International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Grey Wolf Cuckoo Search Algorithm for Training Feedforward Neural Network and Logic Gates Design

Layak Ali
Department of Electronics and Communication Engineering,
School of Engineering, Central University of Karnataka,
Aland Road Kadaganchi, Kalaburagi, 585367, Karnataka, India
e-mail: layakali@cuk.ac.in

Abstract— This paper presents a new hybrid Swarm Intelligence (SI) algorithm based on the Cuckoo Search Algorithm (CSA) and Grey Wolf Optimizer (GWO) called the Grey Wolf Cuckoo Search (GWCS) algorithm. The GWCS algorithm extracts and combines CSA and GWO features for efficient optimization. To carry out the comprehensive validation, the developed algorithm is applied to three different scenarios with their counterparts. The first validation is carried out on standard optimization benchmark problems. Further, they are used to train Feedforward Neural Networks and finally applied to design logic gates. The comprehensive results are presented and it is found that the proposed GWCS algorithms perform better compared to the state-of-the-art.

Keywords- Swarm Intelligence; Artificial Neural Network; Feedforward Neural Network; Logic Gates

I. INTRODUCTION

Swarm Intelligence (SI) algorithms are methods for solving complex optimization problems and are developed by taking inspiration from nature. The complexity of real-world engineering and science optimization problems is increasing day by day, and most of them have multiple peaks (multimodal). Since most of the classical optimization algorithms either gives non-feasible solution or fails on multimodal problems. Thus, global optimization has attracted researchers across the globe. Most of the global optimization domain is dominated by SI algorithms. Recently, many SI algorithms have been developed [6, 27, 26, 16, 11, 21, 28, 22, 9, 25] for global optimization. The SI algorithms are used in lot of fields [8, 20, 5, 12, 24] including Artificial Neural Network (ANN) training [7, 29, 2, 30]. A variant of PSO is used to train FNN for data classifications [23]. The supplier's selection is being addressed by PSO and Fuzzy in [13]. The arithmetic optimization algorithm (AOA) trains the FNN for medical image classification [4]. The detailed survey on ANN using PSO is found in [14].

The "Artificial Neural Network" (ANN) is computational models that mimic the human brain. One of the important classes of ANN is "Feedforward Neural Network" (FNN), where the information processing takes place only in a single direction. The important phase in either ANN or FNN is training. The training of FNN turns out to be hard as the weights of the hidden layers are highly complex, which turns out to be multimodal. Due to the complexity of training FNN, the classical optimization algorithms fail, hence SI algorithms become the alternative. This paper proposes hybrid SI algorithms based on "Grey Wolf Optimizer" and "Cuckoo Search" algorithms called Grey Wolf Cuckoo Search (GWCS). The proposed GWCS algorithm is systematically tested on standard benchmark problems for its efficiency. Further, the GWCS algorithm is used for training FNN and designing logic gates.

The work is presented as shown here; Section II describes Swarm Intelligence algorithms. Section III presents Neural Networks, Problem formulation, and logic gates design. Section IV presents the Grey Wolf Cuckoo Search (GWCS) algorithm; Section V gives detailed results and discussions. The concluding remarks are presented in Section VI.

II. SWARM INTELLIGENCE ALGORITHMS

The "Swarm Intelligence" (SI) algorithms are paradigms that are being developed by inspecting the searching behavior of creatures in nature. The SI algorithms have become very popular these days and are widely used in many applications including neural networks [24, 10, 6, 27, 26, 16]. The main strength of SI algorithms lies in solving very complicated optimization problems.

Following section describe about various SI algorithms "Artificial Bee Colony" (ABC) [10, 6], "Cuckoo Search Algorithm" (CSA) [27], "Firefly Algorithm" (FFA) [26], "Grey Wolf Optimizer" (GWO) [16], "Particle Swarm Optimization" (PSO) [11], "Bat Algorithm" (BAA) [28, 18], "Bald Eagle Search" (BES) [3], "Lightning Search Algorithm" (LSA) [22, 1], "Rat Swarm Optimizer" (RSO) [9] and "Sparrow Search Algorithm" (SSA) [25] used in the simulations.

A. Artificial Bee Colony: ABC

In 2005, Karaboga has developed an "Artificial Bee Colony" (ABC) algorithm [10][6]. It mimics the nectar searching behavior of honey bees. Honey bees are of three categories: onlookers, scouts, and employed bees. Half of the total bees are considered to be employed and the other half as onlookers. Each employed bee is deployed with every food source (nectar) and they have the knowledge about the nectar concentration. The employed bee will become the scout if no more food source is available. The information that the employed bees have with them, they exchange with other bees through waggle dance. The onlooker bees choose the best food source for their next foraging.

B. Bat Algorithm: BAA

The "Bat" (BAA) algorithm is developed by Xin-She Yang and Amir in 2012 [28, 18]. The main inspiration for this algorithm is the echo-sounding actions of bat species. The swarm of bats during their prey search, produces sound with varying frequency, loudness, and emission rate. When they find the prey, they change their strategy of frequency, loudness, and emission rate. This strategy helps the bats to find the best prey hence similar to finding the optimum solution [28, 18]. As an algorithm, the above said parameters become the tuning parameter, hence it does well even in dynamic problems.

C. Bald Eagle Search: BES

The "Bald Eagle Search" (BES) algorithm was developed in 2020 by Sattar and Bilal [3]. The BES algorithm extracts the fish hunting strategy of bald eagles. The hunting strategy is divided into three stages: selecting, searching, and swooping [3]. First, they select the space with densely populated prey, move inside the space and then enter into the swooping stage (best point) [3]. This is how BES will be able to solve optimization problems.

D. Cuckoo Search Algorithm: CSA

"Cuckoo Search Algorithm" (CSA) was developed in 2009 by Xin-she et al. [27]. It is based on pattern of laying eggs by cuckoo species into other bird's nest. The CSA follows rules like each cuckoo bird lays one egg at a time and chooses the nest randomly for laying. Good quality eggs are carried out to the next generation. Further, the host bird recognizes the laid eggs as a foreign egg with a certain probability, then either through away the egg or quits the existing nest and builds the new one.

E. Grey Wolf Optimizer: GWO

The "Grey Wolf Optimizer" (GWO) was introduced by Seyedali et. al. in 2014 [16], which mimics hunting behavior of Wolves. There are four categories of Wolves like alphas, beta, omega, and delta in their hierarchy. The alpha is at the highest, and strongest, and the omega is weak and lowest in the hierarchy. Wolves follow the main steps like tracking, chasing, approaching, encircling, and attacking the prey [16].

F. Lightning Search Algorithm: LSA

The "Lightning Search Algorithm" (LSA) was developed by Mohamed et. al in 2021 [22, 1]. The inspiration for LSA algorithms comes from natural lighting phenomena and propagation. The LSA divides the searching strategy into three stages, one is exponential space projectile, the opposition theory, and local search by the chief projectile. These strategies of LSA be used for solving optimization problems [22, 1].

G. Particle Swarm Optimization: PSO

"Particle Swarm Optimization" (PSO) was developed in 1995 by Kennedy and Eberhart [11]. It is developed based on food searching and social life of birds. During their food search, all the birds exchange their information and follow the best profitable decision. While searching for the food, every particle remembers the best information in their journey and exchanges it, thus they reach the goal [11] and solves optimization problems.

III. Neural Networks and Problem formulation

This section describes the basic foundation of "Artificial Neural Network" (ANN), "Feedforward Neural Network" (FNN), and the problem formulation.

A. Neural Network foundation

The "Artificial Neural Networks" (ANNs) are the computational strategies developed from the working of the human brain. They are basically, interconnected nodes to do simple and specific operations. The output of a node is determined by the mathematical operation that they perform. The interconnection of these simple nodes by appropriate parameter setting would learn even highly complex functions. The ANNs have made technological advancements in robotics, voice / image recognition, communication, cloud, and many more.

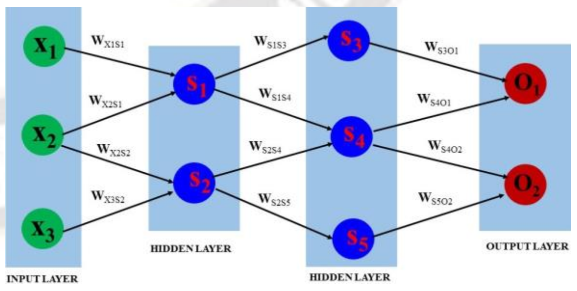

Figure 1. Feedforward Neural Networks Architecture

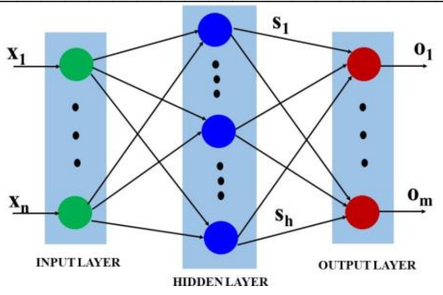

Figure 2. Two Layered structure of Feedforward Neural Networks

The "Feedforward Neural Networks" (FNNs) are the important subset of ANNs. The difference is, in FNNs, the interconnection of nodes does not make a closed circle and is very simple compared to their counterparts. The FNNs are called feedforward, due to the unidirectional information flow as shown in Figure 1 The FNNs are usually used for supervised learning. It does not matter whether data is sequential or time-dependent. Thus in simpler terms, they compute a function *"f"* on fixed size input *x* such that $f(x) \approx y$ for training pairs (x, y).

The simple architecture of ANN comprises a bunch of neurons as input, output, and hidden layers. These are connected to each other by a set of synaptic weights. When ANNs go for the learning process, the weights will be continuously changed until they learn sufficiently. Sufficient learning may be determined by either maximum iterations or error value threshold. Once the learning process is over, the model may be used for valuation over various problems, and thus they can classify with acceptable accuracy.

The training of ANNs or FNNs itself has become an issue recently, mostly due to the complexity of the underlying problem. Additionally, there are several training methods being developed. Since the problems encountered nowadays turn out to be complex and multimodal, hence most of the classical methods fail to achieve the desired solution. Thus some alternate strategies such as Bio-inspired or nature-inspired algorithms are used for ANNs training.

The Bio-inspired or Nature-inspired algorithms are very powerful for solving optimization problems, especially when multimodal and non-continuous. The basic concept of these algorithms is defined in [7]. The work in [7, 29] explains these algorithms as a system with simple and unintelligent agents that have limited capabilities when working together in the social form to develop a tremendous power. Bio-inspired or Nature-inspired algorithms are widely used in training ANNs [29, 2]. The detailed work, on how the Bio-inspired or Nature-inspired algorithms are used for training can be found in [29, 30].

B. *FNN training problem formulation*

The training of ANNs or FNNs turns out to be complex due to the reason that, the interconnected nodes with their weights are highly interdependent and non-separable. Due to these, it becomes difficult to attain the goal by optimizing a single weight or node. These problems may be reduced by assigning random weights and evaluating repeatedly. This strategy may be good with fewer weights and nodes; however, if weights become large in number, the problem will be complex and may diverge the optimizing algorithms. Take a simple example of ANNs with a number of neurons in various layers, like 500 in the input layer, 100 in the hidden layer, and 50 in the output layer. This turns out to be 500x100x50 = 2500000 weights. Further, if the biases are added then it becomes more complex and higher dimensional.

The cost function used for evaluation in this paper is considered as in [31, 15]. Figure 2 shows the FNN with a single input, hidden, and output layer. The number of nodes in each layer is represented as *n*, *h*, and *m* for input, hidden, and output layers respectively. The output of hidden nodes is calculated as follows in the equation.

$$f(s_j) = \frac{1}{1+e^{-\sum_{i=1}^{n} w_{ij} * x_i - \alpha_j}}, \text{ Where } j = 1,2,\ldots h \qquad (1)$$

Here w_{ij} is the weight from ith node to jth hidden node and α_j is the bias of the jth node, s_j and x_i are the number of hidden and input nodes respectively.

The final output can be calculated as in equation (2)

$$O_k = \sum_{j=1}^{h} w_{kj} * f(s_j) - \alpha_k, \text{ Where } k = 1,2,\ldots,m \qquad (2)$$

Where w_{kj} is the weight from jth hidden node to kth output node, the α_k is the bias of the kth output node, the O_k and s_j are the number of output and hidden nodes respectively.

Finally, the error function E (learning function) is calculated as in equation (3)

$$E_k = \sum_{i=1}^{k} (O_i^k - D_i^k)^2 \qquad (3)$$

$$E_k = \sum_{k=1}^{q} \frac{E_k}{q} \qquad (4)$$

Where q is the count of training samples, D_i^k is the desired output of the ith unit when the kth training sample is used. The equation (4) is used as a fitness function for optimizing algorithms.

Thus the fitness function for ith training sample can be defined as follows:

$$Fitness(X_i) = E(X_i) \qquad (5)$$

Further, the encoding strategy used in this article is the matrix encoding strategy as in [15].

C. *Logic gates design*

As is already explained in the previous sections; a neural network receives data through its input layer, and then transmits it to its hidden layers. Through a network of weighted

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

connections, processing occurs in the hidden layers. The data from the input layer is then combined with a set of coefficients by nodes in the hidden layer, and the inputs are then given the proper weights. The sum of these input and weight products is transmitted through a node's activation function, which chooses how far a signal must go through the network before it has an impact on the output. The output layer is where the outputs are retrieved, and the hidden layers link to it. The basic architecture of the FNN is shown in Figure 3. In order to get the desired input, this FNN (Figure 3) is trained using input from logic gates. According to the Perceptron algorithm, if $Wx+b > 0$ then prediction $(y') = 1$; otherwise, it is 0. Similarly, if $Wx+b <= 0$, then, $y' = 0$

Thus, the SI algorithms initialize the weights and biases, forward propagate and check the error with the output obtained and adjust weights and bias till the error is under an acceptable range. This is repeated for all training examples. The dataset used for FNN training is taken from Kaggle [19].

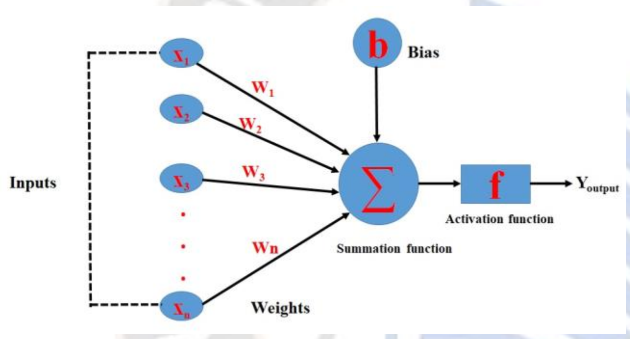

Figure 3. Simplified FNN training model

IV. GREY WOLF CUCKOO SEARCH ALGORITHM: GWCS

This article presents a Hybrid algorithm, which harnesses the searching capability of "Grey Wolf Optimizer" (GWO) and "Cuckoo Search" (CS) algorithms. The following section describes GWO and CS algorithms used for obtaining a Hybrid algorithm called Grey Wolf Cuckoo Search (GWCS). In GWCS, the updating equations of both GWO and CS algorithms are systematically combined and implemented.

According to GWO, wolves follow a very strict hierarchy and divide the population as; alphas, beta, omega, and delta. The power and guidelines that this division follows are explained in Section II and [16]. The goal searching behavior of the GWO algorithm can be modeled as follows.

The position of the whole group (Wolves) can be updated using the following equation and as in [16]

$$\vec{K} = \left|\vec{C} * \vec{X}_p(t) - \vec{X}(t)\right| \qquad (6)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} * \vec{K} \qquad (7)$$

the t represents generation count, $\vec{A}$, and $\vec{C}$ are the coefficients, $\vec{X}_p$ is target location, and $\vec{X}$ is grey wolf location [16]. The coefficients $\vec{A}$, and $\vec{C}$ can found using below equations:

$$\vec{A} = 2\vec{a} * \vec{r}_1 - \vec{a} \qquad (8)$$

$$\vec{C} = 2\vec{r}_2 \qquad (9)$$

here $\vec{a}$ is decreased from 2 to 0 linearly over a given generation and r_1, r_2 are pseudorandom numbers in the range [0, 1] as used in [16].

All the wolves approach the target prey and attack. Their progressive movements are governed by alpha, beta, and delta positions, given by the below equations.

$$\vec{K}_\alpha = \left|\vec{C}_1 * \vec{X}_\beta - \vec{X}\right|,\ \vec{K}_\beta = \left|\vec{C}_2 * \vec{X}_\beta - \vec{X}\right|,\ \vec{K}_\delta = \left|\vec{C}_3 * \vec{X}_\delta - \vec{X}\right|$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 * \vec{K}_\alpha,\quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 * \vec{K}_\beta,\quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 * \vec{K}_\delta \qquad (10)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \qquad (11)$$

Further, the "Cuckoo Search" (CS) algorithms are mainly based on the breeding and egg laying strategy of the Cuckoo bird. These birds give their eggs into different species' nests. The laid Cuckoo eggs will become young if they are not recognized by host birds and they will not be removed. This strategy shows that the Cuckoo bird has found the optimal nest or place for reproduction as seen in Section II and [27]. The CS algorithm follows very simple rules

1. Every Cuckoo bird gives only one egg.
2. The Cuckoo eggs are placed randomly into the given nest.
3. The Good eggs will be retained and taken to further generation.
4. Total foreign nests for laying eggs are constant.
5. The host birds will find the foreign eggs with a certain probability.
6. With this probability host bird can reject an egg or it will leave its own nest and build a new one somewhere else.

The simple and very dominant strategy of the CS algorithm is in generating new solutions using the "Levy" flights method. The "Levy" flight is basically a random strategy that follows the well-calculated step length by a good probability distribution. The following equation shows how the CS algorithm generates new solutions.

$$X(t+1) = X(t) + \varphi Levy(\lambda) \qquad (12)$$

Here, $\varphi > 0$ is the step length, however, it relates to problem dimensions. Generally, its value $\varphi = 1$ considered, similarly the value of λ is considered to be $1 < \lambda < 3$

The hybrid GWCS algorithm use equation 11 and 12 for generating new solutions

The proposed Hybrid GWCS algorithm is implemented as shown in Algorithm 1

```
Algorithm 1: Implementation of GWCS algorithm
1.  Decide the Generation parameter (MaxIter)
2.  Decide the Population size (N)
3.  Randomly initialize the coordinates $X_1$ to $X_N$
4.  Set the parameters a, A, $\alpha$, $\lambda$ , Pa and C
5.  Calculate f (X ) (Fitness of Population)
6.  Find $X_\alpha$ (first best candidate)
7.  Find $X_\beta$ (second best candidate))
8.  Find $X_\delta$ (third best candidate)
9.  While t $\le$ MaxItr do
10.     While i $\le$ N do
11.         Update X as per Equation ( 11)
12.         Update X as per Equation ( 12)
13.     End While
14.     i$\leftarrow$i+1  go to step 10
15.     Update a, A, C, $\alpha$, $\lambda$ and Pa
16.     Update $X_\alpha$ , $X_\beta$ and $X_\delta$
17.     Calculate f (X ) (Fitness of Population)
18.     Check for optimal result
19. End While
20. Report Results
```

V. RESULTS AND DISCUSSIONS

This section presents the details of the simulation setup, results, and discussions.

A. Simulation setup

The simulations are conducted using “MATLAB 2022a” software on the “Windows 11” platform over an “Intel Core I5” processor with 8GB of RAM. The standard benchmark problems are taken from CEC2005 [17]. The dataset used for FNN training is taken from [19]. All the SI algorithms are executed over 1000 iterations with a population size of 30. The other parameters of SI algorithms are set as per their original settings. Since all the SI algorithms are “stochastic” and hence give lots of variation in the results every time they are executed. Thus, the average of 10 trials is considered to record the results

B. Results

The simulations were first carried out to validate SI algorithms on standard benchmark problems [17] and then tested for training FNN [31] [15] over data from [19]. Further, the FNN is used for digital logic gates.

1. Validation on benchmark problems

To validate the given SI algorithms, six benchmark problems viz. *Ackley, Griewank, Powell, Rastrigin, Rosenbrock* and *Zakharov* are taken from CEC2005 [17]. Since the selected problems are scalable in dimension, hence higher dimensions like 50, 100, and 200 are set for all the problems. The algorithms are executed 10 times on each problem and results are documented. The documented results are presented in both numerical and graphical form. The numerical results are shown in Table I, Table II, and Table III as mean results. The mean results convey the quality of SI algorithms on a given problem. Similarly, the graphical results are shown from Figure 4 to Figure 9. The graphical results show the manner in which the SI algorithms approach the optimum results.

TABLE I. AVERAGE VALUES OBTAINED WITH 50D PROBLEMS

Problem	ABC	BAA	BES	CSA	GWO	LSA	PSO	GWCS
Ackley	1.26e+1	1.99e+1	2.16e+1	3.17	8.64	1.13e+1	2.70	**3.46e-14**
Griewank	9.79e-1	2.92	1.44e+1	3.17e-2	1.05	1.17	1.66e-1	**0.0**
Powell	1.67e+5	2.64e+7	1.54e+7	3.81e+2	9.94e+5	3.24e+5	1.25e+2	**1.12e-5**
Rastrigin	5.06e+2	7.97e+3	5.41e+4	3.87e+2	7.56e+2	1.12e+3	3.71e+2	**1.14e-14**
Rosenbrock	1.05e+5	2.89e+8	5.31e+9	8.59e+2	3.50e+7	2.21e+6	4.85e+2	**4.71e+1**
Zakharov	1.13e+4	8.11e+4	1.90e+9	9.62e+3	1.94e+4	1.14e+4	1.54e+2	**6.21e-3**

TABLE II. AVERAGE VALUES OBTAINED WITH 100D PROBLEMS

Problem	ABC	BAA	BES	CSA	GWO	LSA	PSO	GWCS
Ackley	1.68e+1	2.03e+1	2.16e+1	7.78	2.02e+1	1.24e+1	2.80	1.34e-13
Griewank	1.45	5.39	2.78e+1	4.95e-1	3.82	1.36	1.86e-1	0.0
Powell	9.26e+5	1.06e+8	3.21e+7	9.31e+3	6.91e+6	6.76e+5	2.48e+2	7.78e-6
Rastrigin	2.29e+3	1.80e+4	1.08e+5	1.40e+3	1.24e+4	2.17e+3	7.79e+2	3.64e-13
Rosenbrock	1.53e+7	1.03e+9	1.07e+10	1.34e+5	1.08e+9	5.97e+6	1.19e+3	9.82e+1
Zakharov	2.70e+4	7.50e+6	4.69e+19	2.48e+4	4.67e+4	2.18e+4	1.11e+3	6.56e+2

TABLE III. AVERAGE VALUES OBTAINED WITH 200D PROBLEMS

Problem	ABC	BAA	BES	CSA	GWO	LSA	PSO	GWCS
Ackley	1.87e+1	2.02e+1	2.16e+1	1.18e+1	2.10e+1	1.27e+1	2.88	2.91e-11
Griewank	3.45	1.11e+1	5.47e+1	1.15	1.35e+1	1.73	2.42e-1	1.11e-16
Powell	1.75e+7	2.44e+8	6.41e+7	8.73e+4	4.61e+7	2.18e+6	6.96e+2	1.24e-4
Rastrigin	1.14e+4	3.92e+4	2.17e+5	4.17e+3	5.22e+4	4.73e+3	1.60e+3	5.71
Rosenbrock	2.01e+8	1.56e+9	2.16e+10	2.55e+6	3.38e+9	1.04e+7	2.33e+3	1.98e+2
Zakharov	6.00e+4	1.59e+10	1.18e+22	5.10e+4	1.10e+5	3.91e+4	1.57e+5	8.40e+3

The figures are self-explanatory, that show how SI algorithms along with the proposed algorithm achieve the optimal results over 1000 iterations. The best results are shown in bold in Table I, Table II and Table III, it is seen that the proposed GWCS algorithm performs well on almost all the benchmark problems. The convergence graphs are shown from Figure 4 to Figure 9, for Ackley to Zakharov problems respectively. Here also the proposed GWCS algorithm converges fast to optimum results compared to other algorithms on all the problems.

2. *Validation on FNN training*

The FNN training is carried out using dataset from [19]. Here, the number of hidden nodes for FNN is varied from 5 to 30 in the step of 5 and a higher number of hidden layers like 50 and 100 are used. The SI algorithms are applied for training FNN for 10 trails and the results are recorded. The recorded results are presented in two forms: Numeral and Graphical.

The graphical results show the behavior of SI algorithms in attaining the goal. The figures from Figure 10 to Figure 17 show the convergence graphs. From these figures, it is seen that the proposed algorithm GWCS performs well and converges to the optimum results very quickly. These graphs show that as the number of hidden layers increases, the SI algorithms take more iterations and time for convergence. This is due to the fact that the complexity is proportional to the number of hidden layers.

The tables from Table IV to Table XI show the numerical results, which depict the quality of results obtained by SI algorithms on FNN training. The results presented in tables are the average of 10 trials, best and worst results over 1000 iterations. The robustness of SI algorithms is shown by the standard deviation and the exact execution time (in seconds) of SI algorithms is also recorded. From tables Table IV to Table XI, it is seen the GWCS and BES algorithms obtain good results including average, best, and worst results. The same tables show the robustness of algorithms. It can be further noted that the applications where the quality of results matters, the GWCS and BES algorithms may be used.

3. *Validation on Logic gates*

Simulations were carried out to validate SI algorithms on training FNN. The number of neurons for the FNN model is considered to be 10. All the mentioned SI algorithms are initialized with a population of 30 and other parameters of algorithms are set as per the individual algorithmic specification. The dataset used for FNN training is from Kaggle [19]. The SI algorithms follow the Perceptron rule and minimize the error between the expected output and obtained output.

The results are documented in numerical and graphical forms. The convergence characteristics of SI algorithms on optimizing weights and biases are shown in figures from Figure 18 to Figure 22. Figure 18 shows the convergence of all the algorithms over AND gate. It shows that the GWCS algorithm outperforms the other algorithms. A similar trend of the GWCS algorithm is seen on all the other gates. The Figure 19 shows a convergence graph on OR gate, Figure 21 show for NAND gate, Figure 22 shows for NOR gate, and Figure 20 show for XOR gate.

Table XII show the minimum error values obtained after optimizing the cost function. From the table it can be observed

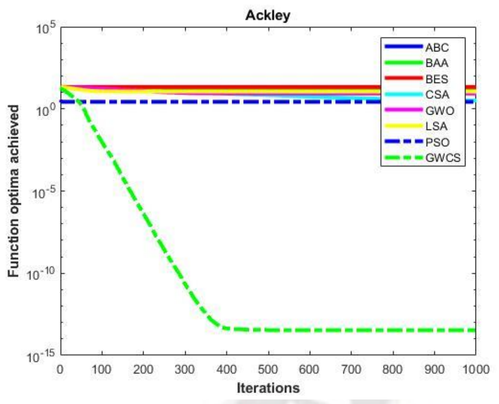

Figure 4. Convergence graph on Ackley with 50D

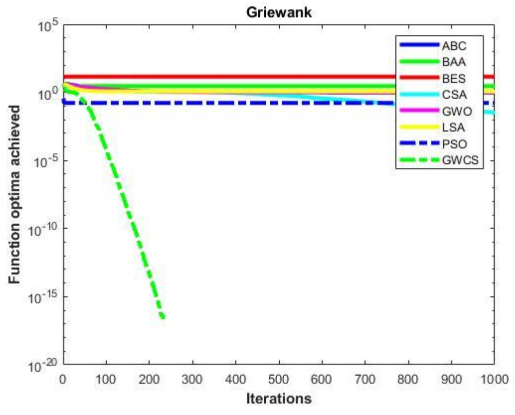

Figure 5. Convergence graph on Griewank with 50D

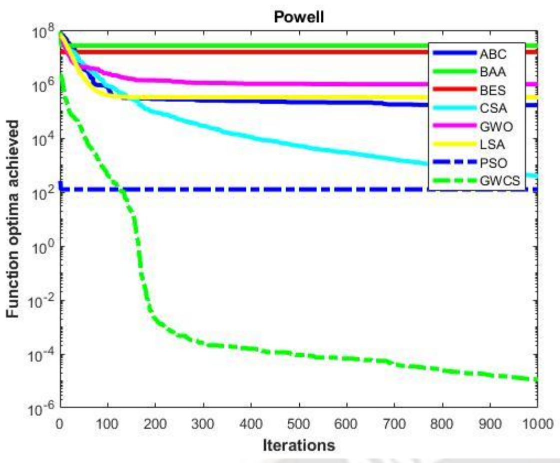

Figure 6. Convergence graph on Powell with 50D

that all the SI algorithms perform better, further the GWCS algorithm outperforms all the SI algorithms.

The FNN model is trained for two input logic gates like AND, OR, NAND, NOR, and XOR using SI algorithms. The well-trained FNN can be called the FNN logic unit, like the FNN logic unit of AND gate gives similar functionality compared to the original digital AND gate and so on.

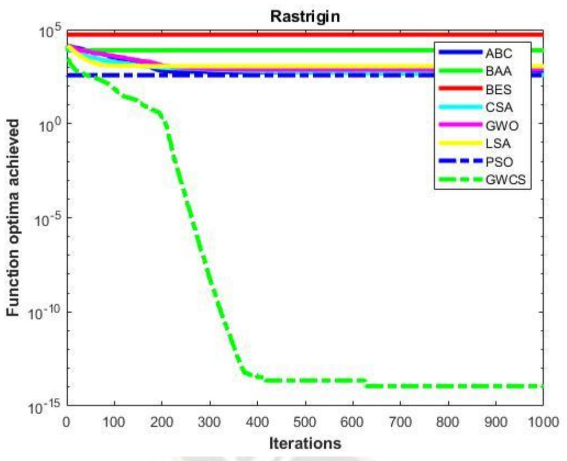

Figure 7. Convergence graph on Rastrigin with 50D

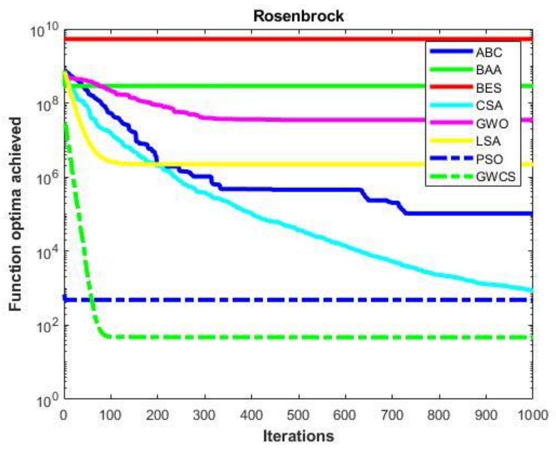

Figure 8. Convergence graph on Rosenbrock with 50D

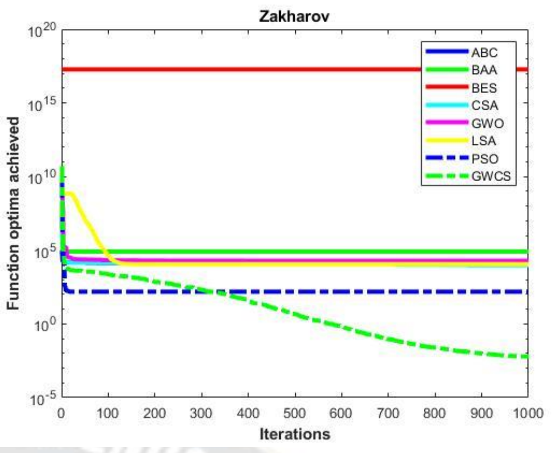

Figure 9. Convergence graph on Zakharov with 50D

These units can be used for independent analysis or can be part of the larger digital logic circuit.

The designed FNN logic unit is used for validation, however, in this paper, only the FNN logic unit trained by GWCS is demonstrated by giving random inputs as shown in Table XIII to Table XVII. These tables show that the logic gates designed by FNN are exactly the same as the original logic gates.

TABLE IV. RESULTS WITH FNN HIDDEN NODES 5

Alg	mean	std	best	worst
ABC	0.553	0.089	0.474	0.681
BAA	0.593	0.154	0.382	0.776
BES	0.025	0.005	0.019	0.031
CSA	0.027	0.003	0.023	0.030
GWO	0.023	0.002	0.021	0.026
LSA	0.313	0.152	0.046	0.418
PSO	0.258	0.064	0.174	0.332
GWCS	**0.004**	0.005	0.001	0.013

TABLE V. RESULTS WITH FNN HIDDEN NODES 10

Alg	mean	std	best	worst
ABC	0.564	0.067	0.445	0.604
BAA	0.566	0.112	0.416	0.684
BES	0.024	0.006	0.018	0.035
CSA	0.036	0.008	0.025	0.047
GWO	0.023	0.003	0.017	0.025
LSA	0.435	0.185	0.253	0.684
PSO	0.254	0.109	0.112	0.382
GWCS	**0.002**	0.003	0.000	0.007

TABLE VI. RESULTS WITH FNN HIDDEN NODES 15

Alg	mean	std	best	worst
ABC	0.475	0.155	0.240	0.666
BAA	0.556	0.084	0.449	0.628
BES	0.021	0.003	0.016	0.025
CSA	0.032	0.004	0.027	0.036
GWO	0.022	0.003	0.018	0.027
LSA	0.345	0.113	0.201	0.447
PSO	0.294	0.075	0.212	0.414
GWCS	**0.001**	0.003	0.000	0.007

***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

TABLE VII. RESULTS WITH FNN HIDDEN NODES 20

Alg	mean	std	best	worst
ABC	0.532	0.050	0.476	0.602
BAA	0.590	0.069	0.531	0.695
BES	0.022	0.003	0.018	0.024
CSA	0.040	0.008	0.027	0.047
GWO	0.021	0.004	0.019	0.029
LSA	0.234	0.088	0.133	0.341
PSO	0.326	0.056	0.270	0.413
GWCS	**0.001**	0.003	0.000	0.007

TABLE VIII. RESULTS WITH FNN HIDDEN NODES 25

Alg	mean	std	best	worst
ABC	0.582	0.074	0.483	0.679
BAA	0.562	0.038	0.521	0.622
BES	0.020	0.004	0.015	0.024
CSA	0.047	0.010	0.037	0.060
GWO	0.016	0.002	0.013	0.019
LSA	0.279	0.144	0.148	0.507
PSO	0.350	0.069	0.234	0.419
GWCS	**0.002**	0.003	0.000	0.007

TABLE IX. RESULTS WITH FNN HIDDEN NODES 30

Alg	mean	std	best	worst
ABC	0.545	0.029	0.509	0.589
BAA	0.610	0.102	0.446	0.692
BES	0.020	0.006	0.014	0.029
CSA	0.049	0.013	0.033	0.067
GWO	0.017	0.004	0.014	0.023
LSA	0.309	0.103	0.178	0.411
PSO	0.306	0.039	0.261	0.368
GWCS	**0.006**	0.005	0.000	0.013

TABLE X. RESULTS WITH HIDDEN FNN NODES 50

Alg	mean	std	best	worst
ABC	0.583	0.049	0.518	0.646
BAA	0.613	0.067	0.545	0.707
BES	0.018	0.002	0.015	0.020
CSA	0.056	0.018	0.036	0.073
GWO	0.015	0.002	0.014	0.017
LSA	0.329	0.083	0.183	0.394
PSO	0.348	0.046	0.280	0.409
GWCS	**0.001**	0.003	0.000	0.007

TABLE XI. RESULTS WITH FNN HIDDEN NODES 100

Alg	mean	std	best	worst
ABC	0.616	0.040	0.548	0.649
BAA	0.611	0.143	0.453	0.798
BES	0.084	0.149	0.010	0.351
CSA	0.048	0.013	0.031	0.066
GWO	0.015	0.003	0.012	0.020
LSA	0.314	0.153	0.151	0.495
PSO	0.371	0.066	0.267	0.433
GWCS	**0.005**	0.006	0.000	0.013

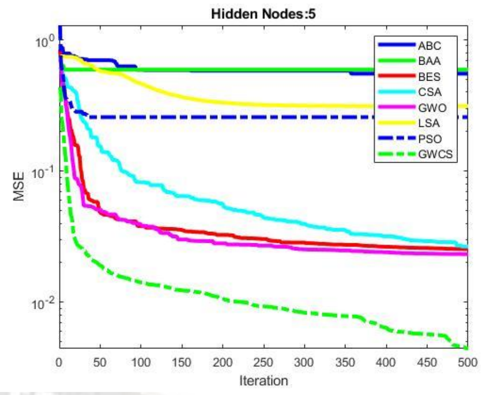

Figure 10 Convergence on FNN with hidden nodes 5

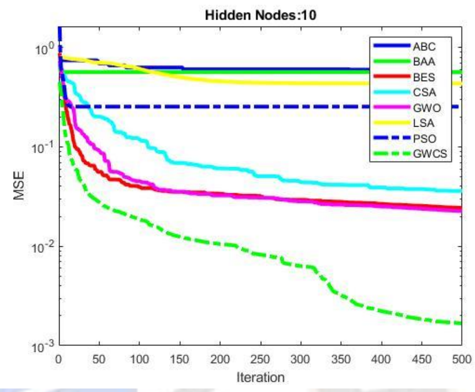

Figure 11 Convergence on FNN with hidden nodes 10

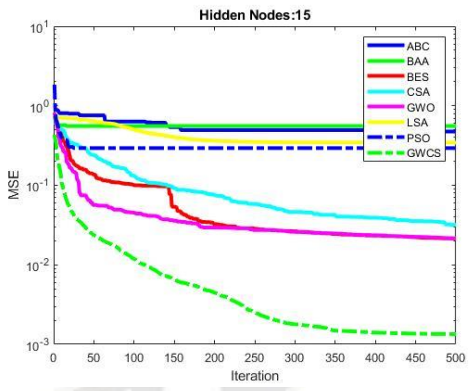

Figure 12 Convergence on FNN with hidden nodes 15

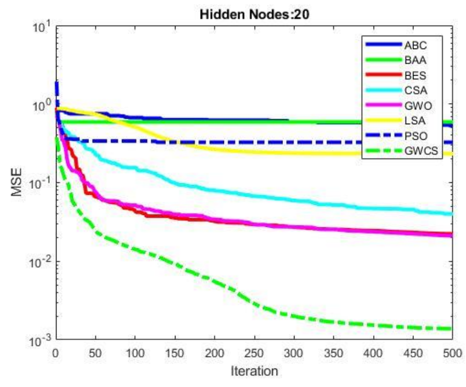

Figure 13 Convergence on FNN with hidden nodes 20

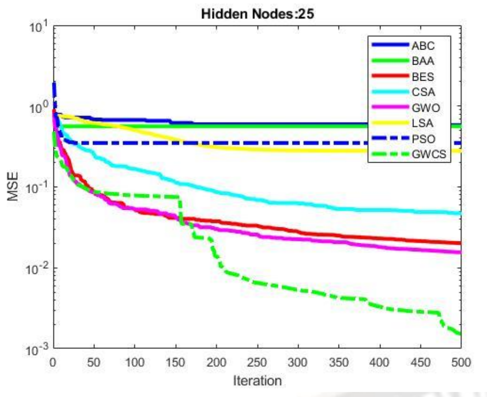

Figure 14 Convergence on FNN with hidden nodes 25

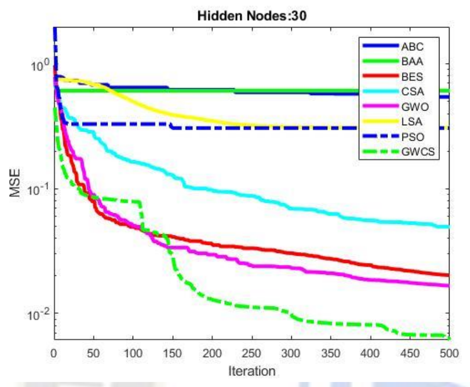

Figure 15 Convergence on FNN with hidden nodes 30

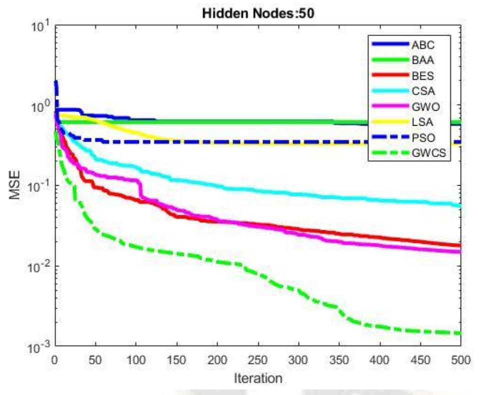

Figure16 Convergence on FNN with hidden nodes 50

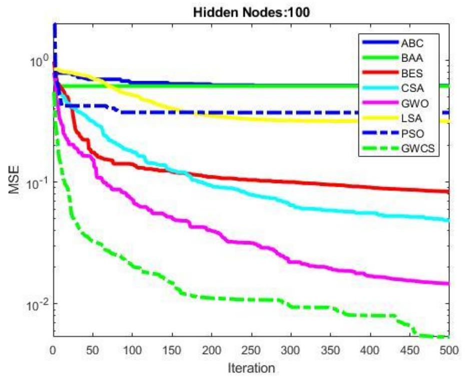

Figure 17 Convergence on FNN with hidden nodes 100

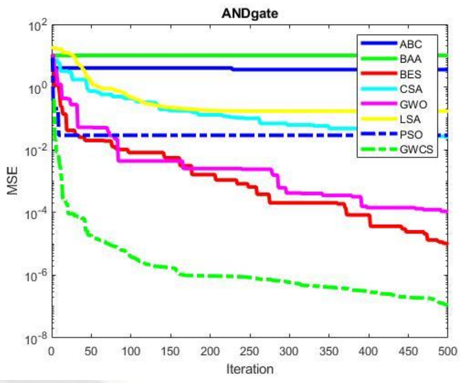

Figure 18 Convergence graph on AND Gate

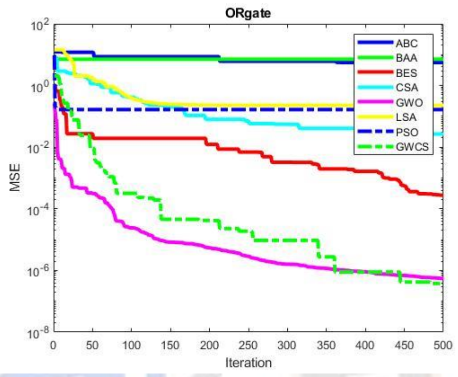

Figure 19 Convergence graph on OR Gate

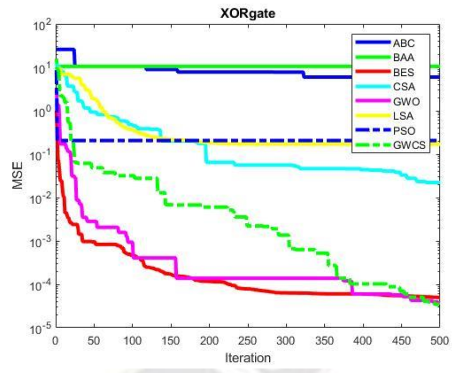

Figure 20 Convergence graph on XOR Gate

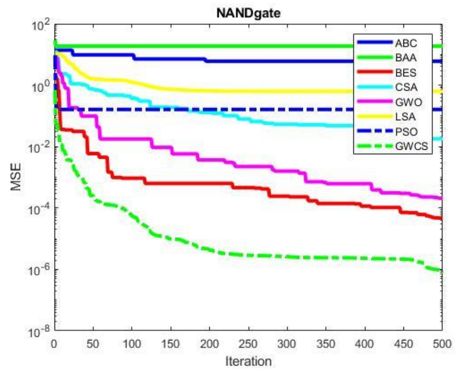

Figure 21 Convergence graph on NAND Gate

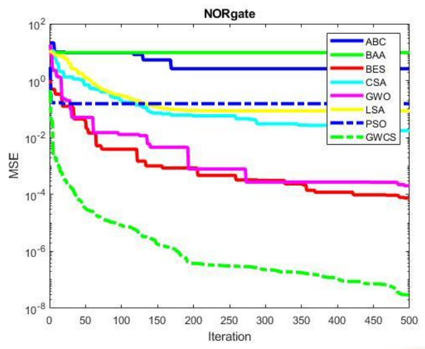

Figure 22 Convergence graph on NOR Gate

TABLE XII. ERROR VALUES ACHIEVED BY SI ALGORITHMS

	AND	OR	XOR	NAND	NOR
ABC	3.6081	5.5651	5.9538	6.0229	2.6209
BAA	240.40	7089.9	321.631	163.02	150.824
BES	9.30e-6	2.70e-4	4.97e-5	4.71e-5	7.39e-5
CSA	0.0258	0.0268	0.02219	0.01791	0.01763
GWO	1.08e-4	5.50e-7	3.71e-5	2.06e-4	1.97e-4
LSA	0.17094	0.2279	0.16963	0.63056	0.08663
PSO	0.0288	0.1663	0.20555	0.16215	0.15451
GWCS	**1.1e-7**	**3.7e-7**	**3.4e-5**	**7.6e-7**	**2.9e-8**

TABLE XIII. AND GATE FNN LOGIC UNIT VALIDATION BY GWCS

X_1	X_2	Y
0.318897034	0.350152533	0
0.037386179	0.642586447	0
0.836101491	0.1899943	0
0.673633733	0.971855481	1

TABLE XIV. OR GATE FNN LOGIC UNIT VALIDATION BY GWCS

X_1	X_2	Y
0.318897034	0.350152533	0
0.037386179	0.642586447	1
0.836101491	0.1899943	1
0.673633733	0.971855481	1

TABLE XV. NAND GATE FNN LOGIC UNIT VALIDATION BY GWCS

X_1	X_2	Y
0.318897034	0.350152533	1
0.037386179	0.642586447	1
0.836101491	0.1899943	1
0.673633733	0.971855481	0

TABLE XVI. NOR GATE FNN LOGIC UNIT VALIDATION BY GWCS

X_1	X_2	Y
0.318897034	0.350152533	1
0.037386179	0.642586447	0
0.836101491	0.1899943	0
0.673633733	0.971855481	0

TABLE XVII. XOR GATE FNN LOGIC UNIT VALIDATION BY GWCS

X_1	X_2	Y
0.318897034	0.350152533	0
0.037386179	0.642586447	1
0.836101491	0.1899943	1
0.673633733	0.971855481	0

VI. CONCLUSIONS

Swarm Intelligence (SI) algorithms have proved to be the alternative for classical optimization algorithm. However, still there is a requirement of efficient SI algorithms for complex optimization problems. This paper presents the development of new hybrid Swarm Intelligence (SI) algorithm based on the Cuckoo Search Algorithm (CSA) and Grey Wolf Optimizer (GWO) called the Grey Wolf Cuckoo Search (GWCS) algorithm. This combines CSA and GWO features for efficient optimization. The developed algorithm is well tested on optimization benchmark problems with higher dimensions for its efficiency. Further this algorithm is used to train “Feedforward Neural Network” (FNN) with higher number of hidden layers and design of logic gates. The results confirm that the developed algorithm is powerful in handling higher dimensional problems, training complex FNN and logic gates design.

REFERENCES

[1] Laith Abualigah, Mohamed Elsayed Abd Elaziz, Abdelazim Hussien, Bisan Alsalibi, Seyed Mohammad Jalali, and Amir Gandomi. Lightning search algorithm: a comprehensive survey. Applied Intelligence, 51, 04 2021. doi: 10.1007/s10489-020-01947-2.

[2] E. Alba and R. Marti. Metaheuristic Procedures for Training Neural Networks. Operations Research/Computer Science Interfaces Series, Springer, New York, NY, USA, 2006.

[3] H. Alsattar, A. Zaidan, and Bilal Bahaa. Novel meta-heuristic bald eagle search optimisation algorithm. Artificial Intelligence Review, 53, 03 2020. doi: 10.1007/s10462-019-09732-5.

[4] Koon Meng Ang, Wei Hong Lim, Sew Sun Tiang, Hameedur Rahman, Chun Kit Ang, Elango Natarajan, Mohamed Khan Afthab Ahamed Khan, and Li Pan. Training feedforward neural networks using arithmetic optimization algorithm for medical classification. In Muhammad Amirul Abdullah, Ismail Mohd. Khairuddin, Ahmad Fakhri Ab. Nasir, Wan Hasbullah Mohd. Isa, Mohd. Azraai Mohd. Razman, Mohd. Azri Hizami Rasid, Sheikh Muhammad Hafiz Fahami Zainal, Barry Bentley, and Pengcheng Liu, editors, Advances in Intelligent Manufacturing and Mechatronics, pages 313– 323, Singapore, 2023. Springer Nature Singapore.

[5] Azrina Abd Aziz, Y. Ahmet Sekercioglu, Paul Fitzpatrick, and Milosh Ivanovich. A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks. IEEE Communications Surveys Tutorials, 15(1):121–144, 2013. doi: 10.1109/SURV.2012. 031612.00124.

[6] B. Basturk and Dervis Karaboga. An artificial bee colony (abc) algorithm for numeric function optimization. in proceedings of the ieee swarm intelligence symposium, indianapolis, in, usa. May, 2006:12–14, 01 2006.

[7] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer, editors, Robots and Biological Systems: Towards a New Bionics?, pages 703–

712, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. ISBN 978-3-642-58069-7.

[8] Gabriel Lobaˇo da Silva Fre´, Jonathan de Carvalho Silva, Felipe Andery Reis, and Lucas Dias Palhaˇo Mendes. Particle swarm optimization implementation for minimal transmission power providing a fully connected cluster for the internet of things. In 2015 International Workshop on Telecommunications (IWT), pages 1–7, 2015. doi: 10.1109/IWT.2015.7224573.

[9] Gaurav Dhiman, Meenakshi Garg, Atulya Nagar, and Vijay Chahar. A novel algorithm for global optimization: Rat swarm optimizer. Journal of Ambient Intelligence and Humanized Computing, 12, 08 2021. doi: 10.1007/s12652-020-02580-0.

[10] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization, technical report - tr06. Technical Report, Erciyes University, 01 2005.

[11] J. Kennedy and R. Eberhart. Particle swarm optimization. In Proceedings of ICNN'95 - International Conference on Neural Networks, volume 4, pages 1942–1948 vol.4, 1995. doi: 10.1109/ICNN.1995.488968.

[12] Raghavendra V. Kulkarni and Ganesh Kumar Venayagamoorthy. Particle swarm optimization in wireless-sensor networks: A brief survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41(2):262–267, 2011. doi: 10.1109/TSMCC.2010.2054080.

[13] R.J. Kuo, S.Y. Hong, and Y.C. Huang. Integration of particle swarm optimization-based fuzzy neural network and artificial neural network for supplier selection. Applied Mathematical Modelling, 34(12):3976–3990, 2010. ISSN 0307-904X. doi: https://doi.org/10.1016/j.apm.2010.03.033. URL https://www.sciencedirect.com/science/article/pii/ S0307904X10001526.

[14] Pooria Mazaheri, Shahryar Rahnamayan, and Azam Asilian Bidgoli. Designing artificial neural network using particle swarm optimization: A survey. In Marco Antonio Aceves-Ferna´ndez, editor, Swarm Intelligence, chapter 3. IntechOpen, Rijeka, 2022.

[15] Seyed Mohammad Mirjalili, S.Z.M. Hashim, and Hossein Moradian Sardroudi. Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl. Math. Comput., 218:11125–11137, 2012.

[16] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. Advances in Engineering Software, 69:46–61, 2014. ISSN 0965-9978. doi: https://doi.org/10.1016/j.advengsoft.2013.12.007. URL https://www.sciencedirect.com/science/article/pii/ S0965997813001853.

[17] Suganthan P N, Hansen N, Liang J J, Deb K, Chen Y P, Auger A, and Tiwari S. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University, Singapore and IIT Kanpur, India,, Technical Report, Nanyang Technological University, Singapore, AND KanGAL Report 2005005, IIT Kanpur, India, 2005.

[18] Rodrigo Nakamura, Luis Pereira, Kellen Costa, Douglas Rodrigues, Joao Papa, and Xin-She Yang, and. Bba: A binary bat algorithm for feature selection. In Brazilian Symposium of Computer Graphic and Image Processing, pages 291 –297, 08 2012. ISBN 9780124051638.

[19] Online. Dataset. https://www.kaggle.com, 2023. [Online; accessed Feb-2023].

[20] V. Rodoplu and T.H. Meng. Minimum energy mobile wireless networks. IEEE Journal on Selected Areas in Communications, 17(8):1333–1344, 1999. doi: 10.1109/49.779917.

[21] Hamed Shah, Hosseini. The intelligent water drops algorithm: a nature-inspired swarm based optimization algorithm. Int. J. Bio-Inspired Comput., 1(1/2):71–79, January 2009. ISSN 1758-0366.

[22] Hussain Shareef, Ahmad Ibrahim, and Ammar Mutlag. Lightning search algorithm. Applied Soft Computing, 36:315–333, 08 2015. doi: 10.1016/j. asoc.2015.07.028.

[23] Omid Tarkhaneh and Haifeng Shen. Training of feedforward neural networks for data classification using hybrid particle swarm optimization, mantegna le´vy flight and neighborhood search. Heliyon, 5(4):e01275, 2019. ISSN 2405-8440. doi: https://doi.org/10.1016/j.heliyon.2019.e01275.

[24] T. White and B. Pagurek. Towards multi-swarm problem solving in networks. In Multi-Agent Systems, International Conference on, page 333, Los Alamitos, CA, USA, jul 1998. IEEE Computer Society. doi: 10.1109/ ICMAS.1998.699217. URL https://doi.ieeecomputersociety.org/ 10.1109/ICMAS.1998.699217.

[25] Jiankai Xue and Bo Shen. A novel swarm intelligence optimization approach: sparrow search algorithm. Systems Science & Control Engineering, 8(1): 22–34, 2020. doi: 10.1080/21642583.2019.1708830. URL https://doi.org/10.1080/21642583.2019.1708830.

[26] Xin-She Yang. Firefly algorithms for multimodal optimization. In Osamu Watanabe and Thomas Zeugmann, editors, Stochastic Algorithms: Foundations and Applications, pages 169–178, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04944-6.

[27] Xin-She Yang and Suash Deb. Cuckoo search via le´vy flights. In 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), pages 210–214, 2009. doi: 10.1109/NABIC.2009.5393690.

[28] Xin-She Yang and Amir Gandomi. Bat algorithm: A novel approach for global engineering optimization. Engineering Computations, 29, 11 2012.

[29] Xin Yao. Evolving artificial neural networks. Proceedings of the IEEE, 87 (9):1423–1447, 1999. doi: 10.1109/5.784219.

[30] Jianbo Yu, Lifeng Xi, and Shijin Wang. An improved particle swarm optimization for evolving feedforward artificial neural networks. Neural Processing Letters, 26:217–231, 10 2007. doi: 10.1007/s11063-007-9053-x.

[31] Jing-Ru Zhang, Jun Zhang, Tat-Ming Lok, and Michael R. Lyu. A hybrid particle swarm optimization back-propagation algorithm for feedforward neural network training. Appl. Math. Comput., 185:1026–1037, 2007