processes

MDPI

Article

An Improved Gradient-Based Optimization Algorithm for Solving Complex Optimization Problems

Saleh Masoud Abdallah Altbawi [1,*], Saifulnizam Bin Abdul Khalid [1], Ahmad Safawi Bin Mokhtar [1], Hussain Shareef [2], Nusrat Husain [3], Ashraf Yahya [3], Syed Aqeel Haider [4], Lubna Moin [3] and Rayan Hamza Alsisi [5]

1 Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Johor, Malaysia
2 Department of Electrical Engineering, United Arab Emirates University, Al Ain P.O. Box 15551, United Arab Emirates
3 Department of Electronics & Power Engineering, Pakistan Navy Engineering College, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
4 Department of Computer & Information Systems Engineering, Faculty of Electrical & Computer Engineering, NED University of Engineering and Technology, Karachi 75270, Pakistan
5 Department of Electrical Engineering, Faculty of Engineering, Islamic University of Madinah, Madinah 41411, Saudi Arabia
* Correspondence: masoud@graduate.utm.my

Abstract: In this paper, an improved gradient-based optimizer (IGBO) is proposed with the target of improving the performance and accuracy of the algorithm for solving complex optimization and engineering problems. The proposed IGBO has the added features of adjusting the best solution by adding inertia weight, fast convergence rate with modified parameters, as well as avoiding the local optima using a novel functional operator (G). These features make it feasible for solving the majority of the nonlinear optimization problems which is quite hard to achieve with the original version of GBO. The effectiveness and scalability of IGBO are evaluated using well-known benchmark functions. Moreover, the performance of the proposed algorithm is statistically analyzed using ANOVA analysis, and Holm–Bonferroni test. In addition, IGBO was assessed by solving well-known real-world problems. The results of benchmark functions show that the IGBO is very competitive, and superior compared to its competitors in finding the optimal solutions with high convergence and coverage. The results of the studied real optimization problems prove the superiority of the proposed algorithm in solving real optimization problems with difficult and indefinite search domains.

Keywords: gradient-based optimizer; improve gradient-based optimizer; metaheuristic; inertia; operator; engineering optimization problems

Citation: Altbawi, S.M.A.; Khalid, S.B.A.; Mokhtar, A.S.B.; Shareef, H.; Husain, N.; Yahya, A.; Haider, S.A.; Moin, L.; Alsisi, R.H. An Improved Gradient-Based Optimization Algorithm for Solving Complex Optimization Problems. *Processes* **2023**, *11*, 498. https://doi.org/10.3390/pr11020498

Academic Editor: Olympia Roeva

Received: 23 December 2022
Revised: 28 January 2023
Accepted: 3 February 2023
Published: 7 February 2023

1. Introduction

In recent years, information technology has had a deep impact on human civilization [1]. Due to this advancement, a massive amount of data needs to be analyzed, more complicated real-world problems need to be solved, and enhancement of the computing efficiency of computers is needed [2]. Artificial Intelligence (AI) has been a persistently hot topic to deal with this development. AI refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions [3]. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. AI methods in MG strategies contain Reasoning and Learning (RL) and Swarm Intelligence (SI) methods [4]. The SI-based algorithms have attained significant popularity among researchers and are considered one of the highest encouraging categories of AI especially metaheuristic algorithms [5]. In general, metaheuristic optimization algorithms try to imitate the physical, biological, or even chemical procedures that take place in the environment. However, some algorithms depend on

mathematical theories [6]. The most generic ones are: Genetic Algorithm, which simulates Darwin's theory of evolution [7], Simulated Annealing (SA) algorithm, that is developed from the thermodynamic process [8], Particle swarm optimization (PSO) algorithms, which simulate the behaviour of fish school or bird flock [9], Differential Evolution (DE) algorithm, that is applied in solving problems and functions by iteratively improving a candidate solution based on an evolutionary process [10], Teaching Learning-Based Optimization (TLBO), which is based on a teaching-learning process [11], Jaya algorithm, that based on the concept that tries to reach the best solution and tries to avoid failure to move away from the worst solution [12], Cuckoo Search (CS) Algorithm, which imitates the brood parasitism conduct of some cuckoo species in conjunction with the Levy flight conduct of certain birds and fruit flies [13], flower pollination algorithm (FPA) that brings its metaphor from the pollination in the flowering cycle of some plants in nature [14]. African Vultures Optimization Algorithm (AVOA) which imitates the nature of African vultures in foraging and navigation [15].

Gradient-based optimization (GBO) is a newly developed metaheuristic optimization to solve optimization problems. It contains search directions defined by the gradient of the function at the current point. GBO algorithm is motivated by Newton's gradient procedure including two principal processes: gradient search rule process and local escaping operator. The gradient-based approach uses the gradient search rule to improve exploring phenomena and quickens the convergent rate of GBO to obtain the optimal position within the search space. However, the local escaping process prevents GBO to avoid getting stuck into the local optima [16].

In swarm-based algorithms, inertia weight is a concept utilized to balance the influence of the current position and the attraction to the best-known position in the search space. This helps the algorithm avoid getting trapped in local optima and to explore the search space more effectively. The value of inertia weight is typically decreased over time to increase the exploration [17].

In general, the optimization algorithm procedure steps, consist of parameter selection, variables (search agents), initialization, exploration, exploitation, randomization formula of the step search, selection of step, and terminating condition [18]. Each search agent in the population interacts with other agents to locate the optimal solution [19]. Generally, the swarm-based algorithms require some common control parameters like population size and the number of iterations [20]. In addition, some algorithms have specific control parameters besides the general parameters, known as hyper-parameters. These parameters exist to improve the performance of the algorithm by tuning their values properly [21].

The search agents in a SI system are designed to have simple rules. There is no central control to give order to how individual agents should perform [22]. The agent's real performance is local, with a degree of arbitrary. However, the relations between such agents and the other parameters in the algorithm take the edge of the occurrence of "intelligent" to mimic the global behavior, while the individual agents cannot find alone [23].

The major contributions of this paper are listed as follows:

- Utilizing modified inertia weight in the original version of GBO to adjust the accuracy of the best solution. Whereas, the inertia weight in optimization algorithm, gives more weight to previous solutions in order to converge faster but also allows for exploration of new solutions.
- Modified parameters are utilized in GBO to boost the convergence speed and provide the proper balance of global and local search capabilities.
- A novel operator (G) is introduced which supports the diversity search in the search space. Whereas, G applied to move search agents toward better solutions leading to suitable performance both in the global search and local search using new developed formula.

The prove the superiority of the proposed IGBO, its performance is compared with GBO, CS, DE, FPA, PSO, TLBO, and AVOA using a wide range of benchmark functions on a few real-world problems.

The rest of the article is organized as follows: In Section 2 background and related works are presented. Section 3 presents a brief description of the GBO while the proposed IGBO is described in Section 4. Section 5 describes the benchmark function and Section 6 explains the real-world problems. The obtained results and performance comparison using benchmark functions against the different optimization algorithms are presented in Section 7. In Section 8 the proposed IGBO is employed to solve a challenging real-world optimization problem in the field of engineering. Finally, Section 9 summarizes the study.

2. Background and Related Works

The improvement of the optimization algorithm depends on the enhancement of the procedure steps or advancements of hybrid algorithms [24]. Every year there is competition in the algorithms on benchmarks function and real-world problems based on time, accuracy, and result to decide which algorithm is efficient [25]. There are different approaches to improving algorithms such as parameters tuning, opposite strategy, inertia weight, chaos strategy, fuzzy logic strategy, adding new operators, or multi-objective theory [26].

Adding inertia weight to adjust the accuracy and convergence speed toward the optimal solution [27]. There are different methods to implement inertia such as fixed and arbitrary inertia weights, and Adaptive inertia weights [28]. Tuning parameters to enhance convergence rate by many approaches. For example, with fixed values that are appropriate to the search process, some approaches gradually change the parameters of operators through the problem search process while, in some approaches, the mechanism will update the parameter for some instances of problems. Some of these methods aim to develop an adaptive mechanism to change the parameter value according to the search process [29]. Adding extra operators to balance exploration and exploitation and enhance diversity. There are different kinds of operators such as comparison/relational operators, stream operators, subscript operators, function operators, and arithmetic operators [30].

Because of the robustness of GBO, it is applied to solve quite complex real-world problems. The authors in [31] used GBO to find the optimal design automatic voltage regulator (AVR) using the GBO algorithm. In [32], GBO is used to calculate the reliability redundancy allocation problem of a series-parallel framework. In [33], GBO is employed in the estimation of the parameters of solar cells and photovoltaic (PV) panels as an effective and precise method. GBO in [34] is utilized in the calculation of the Economic Load Dispatch (ELD) problem for different situations such as ELD with transmission losses, and mixed economic and emission dispatch. In [35], GBO was utilized with Proton Exchange Membrane Fuel Cell to estimate the optimal parameters of three distinct kinds of PEM fuel cells. The scholars in [36] used an ensemble random vector functional link model (ERVFL) incorporated with GBO to model the ultrasonic welding of a polymeric material blend. ERVFL-GBO has the best outcome which indicates its high accuracy over other tested methods.

Despite the effective performance, GBO is trapped in local solution when conducting complicated non-linear functions, thus, it can decrease its accuracy. To overcome these drawbacks, different variants of the GBO have been introduced. The researchers in [37] used a multi-objective GBO algorithm-based Weighted multi-view Clustering (MO-GBO-WMV) to find the consensus clustering among different partitioning generated from individual views and compared this approach with some other methods to demonstrate the advanced ability of this approach. In [38] introduced a multi-objective gradient-based optimizer (MO-GBO) to handle the best solution for more than one objective, where needed. In [39], by improving the GBO using C_a and C_b which are new chaotic numbers generated by various chaos maps. Then IGBO is used to derive the parameters of PV modules. Similarly, in [40] a novel random learning mechanism is designed to improve the performance of GBO. After that, IGBO was utilized to extract parameters of four photovoltaic models. In [41], an improved gradient-based optimizer denoted by (CL-GBO) to build DNA coding sets that contain the Cauchy and Levy mutation operators, which are utilized as readers and addresses of the libraries. In [42] the goal was to solve single and multi-Economic Emission

Dispatch problems, using an elegant method depending on mix-object both Manta ray foraging optimization (MRFO) with GBO, named (MRFO–GBO) to avoid trapped into local optima as well as accelerate the solution process. In [43] enhance the performance of Grey Wolf Optimizer (GWO) by a new procedure to use GBO to produce a new algorithm called (G-GWO). This combination was applied to improve the algorithm's exploitation and exploration and also added Gaussian walk and Levy flight. These two are arbitrary operators utilized to increase the diversity search of exploration in the G-GWO. In [44] modified GBO algorithm using operator D to improve the stability between exploration and exploitation phases in the search process. Similarly, in [45] used binary search, which is an advanced type of search algorithm that finds and fetches data from a sorted list of items. It is called the binary GBO (B-GBO) algorithm. Then B-GBO is used in feature selecting problems of machine learning and data mining.

Optimization algorithms have gained a surge in popularity and have achieved significant attention from both academic and industrial fields. Here the review some recent optimization algorithms. Arithmetic Optimization Algorithm (AOA) has developed depending on well-known Arithmetic theory [46]. In [47], the scholars introduced Aquila Optimizer (AO), which mathematically models and mimics the nature of aquila during the procedure of hunting the prey. Dwarf Mongoose Optimization (DMO) algorithm is introduced in [48] which simulate the teamwork behaviors of the dwarf mongoose. The authors in [49] have developed Ebola Optimization search Algorithm (EOSA) based on the propagation behavior of the Ebola virus disease. Gazelle Optimization Algorithm (GOA) have presented in [50], which is inspired by the survival ability in their predator-dominated environment. The authors in [51] have developed prairie dog optimization (PDO) algorithm, which mimic the behaviour of four prairie dog in foraging and burrow building. Reptile Search Algorithm (RSA) has presented in [52], which is inspired by the nature of Crocodiles in hunting. The authors in [53], introduced oppositional unified particle swarm gradient-based optimizer based on mix of oppositional learning, unified particle swarm algorithm, and GBO algorithm to solve the complex inverse analysis of structural damage problems. In [54], the scholars have developed social engineering particle swarm optimization algorithm, which consists of combination of social engineering optimizer and particle swarm optimization to deal with structural health monitoring as objective function. However, this combination of more than algorithm may lead to slower convergence and inaccuracy in some optimization problems where is the speed is compulsory.

Summarizing the previous studies, the GBO algorithm has superiority over all modern counterparts in solving problems in different fields. However, ordinary GBO has some limitations such as:

GBO will still be trapped into local optima and suffer from the imbalance between exploitation and exploration, premature convergence, and slow convergence speed under some circumstances due to incomplete judgment standard and operators.

The main function of the local escaping operator (LEO) phase algorithm is to avoid the occurrence of local optimal stagnation, but only when the random number is less than 0.5, it will enter the LEO phase.

The former GBO does not identify the optimal solution for discrete functions which have discrete search spaces and decision variables such as feature selection problems.

There is only one guidance towards the best solution during the updating process, which limits the exploitation capability and can lead to the propensity of falling into the local optimal solution.

Former GBO does not have enough internal memory to save optimal solutions among all generations, which leads to a lack the population diversity. Moreover, the performance of the algorithm is affected significantly by the space domain of the objective function. However, an intensive searching process may lead to the deterioration of multimodal objective functions.

3. Original Gradient-Based Optimizer Algorithm

GBO was inspired by Newton's gradient-based method. It combines concepts from gradient-based and population methods to find solutions. GBO uses two principles, the gradient search rule (GSR) and the local escaping operator (LEO). The GSR improves the exploration tendency and accelerates the convergence rate in the search space, while the LEO allows GBO to escape from local optima [16].

Metaheuristic algorithms such as GBO usually generate several stochastic operators without the demand to drop data and depend on steady performance. GBO mainly uses two operators and a set of vectors to explore the entire search space [40].

3.1. Initialization

Every individual in the whole population is denoted as a vector for a simple implementation description, which can be defined as:

$$X_{n,d} = [X_{n,1}, X_{n,2}, \ldots, X_{n,d}],\ n = 1, 2, \ldots, N;\ d = 1, 2, \ldots, D \tag{1}$$

Therefore, the GBO size population is denoted by N with the D-variables.

So, the vector is created randomly and represented as below:

$$x_n = X_{min} + \text{rand}(0, 1) \times (X_{max} - X_{min}) \tag{2}$$

where X_{min}, and X_{max} are the lower and upper limits the variables X, and rand (0, 1) is an arbitrary number in the boundary [0, 1].

3.2. Gradient Search Rule

The GSR operator is utilized in the gradient-based technique to enhance the exploration feasibility and enhance the convergence speed to find the optimal solution in the search domain. GSR expression is represented as

$$GSR = \text{norm} \times \rho_1 \times \frac{2\Delta x \times X_n}{(x_{worst} - x_{best} + \varepsilon)} \tag{3}$$

where norm is a random value chosen within normal distribution, x_{worst} and x_{best} are the worst and optimal solution in the whole search progress of optimization, Δx is the value of increase, and ε is an arbitrary number chosen from the boundary (0 to 0.1).

To attain the balance between the exploration and exploitation phases and follow the search feasibility development, the GSR will be improved depending on the mathematical expression below:

$$\rho_1 = 2 \times rand \times \alpha - \alpha \tag{4}$$

$$\Delta x = \text{rand}(1 : N) \times |\text{step}| \tag{5}$$

$$step = \frac{(x_{best} - x_{r1}^m) + \delta}{2} \tag{6}$$

$$\delta = 2 \times \text{rand} \times \left(\left| \frac{x_{r1}^m + x_{r2}^m + x_{r3}^m + x_{r4}^m}{4} - x_n^m \right| \right) \tag{7}$$

$$\alpha = \left| \beta \times \sin\left(\frac{3\pi}{2} + \sin\left(\beta \times \frac{3\pi}{2} \right) \right) \right| \tag{8}$$

$$\beta = \beta_{min} + (\beta_{max} - \beta_{min}) \times \left(1 - \left(\frac{m}{M} \right)^3 \right)^2 \tag{9}$$

where rand$(1 : N)$ is an arbitrary value with N population, r1, r2, r3, and r4 (r1 $\neq$ r2 $\neq$ r3 $\neq$ r4 $\neq$ n) are integers random numbers chosen from [1, N]. *rand* is a random number between [0, -1], β_{min} = 0.2, β_{max} = 1.2, m is the value of the current iteration, M is the total iterations, and α is a function based on β.

The *step* is a mathematical equation, that signifies the step size, which is based on x_{best} and x_{r1}^{m}, while ρ_2 is represented by:

$$\rho_2 = 2 \times \text{rand} \times \alpha - \alpha \tag{10}$$

For more enhance of the exploitation of the surrounding area of x_n, by using the direction of movement (DM). The DM can be determined by:

$$\text{DM} = \text{rand} \times \rho_2 \times (x_{best} - x_n) \tag{11}$$

The GSR can be expressed as:

$$\text{GSR} = \text{rand} \times \rho_1 \times \frac{2\Delta x \times x_n}{(yp_n - yq_n + \varepsilon)}$$

In which

$$yp_n = \text{rand} \times \left(\frac{[z_{n+1} + x_n]}{2} + \text{rand} \times \Delta x \right) \tag{12}$$

$$yq_n = \text{rand} \times \left(\frac{[z_{n+1} + x_n]}{2} - \text{rand} \times \Delta x \right) \tag{13}$$

$$z_{n+1} = x_n - \text{rand} \times \left(\frac{2\Delta x \times x_n}{x_{worst} - x_{best} + \varepsilon} \right) \tag{14}$$

Concerning the GSR and DM, Equations. The position $X1_n^m$ at any iteration is calculated using GSR and DM as follows:

$$X1_n^m = x_n^m - \text{GSR} + \text{DM} \tag{15}$$

Full expression of $X1_n^m$ can be written as

$$X1_n^m = x_n^m - \text{norm} \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(yp_n^m - yq_n^m + \varepsilon)} + \text{rand} \times \rho_2 \times (x_{r1}^m - x_{r2}^m) \tag{16}$$

By substituting the position of the best vector (x_{best}) with the current vector (x_n^m) in Equation (1), the new vector ($X2_n^m$) can be calculated as follows:

$$X2_n^m = x_{best} - \text{randn} \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(yp_n^m - yq_n^m + \varepsilon)} + \text{rand} \times \rho_2 \times (x_{r1}^m - x_{r2}^m) \tag{17}$$

based on the positions $X1_n^m$, $X2_n^m$ and the existing position (X_n^m), at the next iteration, the new coming solution (x_n^{m+1}) can be calculated as:

$$x_n^{m+1} = r_a \times (r_b \times X1_n^m + (1 - r_b) \times X2_n^m) + (1 - r_a) \times X3_n^m \tag{18}$$

where r_a and r_b are arbitrary values within the range [0, 1].

3.3. Local Escaping Operator Process

The LEO operator provides an important feature of the GBO to escape local solutions and improve the convergence rate. It helps the algorithm escape from local minima or saddle points, which can trap the algorithm and prevent it from finding the global minimum.

There are areas in the search space that can prevent the algorithm from finding the global minimum. The operator works by introducing random variations or perturbations to the model's parameters at each iteration. This helps the algorithm explore other regions of the parameter space and potentially discover a more optimal solution. As the optimization process continues, the amount of noise added is usually reduced to allow the algorithm to arrive at a more accurate result.

By using several positions ($X1_n^m$, $X2_n^m$), so LEO produces an optimal solution with advanced performance (X_{LEO}^m). The pseudo code of the operator LEO is shown in Algorithm 1.

Algorithm 1: Operator LEO

if rand $<$ *pr*
  if rand < 0.5
$X_{LEO}^m = X_n^{m+1} + f_1 \times \left(u_1 \times x_{\text{best}} - u_2 \times x_k^m\right) + f_2 \times \rho_1 \times \left(u_3 \times \left(X2_n^m - X1_n^m\right) + u_2 \times \left(x_{r1}^m - x_{r2}^m\right)\right)/2$
$X_n^{m+1} = X_{LEO}^m$
  else
$X_{LEO}^m = x_{\text{best}} + f_1 \times \left(u_1 \times x_{\text{best}} - u_2 \times x_k^m\right) + f_2 \times \rho_1 \times \left(u_3 \times \left(X2_n^m - X1_n^m\right) + u_2 \times \left(x_{r1}^m - x_{r2}^m\right)\right)/2$
$X_n^{m+1} = X_{LEO}^m$
  End
End

where *pr* is a probability value, *pr* = 0.5, the values f_1, and f_2 are represent distribution arbitrary values between [−1; 1], u_1, u_2, and u_3 are three random numbers, created as follows:

$$u_1 = \begin{cases} 2 \times \text{rand if } \mu_1 < 0.5 \\ 1 \text{ otherwise} \end{cases} \tag{19}$$

$$u_2 = \begin{cases} \text{rand if } \mu_1 < 0.5 \\ 1 \text{ otherwise} \end{cases} \tag{20}$$

$$u_3 = \begin{cases} \text{rand if } \mu_1 < 0.5 \\ 1 \text{ otherwise} \end{cases} \tag{21}$$

where rand is an arbitrary value in the range of [0, 1], and μ_1 is a random number from the interval [0, 1]. The previous equations can be written as:

$$u_1 = L_1 \times 2 \times \gamma_1 + (1 - L_1) \tag{22}$$

$$u_2 = L_1 \times \gamma_2 + (1 - L_1) \tag{23}$$

$$u_3 = L_1 \times \gamma_3 + (1 - L_1) \tag{24}$$

where γ_1, γ_2, and γ_3 are equal to rand. L_1 is a binary parameter with a value of 0 or 1. If parameter μ_1 is less than 0.5, the value of L_1 is 1, otherwise, it is 0. To determine the solution x_k^m in Equation (25), the following scheme is suggested.

$$x_k^m = \begin{cases} x_{\text{rand}} \text{ if } \mu_2 < 0.5 \\ x_p^m \text{ otherwise} \end{cases} \tag{25}$$

$$x_{\text{rand}} = X_{\min} + \text{rand}(0,1) \times (X_{\max} - X_{\min}) \tag{26}$$

4. Improved Gradient-Based Optimizer Algorithm

In this study, an Improved gradient-based optimizer (IGBO) is introduced to solve different real problems accurately. The main motives for enhancing the original GBO are described in the following sections.

4.1. Varying Inertia Weight

The inertia weight concept was introduced in many works of literature. During the last few decades, researchers have developed many inertias weigh strategies, and it has an important role in optimization processes using population-based metaheuristic algorithms. It provides a good balance between the local search and the global search capabilities of the algorithm. A mathematical equation is used to generate a new inertia weight [55].

In this paper time-varying, inertia weight strategy has been used in which the value of the inertia weight is determined based on the iteration number. The linearly decreasing

inertia weight was introduced by many researchers and was shown to be effective in improving the fine-tuning characteristic of algorithms [28]. The weight parameter ω is computed as follows:

$$\omega(\mathrm{Itr}) = \left(\frac{\mathrm{MaxItr} - \mathrm{Itr}}{\mathrm{MaxItr}}\right)^2 * (\omega_{\max} - \omega_{\min}) + \omega_{\min} \quad (27)$$

where $\omega(\mathrm{Itr})$: The inertia weight at iteration Itr

$\omega_{\min}$: The minimum inertia weight (final), $\omega_{\max}$: The maximum inertia weight (initial), MaxItr: Iteration by which inertial weight should be at final.

Different optimization problems need different Inertia boundary values. The need just to adjust set the minimum and the maximum boundaries ($\omega_{\min}$, $\omega_{\max}$) which are mentioned in some literature such as (0 and 1), (-1 and 1), or (0.9 and 1.2). In this study, for some optimization problems the suitable values for $\omega_{\min}$ and $\omega_{\max}$ are 0.4 and 0.9 respectively, and 0.8 and 0.2 for the other optimization problem. In each generation, the inertia weight $\omega(\mathrm{Itr})$ is generated automatically using Equation (27) within the selected range [56]. In this study, the magnitude of inertia weight is selected by the trial-and-error technique, so ($\omega_{\min} = 0.8$, $\omega_{\max} = 1.2$). Then Inertia weight will be used in the first position generated by LEO.

So, the first position x_n^{m+1} can be defined as:

$$x_n^{m+1} = \omega(\mathrm{Itr}) \times r_a \times (r_b \times X1_n^m + (1 - r_b) \times X2_n^m) + (1 - r_a) \times X3_n^m \quad (28)$$

The new strategy holds the simple idea that the particles should visit more positions in the search space at the beginning and have better local search ability at the end.

4.2. Adaptive Parameters

The tuning of the parameters has a huge impact on the optimization performance of the metaheuristic approach [57]. For different problems, they need different values of parameters; some problems need to enhance the diversity and the convergence speed. In another aspect, some optimization problems need to escape the higher level of diversity search that may lead to rash convergence and decreased convergence speed [58]. GBO has very important parameters to find globally the best solutions. However, GBO uses a fixed and random (rand) value. To escape the awkward parameters and achieve fine-tuning, a new parameter-setting-free technique is presented. The mechanism delivers the issue of setting five key parameters inside GSR and LEO [59].

4.2.1. Adaptive Parameters in GSR

The original GBO, the parameters r_a and r_b designed to have a random magnitude in the interval between [0, 1]. However, by the tests, we could find advanced optimal by giving high value to r_a and low values to r_b value within the domain [0, 1]. That can be done by changing the values of these parameters with mathematical equations used in IGBO. These mathematical equations extract the best values of the parameters, so with the increase in the number of iterations, the values of r_a will raise using Equation (27), and the value of r_b will decrease using Equation (28). That would help boost exploration to a larger extent and is likely to select the solutions from the entire feasible range [60]. Figure 1 show the visualization of adaptive parameters in GSR. The IGBO proposed the following equations to set the adjustment:

$$r_a = \frac{\mathrm{Itr}}{\mathrm{MaxItr}} \quad (29)$$

$$r_b = 1 - \left(\frac{\mathrm{Itr}}{\mathrm{MaxItr}}\right) \quad (30)$$

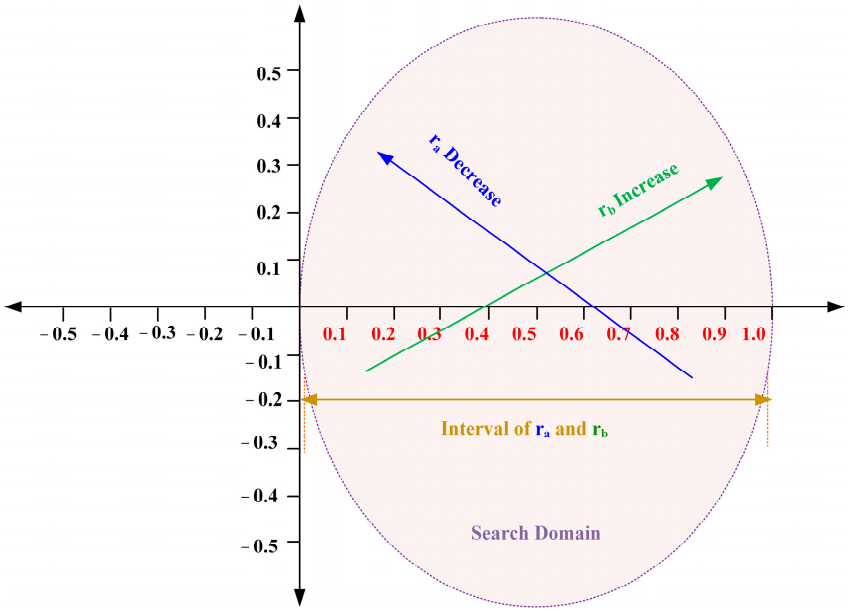

Figure 1. Visualization of adaptive parameters in GSR.

So, the new position at the next iteration x_n^{m+1} can be defined as:

$$x_n^{m+1} = \left(\frac{Itr}{MaxItr}\right) \times \left(\left(1 - \left(\frac{Itr}{MaxItr}\right)\right) \times X1_n^m + \left(1 - \left(1 - \left(\frac{Itr}{MaxItr}\right)\right)\right) \times X2_n^m\right) + \left(1 - \left(\frac{Itr}{MaxItr}\right)\right) \times X3_n^m \quad (31)$$

4.2.2. Adaptive Parameters in LEO

Similarly, in standard GBO the parameters γ_1, γ_2 and γ_3 set to have a random value in the range between [0, 1]. Whereas, to avoid the local optima, these parameters tend to have different approaches to achieve that goal. The parameter γ_1 is likely to increase its rate steadily along with the increase in iterations. In both parameters γ_2 and γ_3 tend to step down the values of them gradually with the increase in iterations, to make the algorithm modify the solutions to get the best one. The best solutions keep on adding the number of iterations until the last [61]. Therefore, the randomness equations of the parameters can be calculated as:

$$\gamma_1 = \frac{Itr}{MaxItr} \quad (32)$$

$$\gamma_2 = 1 - \left(\frac{Itr}{MaxItr}\right) \quad (33)$$

$$\gamma_3 = 1 - \left(\frac{Itr}{MaxItr}\right) \quad (34)$$

So, LEO uses the three random numbers u_1, u_2, and u_3, will be generated as follows:

$$u_1 = L_1 \times 2 \times \left(\frac{Itr}{MaxItr}\right) + (1 - L_1) \quad (35)$$

$$u_2 = L_1 \times \left(1 - \left(\frac{Itr}{MaxItr}\right)\right) + (1 - L_1) \quad (36)$$

$$u_3 = L_1 \times \left(1 - \left(\frac{Itr}{MaxItr}\right)\right) + (1 - L_1) \quad (37)$$

4.3. Adaptive Operator G

Increasing the range of random integers used in a special operator in optimization algorithm can assist the diversity and exploration of the search domain. This modification led

to a higher probability of finding the optimal solution in the case of optimization functions, and that depends on the nature of both the specific algorithm and optimization problem [62].

The operator G enhances the feasibility of the search in the search domain. Moreover, it improves the balancing between exploration and exploitation processes to search for the global optimal solution by increasing the G value throughout iterations instead of being chosen as a constant value or random between (0,1). Subsequently, the operating time of the proposed IGBO is decreased in contrast with the former GBO [63].

This method is enhancing the performance of optimization algorithms seeking to obtain the best solution and decreasing the search space. The convergence of the approach according to how the movement of solutions in the search domain [64]. In the GBO algorithm, the direction of movement (DM) is used to converge near the area of the solution. Therefore, the suggestion is to change the DM randomly with extra values to discover more areas in the search domain [44]. Hence, the DM of IGBO is utilized to be

$$DM = G \times \rho_2 \times (x_{best} - x_n) \tag{38}$$

Here, the value of the operator G changes significantly between specific limits.

The operator G can be expressed as

$$G = g_1 + ((g_2 - g_1) * (rand)) \tag{39}$$

where g_1 and g_2 are real numbers chosen in this study to be (1 and 10). This range selected by trial method according to the nature of IGBO algorithm and nature of the selected optimization functions used in this research.

This operator improves the search process by discovering more areas in the search space. Moreover, it optimizes the balance between exploitation and exploration. The pseudo code of the IGBO algorithm is shown in Algorithm 2.

Algorithm 2. Pseudo code of the IGBO algorithm

Step 1. Initialization
Assign values for parameters pr, ε, ω_{min}, ω_{max} , g_1 , g_2 and M
Generate an initial population $X_0 = [x_{0,1}, x_{0,2}, \ldots, x_{0,D}]$
Evaluate the objective function value $f(X_0)$, $n = 1, \ldots, N$
Specify the best and worst solutions x^m_{best} and x^m_{worst}
Step 2. Main loop
  While m<M (iterations) **do**
    for n = 1 : N (particles) **do**
      for i = 1 : D (dimensions) **do**
        Select randomly r1 ≠ r2 ≠ r3 ≠ r4 ≠ n in the range of [1, N]
        Calculate the operator G using Equation (37)
        Compute the direction of movement (DM) using Equation (36)
    Gradient search Rule
        Calculate the parameters r_a and r_b using Equations (27) and (28).
        Calculate the position $x^{m+1}_{n,i}$ using Equation (29)
      end
    Local escaping operator
      Calculate the parameters γ_1, γ_2 and γ_3 using Equations (30)–(32)
      Created random numbers u_1, u_2, and u_3 using Equations (33)–(35)
        if rand < pr **then**
          Calculate the position x^m_{LEO}
          At the next iteration calculate the new coming position X^{m+1}_n
        end
      Update the positions x^m_{best} and x^m_{worst}
    end
    m=m+1
  end
Step 3. return x^m_{best}

5. Test Process

This process examines IGBO and its counterparts with benchmark functions along with Friedman rank, and Real-world problems to verify their performance.

5.1. Parameter Settings

In all quantitative experiments, the initial parameters of the competitor algorithms were set according to the reference papers, as shown in Table 1.

Table 1. Initial parameters of the competitor algorithms.

Algorithm	**Ref.**	**Parameter**	
		Name	***Value***
IGBO		Probability Parameter (pr)	pr = 0.5
		Minimum Balance Parameter (β_{min})	$\beta_{min} = 0.2$
		Maximum Balance Parameter (β_{max})	$\beta_{max} = 1.2$
		Inertia weights ($\omega_{min}, \omega_{\max}$)	($\omega_{min} = 0.7$, $\omega_{\max} = 1.2$)
GBO	[16]	Probability Parameter (pr)	pr = 0.5
		Minimum Balance Parameter (β_{min})	$\beta_{min} = 0.2$
		Maximum Balance Parameter (β_{max})	$\beta_{max} = 1.2$
CS	[13]	Discovery rate of alien eggs/solutions (p)	$p = 0.25$
DE	[65]	Scale Factor (F)	$F = 0.5$
		Crossover Probability rate (C_r)	$C_r = 0.5$
FPA	[66]	Probability switch (p)	$p = 0.8$
PSO	[67]	Cognitive Constant (C_1)	$C_1 = 2$
		Social Constant (C_2)	$C_2 = 2$
		Minimum Inertia weight (ω_{min})	ω_{min} =0.7
		Maximum Inertia weight (ω_{max})	ω_{max} =0.9
		Maximum Velocity (V_{max})	V_{max} =0.002
TLBO	[65]	Teaching factor (T_f)	$T_f = 1, 2$
		Teaching step (T_s)	T_s chosen randomly between [0, 1]
AVOA	[15]	Probability parameter for selecting the first best vulture (L_1)	($L_1 = 0.8$)
		Probability parameter for selecting the second-best vulture (L_2)	($L_2 = 0.2$)
		A parameter that determines the disruption of the exploration and exploitation phases (k)	($k = 2.5$)
		A parameter to determine the probability of selecting the mechanisms in the exploration phase (p_1)	($p_1 = 0.6$)
		A parameter to determine the probability of selecting the mechanisms in the exploitation phase of the first part (p_2)	($p_2 = 0.4$)
		A parameter to determine the probability of selecting the mechanisms in the exploitation phase of the second phase (p_3)	($p_3 = 0.6$)

5.2. Benchmark Test Functions

The quality of optimization algorithms is commonly examined by using conventional standard literature benchmark functions [68]. There are various categories of these test functions. That variety in functions, to achieve the best representation of a wider range of real-world large-scale optimization functions and compare various metaheuristic algorithms with more convenience and flexibility [69]. Some optimization algorithms, besides the IGBO algorithm, will be tested using different benchmark functions, which are categorized into two groups: (i) Unimodal benchmark functions in Table 2, (ii) Multimodal benchmark functions in Table 3. In every table, (Dim) denotes the number of dimensions of the functions, while the lower and upper bounds of the variables are denoted by (Range), and the global minimum of the function is represented by (f_{min}) [70].

Table 2. Unimodal benchmark functions.

f. No.	Name	Dim	Range	f_{min}
F1	Sphere	30	[−100, 100]	0
F2	Schwefel's 2.20	30	[−100, 100]	0
F3	Schwefel's 2.21	30	[−100, 100]	0

Table 3. Multimodal benchmark functions.

f. No.	Name	Dim	Range	f_{min}
F4	Qing	30	[−500, 500]	0
f5	Alpine N. 1	30	[−10, 10]	0
F6	Xin-She Yang N. 1	30	[−5, 5]	0
F7	Salomon	30	[−100, 100]	0
F8	Xin-She Yang N. 2	30	[−2 pi, 2 pi]	0
F9	Penalized	30	[−50, 50]	0

5.3. Statistical Analysis

In this section, measurement criteria and statical tests are applied to prove the significance of the IGBO algorithm different from the others. These tests are analysis of variance (ANOVA), and Holm–Bonferroni test. These two statistical tests were conducted on results obtained by every algorithm from 50 independent runs with 1000 iterations each.

5.3.1. Measurement Criteria

To evaluate the results five measured criteria were used. These measurements were the worst, best, mean, median, and standard deviation (SD) [16].

5.3.2. Wilcoxon Signed-Rank Test

It is a non-parametric statistical test, which utilized to evaluate if there is a significant difference between two correlated samples or repeated measurements on the same sample. It is employed to determine whether there are a statistically significant difference two related sets of observations.

5.3.3. Friedman Rank Test

The Friedman test is a non-parametric statistical test established by Milton Friedman [64]. The test makes a comparison among the mean ranks in the related groups and identifies how the groups differed. Moreover, tells you which group of data was rated best versus worst. The Friedman test is widely supported by many statistical software packages [65]. IBM SPSS software has been used in this work to calculate Friedman Rank Test.

6. Evaluation of the Igbo Algorithm on Real-World Engineering Problems

In this section, three real-world design problems are tested. The results obtained by IGBO were compared with those of different well-known optimizers suggested in previous studies. The population and the maximum number of iterations were 30 and 500, respectively, for all problems.

6.1. Three-Bar Truss Design Problem

This engineering problem illustrates the truss's form and the forces applied to the structure as it is explained in Figure 2. This problem has two design parameters (x_1, x_2). The purpose of this problem is to minimize the total weight of the structure. It also has various constraints, such as deflection, buckling, and stress [15]; mathematically stated as follows:

$$\text{Minimize Fitness } \left(\vec{x}\right) = \left(2\sqrt{2}x_{A1} + x_{A2}\right) \times l \tag{40}$$

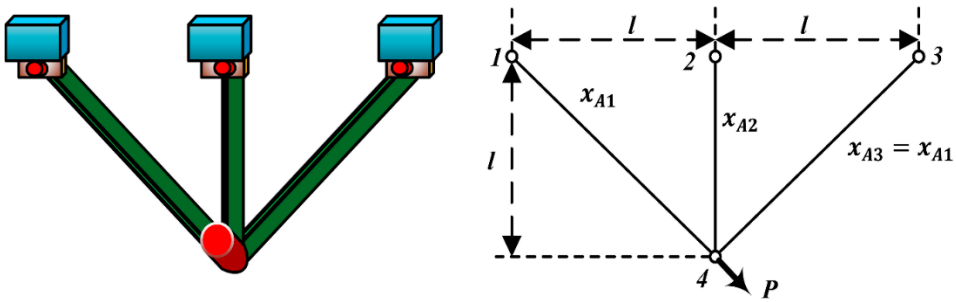

Figure 2. Three-Bar Truss Design.

Subject to:

$$g_1\left(\vec{x}\right) = \frac{\sqrt{2}x_{A1} + x_{A2}}{\sqrt{2}x_{A1}^2 + 2x_{A1}x_{A2}}P - \sigma \leq 0 \tag{41}$$

$$g_2\left(\vec{x}\right) = \frac{x_{A2}}{\sqrt{2}x_{A1}^2 + 2x_{A1}x_{A2}}P - \sigma \leq 0, \tag{42}$$

$$g_3\left(\vec{x}\right) = \frac{1}{\sqrt{2}x_{A2} + x_{A1}}P - \sigma \leq 0, \tag{43}$$

where, $0 \leq x_{A1},\ x_{A2} \leq 1$, $l = 100$ cm, $p = 2\frac{\text{kN}}{\text{cm}^2}$, $\sigma = 2\frac{\text{kN}}{\text{cm}^2}$

6.2. I-Beam Design Problem

It is also a difficult engineering structure optimization problem. The objective of this structure is to reduce the vertical deflection of the I-beam so that the design problem identifies the optimal geometric parameters related to the cross-section as displayed in Figure 3. The design variables in this problem are: length (h or x_1), height (l or x_2), and the thicknesses (t_w or x_3 and t_f or x_4) [71]. This optimization problem and its constraints are explained in the equations as follows:

$$\text{Considering} : X = [x_1, x_2,\ x_3,\ x_4] = [h, l, t_w, t_f] \tag{44}$$

$$\text{Minimize Fitness} = \frac{5000}{\frac{1}{12}t_w(h - 2t_f)^3 + \frac{1}{6}lt_f^3 + 2lt_f\left(\frac{h - t_f}{2}\right)^2} \tag{45}$$

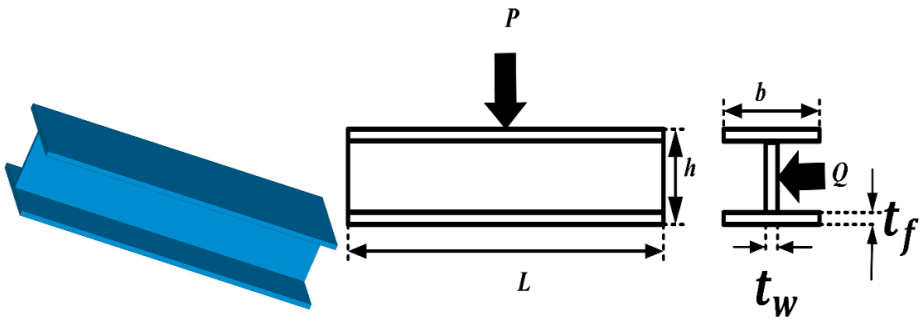

Figure 3. I-beam design.

Subject to:

$$g_1(x) = 2lt_f + t_w(h - 2t_f) \leq 300 \tag{46}$$

$$g_2(x) = \frac{180000x_1}{t_w(h - 2t_f)^3 + 2lt_f\left[4t_f^2 + 3h(h - 2t_f)\right]} + \frac{15000x_2}{(h - 2t_f)t_w^3 + 2t_f l^3} \leq 6 \tag{47}$$

The variables are subject to: $10 \leq h \leq 80$, $10 \leq l \leq 50$, $0.9 \leq t_w \leq 5$, $0.9 \leq t_f \leq 5$

6.3. Automatic Voltage Regulator Design Problem (AVR)

AVR is one of the real-world problems which the researchers tried to solve in different aspects [72]. Moreover, AVR is one of the main components in any power system which is used to control the outcome's voltage under different conditions of the operating process. The objective function of AVR is to estimate the optimal parameters of the Fractional Order Proportional Integrator Derivative (FOPID) controller [73]. The mathematical representation of all the AVR parts using Laplace transformation. The output transfer function of all parts of IGBO-based FOPID controlling in the AVR system is shown in Figure 4. Moreover, the parts are the FOPID controller, amplifier, exciter, generator, and sensor [74].

$$G_{FOPID}(s) = K_P + K_I s^{-\lambda} + K_D s^{\mu} \tag{48}$$

$$G_A(s) = \frac{K_A}{1 + ST_A} \tag{49}$$

$$G_E(s) = \frac{K_E}{1 + ST_E} \tag{50}$$

$$G_G(s) = \frac{K_G}{1 + ST_G} \tag{51}$$

$$G_S(s) = \frac{K_S}{1 + ST_S} \tag{52}$$

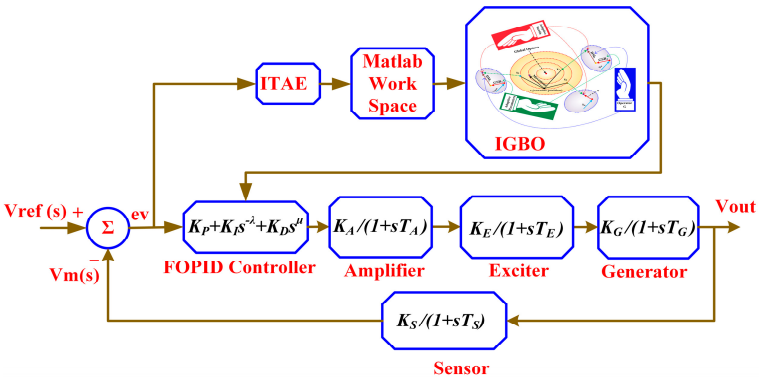

Figure 4. IGBO-based FOPID-AVR FOPID controller.

7. Results and Analysis

The parameters of IGBO and other algorithms are selected according to the subsection parameter settings. Then, they have been tested and executed using the MATLAB (R2021a) desktop computer running Windows 10 Enterprise 64-bit with an Intel ® Core ™ (Santa Clara, CA, USA) i5-8500 CPU processor and 8.00 GB RAM. All results are stored based on 50 population sizes and 50 independent runs with 1000 iterations for every run, then the results are compared using the obtained results.

In benchmark test functions, the results found by IGBO are compared with seven well-known algorithms (GBO, CS, DE, PSO, FPA, TLBO, AVOA). In addition, the results of IGBO in real-world problems are compared with the result of the same counterpart algorithms.

7.1. Benchmark Test Functions Results

This section describes the results of IGBO and the chosen algorithms using variant benchmark functions. Further, the results computes and compares the descriptive statistics in terms of best, worst, median, mean value, and standard deviation of all the algorithms. The best result for each function is highlighted in boldface.

The results of IGBO for unimodal benchmark functions are compared with other algorithms in Table 4. IGBO algorithm obtained the lowest value in the results in the functions (f1, f2) excluding the function (f3). This shows that IGBO has least variation in results of compared to the competitor's algorithms. Hence, IGBO is a better choice.

Table 4. Unimodal Benchmark Functions Results.

f. No	Statistics	IGBO	GBO	CS	DE	FPA	PSO	TLBO	AVOA
f1	Best	**0**	3.16E-281	1.02×10^{-17}	4.29×10^{-12}	$1.75 \times 10^{+04}$	5.13×10^{-13}	7.08×10^{-180}	2.67×10^{-261}
	Worst	0	3.45×10^{-267}	$1.33 \times 10^{+03}$	9.24×10^{-11}	$4.53 \times 10^{+04}$	9.39×10^{-11}	3.49×10^{-176}	3.41×10^{-258}
	Median	0	1.47×10^{-272}	2.77×10^{-10}	2.83×10^{-11}	$2.69 \times 10^{+04}$	9.84×10^{-12}	4.60×10^{-178}	$1.35 \times 10^{+260}$
	Mean	0	1.43×10^{-268}	$3.18 \times 10^{+01}$	3.20×10^{-11}	$2.79 \times 10^{+04}$	2.07×10^{-12}	3.19×10^{-177}	1.92×10^{-259}
	Std	0	8.54×10^{-270}	$1.90 \times 10^{+02}$	1.83×10^{-11}	$6.25 \times 10^{+03}$	2.23×10^{-11}	2.71×10^{-178}	1.96×10^{-125}
f2	Best	0	7.07×10^{-140}	3.78×10^{-07}	1.10×10^{-06}	$3.18 \times 10^{+02}$	1.44×10^{-06}	4.81×10^{-89}	9.34×10^{-137}
	Worst	0	5.95×10^{-134}	$5.10 \times 10^{+02}$	4.15×10^{-06}	$7.83 \times 10^{+02}$	3.37×10^{-04}	3.59×10^{-87}	1.16×10^{-111}
	Median	0	5.91×10^{-137}	$2.33 \times 10^{+00}$	2.04×10^{-06}	$5.64 \times 10^{+02}$	6.61×10^{-06}	5.91×10^{-88}	7.70×10^{-121}
	Mean	0	6.62×10^{-135}	$2.79 \times 10^{+01}$	2.15×10^{-06}	$5.73 \times 10^{+02}$	2.63×10^{-05}	8.23×10^{-88}	2.32×10^{-113}
	Std	0	1.51×10^{-134}	$7.85 \times 10^{+01}$	6.28×10^{-07}	$8.34 \times 10^{+01}$	6.28×10^{-05}	6.91×10^{-88}	2.34×10^{-116}
f3	Best	1.67×10^{-121}	3.87×10^{-130}	$1.05 \times 10^{+01}$	8.88×10^{-01}	$7.81 \times 10^{+01}$	9.33×10^{-03}	9.66×10^{-73}	8.52×10^{-120}
	Worst	3.12×10^{-85}	7.71×10^{-123}	$6.06 \times 10^{+01}$	$1.81 \times 10^{+00}$	$9.22 \times 10^{+01}$	4.73×10^{-02}	1.58×10^{-70}	2.52×10^{-121}
	Median	1.62×10^{-111}	1.04×10^{-125}	$2.57 \times 10^{+01}$	$1.24 \times 10^{+00}$	$8.68 \times 10^{+01}$	1.78×10^{-02}	1.03×10^{-71}	1.58×10^{-123}
	Mean	1.62×10^{-267}	3.37×10^{-124}	$2.79 \times 10^{+01}$	$1.26 \times 10^{+00}$	$8.63 \times 10^{+01}$	1.95×10^{-02}	1.66×10^{-71}	9.28×10^{-122}
	Std	$3.35 \times 10^{+00}$	1.81×10^{-123}	$1.23 \times 10^{+01}$	2.35×10^{-01}	$3.10 \times 10^{+00}$	7.96×10^{-03}	2.39×10^{-71}	2.72×10^{-121}

Table 5 illustrate *p*-values obtained from Wilcoxon rank-sum statistical test with 5% accuracy. By looking at the results, it is evident that IGBO has achieved excellent results with significant differences between the proposed IGBO approach and other optimization methods.

Table 5. Wilcoxon Signed Ranks test for unimodal benchmark functions.

IGBO vs.	GBO	CS	DE	FPA	PSO	TLBO	AVOA
Z	−2.097	−4.486	−5.170	−4.867	−6.245	−1.743	−3.341
***p*-values**	6.21×10^{-05}	9.04×10^{-12}	5.56×10^{-16}	3.55×10^{-14}	1.29×10^{-17}	3.49×10^{-19}	2.70×10^{-09}

By examining the results of unimodal functions using Friedman mean ranking test, it is obvious that the proposed IGBO algorithm performed better than other counterpart algorithms and had been able to achieve the appropriate score of 3.826 in Friedman test as it is clear from Figure 5.

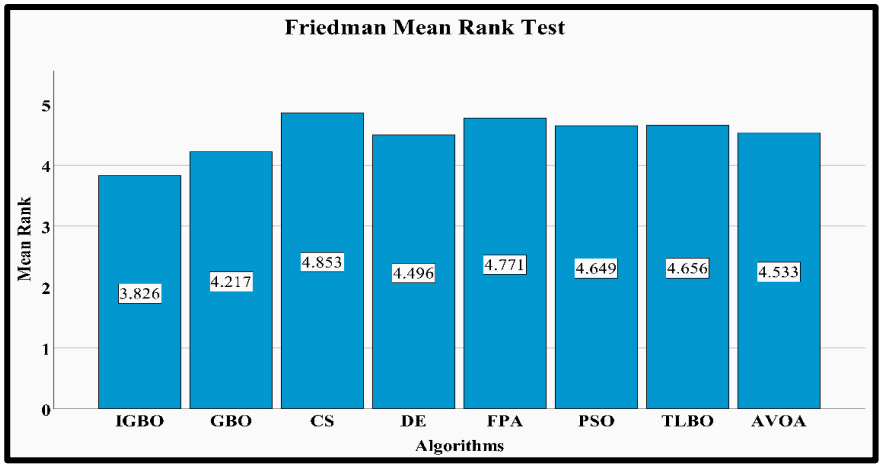

Figure 5. Friedman Mean rank test for unimodal functions.

Multimodal benchmark functions are complicated tasks to test the exploration ability of the optimization algorithms to find the main optimal region of search domain.

The results of IGBO for multimodal benchmark functions are compared with other algorithms in Table 6. Analysis of the results of this table shows that IGBO with its high exploration power, has provided the global optimal for (f4, f5, f6, f8), which indicates the more effective performance of IGBO.

Table 6. Multimodal Benchmark Functions Results.

f. No	Statistics	IGBO	GBO	CS	DE	FPA	PSO	TLBO	AVOA
f4	Best	5.54×10^{-14}	7.80×10^{-11}	2.93×10^{-16}	430.3679	$2.39 \times 10^{+10}$	5.92×10^{-10}	2.71×10^{-20}	1.26×10^{-06}
	Worst	6.29×10^{-09}	31.783	73,933,906	1186.599	$1.67 \times 10^{+11}$	6.82×10^{-07}	2.88×10^{-12}	8.9331
	Median	2.91×10^{-12}	2.3485	0.0000138	874.0392	$7.87 \times 10^{+10}$	1.01×10^{-08}	1.70×10^{-17}	1.68×10^{-05}
	Mean	2.17×10^{-10}	3.863	14,822,130	865.4393	$8.37 \times 10^{+10}$	3.22×10^{-08}	5.84×10^{-14}	0.40381
	Std	9.48×10^{-10}	5.3199	10,455,346	146.055	$3.49 \times 10^{+10}$	9.65×10^{-08}	4.07×10^{-13}	1.6162
f5	Best	1.83×10^{-158}	5.60×10^{-145}	2.60×10^{-08}	0.021702	28.1595	1.78×10^{-08}	1.75×10^{-90}	9.80×10^{-127}
	Worst	5.54×10^{-149}	1.80×10^{-134}	36.36	0.028265	53.1178	4.85×10^{-06}	1.15×10^{-88}	8.90×10^{-116}
	Median	2.67×10^{-154}	1.20×10^{-139}	5.8848	0.025524	44.5526	2.11×10^{-07}	2.67×10^{-89}	3.20×10^{-122}
	Mean	1.44×10^{-152}	4.20×10^{-136}	8.6883	0.025537	44.4585	4.57×10^{-07}	3.57×10^{-89}	1.90×10^{-117}
	Std	7.63×10^{-150}	2.50×10^{-135}	8.7131	0.001555	5.0097	8.49×10^{-07}	2.90×10^{-89}	8.29×10^{-91}
f6	Best	5.54×10^{-129}	5.00×10^{-116}	6.31×10^{-11}	1.02×10^{-10}	256,286.14	0.002556	4.71×10^{-55}	1.01×10^{-86}
	Worst	5.54×10^{-119}	2.58×10^{-60}	79,748,031	5.21×10^{-08}	$1.74 \times 10^{+10}$	1.4607	1.78×10^{-13}	8.67×10^{-03}
	Median	5.54×10^{-125}	4.00×10^{-100}	0.72962	2.22×10^{-09}	52,558,177	0.046811	1.41×10^{-28}	4.51×10^{-65}
	Mean	5.54	5.22×10^{-62}	16,042,004	5.65×10^{-09}	487,357,738	0.12426	3.90×10^{-15}	8.27×10^{-53}
	Std	5.54×10^{-14}	3.64×10^{-61}	11,276,834	9.88×10^{-09}	244,656,464	0.25385	2.52×10^{-14}	7.53×10^{-08}
f7	Best	0.1997	2.00×10^{-130}	4.4226	0.29987	13.0581	2.2999	0.29987	0.3275
	Worst	1.1327	9.00×10^{-101}	17.0238	0.40037	22.4805	4.3999	2.58987	1.86584
	Median	0.199873	3.40×10^{-120}	8.4499	0.31895	17.8066	2.9999	0.39884	2.7887
	Mean	0.234792	1.80×10^{-102}	8.8499	0.33611	17.8965	3.0799	0.46187	0.847516
	Std	0.03587	1.30×10^{-101}	2.9535	0.040328	1.9846	0.51627	0.014142	0.048512
f8	Best	1.015×10^{-12}	3.51×10^{-12}	2.64×10^{-11}	7.13×10^{-11}	5.64×10^{-11}	3.51×10^{-12}	2.79×10^{-11}	9.01×10^{-12}
	Worst	8.72×10^{-12}	3.45×10^{-12}	1.05×10^{-11}	2.32×10^{-11}	1.05×10^{-11}	2.71×10^{-06}	1.12×10^{-10}	3.58×10^{-11}
	Median	2.29×10^{-12}	3.47×10^{-12}	1.73×10^{-11}	2.65×10^{-11}	2.87×10^{-11}	6.54×10^{-10}	6.02×10^{-10}	7.18×10^{-11}
	Mean	2.68×10^{-12}	3.49×10^{-12}	1.82×10^{-11}	2.73×10^{-11}	2.95×10^{-11}	8.40×10^{-10}	7.01×10^{-10}	8.17×10^{-11}
	Std	5.29×10^{-13}	1.68×10^{-13}	1.48×10^{-12}	1.28×10^{-12}	7.45×10^{-13}	6.57×10^{-07}	7.08×10^{-10}	6.84×10^{-12}
f9	Best	1.52×10^{-18}	9.45×10^{-06}	1.24×10^{-16}	9.53×10^{-12}	2.63×10^{-5}	1.52×10^{-18}	6.29×10^{-24}	1.40×10^{-10}
	Worst	2.55×10^{-14}	0.10393	8.93×10^{-08}	2.42×10^{-10}	0.21576	1.5588	1.45×10^{-18}	3.31×10^{-09}
	Median	5.19×10^{-17}	3.40×10^{-05}	3.29×10^{-11}	5.25×10^{-11}	2.14×10^{-03}	0.10367	6.62×10^{-22}	9.63×10^{-10}
	Mean	9.97×10^{-16}	0.004205	1.46×10^{-10}	6.66×10^{-11}	0.014017	0.17841	8.04×10^{-20}	1.16×10^{-09}
	Std	3.81×10^{-15}	0.020548	0.005938	5.15×10^{-11}	6.72×10^{-02}	0.28147	2.69×10^{-19}	7.09×10^{-09}

Non-parametric Wilcoxon sign rank test has archived significant results when it is applied on multimodal benchmark functions, which shown in Table 7. The post hoc analysis confirms the effectiveness of the proposed method and it is statistically significant.

Table 7. Wilcoxon Signed Ranks test for multimodal benchmark functions.

IGBO vs.	GBO	CS	DE	FPA	PSO	TLBO	AVOA
Z	−4.189	−4.803	−7.574	−7.861	−3.986	−8.956	−5.122
***p*-Value**	1.23×10^{-11}	1.34×10^{-12}	3.62×10^{-15}	1.82×10^{-17}	6.90×10^{-09}	8.48×10^{-18}	1.054×10^{-13}

The obtained Friedman test results for multimodal functions are sown in Figure 6. The overall rank demonstrates that IGBO algorithm is superior to its counterparts. It obtained the lower value with 3.193.

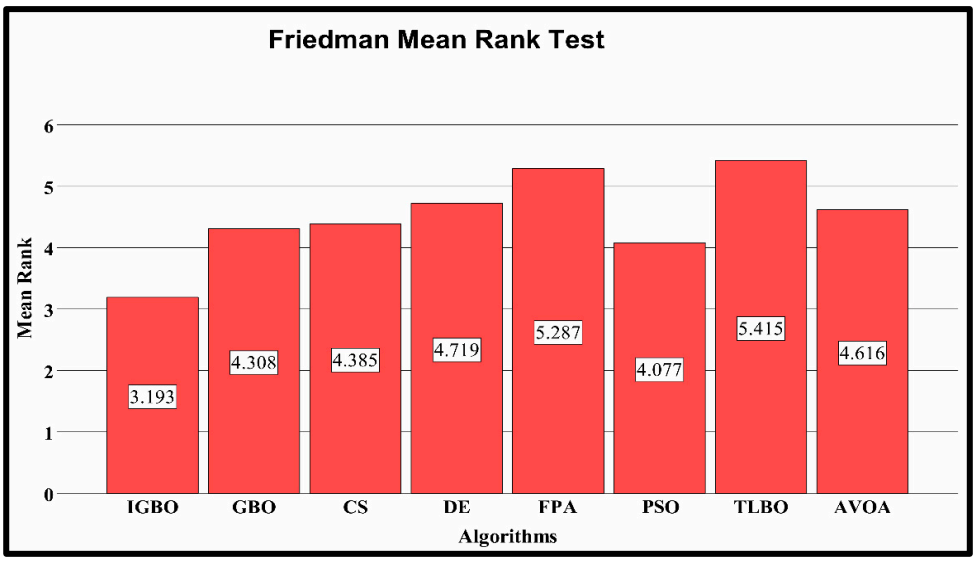

Figure 6. Friedman Mean rank test for multimodal functions.

To identify the nature of the algorithms with the functions, Figure 7 shown the 3D design, which allows for a more visual and intuitive understanding of the function's behavior and properties. The convergence curve is demonstrated in Figure 8, the evaluation of convergence capability shows the robustness of IGBO against different algorithms.

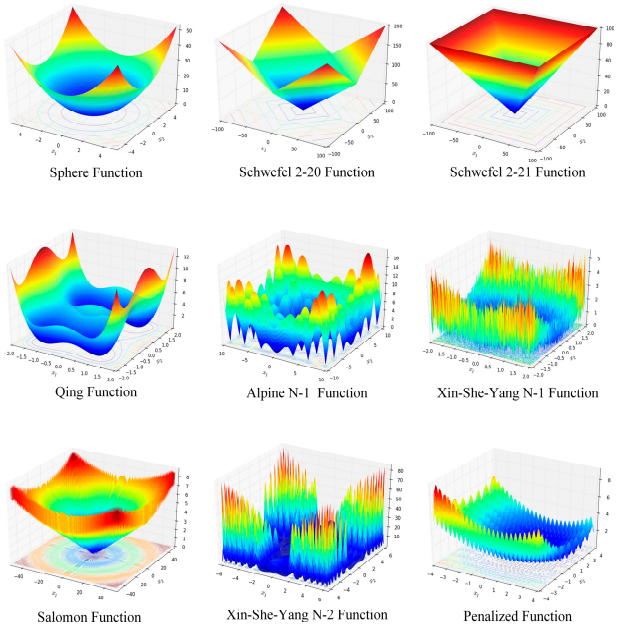

Figure 7. 3D design of the functions.

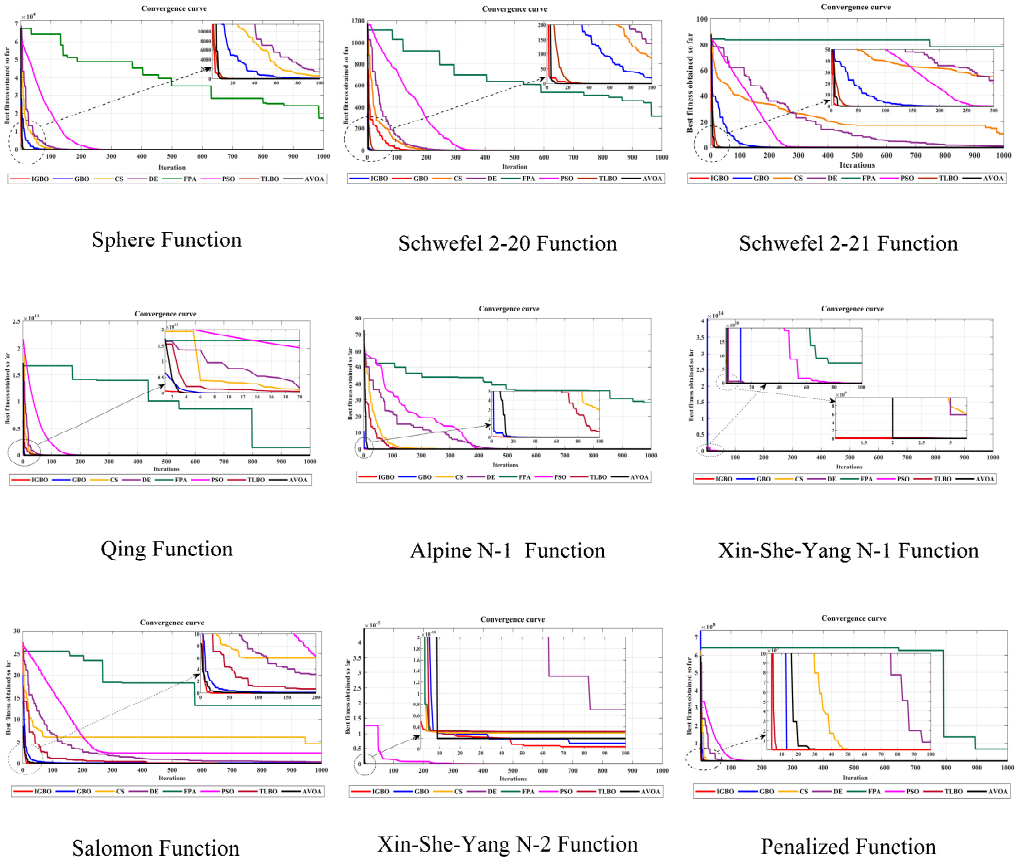

Figure 8. Convergence curve of the functions.

7.2. Comparison of Computational Time

The modifications to improve original GBO, make it able to find optimal from the entire feasible range with the proper balance of global and local search capabilities Moreover, they effect to boost the convergence speed of IGBO compared to its counterparts. Each algorithm runs 1000 iterations and the average of the elapsed time is considered a criterion for computational time. The proposed algorithm needs less time to find the best solution measured in seconds. Table 8 illustrates the comparison of computational time between IGBO and other algorithms.

Table 8. Comparison of computational time between IGBO and other algorithms.

Benchmark Test Functions	Elapsed Time (s)							
	IGBO	**GBO**	**CS**	**DE**	**FPA**	**PSO**	**TLBO**	**AVOA**
Unimodal functions	6.2186	7.9960	12.6258	8.2360	11.1468	9.05327	9.6487	8.9960
Multimodal functions	19.5034	20.6731	21.6094	22.1904	22.6831	20.0217	23.536	21.8772

7.3. Result of Real-World Problems

This part show results of the parameter values of the maximum function evaluations (MFEs), that compare the proposed IGBO algorithm against counterpart algorithms. All algorithms are used to solve real-world problems as mentioned before, with 50 numbers of population and 30 runs containing 1000 iterations.

7.3.1. Three-Bar Truss Design Results

Table 9 shows the results of the comparative algorithms for solving the three-bar truss design problem, and Figure 9 shows the convergence curve and best positions of the three-bar truss design using IGBO.

Table 9. Comparison of best solutions for the three-bar truss design.

Optimal Cost		IGBO	GBO	CS	DE	FPA	PSO	TLBO	AVOA
MFEs		30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000
Best Weight		263.8258	263.9861	264.1753	264.6827	265.1275	264.8643	266.9627	264.5241
Variables	**x(1)**	0.78868	0.78869	0.78677	0.78868	0.78868	0.78865	0.78871	0.78678
	x(2)	0.40825	0.40825	0.41366	0.40825	0.40825	0.40831	0.40815	0.40832

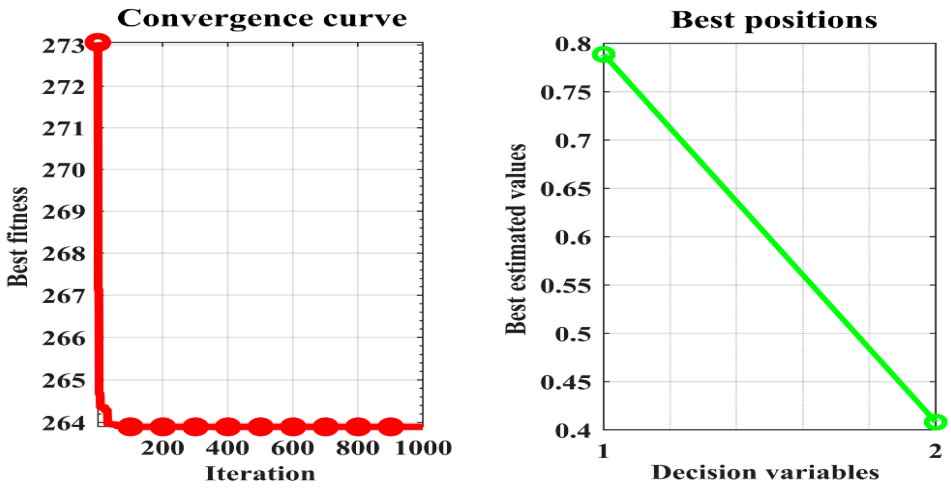

Figure 9. Convergence curve and best positions of three-bar truss design using IGBO.

The non-parametric Wilcoxon sign rank test has achieved significant results when applied to three-bar truss problem, as demonstrated in Table 10. Moreover, In Friedman test, IGBO algorithm has achieved the smallest rank compared with other intelligent optimization algorithms. These tests confirm the effectiveness of the proposed method and it is statistically significant.

Table 10. Wilcoxon and Friedman tests for three-bar truss problem.

	Test Type			
	Wilcoxon		Friedman	
	Z	*p*-Value	Mean Rank	Overall Rank
IGBO vs.	–	–	–	–
IGBO	–	–	1.233	1
GBO	3.014	1.66×10^{-23}	2.862	2
CS	−3.682	7.45×10^{-29}	4.355	4
DE	−5.062	3.08×10^{-35}	6.509	5
FPA	−7.608	5.21×10^{-39}	8.680	7
PSO	−6.853	7.94×10^{-36}	9.754	8
TLBO	−7.911	6.18×10^{-41}	7.953	6
AVOA	−3.849	9.21×10^{-34}	3.561	3

7.3.2. I-Beam Design Results

Table 11 shows the results of the comparative algorithms for solving the I-beam design problem, and Figure 10 shows the convergence curve and best positions of I-beam design using IGBO.

Table 11. Comparison of the best solutions for I-beam design.

Optimal Cost		IGBO	GBO	CS	DE	FPA	PSO	TLBO	AVOA
MFEs		30,000	30,000	30,000	30,000	30,000	30,000	30,000	30,000
Best Weight		0.013069	0.013074	0.013175	0.01593	0.013082	0.013075	0.013096	0.013075
Variables	(h)	79.98	80	80	80	80	80	80	80
	(b)	49.99	50	50	50	50	50	50	50
	(t_w)	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	(t_f)	2.32179	2.32179	2.32181	2.32179	2.32181	2.32179	2.32179	2.32179

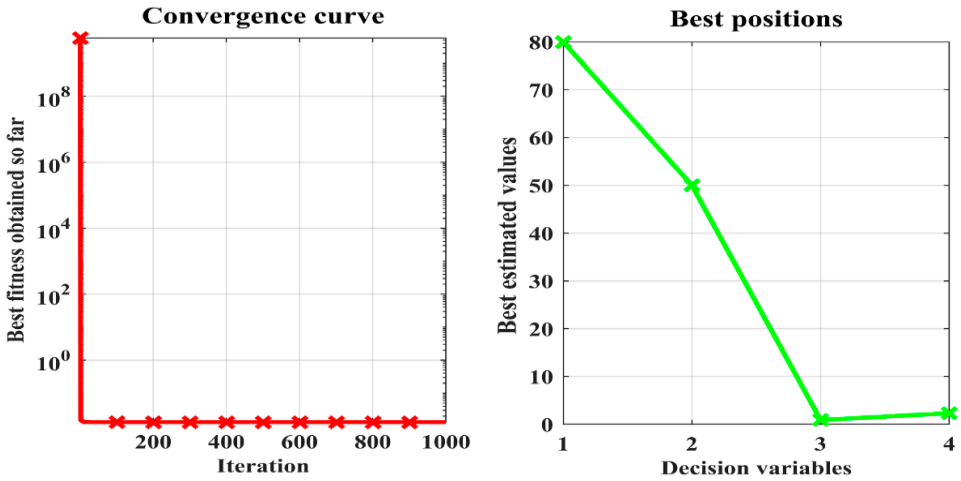

Figure 10. Convergence curve and best positions of I-beam design using IGBO.

Table 12 shows Wilcoxon sign rank test and Friedman mean rank test for I-beam problem. Wilcoxon test has produced significant outcomes when used on I-Beam Design results. Furthermore, the overall ranking using Friedman test proved that IGBO algorithm is superior to other algorithms.

Table 12. Wilcoxon and Friedman tests for I-beam problem.

	Test Type			
	Wilcoxon		Friedman	
	Z	*p*-Value	Mean Rank	Overall Rank
IGBO vs.	–	–	–	–
IGBO	–	–	2.617	1
GBO	−5.231	1.27×10^{-31}	4.108	3
CS	−8.573	7.95×10^{-41}	7.247	8
DE	−9.161	2.66×10^{-46}	4.976	4
FPA	−7.042	3.81×10^{-37}	5.862	6
PSO	−6.268	5.91×10^{-33}	5.354	5
TLBO	−7.843	1.78×10^{-38}	6.059	7
AVOA	−6.715	2.04×10^{-34}	3.865	2

7.3.3. Automatic Voltage Regulator Design Results

Due to the novelty performance and advanced ability in tuning, the proposed study optimization of the AVR system contains a FOPID controller. Figure 11 shows step response of the AVR-based IGBO-FOPID controller.

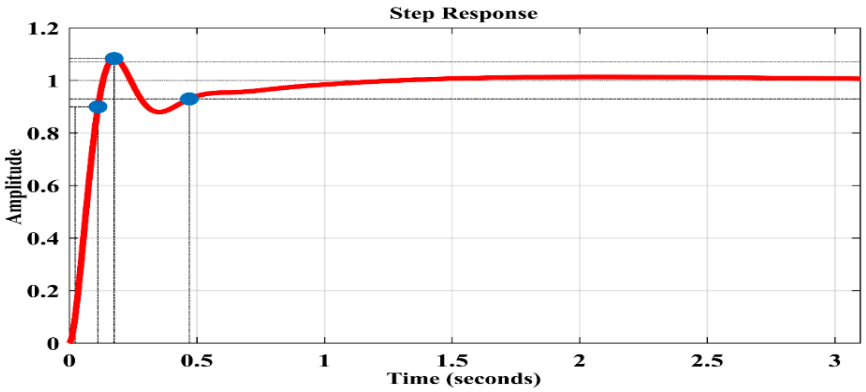

Figure 11. Step response of AVR-based IGBO-FOPID controller.

The FOPID parameters estimated at the end of the IGBO search process are shown in Table 13.

Table 13. Optimized FOPID parameters.

Parameter	Value
KP	1.508478
KI	0.992512
ʎ	1.051341
KD	0.621713
μ	1.219410

The transfer function model for the proposed GBO-based AVR system with the incorporation of optimized variables is given in the equation below:

$$\frac{V_{ref}(s)}{V_m(s)} = \frac{0.032251s^{3.5708} + 3.2251s^{2.5708} + 0.19585s^{2.3513} + 19.585s^{1.3513} + 0.056251s + 5.6251}{0.0004s^{5.3513} + 0.0454s^{4.3513} + 0.555s^{3.3513} + 3.2251s^{2.5708} + 1.51s^{2.3513} + 20.585s^{1.3513} + 5.6251} \quad (53)$$

To validate the effectiveness of the proposed optimal AVR design, its dynamic response is compared with that of the previously designed AVRs under identical operating conditions as shown in Figure 12.

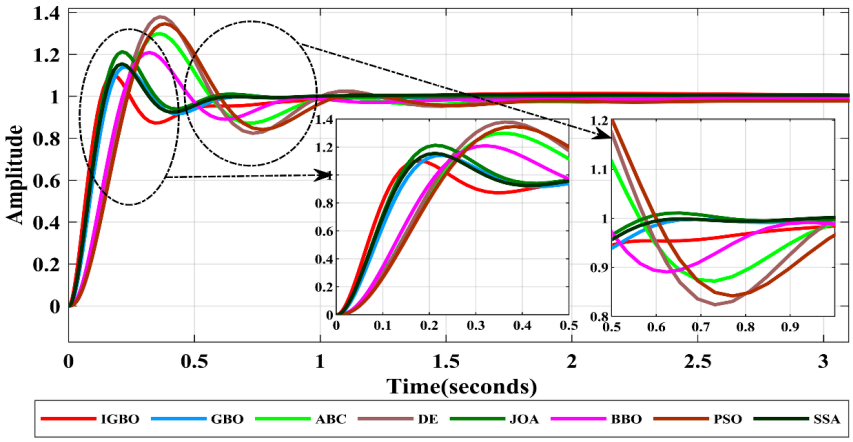

Figure 12. Comparison of proposed AVR design with other methods.

Table 14 provides the quantitative evaluation of the dynamic response based on some of the very important dynamic response indicators such as percentage overshoot, settling time, and peak time.

Table 14. Dynamic response evaluation of proposed GBO-based AVR design.

PID/FOPID Tuning Method	Peak Value (Pv)	Percentage Overshoot (%Mp)	Rise Time (tr)	Peak Time (tp)	Settling Time (ts)
IGBO-FOPID (Proposed)	1.084	8.36	0.0899	0.176	0.932
GBO-FOPID [31]	1.110	11.3	0.0885	0.16	0.653
JOA-FOPID [74]	1.130	13.2	0.0827	0.1750	0.453
SSA-FOPID [75]	1.15	15.5	0.0981	0.209	0.551
DE-PID [76]	1.3285	32.8537	0.1516	0.3655	2.6495
PSO- PID [76]	1.3006	30.0634	0.1610	0.3824	3.3994
ABC-PID [76]	1.2501	25.0071	0.1557	0.3676	3.0939
BBO-PID [77]	1.1552	15.5187	0.1485	0.3165	1.4457
PSA-PID [78]	1.1693	16.93	0.1438	0.3159	0.8039

Conversely, Table 15 describes stability criterion results with the best AVR design-based algorithms. The stability indicators are Phase Margin (PM), Delay Margin (DM), Bandwidth (BW), and Phase Gain (PG). As can be seen from the proposed GBO tuned FOPID-AVR provides the most stable design among the considered AVRs with the highest PM and BW values.

Table 15. Comparative AVR designs are based on stability evaluation indicators.

PID/FOPID-Tuning Method	PM	DM	BW	PG
IGBO-FOPID (Proposed)	95.9	0.0866	19.3	0.971
GBO-FOPID [31]	91.1	0.0753	21.27	1.2
JOA-FOPID [74]	90.3	0.0765	20.60	1.24
SSA-FOPID [75]	89.3	0.0910	17.01	1.21
DE-PID [76]	58.4	0.0920	12.8	4.2
PSO- PID [76]	62.2	0.1030	12.182	3.75
ABC-PID [76]	69.4	0.111	12.879	2.87
BBO-PID [77]	81.6	0.122	14.284	1.56
PSA-PID [78]	79.69	0.115	14.636	1.68

In summary, the results of the real-world problems, demonstrate that IGBO can deal with different challenging problems and various combinatorial optimization problems. Thus, IGBO is the most powerful optimization algorithm with the lowest computational costs and high convergence speed to get the optimal solution.

8. Discussion

In this section, a comprehensive understanding of the performance of the proposed IGBO algorithm and its significance in comparison to the other algorithms studied in this manuscript. The key findings from this study are summarized as follows:

- The initial comparison that was made between IGBO and several state-of-the-art algorithms such as GBO, CS, DE, FPA, PSO, TLBO, and AVOA using unimodal benchmark functions with dimensions of 30. Whereas, the optimization based on 50 population sizes and 50 independent runs with 1000 iterations for every run. the proposed IGBO algorithm demonstrated outstanding performance in most cases.
- The second comparison between IGBO and the same its counterparts' algorithms using the multimodal benchmark functions along with 30 dimensions. The results revealed a remarkable performance from the version of IGBO.

- The implementation of evolutionary algorithms competitors to several real-world constraint mechanical design optimization problems. The IGBO algorithm has shown great performance when applied to solving real-world constraint optimization problems, and as such, it has been successfully utilized to address automatic voltage regulator design.

It is worthwhile to mention that, despite the excellent optimizing capability of the IGBO algorithm, there are some limitations associated with the proposed IGBO such as solutions for the optimal temporary spatial-variation constraint sample are very difficult to achieve and comparatively greater computational complexity than its competing algorithms.

9. Conclusions and Future Works

In this manuscript, an Improved Gradient Optimization (IGBO) algorithm has been presented for solving real-world engineering and optimization problems. The proposed IGBO performance was examined using benchmark test functions to verify its effectiveness. To validate its superior optimization capabilities, its performance is compared with the seven most widely used metaheuristic-based optimization algorithms. The results demonstrate that the proposed algorithm is better than its competing algorithms in terms of achieving the most optimal solution to the benchmark functions and real-world engineering designs. Moreover, the statistical test analysis shows that the IGBO algorithm has better performance than its original version. In addition, the solution to three real-world problems was also examined and compared with other well-known algorithms to justify the performance of the proposed IGBO algorithm. The optimization results showed that IGBO has a high exploration ability in the scanning search domain, escaping local areas, and finding the main optimal area. IGBO is superior than the seven competitor algorithms and provides far more competitive optimization results in solving unimodal and multimodal benchmark functions. Moreover, IGBO performance in evaluating three design problems showed its high ability to solve real- world optimization problems.

Finally, multi-objective and binary forms of IGBO may be established as future works for solving multi-objective and discrete optimization problems. In addition, the utilize of a chaotic map in every iteration will improve the performance to avoid local optima and accelerate convergence. Furthermore, using IGBO for solving real-world optimization problems in different applications and domains can be another valuable future work.

Author Contributions: Conceptualization, S.M.A.A.; Methodology, S.M.A.A.; Software, S.M.A.A.; Validation, S.M.A.A.; Writing—original draft, S.M.A.A.; Validation, N.H., A.Y.,S.A.H., L.M. and R.H.A.; Writing—(review & editing), N.H., A.Y., S.A.H., L.M. and R.H.A.; Supervision, A.S.B.M., S.B.A.K. and H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The first author thanks the Libyan government for supporting this research with a scholarship provided by the Ministry of Higher Education and Scientific Research. The authors would like to acknowledge the facilities provided by Universiti Teknologi Malaysia for the accomplishment of this work. The authors would like to thank the editor and anonymous reviewers for improving this paper with their valuable comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liberti, L.; Maculan, N. *Global Optimization: From Theory to Implementation*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006; Volume 84, ISBN 0387282602.
2. Kochenderfer, M.J.; Wheeler, T.A. *Algorithms for Optimization*; Mit Press: London, UK, 2019; ISBN 0262039427.
3. Rao, S.S. *Engineering Optimization: Theory and Practice*, 5th ed.; John Wiley & Sons: Hoboken, NJ, USA, 2019; ISBN 1119454719.
4. Bozorg-Haddad, O. *Advanced Optimization by Nature-Inspired Algorithms*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 720, ISBN 9789811052200.
5. Nocedal, J.; Wright, S.J. *Numerical Optimization*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 1999; ISBN 9780387244921.

6. Yang, X.-S. *Nature-Inspired Optimization Algorithms*, 1st ed.; Elsevier: Amsterdam, The Netherlands, 2014; ISBN 9780124167452.
7. Goldberg, D.E.; Holland, J.H. Genetic Algorithms and Machine Learning. *Mach. Learn.* **1988**, *3*, 95–99. [CrossRef]
8. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science (1979)* **1983**, *220*, 671–680. [CrossRef]
9. Eberhart, R.; Kennedy, J. A New Optimizer Using Particle Swarm Theory. In *the MHS'95, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995*; pp. 39–43.
10. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341. [CrossRef]
11. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–Learning-Based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. [CrossRef]
12. Rao, R. Jaya: A Simple and New Optimization Algorithm for Solving Constrained and Unconstrained Optimization Problems. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 19–34. [CrossRef]
13. Yang, X.S.; Deb, S. Cuckoo Search via Lévy Flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing, Coimbatore, India, 9–11 December 2009; pp. 210–214.
14. Yang, X.-S. Flower Pollination Algorithm for Global Optimization. In Proceedings of the International Conference on Unconventional Computing and Natural Computation, Orléan, France, 3–7 September 2012; Springer: Berlin/HeiBerlin/Heidelberg, Germany; pp. 240–249.
15. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African Vultures Optimization Algorithm: A New Nature-Inspired Metaheuristic Algorithm for Global Optimization Problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
16. Ahmadianfar, I.; Bozorg-Haddad, O.; Xuefeng, C. Gradient-Based Optimizer: A New Metaheuristic Optimization Algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [CrossRef]
17. Rao, R.V.; Savsani, V.J. *Mechanical Design Optimization Using Advanced Optimization Techniques*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; ISBN 978-1-4471-2747-5.
18. Mirjalili, S. *Evolutionary Algorithms and Neural Networks; Studies in Computational Intelligence*, 1st ed.; Springer International Publishing: Cham, Switzerland, 2019; ISBN 978-3-319-93024-4.
19. Li, Y.; Yu, X.; Liu, J. An Opposition-Based Butterfly Optimization Algorithm with Adaptive Elite Mutation in Solving Complex High-Dimensional Optimization Problems. *Math. Comput. Simul.* **2023**, *204*, 498–528. [CrossRef]
20. Goldanloo, M.J.; Gharehchopogh, F.S. A Hybrid OBL-Based Firefly Algorithm with Symbiotic Organisms Search Algorithm for Solving Continuous Optimization Problems. *J. Supercomput.* **2022**, *78*, 3998–4031. [CrossRef]
21. Ghafori, S.; Gharehchopogh, F.S. Advances in Spotted Hyena Optimizer: A Comprehensive Survey. *Arch. Comput. Methods Eng.* **2021**, *29*, 1569–1590. [CrossRef]
22. Dizaji, Z.A.; Gharehchopogh, F.S. A Hybrid of Ant Colony Optimization and Chaos Optimization Algorithms Approach for Software Cost Estimation. *Indian J. Sci. Technol.* **2015**, *8*, 128–133. [CrossRef]
23. Mohammadzadeh, H.; Gharehchopogh, F.S. A Novel Hybrid Whale Optimization Algorithm with Flower Pollination Algorithm for Feature Selection: Case Study Email Spam Detection. *Comput. Intell.* **2021**, *37*, 176–209. [CrossRef]
24. Abdollahzadeh, B.; Soleimanian Gharehchopogh, F.; Mirjalili, S. Artificial Gorilla Troops Optimizer: A New Nature-Inspired Metaheuristic Algorithm for Global Optimization Problems. *Int. J. Intell. Syst.* **2021**, *36*, 5887–5958. [CrossRef]
25. Jamil, M.; Yang, X.S. A Literature Survey of Benchmark Functions for Global Optimisation Problems. *Int. J. Math. Model. Numer. Optim.* **2013**, *4*, 150–194. [CrossRef]
26. Tang, K.; Yáo, X.; Suganthan, P.N.; MacNish, C.; Chen, Y.-P.; Chen, C.-M.; Yang, Z. Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization. *Nat. Inspired Comput. Appl. Lab. USTC China* **2007**, *24*, 1–18.
27. Ornek, B.N.; Aydemir, S.B.; Duzenli, T.; Ozak, B. A Novel Version of Slime Mould Algorithm for Global Optimization and Real World Engineering Problems: Enhanced Slime Mould Algorithm. *Math. Comput. Simul.* **2022**, *198*, 253–288. [CrossRef]
28. Nickabadi, A.; Ebadzadeh, M.M.; Safabakhsh, R. A Novel Particle Swarm Optimization Algorithm with Adaptive Inertia Weight. *Appl. Soft Comput. J.* **2011**, *11*, 3658–3670. [CrossRef]
29. Drake, J.H.; Kheiri, A.; Ozcan, E.; Burke, E.K. Recent Advances in Selection Hyper-Heuristics. *Eur. J. Oper. Res.* **2020**, *285*, 405–428. [CrossRef]
30. Schmitt, A. On the Number of Relational Operators Necessary to Compute Certain Functions of Real Variables. *Acta Inf.* **1983**, *19*, 297–304. [CrossRef]
31. Masoud Abdallah Altbawi, S.; Safawi Bin Mokhtar, A.; Ahmed Jumani, T.; Khan, I.; Hamadneh, N.N.; Khan, A. Optimal Design of Fractional Order PID Controller Based Automatic Voltage Regulator System Using Gradient-Based Optimization Algorithm. *J. King Saud Univ. Eng. Sci.* **2021**. [CrossRef]
32. Ashraf, Z.; Shahid, M.; Ahmad, F. Gradient Based Optimization Approach to Solve Reliability Allocation System. In Proceedings of the 2021 International Conference on Computing, Communication, and Intelligent Systems, Greater Noida, India, 19–20 February 2021; pp. 337–342.
33. Ismaeel, A.A.K.; Houssein, E.H.; Oliva, D.; Said, M. Gradient-Based Optimizer for Parameter Extraction in Photovoltaic Models. *IEEE Access* **2021**, *9*, 13403–13416. [CrossRef]
34. Deb, S.; Abdelminaam, D.S.; Said, M.; Houssein, E.H. Recent Methodology-Based Gradient-Based Optimizer for Economic Load Dispatch Problem. *IEEE Access* **2021**, *9*, 44322–44338. [CrossRef]

35. Rezk, H.; Ferahtia, S.; Djeroui, A.; Chouder, A.; Houari, A.; Machmoum, M.; Abdelkareem, M.A. Optimal Parameter Estimation Strategy of PEM Fuel Cell Using Gradient-Based Optimizer. *Energy* **2021**, *239*, 122096. [CrossRef]
36. Elsheikh, A.H.; Abd Elaziz, M.; Vendan, A. Modeling Ultrasonic Welding of Polymers Using an Optimized Artificial Intelligence Model Using a Gradient-Based Optimizer. *Weld. World* **2021**, *66*, 27–44. [CrossRef]
37. Ouadfel, S.; Abd Elaziz, M. A Multi-Objective Gradient Optimizer Approach-Based Weighted Multi-View Clustering. *Eng. Appl. Artif. Intell.* **2021**, *106*, 104480. [CrossRef]
38. Premkumar, M.; Jangir, P.; Sowmya, R. MOGBO: A New Multiobjective Gradient-Based Optimizer for Real-World Structural Optimization Problems. *Knowl. Based Syst.* **2021**, *218*, 106856. [CrossRef]
39. Premkumar, M.; Jangir, P.; Ramakrishnan, C.; Nalinipriya, G.; Alhelou, H.H.; Kumar, B.S. Identification of Solar Photovoltaic Model Parameters Using an Improved Gradient-Based Optimization Algorithm with Chaotic Drifts. *IEEE Access* **2021**, *9*, 62347–62379. [CrossRef]
40. Zhou, W.; Wang, P.; Asghar, A.; Zhao, X.; Turabieh, H.; Chen, H.; Heidari, A.A.; Zhao, X.; Turabieh, H.; Chen, H. Random Learning Gradient Based Optimization for Efficient Design of Photovoltaic Models. *Energy Convers. Manag.* **2021**, *230*, 113751. [CrossRef]
41. Zheng, Y.; Wu, J.; Wang, B. CLGBO: An Algorithm for Constructing Highly Robust Coding Sets for DNA Storage. *Front. Genet.* **2021**, *12*, 644945. [CrossRef] [PubMed]
42. Hassan, M.H.; Houssein, E.H.; Mahdy, M.A.; Kamel, S. An Improved Manta Ray Foraging Optimizer for Cost-Effective Emission Dispatch Problems. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104155. [CrossRef]
43. Khalilpourazari, S.; Doulabi, H.H.; Çiftçioğlu, A.Ö.; Weber, G.-W. Gradient-Based Grey Wolf Optimizer with Gaussian Walk: Application in Modelling and Prediction of the COVID-19 Pandemic. *Expert Syst. Appl.* **2021**, *177*, 114920. [CrossRef]
44. Hassan, M.H.; Kamel, S.; El-Dabah, M.A.; Rezk, H. A Novel Solution Methodology Based on a Modified Gradient-Based Optimizer for Parameter Estimation of Photovoltaic Models. *Electronics* **2021**, *10*, 472. [CrossRef]
45. Jiang, Y.; Luo, Q.; Wei, Y.; Abualigah, L.; Zhou, Y. An Efficient Binary Gradient-Based Optimizer for Feature Selection. *Math. Biosci. Eng.* **2021**, *18*, 3813–3854. [CrossRef] [PubMed]
46. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
47. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.A.; Gandomi, A.H. Aquila Optimizer: A Novel Meta-Heuristic Optimization Algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
48. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Dwarf Mongoose Optimization Algorithm. *Comput. Methods Appl. Mech Eng.* **2022**, *391*, 114570. [CrossRef]
49. Oyelade, O.N.; Ezugwu, A.E.S.; Mohamed, T.I.A.; Abualigah, L. Ebola Optimization Search Algorithm: A New Nature-Inspired Metaheuristic Optimization Algorithm. *IEEE Access* **2022**, *10*, 16150–16177. [CrossRef]
50. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Gazelle Optimization Algorithm: A Novel Nature-Inspired Metaheuristic Optimizer. *Neural. Comput. Appl.* **2022**, 1–33. [CrossRef]
51. Ezugwu, A.E.; Agushaka, J.O.; Abualigah, L.; Mirjalili, S.; Gandomi, A.H. Prairie Dog Optimization Algorithm. *Neural. Comput. Appl.* **2022**, *34*, 20017–20065. [CrossRef]
52. Abualigah, L.; Elaziz, M.A.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A Nature-Inspired Meta-Heuristic Optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
53. Alkayem, N.F.; Shen, L.; Al, T.; Qian, X. Inverse Analysis of Structural Damage Based on the Modal Kinetic and Strain Energies with the Novel Oppositional Unified Particle Swarm Gradient-Based Optimizer. *Appl. Sci.* **2022**, *12*, 11689. [CrossRef]
54. Alkayem, N.F.; Cao, M.; Shen, L.; Fu, R.; Šumarac, D. The Combined Social Engineering Particle Swarm Optimization for Real-World Engineering Problems: A Case Study of Model-Based Structural Health Monitoring. *Appl. Soft. Comput.* **2022**, *123*, 108919. [CrossRef]
55. Coelho, D.B.P.; Rodrigues, L.R. A Chaotic Inertia Weight TLBO Applied to Troubleshooting Optimization Problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
56. Hussain, Z.; Yusri, M.; Mat, D.; Leghari, Z.H.; Hassan, M.Y.; Said, D.M.; Jumani, T.A.; Memon, Z.A. A Novel Grid-Oriented Dynamic Weight Parameter Based Improved Variant of Jaya Algorithm. *Adv. Eng. Softw.* **2020**, *150*, 102904. [CrossRef]
57. Gao, L.; Li, S.; Kong, X.; Zou, D. On the Iterative Convergence of Harmony Search Algorithm and a Proposed Modification. *Appl. Math Comput.* **2014**, *247*, 1064–1095.
58. Li, X.; Yin, M. Modified Cuckoo Search Algorithm with Self Adaptive Parameter Method. *Inf. Sci.* **2015**, *298*, 80–97. [CrossRef]
59. Luo, K.; Ma, J.; Zhao, Q. Enhanced Self-Adaptive Global-Best Harmony Search without Any Extra Statistic and External Archive. *Inf. Sci.* **2019**, *482*, 228–247. [CrossRef]
60. Kumar, V.; Chhabra, J.K.; Kumar, D. Parameter Adaptive Harmony Search Algorithm for Unimodal and Multimodal Optimization Problems. *J. Comput. Sci.* **2014**, *5*, 144–155. [CrossRef]
61. Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C. Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients. *IEEE Trans. Evol. Comput.* **2004**, *8*, 240–255. [CrossRef]
62. Li, S.; Gong, W.; Yan, X.; Hu, C.; Bai, D.; Wang, L.; Gao, L. Parameter Extraction of Photovoltaic Models Using an Improved Teaching-Learning-Based Optimization. *Energy Convers. Manag.* **2019**, *186*, 293–305. [CrossRef]

63. Korashy, A.; Kamel, S.; Youssef, A.R.; Jurado, F. Modified Water Cycle Algorithm for Optimal Direction Overcurrent Relays Coordination. *Appl. Soft Comput. J.* **2019**, *74*, 10–25. [CrossRef]
64. Oliva, D.; Rodriguez-Esparza, E.; Martins, M.S.R.; Abd Elaziz, M.; Hinojosa, S.; Ewees, A.A.; Lu, S. Balancing the Influence of Evolutionary Operators for Global Optimization. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; pp. 1–8.
65. Zhao, W.; Wang, L.; Mirjalili, S. Artificial Hummingbird Algorithm: A New Bio-Inspired Optimizer with Its Engineering Applications. *Comput. Methods Appl. Mech. Eng.* **2022**, *388*, 114194. [CrossRef]
66. Nilanjan, D. (Ed.) *Applications of Flower Pollination Algorithm and Its Variants*; Springer: Singapore, 2021.
67. Nadimi-shahraki, M.H.; Fatahi, A.; Zamani, H.; Mirjalili, S.; Oliva, D. *Hybridizing of Whale and Moth-Flame Optimization Algorithms to Solve Diverse Scales of Optimal Power Flow Problem*; Multidisciplinary Digital Publishing Institute: Basel, Switzerland, 2022; Volume 11.
68. Naseri, T.S.; Gharehchopogh, F.S. A Feature Selection Based on the Farmland Fertility Algorithm for Improved Intrusion Detection Systems. *J. Netw. Syst. Manag.* **2022**, *30*, 40. [CrossRef]
69. Zhang, X.; Wang, S.; Zhao, K.; Wang, Y. A Salp Swarm Algorithm Based on Harris Eagle Foraging Strategy. *Math Comput. Simul.* **2023**, *203*, 858–877. [CrossRef]
70. Askari, Q.; Saeed, M.; Younas, I. Heap-Based Optimizer Inspired by Corporate Rank Hierarchy for Global Optimization. *Expert Syst. Appl.* **2020**, *161*, 113702. [CrossRef]
71. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo Search Algorithm: A Metaheuristic Approach to Solve Structural Optimization Problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
72. Altbawi, S.M.A.; bin Mokhtar, A.S.; Arfeen, Z.A. Enhacement of Microgrid Technologies Using Various Algorithms. *Turk. J. Comput. Math. Educ.* **2021**, *12*, 1127–1170. [CrossRef]
73. Memon, A.; Wazir Bin Mustafa, M.; Anjum, W.; Ahmed, A.; Ullah, S.; Altbawi, S.M.A.; Jumani, T.A.; Khan, I.; Hamadneh, N.N. Dynamic Response and Low Voltage Ride-through Enhancement of Brushless Double-Fed Induction Generator Using Salp Swarm Optimization Algorithm. *PLoS ONE* **2022**, *17*, e0265611. [CrossRef] [PubMed]
74. Jumani, A.J.; WazirMustafa., M.; Hussain, Z.; Md. Rasid, M.; Saeed, M.S.; Memon, M.M.; Khan, I.; Sooppy Nisar, K. Jaya Optimization Algorithm for Transient Response and Stability Enhancement of a Fractional-Order PID Based Automatic Voltage Regulator System. *Alex. Eng. J.* **2020**, *59*, 2429–2440. [CrossRef]
75. Khan, I.A.; Alghamdi, A.S.; Jumani, T.A.; Alamgir, A.; Awan, A.B.; Khidrani, A. Salp Swarm Optimization Algorithm-Based Fractional Order PID Controller for Dynamic Response and Stability Enhancement of an Automatic Voltage Regulator System. *Electronics* **2019**, *8*, 1472. [CrossRef]
76. Gozde, H.; Taplamacioglu, M.C. Comparative Performance Analysis of Artificial Bee Colony Algorithm for Automatic Voltage Regulator (AVR) System. *J. Frankl. Inst.* **2011**, *348*, 1927–1946. [CrossRef]
77. Guvenc, U.; Yigt, T.; Isik, A.H.; Akkaya, I. Performance Analysis of Biogeography-Based Optimization for Automatic Voltage Regulator System. *Turk. J. Electr. Eng. Comput. Sci.* **2016**, *24*, 1150–1162. [CrossRef]
78. Sahu, B.K.; Panda, S.; Mohanty, P.K.; Mishra, N. Robust Analysis and Design of PID Controlled AVR System Using Pattern Search Algorithm. In Proceedings of the 2012 IEEE International Conference on Power Electronics, Drives and Energy Systems, Bengaluru, India, 16–19 December 2012; pp. 1–6.