mathematics

MDPI

Article

Levy Flight-Based Improved Grey Wolf Optimization: A Solution for Various Engineering Problems

Bhargav Bhatt [1], Himanshu Sharma [1], Krishan Arora [1], Gyanendra Prasad Joshi [2,*] and Bhanu Shrestha [3,*]

[1] Department of Electrical Engineering, Lovely Professional University, Jalandhar 144411, India; bhargav.12108438@lpu.in (B.B.); himanshu.23441@lpu.co.in (H.S.); krishan.12252@lpu.co.in (K.A.)
[2] Department of Computer Science and Engineering, School of Software Convergence, Sejong University, Seoul 05006, Republic of Korea
[3] Department of Electronic Engineering, Kwangwoon University, Seoul 01897, Republic of Korea
* Correspondence: joshi@sejong.ac.kr (G.P.J.); bnu@kw.ac.kr (B.S.)

Abstract: Optimization is a broad field for researchers to develop new algorithms for solving various types of problems. There are various popular techniques being worked on for improvement. Grey wolf optimization (GWO) is one such algorithm because it is efficient, simple to use, and easy to implement. However, GWO has several drawbacks as it is stuck in local optima, has a low convergence rate, and has poor exploration. Several attempts have been made recently to overcome these drawbacks. This paper discusses some strategies that can be applied to GWO to overcome its drawbacks. This article proposes a novel algorithm to enhance the convergence rate, which was poor in GWO, and it is also compared with the other optimization algorithms. GWO also has the limitation of becoming stuck in local optima when used in complex functions or in a large search space, so these issues are further addressed. The most remarkable factor is that GWO purely depends on the initialization constraints such as population size and wolf initial positions. This study demonstrates the improved position of the wolf by applying strategies with the same population size. As a result, this novel algorithm has enhanced its exploration capability compared to other algorithms presented, and statistical results are also presented to demonstrate its superiority.

Keywords: swarm intelligence (SI); metaheuristic algorithm; local search; global search; multi level inverter; total harmonic distortion (THD)

MSC: 68W99; 65K10

Citation: Bhatt, B.; Sharma, H.; Arora, K.; Joshi, G.P.; Shrestha, B. Levy Flight-Based Improved Grey Wolf Optimization: A Solution for Various Engineering Problems. *Mathematics* **2023**, *11*, 1745. https://doi.org/10.3390/math11071745

Academic Editors: Wanquan Liu, Xianchao Xiu and Xuefang Li

Received: 9 March 2023
Revised: 31 March 2023
Accepted: 3 April 2023
Published: 5 April 2023

1. Introduction

In the last few decades, the importance of swarm intelligence and metaheuristic algorithms has grown in the optimization community. The population-based approach is called swarm intelligence, which converges to produce optimal outcomes. The output of the metaheuristic changes over time and is problem-specific.

Studying swarm intelligence involves looking at the behavior of a decentralized system. It involves a large number of search agents communicating with each other and their environment to accomplish a shared objective [1]. This interdisciplinary field draws on concepts of biology, computer science, and physics to understand how complex behavior can emerge from the interactions of many simple individuals. The study of swarm intelligence has many potential applications, including robotics, distributed computing, and optimization. The concept of swarm intelligence originated from the work of pioneering biologists, such as William Morton Wheeler, who studied the behavior of ants in the early 20th century [2]. However, it was not until the 1950s and 1960s that computer scientists and engineers explored the potential of a decentralized system for solving complicated issues. Gerardo Beni and Jing Wang originally used the term "swarm intelligence" in 1989 to describe the collective behavior of simple agents [3]. Swarm intelligence can be observed

in a broad area of a natural system, such as in ant colonies, bird flocks, and fish schools [4]. Ant colonies, for example, can efficiently search for food and build complex nests through the interactions of individual ants. Bird flocks can perform coordinated maneuvers, such as turning in unison, through the interactions of individual birds. Fish schools can evade predators and locate food through the interactions of the individual fish.

Metaheuristic algorithms are classified into two classes, i.e., population-based and single solution-based. Optimization algorithms, known as population-based metaheuristic algorithms, use the population of a potential solution to iteratively search for the best response [5]. These algorithms frequently draw their inspiration from organic phenomena, including evolution, swarm behavior, and immune systems. Some of the population-based metaheuristic algorithms are the genetic algorithm (GA) [6], particle swarm optimization (PSO) [7], ant colony optimization (ACO) [8], differential evolution (DE) [9], and cultural algorithm (CA) [10]. Single solution-based metaheuristic algorithm start with a single candidate solution and then iteratively refine it by making small changes to it until the optimal solution is found or a stopping criterion is met. These algorithms often employ randomization and probabilistic search techniques to bypass local optima and completely investigate the search space. These algorithms frequently take their cues from physical processes such as annealing or movements in space. Some examples of single solution-based metaheuristic algorithms are simulated annealing (SA) [11], tabu search (TS) [12], and harmony search (HS) [13].

A class of optimization approaches known as metaheuristics is used to solve complex issues. They are called metaheuristics because they are a higher level of heuristics, meaning they are a set of heuristics that are used to guide the search for a solution [14]. These algorithms are known for their ability to find approximate solutions to problems that are difficult or impossible to solve exactly. They are extensively utilized in disciplines including operations research, computer science, and engineering. Metaheuristic algorithms first appeared in the 1940s and 1950s in the work of George Dantzig. Dantzig developed the simplex algorithm, which is still widely used today for solving linear programming problems [15]. However, metaheuristic algorithms are widely used in several fields. The most popular applications include:

- Engineering: Metaheuristic algorithms are used in engineering to optimize the design of structures, such as bridges and buildings.
- Computer science: Metaheuristic algorithms are used in computer science to optimize the performance of the algorithms and systems, such as scheduling and routing.
- Operations research: Metaheuristic algorithms are used in operations research to solve problems in areas such as logistics and supply chain management.

This paper proposes a new metaheuristic algorithm, Levy flight-based improved grey wolf optimization [16], to overcome the limitation that was faced by grey wolf optimization. GWO has several drawbacks as it is stuck in local minima, has a low convergence rate, and has poor exploration. To overcome these drawbacks, IGWO, i.e., improved grey wolf optimization, was proposed recently, but even IGWO did not provide satisfactory results, especially for complex engineering problems. Therefore, an improved version of this technique, viz. IGWO, is presented in this article with statistical analysis proving its superiority over the most popular optimization techniques proposed recently.

Section 2 discusses what optimization is and the different techniques of optimization. Section 3 presents grey wolf optimization, improved grey wolf optimization, and the newly presented technique LF-IGWO is discussed. Section 4 implements the technique on a benchmark function, and the results are evaluated in comparison to other approaches. Section 5 presents the implementation of the engineering problem and 31-level inverter problem.

2. Optimization

Finding the optimum solution to a problem in a list of possible alternatives is the process of optimization [16]. It is a fundamental concept in fields such as engineering, computer science, and operations research. There are many different optimization tech-

niques, each with its own special qualities and traits. A few well-known techniques include mathematical programming, gradient-based methods [17], and metaheuristic algorithms.

The work of mathematicians and engineers in the 19th century is where optimization first emerged. The first optimization methods were based on calculus and were used to solve problems in engineering and physics. However, optimization did not start to be used in other disciplines, including economics and computer science, until the 20th century.

There are many different optimization strategy types, each with unique characteristics. The most popular types include:

- Mathematical programming: These techniques are based on the use of mathematical models to represent the problem and constraints. The solution is then found by solving the mathematical equations.
- Gradient-based methods: These techniques are based on the use of gradients, which are the directions of the steepest ascent or descent. The solution is found by moving in the direction of the gradient until a local or global optimum is reached.
- Metaheuristic algorithm: These techniques are based on the use of metaheuristics, which are rules of thumb, to guide the search for a solution. The solution is found by iteratively improving an initial solution over time.

Many researchers are moving toward the optimization field because of the term no free lunch (NFL). The term NFL means that no technique or algorithm is suitable for all kinds of problems [18]. Therefore, there is a wide area to work on as each problem may lead towards developing new techniques.

In this optimization field, nature-inspired based optimization techniques play a significant role. They are defined as problem-solving methodologies that take inspiration from nature or a natural process [19]. These nature-inspired optimization techniques are further classified, as shown in Figure 1, and a summary of nature-inspired algorithms is given in Table 1.

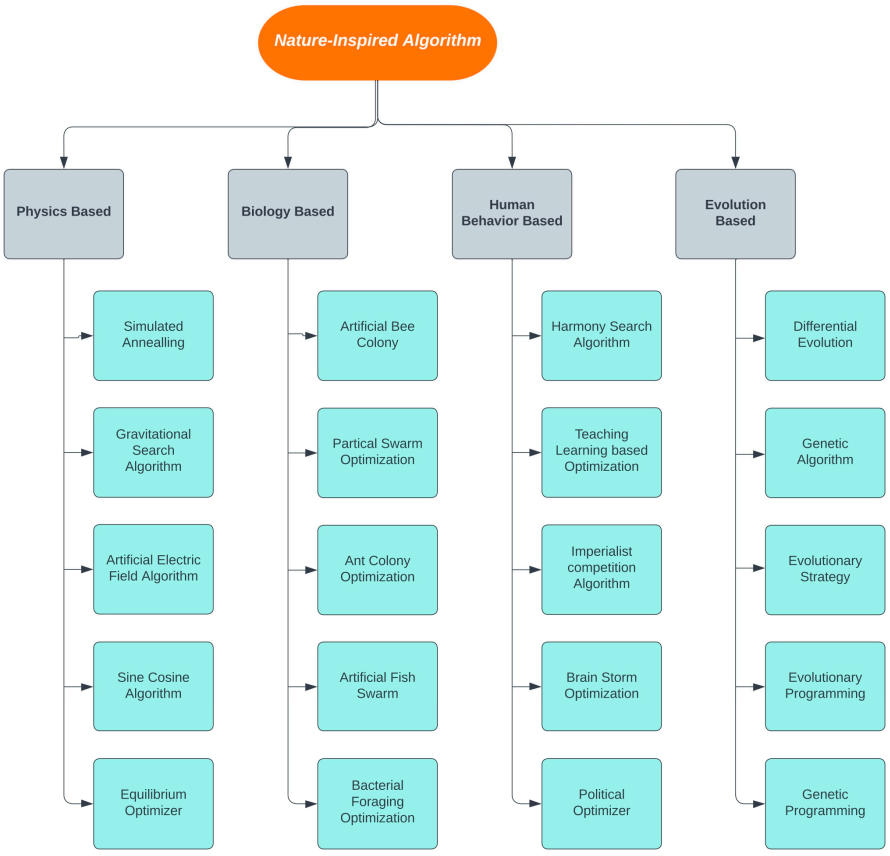


Figure 1. The hierarchical distribution of nature-inspired algorithms.

Table 1. Literature Review.

Optimization Methods	Description
Simulated Annealing	A probabilistic technique is used to identify a function's global optimum by iteratively perturbing a candidate solution and accepting it with a probability based on the temperature parameter. At each iteration, the algorithm compares the energy of the new approach and the existing solution and accepts the new solution if it is better than the current solution or with a probability that decreases with time [20].
Gravitational Search	The law of gravity and the interaction of masses served as inspiration for this population-based optimization system. The algorithm uses a set of masses, which represent candidate solutions that are attracted or repelled by each other based on their positions and masses [21].
Artificial Electric Field	The electrostatic force in physics served as the inspiration for this metaheuristic optimization approach. The algorithm represents each potential solution as a charged particle that interacts with other particles via the electrostatic force [22].
Sine Cosine Algorithm	The sine and cosine functions in mathematics served as inspiration for this metaheuristic optimization approach. Each potential solution is represented by a position vector in the SCA algorithm's high-dimensional search space. The technique creates random vectors that represent the search directions using the sine and cosine functions [23].
Equilibrium Optimizer	It is a metaheuristic optimization method that draws its motivation from physics' equilibrium concept. Each potential solution is modeled by the algorithm as a particle that interacts with other particles via the forces of gravity and elastic deformation [24].
Artificial Bee Colony	It is a metaheuristic optimization method that draws inspiration from honey bee feeding habits. Each bee in the ABC algorithm represents a potential solution to the optimization problem and represents a population of candidate solutions. Three different types of bees are used in the algorithm: working bees, observers, and scout bees [25].
Particle Swarm Optimization	It is a metaheuristic optimization system that draws on social behavior cues from flocks of birds or schools of fish. Each potential solution is represented by the algorithm as a particle in a multidimensional search space. The particles move across the search space, modifying their positions and velocities in response to their own experiences as well as those of their nearby neighbors [7].
Ant Colony Optimization	It is a metaheuristic optimization method that takes its cues from how ants forage. Each ant in the algorithm's representation of a population of potential solutions as an ant colony stands for a potential solution to the optimization issue. The algorithm mimics the actions of ants as they look for food, with the food serving as the ideal answer to the issue [8].
Artificial Fish Swarm	It is a metaheuristic optimization technique that draws its inspiration from how fish forage. Each fish in the algorithm's representation of a population of potential solutions is a potential solution to the optimization issue. The algorithm mimics the actions of fish as they swim and search for food, with the food serving as the ideal answer to the issue [26].
Bacterial Foraging Optimization	It is a metaheuristic optimization algorithm that draws inspiration from how bacteria forage. Each bacterium in the algorithm's model of a population of candidate solutions serves as a potential solution to the optimization problem. The algorithm mimics how bacteria scavenge for nutrition, with the nutrients serving as the ideal solution to the issue [27].
Harmony Search Optimization	It is a metaheuristic algorithm that draws inspiration from the process of musical improvisation. The method searches for a function's global optimum using a population-based approach. The algorithm selects elements from the already-existing solutions and randomly adds some randomness to them in order to produce a new harmony at each iteration. A memory-based method is also incorporated into the algorithm to speed up the search's convergence [13].
Teaching Learning-based Optimization	It is a metaheuristic optimization method that draws its inspiration from classroom teaching and learning procedures. Each student in the algorithm's representation of a population of candidate solutions serves as a potential solution to the optimization problem. The algorithm mimics how students act as they interact with the teacher and one another and learn new things [28].
Imperialist Competition Algorithm	The social rivalry and hierarchical organization principles serve as the foundation for this metaheuristic optimization method. A population of potential solutions is modeled by the algorithm as a collection of empires, where each empire consists of one imperialist and one or more colonies. The algorithm mimics how empires act as they compete and work together to increase their influence and power [29].
Brain Storm Optimization	It is a metaheuristic optimization algorithm that draws inspiration from how human brain neurons behave. The programmer simulates the brainstorming process, in which a group of people come up with and assess solutions to a problem. Each person in BSO represents a potential solution to the optimization problem, and the algorithm adjusts each person's position based on how they interact with other people [30].

Table 1. *Cont.*

Optimization Methods	Description
Political Optimizer	It is a metaheuristic optimization method that draws inspiration from how politicians act in a given political system. The algorithm mimics the process of political rivalry, in which politicians face off against one another and work together to accomplish their objectives. The algorithm in PO adjusts the positions of the politicians depending on their interactions with other politicians in the population. Each possible solution to the optimization issue is represented in PO as a politician [31].
Differential Evolution	For the purpose of resolving optimization issues, it is a stochastic optimization algorithm. DE is a population-based method that uses natural selection to gradually weed out suboptimal solutions from a population of candidate solutions. The fundamental strategy is to generate a population of potential solutions, referred to as individuals, and then develop them utilizing the three crucial operators of mutation, crossover, and selection [9].
Genetic Algorithm	It is a metaheuristic optimization technique that draws inspiration from the evolution and natural selection processes. A population of potential solutions, or people, is created, and genetic operators such as crossover and mutation are used to gradually evolve them over generations. Every member of the population stands for a potential answer to the optimization issue [6].
Evolutionary Strategy	It draws inspiration from the course of evolution and natural selection. However, there are some significant ways in which ES is different from GA. The fundamental tenet of ES is to generate a population of potential solutions, referred to as individuals, and evolve them through mutation and selection across generations. Every member of the population is a potential answer to the optimization problem.
Evolutionary Programming	Similar to evolutionary strategy (ES) and the genetic algorithm (GA), it is a family of optimization algorithms that draws its inspiration from the processes of natural selection and evolution. The fundamental tenet of EP is to generate a population of potential solutions, or individuals, and to use mutation and selection to gradually evolve them over generations. Every member of the population is a potential answer to the optimization problem.
Genetic Programming	It is a machine learning technique that uses a form of evolutionary computation to automatically discover computer programs that solve a problem. GP is a variant of the genetic algorithm (GA) and evolutionary programming (EP), but instead of evolving vectors or individuals, it evolves computer programs represented as trees.
Optimization of Meat and Poultry Farm Inventory Stock Using Data Analytics for Green Supply Chain Network	Optimizing inventory stock in meat and poultry farms is important for maintaining a sustainable and efficient supply chain network. Data analytics can be employed to analyze various factors that affect inventory levels, such as demand, production capacity, and supply chain lead time, to optimize inventory stock levels. The optimization process involves creating a model that considers various factors such as demand patterns, production schedules, and storage capacity. The model is trained using historical data on inventory levels, sales, and other relevant metrics. The trained model can then be used to predict the optimal inventory levels for each item in the meat and poultry farm [32].
Optimum Design for the Magnification Mechanisms Employing Fuzzy Logic–ANFIS	To optimize the design of a centrifugal pump, fuzzy logic and ANFIS (Adaptive Neuro-Fuzzy Inference System) can be employed. Fuzzy logic is a mathematical technique that deals with uncertainty and imprecision in data and is commonly used in control systems. ANFIS is a type of fuzzy inference system that uses neural networks to model the fuzzy logic. The design process involves various parameters such as impeller diameter, number of blades, blade angle, and outlet diameter, which need to be optimized to achieve the desired performance [33].
Minimizing Warpage for Macro-sized Fused Deposition Modeling Parts	There are several methods to minimize warpage in macro-sized FDM parts. The first method is to optimize the design of the part. The design should be modified to avoid features that are susceptible to warping, such as sharp corners, thin walls, and unsupported overhangs. Additionally, the part should be designed with proper wall thickness and infill density to ensure structural integrity and dimensional stability. Minimizing warpage in macro-sized FDM parts involves optimizing the part design, printing process parameters, and support structures, as well as using a heated build platform. By implementing these methods, high-quality parts with minimal warpage can be achieved [34].
Optimal Switching Angle Scheme for a Cascaded H-Bridge Inverter using Pigeon-Inspired Optimization	A cascaded H-bridge inverter is a type of multilevel inverter that is widely used in high-power applications such as electric vehicles, renewable energy systems, and industrial motor drives. It is made up of several H-bridge modules connected in series to produce a stepped waveform output. Each H-bridge module is made up of four power switches (IGBTs or MOSFETs) and a DC voltage source. The switches are controlled with the help of firing angles to create a sinusoidal waveform output [35].

3. Optimization Algorithm and New Proposed Algorithms

The optimization algorithms used in swarm intelligence place a strong emphasis on the social relationships and interactions that occur between members of the swarms as they search for and pursue food sources. Several SI algorithms have been created and suggested over the past few years. One of the most well-known and frequently employed SI-based methods is Grey Wolf Optimization (GWO). The grey wolf's natural behavior

of looking for the most effective way to pursue prey served as the model for the GWO algorithm. This led to a good exploration–exploitation balance. A limitation of this is that it suffers from poor performance in global search. To eliminate this limitation, improved grey wolf optimization (IGWO) was proposed with the help of the dimension learning-based hunting (DLH) search strategy. The advantage of this is that GWO performs better in multi-dimensional functions but is not too efficient in uni-dimensional functions. To resolve this uni-dimensional issue, a new method is proposed in this paper, i.e., Levy flight-based improved grey wolf optimization (LF-IGWO). In this section, GWO, IGWO, and LF-IGWO are discussed.

3.1. Grey Wolf Optimization

The metaheuristic-based grey wolf optimization (GWO) technique was influenced by the communal exploration actions of grey wolves [36]. To identify a given problem's global optimum, the program imitates the hierarchy of command and the hunting style of grey wolves. GWO has been employed to address several difficulties in optimization, including those involving machine learning, image processing, and function optimization.

There are three basic steps in the GWO algorithm: initialization, iteration, and update. The initial population is produced at random during the startup process. Every member of this population is given a fitness value during the iteration phase. In the update step, the individuals are updated based on their fitness values. Leaders are selected from those with the highest fitness rating, and the others follow them to update their positions.

One of the advantages of GWO is its simplicity. In contrast to other optimization algorithms, GWO does not need any settings to be pre-set. Additionally, compared to other methods, such as differential evolution (DE) and particle swarm optimization (PSO), GWO has been demonstrated to converge more quickly and create better solutions.

Numerous optimization issues, such as function optimization, image processing, and machine learning, have been tackled with GWO. In function optimization, GWO has been used to optimize benchmark functions. In image processing, GWO has been used for image enhancement, segmentation, and compression. In machine learning, GWO has been used for feature selection, classification, and clustering.

Despite its successes, GWO is not without its limitations. One limitation is that GWO is sensitive to the original population and could reach a local optimum if it is not sufficiently diversified. Another limitation is that GWO may not perform well on problems with a high number of variables or constraints.

Figure 2 shows the hierarchy level of the grey wolves. This is the population-based algorithm that signifies that there is a group of wolves and they are divided into different levels based on their work or task.

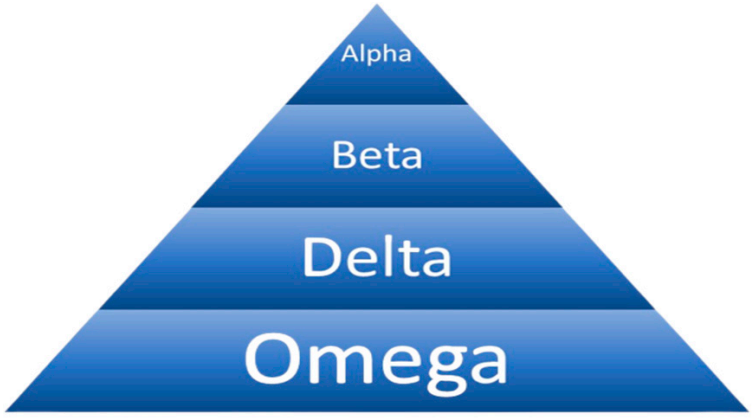


Figure 2. The hierarchy distribution of the grey wolf.

The dominating wolves in this situation are those from alpha packs, and it can be said that they are the group's leaders. They give us the GWO algorithm's best-fitting solution. These groups of wolves make choices regarding hunting, planning, and delegating specific tasks to other wolves in their pack [36]. These wolves are thought to be the strongest in their pack.

Beta is the group's next level. They give us the value or solution that is best suited to our needs after the alpha. This group's responsibility is to guide the alpha group's decision-making and to rule the two groups that follow it.

Omega is the lowest category. They act out the part of the victim. Their solutions have no bearing on the GWO algorithm's final result [37]. They are given their opportunities in the end of the assignment. For example, after hunting, they are permitted to eat last after all other wolves have finished, because they are at the bottom of the pack and must follow the order of their dominant wolf. Below is the pseudo code for GWO [37] in Algorithm 1:

Algorithm 1 GWO Algorithm

Form the grey wolf population.
Evaluate the accuracy of every response.
For each iteration:
For every single grey wolf in the population:
Generate a new solution.
Analyze the new solution's potential.
Update a grey wolf's location in accordance with the updated solution.
Sort the grey wolves based on their fitness.
Each grey wolf's location will be updated dependent on where the other grey wolves in the population are.
Return the best solution as the result.

3.2. Improved Grey Wolf Optimization

In GWO, the omega wolves are led by the α, β, and γ wolves to the areas of the search space where it is most likely that the ideal solution will be found. This behavior could keep you in a locally ideal solution. A decline in population diversity, which causes GWO to enter the regional optimum, is another adverse effect. Improved grey wolf optimization was introduced to address this problem.

IGWO algorithms typically involve modifications to the original GWO algorithm in one or more areas such as initialization, updating steps, and adaptation.

In the initialization phase, IGWO algorithms may use a different method for initializing the population, such as using a combination of random initialization [6].

$$X_{ij} = l_j + rand_j\,[0,1] \times \left(u_j - l_j\right),\ i\ \in [1,N],\ j \in [1,D] \tag{1}$$

Movement phase: An additional movement technique that is part of I-GWO is the dimension learning-based hunting (DLH) search method [38]. The wolves in DLH are aware of each other as possible candidates for the new role [39].

Dimension learning-based hunting (DLH) search strategy: Originally, in GWO, three pop leader wolves are used to produce a new position for each wolf. By doing this, the population loses diversity too soon, the GWO displays delayed convergence, and the wolves become caught in the local optimum. The suggested DLH search approach takes these flaws into account and includes individual wolf hunting that neighbors can observe.

In comparison to the original GWO algorithm, IGWO algorithms have been shown to have the following advantages in Algorithm 2:

Algorithm 2 IGWO Algorithms

- Better convergence: IGWO algorithms have been found to converge faster and produce better solutions than the original GWO algorithm.
- Better handling of constraints: IGWO algorithms have been found to perform better on problems with constraints.
- Better handling of high-dimensional problems: IGWO algorithms have been found to perform better on problems with a vast array of variables.

The pseudo code is provided below [39]:
Set the grey wolf population (solutions).
Evaluate the accuracy of every response.
Arrange solutions in decreasing fitness order.
Make the best response the dominant α wolf.
Make the second best response the β wolf.
Make the third best response the δ wolf.
For each iteration:
Generate new solutions for each grey wolf.
Examine the appropriateness of each new solution.
According to the new answer, adjust each grey wolf's location.
Based on a combination of the α wolf's answer, the β wolf's solution, and the δ wolf's solution, the α wolf's location should be updated.
Using the α wolf's location and their solutions, adjust the positions of the wolves.
Evaluate the fitness of the updated solutions.
Arrange the grey wolves according to their most recent fitness.
Update the α, β, and δ wolves based on the new sorting.
Return the best solution (the alpha wolf) as the result.

3.3. Levy Flight Improved Grey Wolf Optimization

Levy flight is a sort of unplanned walk in which walkers' step sizes are drawn from a Levy distribution, which is a probability distribution with heavy tails. This means that the distribution has a high probability of generating large steps, and these large steps occur more frequently than they would in a normal distribution [40].

The Levy distribution is characterized by a power law tail, and its probability density function can be expressed as:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, exp\Big(-\frac{\gamma}{2\sigma^2}\,|x-\mu|\Big)\,\frac{\gamma}{|x-\mu|^{1+\gamma}} \tag{2}$$

where:

- μ is the parameter for location (the mean of the distribution);
- σ is a scale parameter (the standard deviation of the distribution);
- γ is the tail index parameter (controls the shape of the distribution).

The tail index parameter γ determines the shape of the distribution. When γ is between 0 and 2 [36], the distribution has infinite variance, which means that the variance of the distribution is undefined. When γ is greater than 2, the distribution has finite variance, and when γ is less than or equal to 1, the distribution has an infinite mean.

With the help of this technique, the random position is chosen and updated to improve the exploration capability. The equation for the finding the position of the wolf is:

$$step = \frac{u.}{abs(v).^{1/\beta}} \tag{3}$$

$$stepsize = 0.001 \times step. \times (s) \tag{4}$$

$$s = s + stepsize. \times randn(size(s)) \tag{5}$$

$$position = s. \times (\sim (Flagub + Flaglb)) + ub. \times Flagub + lb. \times Flaglb \tag{6}$$

With the help of these equations, the new position is initialized and given to the wolves. The algorithm updates each wolf's position after each iteration using the following (Equation (7)) [41]:

For the alpha wolf:

$$x_i(t+1) = x_i(t) - A_j.D_j \tag{7}$$

Similarly, the next position of β, δ, and ω is calculated using varying distance vector D. Here, $x_i(t)$ is the current position of the ith wolf at time t, $x_i(t+1)$ is the updated position of the ith wolf at time $t+1$, A_j is the scaling factor for the jth wolf, and D_j is the distance vector between the ith wolf and the jth wolf. The scaling factors A_j are updated in each iteration as follows:

$$A_j = \frac{2 \times (1-u) \times a - u}{2} \tag{8}$$

where u is a uniform random number in the range [0, 1], a is the current iteration number, and j corresponds to the alpha, beta, delta, and omega wolves.

The distance vector D_j is calculated as follows:

$$D_j = \left| C_j \times x_j - x_i \right| \tag{9}$$

where C_j is a random vector in the range [0, 1] that is generated for each wolf in each iteration.

The algorithm terminates when a stopping criterion is met, such as a maximum number of iterations or a minimum level of improvement in the objective function [42]. The pseudo code for the algorithm is given below in Algorithm 3:

Algorithm 3 Newly Proposed

Set the grey wolf population (solutions).
Evaluate the accuracy of every response.
Arrange solutions in decreasing fitness order.
Make the best response the dominant α wolf.
Make the second best response the β wolf.
Make the third best response the δ wolf.
Evaluate the new position of the wolves and update it with the initial position.
For each iteration:
Generate new solutions for each grey wolf using Levy flight.
Evaluate the accuracy of every new solution.
According to the new answer, adjust each grey wolf's location.
The α wolf's location is updated based on a combination of its solution and the solutions of the β and δ wolves.
Based on a combination of the α wolf's answer, the β wolf's solution, and the δ wolf's solution, the α wolf's location should be updated.
Evaluate the fitness of the updated solutions.
Arrange the grey wolves according to their most recent fitness.
Update the α, β, and δ wolves based on the new sorting.
Return the best solution (the α wolf) as the result.

4. Results and Discussion

Through the resolution of the classical benchmark problem, the suggested approach is validated in this section. A total of 23 sets of well-known functions make up the classical benchmark function and the CEC 2017 benchmark functions. In this classical set of functions, F1 through F7 are uni-modal functions, F8–F13 are multi-modal functions, and the last ten functions, i.e., F14–F23, are fixed dimensional functions [43]. Detailed information on this classical benchmark function with dimension and range is given in Tables 2–4. The minima of these 23 traditional benchmark functions are aimed for by the novel algorithm that has been proposed. This is performed on other well-known algorithms such as PSO, GA, GWO, and I-GWO to compare with the LF-IGWO algorithm.

Table 2. Unimodal benchmark function equations.

Function Name	Equation	Range	Dim	f_{min}
$F_1(x)$	$\sum_{t=1}^{n} x_i^2$	$[-100, 100]$	30	0
$F_2(x)$	$\sum_{t=1}^{n} \lvert x_i \rvert + \prod_{t=1}^{n} x_i$	$[-10, 10]$	30	0
$F_3(x)$	$\sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	$[-100, 100]$	30	0
$F_4(x)$	$max_i\{\lvert x_i \rvert,\ 1 \le i \le n\}$	$[-100, 100]$	30	0
$F_5(x)$	$\sum_{t=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	$[-30, 30]$	30	0
$F_6(x)$	$\sum_{i=1}^{n} (x_i + 0.5)^2$	$[-100, 100]$	30	0
$F_7(x)$	$\sum_{i=1}^{n} ix_i^4 + random[0, 1]$	$[-1.28, 1.28]$	30	0

Table 3. Multimodal benchmark function equations.

Function Name	Equation	Range	Dim	f_{min}
$F_8(x)$	$\sum_{i=1}^{n} -x_i \sin(\sqrt{x_i})$	$[-500, 500]$	30	418.9829 ×D
$F_9(x)$	$\sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]$	30	0
$F_{10}(x)$	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$	30	0
$F_{11}(x)$	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	30	0
$F_{12}(x)$	$\frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10sin^2(\pi y_i + z)\right] + (y_{n-1})^2\} +$ $\sum_{i=1}^{n} u(x_i,\ 10,\ 100,\ 4)$ $y_i = 1 + \frac{x_i+1}{4},\ u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m,\ x_i > a \\ 0,\ -a < x_i < a \\ k(-x_i - a)^m,\ x_i < a \end{cases}$	$[-50, 50]$	30	0
$F_{13}(x)$	$0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + sin^2(2\pi x_n)\right]\right\} +$ $\sum_{i=1}^{n} u(x_i,\ 5,\ 100,\ 4)$	$[-50, 50]$	30	0

Table 4. Fixed dimensional benchmark function equations.

Function Name	Equation	Range	Dim	f_{min}
$F_{14}(x)$	$\left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	$[-65, 65]$	2	1
$F_{15}(x)$	$\sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5, 5]$	4	0.00030
$F_{16}(x)$	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]$	2	−1.0316
$F_{17}(x)$	$\left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\cos x_1 + 10\right)$	$[-5, 5]$	2	0.398
$F_{18}(x)$	$\left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$ $\left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$[-2, 2]$	2	3
$F_{19}(x)$	$-\sum_{i=1}^{4} c_i\ exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[1, 3]$	3	−3.86
$F_{20}(x)$	$-\sum_{i=1}^{4} c_i\ exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]$	6	−3.32
$F_{21}(x)$	$-\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	$[0, 10]$	4	−10.1532
$F_{22}(x)$	$-\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	$[0, 10]$	4	−10.4028
$F_{23}(x)$	$-\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	$[0, 10]$	4	−10.5363

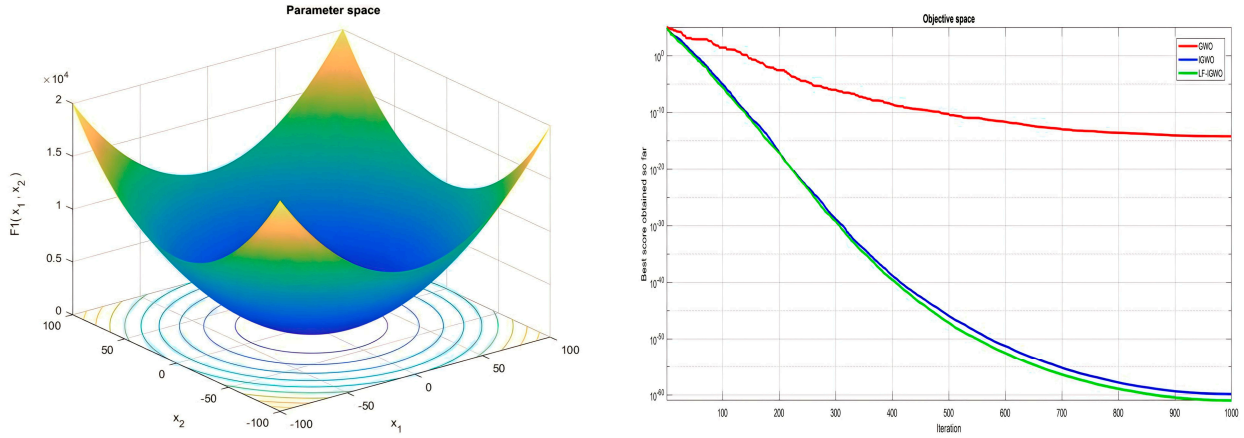
For a fair comparison with all these techniques, the size of the population used is 30 and each algorithm is run 20 times. Table 5 provides the specified parameters. Table 6 compares how these strategies' means compare to one another when applied to the benchmark function. The best value among all the functions is highlighted in bold letters. The functions F1 to F7 are used to test the exploitation capabilities. Apart from functions F5 and F7, the new technique LF-IGWO performs better than other algorithms in these. From function F8 to F13, there are many local minima, so the technique should have exploration capability. Apart from F8, the new technique LF-IGWO performed better than other techniques in these. Additionally, from function F14 to F23, all the algorithms gave almost the same result, except for F14 and F15. Therefore, it can be said that the overall performance of the new technique is more outstanding than the others. The benchmark function model and results of these functions are depicted in Figures 3–25.

Table 5. Selection of parameters for different algorithms [35,41].

Algorithms	Parameters
LF-IGWO	a linearly declines from 2 to 0
IGWO	a linearly declines from 2 to 0
GWO	a linearly declines from 2 to 0
PSO	w decreases linearly from 0.9 to 0.2, and $c1 = c2 = 2$.
GA	0.3 is the crossover probability, while 0.1 is the mutation probability
SSR	$R_f = 15$ and $M = 5$.

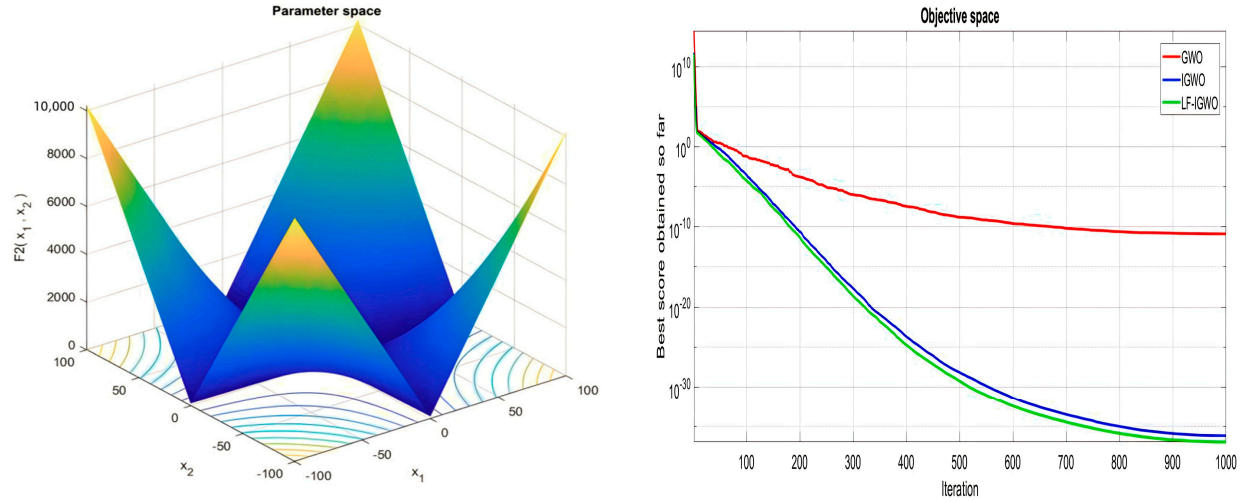
Table 6. Comparison of mean values obtained from classical benchmark functions.

Function Number	PSO	GA	GWO	IGWO	SSR	LF-IGWO
F1	3.80×10^{-8}	2.31×10^{1}	4.96×10^{-14}	2.67×10^{-60}	6.35×10^{-9}	7.37×10^{-61}
F2	4.80×10^{-8}	1.07×10^{0}	2.40×10^{-11}	2.25×10^{-37}	3.52×10^{-5}	7.72×10^{-38}
F3	1.53×10^{1}	5.60×10^{3}	1.86×10	1.08×10^{-10}	1.82×10^{-1}	9.07×10^{-15}
F4	6.05×10^{-1}	1.58×10^{-1}	1.08×10	2.08×10^{-11}	3.03×10^{-5}	5.56×10^{-13}
F5	6.03×10^{1}	1.18×10^{1}	28.93	23.36	1.02×10^{2}	22.65
F6	3.69×10^{-8}	1.10×10^{3}	4.29	6.98×10^{-6}	6.29×10^{-9}	3.65×10^{-6}
F7	7.07×10^{-2}	1.01×10^{-2}	7.90×10^{-4}	1.92×10^{-3}	1.01×10^{-2}	6.61×10^{-4}
F8	-6.06×10^{2}	-2.09×10^{3}	-6.07×10^{3}	-5.58×10^{3}	-6.84×10^{3}	-1.02×10^{4}
F9	4.67×10^{3}	6.59×10^{-1}	23.30×10	2.54×10^{1}	4.77×10^{1}	7.96×10^{0}
F10	7.33×10^{-2}	9.56×10^{-1}	1.67×10^{-8}	1.51×10^{-14}	1.86×10^{-5}	7.99×10^{-15}
F11	9.28×10^{-3}	4.88×10^{-1}	6.89×10^{-13}	0	1.38×10^{-3}	0
F12	7.26×10^{-3}	1.11×10^{-1}	6.42×10^{-1}	2.67×10^{-7}	1.45×10^{-11}	1.96×10^{-7}
F13	2.53×10^{-3}	1.29×10^{-1}	12.67×10	9.72×10^{-2}	2.24×10^{-10}	8.98×10^{-6}
F14	3.46×10^{0}	1.26×10^{0}	7.87×10^{0}	9.98×10^{-1}	1.16×10^{0}	9.98×10^{-1}
F15	8.94×10^{-4}	4.00×10^{-3}	4.54×10^{-4}	3.07×10^{-4}	1.48×10^{-4}	3.07×10^{-4}
F16	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}
F17	3.99×10^{-1}	4.00×10^{-1}	3.98×10^{-1}	3.97×10^{-1}	3.98×10^{-1}	3.097×10^{-1}
F18	3.00×10^{0}	5.70×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}
F19	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}
F20	-3.27×10^{0}	-3.31×10^{0}	-3.24×10^{0}	-3.32×10^{0}	-3.32×10^{0}	-3.32×10^{0}
F21	-8.60×10^{0}	-5.66×10^{0}	-5.05×10^{0}	-10.15×10^{0}	-8.60×10^{0}	-10.15×10^{0}
F22	-9.07×10^{0}	-7.34×10^{0}	-10.39×10^{0}	-10.40×10^{0}	-1.04×10^{1}	-10.40×10^{0}
F23	-9.20×10^{0}	-6.25×10^{0}	-1.04×10^{1}	-1.05×10^{1}	-1.05×10^{1}	-1.05×10^{1}



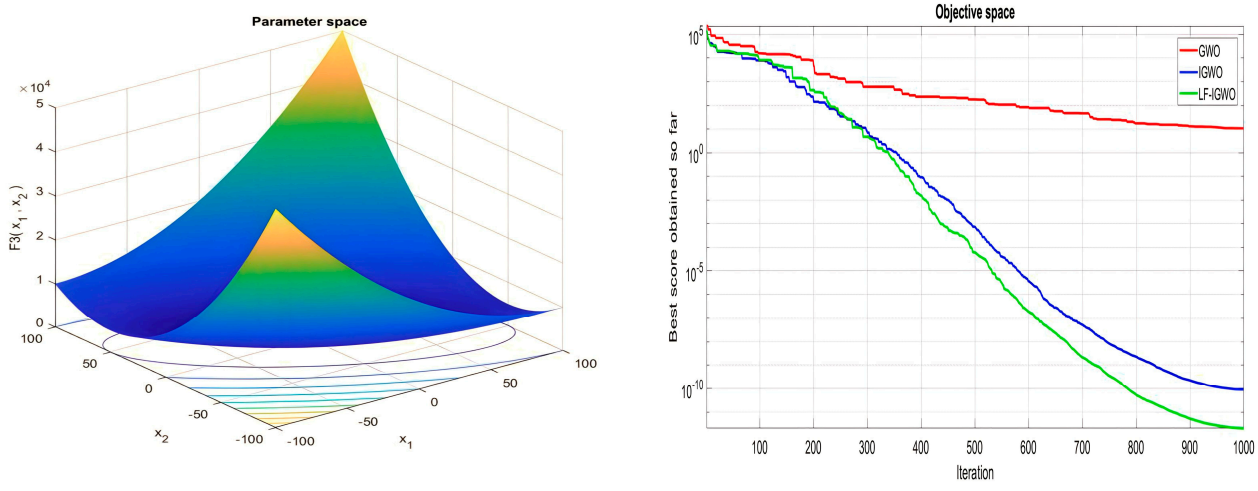
(**a**) (**b**)

Figure 3. (**a**) Search space for unimodel function F01 (**b**) Comparison of convergence curve for F01.



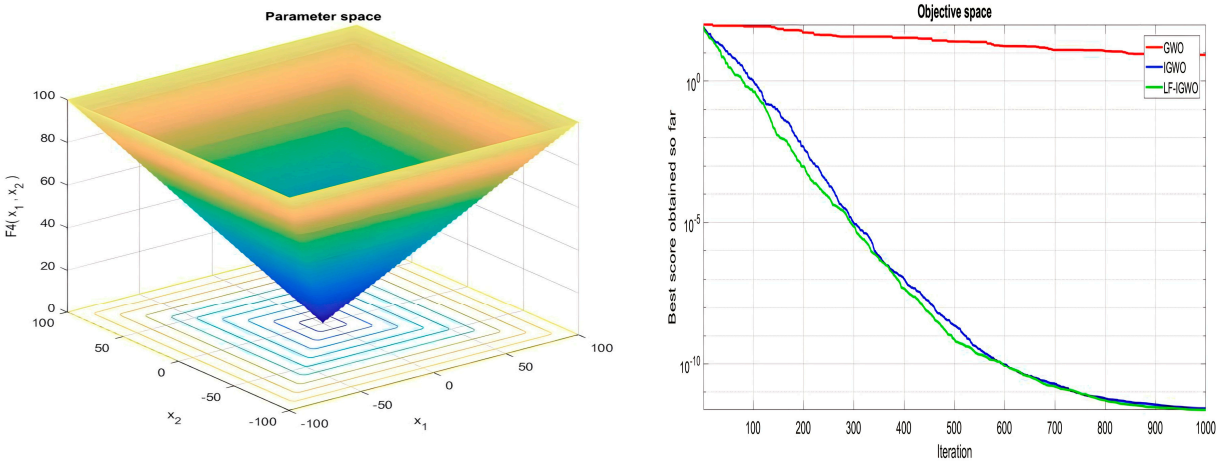
(**a**) (**b**)

Figure 4. (**a**) Search space for unimodel function F02 (**b**) Comparison of convergence curve for F02.



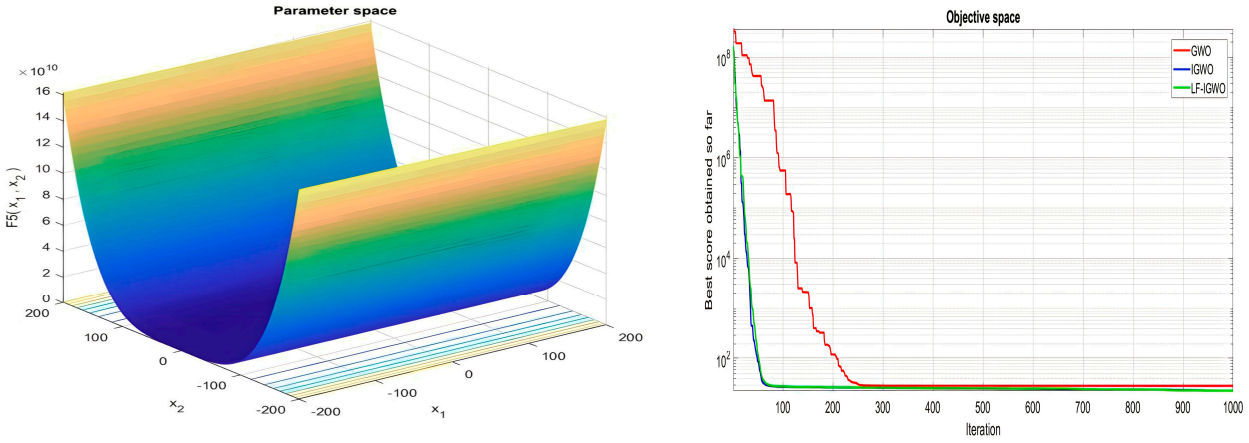
(**a**) (**b**)

Figure 5. (**a**) Search space for unimodel function F03 (**b**) Comparison of convergence curve for F03.



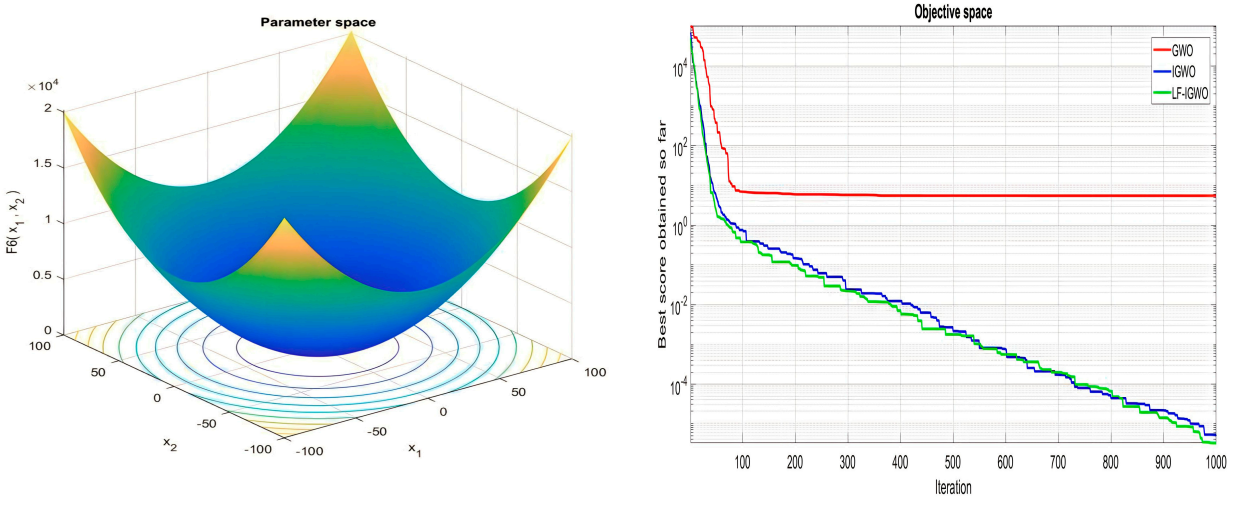
(**a**) (**b**)

Figure 6. (**a**) Search space for unimodel function F04 (**b**) Comparison of convergence curve for F04.



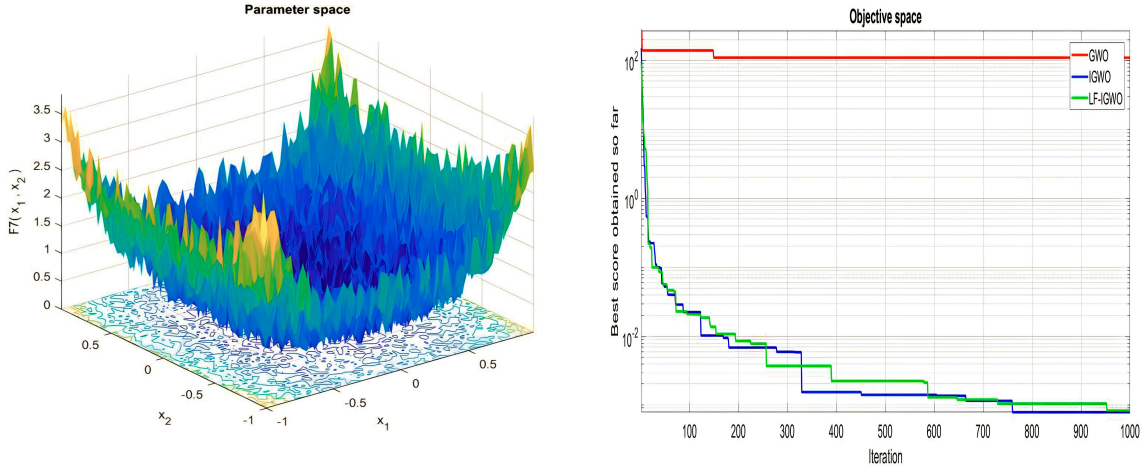
(**a**) (**b**)

Figure 7. (**a**) Search space for unimodel function F05 (**b**) Comparison of convergence curve for F05.



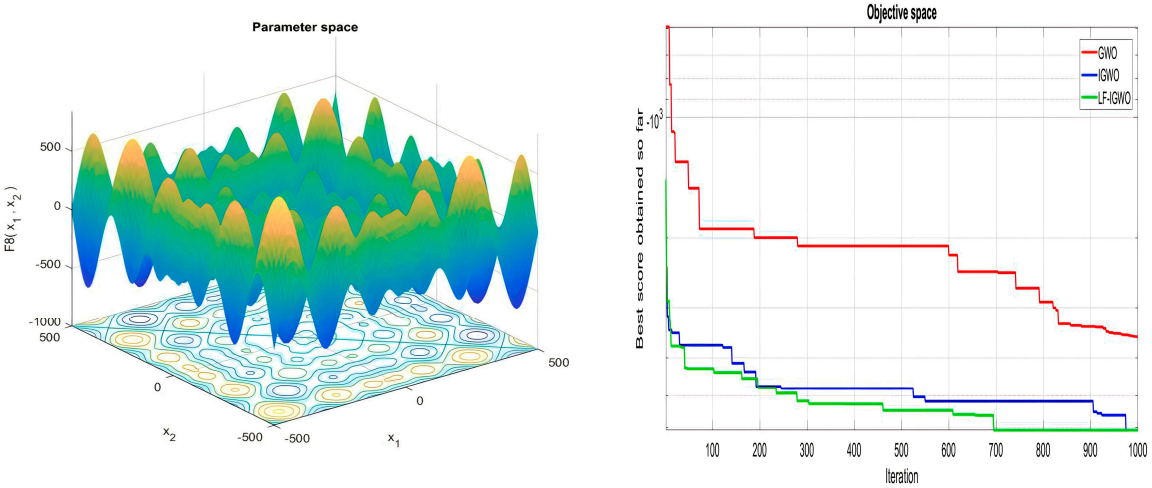
(**a**) (**b**)

Figure 8. (**a**) Search space for unimodel function F06 (**b**) Comparison of convergence curve for F06.



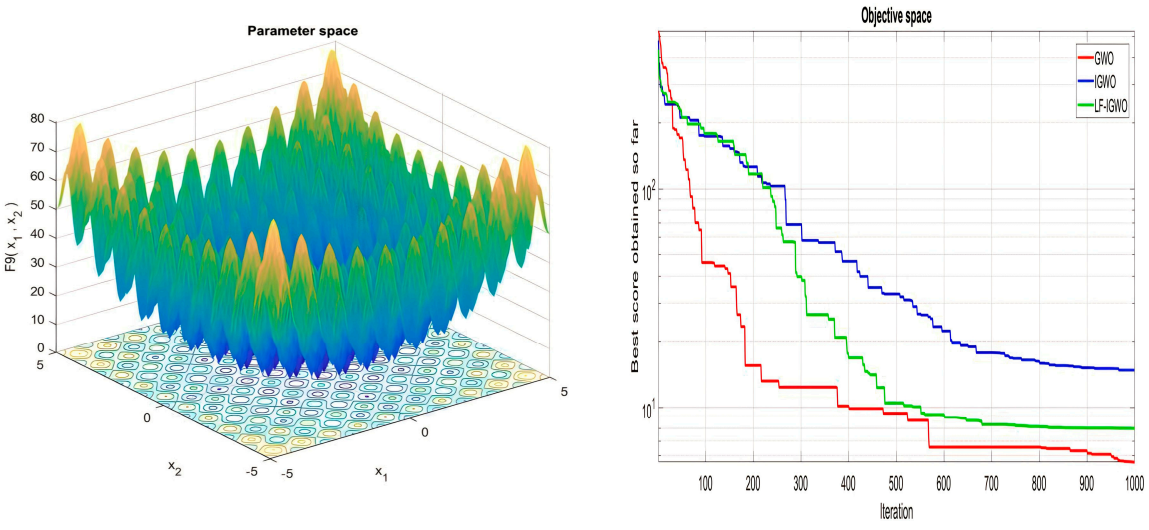
(**a**) (**b**)

Figure 9. (**a**) Search space for unimodal function F07 (**b**) Comparison of convergence curve for F07.



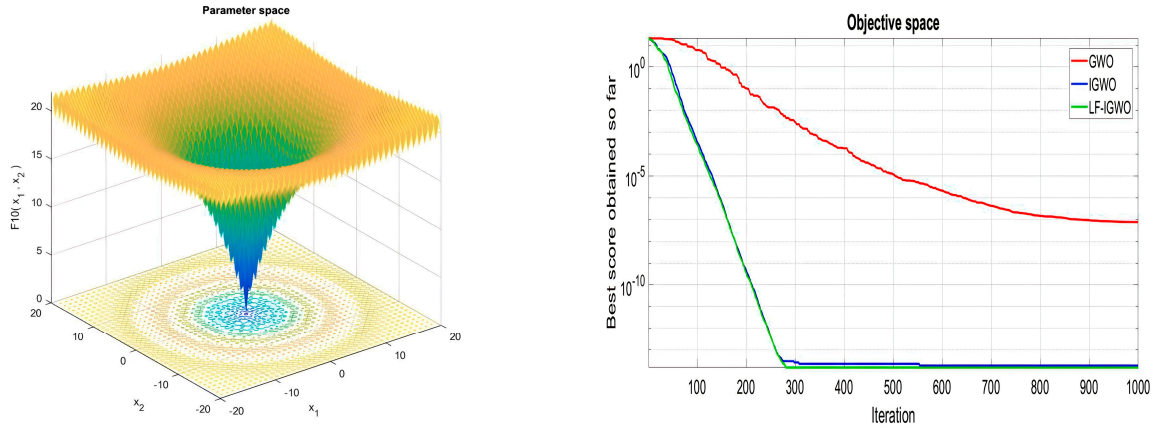
(**a**) (**b**)

Figure 10. (**a**) Search space for multi model function F08 (**b**) Comparison of convergence curve for F08.



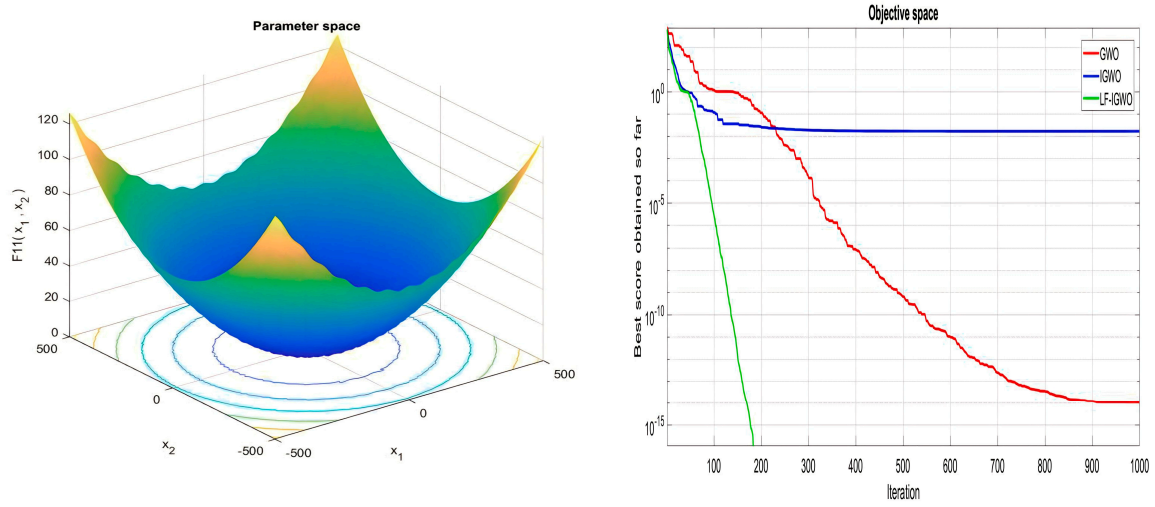
(**a**) (**b**)

Figure 11. (**a**) Search space for multi model function F09 (**b**) Comparison of convergence curve for F09.



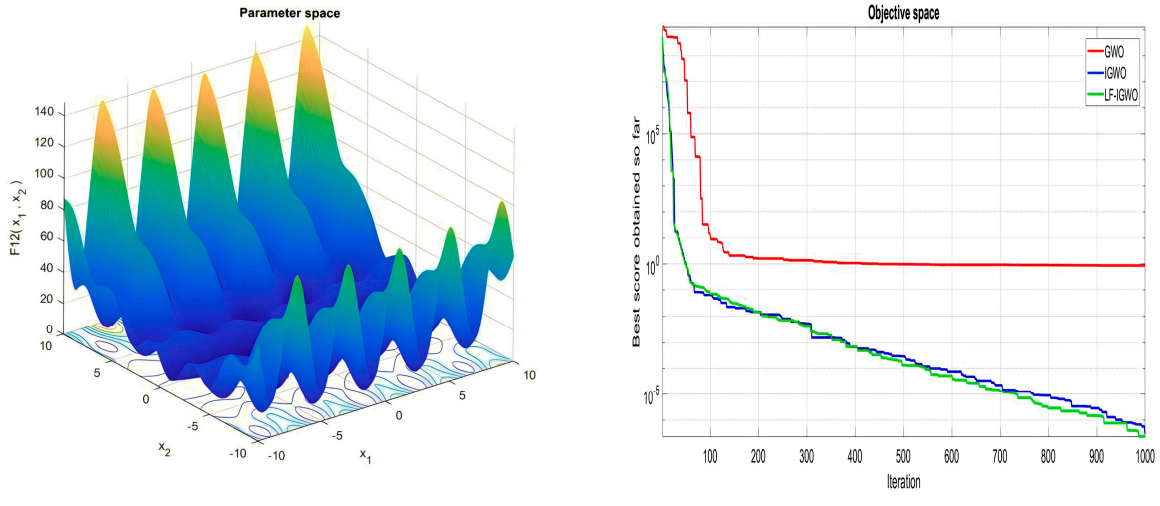
(**a**) (**b**)

Figure 12. (**a**) Search space for multi model function F10 (**b**) Comparison of convergence curve for F10.



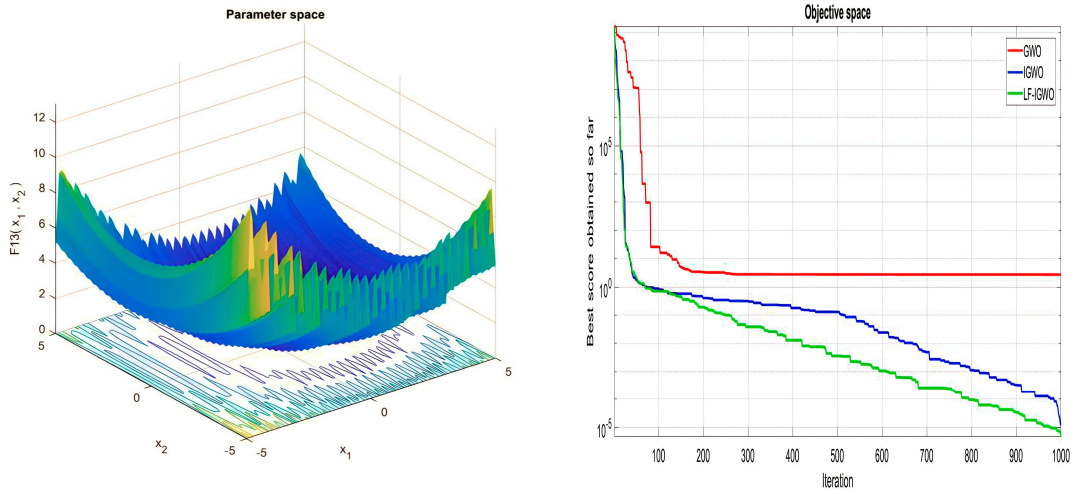
(**a**) (**b**)

Figure 13. (**a**) Search space for multi model function F11 (**b**) Comparison of convergence curve for F11.



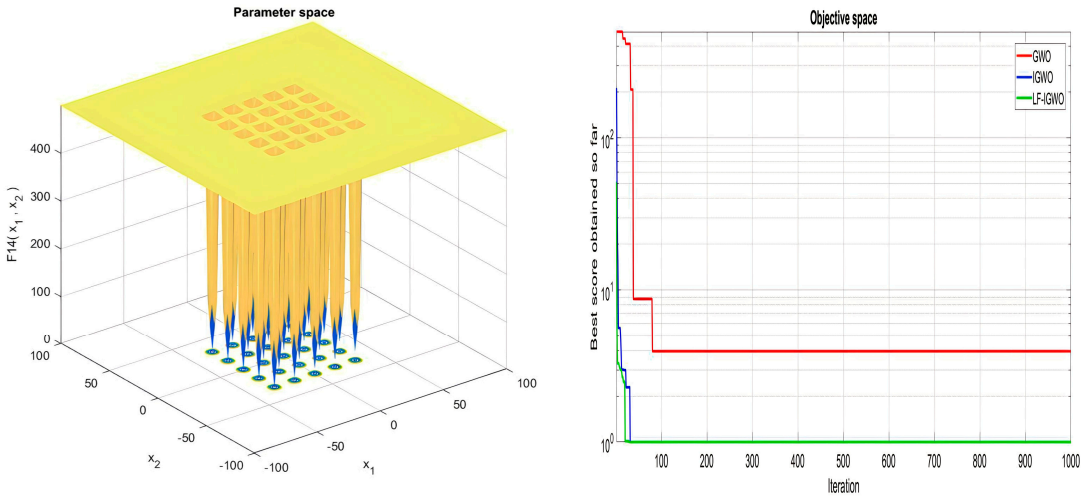
(**a**) (**b**)

Figure 14. (**a**) Search space for multi model function F12 (**b**) Comparison of convergence curve for F12.



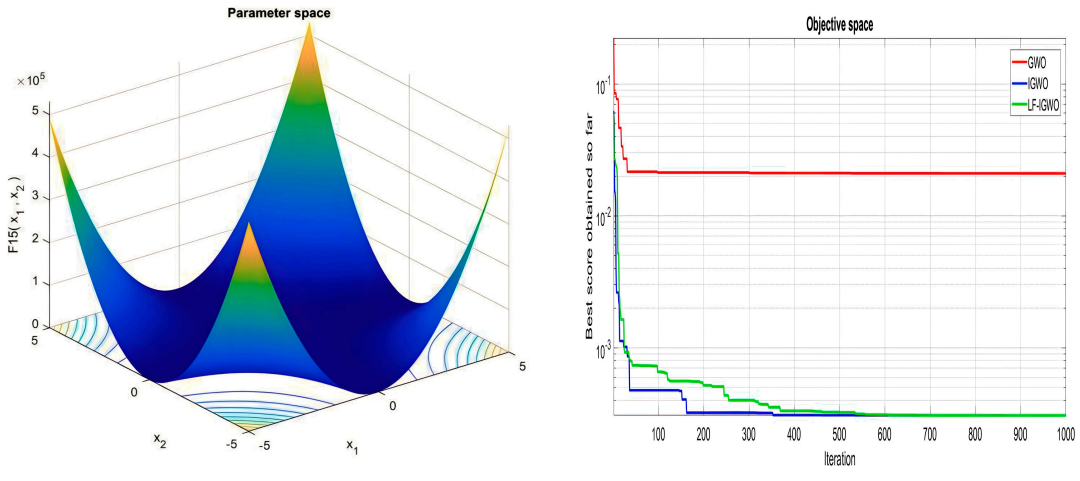
(**a**) (**b**)

Figure 15. (**a**) Search space for multi model function F13 (**b**) Comparison of convergence curve for F13.



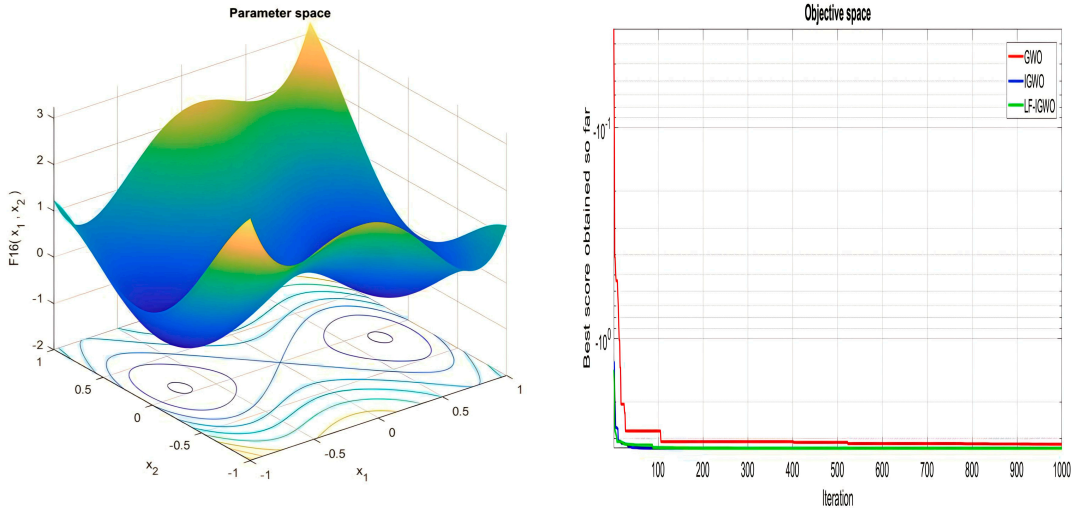
(**a**) (**b**)

Figure 16. (**a**) Search space for fixed dimensional function F14 (**b**) Comparison of convergence curve for F14.



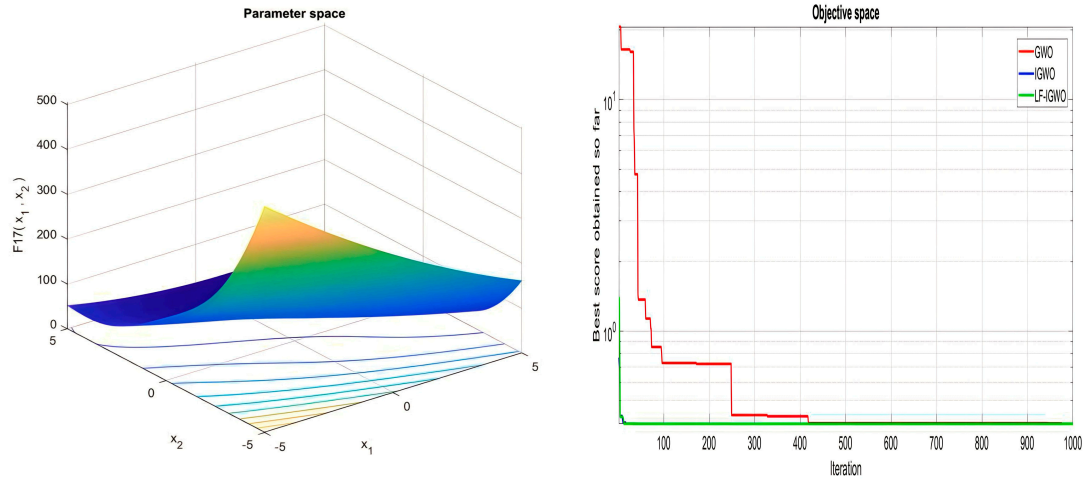
(**a**) (**b**)

Figure 17. (**a**) Search space for fixed dimensional function F15 (**b**) Comparison of convergence curve for F15.



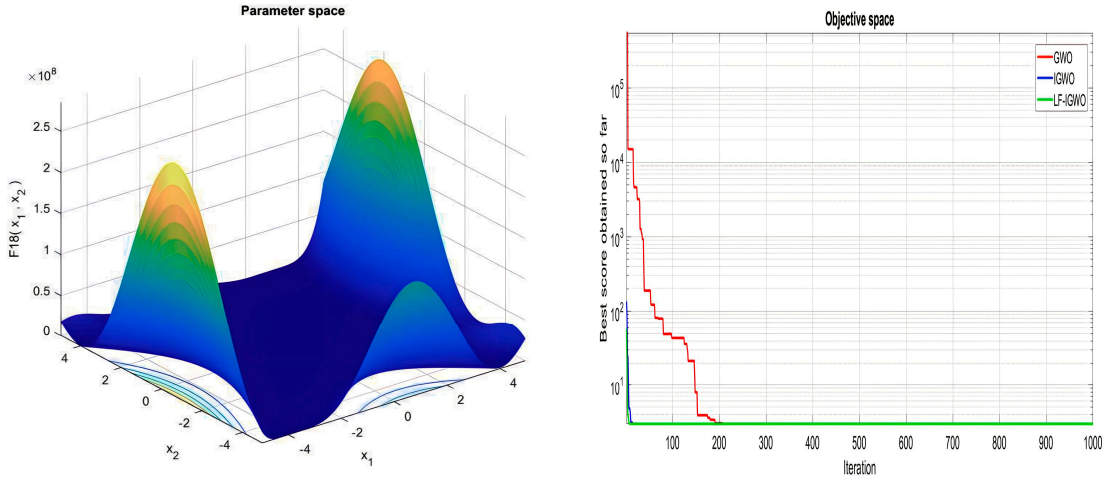
(**a**) (**b**)

Figure 18. (**a**) Search space for fixed dimensional function F16 (**b**) Comparison of convergence curve for F16.



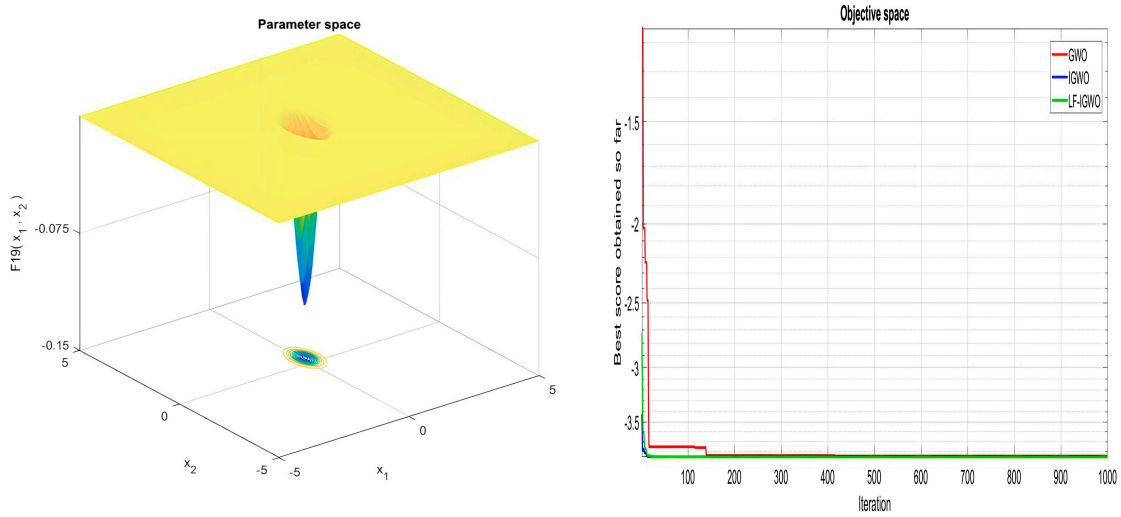
(**a**) (**b**)

Figure 19. (**a**) Search space for fixed dimensional function F17 (**b**) Comparison of convergence curve for F17.



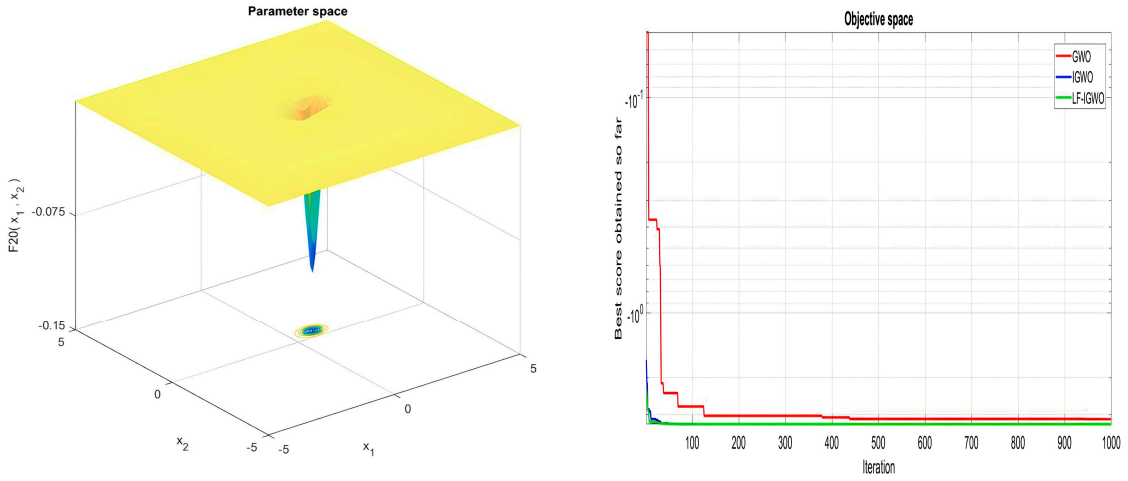
(**a**) (**b**)

Figure 20. (**a**) Search space for fixed dimensional function F18 (**b**) Comparison of convergence curve for F18.



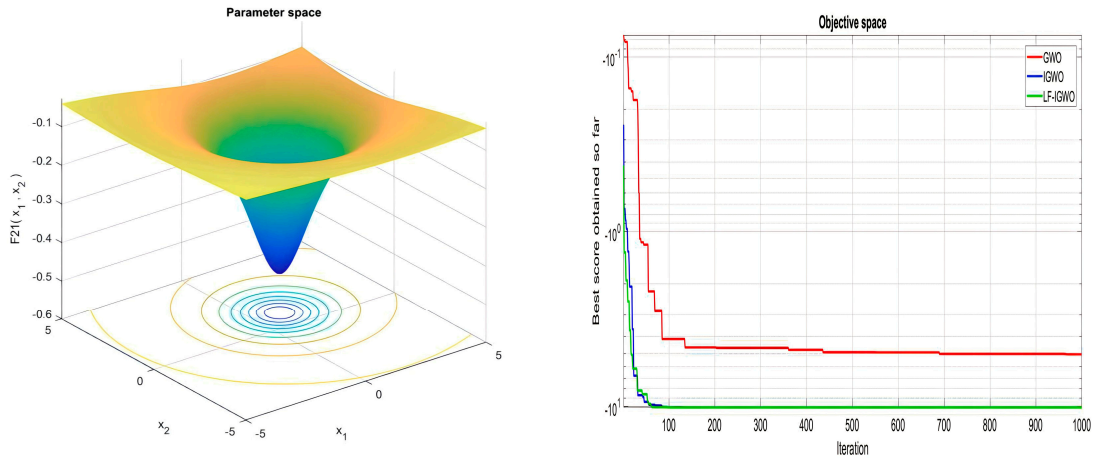
(**a**) (**b**)

Figure 21. (**a**) Search space for fixed dimensional function F19 (**b**) Comparison of convergence curve for F19.



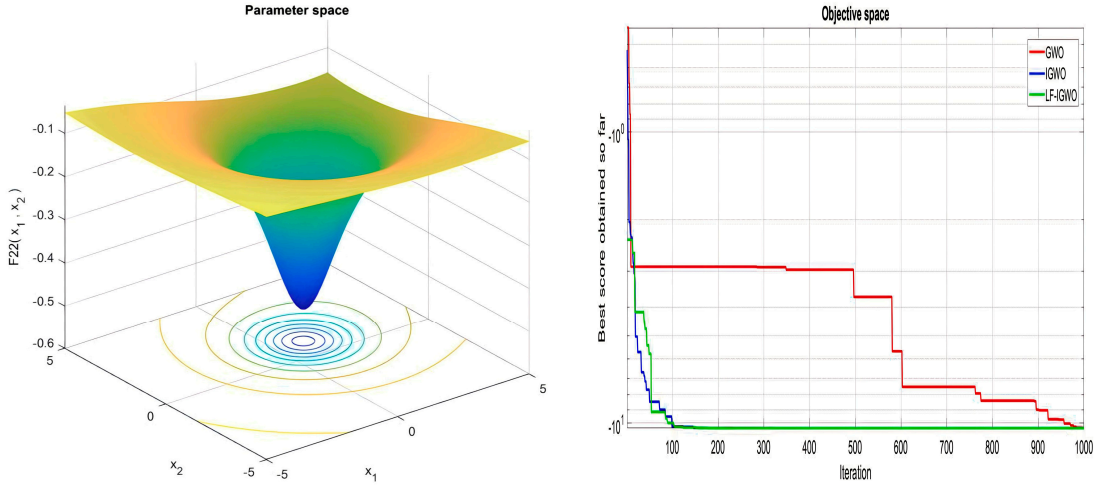
(**a**) (**b**)

Figure 22. (**a**) Search space for fixed dimensional function F20 (**b**) Comparison of convergence curve for F20.



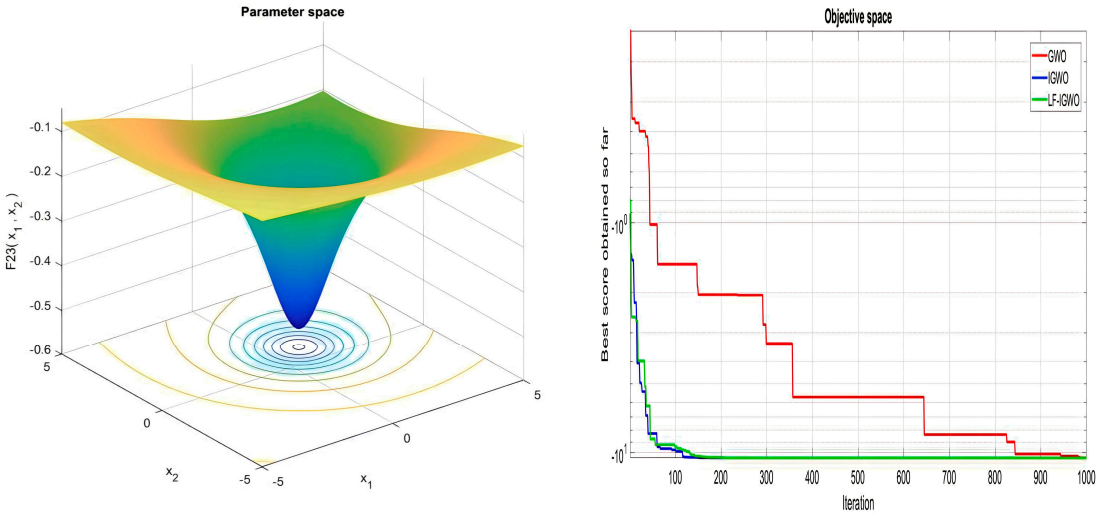
(**a**) (**b**)

Figure 23. (**a**) Search space for fixed dimensional function F21 (**b**) Comparison of convergence curve for F21.



(**a**) (**b**)

Figure 24. (**a**) Search space for fixed dimensional function F22 (**b**) Comparison of convergence curve for F22.



(**a**) (**b**)

Figure 25. (**a**) Search space for fixed dimensional function F23 (**b**) Comparison of convergence curve for F23.

The results of functions F1 to F7 are depicted in Figures 3–9, in which the newly proposed technique is represented in green, and one can infer from the convergence curves shown in the figures that the LF-IGWO has a good convergence rate and gives a better result than the IGWO and GWO. Therefore, the results indicate that the LF-IGWO has good exploitation capabilities.

With regard to the multi-model functions F8–F13 depicted in Figures 10–15, it seems that in function F9, GWO is better than IGWO and the newly proposed technique LF-IGWO. However, here, it seems that the new technique LF-IGWO has a good convergence rate compared to IGWO, which is shown in Figure 12. For functions 10 and 12, the convergence rates of IGWO and LF-IGWO are nearly the same, but LF-IGWO gives a somewhat better result at the end of 1000 iterations. From this, it can be concluded that the newly defined technique LF-IGWO has better exploration capabilities.

For the fixed dimension functions F14–F23, which are depicted in Figures 16–25, it seems that the convergence rate is almost same for the newly proposed technique LF-IGWO and IGWO. However, the final solution of GWO, IGWO, and LF-IGWO are the same at the end of the 1000 iterations.

Table 7 presents a comparison of the algorithms with the newly proposed algorithm, i.e., Levy flight-based improved grey wolf optimization, on the CEC 2017 benchmark functions. From the table, it can be clearly seen that it performs better than other popular algorithms when run with a population size of 30 each with 51 independent runs at 1000 iterations with a dimension of 30.

Table 7. Comparison of mean values obtained from CEC 2017 benchmark functions.

Function Number	PSO	SSR	GWO	I-GWO	LF-IGWO
F1	1.50×10^{11}	1.11×10^{11}	3.00×10^{10}	1.59×10^{10}	5.04×10^{7}
F2	2.64×10^{8}	1.00×10^{9}	2.84×10^{7}	3.49×10^{6}	4.68×10^{5}
F3	4.12×10^{6}	1.51×10^{8}	5.19×10^{4}	8.59×10^{4}	2.92×10^{4}
F4	4.53×10^{4}	2.38×10^{4}	5.30×10^{3}	1.082×10^{3}	491.677
F5	1.13×10^{3}	1.00×10^{3}	778.48	787.30	564.402
F6	751.76	699.66	660.77	646.87	601.758
F7	3.37×10^{3}	1.76×10^{3}	1.22×10^{3}	1.45×10^{3}	805.982
F8	1.37×10^{3}	1.28×10^{3}	1.05×10^{3}	1.068×10^{3}	849.817
F9	4.47×10^{4}	2.51×10^{4}	5.48×10^{3}	6.911×10^{3}	1.07×10^{3}
F10	1.19×10^{4}	1.08×10^{4}	6.32×10^{3}	7.529×10^{3}	3.01×10^{3}
F11	5.02×10^{4}	1.18×10^{4}	4.44×10^{3}	2.379×10^{3}	1.31×10^{3}
F12	3.44×10^{10}	1.66×10^{10}	7.21×10^{9}	3.58×10^{8}	4.38×10^{6}
F13	3.82×10^{10}	2.06×10^{10}	2.89×10^{9}	6.27×10^{7}	2.68×10^{4}
F14	1.59×10^{8}	2.59×10^{7}	1.71×10^{4}	5.87×10^{4}	3.92×10^{3}
F15	8.30×10^{9}	4.22×10^{9}	1.94×10^{4}	1.03×10^{7}	1.11×10^{4}
F16	9.62×10^{3}	1.30×10^{4}	3.566×10^{3}	3.184×10^{3}	2.10×10^{3}
F17	1.56×10^{5}	7.15×10^{3}	2.269×10^{3}	2.458×10^{3}	1.78×10^{3}
F18	9.70×10^{8}	2.56×10^{8}	7.17×10^{5}	1.05×10^{5}	5.98×10^{4}
F19	7.79×10^{9}	1.00×10^{9}	1.77×10^{6}	2.70×10^{7}	2.13×10^{4}
F20	4.09×10^{3}	4.11×10^{3}	2.240×10^{3}	2.484×10^{3}	2.20×10^{3}
F21	2.98×10^{3}	2.79×10^{3}	2.577×10^{3}	2.552×10^{3}	2.36×10^{3}
F22	1.22×10^{4}	1.29×10^{4}	6.644×10^{3}	4.186×10^{3}	2.44×10^{3}
F23	4.27×10^{3}	4.01×10^{3}	3.207×10^{3}	2.902×10^{3}	2.70×10^{3}
F24	4.49×10^{3}	4.41×10^{3}	3.377×10^{3}	3.071×10^{3}	2.87×10^{3}
F25	2.35×10^{4}	6.47×10^{3}	4.228×10^{3}	3.669×10^{3}	2.94×10^{3}
F26	2.02×10^{4}	1.32×10^{4}	9.251×10^{3}	5.105×10^{3}	3.40×10^{3}
F27	4.90×10^{3}	5.86×10^{3}	3.798×10^{3}	3.298×10^{3}	3.20×10^{3}
F28	1.56×10^{4}	1.04×10^{4}	5.270×10^{3}	3.686×10^{3}	3.33×10^{3}
F29	6.16×10^{4}	5.22×10^{4}	5.208×10^{3}	4.106×10^{3}	3.56×10^{3}
F30	8.49×10^{9}	3.25×10^{9}	2.1×10^{8}	1.86×10^{7}	4.53×10^{5}

4.1. 31-Level Cascaded H-Bridge MLI

A multilevel cascaded H-bridge (CHB) inverter is a type of power electronic device that can generate high-voltage, high-quality AC waveforms by synthesizing the output of multiple low-voltage DC sources [44]. It is a popular choice for high-power applications including motor drives and renewable energy sources.

The basic building block of a CHB multilevel inverter is the H-bridge module. An H-bridge is a configuration of four switches (typically MOSFETs or IGBTs) that can control the polarity and magnitude of the output voltage across a load [45]. By cascading multiple H-bridge modules, it is possible to generate a staircase-like voltage waveform that approximates a sinusoidal waveform.

A CHB multilevel inverter can produce several output voltage levels by switching the H-bridge modules on and off in a specific pattern. For instance, a two-level CHB inverter can generate an output voltage that switches between two voltage levels, while a three-level CHB inverter can generate an output voltage that switches between three voltage levels.

A CHB multilevel general equation for the output voltage waveform can be written as:

$$V_o(t) = \frac{2}{\pi} \sum_{m=1}^{\infty} \frac{1}{m} \sin(m\omega t) \times \sum_{k=1}^{N} V_{dc,k} \sin\left(\frac{(2k-1)m\pi}{2N}\right) \quad (10)$$

where:

- $V_{o(t)}$ is the output voltage waveform;
- ω is the resulting waveform's fundamental frequency;
- N is the inverter's H-bridge module count;
- $V_{dc,k}$ is the DC voltage input of the kth H-bridge module.

In practice, the output waveform becomes closer to a sine wave as the number of voltage levels increases.

The output in the inverter is obtained by combining the outputs of the individual H-bridge circuits. This allows for a higher number of voltage levels to be generated, resulting in a more efficient and higher-quality output waveform. Figure 26 depicts the H-bridge circuit [36].

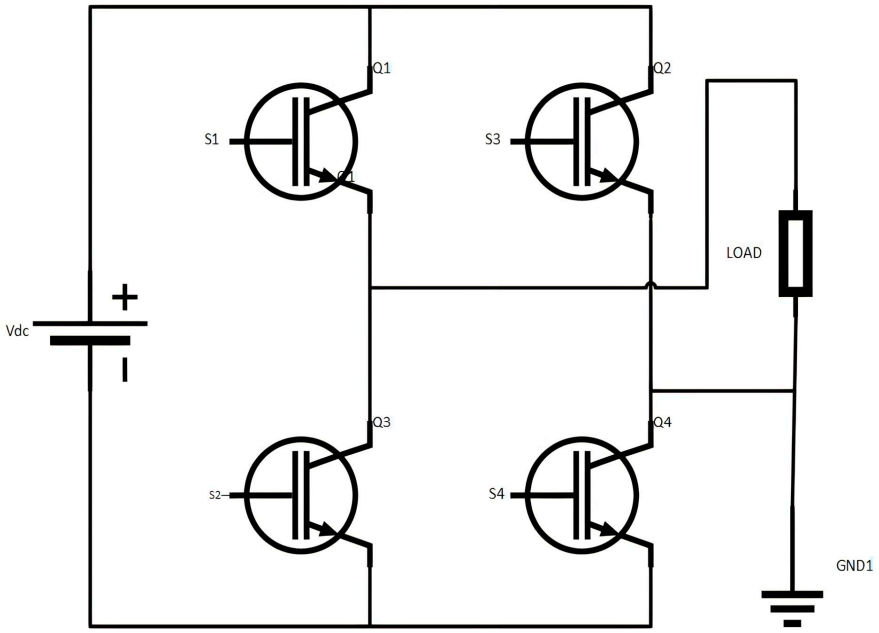


Figure 26. H-Bridge circuit model.

When switches S1 and S4 are closed (and S2 and S3 are open), there will be a positive voltage applied to the entire load. By turning OFF the S1 and S4 switches and turning ON the S2 and S3 switches, the voltage is reversed, permitting a negative voltage [46].

The S1 and S2 switches are never kept off at the same time though, as it may result in a short-circuiting of the supply of I/P voltage. The same caution should be utilized while using switches S3 and S4. The technical term for this situation is "shoot-through."

As can be seen from the waveform in Figure 27, the CHB inverter can produce a waveform that closely approximates a sinusoidal waveform with low harmonic distortion. This makes it a suitable choice for high-power applications that require high-quality AC power.

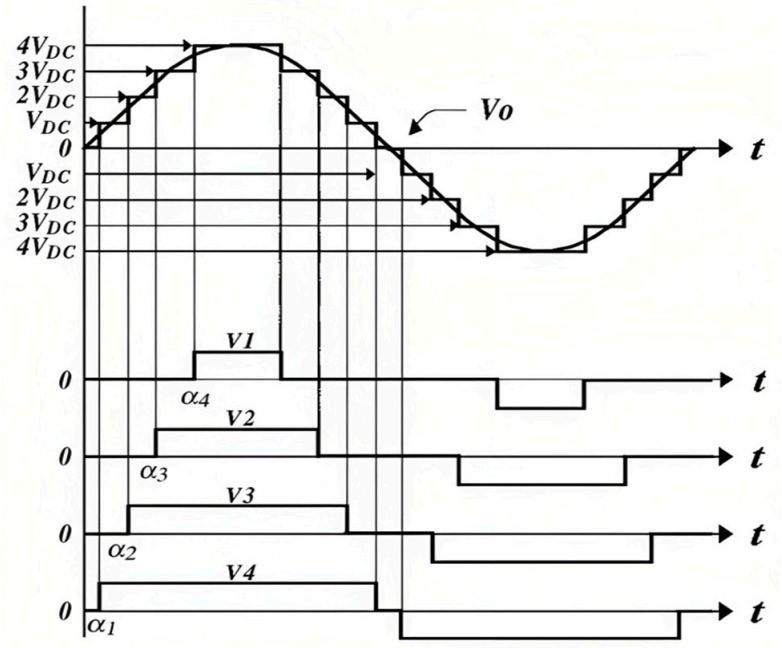


Figure 27. Cascaded H-bridge waveform.

4.2. Harmonics

In electrical engineering, harmonics refer to the sinusoidal components of an alternating current (AC) signal that have frequencies that are integer multiples of the fundamental frequency [47]. In most electrical power systems, the fundamental frequency, which is the lowest frequency included in an AC signal, is commonly 50 or 60 Hz. The presence of harmonics in an AC signal is caused by non-linear electrical loads, such as electronic devices and power electronic equipment, which generate distorted waveforms when they receive an AC signal [48]. Harmonics can have several negative impacts on electrical systems, such as:

1. Increased current levels: Harmonics can cause an increase in the RMS current levels, which can result in the overloading of conductors, transformers, and other electrical equipment.
2. Decreased power factor: A reduction in the power factor, a measurement of an electrical system's efficiency, can be brought on by the presence of harmonics. This can result in increased energy costs and decreased system efficiency.
3. Increased heating: The additional current levels caused by harmonics can result in increased heating in conductors and transformers, which can reduce their life expectancy and cause safety issues.
4. Interference with communication systems: Harmonics can interfere with communication systems and cause problems such as data corruption and interference with radio and television signals.

Several measures can be employed to mitigate these issues, such as harmonic filters, active harmonic filters, and passive filters. These filters work by attenuating or filtering out the harmonic components from the AC signal, resulting in a more sinusoidal waveform with fewer harmonics. Other measures, such as the use of balanced loads and power electronic devices with low harmonic distortion, can also help to reduce the level of harmonics in electrical systems.

4.3. Total Harmonics Distortion

A measure of the amount of distortion present in an alternating current (AC) waveform is called total harmonic distortion (THD) [49]. This is defined as the proportion between the total of all the waveform's harmonic components and the fundamental frequency component. The THD is usually expressed as a percentage and is used to characterize the quality of an AC signal.

Harmonic distortion in an AC waveform can be caused by non-linear loads, such as electronic devices and power electronic equipment, which generate distorted waveforms when they receive an AC signal [50]. These harmonics can have several negative impacts on electrical systems, such as increased current levels, decreased power factor, increased heating in conductors and transformers, and interference with communication systems.

The THD is an important parameter for characterizing the performance of electrical power systems, as well as electronic devices and power electronic equipment. For example, in power systems, a low THD indicates a high-quality waveform and a more efficient system, while a high THD indicates a distorted waveform and a less efficient system. Similarly, in electronic devices, a low THD indicates a high-quality signal and a more reliable device, while a high THD indicates a distorted signal and a less reliable device.

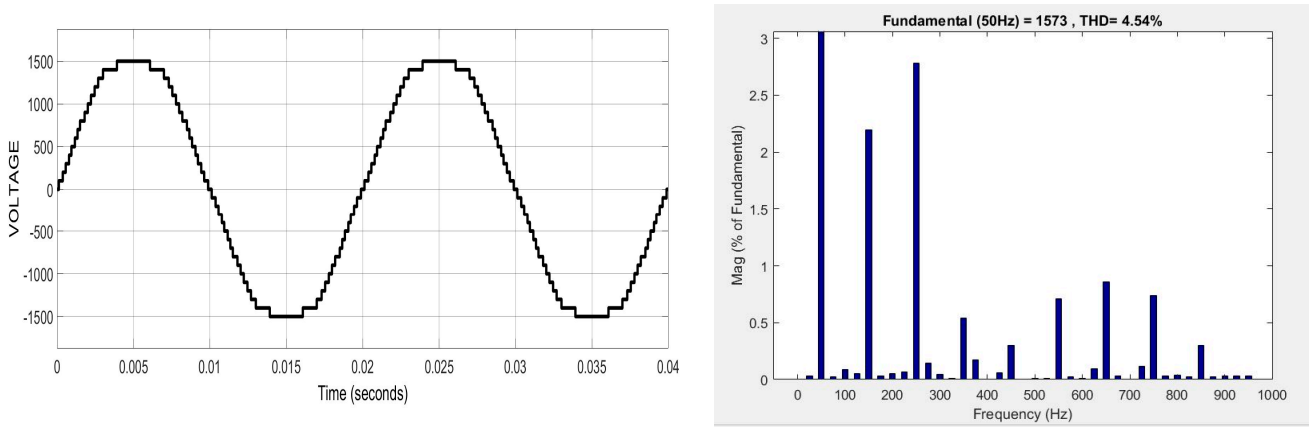
Several measures can be employed to reduce THD in electrical systems, such as harmonic filters, active harmonic filters, and passive filters. These filters work by attenuating or filtering out the harmonic components from the AC signal, resulting in a waveform that is more sinusoidal and has fewer harmonics. Other measures, such as the use of balanced loads and power electronic devices with low harmonic distortion, can also help to reduce THD. To eliminate the passive filter in order to simplify the circuit, optimization techniques have been used, which give a low THD value and make the circuit less complex. The equation for total harmonic distortion (THD) is:

$$THD = \frac{\sqrt[2]{\text{sum of squares of all harmonic frequencies except the fundamental frequency}}}{Fundamental\ Frequency} \times 100 \quad (11)$$

where the fundamental frequency is the first harmonic, and the other harmonic frequencies are multiples of the fundamental frequency. Table 8 shows a comparison of the firing angles with the different optimization techniques. Additionally, Figures 28–31 show the waveforms obtained from the different optimization techniques and THD spectrum. Table 9 shows a comparison of THD values.

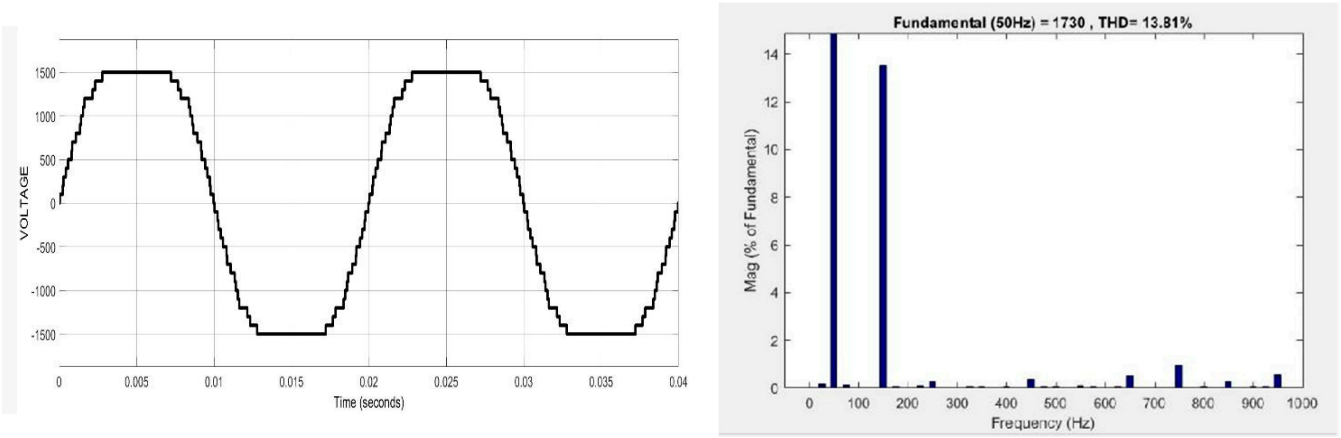
Table 8. Comparison of firing angles of LF-IGWO with other techniques.

Angles	LF-IGWO	IGWO	GWO	PSO
A1	0.036788	−0.25222	0.2074	0.2675
A2	0.84503	0.73168	0	−0.6905
A3	0.24646	0.13577	0.6342	−0.3538
A4	0.94454	−0.87421	0.0864	0.8315
A5	0.17788	0.18032	0.5053	−0.3885
A6	0.69653	−0.26365	0.0428	1.1988
A7	0.47166	0.33732	0.0575	−0.7247
A8	0.29501	0.42087	0.1335	−1.0523
A9	0.4211	−0.4421	0.74	0.8072
A10	0.77193	0.5118	0.3837	0.9868
A11	1.2306	−0.086737	0.3183	0.2315
A12	0.56625	−0.48201	0.5527	−0.4606
A13	0.11742	−0.010185	0.8574	0.0215
A14	0.63186	0.071222	0.2855	−0.1755
A15	0.36631	0.67509	0.4286	−0.3821



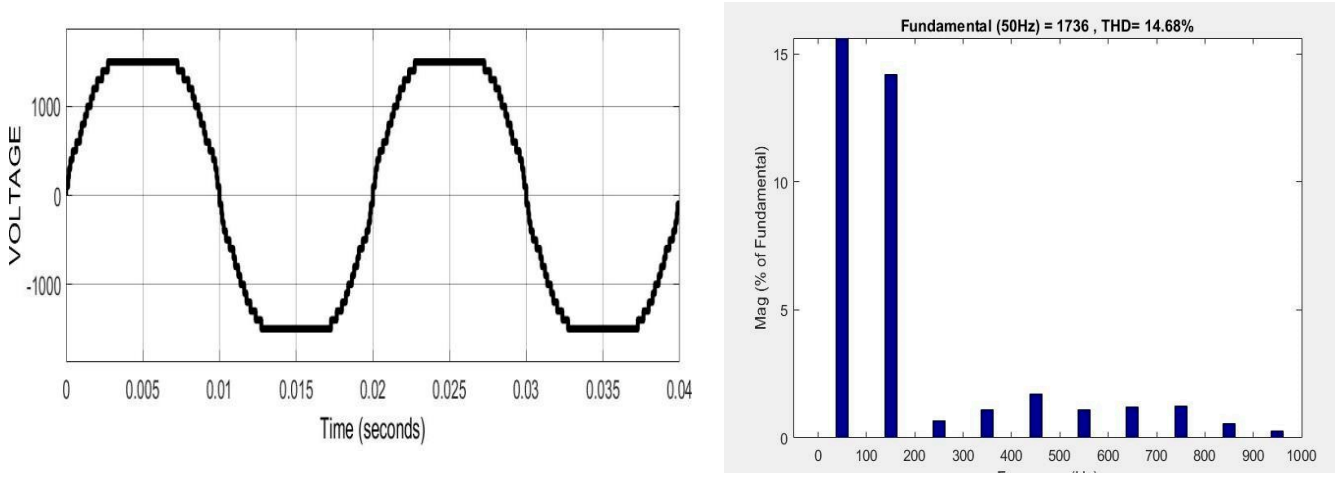
(**a**) (**b**)

Figure 28. In this figure, (**a**) represents LF-IGWO waveform and (**b**) represents the THD spectrum of LF-IGWO.



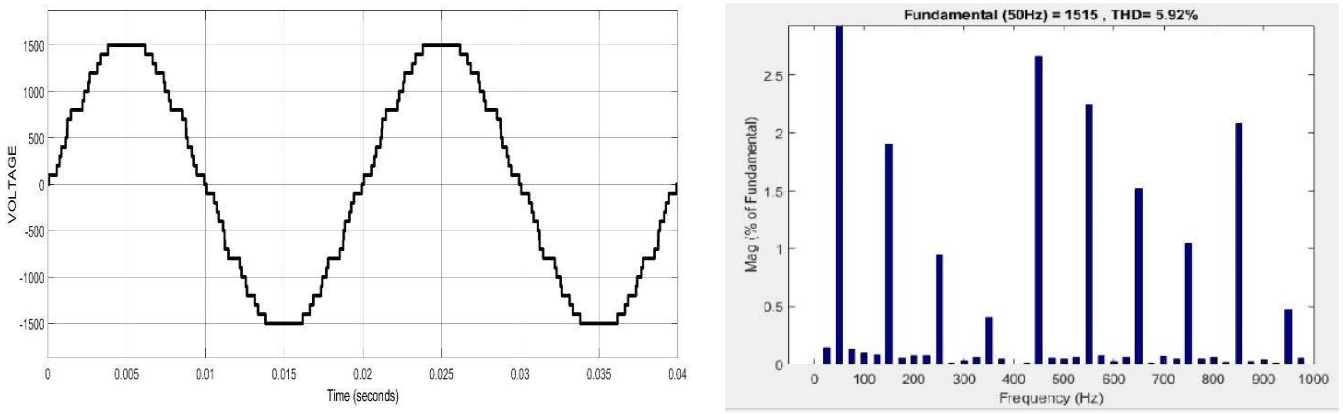
(**a**) (**b**)

Figure 29. In this figure, (**a**) represents IGWO waveform and (**b**) represents the THD spectrum of IGWO.



(**a**) (**b**)

Figure 30. In this figure, (**a**) represents GWO waveform and (**b**) represents the THD spectrum of GWO.



(**a**) (**b**)

Figure 31. In this figure, (**a**) represents PSO waveform and (**b**) represents the THD spectrum of PSO.

Table 9. Comparison of THD values of LF-IGWO with other techniques.

Algorithm	LF-IGWO	IGWO	GWO	PSO
THD (%)	4.54	13.81	14.68	5.92

5. Engineering Problems

5.1. Tension Compression Spring

A tension/compression spring's weight must be reduced while taking into account several constraints, such as shear stress, surge frequency, and minimum deflection [35]. The diameter of the wire (d), the mean diameter of coil (D), and the deciding variable are the number of active coils (N) [51]. Below is the mathematical formulation:

$$\vec{x} = [x_1 x_2 x_3] = [d\ D\ N]\ that\ minimize \tag{12}$$

$$f\left(\vec{x}\right) = \left(N + 2\ Dd^2\right) \tag{13}$$

$$g_1\left(\vec{x}\right) = 71,785x_1^4 - x_2^2 x_3\ \leq 0 \tag{14}$$

$$g_2\left(\vec{x}\right) = \frac{4x_2^2 - x_1 x_2}{12,566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1\ \leq 0 \tag{15}$$

$$g_3\left(\vec{x}\right) = x_2^3\ x_3 - 140.45x_1\ \leq 0 \tag{16}$$

$$g_4\left(\vec{x}\right) = x_1 + x_2 - 1.5\ \leq 0 \tag{17}$$

This issue was resolved using numerous metaheuristic algorithms. The method resolved the problem, and the results in Table 9 show how it performs in comparison to other algorithms.

The ideal weight according to Equation (13) is 0.01267, as determined by the LF IGWO algorithm. This outcome is very similar to the ideal value discovered by GWO and PSO. The ideal weight obtained by the LF IGWO method is less than that of other algorithms, such as GA, PSO, and I-GWO, and is equal to GWO, according to Table 10.

Table 10. Comparison of the data values of LF-IGWO with other techniques.

Algorithms	*d*	*D2*	*N*	Op. Value
LF-IGWO	0.0527485	0.367865	10.665	0.01267
IGWO	0.0517029	0.356964	11.2893	0.012681
GWO	0.05169	0.3567	11.2888	0.012666
PSO	0.051728	0.357644	11.24454	0.012674
GA	0.05148	0.35166	11.6322	0.012704

5.2. Pressure Vessel Design

In this instance, it is asserted that the optimization will result in a reduction in the overall price, which includes material price, overall expense in the welding process, and overall expense of constructing a cylindrical vessel. The decision-making factors are the cylindrical section's length without taking into account the head (L), the thickness of the shell (T_s), the height of the head (T_h), and internal radius (R) [52]. There are four inequality constraints in this problem, three of which are linear and one of which is nonlinear. The following equations are the mathematical representation:

$$\vec{x} = [x_1 x_2 x_3\ x_4] = [T_s T_h\ R\ L\ ]\ that\ minimize \tag{18}$$

$$f\left(\vec{x}\right) = 0.6224 T_s R L + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_s^2 R \tag{19}$$

$$g_1\left(\vec{x}\right) = -x_1 + 0.0193 x_3\ \leq 0 \tag{20}$$

$$g_2\left(\vec{x}\right) = -x_2 + 0.0095 x_3\ \leq 0 \tag{21}$$

$$g_3\left(\vec{x}\right) = \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000\ \leq 0 \tag{22}$$

$$g_4\left(\vec{x}\right) = x_4 - 240 \leq 0 \tag{23}$$

Metaheuristic algorithms were employed to address this issue. Table 11 compares the performance of the LF-IGWO algorithm with that of other techniques. It is clear from Table 11 that, in comparison to many other algorithms, the LF-IGWO method achieves the lowest cost calculated using Equation (19). The acquired cost is reasonably close to GA. GWO appears to perform better than others in this. However, LF-IGWO provides us with a superior outcome than GA, PSO, and IGWO.

Table 11. Comparison of cost values obtained from LF-IGWO with other techniques.

Algorithms	T_s	T_h	*R*	*L*	Op. Value
LF-IGWO	0.8021493	0.5113717	41.54997	183.573	6282.2292
GWO	0.8125	0.4345	42.089181	176.758731	6051.5639
GA	0.8125	0.4345	40.3239	200	6288.7445
PSO	0.883044	0.533053	45.38829	190.0616	7865.233
IGWO	0.9035907	0.5319827	44.20703	154.0763	6793.5848

5.3. Welded Beam Design

For a reduction in the cost of making a design, the optimization approach is put forward. The deciding elements are the width of the weld (h), the height of the bar (t), the length

of the attached component of the bar (l), and the bar's thickness (b) [53]. There are seven inequality restrictions [54]. The given problem has the following mathematical formulation:

$$\vec{x} = [x_1 x_2 x_3\ x_4] = [h\ l\ t\ b]\ that\ minimize \tag{24}$$

$$f\left(\vec{x}\right) = 1.10471h^2 l + 0.04811tb\ (14.0 + l) \tag{25}$$

$$g_1\left(\vec{x}\right) = \tau\left(\vec{x}\right) - \tau_{max}\ \leq 0 \tag{26}$$

$$g_2\left(\vec{x}\right) = \sigma\left(\vec{x}\right) - \sigma_{max}\ \leq 0 \tag{27}$$

$$g_3\left(\vec{x}\right) = \delta\left(\vec{x}\right) - \delta_{max}\ \leq 0 \tag{28}$$

$$g_4\left(\vec{x}\right) = x_1 - x_4\ \leq 0 \tag{29}$$

$$g_5\left(\vec{x}\right) = P - P_c\left(\vec{x}\right)\ \leq 0 \tag{30}$$

$$g_6\left(\vec{x}\right) = 0.125 - x_1 \leq 0 \tag{31}$$

$$g_7\left(\vec{x}\right) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0\ \leq 0 \tag{32}$$

This design challenge was subjected to the LF-IGWO method, and the outcomes are displayed in Table 12. Table 11 depicts how the LF-IGWO algorithm outperforms other competing algorithms in minimizing the cost. It can be concluded that the LF-IGWO method performs better in some design challenges than other algorithms and is capable of tackling limited engineering design problems [55–57].

Table 12. Comparison of cost values obtained from LF-IGWO with other techniques.

Algorithms	T_s	T_h	R	L	Op. Value
LF-IGWO	0.8021493	0.5113717	41.54997	183.573	6282.2292
GWO	0.8125	0.4345	42.089181	176.758731	6051.5639
GA	0.8125	0.4345	40.3239	200	6288.7445
PSO	0.883044	0.533053	45.38829	190.0616	7865.233
IGWO	0.9035907	0.5319827	44.20703	154.0763	6793.5848

The statistical Wilcoxon rank sum test was performed on the newly proposed technique. The statistical data obtained by running the functions given in Tables 2–4 and the engineering problems for the different algorithms are depicted in Table 13. From this test, it was found that the data are not significant when compared to the engineering problem. There is a minor improvement in the result of the newly proposed Levy flight-based improved grey wolf optimization algorithm when it is compared with improved grey wolf optimization. It can be concluded that LF-IGWO is not significantly improved as compared to I-GWO. However, when it is compared with the benchmark function, it performs better than GWO, PSO, and GA, but when compared with the IGWO and SSR, it performs moderately.

Table 13. Statistical data of functions from Tables 2–4 and engineering problem comparison with other algorithms.

Function Number	PSO	GA	GWO	IGWO	SSR
F1	0.001588	0.000156	0.003177	0.027086	0.10499
F2	0.004522	0.000145	0.003177	0.18577	0.10499
F3	0.000145	0.000145	0.000145	1	0.10499
F4	0.009524	0.000145	0.000145	0.37904	0.10499
F5	0.000145	0.28378	0.000145	0.077272	0.10499
F6	0.000145	0.000145	0.000145	0.27071	0.10499
F7	0.0001268	0.10499	0.000145	0.48252	0.10499
F8	0.30815	0.37904	0.31815	0.28378	0.10499
F9	0.62527	0.69913	0.10221	0.98231	0.10499
F10	0.000294	0.000156	0.000145	0.23985	0.10499
F11	0.000156	0.000156	0.000145	0.46427	0.10499
F12	0.351889	0.000145	0.000145	0.063533	0.10499
F13	0.168452	0.000156	0.000145	0.59969	0.10499
F14	0.061837	0.01857	0.071429	0.66667	0.66667
F15	0.62483	0.35684	0.47619	0.88571	0.4
F16	0.07467	0.071429	0.071429	0.66667	0.66667
F17	0.071498	0.071429	0.071429	0.66667	0.66667
F18	0.072795	0.66667	0.071429	1	0.66667
F19	0.069426	0.061584	0.071429	0.7	0.5
F20	0.49509	0.51455	0.48485	0.69913	0.28571
F21	0.0064732	0.0008616	0.009524	0.68571	0.4
F22	0.018654	0.02666	0.009524	0.88571	0.4
F23	0.72364	0.5426	0.009524	1	0.4
Tension Compression Spring	0.7569	0.6	0.71429	1	0.5
Welded Beam	0.09852	0.854554	0.47619	0.68571	1
Pressure Vessel	0.09852	0.854554	0.25714	0.68571	1

6. Conclusions

A unique, metaphor-free metaheuristic algorithm is suggested in this study. Iteratively reducing the search space yields the best result. By utilizing this method, 23 common statistical benchmark functions are minimized. The outcomes demonstrate the strong exploration and exploitation capabilities of the suggested method. The LF-IGWO algorithm performs better than other algorithms, particularly in multimodal benchmark functions, based on the outcomes of the test functions. This highlights the capacity of the LF-IGWO algorithm to avoid local optima, which increases its competitiveness and supports the statements made in the research. One of the drawbacks that can be seen is that the algorithm becomes somewhat complex and hard to understand. As there is already a DLH strategy applied in the improved grey wolf optimization (IGWO) method, the addition of one more strategy to improve IGWO, which is Levy flight, makes it somewhat difficult for new researchers to understand.

Furthermore, three restricted engineering design issues are resolved using the LF-IGWO algorithm. The outcomes demonstrate that the LF-IGWO algorithm is also capable of solving limited design issues. Out of the three design challenges that were taken into consideration, according to performance comparison data, the LF-IGWO approach outperforms the other algorithms in two of them. Overall, it can be said that this method, despite having straightforward logic, may provide very competitive outcomes when compared to the well-known optimization techniques. Additionally, the suggested technique is used to solve the 31-level inverter for THD minimization to demonstrate its functionality. It can be seen from the results that the proposed algorithm gives us the appropriate firing angles that give us a total harmonic distortion (THD) value less than five, as per the standard of IEEE 519.

Author Contributions: Conceptualization, B.B.; Data curation, H.S. and K.A.; Funding acquisition, G.P.J. and B.S.; Investigation, G.P.J. and B.S.; Methodology, B.B. and H.S.; Project administration, B.S.; Resources, G.P.J. and B.S.; Software, B.B., H.S. and K.A.; Supervision, G.P.J. and B.S.; Validation, H.S., K.A. and B.S.; Visualization, H.S. and K.A.; Writing—original draft, K.A.; Writing—review and editing, B.B. and G.P.J. All authors have read and agreed to the published version of the manuscript.

Funding: The present research has been conducted by the Research Grant of Kwangwoon University in 2023.

Data Availability Statement: Data sharing is not applicable to this article as no datasets were generated during the current study.

Conflicts of Interest: The authors declare that they have no conflicts of interest.

References

1. Kaur, K.; Kumar, Y. Swarm Intelligence and its applications towards Various Computing: A Systematic Review. In Proceedings of the 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 17–19 June 2020; pp. 57–62. [CrossRef]
2. Trianni, V.; Tuci, E.; Passino, K.M.; Marshall, J.A.R. Swarm Cognition: An interdisciplinary approach to the study of self-organising biological collectives. *Swarm Intell.* **2011**, *5*, 3–18. [CrossRef]
3. Sneha; Wajahat. Swarm Intelligence. *Int. J. Sci. Eng. Res.* **2017**, *8*, 10.
4. Hazem, A.; Glasgow, J. *Swarm Intelligence: Concepts, Models and Applications*; School of Computing, Queen's University: Kingston, ON, Canada, 2012. [CrossRef]
5. Zahra, B.; Siti Mariyam, S. A review of population-based meta-heuristic algorithm. *Int. J. Adv. Soft Comput. Its Appl.* **2013**, *5*, 1–35.
6. Colin, R. Genetic Algorithms. In *Handbook of Metaheuristics*; Springer: Boston, MA, USA, 2010; pp. 109–139. [CrossRef]
7. Seyedali, M. Particle Swarm Optimisation. In *Evolutionary Algorithms and Neural Networks*; Springer: Cham, Switzerland, 2019; pp. 15–31. [CrossRef]
8. Yi, G.; Jin, M.; Zhou, Z. Research on a Novel Ant Colony Optimization Algorithm. In *Advances in Neural Networks—Lecture Notes in Computer Science*; Zhang, L., Lu, B.L., Kwok, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6063. [CrossRef]
9. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
10. Ribeiro, M.d.R.; Aguiar, M.S.d. Cultural Algorithms: A Study of Concepts and Approaches. In Proceedings of the 2011 Workshop-School on Theoretical Computer Science, Pelotas, Brazil, 24–26 August 2011; pp. 145–148. [CrossRef]
11. Rutenbar, R.A. Simulated annealing algorithms: An overview. *IEEE Circuits Devices Mag.* **1989**, *5*, 19–26. [CrossRef]
12. Prajapati, V.K.; Jain, M.; Chouhan, L. Tabu Search Algorithm (TSA): A Comprehensive Survey. In Proceedings of the 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), Jaipur, India, 7–8 February 2020; pp. 1–8. [CrossRef]
13. Zhang, J.; Zhang, P. A study on harmony search algorithm and applications. In Proceedings of the 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 736–739. [CrossRef]
14. Kytöjoki, J.; Nuortio, T.; Bräysy, O.; Gendreau, M. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Comput. Oper. Res.* **2007**, *34*, 2743–2757. [CrossRef]
15. Mohammad, H.; Mohammad, H. Simplex method to Optimize Mathematical manipulation. *Int. J. Recent Technol. Eng. (IJRTE)* **2019**, *7*, 5.
16. Alonso, G.; del Valle, E.; Ramirez, J.R. 5—Optimization methods. In *Woodhead Publishing Series in Energy, Desalination in Nuclear Power Plants*; Woodhead Publishing: Shaxton, UK, 2020; pp. 67–76. ISBN 9780128200216. [CrossRef]
17. Lian, P.; Wang, C.; Xiang, B.; Shi, Y.; Xue, S. Gradient-based optimization method for producing a contoured beam with single-fed reflector antenna. *J. Syst. Eng. Electron.* **2019**, *30*, 22–29. [CrossRef]
18. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
19. Shukla, P.; Singh, S.K.; Khamparia, A.; Goyal, A. Nature-inspired optimization techniques. In *Nature-Inspired Optimization Algorithms*; Academic Press: Cambridge, MA, USA, 2021. [CrossRef]
20. Ingber, A.; Lester, I. Simulated annealing: Practice versus theory. *Math. Comput. Model.* **2002**, *18*, 29–57. [CrossRef]
21. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
22. Anita; Yadav, A.; Kumar, N. Artificial electric field algorithm for engineering optimization problems. *Expert Syst. Appl.* **2020**, *149*, 113308. [CrossRef]
23. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
24. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl. Based Syst.* **2020**, *191*, 105190. [CrossRef]
25. Bansal, J.C.; Sharma, H.; Jadon, S.S. Artificial bee colony algorithm: A survey. *Int. J. Adv. Intell. Paradig.* **2013**, *5*, 123–159. [CrossRef]

26. Neshat, M.; Sepidnam, G.; Sargolzaei, M.; Toosi, A.N. Artificial fish swarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artif. Intell. Rev.* **2014**, *42*, 965–997. [CrossRef]
27. Guo, C.; Tang, H.; Niu, B.; Lee, C.B.P. A survey of bacterial foraging optimization. *Neurocomputing* **2021**, *452*, 728–746. [CrossRef]
28. Zou, F.; Chen, D.; Xu, Q. A survey of teaching–learning-based optimization. *Neurocomputing* **2019**, *335*, 366–383. [CrossRef]
29. Atashpaz-Gargari, E.; Lucas, C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 4661–4667. [CrossRef]
30. Zhang, T.; Yang, C.; Zhao, X. Using Improved Brainstorm Optimization Algorithm for Hardware/Software Partitioning. *Appl. Sci.* **2019**, *9*, 866. [CrossRef]
31. Abdelhamid, M.; Kamel, S.; Mohamed, M.A.; Aljohani, M.; Rahmann, C.; Mosaad, M.I. Political Optimization Algorithm for Optimal Coordination of Directional Overcurrent Relays. In Proceedings of the 2020 IEEE Electric Power and Energy Conference (EPEC), Edmonton, AB, Canada, 9–10 November 2020; pp. 1–7. [CrossRef]
32. Huynh, N.T.; Nguyen, T.V.T.; Nguyen, Q.M. Optimum Design for the Magnification Mechanisms Employing Fuzzy Logic–ANFIS. *Comput. Mater. Continua.* **2022**, *73*, 5961–5983. [CrossRef]
33. Kler, R.; Gangurde, R.; Elmirzaev, S.; Hossain, S.; Vo, N.V.T.; Nguyen, T.V.T.; Kumar, P.N. Optimization of Meat and Poultry Farm Inventory Stock Using Data Analytics for Green Supply Chain Network. *Discret. Dyn. Nat. Soc.* **2022**, *2022*, 8970549. [CrossRef]
34. Huynh, T.T.; Nguyen, T.V.T.; Nguyen, Q.M.; Nguyen, T.K. Minimizing Warpage for Macro-Size Fused Deposition Modeling Parts. *Comput. Mater. Contin.* **2021**, *68*, 2913–2923. [CrossRef]
35. Al-Khazraji, H. Optimal design of a proportional-derivative state feedback controller based on meta-heuristic optimization for a quarter car suspension system. *Math. Model. Eng. Probl.* **2022**, *9*, 437–442. [CrossRef]
36. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
37. Faris, H.; Aljarah, I.; Al-Betar, M.A.; Mirjalili, S. Grey wolf optimizer: A review of recent variants and applications. *Neural Comput. Appl.* **2018**, *30*, 413–435. [CrossRef]
38. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S. An improved grey wolf optimizer for solving engineering problems. *Expert Syst. Appl.* **2021**, *166*, 113917. [CrossRef]
39. Jia, H.; Sun, K.; Zhang, W.; Leng, X. An enhanced chimp optimization algorithm for continuous optimization domains. *Complex Intell. Syst.* **2022**, *8*, 65–82. [CrossRef]
40. Gai, W.; Qu, C.; Liu, J.; Zhang, J. An improved grey wolf algorithm for global optimization. In Proceedings of the 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 2494–2498. [CrossRef]
41. Banaie-Dezfouli, M.; Nadimi-Shahraki, M.H.; Beheshti, Z. R-GWO: Representative-based grey wolf optimizer for solving engineering problems. *Appl. Soft Comput.* **2021**, *106*, 107328. [CrossRef]
42. Kamaruzaman, A.F.; Zain, A.M.; Yusuf, S.M.; Udin, A. Levy Flight Algorithm for Optimization Problems—A Literature Review. *Appl. Mech. Mater.* **2013**, *421*, 496–501. [CrossRef]
43. Li, J.; An, Q.; Lei, H.; Deng, Q.; Wang, G.-G. Survey of Lévy Flight-Based Metaheuristics for Optimization. *Mathematics* **2022**, *10*, 2785. [CrossRef]
44. Mahesh, A.; Sushnigdha, G. A novel search space reduction optimization algorithm. *Soft Comput.* **2021**, *25*, 9455–9482. [CrossRef]
45. Prasad, K.N.V.; Kumar, G.R.; Kiran, T.V.; Narayana, G.S. Comparison of different topologies of cascaded H-Bridge multilevel inverter. In Proceedings of the 2013 International Conference on Computer Communication and Informatics, Coimbatore, India, 4–6 January 2013; pp. 1–6. [CrossRef]
46. Gaikwad, A.; Arbune, P.A. Study of cascaded H-Bridge multilevel inverter. In Proceedings of the 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), Pune, India, 9–10 September 2016; pp. 179–182. [CrossRef]
47. Krishna, R.A.; Suresh, L.P. A brief review on multi level inverter topologies. In Proceedings of the 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 18–19 March 2016; pp. 1–6. [CrossRef]
48. Hassan, N.; Mohagheghian, I. A Particle Swarm Optimization algorithm for mixed variable nonlinear problems. *IJE Trans. A Basics* **2011**, *24*, 65–78.
49. Firdoush, S.; Kriti, S.; Raj, A.; Singh, S.K. Reduction of Harmonics in Output Voltage of Inverter. *Int. J. Eng. Res. Technol. (IJERT)* **2016**, *4*, 1–6.
50. Jacob, T.; Suresh, L.P. A review paper on the elimination of harmonics in multilevel inverters using bioinspired algorithms. In Proceedings of the 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 18–19 March 2016; pp. 1–8. [CrossRef]
51. Mohd Radzi, M.Z.; Azizan, M.M.; Ismail, B. Observatory case study on total harmonic distortion in current at laboratory and office building. *Phys. Conf. Ser.* **2020**, *1432*, 012008. [CrossRef]
52. BarathKumar, T.; Vijayadevi, A.; Brinda Dev, A.; Sivakami, P.S. Harmonic Reduction in Multilevel Inverter Using Particle Swarm Optimization. *IJISET—Int. J. Innov. Sci. Eng. Technol.* **2017**, *4*, 99–104.
53. Nematollahi, A.F.; Rahiminejad, A.; Vahidi, B. A novel physical based meta-heuristic optimization method known as Lightning Attachment Procedure Optimization. *Appl. Soft Comput.* **2017**, *59*, 596–621. [CrossRef]
54. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]

55. Agushaka, J.O.; Ezugwu, A.E. Advanced arithmetic optimization algorithm for solving mechanical engineering design problems. *PLoS ONE* **2021**, *16*, e0255703. [CrossRef]
56. Abualigah, L.; Elaziz, M.A.; Khasawneh, A.M.; Alshinwan, M.; Ibrahim, R.A.; Al-Qaness, M.A.A.; Mirjalili, S.; Sumari, P.; Gandomi, A.H. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: A comprehensive survey, applications, comparative analysis, and results. *Neural Comput. Appl.* **2022**, *34*, 4081–4110. [CrossRef]
57. Boora, K.; Kumar, A.; Malhotra, I.; Kumar, V. Harmonic reduction using Particle Swarm Optimization based SHE Modulation Technique in Asymmetrical DC-AC Converter. *Int. J. Electr. Comput. Eng. Syst.* **2022**, *13*, 867–875. [CrossRef]