IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 August 2022, accepted 31 August 2022, date of publication 6 September 2022, date of current version 15 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204752

RESEARCH ARTICLE

An Optimized Continuous Dragonfly Algorithm Using Hill Climbing Local Search to Tackle the Low Exploitation Problem

BIBI AAMIRAH SHAFAA EMAMBOCUS, MUHAMMED BASHEER JASSER, (Member, IEEE), AND ANGELA AMPHAWAN, (Member, IEEE)

Department of Computing and Information Systems, School of Engineering and Technology, Sunway University, Petaling Jaya 47500, Malaysia

Corresponding author: Muhammed Basheer Jasser (basheerj@sunway.edu.my)

This work was supported by Sunway University Internal Grant Scheme 2022 under Grant GRTIN-IGS-DCIS[S]-11-2022.

ABSTRACT Optimization problems are usually solved using heuristic algorithms such as swarm intelligence algorithms owing to their ability of providing near-optimal solutions in a feasible amount of time. An example of an optimization problem is the training of artificial neural networks to obtain the most optimal connection weights. Artificial Neural Network (ANN), being the most prominent machine learning algorithm, has a multitude of applications in a myriad of areas. Recently, the use of ANNs has risen exponentially owing to its effective ability of making conclusions based on certain inputs. This ability is primarily achieved during the training phase of the ANN, which is a vital process prior to being able to use the ANN. Gradient descent-based algorithms, which are usually used for the training process, often encounter the problem of local optima, thus being unable to obtain the optimal connection weights of the ANN. Metaheuristic algorithms, including swarm intelligence algorithms, have been found to be a better alternative to train ANNs. The Dragonfly Algorithm (DA) is a swarm intelligence algorithm that has been found to be more effective than multiple swarm intelligence algorithms. However, despite having a good performance, it still suffers from low exploitation. In this paper, we propose to further improve the performance of DA by using hill climbing as a local search technique so as to enhance its low exploitation. The optimized DA algorithm is then used for training artificial neural networks which are employed for classification problems. Based on the experimental results, the optimized DA algorithm has higher effectiveness than the original DA and some other swarm intelligence algorithms as the ANNs trained by the optimized DA have a lower root mean squared error and a higher classification accuracy.

INDEX TERMS Artificial neural networks, swarm intelligence, dragonfly algorithm, optimization.

I. INTRODUCTION

Optimization problems consist of finding the best solution from a set of solutions such that a specified criterion is either maximized or minimized. These problems can be solved using either deterministic or heuristic algorithms. Since deterministic algorithms consist of exact methods for solving optimization problems, they are usually computationally expensive. Heuristic algorithms such as swarm intelligence algorithms are therefore preferred algorithms for solving optimization problems by providing near-optimal solutions in a feasible amount of time. For example, they are employed in the signalized traffic problem [1], in index tracking, [2] and for solving the traveling salesman problem [3], [4]. Another example of an optimization problem is the training of artificial neural networks in which the best values for the connection weights need to be determined during the training process.

Artificial Neural Networks (ANNs) are computational models that aim at simulating the behaviour of the human brain in order to draw conclusions based on some information provided. Being a preponderant algorithm in the field of

The associate editor coordinating the review of this manuscript and approving it for publication was Valentina E. Balas.

machine learning, ANN has propelled advances in numerous areas, for instance, natural language processing, speech recognition, computer vision, computational biology, fraud detection, unassisted control of vehicles, medical diagnosis and recommendation systems [5]. Some recent applications of neural networks include meteorological forecasting [6], fault diagnosis and detection in engineering-related systems [7], prediction of laser-cut edges surface roughness [8], analysis of individual's perception of IoT-based smart healthcare monitoring devices [9], solar energy prediction [10], and prediction of the emission characteristics of biodiesel-based fuel engine [11]. Moreover, it has various industrial applications [12], financial applications [13], engineering design applications [14], geotechnical engineering applications [15], social sciences applications [16], and neuroscientific applications [17].

ANNs usually consist of an input layer, one or multiple hidden layers and an output layer. Each layer consists of nodes, called neurons, which are connected to neurons of the previous and the following layers by weighted links. During the training of an ANN, the connection weights are adjusted so as to allow the ANN to generate the correct output based on the inputs. For supervised learning, these weights are adjusted such that the difference between the output value from the ANN and the known value from the dataset is minimized [18]. Hence, a trained neural network consists of an optimized set of connection weights, and this allows the neural network to produce accurate outputs based on the information provided, even if the inputs were not used during its training.

Feedforward neural networks are artificial neural networks in which information is transferred in only one direction, that is, from the input layer to the hidden layers and then to the output layer. There are no feedback connections or loops in a feedforward neural network. It consists of neurons, also called processing units, in each layer and each neuron is connected to neurons from the previous layer by connection weights. The neurons have the capability of processing the information received through the connection weights. The structure of a feedforward neural network makes it ideal for approximating any continuous function, thereby making it a universal function approximator [19].

Conventional algorithms used for training neural networks are gradient descent-based such as backpropagation, conjugate gradient, Quasi-Newton, Gauss-Newton, or Levenberg-Marquardt [19]. However, since these algorithms are local search algorithms which are beneficial for the exploitation of the state space but not its exploration, they tend to be trapped in local optima [19], and they are unable to get the most optimal set of weights for the neural network. Hence metaheuristic algorithms such as swarm intelligence algorithms which are endowed with both exploration and exploitation capabilities are promising training algorithms for artificial neural networks.

Swarm intelligence algorithms make use of the collective behaviour of simple agents which interact in a decentralized and self-organized manner for solving optimization problems [20]. These algorithms are inspired by the behaviour of biological organisms such as insects and animals as they interact among themselves and their environment. They make use of a population of artificial search agents which aim at obtaining the solution in a search space that corresponds to the maximum or the minimum value of a certain objective function. Some popular swarm intelligence algorithms include the Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Bee Colony Optimization (BCO) which are inspired by bird flocks or fish schools, ant colonies, and bee colonies respectively. Some recent swarm intelligence algorithms include Grey Wolf Optimizer (GWO) [21], Owl Search Algorithm (OSA) [22], Sparrow Search Algorithm (SSA) [23], Harris Hawks Optimization (HHO) [24], Moth-Flame Optimization (MFO) [25] and Rat Swarm Optimizer (RSO) [26].

The Dragonfly Algorithm (DA) [27] is a swarm intelligence algorithm which is inspired by the behaviour of dragonflies, specifically their hunting and migrating behaviours. These two behaviours of dragonflies aptly represent the two crucial phases of optimization algorithms; exploration and exploitation. Hence, DA makes use of these behaviours, and a population of artificial dragonflies so as to get the optimal solution in a search space. DA has been found to perform better than multiple swarm intelligence algorithms in various applications as we have seen from our work in [28]. However, it still has some limitations such as a low exploitation phase [29]. This results in low accuracy of solutions, local optima problem, and low convergence rate.

The Hill Climbing algorithm is a local search optimization algorithm which has high exploitation. This is because it always selects a solution which is better than the current solution, that is, one which optimizes the cost of the objective function, until the local optimal solution is achieved [30]. Hill climbing has been successfully used to increase the effectiveness of various swarm intelligence algorithms, in particular by enhancing their exploitation phase. It has also been used to increase the convergence rate of some swarm intelligence algorithms by accelerating the search process. However, it has not been used in any existing hybrid of DA to enhance the low exploitation of the original dragonfly algorithm.

In [31], we have proposed the idea of improving the low exploitation of the continuous DA by using the hill climbing algorithm as a local search technique. The continuous version of DA is used for solving continuous optimization problems like the training of ANNs. A continuous optimization problem means that the state space consists of real values within a specified range of values, that is, a potential solution can be any real number within a specific range. However, the algorithm was not implemented or applied to any optimization problem in [31]. Since the original dragonfly algorithm has been found to have a higher performance than other swarm intelligence algorithms in various applications [28],

it is worth improving its low exploitation phase and applying it for solving optimization problems.

In this paper, we implement and propose an improved continuous dragonfly algorithm with a better exploitation phase. The exploitation of DA is improved by using the stochastic hill climbing algorithm as a local search technique. The stochastic hill climbing is one of the main types of hill climbing algorithms in which a random neighbouring position which is better than the current position is chosen as the current position in each iteration of the algorithm. The pseudocode of the proposed algorithm is provided along with some explanations. The improved continuous DA algorithm is applied as a training algorithm for feedforward neural networks which are employed for benchmark classification problems, namely, the classification of the iris dataset, the balloon dataset, the glass dataset, and the breast cancer dataset from the UCI Machine Learning Repository [32], and the results are recorded. Based on the experimental results, the proposed optimized DA algorithm provides better solutions as compared to the original DA and other swarm intelligence algorithms.

The remaining of the paper is structured as follows: In Section II, some previous works which have used swarm intelligence algorithms to train artificial neural networks are presented. In Section III, a background on the dragonfly algorithm is presented and in Section IV, a background on the hill climbing algorithm is given. In Section V, a detailed description of the proposed algorithm is provided and in Section VI, a description of how the proposed algorithm is used for training an artificial neural network is provided. In Section VII, the experimental results are provided and finally in Section VIII, the conclusions and future works are provided.

II. SWARM INTELLIGENCE ALGORITHMS USED FOR TRAINING ANN

In this section, some previous works which have employed swarm intelligence algorithms as the training algorithm for artificial neural networks are presented. Table 1 shows a comparison of these works in terms of the swarm intelligence algorithm used for training the ANN, the type and architecture of the ANN used, the application for which the ANN is used, and its performance in terms of effectiveness.

In [33], Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony Optimization (ABC) and firefly algorithm are used for the training of an Artificial Neural Network (ANN) to get the optimal weights for the ANN so as to increase its prediction accuracy. The neural network consists of one input layer with 20 neurons, one hidden layer with 50 neurons and one output layer with one neuron and it is used for fault prediction in object-oriented systems by using the NASA public dataset. Specifically, the model is employed to identify and predict whether the classes in the object-oriented system are faulty. To determine the performance of the ANNs trained using the swarm intelligence algorithms, their prediction accuracy and time taken per run are compared to an ANN trained by gradient descent. The results show that all four neural networks trained by the swarm intelligence algorithms have a higher prediction accuracy than the ANN trained by gradient descent. The ANNs trained by the firefly algorithm, ACO, ABC and PSO have an average improvement of 18.559%, 28.606%, 38.852% and 50.75% respectively in the fault prediction over the ANN trained by gradient descent. The average time taken by the ANN trained by gradient descent is 21.038 seconds per run, and those trained by firefly, ACO, ABC, and PSO are 22.796 seconds, 5.568 seconds, 56.339 seconds and 5.235 seconds respectively. Hence, it can be seen that the ANNs trained by ACO and PSO are more efficient than that trained by gradient descent while those trained by firefly and ABC are less efficient than the ANN trained by gradient descent.

In [34], PSO is used for training an ANN which is then used for the prediction of the load-slip behaviour of channel connectors embedded in normal and high-strength concrete. The ANN consists of an input layer with five neurons, one hidden layer with 10 neurons, and an output layer with one neuron. 70% of the data is used for the training of the ANN, and the other 30% is used for testing the ANN. To assess the performance of the ANN trained by PSO, the root mean squared error (RMSE), Pearson correlation coefficient (r), and determination coefficient (R2) of the resultant ANN are generated. Its performance is also compared to another ANN of similar architecture which is trained by the Levenberg–Marquardt backpropagation algorithm. In both the training and testing phases, the ANN trained by PSO has higher r and R2 values and lower RMSE as compared to the ANN trained by backpropagation, which indicates that the ANN trained by PSO has superior prediction accuracy. In the testing phase, the RMSE value of the ANN trained by PSO is 2.069 while the RMSE value of the ANN trained by backpropagation is 2.569. Hence, the RMSE value of the ANN trained by PSO is 19.5% lower than the RMSE value of the ANN trained by backpropagation.

In [35], Grasshopper Optimization Algorithm (GOA) and Gray Wolf Optimization (GWO) are used for the training of a neural network so as to increase its accuracy in the estimation of the heating load of residential buildings. The data which is obtained from the analysis of 768 residential buildings is randomly split for the training and testing of the ANNs. 80% of the data is utilized for the training phase and 20% is used for the testing phase. Three artificial neural networks, one which is trained by backpropagation, one which is trained by GOA (GOA-MLP), and one which is trained by GWO (GWO-MLP) are tested using the same data. The structure of the neural networks consists of one input layer with eight nodes, one output layer with one node and nine hidden layers with up to 15 neurons. The root mean-square error (RMSE), mean absolute error (MAE), and coefficient

of determination (R^2) are used to compare the performance of the three models. During the training phase, the RMSE of GOA-MLP, and GWO-MLP is 16.53% and 19.19% lower than the RMSE of the MLP trained by backpropagation respectively and the MAE is lower by 13.46% and 15.81%. The R^2 of both the GOA-MLP and GWO-MLP is higher than the R^2 of the MLP trained by backpropagation. During the testing phase, the RMSE of GOA-MLP, and GWO-MLP is 18.08% and 23.31% lower than the RMSE of the MLP trained by backpropagation respectively and the MAE is lower by 16.60% and 20.72%. The R^2 of the GOA-MLP, which is 0.9486, and that of GWO-MLP, which is 0.9551, is higher than the R^2 of the MLP trained by backpropagation, which is found to be 0.9328. These values indicate that the accuracy of both the GOA-MLP, and GWO-MLP are higher than the MLP trained by backpropagation and hence they have a better performance in the estimation of the heating load of residential buildings.

In [36], the original DA and Harris Hawks Optimization (HHO) algorithm are employed for optimization of the connection weights and biases of an ANN which is used in the analysis of the bearing capacity of footings over two-layer foundation soils. The failure probability is predicted by considering seven factors as inputs, namely, unit weight, friction angle, elastic modulus, dilation angle, Poisson's ratio, applied stress, and setback distance. 80% of the data is used for training three Multi-Layer Perceptrons (MLP) and the rest 20% is used for testing the neural networks. One MLP is trained using backpropagation, one is trained using DA (DA-MLP), and one is trained using HHO (HHO-MLP). The mean squared error, and the mean absolute error are used to compare the performance of the three neural networks. During the testing phase, the MSE obtained for the MLP trained by backpropagation is 0.1416, that of HHO-MLP is 0.1350, which is a 4.66% decrease, and that of DA-MLP is 0.1171, which is a 17.30% decrease. The MAE for the MLP trained by backpropagation is found to be 0.3230, that of the HHO-MLP is found to be 0.3200, which is a 0.93% decrease, and that of DA-MLP is found to be 0.2904, which is a 10.09% decrease. Moreover, the accuracy of the MLP trained by backpropagation is 89.0% while that of the HHO-MLP, and DA-MLP are 91.5% and 94.2% respectively. Hence, it can be seen that the DA-MLP outperforms the two other models.

In [37], a hybrid of an improved Nelder Mead ALgorithm and dragonfly algorithm, called INMDA is used for the training of an MLP by optimizing its weight and biases. The MLP is then used for three benchmark classification problems, namely the XOR problem, balloon classification problem, and heart classification problem. The performance of the MLP trained by INMDA is compared to that of MLPs trained by PSO, Evolution Strategy (ES), Grey Wolf Optimizer (GWO), Population-Based Incremental Learning (PBIL), and Genetic Algorithm (GA). The MSE obtained by the MLPs trained by INMDA, GWO, PSO, GA, ES, and PBIL are 4.64e-05, 0.009410, 0.084050, 0.000181, 0.118739, and 0.030228 for the XOR problem, 5.48e-16, 9.38e-15, 0.000585, 5.08e-24, 0.019055, and 2.49e-05 for the balloon classification problem, and 0.114351, 0.122600, 0.188568, 0.188568, 0.192473, and 0.154096 for the heart classification problem respectively. Hence, it can be seen that the MLP trained by INMDA achieves the lowest MSE for the XOR problem, the second lowest MSE for the balloon classification problem, and the lowest MSE for the heart classification problem.

In [38], the original DA algorithm is used to train an ANN which is used for the brain classification of Magnetic Resonance Images (MRI). The DA algorithm is used to avoid the local optimum problem usually faced by the backpropagation algorithm while training ANN and to increase the speed of the training process. The neural network consists of seven inputs that represent seven feature vectors, one output that can indicate either a normal or an abnormal brain, and one hidden layer with four nodes. DA is used to optimize the weights of the ANN and the sensitivity, specificity, and accuracy of the resultant neural network are calculated. The performance of the DA-based ANN is compared to that of GA-based ANN, PSO-based ANN, and backpropagation (BP)-based ANN. The sensitivity of the DA-based ANN, PSO-based ANN, GA-based ANN, and BP-based ANN is found to be 89%, 83%, 82%, and 77% respectively, the specificity is found to be 83%, 72%, 70%, and 68% respectively, and the accuracy is found to be 85%, 80.5%, 80%, and 75% respectively. Since, the values for the sensitivity, specificity, and accuracy of the DA-based ANN are higher than those of the other models, this indicates that the DA-based ANN has a better performance.

In [39], the original dragonfly algorithm is employed as the training algorithm for an MLP which is applied for the classification of sonar targets. DA is used to obtain the optimal values for the connection weights and biases of the MLP. The neural network used consists of one input layer with two neurons, one hidden layer with three neurons, and one output layer with one neuron. The classification accuracy of the MLP trained by DA is compared to that of MLPs trained by Grey Wolf Optimizer (GWO), Biogeography-Based Optimization (BBO), Multi-Verse Optimizer (MVO), Gravitational Search Algorithm (GSA), PSO, and ACO. The classification accuracy of the MLP trained by DA is found to be 92.1457%, 94.2154%, and 96.6523% for the three different datasets used. For all the datasets used, the classification accuracy of the MLP trained by DA is higher than that of the MLPs trained by GWO, BBO, MVO, GSA, PSO, and ACO.

III. DRAGONFLY ALGORITHM

The dragonfly algorithm makes use of the static and dynamic swarming behaviours of dragonflies during hunting and migration respectively [27]. While hunting, the population of dragonflies divide into small groups and they fly over a

TABLE 1. Comparison of artificial neural networks trained by swarm intelligence algorithms.

Paper	SI Algorithm	Type of ANN	ANN Architecture	Application	Effectiveness
[33]	Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony Optimization (ABC) and firefly algorithm	Multi Layer Perceptron (MLP)	One input layer with 20 neurons, one hidden layer with 50 neurons and one output layer with one neuron	Fault prediction in object oriented systems	Average improvement of 18.559% (firefly), 28.606% (ACO), 38.852% (ABC) and 50.75% (PSO) over gradient descent
[34]	Particle Swarm Optimization	MLP	One input layer with five neurons, one hidden layer with 10 neurons and one output layer with one neurons	Prediction of load-slip behaviour of channel connectors embedded in normal and high-strength concrete	19.5% lower RMSE value than ANN trained by Levenberg–Marquardt backpropagation
[35]	Grasshopper Optimization Algorithm and Gray Wolf Optimization	MLP	One input layer with eight nodes, one output layer with one node and nine hidden layers with up to 15 neurons	Estimation of heating load of residential buildings	18.08% and 23.31% lower RMSE than MLP trained by backpropagation, 16.60% and 20.72% lower MAE and higher R^2 value than MLP trained by backpropagation
[36]	Dragonfly Algorithm and Harris Hawks Optimization	MLP	One input layer with seven neurons, one hidden layer with six neurons, and one output layer with one neuron	Prediction of the failure probability of the bearing capacity of footings over two-layer foundation soils	4.66 % and 17.30 % lower MSE, and 0.93% (HHO) and 10.09% (DA) lower MAE than MLP trained by backpropagation
[37]	Hybrid DA-Nelder Mead (INMDA)	MLP	One input layer with three neurons, one hidden layer with seven neurons, and one output layer with one neuron (XOR Problem), one input layer with four neurons, one hidden layer with nine neurons, and one output layer with one neuron (Balloon classification problem), One input layer with 22 neurons, one hidden layer with 45 neurons, and one output layer with one neuron (Heart classification problem)	Benchmark classification problems (XOR Problem, Balloon Classification Problem, Heart Classification Problem)	4.64e-05 (XOR problem), 5.48e-16 (balloon classification problem), and 0.114351 (Heart classification problem) MSE
[38]	Original DA	MLP	One input layer with seven nodes, one hidden layer with four nodes, and one output layer with one node	Classification of MRI brain images	Higher sensitivity(89%), specificity(83%), accuracy (85%) than PSO-based ANN, GA-based ANN, BP-based ANN
[39]	Original DA	MLP	One input layer with two nodes, one hidden layer with three nodes, and one output layer with one node	Classification of sonar targets	Classification accuracy of 92.1457%, 94.2154%, and 96.6523% for three datasets

small area by abruptly changing their flying path in order to hunt other flying insects. This behaviour is congruent with the exploration phase of optimization algorithms where search agents try to find good regions of the state space. Conversely, while migrating, the whole population of dragonflies come together to form one big group and they fly together for a long distance in one direction. This behaviour is congruent with the exploitation phase of optimization algorithms where search agents try to locate the global best solution after a promising search area is found. Hence, the dragonfly algorithm employs these two behaviours for solving optimization problems. Fig. 1 shows a static and a dynamic swarm of dragonflies.

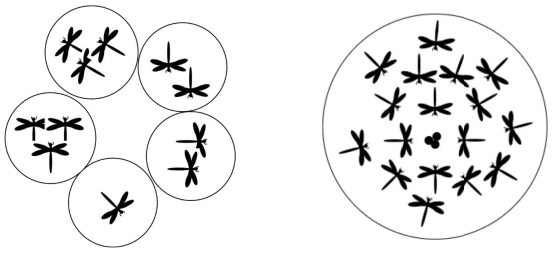

FIGURE 1. Static and dynamic swarms of dragonflies [27].

During both the exploration and exploitation phases, five factors are used to control the movement of the dragonflies

in the search space, namely, separation, alignment, cohesion, attraction to food and distraction from enemy. Each of these factors has a corresponding weight.

Separation is used to avoid static collision of a dragonfly and its neighbours, and is calculated as follows:

$$S_i = -\sum_{j=1}^{N} X_i - X_j \tag{1}$$

where S_i, X_i, X_j, and N represent the separation of the $i-$th dragonfly, the position of the $i-$th dragonfly, the position of the $j-$th neighbour, and the number of neighbouring dragonflies respectively.

Alignment is used to match the velocity of a dragonfly to that of its neighbours, and is calculated as follows:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \tag{2}$$

where A_i is the alignment of the $i-$th dragonfly, and V_j is the velocity of the j-th neighbour.

Cohesion is the tendency of one dragonfly towards the center of mass of the neighborhood, and is calculated as follows:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X_i \tag{3}$$

where C_i is the cohesion of the $i-$th dragonfly, and X_j is the position of the j-th neighbour.

Attraction to food is used to attract a dragonfly towards the food source which is taken as the best position obtained by the population, and is calculated as follows:

$$F_i = X^+ - X_i \tag{4}$$

where X^+ represent the position of the food source.

Distraction from enemy is used to distract a dragonfly away from the enemy which is taken as the worst position obtained by the population, and is calculated as follows:

$$E_i = X^- + X_i \tag{5}$$

where X^- represent the position of the enemy.

In order for the dragonflies to move in the search space by making use of these factors, two vectors are used, namely, step vector (ΔX), and position vector (X). The step vector indicates the direction of movement, and is calculated as follows:

$$\Delta X_i^{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_i^t \tag{6}$$

where s, S_i, a, A_i, c, C_i, f, F_i, e, E_i, w, and t represent the separation weight, the separation of the i-th dragonfly, the alignment weight, the alignment of i-th dragonfly, the cohesion weight, the cohesion of the i-th dragonfly, the weight of the food factor, the food factor of the i-th dragonfly, the weight of the enemy factor, the enemy factor of the i-th dragonfly, the inertia weight, and the iteration counter respectively.

The position vector allows the dragonflies to move in the search space by updating their positions using:

$$X_i^{t+1} = X_i^t + \Delta X_i^{t+1} \tag{7}$$

In DA, the neighbourhood of the dragonflies is an important aspect. This is considered by assuming a radius around each dragonfly. The radius is incremented proportionally to the iteration counter to enable the transition from the exploration to the exploitation phase, thereby changing the static swarms at the early iterations into dynamic swarms. During the final iterations, the whole population forms one dynamic swarm and converges to the global optimal solution. Another way by which the algorithm transitions from exploration to exploitation is by adaptively tuning the weights for the different factors.

If a dragonfly has no neighbours at some point, its position is updated using the Lévy flight mechanism. This is a random walk that increases the randomness of the algorithm. The position vector used is:

$$X_i^{t+1} = X_i^t + Levy(d) \times X_i^t \tag{8}$$

where t, and d represent the current iteration number and the dimension of the position vectors respectively.

The Lévy flight mechanism is calculated using (9):

$$Levy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \tag{9}$$

where r_1, and r_2 are random numbers between 0 and 1. β is a constant which is taken as 1.5 [27], and σ is calculated using (10):

$$\sigma = \left(\frac{\Gamma(1+\beta) \times sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{1/\beta} \tag{10}$$

where $\Gamma(x) = (x-1)!$.

Algorithm 1: Dragonfly Algorithm

```
Initialize the population's positions randomly;
Initialize the step vectors;
while end condition do
    Calculate the objective values of all dragonflies;
    Update the food source and enemy;
    Update the weights;
    Calculate the factors using (1)–(5);
    Update radius of neighbourhoods;
    if dragonfly has one or more neighbours then
        Update step vector using (6);
        Update position vector using (7);
    else
        Update position vector using (8);
    end
    Check and correct new positions based on upper and lower bounds;
end
```

FIGURE 2. Dragonfly algorithm pseudocode.

The pseudocode of DA is given in Fig. 2.

IV. HILL CLIMBING ALGORITHM

The Hill Climbing algorithm is a heuristic local search algorithm usually employed to solve optimization problems. Hill climbing works by optimizing a specified objective function. It starts at a random solution and then selects other neighbouring solutions which improve the value of the objective function. It is terminated when the value of the objective function can no longer be improved, that is when the local optimum is reached [30]. Hence, hill climbing is an effective method to quickly locate the local optimum of a search region.

The stochastic hill climbing algorithm is one of the main types of hill climbing algorithms. It works by starting at an initial position, called the current position. Then it chooses a random neighbour of the current position which is better than the current position as the current position. This process is repeated until the current position can no longer be optimized.

The pseudocode of the stochastic hill climbing algorithm is given in Fig. 3.

Algorithm 2: Stochastic Hill Climbing Algorithm

```
current position = initial solution;
repeat
    for All neighbours of current position do
        Obtain a random neighbour;
        if cost of neighbour ≤ cost of current position then
            current position = neighbour position;
            break;
        end
    end
until cost of current position ≤ cost of all its neighbours;
```

FIGURE 3. Stochastic hill climbing pseudocode.

Hill climbing has been employed as a local search technique for multiple swarm intelligence algorithms to improve their performance, especially by improving their exploitation. For example, it has been used to improve the exploitation phase of Ant Colony Optimization (ACO) [40], salp swarm algorithm [41], Artificial Bee Colony (ABC) [42], moth-flame optimization [43], and cuckoo search [44]. Moreover, it has also been used to increase the convergence rate of ABC [45], cuckoo search [44], moth-flame optimization [43], and PSO [46].

V. PROPOSED OPTIMIZED CONTINUOUS DA ALGORITHM

In this section, a description of the proposed algorithm is presented. The proposed algorithm enhances the exploitation of the original dragonfly algorithm. This is done by using the stochastic hill climbing algorithm as a local search technique. In every iteration of the algorithm, after the position of the dragonflies is updated using the step and position vectors of DA, the stochastic hill climbing algorithm is applied to further improve the position by further exploiting the search area. Specifically, lines one to ten from Algorithm 2 are integrated after line 11 in Algorithm 1 so as to further update the position which has been found by equation 7. The position found by equation 7 is taken as the initial solution for the hill climbing algorithm, that is, it is taken as '*current position*' from line one in Algorithm 2. The final '*current position*' obtained from Algorithm 2 is then taken as the new position of the dragonfly for the optimized DA algorithm.

Hill climbing is not applied after the position of the dragonflies is updated using equation 8 since this equation is used for exploration of the search space and not its exploitation. Moreover, equation 8 makes use of the levy flight mechanism to update the position of dragonflies which have no neighbours. This is a random walk which updates the position of the dragonfly in a stochastic manner. Hence, this may mean that the dragonfly is not in a good region of the search space and hence there is no need to exploit the region using the hill climbing algorithm.

Algorithm 3: Optimized Continuous Dragonfly Algorithm

```
Initialize the population's positions randomly;
Initialize the step vectors;
Initialize step size for hill climbing;
while end condition do
    Calculate the objective values of all dragonflies;
    Update the food source and enemy;
    Update the weights;
    Calculate the factors using (1)–(5);
    Update radius of neighbourhoods;
    if dragonfly has one or more neighbours then
        Update step vector using (6);
        Update position vector using (7);
        Initialize current position as initial position for hill climbing;
        repeat
            for All neighbours of current position do
                Generate a random neighbour using step size;
                if cost of neighbour ≤ cost of current position then
                    current position = neighbour position;
                    break;
                end
            end
        until cost of current position ≤ cost of all its neighbours;
    else
        Update position vector using (8);
    end
    Check and correct new positions based on upper and lower bounds;
end
```

FIGURE 4. Optimized continuous DA pseudocode.

This method of employing the hill climbing algorithm as a local search technique after updating the position of the dragonflies using equation 7, allows the dragonflies to update their position to a better one in the area that has been obtained by DA. This is because the hill climbing algorithm starts at the position obtained by DA and then only updates it to a better one until it can no longer be updated to a better position. Hence, the exploitation phase of DA is improved, and this increases the effectiveness of the dragonfly algorithm, that is, better solutions are obtained as compared to the original dragonfly algorithm. The pseudocode of the proposed algorithm is given in Figure 4.

In the initialization phase of the proposed algorithm, the population of artificial dragonflies is first initialized with random positions and step vectors. Then the step size which is used to generate neighbours for the hill climbing algorithm is initialized. In this paper, the step size used is 0.05.

In the main loop of optimization, the objective values of all the dragonflies are first calculated based on their positions. The food source and the enemy are then updated by taking the positions of the dragonflies which correspond to the best, and the worst objective values respectively. The weights, s, a, c, f, and e are then updated and the separation, alignment, cohesion, attraction to food, and distraction from enemy factors of each search agent are calculated using equations (1)–(5). The radius of neighbourhoods is then updated by increasing it based on the iteration number.

The step and position vectors are then used for updating the position of the dragonflies. For dragonflies having at least one neighbour, the step vector is first updated using equation 6 and then the position vector is updated using equation 7. The position obtained is then further updated using hill climbing as follows: firstly, the position obtained by DA is initialized as the initial position of the hill climbing algorithm. A random neighbouring solution is generated by using a step size of 0.05. This step size is either added or subtracted from a dimension of the current position to get a neighbouring position. If the objective value of the neighbouring solution is less than that of the current position, the neighbouring solution is taken as the current position. Then a random neighbour of the new current position is generated, and the process of generating neighbours, and updating the current position is repeated. If the objective value of a neighbour solution is not less than that of the current position, then other random neighbours are generated until one which is better than the current position is found or until all the neighbours are checked.

For dragonflies having no neighbours, the position is updated using equation 8. This equation makes use of the Lévy flight mechanism which is a random walk for updating the position of the dragonflies. This is because if a dragonfly has no neighbours, it may mean that it is in a bad region of the search space and hence its position is updated in a stochastic and random way to allow it to explore other regions of the search space. Hill climbing is not employed after this equation since the dragonflies having no neighbours are required to further explore the search space. Moreover, since it may mean that this region of the search space is not a promising one, there is no need to exploit this region.

After updating the positions of the dragonflies, the new positions are checked and corrected based on the upper and lower bounds, that is, if the new position is greater than the upper bound, it is given the value of the upper bound and similarly, if it is lower than the lower bound, it is given the value of the lower bound.

The loop of optimization is repeated until the end criteria is met, that is when the maximum number of iterations is reached.

VI. TRAINING OF ANN USING OPTIMIZED CONTINUOUS DA

In this section, we provide a description of how the optimized DA algorithm is used for the training of feedforward neural networks which are applied to classification problems by using the iris, balloon, glass, and breast cancer datasets from the UCI Machine Learning Repository [32].

Firstly, the data consisting of the inputs and the targets is loaded and it is split. 70% of the data is used for the training of the ANN and 30% is used for testing the ANN.

Secondly, the feedforward neural network is constructed. The architecture of the ANNs used for the different datasets is described in Section VII-A. An example of the architecture used for the iris dataset is shown in Fig. 5.

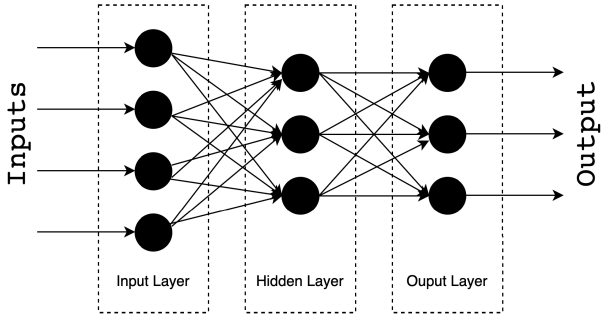

FIGURE 5. Architecture of feedforward neural network used.

After constructing the network, the total number of parameters, that is, the total number of weights and biases to be optimized during the training process is determined using the formula 11. This number is used as the dimension of the optimized DA algorithm since the set of all the connection weights and the biases need to be optimized. One set of weights and biases represents the position of one dragonfly in optimized DA.

$$(i \times n) + n + (o \times n) + o \tag{11}$$

where i, n, and o represent the input size, the number of hidden neurons, and the output size respectively.

The optimized DA algorithm is then employed to obtain the optimal set of connection weights and biases for the neural network. This step is the training stage of the ANN. Random sets of connection weights and biases are first generated and are used as the initial positions of the dragonflies. The positions are updated using the step and position vectors of the optimized DA as described in Section V. The Root Mean Squared Error (RMSE) of the ANN is used as the objective function. To calculate the objective value of a position, the set of connection weights and biases representing that position is assigned to the ANN. The RMSE of the ANN with

that set of weights and biases is calculated using 12. The upper and lower bound values are taken as 2 and -2 respectively. These values are determined based on experimental analysis.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_i - t_i)^2}{n}} \tag{12}$$

where y_i, t_i, and n represent the predicted value, the target value, and the total number of data samples respectively.

When the optimized DA algorithm converges to the global optimal solution, the most optimal set of connection weights and biases of the ANN, which results in the least RMSE for the testing data, is obtained. These connection weights and biases are then assigned to the neural network and the ANN is now considered a trained neural network.

After training the ANN, it is tested using the testing dataset. The RMSE of the resultant neural network is calculated using 12 and its accuracy is calculated using 13.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

where TP, TN, FP, and FN represent the number of true positives, the number of true negatives, the number of false positives, and the number of false negatives respectively.

VII. EXPERIMENTAL RESULTS

In this section, a description of the experimental dataset, experimental setup, and the results with some discussions are provided.

A. EXPERIMENTAL DATASET

Four classification datasets from the UCI Machine Learning Repository are used for conducting the experiments; the iris dataset, the balloon dataset, the glass dataset, and the breast cancer dataset. The iris dataset is a balanced dataset while the balloon, glass, and breast cancer datasets are unbalanced datasets.

The iris dataset consists of 150 instances and three classes. Each class represent a type of iris plant. Each instance has the following attributes: sepal length, sepal width, petal length, petal width, and class. The sepal length, sepal width, petal length, and petal width can be any real number while the class can be either 'Iris Setosa', 'Iris Versicolor' or 'Iris Virginica'. The aim of this problem is to correctly identify the class of the plant based on the other attributes, that is, the sepal length, sepal width, petal length, and petal width.

The balloon dataset consists of 16 instances and two classes. Four attributes, namely, color, size, act, and age are used to classify each instance into either one of the two classes, namely, 'inflated', or 'not inflated'. The aim is to correctly classify whether a balloon is inflated or not based on the mentioned attributes.

The glass dataset consists of 214 instances and two main classes. Nine different attributes representing the composition of the glass are used to identify the type of glass, specifically whether the glass is float processed or not.

The breast cancer dataset consists of 569 instances, 10 attributes, and two classes. The attributes consist of features of a cell nucleus and the aim is to classify whether it is malignant, or benign.

B. EXPERIMENTAL SETUP

The optimized DA algorithm is employed for the training of feedforward neural networks as described in section VI by using the iris dataset, the balloon dataset, the glass dataset, and the breast cancer dataset.

For all four datasets, 70% of the data is used for the training of the ANN and 30% is used for testing the ANN.

For the iris dataset, the architecture of the neural network used is one input layer with four neurons, one hidden layer with three neurons, and one output layer with three neurons.

To determine the number of neurons in the hidden layer, different ANNs are trained using the gradient-descent algorithm by changing the number of neurons in the hidden layer from one to five. The number of neurons which results in the least average RMSE and the highest average accuracy is then chosen. The average RMSE and accuracy obtained when the different ANNs are trained using gradient-descent are shown in Table 2. Since the least average RMSE and highest average accuracy are obtained when the number of neurons in the hidden layer is three, this architecture is chosen for the ANN.

TABLE 2. Comparison of different ANNs trained by gradient-descent.

Number of neurons in hidden layer	Average RMSE	Average Accuracy (%)
1	0.5908	22.0
2	0.5184	33.3
3	0.4950	36.4
4	0.6102	33.1
5	0.6285	23.97

For the balloon dataset, the architecture of the neural network used is one input layer with four neurons, one hidden layer with three neurons, and one output layer with one neuron. For the glass dataset, the architecture is one input layer with nine neurons, one hidden layer with three neurons, and one output layer with one neuron, and for the breast cancer dataset, the architecture is one input layer with nine neurons, one hidden layer with three neurons, and one output layer with one neuron.

For all four datasets, the performance of the optimized DA algorithm in training an ANN is compared to that of the original DA algorithm. This is done by training similar ANNs with the same number of layers and neurons using the original DA algorithm. Then the two neural networks trained by the optimized DA and the original DA are compared in terms of the final RMSE of the neural network during the training phase, the RMSE of the resultant neural network during the testing phase, the accuracy of the resultant neural network, and the time taken for the algorithms

TABLE 3. Performance comparison of optimized DA and original DA in training ANN with 3 neurons in hidden layer using iris dataset.

Number of Search Agents	Max num of iterations	Original DA				Optimized DA			
		Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)	Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)
5	10	0.39094	0.38897	66.7	1.586	0.13035	0.1041	100.0	968.9189
10	10	0.50835	0.49418	57.8	3.055	0.13563	0.11344	100.0	37.9407
10	20	0.34219	0.36114	80.0	5.6102	0.13059	0.099124	100.0	3886.9033

TABLE 4. Performance comparison of optimized DA and original DA in training ANN with 3 neurons in hidden layer using balloon dataset.

Number of Search Agents	Max num of iterations	Original DA				Optimized DA			
		Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)	Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)
5	10	0.56337	0.44302	100.0	1.7518	0.0042219	0.78259	100.0	1051.3094
10	10	0.3821	0.29864	100.0	1.859	0.025127	0.49063	100.0	316.1989
10	20	0.31102	0.38359	100.0	6.305	0.0093682	0.45733	100.0	2431.062

to converge to the optimal solution. Moreover, the convergence curve for both the optimized DA and the original DA in minimising the RMSE of the ANNs is drawn to compare the rate of convergence of the optimized DA and the original DA.

Different experiments are conducted by changing the number of search agents and the maximum number of iterations for both the optimized DA and the original DA by using the four different classification datasets. Specifically, 5, and 10 search agents are used for 10, and 20 iterations.

Both the optimized DA and the original DA were implemented in MATLAB and all experiments were conducted on a macOS Big Sur operating system, 2.9 GHz Dual-Core Intel Core i5 CPU, and 8 GB RAM.

To further analyse the performance of the Optimized DA algorithm, its performance in training ANNs using the iris, balloon, glass, and breast cancer datasets is compared to that of other swarm intelligence algorithms. The results obtained when several other swarm intelligence algorithms are used for training ANNs using the iris, balloon, glass, and breast cancer datasets are taken from [37], [47], and [48]. Specifically, the results obtained when ANNs are trained using Ant Colony Optimization (ACO), Ant Lion Optimization (ALO), Bat Algorithm (BAT), Biogeography-based Optimization (BBO), Cuckoo Search (CS), Differential Evolution (DE), Elephant Herding Optimization (EHO), Evolution Strategy (ES), Genetic Algorithm (GA), Gravitational Search Algorithm (GSA), Grey Wolf Optimization (GWO), Harmony Search (HS), Moth-Flame Optimization (MFO), Multiple Sequence Alignment (MSA), Particle Swarm Optimization (PSO), Sine Cosine Algorithm (SCA), Whale Optimization Algorithm (WOA), and Hybrid ABC-DA (HAD) algorithms are taken from [47], those trained by Hybrid Nelder-Mead and DA (INMDA) and Population-Based Incremental Learning (PBIL) algorithms are taken from [37], and those trained by Multiple Leader Salp Swarm Algorithm (MLSSA) and Salp Swarm Algorithm (SSA) algorithms are taken from [48].

In order to have a fair comparison, the optimized DA is used to train ANNs with the same architecture as in [37], [47], and [48]. Hence, the number of neurons in the hidden layer is changed to nine for the iris dataset and balloon dataset, and to 19 for the glass and breast cancer datasets. More experiments are then conducted by using the optimized DA to train ANNs with the mentioned architectures and the results are recorded and compared to that obtained by other swarm intelligence algorithms.

C. RESULTS AND DISCUSSIONS

1) PERFORMANCE COMPARISON OF OPTIMIZED DA AND ORIGINAL DA IN TRAINING ANN

Tables 3, 4, 5, and 6 show the results obtained when the optimized DA and the original DA are used for training ANNs using the iris dataset, the balloon dataset, the glass dataset, and the breast cancer dataset.

For the iris dataset, the following architecture is used for the neural network: one input layer with four neurons, one hidden layer with three neurons, and one output layer with three neurons. For the balloon dataset, the architecture used for the neural network is: one input layer with four neurons, one hidden layer with three neurons, and one output layer with one neuron. For the glass dataset, the following architecture is used: one input layer with nine neurons, one hidden layer with three neurons, and one output layer with one neuron. For the breast cancer dataset, the architecture used is: one input layer with nine neurons, one hidden layer with three neurons, and one output layer with one neuron.

The number of search agents used is five and 10 and the maximum number of iterations used is 10, and 20. The results

TABLE 5. Performance comparison of optimized DA and original DA in training ANN with 3 neurons in hidden layer using glass dataset.

Number of Search Agents	Max num of iterations	Original DA				Optimized DA			
		Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)	Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)
5	10	1.9915	1.4487	96.9	1.4611	0.89462	1.2524	100.0	3683.5255
10	10	2.1809	2.8743	96.9	2.7182	0.87841	1.2398	100.0	2858.5367
10	20	1.4505	2.2303	95.3	5.5139	0.84181	1.2694	100.0	11596.3256

TABLE 6. Performance comparison of optimized DA and original DA in training ANN with 3 neurons in hidden layer using breast cancer dataset.

Number of Search Agents	Max num of iterations	Original DA				Optimized DA			
		Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)	Training RMSE	Testing RMSE	Accuracy (%)	Time Taken To Converge (s)
5	10	0.47874	0.50761	63.3	1.8333	0.15693	0.15877	96.7	4086.7443
10	10	0.4692	0.432	91.0	1.5417	0.15263	0.15594	97.1	4743.1829
10	20	0.23353	0.20903	94.3	5.8366	0.14695	0.14045	97.1	18828.3563

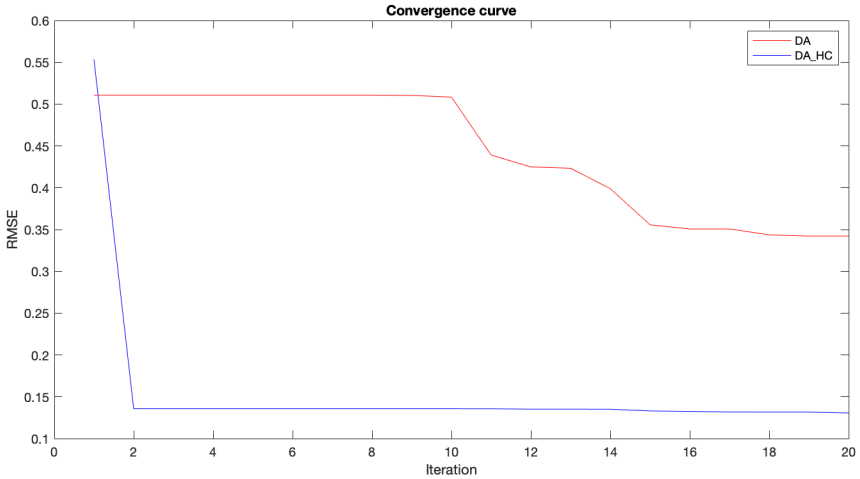

FIGURE 6. Effectiveness and convergence of original DA and optimized DA for iris dataset.

are compared using the Root Mean Square Error (RMSE) obtained in the training and testing phases, the accuracy obtained in the testing phase, the time taken to converge to the global optimal solution, and the total time taken. A larger number of search agents and maximum number of iterations is not used for conducting the experiments and comparing the performance of the optimized DA and the original DA since the optimized DA can provide very high accuracy with five and 10 search agents for a maximum of 10, and 20 iterations. Moreover, even if the number of search agents and maximum iteration is increased, the optimized DA will still provide better solutions than the original DA. This is because the optimized DA makes use of the hill climbing algorithm to improve the exploitation of the original DA, thereby considering other better solutions which might never be considered using the original DA. Hence, the optimized DA algorithm considers more solutions in the search space and is able to provide better solutions than the original DA.

Fig. 6, 7, 8, and 9 show the convergence curve of the original DA and the optimized DA algorithms in training ANNs with one hidden layer consisting of three neurons for the iris, balloon, glass, and breast cancer datasets respectively.

From Tables 3, 4, 5, and 6, and from Fig. 6, 7, 8, and 9, it can be deduced that the optimized DA algorithm has a better performance as compared to the original DA in terms of the effectiveness, that is, it is able to better optimize the connection weights and biases of the neural networks during the training phase, which leads to better accuracy of the resultant neural network.

From Tables 3, 4, 5, and 6, it can be seen that the RMSE obtained when the ANN is trained using the optimized DA

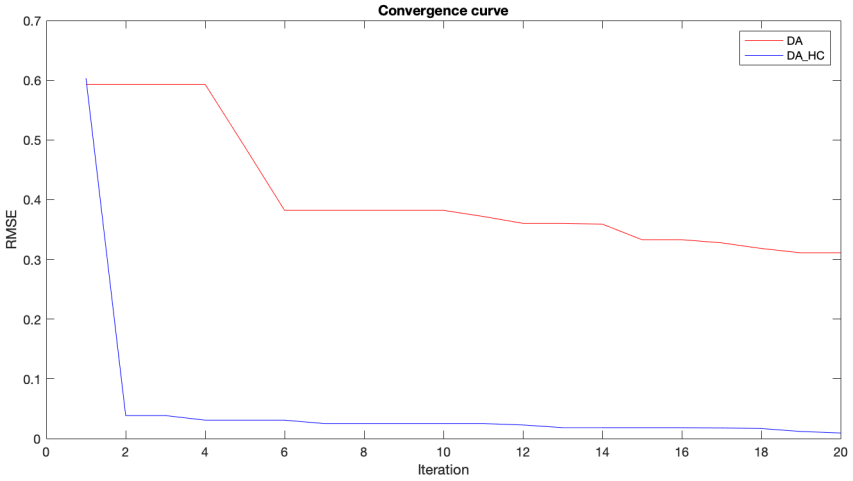

FIGURE 7. Effectiveness and convergence of original DA and optimized DA for balloon dataset.

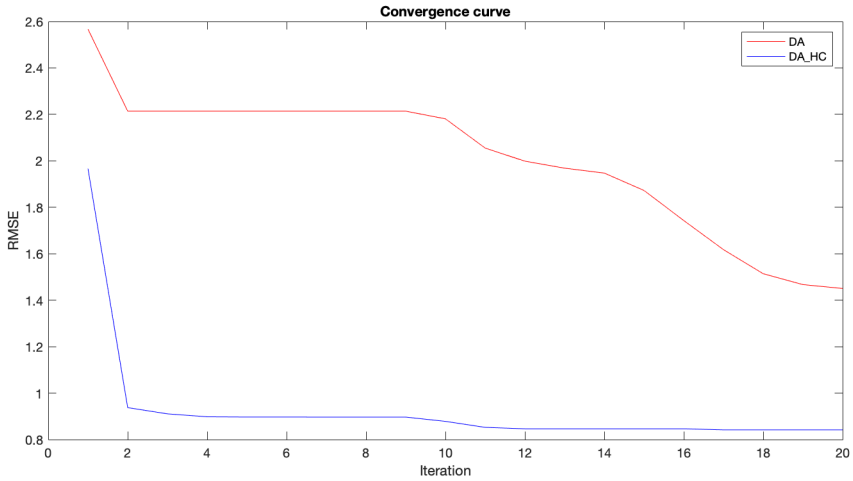

FIGURE 8. Effectiveness and convergence of original DA and optimized DA for glass dataset.

is lower than the RMSE obtained when it is trained by the original DA for all the conducted experiments and for both the training and the testing phases of the ANN. Moreover, in all the experiments conducted, the resultant neural network, which has been trained by the optimized DA algorithm, achieves a higher accuracy as compared to the neural network trained by the original DA. For the iris, balloon, and glass datasets, the ANN trained by the optimized DA can achieve 100% accuracy, and for the breast cancer dataset, it can achieve very high accuracy which is close to 100%. As for the ANN trained by the original DA, it can only achieve 100% accuracy for the balloon dataset which is a simple dataset consisting of only 16 instances.

From Fig. 6, 7, 8, and 9, it can be seen that the convergence rate of the optimized DA is higher as compared to the original DA as the optimized DA converges to the global optimal solution in fewer iterations than the original DA. For example, for the iris dataset in Fig. 6, it can be seen that the optimized DA algorithm converges to the optimal solution at around iteration 15 while the original DA converges at around iteration 18. Moreover, it can be seen that the optimized DA converges to much better solutions than the original DA as the value of the objective function, that is the RMSE of the neural network is much lower.

2) PERFORMANCE COMPARISON OF OPTIMIZED DA AND OTHER SWARM INTELLIGENCE ALGORITHMS IN TRAINING ANN

Tables 7, 8, 9, and 10 show a comparison of the results obtained when the optimized DA and the other swarm intelligence algorithms are used for training the ANN using the iris, balloon, glass, and breast cancer datasets respectively.

In order to have a fair comparison, the optimized DA algorithm is used for training ANNs with the same architecture as the other works in [37], [47], and [48] for the four datasets.

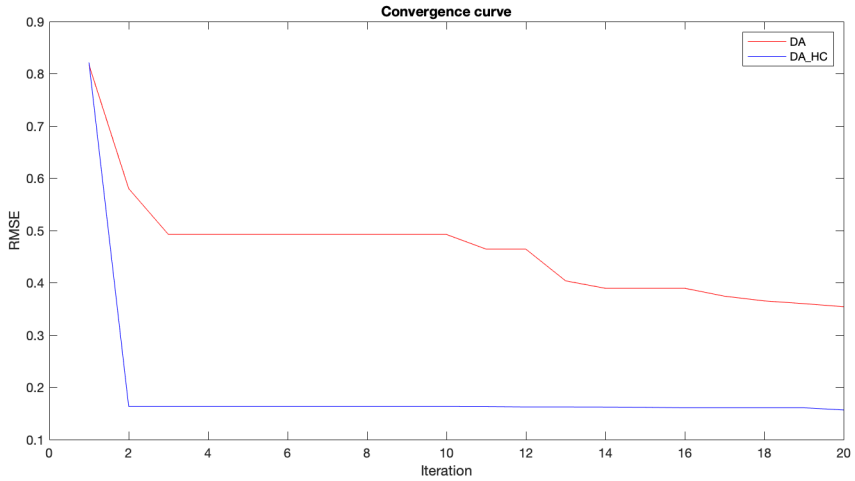

FIGURE 9. Effectiveness and convergence of original DA and optimized DA for breast cancer dataset.

TABLE 7. Performance comparison of optimized DA and other swarm intelligence algorithms in training ANN using iris dataset.

Algorithm	MSE	Accuracy(%)
Optimized DA	1.14E-02	97.78
ACO	4.28E-01	26.33
ALO	1.67E-01	31.00
BAT	2.00E-01	36.00
BBO	3.19E-02	81.25
CS	9.50E-02	49.00
DE	2.25E-01	42.00
EHO	3.49E-01	8.00
ES	3.92E-01	41.00
GA	8.38E-02	59.10
GSA	4.71E-01	51.00
GWO	2.22E-02	87.67
HS	3.19E-01	29.00
MFO	5.76E-02	76.20
MSA	1.32E-01	5.00
PSO	3.12E-01	14.00
SCA	2.09E-01	37.00
WOA	1.94E-01	20.15
HAD	1.73E-02	91.67
MLSSA	4.17E-02	93.33
SSA	5.41E-02	90.67

TABLE 8. Performance comparison of optimized DA and other swarm intelligence algorithms in training ANN using balloon dataset.

Algorithm	MSE	Accuracy(%)
Optimized DA	1.55E-06	100.00
ACO	6.39E-02	80.75
ALO	1.95E-06	100.00
BAT	2.60E-06	100.00
BBO	2.19E-20	100.00
CS	4.21E-09	100.00
DE	1.91E-06	100.00
EHO	2.49E-02	35.00
ES	1.47E-02	91.00
GA	4.23E-17	100.00
GSA	2.51E-02	95.00
GWO	6.87E-24	100.00
HS	2.55E-03	91.00
MFO	1.58E-12	100.00
MSA	8.85E-08	100.00
PSO	3.49E-04	90.33
SCA	1.22E-03	95.15
WOA	2.52E-02	100.00
HAD	1.93E-18	100.00
INMDA	5.48E-16	100.00
PBIL	2.49E-05	100.00
MLSSA	5.67E-10	100.00
SSA	6.39E-05	100.00

For the iris dataset, the following architecture is used: one input layer with four neurons, one hidden layer with nine neurons, and one output layer with three neurons, for the balloon dataset, the following architecture: one input layer with four neurons, one hidden layer with nine neurons, and one output layer with one neuron, for the glass dataset, the architecture used is: one input layer with nine neurons, one hidden layer with 19 neurons, and one output layer with one neuron, and for the breast cancer dataset, the following architecture is used: one input layer with nine neurons, one hidden layer with 19 neurons, and one output layer with one neuron.

The results are compared in terms of the Mean Square Error (MSE) obtained during the training phase and the accuracy obtained during the testing phase.

From Tables 7, 8, 9, and 10 it can be deduced that the proposed optimized DA algorithm has a higher effectiveness as compared to multiple other swarm intelligence algorithms in training artificial neural networks. For the iris, glass, and breast cancer datasets, the ANN trained by the proposed optimized DA achieves a higher accuracy than all other swarm intelligence algorithms used to train ANNs in [37], [47], and [48]. For the balloon dataset, the accuracy obtained by

TABLE 9. Performance comparison of optimized DA and other swarm intelligence algorithms in training ANN using glass dataset.

Algorithm	MSE	Accuracy(%)
Optimized DA	5.74E-01	95.30
ACO	4.22E-01	3.00
ALO	1.65E-01	21.40
BAT	3.34E-01	12.00
BBO	8.27E-03	70.66
CS	6.17E-02	31.00
DE	8.58E-02	3.00
EHO	1.77E-01	0.00
ES	2.43E-01	20.00
GA	9.45E-02	34.20
GSA	2.45E-01	24.00
GWO	1.59E-02	73.25
HS	1.15E-01	32.50
MFO	3.61E-02	72.20
MSA	2.44E-02	0.00
PSO	1.67E-01	1.00
SCA	1.36E-01	23.33
WOA	7.53E-02	10.00
HAD	2.10E-04	77.15

TABLE 10. Performance comparison of optimized DA and other swarm intelligence algorithms in training ANN using breast cancer dataset.

Algorithm	MSE	Accuracy(%)
Optimized DA	0.014799	98.57
ACO	4.45E-01	6.00
ALO	2.26E-01	63.00
BAT	1.08E-01	90.00
BBO	5.30E-02	94.00
CS	8.42E-02	82.25
DE	1.32E-01	7.00
EHO	2.14E-01	0.00
ES	2.44E-01	75.00
GA	6.94E-02	92.25
GSA	3.05E-01	73.00
GWO	4.02E-02	95.00
HS	2.21E-01	72.30
MFO	6.11E-02	94.00
MSA	5.79E-02	7.00
PSO	2.21E-01	7.00
SCA	1.72E-01	77.00
WOA	2.13E-01	74.75
HAD	3.46E-02	96.00
MLSSA	4.19E-03	96.67
SSA	6.32E-03	94.5

the ANN trained by the optimized DA is 100% which is the same as the ANNs trained by several other algorithms.

VIII. CONCLUSION AND FUTURE WORK

The dragonfly algorithm is a recent swarm intelligence algorithm which is inspired by the static and dynamic swarming behaviours of dragonflies. It has been found to have a higher performance than various swarm intelligence algorithms in multiple optimization problems such as in the training of ANNs. Despite having a good performance, DA suffers from a low exploitation phase which affects its effectiveness, that is, its ability to provide high-quality solutions. Hence, its performance can be improved by overcoming its limitations such as the low exploitation phase.

The training of artificial neural networks is a crucial process as it is a requisite step in order to be able to use neural networks. This process primarily allows the neural network to learn how to generate the correct output based on the inputs provided, thus enabling the neural network to be used for various tasks such as classification and regression. Conventional algorithms used for training ANNs such as the Backpropagation algorithm have some limitations such as being trapped in local optima and hence they are unable to find the optimal connection weights for the neural network. Recently, the use of swarm intelligence algorithms to train ANN has been increasing owing to their high exploration and exploitation capabilities.

In this paper, an optimized dragonfly algorithm is proposed and used as a training algorithm for feedforward neural networks which are used for benchmark classification problems, namely the iris, balloon, glass, and breast cancer classification problems. The performance of the dragonfly algorithm is improved by overcoming its low exploitation phase. This is achieved by using the stochastic hill climbing algorithm as a local search technique.

From the experimental results, it can be deduced that the optimized DA algorithm has a better performance in training ANN as compared to the original DA and multiple other swarm intelligence algorithms. The RMSE of the ANNs trained by the optimized DA is found to be lower than the RMSE of the ANNs trained by the original DA for both the training and testing phases. The classification accuracy for the ANN trained by the optimized DA is also higher than the ANN trained by the original DA. Moreover, the ANNs trained by the proposed algorithm have higher accuracy as compared to those trained by multiple other swarm intelligence algorithms.

For future work, the ANN trained by the optimized DA algorithm can be applied to more classification datasets and also to regression datasets so as to use it for regression problems in addition to classification problems. Moreover, it can be used for some real-world applications with real-world datasets instead of benchmark datasets. For example, the ANN trained by the optimized DA can be used as prediction systems in smart cities, and for channel estimation in optical systems [49].

REFERENCES

[1] M.-C. Yuen, S.-C. Ng, and M.-F. Leung, "A competitive mechanism multi-objective particle swarm optimization algorithm and its application to signalized traffic problem," *Cybern. Syst.*, vol. 52, no. 1, pp. 73–104, Jan. 2021.

[2] M.-C. Yuen, S.-C. Ng, M.-F. Leung, and H. Che, "A metaheuristic-based framework for index tracking with practical constraints," *Complex Intell. Syst.*, pp. 1–16, Dec. 2021.

[3] B. A. S. Emambocus, M. B. Jasser, M. Hamzah, A. Mustapha, and A. Amphawan, "An enhanced swap sequence-based particle swarm optimization algorithm to solve TSP," *IEEE Access*, vol. 9, pp. 164820–164836, 2021.

[4] B. A. S. Emambocus, M. B. Jasser, and A. Amphawan, "A discrete adapted dragonfly algorithm for solving the traveling salesman problem," in *Proc. 5th Int. Conf. Intell. Comput. Data Sci. (ICDS)*, Oct. 2021, pp. 1–6.

[5] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.
[6] T. Chen, N. Kapron, and J. C.-Y. Chen, "Using evolving ANN-based algorithm models for accurate meteorological forecasting applications in Vietnam," *Math. Problems Eng.*, vol. 2020, pp. 1–8, Jun. 2020.
[7] A. A. A. M. Amiruddin, H. Zabiri, S. A. A. Taqvi, and L. D. Tufa, "Neural network applications in fault diagnosis and detection: An overview of implementations in engineering-related systems," *Neural Comput. Appl.*, vol. 32, no. 2, pp. 447–472, Jan. 2020.
[8] J. D. Kechagias, A. Tsiolikas, M. Petousis, K. Ninikas, N. Vidakis, and L. Tzounis, "A robust methodology for optimizing the topology and the learning parameters of an ANN for accurate predictions of laser-cut edges surface roughness," *Simul. Model. Pract. Theory*, vol. 114, Jan. 2022, Art. no. 102414.
[9] K. Ganji and S. Parimi, "ANN model for users' perception on IoT based smart healthcare monitoring devices and its impact with the effect of COVID 19," *J. Sci. Technol. Policy Manage.*, vol. 13, no. 1, pp. 6–21, Feb. 2022.
[10] G. Perveen, P. Anand, and A. Kumar, "Short-term power prediction using ANN," in *Proc. IEEE Int. Conf. Signal Image Process. Appl. (ICSIPA)*, Sep. 2021, pp. 233–237.
[11] A. T. Hoang, S. Nižetić, H. C. Ong, W. Tarelko, V. V. Pham, T. H. Le, M. Q. Chau, and X. Phuong Nguyen, "A review on application of artificial neural network (ANN) for performance and emission characteristics of diesel engine fueled with biodiesel-based fuels," *Sustain. Energy Technol. Assessments*, vol. 47, Oct. 2021, Art. no. 101416.
[12] B. Eren, M. A. Guvenc, and S. Mistikoglu, "Artificial intelligence applications for friction stir welding: A review," *Met. Mater. Int.*, vol. 27, no. 2, pp. 193–219, Feb. 2021.
[13] A. M. Ozbayoglu, M. U. Gudelek, and O. B. Sezer, "Deep learning for financial applications: A survey," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106384.
[14] D. Wu and G. G. Wang, "Causal artificial neural network and its applications in engineering design," *Eng. Appl. Artif. Intell.*, vol. 97, Jan. 2021, Art. no. 104089.
[15] H. Moayedi, M. Mosallanezhad, A. S. A. Rashid, W. A. W. Jusoh, and M. A. Muazu, "A systematic review and meta-analysis of artificial neural network application in geotechnical engineering: Theory and applications," *Neural Comput. Appl.*, vol. 32, no. 2, pp. 495–518, Jan. 2020.
[16] G. Di Franco and M. Santurro, "Machine learning, artificial neural networks and social research," *Quality Quantity*, vol. 55, no. 3, pp. 1007–1025, Jun. 2021.
[17] G. R. Yang and X.-J. Wang, "Artificial neural networks for neuroscientists: A primer," *Neuron*, vol. 109, no. 4, p. 739, Feb. 2021.
[18] A. El-Shahat, *Advanced Applications for Artificial Neural Networks*. Rijeka, Croatia: InTech, 2018.
[19] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: A review of two decades of research," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 97–116, Apr. 2017.
[20] S. Selvaraj and E. Choi, "Survey of swarm intelligence algorithms," in *Proc. 3rd Int. Conf. Softw. Eng. Inf. Manage.*, Jan. 2020, pp. 69–73.
[21] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[22] M. Jain, S. Maurya, A. Rani, and V. Singh, "Owl search algorithm: A novel nature-inspired heuristic paradigm for global optimization," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1573–1582, 2018.
[23] J. Xue and B. Shen, "A novel swarm intelligence optimization approach: Sparrow search algorithm," *Syst. Sci. Control Eng.*, vol. 8, no. 1, pp. 22–34, Jan. 2020.
[24] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[25] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.
[26] G. Dhiman, M. Garg, A. Nagar, V. Kumar, and M. Dehghani, "A novel algorithm for global optimization: Rat swarm optimizer," *J. Ambient Intell. Humanized Comput.*, vol. 12, pp. 8457–8482, Oct. 2020.
[27] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, 2016.
[28] B. A. S. Emambocus, M. B. Jasser, A. Mustapha, and A. Amphawan, "Dragonfly algorithm and its hybrids: A survey on performance, objectives and applications," *Sensors*, vol. 21, no. 22, p. 7542, Nov. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/22/7542
[29] C. M. Rahman and T. A. Rashid, "Dragonfly algorithm and its applications in applied science survey," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–21, Dec. 2019.
[30] B. Selman and C. P. Gomes, "Hill-climbing search," in *Encyclopedia of Cognitive Science*. Chichester, U.K.: Wiley, 2006.
[31] B. A. S. Emambocus and M. B. Jasser, "Towards an optimized dragonfly algorithm using Hill climbing local search to tackle the low exploitation problem," in *Proc. Int. Conf. Softw. Eng. Comput. Syst. 4th Int. Conf. Comput. Sci. Inf. Manage. (ICSECS-ICOCSIM)*, Aug. 2021, pp. 306–311.
[32] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[33] N. Kayarvizhy, S. Kanmani, and V. Uthariaraj, "Ann models optimized using swarm intelligence algorithms," *WSEAS Trans. Comput.*, vol. 13, pp. 501–519, Jan. 2014.
[34] M. Shariati, M. S. Mafipour, P. Mehrabi, A. Bahadori, Y. Zandi, M. N. A. Salih, H. Nguyen, J. Dou, X. Song, and S. Poi-Ngian, "Application of a hybrid artificial neural network-particle swarm optimization (ANN-PSO) model in behavior prediction of channel shear connectors embedded in normal and high-strength concrete," *Appl. Sci.*, vol. 9, no. 24, p. 5534, Dec. 2019, doi: 10.3390/app9245534.
[35] H. Moayedi, H. Nguyen, and L. K. Foong, "Nonlinear evolutionary swarm intelligence of grasshopper optimization algorithm and gray wolf optimization for weight adjustment of neural network," *Eng. Comput.*, vol. 37, no. 2, pp. 1265–1275, Apr. 2021.
[36] H. Moayedi, M. M. Abdullahi, H. Nguyen, and A. S. A. Rashid, "Comparison of dragonfly algorithm and Harris hawks optimization evolutionary data mining techniques for the assessment of bearing capacity of footings over two-layer foundation soils," *Eng. Comput.*, vol. 37, no. 1, pp. 437–447, Jan. 2021.
[37] J. Xu and F. Yan, "Hybrid Nelder–Mead algorithm and dragonfly algorithm for function optimization and the training of a multilayer perceptron," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 3473–3487, Apr. 2019, doi: 10.30526/31.1.1834.
[38] A. T. Abdulameer, "An improvement of MRI brain images classification using dragonfly algorithm as trainer of artificial neural network," *Ibn AL-Haitham J. Pure Appl. Sci.*, vol. 31, no. 1, p. 268, May 2018. [Online]. Available: http://jihcoed.com/ihj/index.php/j/article/view/1834
[39] M. Khishe and A. Safari, "Classification of sonar targets using an MLP neural network trained by dragonfly algorithm," *Wireless Pers. Commun.*, vol. 108, pp. 1–20, May 2019.
[40] A. Lim, J. Lin, B. Rodrigues, and F. Xiao, "Ant colony optimization with Hill climbing for the bandwidth minimization problem," *Appl. Soft Comput.*, vol. 6, no. 2, pp. 180–188, Jan. 2006.
[41] L. Abualigah, M. Shehab, A. Diabat, and A. Abraham, "Selection scheme sensitivity for a hybrid SALP swarm algorithm: Analysis and applications," *Eng. Comput.*, vol. 38, pp. 1149–1175, Jul. 2020, doi: 10.1007/S00366-020-01067-Y.
[42] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "University course timetabling using hybridized artificial bee colony with Hill climbing optimizer," *J. Comput. Sci.*, vol. 5, no. 5, pp. 809–818, Sep. 2014.
[43] M. Shehab, H. Alshawabkah, L. Abualigah, and N. Al-Madi, "Enhanced a hybrid moth-flame optimization algorithm using new selection schemes," *Eng. Comput.*, vol. 37, no. 4, pp. 2931–2956, Feb. 2020, doi: 10.1007/S00366-020-00971-7.
[44] M. Shehab, A. T. Khader, M. A. Al-Betar, and L. M. Abualigah, "Hybridizing cuckoo search algorithm with Hill climbing for numerical optimization problems," in *Proc. 8th Int. Conf. Inf. Technol. (ICIT)*, May 2017, pp. 36–43. [Online]. Available: http://ieeexplore.ieee.org/document/8079912/
[45] M. Alzaqebah and S. Abdullah, "An adaptive artificial bee colony and late-acceptance Hill-climbing algorithm for examination timetabling," *J. Scheduling*, vol. 17, no. 3, pp. 249–262, Jun. 2014.
[46] A. Lim, J. Lin, and F. Xiao, "Particle swarm optimization and Hill climbing for the bandwidth minimization problem," *Int. J. Speech Technol.*, vol. 26, no. 3, pp. 175–182, Jun. 2007.
[47] W. A. H. M. Ghanem and A. Jantan, "A cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for training multi-layer perceptrons," *Cognit. Comput.*, vol. 10, no. 6, pp. 1096–1134, Dec. 2018.
[48] D. Bairathi and D. Gopalani, "Numerical optimization and feed-forward neural networks training using an improved optimization algorithm: Multiple leader SALP swarm algorithm," *Evol. Intell.*, vol. 14, no. 3, pp. 1233–1249, Sep. 2021.
[49] B. A. S. Emambocus, M. B. Jasser, and A. Amphawan, "Towards an optimized channel estimation in optical spatial multiplexing systems via swarm intelligence algorithms," in *Proc. IEEE 13th Control Syst. Graduate Res. Colloq. (ICSGRC)*, Jul. 2022, pp. 77–82.

BIBI AAMIRAH SHAFAA EMAMBOCUS received the B.Sc. degree in computer science from Sunway University, Malaysia, in 2020, where she is currently pursuing the M.Sc. degree in computer science. Her research interests include swarm intelligence, evolutionary algorithms, artificial intelligence, and machine learning.

MUHAMMED BASHEER JASSER (Member, IEEE) received the master's and Ph.D. degrees in software engineering from University Putra Malaysia (UPM). He was granted the Malaysian Technical Cooperation Program Scholarship (MTCP) from the Ministry of Higher Education (Malaysia) for his postgraduate studies. He is currently a Program Leader and a Senior Lecturer at Sunway University. Prior to that, he was a Research Assistant at UPM and a Lecturer at Aleppo University. He is also working on several fundamental and industrial research projects in the area of artificial intelligence and software engineering funded by companies and universities. His research interests include optimization algorithms, evolutionary computation, model-driven software engineering, formal specification, verification and theorem proving, artificial intelligence, and machine learning. He is a member of several professional academic bodies including the Institute of Electronics, Information and Communication Engineers (IEICE), and Formal Methods Europe.

ANGELA AMPHAWAN (Member, IEEE) received the Ph.D. degree in optical engineering from the University of Oxford, U.K. She is currently Leading the Photonics Research Laboratory, School of Engineering and Technology, Sunway University. Prior to this, she was the Deputy Vice Chancellor at the University Malaysia of Computer Science and Engineering. Her research projects have been funded by the U.S. Department of States, German Government, Malaysian Ministry of Education and Telekom Malaysia. Her research interests include optical communications and sensing, covering optical fibers, free-space optics, radio-over free space optics, digital imaging, and the Internet-of-Things. She was a recipient of the Fulbright Award at the Research Laboratory of Electronics, Massachusetts Institute of Technology. She has also won several best paper awards and exhibition medals. She was a Publicity Co-Chair for the IEEE Wireless Communications and Networking Conference. She serves on the Editorial Board for the *APL Photonics* Journal with the American Institute of Physics and is on the National 5G Task Force. She was previously on the Editorial Board of Wiley *Transactions on Emerging Telecommunications Technologies*. She has given keynote addresses at several Fulbright and IEEE events.