Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Efficient Constrained Real-World Problem-Solving Using Gradient-Based Mutation Manta Ray Foraging Optimization (GM-MRFO)

Ahmad Azwan Abdul Razak[1], Ahmad Nor Kasruddin Nasir [1,*], Nor Maniha Abdul Ghani[1], Mohd Redzuan Ahmad[1], Julakha Jahan Jui[2]

[1] Faculty of Electrical & Electronics Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia
[2] Institute for Intelligent Systems Research and Innovation, Deakin University, Waurn Ponds, 3216, Victoria, Australia

ABSTRACT

This paper presents a Gradient-based Mutation Manta ray foraging optimization (GM-MRFO) that is designed to solve real-world optimization problems with constraints. GM-MRFO combines the basic strategy of MRFO with the Gradient-based Mutation (GM) strategy, which is a feasibility-and-solution repair strategy adopted from the ϵ-Matrix-Adaptation Evolution Strategy (ϵMAgES). MRFO algorithm is not immune to the common problems confronted by constrained optimization algorithms where constraints in the optimization problem are incompatible, and a solution that satisfies all constraints does not exist. In such cases, the MRFO algorithm may not be able to find a feasible solution. Another challenge is the optimization algorithm converges to a solution that is not globally optimal. By introducing the GM strategy and using Jacobian approximation in finite differences, GM-MRFO can improve the feasibility of solutions throughout the search process, which enables it to handle constraints more effectively than its predecessor. The proposed algorithm's performance is evaluated by assessing the accuracy of the best solution produced, the feasibility rate, mean of violation, success rate, and the ranking based on a ranking scheme in the Congress on Evolutionary Computation 2020 (CEC2020). Specifically, GM-MRFO achieved a feasibility rate of 100% on 47 out of 57 CEC2020 real-world problems and improved adequately best-known solutions.

Keywords:
Constraint optimization problem; manta ray foraging algorithm; gradient-based mutation; computational evolutionary competition 2020; real-world problem

1. Introduction

Optimization algorithms are gaining popularity in the current modern world. This is due to their reliability in providing an optimal solution for many complex real-world problems. The study conducted by Shatnawi *et al.*, [1] shows that optimization algorithms are widely used in machine learning to optimize artificial intelligence (AI) model. Moreover, with the availability of huge data, the application of such algorithm is inevitable. This is also true due to the fact that AI can solve a more complex system and handling uncertainty better than other models as stated in the study [2]. In literature, Noor *et al.*, [3] applied a linear integer programming to solve bus scheduling problem in

* *Corresponding author.*
E-mail address: kasruddin@umpsa.edu.my

https://doi.org/10.37934/araset.61.1.138157

deciding an optimal number of the bus trips in a certain period of time. Roslan *et al.*, [4] applied a linear regression optimization technique to evaluate an ideal spectral wavelength in detecting a buried archaeological proxy.

Optimization problems can be categorized into non-constraint and constraint. Non-constraint problem is normally adopted to test a newly develop optimization algorithm as reported in the study by Musa *et al.*, [5]. Real-world examples of such problems are presented by Mohanprakash *et al.*, [6] and Oleolo *et al.*, [7]. The search for an optimal solution for the problems is not restrained by too many constraints. On the other hand, the search for an optimal solution to an objective function in all constrained real-world problems is restricted by specific constraints on the components of the parameter vector. In optimization problems, constraints may arise due to factors such as limited resources or multiple sources [8], trade-offs specific to the problem [9], or physical boundaries [10]. These constraints make the problem more complex and require the development of specialized algorithms to solve them.

Generally, a constraint optimization problem can be expressed mathematically as shown in Eq. (1).

$$min\, f\,(x),\;\; x = (x_1, x_2, x_3, \ldots, x_D) \in \Omega \tag{1}$$

subject to

$$g_i(x) \leq 0,\;\; i = 1,2,3,\ldots,p$$

$$h_j(x) = 0,\;\; j = p+1, p+2, p+3, \ldots, m$$

$$\underline{x} \leq x \leq \overline{x}$$

Additionally, $g_i\,(x)$ represents the i^{th} inequality constraint, while $h_j\,(x)$ represents $the\; j^{th}$ equality constraint, which can be either a linear or non-linear constraint. Usually, equality constraints are transformed into inequality constraints as shown in Eq. (2).

$$\left|h_j(x)\right| - \varepsilon = 0,\;\; \text{for } j = p+1, p+2, p+3, \ldots, m \tag{2}$$

Recently, there has been increased attention on solving constraint problems, as reflected in the introduction of competitions to evaluate and compare state-of-the-art algorithms. While artificial test problems are commonly used to test optimization algorithms, they do not fully capture the complexity of real-world problems with constraints. Therefore, benchmark collections as in Congress on Evolutionary Computation 2020 (CEC2020) [11] that incorporated real-world constraints are used. A recent proposed algorithm called Manta ray foraging optimization (MRFO) offers a promising solution due to its unique approach. The MRFO as introduced by Zhao *et al.*, [12] is an optimization algorithm that mimics the foraging behavior of the manta ray, a cartilaginous fish species found in warm oceans around the world. Ma *et al.*, [13] investigated that Manta rays have a flat body with cephalic lobes on their mouth, which they use to channel and filter plankton, their main food source. They forage individually or in schools, forming a chain-like formation to scoop up plankton. When they encounter a location with abundant plankton, they use a cyclone foraging movement to concentrate the plankton and ease feeding. Finally, they perform somersault foraging, circling and moving backwards to draw more plankton towards them. The MRFO algorithm models all of these behaviors mathematically to guide agents towards the best solution in an optimization problem. The

fittest agent corresponds to the most forward manta ray in the chain, and the chain, cyclone, and somersault foraging behaviors guide the agents' movements towards the best solution.

The MRFO algorithm has exhibited encouraging outcomes in resolving several artificial benchmark issues and practical applications that possess varying types of fitness landscapes as indicated in the literature [14]. Its success is due to its powerful randomness strategy [15], combination of linear and spiral model equations [16], and elitism [17] in all foraging mechanisms. Despite its advantages, the MRFO algorithm is not immune to the common problems faced by constrained optimization algorithms. One of the major problems is infeasibility, which occurs when the constraints in the optimization problem are incompatible, and there is no solution that satisfies all the constraints. In such cases, the MRFO algorithm may fail to find a feasible solution, leading to suboptimal or even invalid results. Another problem is local optima, where the optimization algorithm converges to a solution that is not globally optimal but only locally optimal. When the optimization process gets stuck in a local optimum, it may fail to explore the entire solution space and find the global optimum.

To address the limitations of the MRFO algorithm, a constraint handling technique known as Gradient-based mutation (GM) has been incorporated into the MRFO procedure. In literatures, GM has been applied in evolutionary multi-objective algorithm utilizing a local search [18] and stochastic mutation [19]. It also has been applied in differential evolutionary (DE) algorithm [20], archive-DE [21], elite-DE [22] and crossover-mutation DE [23]. This approach improves the algorithm's ability to handle constraints by incorporating gradient information into the search process. The GM technique modifies the search direction and step size based on the gradient of the objective function and the constraints, allowing the algorithm to better navigate the solution space and find feasible solutions that satisfy all constraints.

The aforementioned literatures have shown the limitation of the MRFO in dealing with constraint optimization problems. On top of that, there is limited literature is found on evaluating MRFO in solving such problems which indicates a potential gap of the algorithm. In this paper, GM is combined with MRFO to solve problems with constraints. This combination is called Gradient-based Mutation Manta Ray Foraging Optimization (GM-MRFO). The GM-MRFO method employs a deterministic and gradient-based searching method to improve both the fitness and feasibility of the solution. Additionally, it adopts an adjustment technique for the threshold of constraint violation value. The GM technique is also modified to match and blend with MRFO, resulting in an improved algorithm for solving constrained optimization problems. Subsequently, in order to assess the efficacy of the suggested algorithm, a set of 57 real-world constrained problems from different fields with varying numbers of decision variables and constraints listed in CEC2020 are employed. The paper is organized in several sections covering the background as in Section 2, the proposed technique as in Section 3, experimental setup as in Section 4, and summary as in Section 5.

2. Background and Related Work

2.1 Manta Ray Foraging Optimization (MRFO)

The algorithm MRFO is inspired by three foraging behaviors of manta rays, namely chain, cyclone, and somersault foraging. The first strategy utilized in MRFO is chain foraging. In real-life, a location consisting of plankton can be detected by manta rays, which they will swim towards. The richness of the location is determined by the concentration of the plankton. During movement, manta rays line up by linking their head to the tail of the manta ray in front of them, creating a chain-like formation. In MRFO, the best location is considered as the location with the highest concentration of plankton, and it is the location that the agent moves towards throughout the searching process. All individuals

except the first-ranked agent, which is considered as the best location, move towards the best location as well as move with respect to the best solution. The best solution, which is updated for each iteration, guides the movement, and the movement with respect to the front individuals is the strategy to avoid premature convergence. Chain foraging can be modeled mathematically as shown in Eq. (3) and Eq. (4).

$$x_i^d(k+1) = \begin{cases} x_i^d(k) + r.\left(x_{best}^d(k) - x_i^d(k)\right) + \alpha.\left(x_{best}^d(k) - x_i^d(k)\right), & i = 1 \\ x_i^d(k) + r.\left(x_{i-1}^d(k) - x_i^d(k)\right) + \alpha.\left(x_{best}^d(k) - x_i^d(k)\right), & i = 2,3,4,\ldots,N \end{cases} \quad (3)$$

$$\alpha = 2r\sqrt{|log\, r|} \quad (4)$$

The Eq. (5) and Eq. (6) defines the next position of the i^{th} individual in the D^{th} dimension, denoted as $x_i^d(k+1)$, where $x_i^d(k)$, represents the current position of the same individual. The variable *r* represents a random vector within the range of [0,1], α is a weight coefficient of the chain foraging, and x_{best}^D denotes the current best solution found so far. The i^{th} individual's position update is based on the $(i\text{-}1)^{th}$ individual's position and the best solution x_{best}^d. Moving on to the next step in MRFO, the agents are moved in a spirally path. The manta rays, while remaining in chain formation, simultaneously swim towards the plankton in a spiral move once they detect the best location of food. This motion of agents can be expressed mathematically in a multi-dimensional space using Eq. (5).

$$x_i^d(k+1) = \begin{cases} x_i^d(k) + r.\left(x_{best}^d(k) - x_i^d(k)\right) + \alpha.\left(x_{best}^d(k) - x_i^d(k)\right), & i = 1 \\ x_i^d(k) + r.\left(x_{i-1}^d(k) - x_i^d(k)\right) + \alpha.\left(x_{best}^d(k) - x_i^d(k)\right), & i = 2,3,4,\ldots,N \end{cases} \quad (5)$$

$$\beta = 2e^{r_1\frac{K-k+1}{K}}.sin\,2\pi r_1 \quad (6)$$

Eq. (7) and Eq. (8) represent the mathematical expressions of the random-cyclone foraging strategy in MRFO. This strategy, characterized by the weight coefficient β, aims to improve the exploration ability of the agents in the feasible region by inducing random movements in new directions. By doing so, the agents are forced to conduct a more comprehensive global search, thus enhancing the overall effectiveness of the MRFO algorithm. The maximum number of iterations is represented by *K*, while r_1 stands for the random vector with a range of [0,1].

$$x_{rand}^d(k) = \underline{x^d} + rand\left(\overline{x^d} - \underline{x^d}\right) \quad (7)$$

$$x_i^d(k+1) = \begin{cases} x_{rand}^d(k) + r.\left(x_{rand}^d(k) - x_i^d(k)\right) + \beta.\left(x_{rand}^d(k) - x_i^d(k)\right), & i = 1 \\ x_{rand}^d(k) + r.\left(x_{i-1}^d(k) - x_i^d(k)\right) + \beta.\left(x_{rand}^d(k) - x_i^d(k)\right), & i = 2,3,4,\ldots,N \end{cases} \quad (8)$$

The equation includes $x_{rand}^d(k)$, a randomly generated position within the feasible region, $\overline{x^d}$ and $\underline{x^d}$, which are the upper and lower boundaries of the d^{th} dimension respectively. MRFO employs the somersault foraging strategy to enhance the exploitation of the best solution. In nature, somersault refers to the movement of a manta ray around a food location in a to-and-fro manner, appearing as a pivot. In a similar way, the searching agents in MRFO move in a circular motion around the best agent in the random search domain. This helps to improve the solution without being trapped in local optima. The mathematical representation of the somersault foraging strategy is

shown in Eq. (9). It includes the weight coefficient, *S*, and two random numbers, r_2 and r_3, within the range of [0,1].

$$x_i^d(k+1) = x_i^d(k) + S.\left(r_2.x_{best}^d(k) - r_3.x_i^d(k)\right), \quad i = 1,2,3,\ldots,N \tag{9}$$

2.2 Gradient-based Mutation for Constraint Handling

The approach aims to systematically repair infeasible solutions in optimization problems by utilizing gradient information derived from the constraint set as described in the literature [21]. The gradient is utilized to direct the infeasible solutions towards the feasible region, which can be derived directly from the constraints or approximated by an inexact method. This method was implemented using feasibility rules based on the "*treatment of box constraints*" constraint handling techniques as noted in the studies by Wessing [24] and Kundu *et al.*, [25]. This ensures that all individuals satisfy the box-constraints and respect the lower and upper bounds of the feasible region. By combining the use of gradient information and feasibility rules, the proposed method provides a systematic and effective approach for repairing infeasible solutions. Any individual that exceeds the value of $i \in \{1,2,3,\ldots,N\}$ of x is then transformed into the box by a mathematical operation to comply with the constraints, as expressed in Eq. (10).

$$x_i = \begin{cases} \underline{x}_D + \left((\underline{x}_D - x_i) - \left\lfloor \frac{\bar{x}_D - x_i}{\omega_i} \right\rfloor \omega_i\right), & \text{if } x_i \leq \underline{x}_D \\ \bar{x}_D + \left((x_i - \bar{x}_D) - \left\lfloor \frac{x_i - \bar{x}_D}{\omega_i} \right\rfloor \omega_i\right), & \text{if } x_i \leq \bar{x}_D \\ x_i, & \text{if } else \end{cases} \tag{10}$$

In the equation, x_i refers to an i^{th} individual in a population of searching agents, while $\overline{x^d}$ and $\underline{x^d}$ denote the upper and lower boundaries of the feasible region respectively. Additionally, ω_i represents the component wise distance between $\overline{x^d}$ and $\underline{x^d}$ by taking the difference between the upper and lower boundaries for each dimension.

The procedure involves a crucial step of ranking all agents, where feasible solutions are given priority over infeasible ones. The ranking process uses a lexicographic ordering method or *lex*, which considers feasibility as the primary criterion followed by the fitness value of the objective function. To handle infeasible solutions, a relaxation threshold known as ε-level ordering is employed. This threshold allows the algorithm to handle solutions that violate constraints but still fall below the threshold. In constraint optimization, the threshold refers to a predefined limit or value used to measure the degree of violation of a constraint. If the degree of violation is within the threshold, the solution is considered feasible (ε-feasible), while exceeding the threshold renders the solution infeasible (ε-infeasible). The threshold value can be set by the user or determined by the algorithm. The ε-level threshold is adjusted based on the current iteration k to determine the feasibility of the solution. Reducing the ε-level not only encourages exploration in the feasible region during the early phase but also facilitates exploitation of the solution in the later phase. However, it is essential to be cautious during exploration to avoid searching for solutions outside the boundaries, particularly for monotonic problems, as this may lead to constraint violation and defeat the algorithm's purpose.

Based on the previous study by Hellwig *et al.*, [26], the ε-level order relation, denoted as $\leq_\varepsilon$, can be mathematically expressed as follows.

i. Consider x_i and x_j as two possible solutions, i^{th} and j^{th}, respectively, in the entire *N*-dimensional region, R^N.
ii. Let $(f_i, v_i) = (f(x_i), v(x_i))$ and $(f_j, v_j) = (f(x_j), v(x_j))$, where f_i and f_j are the objective function values of x_i and x_j, respectively, and v_i and v_j are their corresponding constraint violation levels as in Eq. (11).

$$x_i \leq_\varepsilon x_j \Leftrightarrow \begin{cases} f_i \leq f_j, & \text{if } (v_i \leq \varepsilon^k) \wedge (v_j \wedge \varepsilon^k) \\ f_i \leq f_j, & \text{if } v_i = v_j \\ v_i < v_j, & \text{if otherwise.} \end{cases} \tag{11}$$

Based on these equations, the searching agents are evaluated according to the following criteria:

i. IF both of x_i and x_j are ε-feasible, THEN ranked them based on fitness value;
ii. IF both of x_i and x_j are ε-infeasible solution, THEN ranked based on constraint violation values.

The ε-level ordering is aided by a technique called GM repair, which attempts to fix any infeasible solutions by mutating them a certain number of times (p). The mutation is based on the gradient of the constraint functions and is carried out using finite difference approximation of the Jacobian as noted in the study [26]. Although this procedure can be effective in improving the quality of solutions, it also increases the computational cost and complexity of the algorithm by requiring additional function evaluations. After γ_T trials, the newly mutated individuals will be considered as new possible solutions, and may either become ε-feasible or remain ε-infeasible.

To explain GM, we denote a sequence of vectors that contain all *n* constraint values, (κ) of a problem using the expression in (12) as noted in the studies by Takahama *et al.*, [21] and Hellwig *et al.*, [26].

$$\kappa(x) = \big(g_1(x), g_2(x), \ldots, g_l(x), h_1(x), h_2(x), \ldots, h_k(x)\big) \tag{12}$$

One can determine the extent to which an infeasible solution x fails to meet the constraints by calculating the constraint violations, $\Delta\kappa$, which can be expressed in the manner as in Eq. (13).

$$\Delta\kappa(x) = \begin{pmatrix} max\big(0, g_1(x)\big), max\big(0, g_2(x)\big), \ldots, max\big(0, g_l(x)\big), \\ h_1(x), h_2(x), \ldots, h_k(x) \end{pmatrix} \tag{13}$$

The next step of the proposed method is to fix the infeasible agents x by incorporating a correction vector, Δx as shown in Eq. (14).

$$x^{new} = x + \Delta x \tag{14}$$

The given Eq. (15) represents the Jacobian matrices, $\nabla\kappa$ utilizes the linear system presented in Eq. (14) to compute the Δx.

$$\nabla\kappa(x)\Delta x = -\Delta\kappa(x) \tag{15}$$

In order to apply this method, it is required to establish the Jacobian matrix with respect to x. However, due to the characteristics of constrained-black-box optimization, an estimation of the Jacobian matrix is utilized, which is then used to compute finite differences. Therefore, the Jacobian matrix can be expressed as follows:

$$\nabla\kappa(x) = \begin{pmatrix} \frac{\partial g_1(x)}{\partial x_1} & \frac{\partial g_1(x)}{\partial x_2} & \cdots & \frac{\partial g_1(x)}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_q(x)}{\partial x_1} & \frac{\partial g_q(x)}{\partial x_2} & \cdots & \frac{\partial g_q(x)}{\partial x_n} \\ \frac{\partial h_{q+1}(x)}{\partial x_1} & \frac{\partial h_{q+1}(x)}{\partial x_2} & \cdots & \frac{\partial h_{q+1}(x)}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial h_m(x)}{\partial x_1} & \frac{\partial h_m(x)}{\partial x_2} & \cdots & \frac{\partial h_m(x)}{\partial x_n} \end{pmatrix} \tag{16}$$

The Jacobian matrix is a mathematical matrix consisting of all the first-order partial derivatives of a vector-valued function. This means that it contains information on how each element of a vector function changes in response to changes in each of its variables. However, since constrained-black-box optimization makes it challenging to define the Jacobian matrix accurately, an approximation method is used, which involves utilizing the pseudo-inverse of the matrix, denoted as $\nabla\kappa(x)^{-1}$, as shown in Eq. (17), to compute an approximate solution for Eq. (15).

$$\Delta x = -\nabla\kappa(x)^{-1}\Delta\kappa(x) \tag{17}$$

The pseudo-inverse is a mathematical tool used to solve linear equations in cases where there are more equations than unknowns, unlike regular inverse which only exists for square matrices. It has various applications in fields such as control theory, signal processing, and optimization. In optimization, the pseudo-inverse can be used to approximate the gradient of a function when the actual gradient is unknown or difficult to compute. If the GM repair method fails to produce a feasible solution, it includes the last modified individuals in the subsequent search phase, similar to the ε-level threshold. In typical optimization methods, constraint violations tend to decrease gradually.

3. The Proposed Gradient-Based Manta Ray Foraging Optimization (GM-MRFO)

The main difference between the MRFO and GM-MRFO algorithms is the inclusion of constraint handling in GM-MRFO. While both algorithms are based on the nature-inspired manta ray optimization, GM-MRFO includes additional procedures to handle constraints. In MRFO, the algorithm mainly focuses on minimizing the objective function and does not take into account any constraints. This can lead to solutions that violate constraints and are therefore infeasible. On the other hand, GM-MRFO incorporates a constraint handling mechanism that considers both the objective function and the constraints to ensure that only feasible solutions are generated. The constraint handling procedure in GM-MRFO involves two steps. First, the GM is used to punish infeasible solutions by adding a penalty term to the objective function. This penalty term increases with the degree of constraint violation. Second, a constraint handling technique called ε-level ordering is employed to ensure that the search process prioritizes feasible solutions over infeasible ones. This technique involves comparing the feasibility of solutions based on a predefined threshold or constraint violation level, with feasible solutions being ranked higher. Overall, the inclusion of constraint handling in GM-MRFO ensures that the algorithm is more effective at generating feasible

solutions for real-world optimization problems with constraints. The pseudocode of GM-MRFO is shown as follows.

set optimization problem parameters
set maximum number of function evaluations
set maximum number of iterations
initialize the population, pop with a number m of manta ray, X_m.
evaluate fitness of each individual
evaluate constraint violated, v of the new individual
if violated, repair for p times
    Apply Gradient-based Mutation in Eqs. (10)-(17).
endif
set the global best individual based on the best fitness value
set $k = 1$
while $k < K$
for i = 1 : m
    determine the coefficient k/K
    for each individual in the population
        If $rand < 0.5$
            apply cyclone foraging using Eq. (5)
        elseif $k/K < rand$
            apply random exploration using Eq. (8)
        endif
        Apply chain foraging using Eq. (3).
    End for
    evaluate the fitness $f(x)$
    evaluate constraint violation, v of the new individual
    if violated, repair for p times
        Apply Gradient-based Mutation in Eqs. (10)-(17).
    endif
end for
update the global best individual if a better solution is found
For *i* = 1 : *m*
    Apply somersault foraging using Eq. (9).
    evaluate the fitness $f(x)$
    evaluate constraint violated, v of the new individual
    if violated, repair for p times
        Apply Gradient-based Mutation in Eqs. (10)-(17).
    endif
    update the global best individual if a better solution is found
end for
update ε-value based on v of the best individual
rank the population using the ε-constraint handling approach
store the best fitness value of the current iteration
update $k = k + 1$
endwhile

return the global best individual, evaluation results, population ranking, and best fitness values of each iteration

end

return x_{best}

4. Experimental Setup and Results of the Benchmark Functions Test

In this section, the experimental setup for testing the benchmark functions and the benchmark functions themselves are presented. These were used to evaluate the performance of the proposed GM-MRFO.

4.1 Benchmark Functions and Hardware

The main goal of creating a benchmark function is to assess how well a newly developed algorithm performs. In the work, CEC2020, which comprises 57 real-world constrained problems, was adopted as the benchmark functions. These problems were selected from various real-world problems to cover a wide range of fitness landscapes and features. The number of decision variables ranging from 2 to 158 and the number of equality constraints varying from 0 to 148. During the optimization process, all the problems in the benchmark suite are treated as black-box systems, which reflects most real-world problems. Table 1 shows the categories of problems included in the collection, which have varying numbers of equality and inequality constraints as reported in the literature [11].

Table 1
The categories of problems in CEC2020

Real-world application	Number of problems
Industrial chemical processes	7 (RC01-RC07)
Process synthesis and design problems	7 (RC08-RC14)
Mechanical engineering problems	19 (RC15-RC33)
Power system problems	11 (RC34-RC44)
Power electronic problems	6 (RC45-RC50)
Livestock feed ration optimization	7 (RC51-RC57)

The MATLAB software has been used to implement all the algorithms mentioned above. The performance evaluation of the proposed benchmark suite was conducted using MATLAB R2020b on a computer with an INTEL Core i7 CPU, 8GB RAM, and the Microsoft Windows 10 operating system. The parameter settings of each algorithm were adopted directly from their respective papers, as cited in [22-24]. A stopping rule based on the number of decision variables was employed to terminate the optimization process in all algorithms based on the following criteria.

$$K = \begin{cases} 1 \times 10^5, & \text{if } D \leq 10 \\ 2 \times 10^5, & \text{if } 10 < D \leq 30 \\ 4 \times 10^5, & \text{if } 30 < D \leq 50 \\ 8 \times 10^5, & \text{if } 50 < D \leq 150 \\ 10^6, & \text{if } 150 < D \end{cases} \tag{18}$$

The formula given calculates the maximum allowable number of function evaluations (*K*) for a problem based on its dimensionality (*D*).

4.2 Performance Evaluation Procedure

In the work, procedures were adopted to determine the relative difficulty level of each problem in the proposed benchmark suite. As the difficulty level and complexity of the problems are different from each other, the following steps were taken.

i. Each problem of the benchmark suite is independently run 25 times with the above-mentioned algorithms.
ii. The results of the algorithms are reported as the mean objective function (Mean), mean constraint violation (MV), Feasibility Rate (FR), and Success Rate (SR) based on the 25 runs.

   a. The mean constraint violation (v) is determined using the Eq. (19).
   b. Feasibility Rate is defined as the proportion of runs in which at least one feasible solution is obtained within *K*, divided by the total number of runs.
   c. The Success Rate refers to the ratio of the total number of runs in which an algorithm achieves a feasible solution x that satisfies the condition $f(x) - f(x^*) \leq 10^{-8}$ within K, to the total number of runs.

$$\bar{v} = \frac{\sum_{i=1}^{p} max(g_i(\bar{x}),0)+\sum_{i=1}^{p} max(|h_i(\bar{x})|-\varepsilon,0)}{m}, \text{ where } \varepsilon = 0.0001 \quad (19)$$

The problems are evaluated in terms of difficulty level based on the following criteria. In the next phase of performance test, the difficulty level of problems is assessed using the following steps:

i. The problems are first evaluated based on SR.
ii. Then, the problems are evaluated based on FR.
iii. Lastly, the problems are evaluated based on MV.

In addition to evaluating the performance of individual algorithms, the purpose of a benchmark suite is to determine which algorithm is the most effective in solving the benchmark problems. To facilitate a comparative analysis of algorithm performance, a suitable ranking procedure is necessary. In the realm of constrained optimization, a commonly used approach for comparative analysis of algorithm performance is through an ordering method or a quality indicator that can distinguish between feasible and infeasible solutions. One popular ordering method is based on the superiority of feasible solutions, where solutions are compared based on the following criteria:

i. a feasible solution is considered better than an infeasible solution,
ii. two feasible solutions are ranked based on objective function value, with the one having a lower value being considered better, and
iii. among two infeasible solutions, the one with a lower constrained violation value is preferred.

In mathematical terms, this ordering approach can be defined as follows:

$$\bar{x}_1 \geq_{lex} \bar{x}_2 \Leftrightarrow \begin{cases} f(\bar{x}_1) \geq f(\bar{x}_2), & \text{if } v(\bar{x}_1) == v(\bar{x}_2) \\ v(\bar{x}_1) \geq v(\bar{x}_2), & \text{else} \end{cases} \quad (20)$$

The commonly used approach for ranking schemes in CEC benchmarks is based on the superiority of feasible solutions, as defined by the criteria of any feasible solution being considered better than an infeasible solution, ranking two feasible solutions based on objective function value, and preferring

the infeasible solution with the lower constrained violation value. The performance measure for each algorithm is defined as Eq. (21).

$$PM_i = 0.5\sum_{j=1}^{57} w_j \widehat{\Upsilon}_{i,j}^{best} + 0.3\sum_{j=1}^{57} w_j \widehat{\Upsilon}_{i,j}^{mean} + 0.2\sum_{j=1}^{57} w_j \widehat{\Upsilon}_{i,j}^{medium} \quad (21)$$

The values of $\widehat{\Upsilon}_{i,j}^{best}$, $\widehat{\Upsilon}_{i,j}^{mean}$, and $\widehat{\Upsilon}_{i,j}^{medium}$ represent the normalized adjusted objective function value of the best, mean, and medium solution, respectively, of the j^{th} problem for the i^{th} algorithm. The weight value of the j^{th} problem, denoted as w_j, is set based on a specific criterion.

$$w_i = \begin{cases} 0.008, & \text{if } D_j \leq 10 \\ 0.016, & \text{if } 10 < D_j \leq 30 \\ 0.024, & \text{if } 30 < D_j \leq 50 \\ 0.032, & \text{if } 50 < D_j \leq 150 \\ 0.040, & \text{if } 150 < D_j \end{cases} \quad (22)$$

In order to determine the normalized adjusted objective function value of the best solution obtained by an algorithm on a given benchmark problem, the following approach is employed.

i. To determine the worst feasible solution in the competition for the j^{th} problem, the following process is adopted: the combined set of best solutions from all algorithms is considered, and the solution with the highest fitness value is selected as the worst feasible solution (*f, F, best, worst, j*). If there is no feasible solution in the combined set, the worst feasible solution is set to 0.
ii. After obtaining the worst feasible solution from the combined set of best solutions of all algorithms in the competition for the j^{th} problem, the adjusted objective function value of the best solution for each algorithm is computed using the Eq. (23).
iii. Finally, the adjusted objective function value of the best solution for each algorithm is normalized using the Eqs. (24) to (26).

$$\Upsilon_{i,j}^{best} = \begin{cases} f_{worst,j}^{F,best} + v_{i,j}^{best}, & v_{i,j}^{best} > 0 \\ f_{i,j}^{best}, & \text{if } v_{i,j}^{best} \leq 0 \end{cases} \quad (23)$$

$$\widehat{\Upsilon}_{i,j}^{best} = \frac{\Upsilon_{i,j}^{best} - \Upsilon_{min,j}^{best}}{\Upsilon_{max,j}^{best} - \Upsilon_{min,j}^{best}} \quad (24)$$

$$\Upsilon_{min,j}^{best} = min\left\{\Upsilon_{1,j}^{best}, \Upsilon_{2,j}^{best}, \Upsilon_{3,j}^{best}, \ldots, \Upsilon_{i,j}^{best}, \ldots\right\} \quad (25)$$

$$\Upsilon_{max,j}^{best} = max\left\{\Upsilon_{1,j}^{best}, \Upsilon_{2,j}^{best}, \Upsilon_{3,j}^{best}, \ldots, \Upsilon_{i,j}^{best}, \ldots\right\} \quad (26)$$

A comparable approach is used to compute the adjusted objective function value of the mean and median solutions of the algorithms.

4.3 Discussion on the Outcomes

Table 2 displays the results of the evaluation of three optimization algorithms, namely GM-MRFO, ε-Matrix-Adaptation Evolution Strategy (ϵMAgES), and COLSHADE, on the Livestock Feed Ration Optimization Problems in terms of four performance metrics: best accuracy, feasibility rate (FR), mean of violation (MV), and success rate (SR). Regarding the best accuracy, GM-MRFO exhibited excellent performance for all functions RC01-RC07. On the other hand, ϵMAgES and COLSHADE did not show good accuracy results for any of the functions. In terms of the FR test, GM-MRFO achieved a 100% FR for RC01, RC02, RC04, and RC06, while it yielded a 0% FR for RC03 and RC05. ϵMAgES, on the other hand, had a 0% FR for all functions except for RC04 and RC05, where it scored 60% and 76%, respectively. COLSHADE had an 88%-100% FR for all functions except for RC06 and RC07, where it scored 0%. For the MV metric, GM-MRFO displayed excellent performance (0.00E+00) for RC01, RC02, RC04, and RC06. ϵMAgES did not show good performance for any of the functions, while COLSHADE performed well (0.00E+00) for RC02, RC03, RC04, and RC05. Finally, in terms of the SR metric, GM-MRFO, ϵMAgES, and COLSHADE scored 0% for all functions except for COLSHADE, which achieved a success rate of 84% and 4% for RC04 and RC05, respectively. In conclusion, the evaluation results suggest that GM-MRFO is the most effective algorithm among the three for solving the optimization problem for all functions RC01-RC07. COLSHADE, on the other hand, exhibited competitive performance in terms of the FR and MV metrics but lagged behind in terms of the success rate. ϵMAgES, however, did not perform well in any of the metrics evaluated.

Table 2
The outcomes of solving the livestock feed ration optimization problems (RC01-RC07) with GM-MRFO, ϵMAgES, and COLSHADE

Function	Algorithm	Best	Median	Mean	Worst	Std	FR	MV	SR
RC01	GM-MRFO	1.89E+02	1.89E+02	1.89E+02	1.90E+02	6.45E-02	100	0.00E+00	0
	ϵMAgES	1.91E+02	1.91E+02	1.90E+02	1.90E+02	3.87E-01	0	2.34E+05	0
	COLSHADE	1.91E+02	1.91E+02	1.90E+02	1.90E+02	2.76E-01	88	7.09E-06	0
RC02	GM-MRFO	7.12E+03	7.21E+03	7.21E+03	7.37E+03	5.15E+01	100	0.00E+00	0
	ϵMAgES	7.12E+03	7.21E+03	7.21E+03	7.37E+03	5.26E+01	0	1.46E+05	0
	COLSHADE	7.12E+03	7.21E+03	7.21E+03	7.37E+03	5.21E+01	100	0.00E+00	0
RC03	GM-MRFO	-1.92E+04	-1.53E+04	-1.37E+04	1.52E+03	4.14E+03	0	2.77E+01	0
	ϵMAgES	-1.92E+04	-1.53E+04	-1.37E+04	1.52E+03	4.14E+03	0	1.84E+02	0
	COLSHADE	-1.92E+04	-1.53E+04	-1.37E+04	1.52E+03	4.14E+03	100	0.00E+00	0
RC04	GM-MRFO	-3.88E-01	-3.88E-01	-3.88E-01	-3.87E-01	1.27E-04	100	0.00E+00	0
	ϵMAgES	1.94E-01	3.50E-01	9.64E-01	-2.36E-01	6.73E-01	60	5.24E-04	0
	COLSHADE	5.73E-01	3.50E-01	1.02E+00	3.65E-03	9.71E-01	100	0.00E+00	84
RC05	GM-MRFO	-1.50E+03	2.70E+01	-3.73E+01	2.47E+02	3.17E+02	0	6.54E+00	0
	ϵMAgES	-1.50E+03	2.75E+01	-3.64E+01	2.48E+02	3.18E+02	76	1.34E+01	0
	COLSHADE	-1.50E+03	2.76E+01	-3.63E+01	2.48E+02	3.19E+02	100	0.00E+00	4
RC06	GM-MRFO	1.91E+00	2.03E+00	2.03E+00	2.13E+00	6.86E-02	100	0.00E+00	0
	ϵMAgES	3.15E+00	3.70E+00	2.86E+00	3.74E+00	8.92E-01	0	8.24E+01	0
	COLSHADE	3.39E+00	3.70E+00	3.08E+00	3.41E+00	2.59E-01	0	2.18E-02	0
RC07	GM-MRFO	1.51E+00	2.10E+00	2.08E+00	2.46E+00	2.07E-01	0	5.98E-01	0
	ϵMAgES	2.45E+00	3.31E+00	3.23E+00	4.20E+00	8.60E-01	0	8.09E+01	0
	COLSHADE	2.71E+00	2.71E+00	3.48E+00	3.32E+00	8.49E-01	0	3.98E-02	0

The results of the experiments showed that GM-MRFO exhibited good performance in terms of best accuracy for all functions in Process Synthesis and Design Problems (RC08-RC14), as shown in Table 3. On the other hand, ϵMAgES and COLSHADE did not perform well in this aspect. When it comes to the feasibility rate (FR) test, GM-MRFO produced 100% for RC08-RC014, while ϵMAgES

produced 100% for RC08-RC014 except for RC013, which was 0%, and COLSHADE produced 100% for RC08-RC014 except for RC013 and RC014, which were 24% and 100%, respectively. In terms of mean of violation (MV), GM-MRFO demonstrated good performance (0.00E+00) for functions RC08-RC11, RC13, and RC14. ϵMAgES exhibited good performance (0.00E+00) for functions RC08-RC10 and RC12-RC14, while COLSHADE demonstrated good performance (0.00E+00) for functions RC08-RC10 and RC12-RC14. As for the success rate (SR), GM-MRFO exhibited 100% success rate for RC08, RC010, and RC014, while ϵMAgES had 0% success rate for all functions RC08-RC014, and COLSHADE had a 100% success rate for RC08-RC010 and RC012, 84% for RC011, and 0% for RC013 and RC014. These results suggest that GM-MRFO outperformed ϵMAgES and COLSHADE in various aspects, making it a promising approach for function optimization problems.

Table 3
The outcomes of solving the of process synthesis and design problems (RC08-RC14) with GM-MRFO, ϵMAgES, and COLSHADE

Function	Algorithm	Best	Median	Mean	Worst	Std	FR	MV	SR
RC08	GM-MRFO	2.00E+00	2.00E+00	2.00E+00	2.00E+00	0.00E+00	100	0.00E+00	100
	ϵMAgES	3.41E+00	3.14E+00	2.38E+00	2.44E+00	1.30E+00	100	0.00E+00	0
	COLSHADE	3.26E+00	3.40E+00	2.95E+00	2.12E+00	1.18E+00	100	0.00E+00	100
RC09	GM-MRFO	2.56E+00	2.56E+00	2.56E+00	2.56E+00	2.56E-06	100	0.00E+00	0
	ϵMAgES	4.25E+00	3.46E+00	3.52E+00	3.99E+00	5.14E-01	100	0.00E+00	0
	COLSHADE	4.09E+00	4.10E+00	3.10E+00	3.94E+00	1.14E+00	100	0.00E+00	100
RC10	GM-MRFO	1.08E+00	1.08E+00	1.08E+00	1.08E+00	4.53E-16	100	0.00E+00	100
	ϵMAgES	2.11E+00	1.46E+00	2.50E+00	2.17E+00	2.77E-01	100	0.00E+00	0
	COLSHADE	1.84E+00	2.33E+00	2.69E+00	2.56E+00	9.10E-01	100	0.00E+00	84
RC11	GM-MRFO	9.92E+01	9.92E+01	9.92E+01	9.93E+01	5.91E-03	100	0.00E+00	0
	ϵMAgES	1.00E+02	1.01E+02	1.00E+02	1.01E+02	6.21E-01	0	9.85E-02	0
	COLSHADE	1.00E+02	1.00E+02	1.00E+02	1.01E+02	1.02E+00	24	9.50E-02	0
RC12	GM-MRFO	2.92E+00	2.94E+00	2.94E+00	2.97E+00	1.26E-02	96	5.52E-15	0
	ϵMAgES	4.15E+00	3.41E+00	3.75E+00	4.72E+00	1.66E+00	100	0.00E+00	0
	COLSHADE	4.64E+00	3.70E+00	3.41E+00	3.97E+00	1.19E+00	100	0.00E+00	100
RC13	GM-MRFO	2.73E+04	2.82E+04	2.82E+04	2.94E+04	5.28E+02	100	0.00E+00	0
	ϵMAgES	2.73E+04	2.82E+04	2.82E+04	2.94E+04	5.29E+02	100	0.00E+00	0
	COLSHADE	2.73E+04	2.82E+04	2.82E+04	2.94E+04	5.29E+02	100	0.00E+00	0
RC14	GM-MRFO	6.46E+04	1.11E+05	1.09E+05	1.54E+05	2.12E+04	100	0.00E+00	0
	ϵMAgES	6.46E+04	1.11E+05	1.09E+05	1.54E+05	2.12E+04	100	0.00E+00	0
	COLSHADE	6.46E+04	1.11E+05	1.09E+05	1.54E+05	2.12E+04	100	0.00E+00	0

In this thesis section, the performance of various methods for solving Mechanical Engineering Problems (RC15 -RC33) is evaluated based on their accuracy, feasibility rate, mean violation, and success rate, as shown in Table 4. In terms of accuracy, GM-MRFO demonstrated good performance for all functions RC15-RC20 and achieved the best accuracy of solution for all functions RC21-RC33. However, ϵMAgES and COLSHADE did not perform well for any of the functions in terms of accuracy. Regarding feasibility rate, GM-MRFO produced high rates of feasibility for RC15-RC20 (84-100%) and RC28-RC33 (76-100%), whereas ϵMAgES and COLSHADE produced 100% feasibility rates for most functions. In the case of RC21-RC27, GM-MRFO had lower feasibility rates than the other methods for some functions. Furthermore, in terms of mean violation, GM-MRFO performed well with a mean violation of 0.00E+00 for most of the functions, while ϵMAgES and COLSHADE also performed well for some functions but not as consistently as GM-MRFO. Finally, the success rate of the methods was evaluated, and GM-MRFO and COLSHADE achieved better success rates than ϵMAgES for most functions. Specifically, GM-MRFO had 100% success rate for some functions in RC15-RC20 and RC28-RC33, but lower success rates for some functions in RC21-RC27. On the other hand, COLSHADE had

a high success rate for RC21-RC27 but lower rates for some functions in RC15-RC20 and RC28-RC33. For this cluster, GM-MRFO showed good overall performance, particularly for accuracy and mean violation, while ϵMAgES and COLSHADE had some limitations in their performance for solving mechanical engineering problems.

Table 4
The outcomes of solving the of mechanical engineering problems (RC15-RC33) with GM-MRFO, ϵMAgES, and COLSHADE

Function	Algorithm	Best	Median	Mean	Worst	Std	FR	MV	SR
RC15	GM-MRFO	3.04E+03	3.30E+03	3.52E+03	5.55E+03	6.69E+02	84	1.29E-17	0
	ϵMAgES	3.04E+03	3.30E+03	3.52E+03	5.55E+03	6.70E+02	100	0.00E+00	0
	COLSHADE	3.04E+03	3.30E+03	3.52E+03	5.55E+03	6.70E+02	100	0.00E+00	0
RC16	GM-MRFO	1.12E+00	2.81E+01	1.36E+02	1.39E+03	3.06E+02	100	0.00E+00	0
	ϵMAgES	2.36E+00	2.91E+01	1.37E+02	1.39E+03	3.07E+02	100	0.00E+00	0
	COLSHADE	2.44E+00	2.88E+01	1.37E+02	1.39E+03	3.07E+02	100	0.00E+00	100
RC17	GM-MRFO	1.29E-02	1.36E-02	1.38E-02	1.60E-02	7.44E-04	100	0.00E+00	0
	ϵMAgES	1.56E+00	1.56E+00	1.09E+00	9.87E-02	1.05E+00	100	0.00E+00	0
	COLSHADE	9.51E-01	1.31E+00	8.53E-01	7.30E-01	1.04E+00	100	0.00E+00	96
RC18	GM-MRFO	6.28E+03	8.49E+03	8.55E+03	9.62E+03	8.78E+02	100	0.00E+00	0
	ϵMAgES	6.28E+03	8.49E+03	8.55E+03	9.62E+03	8.80E+02	100	0.00E+00	0
	COLSHADE	6.28E+03	8.49E+03	8.55E+03	9.62E+03	8.80E+02	100	0.00E+00	0
RC19	GM-MRFO	1.67E+00	1.67E+00	1.67E+00	1.67E+00	1.55E-11	100	0.00E+00	100
	ϵMAgES	3.04E+00	2.36E+00	3.12E+00	2.31E+00	3.42E-01	100	0.00E+00	0
	COLSHADE	2.63E+00	2.18E+00	3.00E+00	2.63E+00	4.57E-01	100	0.00E+00	100
RC20	GM-MRFO	2.64E+02	2.64E+02	2.64E+02	2.64E+02	1.18E-04	100	0.00E+00	0
	ϵMAgES	2.65E+02	2.65E+02	2.65E+02	2.65E+02	9.77E-01	100	0.00E+00	0
	COLSHADE	2.65E+02	2.65E+02	2.66E+02	2.65E+02	9.68E-01	100	0.00E+00	100
RC21	GM-MRFO	2.48E-01	2.65E-01	2.65E-01	2.76E-01	7.47E-03	100	0.00E+00	0
	ϵMAgES	2.17E+00	5.48E-01	1.50E+00	1.44E+00	7.32E-01	100	0.00E+00	0
	COLSHADE	2.08E+00	8.44E-01	1.31E+00	1.41E+00	1.11E+00	100	0.00E+00	100
RC22	GM-MRFO	5.74E-01	9.36E-01	1.05E+00	1.94E+00	3.73E-01	60	5.00E-02	0
	ϵMAgES	2.16E+00	2.09E+00	2.28E+00	2.97E+00	2.06E+00	100	0.00E+00	0
	COLSHADE	2.07E+00	2.83E+00	2.47E+00	3.01E+00	1.24E+00	100	0.00E+00	4
RC23	GM-MRFO	7.71E+00	1.19E+01	1.23E+01	2.06E+01	3.15E+00	0	7.52E+00	80
	ϵMAgES	8.70E+00	1.27E+01	1.28E+01	2.17E+01	4.34E+00	96	7.53E-06	0
	COLSHADE	8.97E+00	1.27E+01	1.33E+01	2.16E+01	4.20E+00	100	0.00E+00	0
RC24	GM-MRFO	4.43E+00	6.83E+00	6.78E+00	1.23E+01	1.76E+00	100	0.00E+00	0
	ϵMAgES	5.34E+00	7.88E+00	7.65E+00	1.37E+01	2.45E+00	100	0.00E+00	0
	COLSHADE	5.26E+00	7.89E+00	7.94E+00	1.28E+01	2.25E+00	100	0.00E+00	0
RC25	GM-MRFO	1.85E+03	2.21E+03	2.53E+03	5.94E+03	8.91E+02	100	0.00E+00	0
	ϵMAgES	1.85E+03	2.21E+03	2.53E+03	5.94E+03	8.91E+02	100	0.00E+00	0
	COLSHADE	1.85E+03	2.21E+03	2.53E+03	5.94E+03	8.92E+02	100	0.00E+00	92
RC26	GM-MRFO	4.84E+01	2.19E+02	2.03E+02	4.31E+02	9.99E+01	0	3.10E+00	0
	ϵMAgES	4.91E+01	2.20E+02	2.05E+02	4.31E+02	1.01E+02	0	2.56E+01	0
	COLSHADE	4.87E+01	2.20E+02	2.05E+02	4.31E+02	1.01E+02	100	0.00E+00	20
RC27	GM-MRFO	5.38E+02	5.62E+02	5.62E+02	5.79E+02	1.04E+01	100	0.00E+00	0
	ϵMAgES	5.39E+02	5.63E+02	5.63E+02	5.81E+02	1.16E+01	100	0.00E+00	0
	COLSHADE	5.38E+02	5.62E+02	5.62E+02	5.80E+02	1.11E+01	100	0.00E+00	100
RC28	GM-MRFO	1.85E+04	2.96E+04	2.98E+04	4.68E+04	7.37E+03	76	8.26E-03	0
	ϵMAgES	1.85E+04	2.96E+04	2.98E+04	4.68E+04	7.37E+03	100	0.00E+00	0
	COLSHADE	1.85E+04	2.96E+04	2.98E+04	4.68E+04	7.37E+03	100	0.00E+00	0
RC29	GM-MRFO	2.98E+06	3.20E+06	3.21E+06	3.61E+06	1.64E+05	100	0.00E+00	0
	ϵMAgES	2.98E+06	3.20E+06	3.21E+06	3.61E+06	1.64E+05	100	0.00E+00	0
	COLSHADE	2.98E+06	3.20E+06	3.21E+06	3.61E+06	1.64E+05	100	0.00E+00	100

Table 4. Continued
The outcomes of solving the of mechanical engineering problems (RC15-RC33) with GM-MRFO, ϵMAgES, and COLSHADE

Function	Algorithm	Best	Median	Mean	Worst	Std	FR	MV	SR
RC30	GM-MRFO	2.63E+00	2.86E+00	2.86E+00	3.19E+00	1.70E-01	100	0.00E+00	0
	ϵMAgES	2.69E+00	4.31E+00	3.66E+00	4.29E+00	8.67E-01	100	0.00E+00	0
	COLSHADE	3.09E+00	4.44E+00	4.02E+00	4.74E+00	8.91E-01	100	0.00E+00	0
RC31	GM-MRFO	2.36E-12	1.03E-09	3.80E-09	1.96E-08	5.63E-09	100	0.00E+00	100
	ϵMAgES	5.81E-01	6.67E-01	6.42E-01	8.03E-01	1.53E+00	100	0.00E+00	0
	COLSHADE	1.16E+00	6.05E-01	1.21E+00	1.25E-01	1.32E+00	100	0.00E+00	100
RC32	GM-MRFO	-3.06E+04	-3.05E+04	-3.05E+04	-3.03E+04	8.57E+01	100	0.00E+00	0
	ϵMAgES	-3.06E+04	-3.05E+04	-3.05E+04	-3.03E+04	8.59E+01	100	0.00E+00	0
	COLSHADE	-3.06E+04	-3.05E+04	-3.05E+04	-3.03E+04	8.63E+01	100	0.00E+00	100
	GM-MRFO	2.66E+00	3.15E+00	3.14E+00	4.06E+00	3.75E-01	100	0.00E+00	0
RC33	ϵMAgES	4.31E+00	3.95E+00	3.91E+00	5.80E+00	1.28E+00	100	0.00E+00	0
	COLSHADE	4.40E+00	3.36E+00	3.95E+00	5.83E+00	7.74E-01	100	0.00E+00	100

Table 5 provides an overview of the performance of the competing algorithms in solving Power System Problems (RC34 -RC44). In terms of best accuracy, GM-MRFO demonstrated the best performance among the three algorithms for all functions in the RC34-RC44 range. However, ϵMAgES and COLSHADE did not show good accuracy for any of the functions in the range. Furthermore, the feasibility rate (FR) test revealed that GM-MRFO produced a 100% success rate for all functions in the RC34-RC44 range. In contrast, ϵMAgES and COLSHADE only showed a 100% success rate for function RC44, with 0% success rate for the remaining functions. Regarding the mean of violation (MV) test, GM-MRFO performed well with a score of 0.00E+00 for all functions in the RC34-RC44 range. ϵMAgES also showed good performance with a score of 0.00E+00 for function RC44, while COLSHADE scored 0.00E+00 for function RC44 only. Finally, the success rate (SR) test revealed that GM-MRFO had a 100% success rate for functions RC36, RC37, RC39, RC40, RC43, and RC44, with a lower success rate for the remaining functions in the RC34-RC44 range. ϵMAgES and COLSHADE, on the other hand, showed no success rate for any of the functions in the range except for ϵMAgES, which had a 100% success rate for function RC44.

Table 5
The outcomes of solving the of power system problems (RC34-RC44) with GM-MRFO, ϵMAgES, and COLSHADE

Function	Algorithm	Best	Median	Mean	Worst	Std	FR	MV	SR
RC34	GM-MRFO	7.95E-02	3.38E-01	3.27E-01	5.42E-01	1.19E-01	100	0.00E+00	0
	$FCHA$	1.40E+00	1.41E+00	1.35E+00	1.26E+00	1.05E+00	0	6.78E+01	0
	COLSHADE	1.12E+00	1.05E+00	1.92E+00	1.35E+00	1.68E-01	0	6.07E-03	0
RC35	GM-MRFO	8.23E-02	8.44E-02	8.44E-02	8.70E-02	1.34E-03	100	0.00E+00	100
	ϵMAgES	1.46E+00	1.96E+00	1.20E+00	1.04E+00	1.59E+00	0	2.99E+03	0
	COLSHADE	1.57E+00	1.82E+00	1.41E+00	7.56E-01	1.75E+00	0	1.36E-01	0
RC36	GM-MRFO	5.65E-02	7.54E-02	7.75E-02	9.79E-02	9.27E-03	100	0.00E+00	16
	ϵMAgES	8.46E-01	5.96E-01	1.53E+00	1.78E-01	1.82E+00	0	2.96E+03	0
	COLSHADE	1.00E+00	4.51E-01	1.67E+00	2.56E-01	1.05E+00	0	1.56E-01	0
RC37	GM-MRFO	3.88E-02	6.58E-02	6.64E-02	1.01E-01	1.58E-02	100	0.00E+00	0
	ϵMAgES	1.47E+00	1.61E-01	8.79E-01	1.27E+00	5.33E-01	0	1.35E+02	0
	COLSHADE	1.18E+00	3.36E-01	1.09E+00	8.95E-01	5.27E-01	0	1.84E-02	0
RC38	GM-MRFO	3.22E+00	3.74E+00	3.69E+00	4.00E+00	2.32E-01	100	0.00E+00	0
	ϵMAgES	4.43E+00	4.87E+00	4.49E+00	4.51E+00	7.02E-01	0	1.31E+02	0
	COLSHADE	4.61E+00	4.82E+00	4.65E+00	4.74E+00	1.15E+00	0	1.63E-02	0
RC39	GM-MRFO	3.05E+00	3.81E+00	3.77E+00	4.46E+00	3.25E-01	100	0.00E+00	0
	ϵMAgES	4.82E+00	4.28E+00	4.42E+00	5.23E+00	5.95E-01	0	1.32E+02	0

	COLSHADE	4.15E+00	4.01E+00	4.56E+00	5.43E+00	1.35E+00	0	1.66E-02	0
RC40	GM-MRFO	1.12E-18	4.07E-18	9.37E-18	7.51E-17	1.47E-17	100	0.00E+00	100
	ϵMAgES	1.64E+00	2.38E-01	6.77E-01	1.04E+00	3.38E-01	0	6.81E+02	0
	COLSHADE	1.01E+00	8.90E-01	5.72E-01	1.80E+00	3.47E-01	0	9.20E-01	0
RC41	GM-MRFO	1.79E-21	1.92E-20	2.64E-20	7.83E-20	1.96E-20	100	0.00E+00	100
	ϵMAgES	9.78E-01	1.08E+00	5.23E-01	9.71E-01	5.78E-01	0	1.41E+03	0
	COLSHADE	8.44E-01	1.05E+00	8.75E-01	1.35E+00	7.45E-01	0	6.39E-01	0
RC42	GM-MRFO	8.99E-02	9.47E-02	9.44E-02	1.02E-01	3.09E-03	100	0.00E+00	0
	ϵMAgES	1.94E+00	7.98E-01	1.20E+00	8.52E-01	2.48E-01	0	4.97E+03	0
	COLSHADE	1.93E+00	8.56E-01	9.47E-01	1.50E+00	1.07E+00	0	1.03E+00	0
RC43	GM-MRFO	8.26E-02	8.68E-02	8.68E-02	9.35E-02	3.36E-03	100	0.00E+00	0
	ϵMAgES	1.82E+00	1.38E+00	9.16E-01	1.27E+00	1.39E+00	0	4.99E+03	0
	COLSHADE	1.09E+00	1.02E+00	1.56E+00	1.28E+00	1.52E+00	0	1.04E+00	0
RC44	GM-MRFO	-5.30E+03	-5.11E+03	-5.11E+03	-4.99E+03	7.50E+01	100	0.00E+00	0
	ϵMAgES	-5.30E+03	-5.11E+03	-5.11E+03	-4.99E+03	7.55E+01	100	0.00E+00	0
	COLSHADE	-5.30E+03	-5.11E+03	-5.11E+03	-4.99E+03	7.58E+01	100	0.00E+00	0

Table 6 presents the results of the best accuracy, feasibility rate, mean of violation, and success rate for six constrained optimization functions of Power Electronic Problems (RC45-RC50). According to the table, GM-MRFO outperformed the other two algorithms in terms of best accuracy for all functions RC45-RC50, while ϵMAgES and COLSHADE had poor performance in this regard. In terms of the feasibility rate test, GM-MRFO achieved 100% feasibility for all functions RC45-RC50, while ϵMAgES and COLSHADE had low feasibility rates for some functions. Specifically, ϵMAgES produced a feasibility rate of 0% for most functions and only achieved 4% for RC48 and RC50. On the other hand, COLSHADE achieved 96-100% feasibility rates for all functions except RC50, where it had a feasibility rate of 100%. Regarding the mean of violation, GM-MRFO and ϵMAgES performed well for different functions. GM-MRFO achieved a mean of violation of 0.00E+00 for all functions RC45-RC50, while ϵMAgES had a similar performance for NO function. COLSHADE, on the other hand, had a good performance for functions RC45-RC49 with a mean of violation of 0.00E+00. Finally, in terms of the success rate, GM-MRFO and ϵMAgES had no success in solving any of the functions RC45-RC50, while COLSHADE achieved some success rates for different functions. Specifically, COLSHADE had success rates of 16-36% for functions RC45-RC49, while it had no success in solving RC50.

Table 6
The outcomes of solving the of power electronic problems (RC45-RC50) with GM-MRFO, ϵMAgES, and COLSHADE

Function	Algorithm	Best	Median	Mean	Worst	Std	FR	MV	SR
RC45	GM-MRFO	3.80E-01	4.52E-01	4.57E-01	5.58E-01	4.00E-02	100	0.00E+00	0
	ϵMAgES	1.51E+00	1.70E+00	9.45E-01	1.47E+00	1.07E+00	4	1.45E+01	0
	COLSHADE	1.63E+00	2.04E+00	1.28E+00	9.48E-01	6.23E-01	100	0.00E+00	16
RC46	GM-MRFO	1.50E-01	1.74E-01	1.75E-01	2.02E-01	1.65E-02	100	0.00E+00	0
	ϵMAgES	1.34E+00	5.77E-01	8.97E-01	2.26E-01	9.40E-01	0	1.20E+01	0
	COLSHADE	1.63E+00	6.90E-01	3.54E-01	8.75E-01	5.94E-01	100	0.00E+00	36
RC47	GM-MRFO	1.11E-01	1.50E-01	1.55E-01	2.22E-01	3.41E-02	100	0.00E+00	0
	ϵMAgES	1.10E+00	5.49E-01	5.71E-01	8.14E-01	1.53E+00	4	1.38E+01	0
	COLSHADE	8.83E-01	9.11E-01	1.15E+00	6.89E-01	1.49E+00	100	0.00E+00	8
RC48	GM-MRFO	7.02E-02	9.35E-02	1.18E-01	2.16E-01	4.32E-02	100	0.00E+00	0
	ϵMAgES	7.47E-01	1.18E+00	1.47E+00	8.60E-01	8.12E-01	0	2.45E+01	0
	COLSHADE	1.09E+00	1.02E+00	1.28E+00	6.78E-01	1.00E+00	100	0.00E+00	0
RC49	GM-MRFO	4.79E-02	1.01E-01	9.28E-02	1.07E-01	1.74E-02	100	0.00E+00	0
	ϵMAgES	6.92E-01	6.07E-01	1.09E+00	7.98E-01	5.56E-01	0	2.95E+01	0
	COLSHADE	1.00E+00	3.12E-01	2.28E-01	1.60E+00	4.06E-01	100	0.00E+00	0
RC50	GM-MRFO	5.96E-02	7.22E-02	7.37E-02	9.07E-02	6.65E-03	100	0.00E+00	0

	ϵMAgES	6.88E-01	1.67E+00	8.67E-01	7.14E-01	8.97E-01	0	3.72E+01	0
	COLSHADE	6.35E-01	1.41E+00	1.52E+00	6.57E-01	1.07E+00	96	4.02E-05	0

Table 7 presents the results on functions of Livestock Feed Ration Optimization Problems (RC51-RC57). The table reveals that GM-MRFO achieved the best accuracy of solution for all functions RC51-RC57, while ϵMAgES and COLSHADE had poor performance for these functions. In terms of the FR test, GM-MRFO produced a 100% feasibility rate for all functions RC51-RC57, while ϵMAgES had low rates ranging from 0-12% and COLSHADE had a mix of high and low rates. For the MV, GM-MRFO and ϵMAgES both achieved excellent performance with a value of 0.00E+00 for RC57 and no function, respectively, while COLSHADE performed well for functions RC52, RC53, RC54, and RC57. Finally, in terms of the SR, GM-MRFO had the highest success rate for functions RC51, RC54, and RC57, while ϵMAgES and COLSHADE had no success rate for any of the functions. These results demonstrate the effectiveness of GM-MRFO in optimizing the functions RC51-RC57.

Table 7
The outcomes of solving the of livestock feed ration optimization problems (RC51 -RC57) with GM-MRFO, ϵMAgES, and COLSHADE

Function	Algorithm	Best	Median	Mean	Worst	Std	FR	MV	SR
RC51	GM-MRFO	3.73E+03	4.23E+03	4.34E+03	5.59E+03	4.48E+02	84	3.28E-01	84
	ϵMAgES	3.73E+03	4.23E+03	4.34E+03	5.59E+03	4.49E+02	0	7.43E-01	0
	COLSHADE	3.73E+03	4.23E+03	4.34E+03	5.59E+03	4.49E+02	0	2.82E-06	0
RC52	GM-MRFO	3.89E+03	5.83E+03	5.92E+03	9.31E+03	1.13E+03	0	2.28E-01	0
	ϵMAgES	3.90E+03	5.83E+03	5.92E+03	9.31E+03	1.13E+03	4	2.71E-01	0
	COLSHADE	3.90E+03	5.83E+03	5.92E+03	9.31E+03	1.13E+03	100	0.00E+00	0
RC53	GM-MRFO	4.54E+03	6.02E+03	6.09E+03	8.26E+03	8.58E+02	4	4.30E-01	4
	ϵMAgES	4.54E+03	6.02E+03	6.09E+03	8.26E+03	8.59E+02	0	8.29E-01	0
	COLSHADE	4.54E+03	6.02E+03	6.09E+03	8.26E+03	8.59E+02	100	0.00E+00	0
RC54	GM-MRFO	2.72E+03	3.01E+03	3.30E+03	1.05E+04	1.50E+03	0	2.69E-01	100
	ϵMAgES	2.72E+03	3.01E+03	3.31E+03	1.05E+04	1.50E+03	0	1.02E+00	0
	COLSHADE	2.72E+03	3.01E+03	3.31E+03	1.05E+04	1.50E+03	100	0.00E+00	0
RC55	GM-MRFO	6.72E+03	7.69E+03	7.71E+03	9.20E+03	5.88E+02	0	1.63E-02	0
	ϵMAgES	6.72E+03	7.69E+03	7.72E+03	9.20E+03	5.89E+02	4	3.60E-01	0
	COLSHADE	6.72E+03	7.69E+03	7.72E+03	9.20E+03	5.89E+02	84	1.81E-05	0
RC56	GM-MRFO	1.25E+04	1.45E+04	1.46E+04	1.68E+04	1.03E+03	0	2.26E-02	8
	ϵMAgES	1.25E+04	1.45E+04	1.46E+04	1.68E+04	1.04E+03	12	1.89E+00	0
	COLSHADE	1.25E+04	1.45E+04	1.46E+04	1.68E+04	1.04E+03	48	1.52E-04	0
RC57	GM-MRFO	5.59E+03	7.90E+03	7.96E+03	1.14E+04	1.65E+03	100	0.00E+00	0
	ϵMAgES	5.59E+03	7.90E+03	7.96E+03	1.14E+04	1.65E+03	8	5.95E+00	0
	COLSHADE	5.59E+03	7.90E+03	7.96E+03	1.14E+04	1.65E+03	100	0.00E+00	0

The Table 8 shows the performance comparison of three algorithms (GM-MRFO, COLSHADE, ϵMAgES) based on their scores. The scores were obtained by running the algorithms on the given functions and evaluating their solutions against a set of criteria. The table lists the scores obtained by each algorithm for Score 1, Score 2, and Score 3, along with their respective total score and rank. According to the table, GM-MRFO obtained the highest total score (0.6254) and rank (1), indicating its superior performance compared to the other algorithms. COLSHADE obtained the second-highest total score (0.6264) and rank (2), followed by ϵMAgES with a total score of NaN. It is worth noting that ϵMAgES obtained the lowest score (0.0219) for Score 1, while COLSHADE had the lowest score (0.0649) for Score 3. Overall, the results suggest that GM-MRFO and COLSHADE are the better-performing algorithms for the given functions, while ϵMAgES requires further investigation due to the missing data.

The concept of the No Free Lunch Theorem (NFL) in optimization is that it is stated that no single optimization algorithm is universally better than all others for all problems. It is further explained that there is no "silver bullet" algorithm that can solve every problem optimally, and that the effectiveness of an algorithm is problem-dependent, meaning that it depends on the specific problem being solved. In the paragraphs provided, the performance of three different optimization algorithms (GM-MRFO, ϵMAgES, and COLSHADE) was evaluated by the authors on various optimization problems with different performance metrics such as accuracy, feasibility rate, mean violation, and success rate. It was suggested by the evaluation results that GM-MRFO is the most effective algorithm among the three for solving the optimization problem for all functions RC01-RC07 and RC08-RC14, as well as for functions RC15-RC33.

Table 8
The outcomes of solving the of livestock feed ration optimization problems (RC51 RC57) with GM-MRFO, ϵMAgES, and COLSHADE

Algorithm	Score 1	Score 2	Score 3	Total Score	Rank
GM-MRFO	0.6313	0.6191	0.6201	0.6254	1
COLSHADE	0.5844	0.6764	0.6565	0.6264	2
ϵMAgES	0.0219	NaN	0.0649	NaN	3

5. Conclusions

In conclusion, the proposed method presented in the paper is a modified version of the MRFO algorithm that incorporates a modified Gradient-based mutation (GM) technique. It addresses the limitations of the MRFO algorithm in handling constraints and finding feasible solutions. The modified GM approach helps the algorithm to navigate the solution space and find feasible solutions that satisfy all constraints by modifying the search direction and step size based on the gradient of the objective function and the constraints. The GM-MRFO approach uses a deterministic and gradient-based searching method to improve both the fitness and feasibility of the solution. It also adopts an adjustment technique for the threshold of constraint violation value. Subsequently, the proposed method was tested on 57 real-world constrained problems from diverse fields. The performance of the algorithm is assessed based on the best produced accuracy, feasibility rate, mean of violation, success rate, as well as employing the proposed ranking scheme in CEC2020 competition. The result of the analysis shows that the proposed method outperforms the original MRFO algorithm and other state-of-the-art algorithms in terms of accuracy, feasibility rate, and success rate. The limitation of the proposed method is that it may search for solutions outside the boundaries of the feasible region. This especially true for problems that are monotonic in at least one axial direction, leading to violation of constraints which is considered as the future work.

Acknowledgement
The authors would like to thank the Ministry of Higher Education for providing financial support under Fundamental Research Grant Scheme (FRGS) No. FRGS/1/2021/ICT02/UMP/02/2 (University reference RDU210110) and FRGS/1/2021/ICT02/UMP/03/2 (University reference RDU210116) via Research and Innovation Department, Universiti Malaysia Pahang Al-Sultan Abdullah.

References
[1] Shatnawi, Hashem, and Mohammad N. Alqahtani. "Delving into the revolutionary impact of artificial intelligence on mechanical systems: A review." *Semarak International Journal of Machine Learning* 1, no. 1 (2024): 31-40. https://doi.org/10.37934/sijml.1.1.3140

[2] Kamarudin, Nurzatulshima, Nik Mawar Hanifah Nik Hassan, Mohd Mokhtar Muhamad, Othman Talib, Haryati Kamarudin, Norhafizan Abdul Wahab, Aidatul Shima Ismail, Haza Hafeez Borhan, and Nazihah Idris. "Unveiling collaborative trends in Fuzzy Delphi Method (FDM) research: A co-authorship bibliometrics study." *International Journal of Computational Thinking and Data Science* 2, no. 1 (2024): 1-20. https://doi.org/10.37934/CTDS.2.1.120

[3] Noor, Norlenda Mohd, Zuraida Alwadood, Nuramelissa Ahmad Murad, and Nur Maisarah Mohamad Termizi. "Optimization of private bus scheduling in UITM Shah Alam using integer linear programming." *Journal of Advanced Research in Computing and Applications* 15, no. 1 (2019): 26-34.

[4] Roslan, Shairatul Akma, Fitri Yakub, Shuib Rambat, Sharifah Munawwarah, Mokhtar Saidin, Farah Liana, Nurshafinaz Mohd Maruai, Mohamed Sukri Mat Ali, and Ahmad Faiz Mohammad. "The novel method in validating the spectral wavelength optimization to determine archaeological proxies by the integration of aerial and ground platforms." *Journal of Advanced Research in Applied Mechanics* 108, no. 1 (2023): 1-15. https://doi.org/10.37934/aram.108.1.115

[5] Musa, Zulkifli, Zuwairie Ibrahim, Mohd Ibrahim Shapiai, and Yusei Tsuboi. "Cubature Kalman optimizer: A novel metaheuristic algorithm for solving numerical optimization problems." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 33, no. 1 (2023): 333-355. https://doi.org/10.37934/araset.33.1.333355

[6] Mohanaprakash, Thotipalayam Andavan, Madhumitha Kulandaivel, Samuel Rosaline, Pasham Nithish Reddy, Shankar Nayak Bhukya, Ravindra Namdeorao Jogekar, and Rengaraj Gurumoorthy Vidhya. "Detection of brain cancer through enhanced particle swarm optimization in artificial intelligence approach." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 33, no. 2 (2023): 174-186. https://doi.org/10.37934/araset.33.2.174186

[7] Oleolo, Ibrahim, Hayati Abdullah, Ismail Mustapha, Maziah Mohamad, Mohammad Nazri Mohd Jaafar, Akeem Olowolayemo, and Sapiah Sulaiman. "Long short-term memory neural network model for the control of temperature in a multi-circuit air conditioning system." *CFD Letters* 14, no. 12 (2022): 84-98. https://doi.org/10.37934/cfdl.14.12.8498

[8] Doolaard, Floris, and Neil Yorke-Smith. "Online learning of variable ordering heuristics for constraint optimisation problems." *Annals of Mathematics and Artificial Intelligence* (2022): 1-30. https://doi.org/10.1007/s10472-022-09816-z

[9] Chootinan, Piya, and Anthony Chen. "Constraint handling in genetic algorithms using a gradient-based repair method." *Computers & operations research* 33, no. 8 (2006): 2263-2281. https://doi.org/10.1016/j.cor.2005.02.002

[10] Star, Susan Leigh, and Karen Ruhleder. "Steps towards an ecology of infrastructure: complex problems in design and access for large-scale collaborative systems." In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, p. 253-264. 1994. https://doi.org/10.1145/192844.193021

[11] Kumar, Abhishek, Guohua Wu, Mostafa Z. Ali, Rammohan Mallipeddi, Ponnuthurai Nagaratnam Suganthan, and Swagatam Das. "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results." *Swarm and Evolutionary Computation* 56 (2020): 100693. https://doi.org/10.1016/j.swevo.2020.100693

[12] Zhao, Weiguo, Zhenxing Zhang, and Liying Wang. "Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications." *Engineering Applications of Artificial Intelligence* 87 (2020): 103300. https://doi.org/10.1016/j.engappai.2019.103300

[13] Ma, Yunlong, Qiaogao Huang, Guang Pan, and Pengcheng Gao. "Investigation of the hydrodynamic characteristics of two manta rays tandem gliding." *Journal of Marine Science and Engineering* 10, no. 9 (2022): 1186. https://doi.org/10.3390/jmse10091186

[14] Abdul Razak, Ahmad Azwan, Ahmad Nor Kasruddin Nasir, Nor Maniha Abdul Ghani, and Mohd Falfazli Mat Jusof. "Manta ray foraging optimization with quasi-reflected opposition strategy for global optimization." In *Proceedings of the 6th International Conference on Electrical, Control and Computer Engineering: InECCE2021, Kuantan, Pahang, Malaysia, 23rd August*, pp. 477-485. Singapore: Springer Singapore, 2022. https://doi.org/10.1007/978-981-16-8690-0_43

[15] Abdul Razak, Ahmad Azwan, Ahmad Nor Kasruddin Nasir, Nor Maniha Abdul Ghani, Shuhairie Mohammad, Mohd Falfazli Mat Jusof, and Nurul Amira Mhd Rizal. "Non-dominated sorting manta ray foraging algorithm with an application to optimize PD control." In *Recent Trends in Mechatronics Towards Industry 4.0: Selected Articles from iM3F 2020, Malaysia*, pp. 463-474. Springer Singapore, 2022. https://doi.org/10.1007/978-981-33-4597-3_42

[16] Zhu, Fang, Wenhao Wang, and Shan Li. "Application of improved manta ray foraging optimization algorithm in coverage optimization of wireless sensor networks." *Computational Intelligence and Neuroscience* 2022, no. 1 (2022): 3082933. https://doi.org/10.1155/2022/3082933

[17] bin Abdul Razak, Ahmad Azwan, Ahmad Nor Kasruddin bin Nasir, Nor Maniha Abdul Ghani, Shuhairie Mohammad, Mohd Falfazli Mat Jusof, and Nurul Amira Mhd Rizal. "Hybrid genetic manta ray foraging optimization and its

application to interval type 2 fuzzy logic control of an inverted pendulum system." In *IOP Conference series: materials science and engineering*, vol. 917, no. 1, p. 012082. IOP Publishing, 2020. https://doi.org/10.1088/1757-899X/917/1/012082

[18] Shukla, Pradyumn Kumar. "On gradient based local search methods in unconstrained evolutionary multi-objective optimization." In *International Conference on Evolutionary Multi-Criterion Optimization*, p. 96-110. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. https://doi.org/10.1007/978-3-540-70928-2_11

[19] Shukla, Pradyumn Kumar. "Gradient based stochastic mutation operators in evolutionary multi-objective optimization." In *International Conference on Adaptive and Natural Computing Algorithms*, p. 58-66. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. https://doi.org/10.1007/978-3-540-71618-1_7

[20] Takahama, Tetsuyuki, and Setsuko Sakai. "Solving difficult constrained optimization problems by the ε constrained differential evolution with gradient-based mutation." *Constraint-Handling in Evolutionary Optimization* (2009): 51-72. https://doi.org/10.1007/978-3-642-00619-7_3

[21] Takahama, Tetsuyuki, and Setsuko Sakai. "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation." In *IEEE congress on evolutionary computation*, pp. 1-9. IEEE, 2010. https://doi.org/10.1109/CEC.2010.5586484

[22] Takahama, Tetsuyuki, and Setsuko Sakai. "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation." In *IEEE congress on evolutionary computation*, p. 1-9. IEEE, 2010. https://doi.org/10.1109/CEC.2010.5586484

[23] Zhang, Qing, Sanyou Zeng, Rui Wang, Hui Shi, Guang Chen, Lixin Ding, and Lishan Kang. "Constrained optimization by the evolutionary algorithm with lower dimensional crossover and gradient-based mutation." In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, p. 273-279. IEEE, 2008. https://doi.org/10.1109/CEC.2008.4630810

[24] Wessing, Simon. "Repair methods for box constraints revisited." In *Applications of Evolutionary Computation: 16th European Conference, EvoApplications 2013, Vienna, Austria, April 3-5, 2013. Proceedings 16*, pp. 469-478. Springer Berlin Heidelberg, 2013. https://doi.org/10.1007/978-3-642-37192-9_47

[25] Kundu, Souvik, Swagatam Das, Athanasios V. Vasilakos, and Subhodip Biswas. "A modified differential evolution-based combined routing and sleep scheduling scheme for lifetime maximization of wireless sensor networks." *Soft Computing* 19 (2015): 637-659. https://doi.org/10.1007/s00500-014-1286-9

[26] Hellwig, Michael, and Hans-Georg Beyer. "A matrix adaptation evolution strategy for constrained real-parameter optimization." In *2018 IEEE congress on evolutionary computation (CEC)*, pp. 1-8. IEEE, 2018. https://doi.org/10.1109/CEC.2018.8477950