssein EH, Emam MM, Ali AA. Improved manta ray foraging optimization for multi-level thresholding using COVID-19 CT images. Neural Computing and Applications 2021; 33(24): p. 16899-16919.

[23] Hemeida MG, Ibrahim AA, Mohamed AAA, Alkhalaf S, El-Dine AMB. Optimal allocation of distributed generators DG based Manta Ray Foraging Optimization algorithm (MRFO). Ain Shams Engineering Journal 2021; 12(1): p. 609-619.

[24] Tang A, Zhou H, Han T, Xie L. A Modified Manta Ray Foraging Optimization for Global Optimization Problems. IEEE Access 2021; 9: p. 128702-128721.

[25] Gokulkumari G. Classification of brain tumor using manta ray foraging optimization-based DeepCNN classifier. Multimedia Research 2020; 3(4): p. 32-42.

[26] Hassan MH, Houssein EH, Mahdy MA, Kamel S. An improved manta ray foraging optimizer for cost-effective emission dispatch problems. Engineering Applications of Artificial Intelligence 2021; 100: p. 104155.

[27] Micev M, Ćalasan M, Ali ZM, Hasanien HM, Abdel Aleem SHE. Optimal design of automatic voltage regulation controller using hybrid simulated annealing – Manta ray foraging optimization algorithm. Ain Shams Engineering Journal 2021; 12(1): p. 641-657.

[28] Kahraman HT, Akbel M, Duman S. Optimization of Optimal Power Flow Problem Using Multi-Objective Manta Ray Foraging Optimizer. Applied Soft Computing 2022; 116: p. 108334.

[29] Got A, Zouache D, Moussaoui A. MOMRFO: Multi-objective Manta ray foraging optimizer for handling engineering design problems. Knowledge-Based Systems 2022; 237: p. 107880.

[30] Elaziz MA, Abualigah L, Ewees AA, Al-qaness MAA, Mostafa RR, Yousri D, Ibrahim RA. Triangular mutation-based manta-ray foraging optimization and orthogonal learning for global optimization and engineering problems. Applied Intelligence 2023; 53(7): p. 7788-7817.

[31] Zouache D, Abdelaziz FB. Guided Manta Ray foraging optimization using epsilon dominance for multi-objective optimization in engineering design. Expert Systems with Applications 2022; 189: p. 116126.

[32] Ekinci S, Izci D, Kayri M. An Effective Controller Design Approach for Magnetic Levitation System Using Novel Improved Manta Ray Foraging Optimization. Arabian Journal for Science and Engineering 2022; 47(8): p. 9673-9694.

[33] Yousri D, AbdelAty AM, Al-qaness MAA, Ewees AA, Radwan AG, Abd Elaziz M. Discrete fractional-order Caputo method to overcome trapping in local optima: Manta Ray Foraging Optimizer as a case study. Expert Systems with Applications 2022; 192: p. 116355.

[34] Daqaq F, Kamel S, Ouassaid M, Ellaia R, Agwa AM. Non-Dominated Sorting Manta Ray Foraging Optimization for Multi-Objective Optimal Power Flow with Wind/Solar/Small- Hydro Energy Sources. Fractal and Fractional 2022; 6(4):194.

[35] Zhu D, Wang S, Zhou C,Yan S. Manta ray foraging optimization based on mechanics game and progressive learning for multiple optimization problems. Applied Soft Computing 2023; 145: p. 110561.

[36] Yang J, Liu Z, Zhang X, Hu G. Elite Chaotic Manta Ray Algorithm Integrated with Chaotic Initialization and Opposition-Based Learning. Mathematics 2022; 10(16):2960.

[37] Ghosh KK, Guha R, Bera SK, Kumar N, Sarkar R. S-shaped versus V-shaped transfer functions for binary Manta ray foraging optimization in feature selection problem. Neural Computing and Applications 2021; 33(17): p. 11027-11041.

[38] Yildizdan G. Bin_MRFOA: A Novel manta ray foraging optimization algorithm for binary optimization. Konya Journal Of Engineering Sciences 2023; 11(2): p. 449-467.

[39] Wang D, Liu X, Li Z, Lu B, Guo A, Chai G. Discrete Manta Ray Foraging Optimization Algorithm And Its Application In Spectrum Allocation. Journal Of Computer Applications 2022; 42(1): p. 215.

[40] Ozsoydan FB, Baykasoglu A. Analysing the effects of various switching probability characteristics in flower pollination algorithm for solving unconstrained function minimization problems. Neural Computing and Applications 2019; 31(11): p. 7805-7819.

[41] Bansal JC, Singh PK, Saraswat M, Verma A, Jadon SS, Abraham A. Inertia Weight strategies in Particle Swarm Optimization. in 2011 Third World Congress on Nature and Biologically Inspired Computing 2011.

[42] Jusof MFM, Nasir ANK, Razak AAA, Rizal NAM, Ahmad MA, Muhamad IH. Adaptive-Somersault MRFO for Global Optimization with an Application to Optimize PD Control. in Proceedings of the 12th National Technical Seminar on Unmanned System Technology 2020: NUSYS'20 2022:Springer.

[43] Hakli H. BinEHO: a new binary variant based on elephant herding optimization algorithm. Neural Computing and Applications 2020; 32: p. 16971-16991.

[44] Ma J, Xia D, Guo H, Wang Y, Niu X, Liu Z, Jiang S. Metaheuristic-based support vector regression for landslide displacement prediction: A comparative study. Landslides 2022; 19(10): p. 2489-2511.

[45] Korkmaz S, Şahman MA, Cinar AC, Kaya E. Boosting the oversampling methods based on differential evolution strategies for imbalanced learning. Applied Soft Computing 2021; 112: p.107787.

[46] Kumar A, Wu G, Ali MZ, Mallipeddi R, Suganthan PN, Das S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm and Evolutionary Computation 2020; 56: p. 100693.

[47] Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the american statistical association 1937; 32(2012): p. 675-701.

[48] Uymaz SA. Evaluation of the Most Valuable Player Algorithm for Solving Real-World Constrained Optimization Problems. Bilişim Teknolojileri Dergisi 2021.

[49] Wansasueb K, Panmanee S, Panagant N, Pholdee N, Bureerat S, Yildiz AR. Hybridised differential evolution and equilibrium optimiser with learning parameters for mechanical and aircraft wing design. Knowledge-Based Systems 2022; 239: p. 107955.