applied sciences

MDPI

Article

A New Bipolar Approach Based on the Rooster Algorithm Developed for Utilization in Optimization Problems

Mashar Cenk Gençal

Department of Management Information Systems, Osmaniye Korkut Ata University, Osmaniye 80000, Türkiye; masharcenkgencal@osmaniye.edu.tr

Abstract: Meta-heuristic algorithms are computational methods inspired by evolutionary processes, animal or plant behaviors, physical events, and other natural phenomena. Due to their success in solving optimization problems, meta-heuristic algorithms are widely used in the literature, leading to the development of novel variants. In this paper, new swarm-based meta-heuristic algorithms, called Improved Roosters Algorithm (IRA), Bipolar Roosters Algorithm (BRA), and Bipolar Improved Roosters Algorithm (BIRA), which are mainly based on Roosters Algorithm (RA), are presented. First, the new versions of RA (IRA, BRA, and BIRA) were compared in terms of performance, revealing that BIRA achieved significantly better results than the other variants. Then, the performance of the BIRA algorithm was compared with the performances of meta-heuristic algorithms widely used in the literature, Standard Genetic Algorithm (SGA), Differential Evolution (DE), Particle Swarm Optimization (PSO), Cuckoo Search (CS), and Grey Wolf Optimizer (GWO), and thus, its success in the literature was tested. Moreover, RA was also included in this test to show that the new version, BIRA, is more successful than the previous one (RA). For all comparisons, 20 well-known benchmark optimization functions, 11 CEC2014 test functions, and 17 CEC2018 test functions, which are also in the CEC2020 test suite, were employed. To validate the significance of the results, Friedman and Wilcoxon Signed Rank statistical tests were conducted. In addition, three commonly used problems in the field of engineering were used to test the success of algorithms in real-life scenarios: pressure vessel, gear train, and tension/compression spring design. The results indicate that the proposed algorithm (BIRA) provides better performance compared to the other meta-heuristic algorithms.

Keywords: engineering design problems; evolutionary computation; meta-heuristics; optimization

check for updates

Received: 17 March 2025
Revised: 20 April 2025
Accepted: 27 April 2025
Published: 29 April 2025

Citation: Gençal, M.C. A New Bipolar Approach Based on the Rooster Algorithm Developed for Utilization in Optimization Problems. *Appl. Sci.* **2025**, *15*, 4921. https://doi.org/10.3390/app15094921

Copyright: © 2025 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Nature has long inspired human discoveries, and with advances in technology, researchers have developed new optimization algorithms based on natural phenomena. These stochastic algorithms, also known as meta-heuristics, have gained popularity since they can be easily formulated and modeled thanks to their structures that mimic evolutionary processes, animal or plant behavior, or physical events. Another reason for the widespread adoption of meta-heuristic algorithms is their ability to avoid local optima entrapment and premature convergence, issues that often affect traditional methods [1–4].

Typically, meta-heuristic s commence by generating an initial population of potential solutions obtained through random means. Once this initial population is formed, they evaluate the fitness of individuals using a fitness function. Subsequently, they embark on their nature-inspired search procedures. This iterative process continues until termination

criteria are met; either the optimum solution is discovered or the maximum number of generations is reached.

One of the widely used meta-heuristic algorithms in the literature are Evolutionary Algorithms (EAs). EAs generally includes the following intelligent strategies: selection, recombination, and mutation. Some of the most known EAs are Genetic Algorithms (GAs) [5], Differential Evolution (DE) [6], Memetic Algorithms (MAs) [7], and Biogeography-Based Optimizer (BBO) [8].

The fact that living creatures in nature generally form flocks has led researchers to develop Swarm-based Algorithms (SBA). Among these algorithms, perhaps the most well-known is Particle Swarm Optimization (PSO). PSO simulates the social behavior of bird flocks and fish swarms [9]. Each particle (potential solution) in PSO also possesses a velocity to explore the search space. Other renowned SBAs encompass Ant Colony Optimization (ACO) [10], Cuckoo Search (CS) [11], Artificial Bee Colony (ABC) [12], Firefly Algorithm (FFA) [13], and Grey Wolf Optimizer (GWO) [14].

Furthermore, researchers inspired by fundamental subjects that have an important place in sciences such as physics, chemistry, and astronomy have led to the emergence of new meta-heuristic algorithms. Some common examples include Simulated Annealing (SA) [15], Big-Bang Big-Crunch (BBBC) [16], Gravitational Search (GS) [17], Black Hole (BH) [18], and Momentum Search algorithm (MSA) [19].

Apart from the meta-heuristic algorithms mentioned above, there are also algorithms inspired by human behavior such as Tabu Search (TS) [20], Harmony Search (HS) [21], Imperialist Competitive Algorithm (ICA) [22], Teaching Learning Based Optimization (TLBO) [23], and Fireworks Algorithm (FWA) [24].

In meta-heuristic algorithms, the idea that the population will improve by selecting the best individual is prevalent. However, the idea of more towards the good can cause the algorithm to get stuck in the local and not be able to reach the optimum value, especially in challenging test functions. One way to get rid of this situation may be to adapt bipolar behavior to the algorithm. Studies in this field have shown that adopting bipolar behavior increases the performance of the algorithm [25,26]. Thus, the idea of testing bipolar behavior on a new algorithm was born and the idea of adding bipolar behavior to Roosters Algorithm (RA) [27], which is the basis of the algorithms introduced in this paper, was formed.

However, subsequent tests on RA have shown that under such challenging situations, RA cannot offer the expected results; therefore, it was determined that the algorithm required further improvement. The main purpose of this paper is to improve RA and present to the literature a new meta-heuristic algorithm that can provide much more successful results than both RA and meta-algorithms commonly used in optimization problems.

While selecting the meta-heuristic studies used in this paper, the algorithms used more commonly in the literature were taken into account. Accordingly, a comprehensive study on meta-heuristic algorithms shows that PSO and GA are the most cited publications (cited approximately 75,000 and 70,000 times by 2022, respectively) [28]. Additionally, in this study, it can be seen that DE, CS, and GWO are the other most cited algorithms. Therefore, to test the performance of the proposed algorithm, BIRA was evaluated by comparing it with five of the most cited meta-heuristic algorithms: Standard GA (SGA), DE, PSO, CS, and GWO.

The contribution of the paper to the literature is as follows:

- Besides improving the previous proposed algorithm, RA, new swarm-based meta-heuristic algorithms called IRA, BRA, and BIRA are introduced to the literature.
- The evaluation involved 20 well-known benchmarks, 11 CEC2014, and 17 CEC2018 test functions (which are also CEC2020 test functions), three real-world engineering problems.

- The significance of the results was assessed using Friedman and Wilcoxon Signed Rank statistical tests.
- BIRA's Matlab code will be made available for sharing in order to test the accuracy of the results described in this paper and to enable researchers to use this new algorithm in their studies.

The paper is organized as follows: Section 2 describes how the idea of new versions of RA were achieved and introduces the proposed algorithms, IRA, BRA, and BIRA. In Section 3, the information of the meta-heuristic algorithms that were employed in this study are given. In addition to presenting the utilized test functions, statistical tests, and the engineering problems, Section 4 also shows the obtained results. Finally, the paper is concluded with Section 5.

2. The New Versions of Roosters Algorithm

There are two key features that distinguish the new proposed algorithms from RA: bipolar behavior and dance technique.

The majority of meta-heuristic algorithms widely used in the literature generally allow only the best individuals to be selected from the population. However, this method causes individuals in a population to become uniform in subsequent generations, meaning that genetic diversity is lost. Unlike other meta-algorithms, algorithms with bipolar behavior have the advantage of increasing genetic diversity by allowing the worst individuals to mate at a certain rate, thus, overcoming problems such as localization.

In addition, the dance technique was designed to improve the RA algorithm enabled the algorithm to provide a more successful performance in investigating the search space, and assist the algorithm to find decent points in the space.

2.1. The Inspired Algorithm (RA)

Female organisms can engage in mating with multiple males [29]. In the case of these polygamous species, a male's reproductive success is influenced by the quality of his sperm, a concept rooted in the "sperm competition" theory [29]. Chickens fall into this category of creatures and have the capacity to mate with multiple roosters, particularly those that are considered attractive.

Similar to humans, the courtship process between roosters and chickens commences with flirtatious behaviors [27]. Typically, a rooster offers food to a chicken and showcases his dancing prowess to capture her interest. If the chicken finds him appealing, mating may occur. Nevertheless, there are instances where a rooster attempts to mate with a chicken without her consent.

On the other hand, the age of the rooster plays a significant role in the mating dynamics [27]. For instance, chickens tend to avoid mating with older roosters due to their limited ability to fertilize eggs effectively.

The stronger rooster can mate with more chickens. However, since a chicken can mate with more than one rooster, sperm obtained from different roosters in the chicken's pouch will compete. At this stage, the VSL (sperm cell velocity) value of the sperm will play a role in determining which one will win this competition [29]. The sperm of the rooster with higher VSL is more appropriate to outcompete his rivals in the race to fertilize the egg.

Following the completion of the mating process, competition among sperm comes into play when a chicken has multiple male partners. The velocity of sperm cells, often estimated by the VSL (sperm cell velocity), holds considerable importance in determining which male is more likely to succeed in fertilizing the egg [29]. A male with higher VSL is more appropriate to outcompete his rivals in the race to fertilize the egg.

Additionally, if a chicken is under stress or dealing with a health issue, it directly impacts the quality of the eggs she produces [27], potentially causing disruptions in the egg's DNA structure.

Inspired by the mating behavior of roosters and chickens, the RA algorithm was introduced [27]. The RA could only give successful results in low populations, but when the population size was increased, it fell behind its counterparts and even got stuck at the local value. Having these disadvantages led to the idea that the algorithm should be improved, and by adding bipolar behavior to the developed algorithm, a new swarm-based meta-heuristic algorithm called BIRA emerged.

2.2. Improved Roosters Algorithm (IRA)

Although the RA algorithm is based on the mating behavior of roosters and chickens, it does not include some behavioral models. For instance, the principle of the rooster influencing the chicken by dancing is not modeled in the RA algorithm. Instead, the algorithm compares the fitness value of the chicken with the fitness value of the rooster who wants to mate. As a result of this comparison, if the value of the rooster is better than the chicken's, it is assumed that the rooster impressed the chicken.

The idea that adding a more advanced mechanism instead of the above mentioned method could positively affect the performance of RA led to the modeling of the Improved Roosters Algorithm (IRA). In IRA, in order to impress the chicken, a new dance model based on the Arithmetic Spiral (AS) [30] has been developed for a rooster. The equation of AS in polar coordinates can be:

$$r = a + b.\theta \tag{1}$$

where r is the length of the radius, θ is the amount of the rotation (angular position of the radius), and "a" and "b" are real numbers. Thanks to the "a" parameter, the spiral is rotated by moving it outwards from the center point, while the distance between the loops is controlled through the "b" parameter. In this way, it is possible for successive turns of the spiral to occur at fixed intervals (the distance between successive turns of the spiral equals 2Πa if θ is measured in radians). This is why the spiral is called arithmetic.

The location of a point obtained using AS in the Cartesian Plane is as follows:

$$x = (vt + c).cos(wt) \tag{2}$$

$$y = (vt + c).sin(wt) \tag{3}$$

where v is the velocity at time t and w is the angular velocity with an arbitrary point $(c, 0, 0)$.

In the IRA, the dance method based on AS, given according to the above equations (Equations (1)–(3)), has been developed. Thanks to this method, the algorithm places the chicken at the center point, allowing the rooster trying to impress her to dance around her according to AS, as shown in Figure 1. The values a and b, which adjust the rotation of the spiral and the distance between its arms, are determined automatically by Matlab functions to create a two-turn spiral depending on the distance of the chicken to the rooster. In addition, for the full implementation of AS, the values v and w are determined to be the position of the rooster and the angle change during the two-turn cycle, respectively. While dancing, if the rooster finds a better position than the chicken has, this refers to the fact that she is influenced by him.

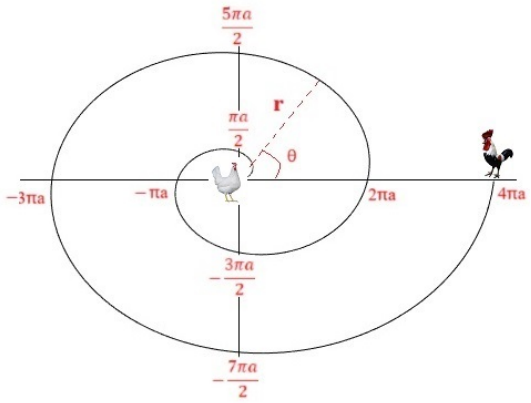

Figure 1. The dance of a rooster around a chicken.

2.3. Bipolar Roosters Algorithm (BRA)

The bipolar behavioral system is a method that allows the selection of the best individual as well as the worst individual in the population at higher rates compared to other methods. Our previous studies, which were mentioned in Section 1, have proved that adapting this system to an algorithm can increase the performance of the algorithm and prevent it from getting stuck at the local minima. This led to the idea of adding bipolar behavior to RA by applying the adaptation process mentioned above, which led to the formation of the Bipolar Roosters Algorithm (BRA).

In fact, the BRA algorithm behaves almost exactly like RA. The only difference with BRA is that the selection process has a bipolar behavior. At this process, depending on the individual's current mood (which is bipolar), the individual mates with either the best mate candidate or the worst mate candidate.

The current mood of the individual is decided through an experimentally determined value, which we call the bipolarity value [25]. In this study, based on the tests we performed in our previous studies, the bipolarity value was accepted as 0.25.

2.4. Bipolar Improved Roosters Algorithm (BIRA)

Although the dancing mechanism in IRA was successful in scanning the search space, it could not provide satisfactory results for some challenging test functions (as will be shown in Section 4) due to getting stuck in local minima. The bipolar behavior of BRA helped to increase the performance of the algorithm and, therefore, much better results were obtained compared to IRA. However, the fact that BRA could not explore the search space sufficiently like IRA affected the performance of the algorithm, and again, the desired results were not obtained.

By taking into account the results acquired from the first two versions, the above-mentioned decent aspects of IRA and BRA were used and the hybrid version of these two algorithms, Bipolar Improved Roosters Algorithm (BIRA), was created (see Algorithm 1).

First of all, the algorithm specifies the input values: bipolarity value and the number of roosters that want to mate with the same chicken, called the "number of cages". As all other meta algorithms do, the BIRA algorithm starts its process by creating an initial population. Individuals in the population are determined as chickens or roosters based on their fitness values. At this stage, the top 50% of the best individuals in the population are considered chickens. Then, BIRA randomly selects a certain number of roosters based on the cage size. Afterwards, each chicken in the population chooses a mate among the roosters in the cage. During this process, which depends on the chicken's current mood, the chicken can choose either the best candidate or the worst candidate as its mate.

Furthermore, the algorithm also permits a chicken to mate with more than one rooster. In that case, in order to decide who is going to fertilize the egg, based on their VSL values, which are actually fitness values, the competition between the sperms of the roosters occur. As a result of the rivalry, the sperm which have the higher VSL value have a chance to fertilize the egg. Finally, the algorithm identifies the gender of the offspring, just as how gender is decided at the beginning of the algorithm.

The results that can be obtained with the algorithm are restricted by the given limit values. However, since the results are real number values, they can go beyond the limit values in some tests, which can cause the algorithm to produce incorrect results. For this reason, a repair section was added to the last stage of the algorithm, and thanks to this, when there is an individual that goes outside the limit value, the relevant value of this individual is assigned as the limit value.

Algorithm 1 Bipolar Improved Roosters Algorithm

```
Identify bipolarity value and number of cages (n)
Create the initial population
Initialize chickens and roosters
i ← 1
while i ≤ populationsize do
  for j = 1 : chicken size do
    Randomly identify roosters in the cage
    Determine the best, r_best, and the worst roosters, r_worst, in the cage
    r_best dances around the chicken_j
    if r_best impresses the attractive chicken && random() ≤ bipolarity value then
      The mate is the best one
    else
      The mate is the worst one
    end if
    if chicken_j has more than one male then
      Calculate VSL values of all sperms
      Allow sperm competitions
      The winner offspring fertilizes the egg
    else
      The male of the offspring fertilizes the egg
    end if
    return offspring_j
  end for
  Identify males and females in the offspring
  i = i + 1
end while
```

3. Utilized Meta-Heuristic Algorithms

3.1. Standard Genetic Algorithm

Holland introduced Genetic Algorithms (GAs) by inspiring Darwin's Evolution Theory in order to establish a novel optimization method [1]. The success of GAs in navigating search spaces [5] has inspired numerous researchers, solidifying GAs as frequently employed algorithms in optimization problems.

The GA initiates its stochastic procedure by randomly generating the initial population, shown in Algorithm 2. Subsequently, the evaluation process commences through a fitness function, computing the fitness value for each individual in the population. Following the establishment of the initial population, the bio-inspired search arises with selection, crossover (also known as recombination), and mutation steps.

Selection methods typically favor the best individuals in the population (having the the highest fitness value) by allowing only them to survive for the next generation, and to mate.

After a selection method chooses adequate individuals for mating, the crossover step commences. In this step, selected individuals are combined to yield offspring superior to their parents [5]. Participation in the crossover step is determined by the crossover probability value, (p_c).

The mutation step aims to preserve genetic diversity and prevent convergence to local minima [5]. Similar to the crossover step, the decision for an individual to undergo mutation is dictated by the mutation probability value, (p_m). By altering genetic structures of individuals, the mutation step assures that the individuals change to become a different individuals from their previous states.

In the last stage of the algorithm, a new generation is obtained by mixing the individuals in the population with the newly generated individuals in a certain ratio. This process repeats, with the algorithm returning to the selection step until either the optimal solution is found or the maximum number of generations is reached.

The ST method is one of the most common selection methods in GAs due to its advantages [31]. GAs utilizing ST as the selection method are commonly referred to as Standard Genetic Algorithms (SGA) in the literature [32].

Algorithm 2 Standard Genetic Algorithm

```
Population size = N;
Randomly create the initial population
Evaluate the fitness value of individuals in the population
count ← 1
while count ≤ The number of iteration do
  % Selection and Crossover
  for i = 1 : N do
    Choose individual i via ST
    if p_i ≤ p_c then
      i attends Crossover
      Randomly choose a partner for individual i via ST
      Mate them
    else
      i is saved for the next generation
    end if
  end for
  % Mutation
  for j = 1 : N do
    Randomly choose individual j
    if p_j ≤ p_m then
      j attends Mutation
      Changes the genetic structures of j
    else
      j passes Mutation step
    end if
  end for
  Combine the new individuals and the current population members
  Evaluate the fitness value of each individual
  count = count + 1
end while
```

3.2. Differential Evolution

Another widely recognized EA employed in optimization problems is the DE algorithm [6], which is a population-based stochastic algorithm.

Similar to GAs, DE begins its procedure by creating the first population (called initial). Unlike GAs, DE advances to the mutation step instead of proceeding with the selection step. While the algorithm is in this step, it generates a vector, known as the *donor vector*, for each individual in the population. The donor vector is composed using three randomly chosen individuals from the population. In the mutation step, the scaling factor F, a crucial parameter controlling differential variation [33], plays a significant role. It is essential that F falls within the range [0, 2] [6].

Following the creation of the donor vector, the recombination (crossover) step commences. This step determines whether the individual or its donor vector is sent for the selection step, resulting in the creation of a *trial vector*. The crossover rate value CR guides this decision-making process. It is recommended to take the value in the range [0, 1] [6].

Finally, the selection process is carried out by comparing the fitness value of each individual with the trial vector. Whichever of these two compared values is better, the individual continues to the next generation with that value (see Algorithm 3).

Algorithm 3 Differential Evolution

Population size = N;
Specify CR (crossover rate) and F (scaling factor) values
Randomly create the initial population
Evaluate the fitness value of individuals in the population
$count \leftarrow 1$
while $count \leq The\ number\ of\ iteration$ **do**
  for $i = 1 : N$ **do**
    Randomly choose three individuals i_1, i_2, i_3 where $i_1 \neq i_2 \neq i_3$
    Generate a random integer $I_{rand} = \{0, 1, 2, ..., D\}$ where D is dimension
    % Trial Vector
    for $j = 1 : D$ **do**
      if $rand(0, 1) \leq CR$ or $j == I_{rand}(i, j)$ **then**
        $trial_i = F * |i_1 - i_2| + i_3$
      else
        $trial_i = pop_i$
      end if
    end for
    Replace $trial_i$, if it is better than pop_i
  end for
  Evaluate the fitness value of individuals in the new population
  $count = count + 1$
end while

3.3. Particle Swarm Optimization

Originally, PSO was not conceptualized for optimization; rather, its inception aimed to emulate the collective behavior of bird flocks and fish swarms in nature [9]. Upon examining how PSO behaved, it became clear that the algorithm was actually performing optimization.

By creating an initial population (swarm), where each member (particle) has a velocity to explore the search space, PSO begins its process, shown in Algorithm 4. On the other hand, two significant positions, the best position called pbest and the global best position called gbest, are also directly affect the movement of particles. pbest, gbest, or both are improved if a particle finds a better position than before.

The velocity vector for the ith individual can be computed as in Equation (4).

$$v_i(t+1) = w.v_i(t) + c_1.r_1.(p_{Best} - x_i(t)) + c_2.r_2.(g_{Best} - x_i(t)) \quad (4)$$

where c_1 and c_2 are acceleration coefficients, w is the inertia weight, r_1 and r_2 are real values from the range $[0, 1]$ (randomly selected), and $x_i(t)$ is the position of the ith individual at time t.

Algorithm 4 Particle Swarm Optimization

```
Number of particles in the swarm = N;
Assign random values to each particle in the swarm
Evaluate the fitness value of particles in the swarm
Specify p_Best and g_Best values
count ← 1
while count ≤ The number of iteration do
  for i = 1 : N do
    Determine the velocity of particle i
    Determine the position of particle i
    Evaluate the fitness value of particle i, f_i
    % For a minimization problem
    if f_i ≤ p_Best then
      Assign f_i as p_Best
      if p_Best ≤ g_Best then
        Assign p_Best as g_Best
      end if
    end if
  end for
  Determine the velocity of particle i
  Determine the position of particle i
  count = count + 1
end while
```

3.4. Cuckoo Search

CS is a swarm-based optimization algorithm that was inspired from the behaviour of cuckoo birds [11]. In the CS algorithm, potential solutions are associated with cuckoo eggs. Cuckoos typically deposit their fertilized eggs in the nests of other cuckoos, hoping that their offspring will be raised by surrogate parents. There are instances when cuckoos realize that the eggs in their nests do not belong to them, leading to either the removal of foreign eggs from the nests or the complete abandonment of the nests.

Lévy flights are stochastic walks where both direction and step lengths are determined by the Lévy distribution. These flights are observed in various animals and insects and are characterized by sequences of straight flights followed by sudden 90-degree turns. Compared to standard random walks, Lévy flights are more efficient for exploring large-scale search areas. This increased efficiency is mainly attributed to the fact that the variances of Lévy flights increase much more rapidly than those of normal random walks [11].

The algorithm, shown in Algorithm 5, is governed by three fundamental rules:

1. Each cuckoo lays one egg at a time and places it in a randomly selected nest.
2. Nests with high-quality eggs are preserved and passed on to the next generations.
3. There is a fixed number of available host nests, and the likelihood of a cuckoo's egg being discovered by the host bird is determined by a probability parameter, p_a, within the range of [0, 1].

In such cases, the host bird has the option to either discard the egg or abandon the nest entirely and construct a brand-new one.

Algorithm 5 Cuckoo Search

Objective function f(x) where $x = (x_1, x_2, ..., x_d)^T$
Generate an initial population of n host nests from x_i where i = 1, 2, ... n
$count \leftarrow 1$
while $count \leq$ *The number of iteration* or stopping criteria **do**
  Randomly get a cuckoo by Lévy flights
  Evaluate the fitness value of the cuckoo, F_i
  Randomly select a nest (say, j) from the host nest
  % For a minimization problem
  if $F_i \leq F_j$ **then**
    Replace j by the new solution
  end if
  A fraction (p_a) of the worst nests are abandoned and new ones are built
  Keep the best solutions
  Rank the solutions and find the current best
  $count = count + 1$
end while

3.5. Grey Wolf Optimizer

GWO drew its inspiration from the hunting tactics observed in grey wolf packs and the intricate dynamics of their hierarchical structure [14]. Within this hierarchy, four distinct wolf roles emerge: alpha, beta, omega, and delta.

Alpha wolves assume leadership within the pack, tasked with pivotal decisions such as hunting strategies, choice of resting grounds, and determining the optimal time to rouse the pack. Betas play a supportive role, aiding alphas in decision-making processes. Deltas, however, encompass a diverse array of functions including scouting (alerting the pack to potential threats), sentineling (guarding the pack), elders (seasoned wolves, formerly alpha or beta), hunters (actively pursuing prey or procuring sustenance), and caretakers (attending to the needs of the infirm or injured). Omegas, often designated as scapegoats, are obliged to defer to the directives of all other ranks. It is upon this intricate social framework that GWO finds its conceptual basis.

GWO initiates its procedures by generating an initial population (referred to as a swarm) through randomization. Following this, it sets in motion its core parameters: $\overrightarrow{A}$ and $\overrightarrow{C}$, see Algorithm 6. $\overrightarrow{A}$ comprises a vector of randomly selected values, falling in the range of $[-1, 1]$, encouraging search agents to diverge from their target. On the other hand, $\overrightarrow{C}$ is also a vector, populated with random values within the range of $[0, 2]$, thereby imbuing GWO with an element of stochastic behavior throughout its runtime [14].

Subsequent to this initialization phase, the positions of grey wolves are established, thereby designating them as alpha, beta, or delta based on their proximity to the prey. This determination of positions extends to omega wolves as well, with their placements being contingent upon the positioning of the most proficient search agents.

Algorithm 6 Grey Wolf Optimizer (GWO)

Population size of grey wolves = N;
Randomly create the initial population
Initialize A and C
Randomly initialize the current positions of wolves
Evaluate the fitness value of wolves in the population
The wolf with the best fitness value is assigned as Alpha
The wolf with the second best fitness value is assigned as Beta
The wolf with the third best fitness value is assigned as Delta
$count \leftarrow 1$
while $count \leq The\ number\ of\ iteration$ **do**
  for $i = 1 : N - 3$ **do**
    Determine the position of wolf i via the positions of best search agents
  end for
  Update the current positions of wolves
  Evaluate the fitness value of wolves in the population
  Update the wolves: Alpha, Beta, and Delta
  $count = count + 1$
end while

4. Tests, Results, and Discussions

4.1. Preliminary Information About Tests

Researchers working in the field of optimization have initially used benchmarks to analyze the algorithms they modelled. With the advancement of technology, these test functions became insufficient; therefore, researchers have started to create new test functions by shifting, rotating, expanding, and hybridizing the well-known benchmarks.

In this paper, in addition to using twenty well-known benchmarks in the field of metaheuristics [34,35], which are shown in Table 1, eleven CEC 2014 test functions [36], see Table 2, and seventeen CE2018 test functions [37], displayed in Table 3, were employed to examine the performance of the algorithms that were used in this article. The dimensions of the test functions in Tables 1–3 were chosen as 2, 10, and 30, respectively. Since the search space of all CEC functions (CEC2014 and CEC2018) is $[-100\ ,\ 100]$, there is no need to add this information to the tables separately.

Table 1. Benchmarks and their features.

Function	Definition	Class	Global Value	Search Area
F1	Ackley	Multimodal	0	[33, −33]
F2	Axis Parallel Hyper−Ellipsoid	Unimodal	0	[5, −5]
F3	Booth	Multimodal	0	[10, −10]
F4	Branins	Unimodal	0.397887	[15, −5]
F5	Sixth Function of Bukin	Multimodal	0	[3, −15]
F6	De Jong	Unimodal	0	[5, −5]
F7	Fifth Function of De Jong	Multimodal	Many	[66, −66]
F8	Drop Wave	Multimodal	0	[5, −5]
F9	Easom	Unimodal	−1	[100, −100]
F10	Goldstein−Price	Unimodal	3	[2, −2]
F11	Griewangk	Multimodal	0	[600, −600]
F12	Langermann	Multimodal	Many	[10, 0]
F13	Michalewicz	Multimodal	−1.8013034	[3, 0]
F14	Rastrigin	Multimodal	0	[5, −5]
F15	Rosenbrock's Valley	Unimodal	0	[10, −5]
F16	Rotated Hyper−Ellipsoid	Unimodal	0	[66, −66]

Table 1. *Cont.*

Function	Definition	Class	Global Value	Search Area
F17	Schubert	Multimodal	Many	[5, −5]
F18	Schwefel	Multimodal	−837.9658	[500, −500]
F19	Six Hump Camel Back	Multimodal	−1	[3, −3]
F20	Sum of Different Powers	Unimodal	0	[1, −1]

Table 2. Utilized CEC2014 test functions and their features.

Function	Definition	Class	Global Value
F21	Sphere Function	Unimodal	−1400
F22	Different Powers Function	Unimodal	−1000
F23	Rotated Weierstrass Function	Multimodal	−600
F24	Rotated Griewank's Function	Multimodal	−500
F25	Rastrigin's Function	Multimodal	−400
F26	Schwefel's Function	Multimodal	−100
F27	Rotated Schwefel's Function	Multimodal	100
F28	Rotated Katsuura Function	Multimodal	200
F29	Lunacek Bi_Rastrigin Function	Multimodal	300
F30	Rotated Lunacek Bi_Rastrigin Function	Multimodal	400
F31	Expanded Griewank's plus Rosenbrock's Function	Multimodal	500

Table 3. Utilized CEC2018 test functions and their features.

Function	Definition	Class	Global Value
F32	Shifted and Rotated Bent Cigar Function	Unimodal	100
F33	Shifted and Rotated Rosenbrock's Function	Multimodal	300
F34	Shifted and Rotated Rastrigin's Function	Multimodal	400
F35	Shifted and Rotated Expanded Scaffer's F6 Function	Multimodal	500
F36	Shifted and Rotated Lunacek Bi_Rastrigin Function	Multimodal	600
F37	Shifted and Rotated Non-Continuous Rastrigin's Function	Multimodal	700
F38	Shifted and Rotated Levy Function	Multimodal	800
F39	Shifted and Rotated Schwefel's Function	Multimodal	900
F40	Hybrid Function 1 (N = 3)	Hybrid	1000
F41	Composition Function 2 (N = 3)	Composition	2100
F42	Composition Function 3 (N = 4)	Composition	2200
F43	Composition Function 4 (N = 4)	Composition	2300
F44	Composition Function 5 (N = 5)	Composition	2400
F45	Composition Function 6 (N = 5)	Composition	2500
F46	Composition Function 7 (N = 6)	Composition	2600
F47	Composition Function 8 (N = 6)	Composition	2700
F48	Composition Function 9 (N = 3)	Composition	2800

The functions used in the CEC 2020 test suite were obtained by selecting some of the CEC 2014 and CEC 2018 functions [38]. Since all functions in the CEC 2020 test suite are included in Table 2 or Table 3 in this paper, it was not necessary to classify the CEC 2020 functions in a distinct table.

In order to display the significance of the results, Friedman and Wilcoxon Signed Rank tests were employed. The Friedman test, originally introduced by Friedman, is a non-parametric statistical test [39]. It is commonly operated to assess variations in the performance of multiple algorithms. In the Friedman test, the test cases are organized in rows, while the outcomes of the compared algorithms are recorded in columns.

On the other hand, the Wilcoxon signed rank test is another non-parametric statistical test used to identify distinctions between two sets of data, which could represent samples or algorithms [40]. In the standard procedure, this test begins by calculating the differences between the outcomes of two algorithms, both of which consist of N observations (where N denotes the number of tests). These differences form a vector, which is then ranked from 1 to N. The smallest value receives a rank of 1. Subsequently, the test computes two values: R_+ and R_-. Based on these values, the test statistic, denoted as T, is determined as the minimum of either R_+ or R_-. This T value is then used to compute the significant probability value (p) associated with the test.

4.2. Results of the Test Functions

Table 4 shows the parameter settings of the meta-heuristic algorithms utilized. Based on standard parameter settings of SGA, DE, PSO, and GWO, respectively in [41], [33], [9] and [42], the parameter settings in Table 4 were chosen. Moreover, the number of iterations and population size (number of nests for CS) were accepted as 100. The code of RA, BIRA, SGA, and PSO were modeled by implementing Matlab 2019a, while the code of GWO, DE, and CS were taken from [42], [43], and [44], respectively.

In order to minimize the effects of randomization, the tests were repeated with 30 runs that have 30 different random seeds. As a result of these 30 runs, the median and standard deviation values of the obtained results are shared in the result tables. The reason for utilizing the median value instead of the mean value is that a highly deviant outcome that may exist within 30 runs can seriously affect the average. The evaluation will be made taking into account the best performance of the compared algorithms. In each table, the values in bold indicate the best value found for that test function.

Table 4. The parameter settings of the algorithms.

Algorithm	Parameter	Value
SGA	Tournament size	3
	Crossover rate	0.7
	Mutation probability	0.05
DE	Crossover rate	0.7
	Scaling factor	0.8
PSO	w	0.2
	c1	2
	c2	2
GWO	A	$[-1, 1]$
	C	$[0, 2]$
RA	Rooster's size	4
BIRA	Number of cages	4
	Bipolarity value	0.25

4.2.1. Comparing Three Versions of RA

In this section, the BIRA algorithm is compared with the IRA and BRA algorithms to show that BIRA is superior to the others.

Table 5 shows the median values of the results obtained from tests with 30 random seeds for each function. When the results given in Table 5 are examined, it is seen that BIRA gives much more successful results than IRA and BRA. Considering the obtained results, we aimed to show the contribution of the BIRA algorithm to the literature by comparing it with the commonly used meta-heuristic algorithms in terms of performance.

Table 5. Obtained results from IRA, BRA, and BIRA.

Functions	IRA	BRA	BIRA
F1	1.36438×10^{-6}	$\mathbf{8.88178 \times 10^{-16}}$	$\mathbf{8.88178 \times 10^{-16}}$
F2	2.55272×10^{-14}	**0**	**0**
F3	$\mathbf{1.74694 \times 10^{-11}}$	8.83137×10^{-10}	4.0894×10^{-10}
F4	**0.397887358**	**0.397887358**	**0.397887358**
F5	0.111910241	**0.1**	**0.1**
F6	2.23152×10^{-14}	**0**	**0**
F7	**0.998003838**	0.998061971	**0.998003838**
F8	**−1**	**−1**	**−1**
F9	**−1**	**−1**	**−1**
F10	**3**	**3**	**3**
F11	0.001981467	**0**	**0**
F12	−4.130034071	**−4.131012716**	−4.129057172
F13	**−1.80130341**	**−1.80130341**	**−1.80130341**
F14	6.15845×10^{-11}	**0**	**0**
F15	2.31066×10^{-14}	0.00961702	**0**
F16	2.2919×10^{-12}	**0**	**0**
F17	**−210.482294**	**−210.482294**	**−210.482294**
F18	**−837.9657745**	**−837.9657745**	**−837.9657745**
F19	**−1.031628453**	**−1.031628453**	**−1.031628453**
F20	5.72526×10^{-19}	**0**	**0**
F21	**−1400**	**−1400**	**−1400**
F22	**−1000**	**−1000**	**−1000**
F23	−599.3679338	−599.2833048	**−599.8192404**
F24	1.4912×10^{13}	**−500**	**−500**
F25	**−400**	**−400**	**−400**
F26	−99.99862854	**−100**	**−100**
F27	1.72134×10^{12}	**100**	**100**
F28	200	**200**	**200**
F29	301.1885887	**300**	**300**
F30	400.768801	400.0983697	**400.0228551**
F31	**500**	**500**	**500**
F32	140.7226999	139.5424729	**100.1913746**
F33	**300**	**300**	**300**
F34	**400**	**400**	**400**
F35	**500**	500.1325064	**500**
F36	**600**	**600**	**600**
F37	702.0308007	702.2932556	**702.0163303**
F38	800.081324	800.7911075	**800**
F39	**900**	**900**	**900**
F40	**1000.312174**	1000.624347	**1000.312173**
F41	**2100**	**2100**	**2100**
F42	**2200**	**2200**	**2200**
F43	**2300**	**2300**	**2300**
F44	2500.000008	2500.000088	**2400**
F45	**2500**	2500.739217	**2500**
F46	**2600**	**2600**	**2600**
F47	2700.017989	2700.069858	**2700**
F48	2895.685681	2939.079316	**2800**

4.2.2. Comparing BIRA with the Metaheuristics

In Table 6, the results of the compared algorithms for 20 test functions commonly used in the literature are shared. If these results are examined, it can be observed that RA offers successful results in only six test functions (F4, F8, F9, F10, F13, and F17), while BIRA, the improved version of RA, presents its best performance in 13 test functions (F1, F2, F4, F6,

F7, F8, F9, F10, F11, F14, F16, F19, and F20). In addition, when the performance of BIRA is compared with the performance of other algorithms, it is possible to observe that BIRA gives superior results compared to other algorithms, except for PSO.

Table 7 shows the results of the algorithms in CEC2014 test functions. Although SGA, GWO, and RA display a much better performance in these test functions, they could not match the performance of BIRA. BIRA offered the best results in all tested functions. On the other hand, PSO could not show its success in the first test (shown in Table 6) and fell far behind SGA, GWO, RA, and BIRA algorithms.

According to Table 8, which demonstrates the results of CEC2018 test functions, the BIRA algorithm manages to find the global values of the functions in 15 of the 17 test functions used (except F32 and F40) and introduces predominant results compared to all compared algorithms. Even though it provides the best result for F40, it could not reach the global value of 1000. The same problem can be seen for the F32 function. Although BIRA's inclusion of bad individuals in the selection process prevents local congestion, it can also cause deviation from the best in some cases, as with these functions. Furthermore, PSO performed better in this test than in the previous one; however, it still falls short of BIRA.

Figure 2 demonstrates the convergence curves of the compared algorithms in benchmarks. Based on the figure, BIRA is able to find the best result for F1, F2, F6, F8, F11, F15, F17, and F21 in the first iteration. Hence, BIRA's convergence curve may not be seen in these test functions. When Figure 2 is examined, it can be seen that GWO and PSO give the closest performance to BIRA. Although CS and DE generally improve the best results they find during each iteration, they cannot approach the performance of BIRA. Although SGA and RA provide successful results in some test functions such as F10, they are generally stuck in the locals and cannot reach the global values.

While Figure 3 displays the trajectory of the algorithms in the CEC2014 test functions, Figure 4 shows the convergence curves in tested CEC2018 functions. In Figure 3, all algorithms were able to find the best result in the first iteration for the functions of F22 and F28. Additionally, BIRA was able to discover the global value in the first 15 iterations in all functions, except F27 and F30 functions. Considering the performances of other algorithms as indicated in Figure 3, it is seen that DE and CS are stuck at the local minimums at the end of the first iterations and do not show any improvement. On the other hand, it can be observed that RA, SGA, PSO, and GWO give more successful results and continue to search for the best value throughout the iterations.

Besides, BIRA shows superior success in all of the test functions shown in the Figure 4 and managed to reach the global value in its initial iterations. Based on the performance of other algorithms, it is obvious that PSO offers the closest performance to BIRA. While DE and CS generally continue to get stuck locally, SGA, RA, and GWO are successful in searching the space but they are not successful in finding the global value.

Table 6. The results of the benchmarks.

Methods	SGA		DE		PSO		CS		GWO		RA		BIRA	
Function	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**
F1	0.001255383	0.21086953	9.32339×10^{-9}	4.70465×10^{-8}	**8.88178** $\times 10^{-16}$	0	0.005537351	0.007143004	8.88178×10^{-16}	0	8.68705×10^{-6}	0.000350781	**8.88178** $\times 10^{-16}$	0
F2	4.56022×10^{-7}	1.15096×10^{-5}	3.90718×10^{-19}	4.2705×10^{-18}	1.56782×10^{-45}	5.92298×10^{-43}	7.41274×10^{-9}	2.37002×10^{-8}	1.90165×10^{-88}	2.31024×10^{-82}	3.39742×10^{-12}	1.76495×10^{-6}	**0**	0
F3	1.76142×10^{-5}	0.001168238	0.150005705	0.187894262	**0**	0	2.83211×10^{-8}	7.5136×10^{-8}	2.68482×10^{-6}	1.53986×10^{-6}	2.21762×10^{-9}	0.002585064	0.001473631	0.005401886
F4	0.397892113	0.00053846	0.44846131	0.059982806	**0.397887358**	0	0.397887996	2.52321×10^{-6}	0.397890178	4.39762×10^{-5}	**0.397887358**	0.000183193	**0.397887358**	0.001414507
F5	**0.017140132**	0.65868877	6.46979681	4.276563449	0.029654739	0.016355628	0.118091719	0.038444932	0.238699616	0.089605473	0.062106031	0.187623147	0.880331438	0.497230168
F6	3.91679×10^{-6}	7.50843×10^{-5}	7.39729×10^{-19}	1.79729×10^{-16}	6.12258×10^{-43}	4.38547×10^{-42}	3.09764×10^{-9}	1.27952×10^{-8}	5.15634×10^{-92}	4.71801×10^{-79}	4.87848×10^{-13}	9.26803×10^{-7}	**0**	0
F7	**0.998003838**	0.314174056	**0.998003838**	0.286095652	**0.998003838**	1.254478461	**0.998003838**	2.09837×10^{-5}	1.990054498	3.001257169	2.71049356	0.005637995	**0.998003838**	0.415112038
F8	−0.986790052	0.025601172	**−1**	0.000855671	**−1**	0.020160998	**−1**	0.000774638	**−1**	0	**−1**	0.002415216	**−1**	0
F9	−0.680168485	0.472333796	−0.926679733	0.082815898	**−1**	0	−0.982954854	0.027324701	**−1**	5.82547×10^{-6}	**−1**	0.309335018	**−1**	0.000505397
F10	**3**	0.002495585	3.254522018	0.327966517	**3**	2.05116×10^{-15}	**3**	1.89619×10^{-5}	**3**	2.64525×10^{-5}	**3**	0.004444455	**3**	0.001442786
F11	0.024931817	0.018285389	0.000444261	0.003246811	**0**	0.004198913	0.00372588	0.002488055	0.007397297	0.0040038	0.007568114	0.011961397	**0**	0
F12	−4.12399423	0.459663532	−3.640964036	0.635895625	−4.127576741	0.026266062	−4.151315657	0.004717896	**−4.155700465**	0.013606751	−4.127575369	0.022678944	−4.152015919	0.019450089
F13	−1.801302999	0.000263827	−1.793002634	0.085143795	**−1.80130341**	0	−1.801303326	1.82501×10^{-7}	−1.801253697	3.02298×10^{-5}	**−1.80130341**	1.74556×10^{-13}	−1.772639533	0.026774072
F14	0.002355076	0.329522764	0.000624907	0.289808584	**0**	0.31463368	0.001224302	0.000874741	**0**	0	1.23787×10^{-7}	0.000581032	**0**	0
F15	0.004964854	0.062455909	0.003011576	0.001763669	**0**	0	1.44603×10^{-5}	6.85264×10^{-5}	1.84457×10^{-6}	1.5792×10^{-6}	4.20316×10^{-5}	0.001771121	0.001283762	0.00263776
F16	0.000383693	0.006796323	9.81193×10^{-17}	4.07883×10^{-16}	2.45812×10^{-40}	6.63358×10^{-39}	2.21338×10^{-7}	2.7021×10^{-6}	2.51032×10^{-85}	2.0348×10^{-72}	2.2815×10^{-9}	3.3256×10^{-5}	**0**	0
F17	−210.4164534	14.02226014	−209.9070472	0.744934948	**−210.482294**	5.99182 $\times 10^{-14}$	−210.4467086	0.080444711	−210.4822688	14.17420461	**−210.482294**	14.17406413	−210.4753996	0.005144005
F18	−837.9118592	1.144757616	-4.94982×10^{15}	4.61046×10^{15}	**−837.9657745**	61.161293	−837.9544671	0.033940138	−837.9536623	49.93519259	−837.9656277	0.002715216	−837.9241669	0.335437403
F19	−1.031621243	0.000733737	−1.031628049	8.58228×10^{-6}	−1.031628453	2.34056×10^{-16}	−1.031628441	9.4721×10^{-9}	−1.03162835	8.58609×10^{-8}	−1.031628453	3.67124×10^{-6}	**−1.031484529**	0.000162441
F20	5.02617×10^{-10}	5.19065×10^{-6}	5.47094×10^{-24}	1.48927×10^{-21}	4.3503×10^{-45}	2.99516×10^{-43}	3.62377×10^{-12}	9.05276×10^{-12}	7.86221×10^{-86}	3.91143×10^{-75}	8.04×10^{-22}	3.2225×10^{-12}	**0**	0

Table 7. The results of the CEC2014 test functions.

Methods	SGA		DE		PSO		CS		GWO		RA		BIRA	
Function	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**	**Med.**	**Stdv.**
F21	**−1400**	0	**−1400**	1.21536×10^{-13}	**−1400**	0	−1338.806573	147.2269615	**−1400**	0	**−1400**	0	**−1400**	0
F22	**−1000**	0	**−1000**	0.000155518	**−1000**	0	−975.114814	10.195097	**−1000**	0	**−1000**	0	**−1000**	0
F23	−599.4367431	0.120880547	−599.0285293	0.374881727	−599.3240103	0.17259186	−598.7555714	0.282068685	−599.3514859	0.148659384	−599.398061	0.127414184	**−599.8192404**	0.156073768
F24	**−500**	0	585480767	5649826967	−499.992604	0.009101558	10000000000	0	−499.9926012	0.005782645	−499.9926	0	**−500**	0.007208996
F25	**−400**	0	−399.9230974	0.274078088	**−400**	0.733074289	−393.9446495	4.299776882	**−400**	0	−399.9230974	0	**−400**	0
F26	**−100**	0	−91.47593149	9.666212569	−99.37565349	12.27526502	23.95348888	95.39028355	**−100**	0.237106642	**−100**	0	**−100**	0.241629941
F27	**100**	0	50658512.92	1.98676×10^{11}	100.6243465	8.088389982	**100**	0	100.3153375	31.08688472	**100**	0	**100**	4.278986666
F28	**200**	0	**200**	0	**200**	0	**200**	0	**200**	0	**200**	0	**200**	0
F29	**300**	0	303.6375124	0.916037822	302.0499329	0.774397081	307.6482223	2.212854927	302.0371546	0.848517265	303.630541	0	**300**	0.935655053
F30	400.2843609	0.351942669	403.2819854	0.819129445	402.0441654	0.971830428	405.1775388	2.802958474	402.0559464	0.045123748	400.2730914	0.186846122	**400.0228551**	0.845632081
F31	**500**	0	500.0971477	0.049512952	**500**	0.025470474	501.0337751	0.99027054	**500**	0.008249264	**500**	0	**500**	0.008259574

Table 8. The results of the CEC2018 test functions.

Methods	SGA		DE		PSO		CS		GWO		RA		BIRA	
Function	Med.	Stdv.	Med.	Stdv.	Med.	Stdv.	Med.	Stdv.	Med.	Stdv.	Med.	Stdv.	Med.	Stdv.
F32	3733.047508	1982.314114	21219.49737	136390.19	**100**	3.51988×10^{-5}	670790.3426	6853893.931	3623.867616	2098.689338	3211.281068	1873.978709	100.1913746	319.3145928
F33	305.8220134	5.790716103	337.137691	123.1283951	**300**	0	494.2941107	613.0535187	300.0001298	0.000177653	304.3859047	7.983328908	**300**	0
F34	400.0058671	0.00241195	400.0081386	0.016296401	**400**	0	400.2416491	0.34340415	400.0000011	0.002072102	400.0057821	0.001972981	**400**	0.003970803
F35	501.2674373	0.332339626	501.1943444	0.437915103	**500**	0.504536328	505.1343593	2.54250947	500.0002161	0.256838113	500.660966	0.487130477	**500**	0.256897324
F36	602.8636419	0.885741706	605.0460519	2.663192685	**600**	0	612.5651632	5.025044491	600.0322167	0.026030772	603.1431628	1.704578608	**600**	0
F37	702.4610542	0.500189189	701.3133625	0.716806059	702.0163303	1.070590377	708.1887943	3.016417172	702.0694762	0.065216495	702.4608958	0.250050535	**700**	0.529619581
F38	801.2411375	0.565684226	801.2411375	0.528087033	**800**	0.485490308	804.938814	2.612302825	800.0001848	0.34998426	801.0598326	0.456294059	**800**	0.256897324
F39	900.4974524	0.232615384	900.3795356	1.534062494	**900**	0	908.3854275	6.045084294	900.0000033	4.33222×10^{-6}	900.276645	0.247507428	**900**	0
F40	1007.349164	6.523580623	1008.88372	8.423574352	1000.624347	7.518652985	1089.381983	59.20614512	1000.626185	15.47141869	1010.047227	6.486059894	**1000.312173**	0.225934415
F41	2103.931704	3.270594593	2104.429522	3.299763333	**2100**	0	2132.108561	21.01620031	2100.023419	0.016527011	2103.476163	2.13588005	**2100**	0
F42	2207.056553	5.223529849	2220.644636	6.642326434	**2200**	0	2236.888667	19.69467916	2200.020739	0.01753867	2205.687197	5.540949037	**2200**	0
F43	2319.984204	6.954994809	2318.005232	10.33034652	**2300**	0	2379.72972	85.58481772	2300.069152	0.040891435	2313.384514	9.254544324	**2300**	0
F44	2417.399203	30.43009793	2439.510105	16.39879916	**2400**	35.18657753	2468.158674	38.98455995	2400.823459	35.24296832	2407.233297	26.93127988	**2400**	48.7689698
F45	2500.31963	52.13175542	2501.869345	44.76147861	**2500**	50.71025921	2538.305402	78.44271604	2500.004122	48.79054796	2500.897851	54.49892813	**2500**	51.64053456
F46	2620.561692	8.818209562	2649.682581	18.26688104	**2600**	0	2704.255671	39.83876813	2600.165549	0.068705286	2619.657675	11.49987581	**2600**	0
F47	2731.544225	10.70731481	2747.966241	19.18967408	**2700**	1.19×10^{-11}	2765.569686	51.24391461	2701.104677	0.334538924	2723.403363	7.153095241	**2700**	0.007345873
F48	2911.787209	24.56099767	2885.239105	37.1951585	2900	79.88086367	2996.965144	42.08608749	2900.801663	71.06671983	2907.072421	31.3928188	**2800**	58.3242412

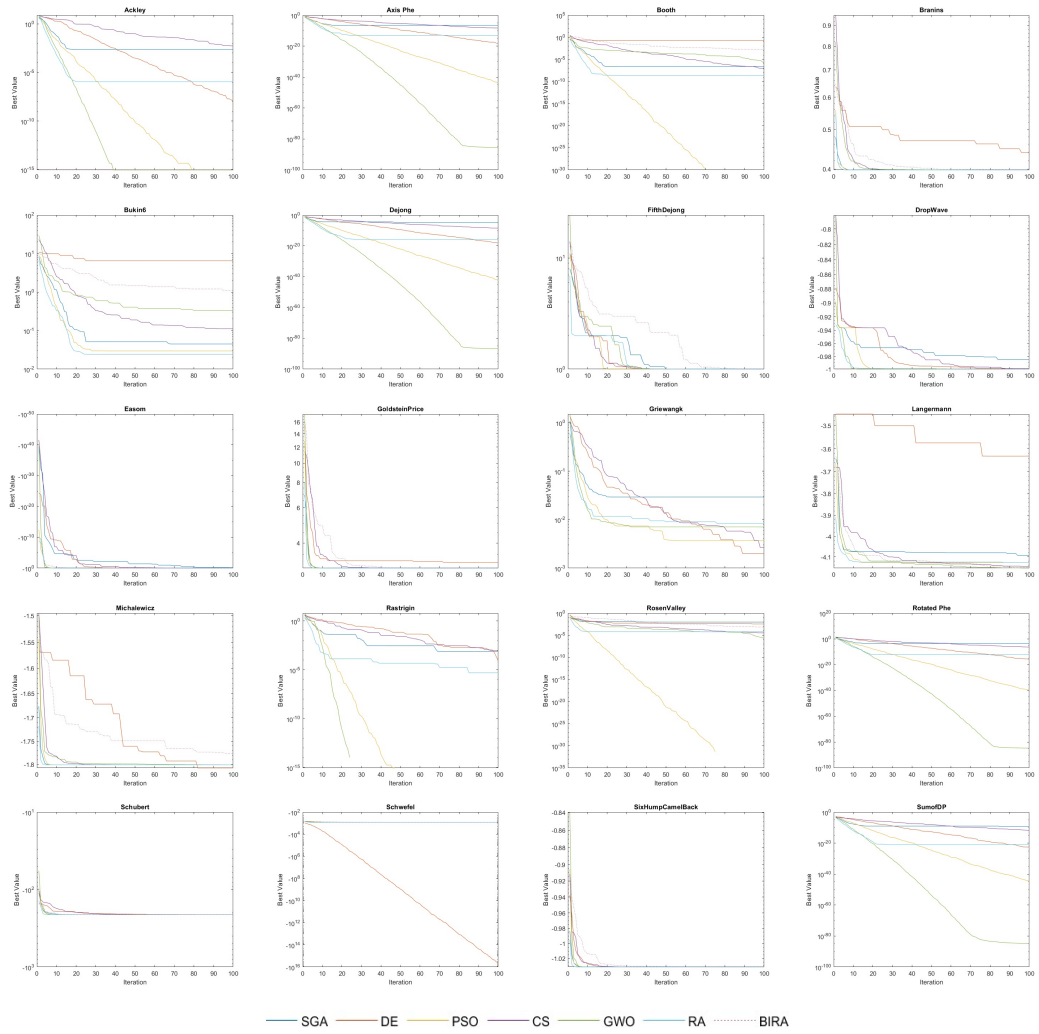

Figure 2. Convergence curve of the algorithms in benchmarks.

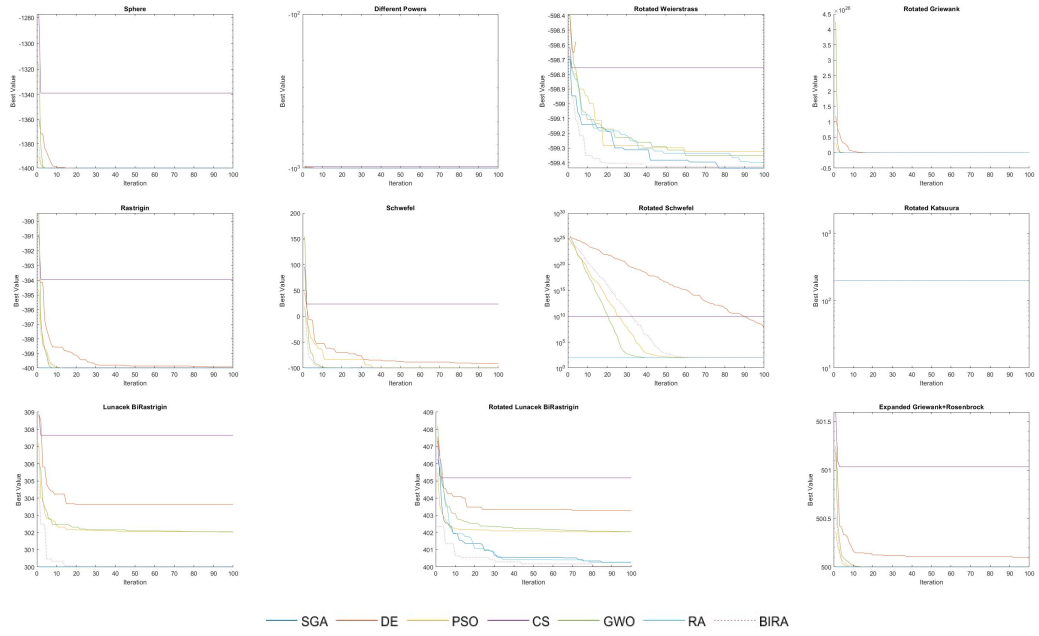

Figure 3. Convergence curve of the algorithms in the CEC2014.

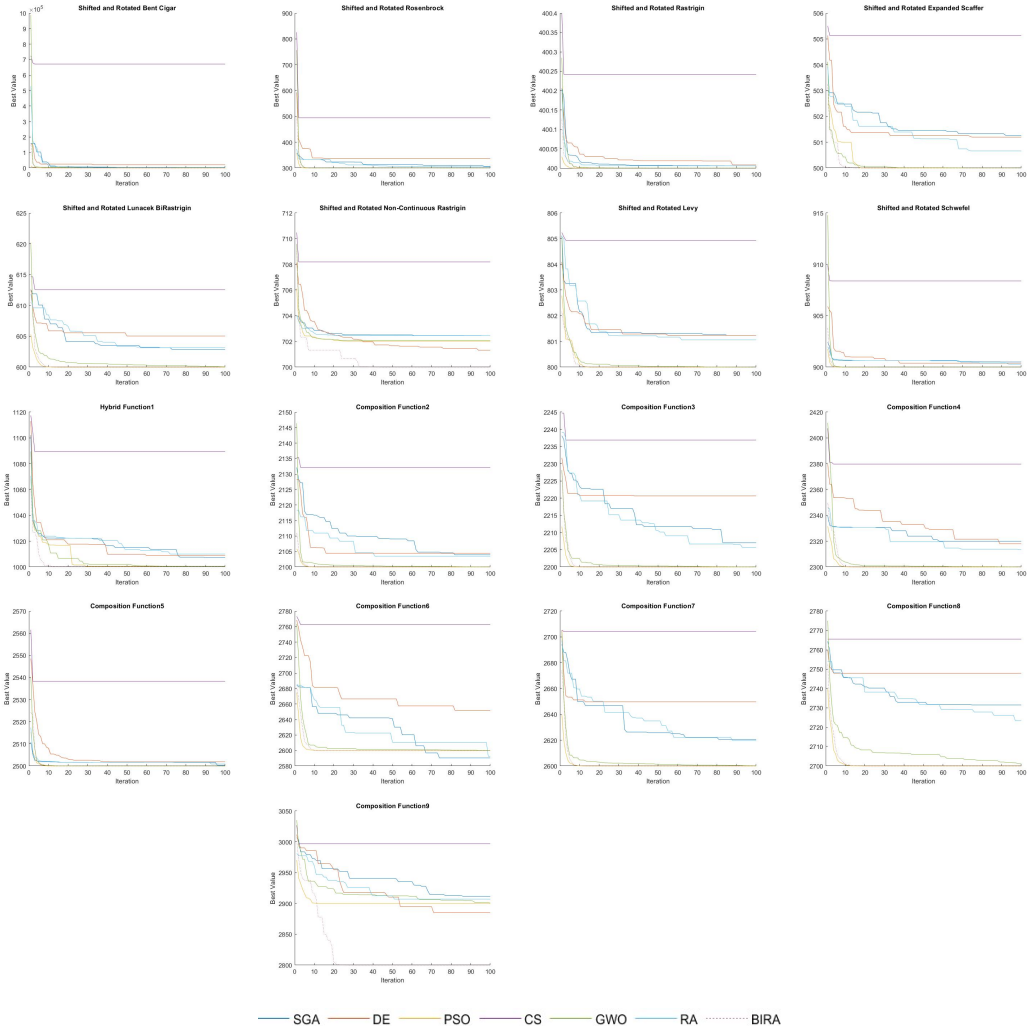

Figure 4. Convergence curve of the algorithms in the CEC2018.

4.3. Statistical Results

IBM SPSS Statistics 22 was performed to acquire the test results of the Friedman while the *signrank* function of Matlab 2019a was utilized to obtain the test results of the Wilcoxon Signed Rank.

The Friedman Test commences its procedure by assigning ranks to each row based on the values within the respective columns. Subsequently, it calculates the total rank values for each column. To evaluate this test, the X^2 statistic (also known as Chi-square) value, and the k − 1 degrees of freedom (df) value must be known, where k means the number of compared algorithms. The appropriate Chi-square values corresponding to the degrees of freedom can be found in [45]. For instance, in our case, with df = 6 and $\alpha = 0.05$, the expected X^2 value is 12.592. If the computed X^2 value exceeds the expected one, it leads to the rejection of the hypothesis that states "There is no difference between compared algorithms".

It becomes evident which algorithm excels by referencing Table 9. An algorithm with the lowest rank value indicates superior performance in comparison to the others. Table 9 clearly shows that BIRA offers a better performance when comparing it to the other tested algorithms. Moreover, not only do the probability values fall below $\alpha = 0.05$, but the computed X^2 values in Table 10 also surpass the expected value of 12.592. Consequently, the hypothesis must be rejected, signifying that "There is a difference between compared algorithms". Nevertheless, this test does not specifically identify which algorithms differ

from the others. Therefore, the Wilcoxon Signed Rank Test is employed to elucidate these differences.

Table 9. The Friedman test mean rank values.

Methods	Mean Rank
SGA	4.87500
DE	5.17708
PSO	2.67708
CS	5.84375
GWO	3.41666
RA	3.61458
BIRA	**2.39583**

Table 10. Statistical values for the Friedman test.

Statistical Variant	Value
Number of functions	48
X^2	116.99142
df	6
Probability value	6.97748×10^{-23}

The Wilcoxon Signed Rank Test is performed in pairs: one algorithm is systematically selected from the first column of the table and then compared with all subsequent algorithms. Additionally, the p values indicate the degree of similarity between the compared algorithms. A value of '0' implies no similarity between the outcomes of the compared algorithms, while '1' indicates that they are identical.

In Table 11, it can be seen that the p values resulting from the Wilcoxon Signed Rank Test are generally very close to 0. This infers that no algorithm is similar to another in terms of their performance on all test functions. However, it is obvious to say that the performance of BIRA and PSO are closer than the others since their pairwise value is 0.269221.

Table 11. The pairwise results of the Wilcoxon Signed Rank Test.

Methods	SGA	DE	PSO	CS	GWO	RA	BIRA
SGA	1	0.005677	0.000357	3.545623×10^{-5}	0.000474	0.000370	4.121701×10^{-6}
DE		1	3.050312×10^{-6}	1.338213×10^{-5}	1.338213×10^{-5}	0.00012	3.463021×10^{-7}
PSO			1	1.068801×10^{-8}	0.000706	0.000239	0.269221
CS				1	1.013714×10^{-6}	4.547581×10^{-8}	6.118679×10^{-7}
GWO					1	0.059862	0.000991
RA						1	0.000236
BIRA							1

4.4. Real-World Optimization Problems

To analyze the performance of the algorithms, three well-known real-world engineering problems were utilized: pressure vessel design, gear train design, and tension/compression spring design.

The parameter settings given in Table 4 were applied while testing the algorithm performance in the real-life scenarios. While the algorithms used in this study were tested in real-life scenarios, each algorithm was run 30 times using 30 different random seeds and the best results obtained from these 30 repetitions were shared.

The pressure vessel design and tension/pressure spring design problems used in this section contain constraints. The algorithms must produce results by taking these constraints into account. However, as can be expected, some results in the tests performed

may not comply with these constraints. That is why, for each iteration, if individuals in the population violate the constraints, these individuals are sentenced to death and the new generation is recreated from variants of individuals that did not violate the rules, and the process continues.

4.4.1. Pressure Vessel Design Problem

The pressure vessel, a containment unit for various gas or liquid pressures of different sizes, is susceptible to critical faults during manufacturing or operation, which could lead to severe damage or injury. Hence, it is crucial to design the pressure vessel with the correct structure [46]. The design of the vessel, shown in Figure 5, necessitates the adjustment of several parameters (materials): shell thickness ($T_s = x_1$), spherical head thickness ($T_h = x_2$), radius of cylindrical shell ($R = x_3$), and shell length ($L = x_4$).

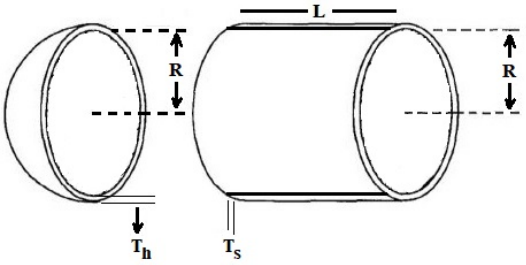

Figure 5. The structure of a pressure vessel (taken from [25]).

The aim of the problem is to minimize the total cost of the materials, given in Equation (5), without violating any constraints (given in Equations (6)–(11)).

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_3^2x_2 + 3.1611x_1^2x_4 + 19.8621x_1^2x_3 \quad (5)$$

$$g_1(x) = 0.0193x_3 - x_1 \leq 0 \quad (6)$$

$$g_2(x) = 0.00954x_3 - x_2 \leq 0 \quad (7)$$

$$g_3(x) = 750X1728 - \pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 \leq 0 \quad (8)$$

$$g_4(x) = x_4 - 240 \leq 0 \quad (9)$$

$$g_5(x) = 1.1 - x_1 \leq 0 \quad (10)$$

$$g_6(x) = 0.6 - x_2 \leq 0 \quad (11)$$

where $0.0625 \leq x_1, x_2 \leq 6.1875$ and x_1 and x_2 are integer multipliers of 0.0625 in, and x_3 and x_4 are continuous integer values in the respectively following interval: 40 in $\leq x_3 \leq$ 80 in and 20 in $\leq x_4 \leq$ 60 in.

In Table 12, the best results where the algorithms did not violate the constraints (given in Equations (6)–(11)) are shared.

In the pressure vessel design problem, PSO could not show the performance it showed in the test functions and fell behind GWO and BIRA. According to Table 12, BIRA presented the most successful result compared to all tested algorithms by reaching a value of 7140.59.

Table 12. The results of algorithms the for designing the pressure vessel.

Values	SGA	DE	PSO	CS	GWO	BIRA
x_1	1.2526	4.4106	1.267	1.2211	1.2614	1.1
x_2	2.5953	10.6419	0.6263	0.6228	0.624	0.6
x_3	55.492	52.8848	65.6414	56.729	64.9616	55.4458
x_4	75.5108	145.4679	10	56.6558	11.1642	60.5331
$g_1(x)$	−0.1816	−3.39	-1.3294×10^{-4}	−0.1262	−0.0076	−0.0299
$g_2(x)$	−2.0659	−10.1374	-5.6956×10^{-5}	−0.0816	−0.0043	−0.071
$g_3(x)$	-1.6963×10^{5}	-6.017×10^{5}	-2.4103×10^{4}	-4.1523×10^{4}	−320.3189	-2.6226×10^{3}
$g_4(x)$	−162.4892	−94.5321	−230	−183.3442	−228.8358	−179.4669
$g_5(x)$	−0.1526	−3.3106	−0.167	−0.1211	−0.1614	0
$g_6(x)$	−1.9953	−10.0419	−0.0263	−0.0228	−0.024	0
Cost ($)	19676.01	103412.12	7457.3	7952.03	7358.68	**7140.59**

4.4.2. Gear Train Design Problem

In the gear train design problem, the objective is to approach a gear ratio value of 1/6.931, or 0.1442793, as closely as possible [47]. As shown in Figure 6, a set of four gears constitutes the train. The tuning process involves determining the numbers of teeth on these gears.

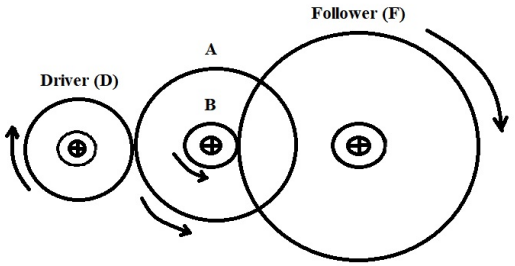

Figure 6. The set of four gears of the train (taken from [25]).

If x_1, x_2, x_3, and x_4 represent the numbers of teeth on gears A, B, D, and F, respectively, the goal is to minimize Equation (12):

$$f(x) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2 \tag{12}$$

where x_1, x_2, x_3 and x_4 must be in the interval of [12, 60] and gear ratio is:

$$gear_{ratio} = \frac{x_3 x_2}{x_1 x_4} \tag{13}$$

If the performances of the algorithms given in Table 13 are examined, it is obvious that the performances of the PSO, CS, and BIRA algorithms are better than the other algorithms. Though the gear ratio values, obtained from Equation (13), of all three algorithms are the same (the first six digits of the decimal part of the results), it can be observed that PSO is more successful than BIRA and CS (albeit by a small margin).

Table 13. The results of the algorithms for designing the gear train.

Values	SGA	DE	PSO	CS	GWO	BIRA
x_1	51.6703	35.71	44.9135	60	45.5971	39.5025
x_2	15.1419	13.3157	14.0528	30.2716	31.1309	12
x_3	15.7607	22.4754	18.2749	14.0054	12	16.3961
x_4	31.8489	56.8036	39.6311	48.9752	56.7843	34.5217
Gear ratio	0.1450175	0.1475386	**0.1442795**	0.1442791	0.1442804	0.1442796
Cost	5.4429×10^{-7}	1.0627×10^{-5}	**0**	1.1755×10^{-15}	1.216×10^{-12}	2.3559×10^{-28}

4.4.3. Tension/Compression Spring Design Problem

Firstly, Belegundu [48] presented the tension/compression spring design problem, shown in Figure 7. The main purpose of the problem is to minimize the cost of three parameters: wire diameter d (x_1), mean coil diameter D (x_2), and the number of active coils P (x_3). In addition to minimizing Equation (14), it must not violate any constraints from Equations (15)–(18).

$$f(x) = (x_3 + 2)x_2x_1^2 \tag{14}$$

$$g_1(x) = 1 - \frac{x_2^2x_3}{71785x_1^4} \leq 0 \tag{15}$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \tag{16}$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \tag{17}$$

$$g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \leq 0 \tag{18}$$

where $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.30$ and $2 \leq x_3 \leq 15$.

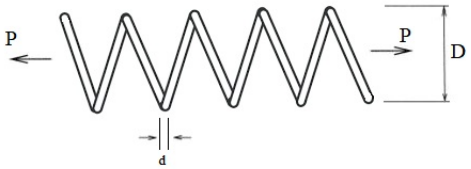

Figure 7. Tension/compression spring design.

When the performances of the algorithms in Table 14 are observed, it can be seen that SGA and DE are stuck at the same local value. Although PSO and GWO provide decent performances, they are behind BIRA.

Table 14. The results for designing the tension/compression spring.

Values	SGA	DE	PSO	CS	GWO	BIRA
x_1	0.1262	0.1262	0.0793	0.1369	0.08622	0.08888
x_2	0.4283	0.4283	0.4350	0.3629	0.3235	0.2775
x_3	13.4159	13.4159	6.3168	4.9342	8.3225	2.3680
Cost	0.1050895	0.1050895	0.0227434	0.04714145	0.02482801	**0.0095515**

5. Conclusions

This study introduces novel swarm-based meta-heuristic algorithms, IRA, BRA, and BIRA, which are modeled on the mating behavior of chickens and roosters. First of all, these three new versions were compared among themselves in terms of performance and it was observed that BIRA offered more successful results.

To evaluate the contribution of BIRA's performance to the literature, it is compared with well-known metaheuristic algorithms: SGA, DE, PSO, CS, and GWO. Additionally, RA was included in this assessment to highlight BIRA's superior performance compared to its predecessor (RA). This comparison involved the use of 20 widely recognized benchmark optimization functions, 11 CEC2014 test functions, and 17 CEC2018 test functions (which are also in the CE2020 test suite). Furthermore, Friedman and Wilcoxon Signed Rank statistical

tests were conducted to establish the significance of the results. Moreover, three real-world engineering problems, pressure vessel, gear train design, and tension/compression spring design, were also utilized to test the performance of the algorithms.

When all the tests are examined, it is obvious that BIRA offers a much superior performance compared to RA. While the BIRA algorithm analyzes the search space better with the dance technique, it also tries to prevent it from getting stuck in local minima with its bipolar behavior.

BIRA showed successful results not only compared to RA but also compared to all tested algorithms, albeit by a small margin compared to PSO. However, BIRA not only produced new values for studies in the field of real-world engineering problems, but also provided reasonable results. Thanks to the superior performance offered by BIRA, it has been proven that bipolar behavior has a good effect on meta-heuristic algorithms.

Even though bipolar behavior is beneficial in reaching different points of the algorithm, it is obvious that the number of the worst individuals will decrease since the population will evolve throughout iterations. For this reason, one of the most important disadvantages of the algorithm is that the mutation phase, which allows the discovery of different points of an algorithm, is not included in this algorithm.

Hence, a mutation phase will be added to the algorithm as a future work, and then, its performance will be analyzed in various optimization challenges. Moreover, based on the idea that bipolar behavior can improve the performance of meta-heuristic algorithms, bipolar behavior will be added to several meta-heuristics and the performance of these algorithms will be examined.

This study contributes to the literature by providing a new perspective on algorithm design, demonstrating that bipolar behavior and dance-inspired movements can enhance optimization performance. These findings highlight the potential of biologically inspired techniques in advancing meta-heuristic algorithms and solving complex optimization problems more effectively.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IRA	Improved Roosters Algorithm
BRA	Bipolar Roosters Algorithm
BIRA	Bipolar Improved Roosters Algorithm
RA	Roosters Algorithm
SGA	Standard Genetic Algorithm
GAs	Genetic Algorithms
DE	Differential Evolution
PSO	Particle Swarm Optimization
CS	Cuckoo Search
GWO	Grey Wolf Optimizer
EAs	Evolutionary Algorithms

MAs	Memetic Algorithms
BBO	Biogeography-Based Optimizer
SBA	Swarm-based Algorithms
ACO	Ant Colony Optimization
ABC	Artificial Bee Colony
FFA	Firefly Algorithm
SA	Simulated Annealing
BBBC	Big-Bang Big-Crunch
GS	Gravitational Search
BH	Black Hole
MSA	Momentum Search Algorithm
TS	Tabu Search
HS	Harmony Search
ICA	Imperialist Competitive Algorithm
TLBO	Teaching Learning Based Optimization
FWA	Fireworks Algorithm
BMT	Bipolar Matching Tendency
ST	Standard Tournament
AS	Arithmetic Spiral

References

1. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA , 1992.
2. Baker, J.E. Reducing bias and inefficiency in the selection algorithm. In Proceedings of the Second International Conference on Genetic Algorithms, Cambridge, MA, USA, 28–31 July 1987; Volume 206, pp. 14–21.
3. Greffenstette, J.J.; Baker, J.E. How genetic algorithms work: A critical look at implicit parallelism. In Proceedings of the 3rd International Conference on Genetic Algorithms, Fairfax, VA, USA, 4–7 June 1989; pp. 20–27.
4. Whitley, D.; Kauth, J. *GENITOR: A Different Genetic Algorithm*; Department of Computer Science, Colorado State University: Fort Collins, CO, USA, 1988.
5. Alhijawi, B.; Awajan, A. Genetic algorithms: Theory, genetic operators, solutions, and applications. *Evol. Intell.* **2024**, *17*, 1245–1256. [CrossRef]
6. Ahmad, M.F.; Isa, N.A.M.; Lim, W.H.; Ang, K.M. Differential evolution: A recent review based on state-of-the-art works. *Alex. Eng. J.* **2022**, *61*, 3831–3872. [CrossRef]
7. Zhu, Y.; Tang, Q.; Cheng, L.; Zhao, L.; Jiang, G.; Lu, Y. Solving multi-objective hybrid flowshop lot-streaming scheduling with consistent and limited sub-lots via a knowledge-based memetic algorithm. *J. Manuf. Syst.* **2024**, *73*, 106–125. [CrossRef]
8. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
9. Abualigah, L.; Sheikhan, A.; Ikotun, A.M.; Zitar, R.A.; Alsoud, A.R.; Al-Shourbaji, I.; Hussien, A.G.; Jia, H. Particle swarm optimization algorithm: Review and applications. In *Metaheuristic Optimization Algorithms*; Elsevier: Amsterdam, The Netherlands, 2024; pp. 1–14.
10. Shen, Z.; Leite, W.; Zhang, H.; Quan, J.; Kuang, H. Using ant colony optimization to identify optimal sample allocations in cluster-randomized trials. *J. Exp. Educ.* **2025**, *93*, 167–185. [CrossRef]
11. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; IEEE: New York, NY, USA, 2009; pp. 210–214.
12. Karaboga, D.; Akay, B. A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
13. Yang, X.S. Firefly algorithms for multimodal optimization. In *International Symposium on Stochastic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 169–178.
14. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
15. Guilmeau, T.; Chouzenoux, E.; Elvira, V. Simulated annealing: A review and a new scheme. In Proceedings of the 2021 IEEE Statistical Signal Processing Workshop (SSP), Rio de Janeiro, Brazil, 11–14 July 2021; IEEE: New York, NY, USA, 2021; pp. 101–105.
16. Erol, O.K.; Eksin, I. A new optimization method: Big bang–big crunch. *Adv. Eng. Softw.* **2006**, *37*, 106–111. [CrossRef]
17. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
18. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [CrossRef]
19. Dehghani, M.; Samet, H. Momentum search algorithm: A new meta-heuristic optimization algorithm inspired by momentum conservation law. *Appl. Sci.* **2020**, *2*, 1720. [CrossRef]

20. Szénási, S.; Légrádi, G. Machine learning aided metaheuristics: A comprehensive review of hybrid local search methods. *Expert Syst. Appl.* **2024**, *258*, 125192. [CrossRef]
21. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
22. 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; IEEE: New York, NY, USA, 2007; pp. 4661–4667.
23. Rao, R.V.; Savsani, V.J.; Vakharia, D. Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems. *Inf. Sci.* **2012**, *183*, 1–15. [CrossRef]
24. Tan, Y.; Zhu, Y. Fireworks algorithm for optimization. In Proceedings of the Advances in Swarm Intelligence: First International Conference, ICSI 2010, Beijing, China, 12–15 June 2010; Proceedings, Part I 1; Springer: Berlin/Heidelberg, Germany, 2010; pp. 355–364.
25. Gençal, M.C.; Oral, M. Bipolar Mating Tendency: Harmony Between the Best and the Worst Individuals. *Arab. J. Sci. Eng.* **2022**, *47*, 1849–1871. [CrossRef]
26. Gençal, M.C. Bipolar Parçacık Sürü Optimizasyonu Algoritması. *Çukurova Üniversitesi Mühendislik Fakültesi Dergisi* **2022**, *37*, 617–626. [CrossRef]
27. Gencal, M.; Oral, M. Roosters Algorithm: A Novel Nature-Inspired Optimization Algorithm. *Comput. Syst. Sci. Eng.* **2022**, *42*, 727–737. [CrossRef]
28. Rajwar, K.; Deep, K.; Das, S. An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges. *Artif. Intell. Rev.* **2023**, 56, 13187–13257 . [CrossRef]
29. Santiago-Moreno, J.; Esteso, M.; Castaño, C.; Toledano-Díaz, A.; López-Sebastián, A. Post-Coital Sperm Competence in Polygamous Animals: The Role of Sperm Traits in Species–Specific Strategies. *Androl. S1* **2015**, *3*, 2167–0250.
30. Frenkel, M.; Legchenkova, I.; Shvalb, N.; Shoval, S.; Bormashenko, E. Voronoi Diagrams Generated by the Archimedes Spiral: Fibonacci Numbers, Chirality and Aesthetic Appeal. *Symmetry* **2023**, *15*, 746. [CrossRef]
31. Kocyigit, E.; Korkmaz, M.; Sahingoz, O.K.; Diri, B. Enhanced feature selection using genetic algorithm for machine-learning-based phishing URL detection. *Appl. Sci.* **2024**, *14*, 6081. [CrossRef]
32. Jiacheng, L.; Lei, L. A hybrid genetic algorithm based on information entropy and game theory. *IEEE Access* **2020**, *8*, 36602–36611. [CrossRef]
33. Ji, S.; Karlovšek, J. Optimized differential evolution algorithm for solving DEM material calibration problem. *Eng. Comput.* **2022**, 39, 2001–2016 . [CrossRef]
34. Stanton, S.; Alberstein, R.; Frey, N.; Watkins, A.; Cho, K. Closed-Form Test Functions for Biophysical Sequence Optimization Algorithms. *arXiv* **2024**, arXiv:cs.LG/2407.00236.
35. Surjanovic, S.; Bingham, D. Virtual Library of Simulation Experiments: Test Functions and Datasets. 2024. Available online: http://www.sfu.ca/~ssurjano (accessed on 16 March 2025).
36. Liang, J.J.; Qu, B.; Suganthan, P.N.; Hernández-Díaz, A.G. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*; Technical Report; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2013; Volume 201212, pp. 281–295.
37. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Technical Report; National University of Defense Technology: Changsha, China; Kyungpook National University: Daegu, Republic of Korea; Nanyang Technological University: Singapore, 2017.
38. Yue, C.; Price, K.V.; Suganthan, P.N.; Liang, J.; Ali, M.Z.; Qu, B.; Awad, N.H.; Biswas, P.P. *Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization*; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2019; Volume 201911.
39. Liapis, C.M.; Karanikola, A.; Kotsiantis, S. A multi-method survey on the use of sentiment analysis in multivariate financial time series forecasting. *Entropy* **2021**, *23*, 1603. [CrossRef] [PubMed]
40. Taheri, S.; Hesamian, G. A generalization of the Wilcoxon signed-rank test and its applications. *Stat. Pap.* **2013**, *54*, 457–470. [CrossRef]
41. Ata, B.; Gencal, M.C. Comparison of optimization approaches on linear quadratic regulator design for trajectory tracking of a quadrotor. *Evol. Intell.* **2024**, *17*, 3225–3240. [CrossRef]
42. Mirjalili, S. Grey Wolf Optimizer (GWO) MATLAB Central File Exchange. 2022. Available online: https://www.mathworks.com/matlabcentral/fileexchange/44974-grey-wolf-optimizer-gwo (accessed on 16 March 2025).
43. Gencal, M.C. The Implementation of Differential Evolution in Matlab MATLAB Central File Exchange. 2022. Available online: https://www.mathworks.com/matlabcentral/fileexchange/77617-the-implementation-of-differential-evolution-in-matlab (accessed on 16 March 2025).

44. Yang, X.S. Cuckoo Search (CS) Algorithm MATLAB Central File Exchange. 2022. Available online: https://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm (accessed on 16 March 2025).
45. Pham, H. *Springer Handbook of Engineering Statistics*; Springer Nature: Berlin/Heidelberg, Germany, 2023.
46. Kumari, C.L.; Kamboj, V.K.; Bath, S.; Tripathi, S.L.; Khatri, M.; Sehgal, S. A boosted chimp optimizer for numerical and engineering design optimization challenges. *Eng. Comput.* **2023**, *39*, 2463–2514. [CrossRef]
47. Eker, E. Performance evaluation of capuchin search algorithm through non-linear problems, and optimization of gear train design problem. *Eur. J. Tech. (EJT)* **2023**, *13*, 142–149. [CrossRef]
48. Tzanetos, A.; Blondin, M. A qualitative systematic review of metaheuristics applied to tension/compression spring design problem: Current situation, recommendations, and research direction. *Eng. Appl. Artif. Intell.* **2023**, *118*, 105521. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.