Optimizing Weights And Biases in MLP Using Whale Optimization Algorithm

HARIT, ANOUSHKA

Optimizing Weights And Biases in MLP Using Whale Optimization Algorithm

A thesis presented for the degree of

Master by Research

Department of Computer Science

The University of Durham

United Kingdom

February 2022

Whale Optimization

Anoushka Harit

Abstract

Artificial Neural Networks are intelligent and non-parametric mathematical models inspired by the human nervous system. They have been widely studied and applied for classification, pattern recognition and forecasting problems. The main challenge of training an Artificial Neural network is its learning process, the nonlinear nature and the unknown best set of main controlling parameters (weights and biases). When the Artificial Neural Networks are trained using the conventional training algorithm, they get caught in the local optima stagnation and slow convergence speed; this makes the stochastic optimization algorithm a definitive alternative to alleviate the drawbacks. This thesis proposes an algorithm based on the recently proposed Whale Optimization Algorithm(WOA). The algorithm has proven to solve a wide range of optimization problems and outperform existing algorithms. The successful implementation of this algorithm motivated our attempts to benchmark its performance in training feed-forward neural networks. We have taken a set of 20 datasets with different difficulty levels and tested the proposed WOA-MLP based trainer. Further, the results are verified by comparing WOA-MLP with the back propagation algorithms and six evolutionary techniques. The results have proved that the proposed trainer can outperform the current algorithms on the majority of datasets in terms of local optima avoidance and convergence speed.

Supervisors: First Supervisor and Second Supervisor

Acknowledgements

Foremost, I would like to thanks my supervisor, Prof. Iain.A Stewart, for his infinite patience,kindness and invaluable guidance throughout my studies. He is, without doubt, the most influential person in my research career, both in his teaching and in his research. I would also like to extend my thanks to Dr. Noura Al Moubayed for her extreme support and for being a inspiration to me.

Besides, I would also like to thank the thesis committee Dr Kathleen Steinhoefel and Professor Alexandra Cristea , for taking their valuable time to examine this Master by Research work. I am also very grateful to my fellow lab mates for the endless discussions and for the great time when we were pushing the deadlines. More specifically, I would like to thank Mr.Zhongtian Sun and Mr. Jialin Yu for the being my strength during the tough times and taking me forward in the field of research . I am also grateful to Dr. Neelanjan Bhowmik for helping me resolve technical issues.I would also like to say thank you to my Grandmother,Mum and Dad, for always believing in me and showing their support even from 4,758 miles away. I very much appreciate my Dad's help, who has been a great mentor for me throughout my entire life.

Declaration

The work in this thesis is based on research carried out within the Algorithm and Complexity ,Innovative Computing Group at the Department of Computer Science at Durham University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all the author's own work unless referenced to the contrary in the text.

Publications

Relevant Publications

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scientists have widely used and applied ANN for prediction modelling, forecasting problems, and engineering fields. They have been more favoured due to their flexibility in performing various tasks over the conventional algorithms. The efficiency of an ANN is highly affected by its parameters and the learning process.ANN models are robust nonlinear modelling techniques, and they facilitate links between input and output variables via allocated weights and activation functions (Mohammadi (2020)). The most commonly used and applied ANN is the multi-layer perceptron (MLP) neural network.MLPs are based on the human nervous system and are highly compatible in solving the nonlinear behaviour of complex systems. There are two main categories of supervised training methods: gradient-based and stochastic. At the same time, the back-propagation algorithm and its variant (Wang et al. (2015)) are considered standard examples of gradient-based methods and the most popular among researchers. The main drawbacks of the gradient-based algorithm are that they get into the loop of getting stuck in local minima, slow convergence speed, and they are too dependent on the initial parameters (Faris et al. (2016);Mirjalili (2015)). The ideal alternative to gradient-based methods is the heuristic search algorithms; this thesis has proposed to optimise the MLP neural networks. When compared with the gradient methods, metaheuristics have shown higher confidence in avoiding local minima (Črepinšek et al. (2013);Mirjalili et al. (2012)). Meta-heuristics can be divided into two prominent families known as evol-

utionary and swarm-based algorithms. These algorithms are population-based, in which a number of possible random solutions are generated, evolved and updated until we reach a satisfactory result or reach the maximum no of iterations. This method is the most applied and investigated method by researchers recently to train the MLP networks as these methods incorporate randomness to move from local to global search, which makes them more suitable for global optimisation (Yang (2014)). In the supervised learning of MLP networks, evolutionary algorithms were used in three primary schemes: automatic design of the network topology, optimisation of the network's link weights and biases, and evolution of the learning rules (Yu et al. (2008)). It is important to note that optimising the topology and weights of the MLP network at the same time can significantly expand the number of parameters, making it a large-scale optimisation task (Karaboga (2005)).In this thesis, we solely optimise the link weights and biases in the MLP network.

The genetic algorithm (GA) is a classic evolutionary algorithm and is one of the most studied meta-heuristics in neural network training.GA is based on Darwinian theories of evolution and natural selection, and it was proposed first by (Holland et al. (1992)); Goldberg (1989)) and Sastry et al. (2005)). The author of (Seiffert (2001)) used GA to train the link weights in an MLP network and concluded that GA outperforms the back-propagation technique when the targeted problems are more complex. Swarm-based stochastic search algorithms, inspired by the motions of birds, insects, and other organisms in nature, are another notable type of meta-heuristics gaining popularity. These algorithms use mathematical models to update the generated random solutions rather than reproduction operators like those used in GA. Particle swarm optimization (PSO) (Zhang et al. (2015);Sammut and Webb (2011)), ant colony optimization (ACO) (Dorigo et al. (2006) and the artificial bee colony(Karaboga et al. (2014);Karaboga (2005)) are the most well-known examples of swarm-based techniques. These algorithms have been applied to train Multi-layer Perceptron networks but the problems have been reported in (Mendes et al. (2002);Blum and Socha (2005);Karaboga et al. (2007)). Hybrid Algorithms have

been recently introduced to enhance the weights and biases of the MLP. The hybrid algorithms have been employed to train the MLP to resolve the issue of trapping in local minima and slow convergence rate. Using metaheuristics for training different types of neural networks has been investigated by (Kolay et al. (2016)).

Previously, many evolutionary and swarm-based algorithms have been deployed and investigated in the literature for training MLP, but the problem of local minima persists. Motivated by the following reasons, we have presented the MLP training method based on the recently proposed Whale Optimization Algorithm (WOA) for training a single hidden layer neural network. WOA is a meta-heuristic algorithm that was introduced and developed by (Mirjalili and Lewis (2016)) and is inspired by the bubble-net hunting strategy of humpback whales. Unlike the previous works, the WOA-based approach is tested and evaluated on 20 popular classification datasets. Also, the results are compared to those previously introduced trainers in the literature(GA, DE,ES and the population-based incremental learning algorithm(PBIL))(Baluja (1994);Meng et al. (2014)) , two swarm intelligent algorithms (PSO and ACO) Zhang et al. (2015); Sammut and Webb (2011) and the most popular gradient-based back-propagation algorithm Li et al. (2012).

1.1 Thesis Structure and Contribution

This thesis is organized as follows: A literature review and introduction to the meta-heuristics algorithm, Optimizing MLP using Meta-heuristics, and finally present the Whale Optimization Algorithm followed by the experimental evaluation. Further, we discuss the motivation for the research and the important terms we have used in the proposed work.

1. We have implemented the Whale Optimization Algorithm (WOA) to optimize weight and biases in Multi-layer Perceptron in the presented work. We have tested them on a set of 20 datasets with different levels of difficulty. The

results of the introduced WOA-MLP trainer are verified by comparisons with the back-propagation algorithm and six evolutionary techniques.

2. We have also re-implement the work presented by Aljarah et al. (2018) in **'Optimizing Connection and Weights in neural networks using the Whale Optimization Algorithm'**. We use python as the programming language to implement the computational experiment, observe the behaviour of all the metaheuristics and compare it with our WOA-MLP model.

3. The qualitative and quantitative results prove that the proposed trainer is able to outperform the current algorithms on the majority of datasets in terms of both local optima avoidance and convergence speed.

1.1.1 Research Questions ?

We have intensively researched , studied and applied Whale Optimization to optimize weight and biases in the thesis , based on our research goals we have concluded the following research questions.

1. Will Whale Optimization Algorithm prove to be a reliable alternative as a trainer for MLP?

2. Can Whale Optimization perform better than other Swarm Based or Evolutionary Algorithm ?

3. Will Whale Optimization Algorithm be able to achieve the global optimum for the 20 Classification dataset in our experiment?

Keywords: Optimization, Multi-layer Perceptron, Meta-heuristics , Weights , Biases and Evolutionary algorithm

Chapter 2

Literature Review:

The FNN optimization is often viewed from the various perspectives: the optimization of weights, network architecture, activation nodes, learning parameters, learning environment, etc. Researchers adopted such different viewpoints mainly to improve the FNN's generalization ability. The gradient-descent algorithm such as back-propagation Li et al. (2012) has been widely applied to optimize the FNN's. Its success is evident from the FNN's application to numerous real-world problems. However, due to the limitations of the gradient-based optimization methods, the metaheuristic algorithms including the evolutionary algorithms, swarm intelligence, etc., are still being widely explored by the researchers aiming to obtain generalized FNN for a given problem.

The gradient-descent based algorithms Lopes and Ribeiro (2003) are also known as local search algorithms. They are good at exploiting the obtained solutions to find new solutions. However, to find a global optimum solution, any optimization algorithm must use two techniques: (1) exploration-to search new and unknown areas in a search space and (2) exploitation-to take advantage of the already discovered solution (March, 1991). The exploration and exploitation are two contradictory strategies and a good search algorithm must find a trade-off between these two. Metaheuristic is the procedure that implements nature-inspired heuristics to combine these two strategies Wolpert and Macready (1997). Hence, metaheuristic

approaches are alternative to the conventional approaches for optimizing the FNNs.

Unlike the conventional methods, which require the objective function to be continuous and differential, the metaheuristic algorithms have the ability to address complex, nonlinear, and non-differentiable problems. However, the optimization algorithms are often biased towards a specific class of problems, that is, "there is no such universal optimizer which may solve all class of problem," which is evident from no free lunch theorem Wolpert and Macready (1997).

Heuristic and metaheuristic algorithms are used to solve complex optimization problems, and both provide inexactly or close to optimal solutions. Heuristic algorithms follow algorithmic type procedures and are deterministic. Heuristic algorithms solve a specific problem and are purposeless for minor and polynomial-time issues. Metaheuristic algorithms (MA's) are beyond the next level of heuristic algorithms, based on guided random search techniques and nature-inspired. Metaheuristic algorithms utilize randomization and deterministic approach to solve various global optimization problems. Metaheuristic algorithms are robust to global search solutions due to derivation free structure, avoid entrapment into local optima, randomly generated population in the search space, and utilize biological or physical spectacles. Metaheuristic randomly generated population or multiple solution based metaheuristic explores search space more efficiently than a single solution based metaheuristic and is categorized into evolution-based, Physics-based and Swarm based algorithms. There are no assumptions for these methods with regard to the basic fitness landscape. Genetic Algorithm (GA) Holland et al. (1992) , that is a category of well-known optimization methods, uses the theory of Darwinian evolution. Bio-geography based optimization (BBO) (Bhattacharya and Chattopadhyay (2010))algorithm is a new kind of optimization technique based on bio-geography concept. Differential evolution (DE)(Opara and Arabas (2019)) is one of the most popular and efficient evolutionary algorithms for numerical optimization and it have gained much success in a series of benchmark functions as well as real applications.

Algorithms that are Physics-based are the second group of algorithms. In this type, each search agent can interact and move at the search space in accordance with some physics rules. The rules of gravitational force, inertia force, electromagnetic force, and the others can be mentioned. As an example, Simulated Annealing (SA) (Ingber (1989)) was inspired by the metallurgic annealing process, Ray Optimization (RO) by the Snell's light refraction law (Kaveh and Khayatazad (2012)), Electromagnetic Field Optimization (EFO) (Abedinpourshotorban et al. (2016)), and Gravitational Search Algorithm (GSA) (Yin et al. (2011)) by the law of gravity and mass interactions.

The last division is swarm-based algorithms driven from the collective behavior of social creatures (Pacini et al. (2014)).This collective intelligence is based on swarm interactions with each other. It is easier to implement swarm-based algorithms than the evolutionary based algorithms because of including the lower number of operators (i.e., selection, crossover, mutation). Furthermore, swarm-based algorithms have some advantages examples of which are as follows:

1. The information about the search space can be maintained by swarm-based algorithms at the time of iterations whereas the information of the previous generations can be maintained by evolutionary-based algorithms can.

2. The evolutionary techniques have more input parameters as compared to swarm-based algorithms.

3. Memory space is less utilized by swarm-based algorithms so as to save the best optimal solutions.

Meta-heuristics are high quality methods guiding the search agents to gradually enhance the whole solution Pacini et al. (2014).Henceforth, the optimization of the solution is carried out by checking across some random elements and possibilities, hoping to find a more qualified solution, whereas the candidate solution is inherited from one stochastic iteration to another.

2.1 Overview of Some Important Terms

In this section we discuss the important keywords that we have used and will be defining in the the proposed . The important terms are presented in the form of section explaining in brief about their application in the presented work.

2.1.1 What is Optimization Problem ?

The optimization Problem can be defined as computational problem in which the objective is to find the best of all possible solutions or the problem of selecting the best solution from all viable alternatives, given restrictions defining which solutions are feasible and a target function indicating which solution is the best.Maximizing or minimizing functions are used to determine the problem and the functions are also relative to some set to represent a range of choices available in a certain situation . The function allows the comparison of different choices to determine which is the best choice. The optimization problem help us analyse how can a problem be categorized and characterized , also find the best solution after all the perturbation.

An optimization problem can be described as a finite set of variables, with the optimal solution specified by the correct values for the variables. The issue is considered continuous if the variables have a range of real numbers, and combinatorial if they can only take a finite set of different values.

2.1.2 What is the meaning of local and global optima in a optimization problem?

Optimization refers to finding the set of inputs to an objective function that results in the maximum or minimum output from the objective function.It is common to describe optimization problems in terms of local vs. global optimization.A local minimum of a function is a point where the function value is smaller than at nearby

points, but possibly greater than at a distant point whereas the global minimum is a point where the function value is smaller than at all other feasible points.

2.1.3 What is Convergence in Optimization Problem ?

Convergence refers to the limit of a process and can be a useful analytical tool when evaluating the expected performance of an optimization algorithm. It is a useful empirical tool when exploring the learning dynamics of an optimization algorithm and machine learning algorithms trained using an optimization algorithm, such as deep learning neural networks. This motivates the investigation of learning curves and techniques, such as early stopping.

We know that optimization is a process that generates candidate solutions. The convergence represents a stable point at the end of the process when no further changes or improvements are expected. Premature convergence refers to a failure mode for an optimization algorithm where the process stops at a stable point that does not represent a globally optimal solution.

2.1.4 What are Weights and Biases ?

Weights and biases (abbreviated w and b) are learnable parameters of some machine learning models, such as neural networks.Neurons are the fundamental building blocks of a neural network. Each neuron in a layer of an ANN is connected to some or all of the neurons in the next layer. When inputs are transferred across neurons, the weights and bias are applied to the inputs.

$$Y = \sum \text{ (weight * input) + bias} \tag{2.1}$$

Weights: The signal (or the strength of the link) between two neurons is controlled by weights. In other words, a weight determines the amount of influence the input has on the output.

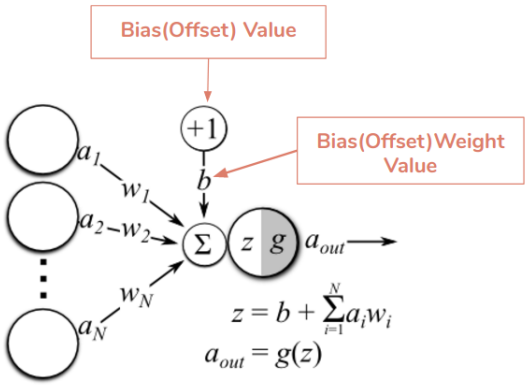


Fig 2.1.5: Weights and Bias

Bias: Bias are constant and also the additional input in the next layer that always have the value of 1 .The previous layer has no influence on bias units (they have no incoming connections), but they do have outbound connections with their own weights. The bias unit ensures that even if all of the inputs are zeros, the neuron will still be activated.

2.1.5 What is Heuristics and how is it different from meta-heuristics?

A heuristic method is a strategy for solving problems that derives from the ancient Greek word 'eurisko,' which means to 'find,"search,' or 'discover.' It is about employing a practical method that does not have to be perfect. Heuristic methods shorten the time it takes to find a good solution.

It is frequently required to adopt algorithms that do not guarantee an optimal solution when working with NP-hard situations.This class of algorithms is known as heuristics. A heuristic is a "intuitive" method for finding a valid and often reasonably good solution to a given problem in a "reasonable" amount of time, i.e. a heuristic is based on "rules-of-thumb," ideas that appear to be helpful in some typical instances but do not provide any guarantee of the quality of the solution.

Heuristics methods are most feasible when time is an essential factor. The heuristics methods are methods may not lead up to optimal and ideal solution , but it allows to speed up the decision making process to achieve an adequate solution in short term. Heuristics methods are always confused with metaheuristics , when the randomization is added as a feature to heuristics it speeds up the process and gives the exact optimal solution we are looking for , this is known as the metaheuristics. A meta heuristic is a general purpose algorithmic framework which can be applied to different optimisation problems. It guides and modifies heuristics to efficiently produce good quality solution.

2.1.6 What are Meta-heuristic Algorithm ?

The term "meta-heuristics" (Glover (1986)) is composed of two Greek words , the prefix "meta" meaning "beyond" and "heuristic" means "discover". The meta-heuristics performs better than simple heuristic, the metaheuristics algorithm use trade off and global exploration. Randomization and global exploration is known as the best qualities of the meta-heuristics, as it provides a good way to move away from local search to the search in global scale. Therefore , most of the metaheuristics algorithms are suitable for nonlinear and global optimization.

Meta-heuristics are high level procedures to efficiently utilize heuristics to find the best the optimal solution to complex optimization problems (Blum and Roli (2003)). The meta-heuristics can be a most efficient way to produce most acceptable solutions by trial and error to a complex problem in a reasonably practical time . In the presented work we use nature-inspired metaheuristics solve optimization problems by mimicking biological or physical phenomenon. They can be divided into four main categories : evolution-based, physics-based, and swarm-based methods. Evolution-based methods are inspired by the laws of natural evolution. The search begins with a randomly created population that evolves over several generations. The strength of these methods is that the best individuals are always

combined to generate next best generation of individuals. This enables the population to be optimised over the next course of generations.

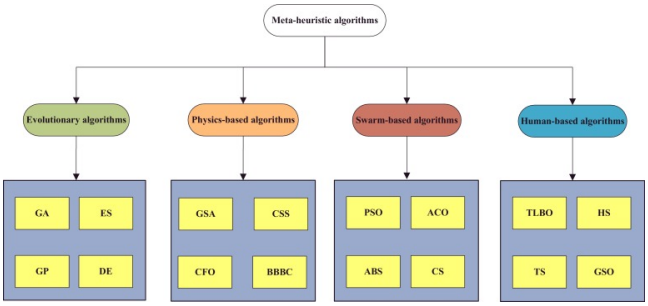


Fig 2.1.6: Nature-Inspired Meta-heuristics Classification

Meta-heuristic optimization refers to a wide range of optimization techniques that require only the appropriate objective function as well as important specifications such as variable boundaries and parameter values. These algorithms can find the near-optimal, or even optimal, values of that objective function.Some Meta-heuristics algorithms are nature inspired as they imitate processes in the natural systems (Blum et al. (2011)). Meta-heuristics have two main components as : intensification and diversification or exploitation and exploration , the diversification means to generate the diverse solutions to search the local space on the global scale , while the intensification means to search in the local region by exploiting the information to find the best optimal solution in the region.

2.1.7 What is Evolutionary Algorithm ?

Evolutionary algorithm can be described as the efficient heuristics methods based on Darwinian evolution with powerful features of robustness and flexibility to capture global solution to complex optimization problems. When Evolutionary algorithms are used in the problems the probability of finding a near optimum solution in a optimization process is distinctively high. An evolutionary algorithm follows five overall step.

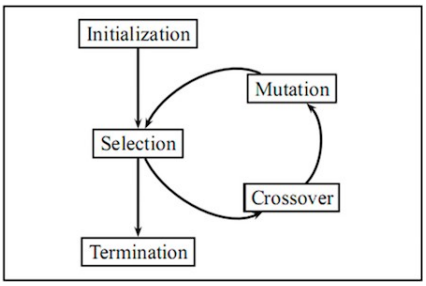


Fig 2.1.7: Phases of Evolutionary Algorithm

Evolutionary Algorithm states that fittest member will survive and proliferate , while unfit members will not contribute to gene pool for further generations

EAs are used to solve situations where standard exploitative or pure stochastic algorithms fail or struggle to find a solution. The struggle due to constraint on the resources , high number of dimensions or complex functionality.

2.1.8 What is Neural Network?

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

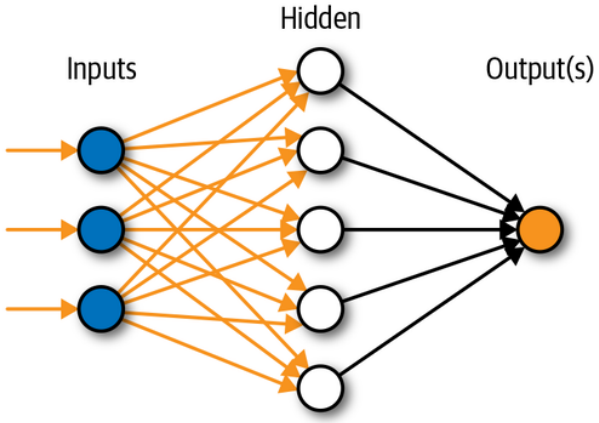


Fig.2.1.8: Artifical Neural Network

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well-known neural networks is Google's search algorithm.

2.1.9 How does a Neural Network Work ?

In this section we discuss about how a Neural Network operates with a hidden layers, an input and output layer. Neural networks are composed of layers of computational units (neurons), with connections among the neurons in different layers.Each neuron in a network transforms data using a series of computations: a neuron multiplies an initial value by some weight, sums results with other values coming into the same neuron, adjusts the resulting number by the neuron's bias, and then normalizes the output with an activation function. The bias is a neuron-specific number that adjusts the neuron's value once all the connections are processed, and the activation function ensures values that are passed on lie within

a tunable, expected range.This process is repeated until the final output layer can provide scores or predictions.

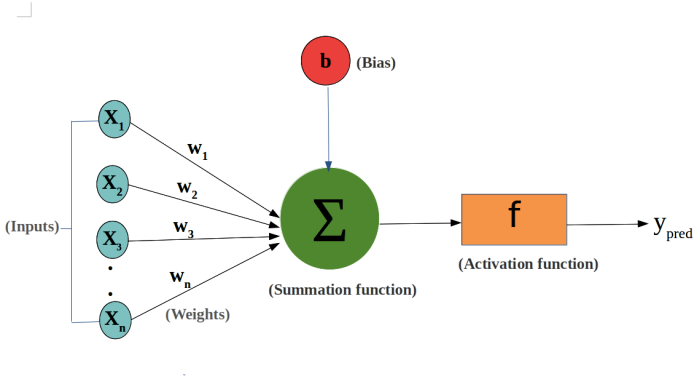


Fig 2.1.9: Neural Network with Weight and Bias

When a neural network is trained on the training set, it is initialised with a set of weights. These weights are then optimised during the training period and the optimum weights are produced. A neuron first computes the weighted sum of the inputs.

$$Y = \sum(\text{ weight } * \text{ input }) + \text{ bias} \tag{2.2}$$

As an instance, if the inputs are:

$$x_1, x_{2,}, \ldots, x_n \tag{2.3}$$

And the weights are:

$$w_1, w_{2.}, \ldots, w_n \tag{2.4}$$

Then a weighted sum is computed as:

$$x_1 w_1 + x_2 w_2 + \cdots + x_n w_n \tag{2.5}$$

Subsequently, a bias (constant) is added to the weighted sum

$$x_1 w_1 + x_2 w_2 + \cdots + x_n w_n + \text{ bias} \tag{2.6}$$

Bias is simply a constant value (or a constant vector) that is added to the product of inputs and weights. Bias is utilised to offset the result. The bias is also used to shift the result of activation function towards the positive or negative side.

Once an input layer is determined, weights are assigned. These weights help determine the importance of any given variable, with larger ones contributing more significantly to the output compared to other inputs. All inputs are then multiplied by their respective weights and then summed. Afterward, the output is passed through an activation function, which determines the output. If that output exceeds a given threshold, it "fires" (or activates) the node, passing data to the next layer in the network. This results in the output of one node becoming in the input of the next node. This process of passing data from one layer to the next layer defines this neural network as a feed-forward network.

2.1.10 Multi-layer Perceptron Network

A feed-forward artificial neural network that creates a set of outputs from a set of inputs is known as a multi-layer perceptron (MLP).An MLP is defined by several layers of input nodes that are linked as a directed graph between the input and output layers. Each node except from the input nodes have the nonlinear activation function. MLP is also a deep learning technique as it has multilayer of neurons. The Multi-Layer Perceptron (MLPs) breaks the restriction of XOR problem and classifies datasets which are not linearly separable. They do this by using a more robust and complex architecture to learn regression and classification models for difficult datasets.

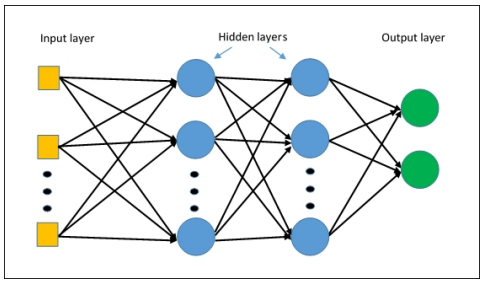


Fig 2.6: Multi-layer Perceptron Network

Multi-layer Perceptron is the basis for all neural networks and the concept of MLP has greatly improved the power of computers when applied to classification and regression problem.

2.2 Conclusion

In the following we have discussed Feed Forward Neural Network Optimization using Meta-heuristics algorithm . We have studied Metaheuristic in details explaining what is definition of meta-heuristic and how it is different from heuristic algorithm. We have also defined all the important terms that are used in optimization problem and will be frequently used in the thesis.In the next chapter we address hoe Neural Network faces the issue of training and How meta-heuristic help in solving it .

Chapter 3

Optimizing MLP using Meta-heuristic Approach

The basic form of MLP optimization is searching its weights (free parameters of MLP) such that the cost function or similar function can be minimized. We believe that the goodness (performance) of FNN depends on multiple factors such as optimized weight, learning factor ..etc. This section addresses using a meta-heuristic algorithm to train feed-forward neural networks. The meta-heuristics algorithm does not guarantee a global optimal solution, but it can offer a near-optimal (satisfactory) solution.

3.1 Formulating Feed-Forward Neural Network Training Issue

This section describes and formalizes the optimization problem which arises from the training of a FNN. In a FNN, also known as multilayer perceptron (MLP), the network topology is composed by the input layer, a set of hidden layers and, finally, the output layer. The connections between the neurons from different layers are always forward, and usually, all the neurons of one layer are connected to all neurons from the next layer. A general scheme of a FNN or MLP is shown in Fig.

3.1. In the figure, n_j is the number of neurons in the jth layer and the parameter c denotes the total number of layers in the architecture.

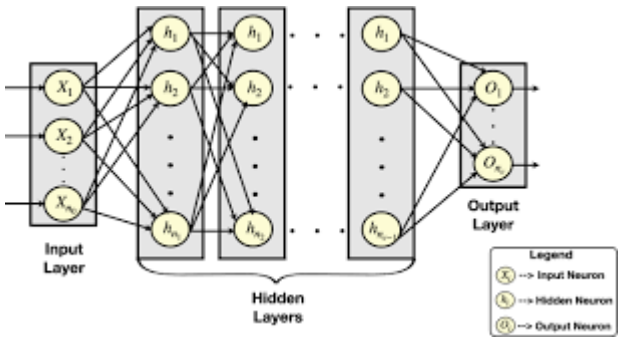


Fig 3.1: Structure of Feed-Forward Neural Network

In every FNN or MLP, the connections between the neurons from different layers are associated with a real number, the so-called weight of the connection. The weight of a connection expresses the synaptic power of a given connection, which can be excitable (positive sign) or inhibitory (negative sign). Furthermore, each neuron in the topology has an internal threshold (b). It is used as a comparison factor to produce the network's output and activate or not a neuron in the topology. There are two main steps in the training process of an FNN. First, the inputs of the network are propagated forward in order to obtain an output. Next, the error between the output produced by the network and the desired value is computed. Finally, in a second step, the errors are propagated backwards, adjusting the weights and thresholds for each neuron in the topology to minimize an error or loss function. This last step will be carried out by using an optimization method.

We consider a canonical FNN with fully connected layers. The first is the input layer, the following layers are hidden layers, and the last is the output layer. The sizes of the layers are n_j for $j = 1, \ldots, c$. Let $W_j \in R^{n_j \times n_{j-1}}$ be the weight matrix associated with the connections from layer $j-1$ to layer j, and let $b_j \in R^{n_j}$ be the threshold vector of the neurons from layer j for layers $j = 1, \ldots, c$. The process of propagating the inputs of the network forward is the following:

1. Computing the activation for the neurons from the input layer. Given an input vecto $x_i \in \mathbf{R}^{n_0}$ for the FNN, the neurons of the input layer only transmit the received sign to the next layer, so the activation of the $x_i^{(0)}$ is computed using Eq. (3.1).

$$x_i^{(0)} = x_i \tag{3.1}$$

2. Computing the activation for the neurons from the hidden layer. The neurons from the hidden layer process the information received from the neurons of the input layer, applying the activation function to the weighted sum of the activation received. FNN applies successive transformations to the given input x_i by means of Eq. (3.2).

$$x_i^{(j)} = s\left(W_j \cdot x_i^{(j-1)} + b_j\right) \in R^{n_j}, \quad j = 1, \ldots, c-1 \tag{3.2}$$

where the vector $b_j \in \mathbf{R}^{n_j}$ contains the j th layer parameters (biases) and s is a component-wise nonlinear activation function. The activation function of a neural network includes many alternatives such as sigmoid, hyperbolic tangent, ReLU (rectified linear unit), Leaky ReLU and Softmax. The activation function is necessary to avoid the linearity of the computation since, if a set of neurons are each linked by a process, but without applying a nonlinear activation function; then, the computation of those neurons could have been performed by only one of them. Thus, to provide suitable computing capacity to a neural network it is necessary to establish a nonlinear relationship between the input of each neuron and its output. The most widely used activation functions s of FNNs include the sigmoid function $s(x) = 1/(1+\exp(-x))$ and the ReLU function $s(x) = \max\{0, x\}$ or the hyperbolic tangent $s(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. In this paper, the sigmoid function has been chosen. Hence, Eq. (3.2) can be rewritten as:

$$x_i^{(j)} = \frac{1}{1+\rho^{-\left(W_j \cdot x_i^{(j-1)} + b_j\right)}}, \quad j = 1, \ldots, c-1 \tag{3.3}$$

3. Computing the activation for the neurons from the output layer. The last step is to compute the activation of the neurons from the output layer in the same way as it was made in the hidden layer. However, in this case, the activation of the neurons will be the output of the network o_i, as it is shown in Eq. (3.4).

$$o_i = x_i^{(c)} = s\left(W_c \cdot x_i^{(c-1)} + b_c\right) \in R^{n_c} \tag{3.4}$$

where o_i is the output vector provided by the network for the input vector x_i.

This process in which the neural network obtains the output vector o_i for an input vector x_i, where the parametrization of the network $\boldsymbol{\omega} = \{W_j, b_j\}_{j=1}^{c}$ is known, can be schematically represented by $o_i\left(x_i, \boldsymbol{\omega}\right)$. Without loss of generality, it can be considered that $\boldsymbol{\omega}$ is a parameter vector $\boldsymbol{\omega} \in R^n$.

The training problem consists of determining the parameter vector $\boldsymbol{\omega}$ and involves a training dataset $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ and the choice of a loss function ℓ, obtaining the resulting optimization problem:

$$\underset{\omega \in R^n}{\text{Minime}} \frac{1}{N} \sum_{i=1}^{N} \ell\left(o_i\left(x_i, \boldsymbol{\omega}\right), y_i\right) \tag{3.5}$$

A popular choice for the loss function ℓ is MSE

$$MSE = \ell\left(o_i\left(x_i, \boldsymbol{\omega}\right), y_i\right) = \left\|y_i - o_i\left(x_i, \boldsymbol{\omega}\right)\right\|_2^2 \tag{3.6}$$

where $\|\cdot\|_2$ is the Euclidean norm. Hence, Eq. (3.6) is the objective function of the problem, which can be formalized through Eq. (3.7).

$$\underset{\omega \in \mathbf{R}^n}{\text{Minimize}} \frac{1}{N} \sum_{i=1}^{N} \left\|y_i - o_i\left(x_i, \boldsymbol{\omega}\right)\right\|_2^2 \tag{3.7}$$

Therefore, and in accordance with the prior formalization, the training problem of a FNN (without specifying a loss function) can be stated as

$$\underset{\boldsymbol{\omega} \in R^n}{\text{Minimize}} f(\boldsymbol{\omega}) \tag{3.8}$$

A small example is shown to illustrate the encoding strategy for the optimization algorithms. Given the fully connected FNN shown in Fig. 3.2 with three layers where $n_0 = 3, n_1 = 4, n_2 = 2$, a parametrization would be encoded using Eq. (3.9).

$$\begin{aligned} W_1 &= \begin{bmatrix} w_{1-4} & w_{2-4} & w_{3-4} \\ w_{1-5} & w_{2-5} & w_{3-5} \\ w_{1-6} & w_{2-6} & w_{3-6} \\ w_{1-7} & w_{2-7} & w_{3-7} \end{bmatrix} \\ W_2 &= \begin{bmatrix} w_{4-8} & w_{5-8} & w_{6-8} & w_{7-8} \\ w_{4-9} & w_{5-9} & w_{6-9} & w_{7-9} \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \\ \mathbf{b}_3 &= \begin{bmatrix} b_8 \\ b_9 \end{bmatrix} \end{aligned} \tag{3.9}$$

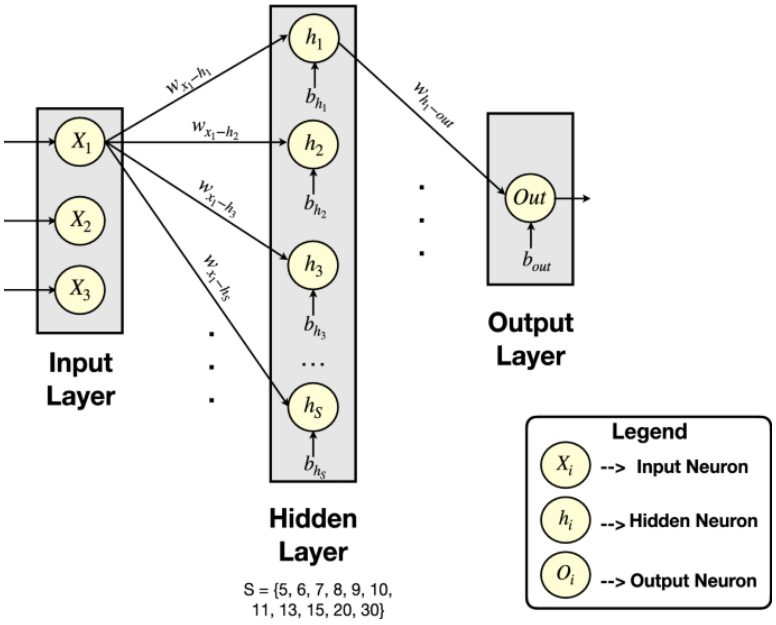


Fig 3.1.1: Fully Connected Feed-Forward Neural Network

We use the vector $\boldsymbol{\omega}$ to represent the parametrization of the network

$$\begin{aligned}\boldsymbol{\omega} =&[w_{1-4}, w_{1-5}, w_{1-6}, w_{1-7}, \ldots, w_{4-8}, w_{4-9} \\ &\ldots, b_4, b_5, \ldots, b_9]^\top\end{aligned} \tag{3.10}$$

In population-based metaheuristics, each individual of the population is represented by a vector $\boldsymbol{\omega}$ which includes the weights and biases of the different layers of the network. At the end of the optimization process, the best individual in the population will be used to set the parameters of the neural network.

3.2 Meta-Heuristic Formulation of FNN Components

In the previous section, we studied how a feed-forward neural network or a neural network faces issues during the training phase. In this thesis, we have tried to overcome the following problem using a meta-heuristic. We will describe how the researchers have applied the meta-heuristic to evolve the FNN, and we will study each component of the FNN and how we can optimize it using the meta-heuristic.

Each FNN component can be separately optimized on a one-by-one basis. Therefore, firstly, the weights can be optimized by keeping a fixed architecture; secondly, the architecture can be optimized by keeping weights fixed; thirdly, the activation function can be optimized by maintaining architecture and/or weights fixed; and so on. Another way is to optimize all or a combination of FNN components simultaneously. Therefore, weights and architecture can be optimized simultaneously; or weights and activation functions, simultaneously; or weights, architecture, and activation functions simultaneously; and so on. In the simultaneous optimization of all or a combination of components, a vectored representation of all components or a combination of components can be optimized, respectively. Once a vectored representation (genotype) is designed, the metaheuristic algorithm can be applied to optimize the designed vector to obtain an optimum FNN.

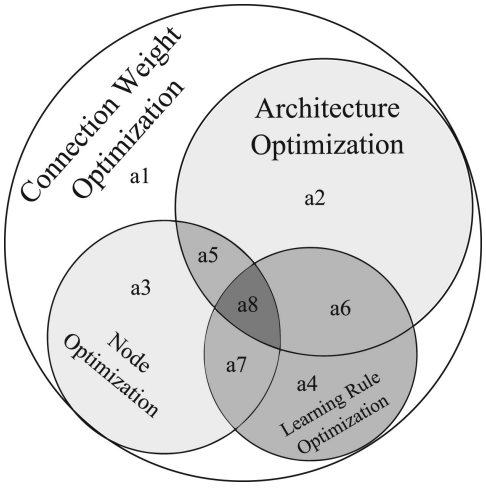
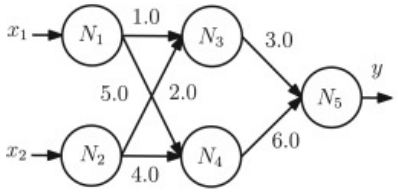


Fig 3.2 : Meta-Heuristic Formulation of FNN Components

3.2.1 Weight Optimization

FNN weight optimization is the most common and widely studied approach, in which the weights are mapped onto an n-dimensional weight vector w, where n is the total number of weights in a network. The vector $\mathbf{w}$ is a genotype representation of a phenotype (FNN structure), where the weight $\mathbf{w} \in R^n$. The weights w_i, an element of vector $\mathbf{w}$, may be encoded in the following ways: by assigning a real

value, l-bits binary coding, l-bits gray coding, IEEE floating point coding, etc. Fig. 3.2.1 is an example of phenotype to genotype mapping, where a phenotype shown in Fig. 3.2.1(a) that has the connectivity matrix c as per Fig.3.2.1 (b) is encoded into three different weight vectors shown in Fig. 3.2.1(c).



(a)

$$c = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \mathbf{0} & 0 & 1 & 1 & 0 \\ 2 & 0 & \mathbf{0} & 1 & 1 & 0 \\ 3 & 0 & 0 & \mathbf{0} & 0 & 1 \\ 4 & 0 & 0 & 0 & \mathbf{0} & 1 \\ 5 & 0 & 0 & 0 & 0 & \mathbf{0} \end{array}$$

(b)

(i) $\mathbf{w} = \langle 1.0, 5.0, 2.0, 4.0, 3.0, 6.0 \rangle$

(ii) $\mathbf{w} = \langle 0001, 0101, 0010, 0100, 0011, 0110 \rangle$

(iii) $\mathbf{w} = \langle 0001, 0111, 0011, 0110, 0010, 0101 \rangle$

(c)

Fig 3.2.1 :Mapping of phenotype to genotype. (a) Phenotype of a three-layer FNN. (b) Adjacency matrix. (c) Weights encoding: (i) real value; (ii) 4-bits binary; (iii) 4-bits gray encoding.

FNN weights optimization using metaheuristic is practiced from early 80's when even the term metaheuristic was not used. Engel (1988) work on FNN weight vector optimization using SA was the first evidence of metaheuristic application. To optimize weight vector using SA, first, the phenotype was mapped onto a real-valued weight vector (Fig. 3.2.1(c)), and to compute fitness of the FNN, a reverse mapping from genotype (weight vector w) to phenotype (FNN) was used. Such process was continued until a satisfactory solution was found. SA based FNN weight optimization was found to be performing better in comparison to conventional approaches Shang and Wah (1996); Sexton et al. (1998)

For optimizing the FNN weights, EAs use two types of vector representation: real-valued and binary valued vector representation. In Fig. 3.2.1(c), following weight vector representation are illustrated: (1) real-valued coded chromosome, (2) binary coded chromosome, and (3) binary gray-coded chromosome.

Goldberg (2006)gave the idea of FNN training using GA. However, Whitley (1989) were the first to propose "GENITOR," a GA based FNN training procedure that

used binary-coded chromosome (Fig.3.2.1(c)) for optimizing the weights. Many others followed the idea of GENITOR with some additional improvements such as connectivity optimization and reduced search space introduction Whitley et al. (1990). On the other hand, Belew et al. (1990) used a binary gray coding (Fig. 3.2.1(c)) scheme for optimizing the weights, where at first, GA was used for finding initial weights that were further optimized by using BP and vice versa.

The binary bit-string representation of the weights leads to a precision problem, i.e., how many bits would be sufficient for representing weights and what would be the total length of a chromosome. Moreover, the binary representation is computationally expensive because, in each training iteration, a binary to real-valued mapping and vice versa is required. Hence, it is advantageous to use the real-coded chromosome (Fig. 3.2.1(c)) directly Montana et al. (1989); Fogel et al. (1990); Sietsma and Dow (1991).

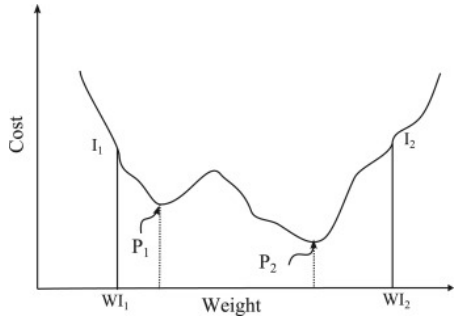


Fig 3.2.2: Metaheuristic may be used for finding initial weights WI_2 and conventional algorithms may be for finding global optima P_2 and vice versa Yao et al. (2007).

The swarm-based or bio-inspired based metaheuristics directly apply heuristics inspired by nature on a real-valued vector. Hence, they are advantageous in comparison to an EA-based algorithm that needs to simulated mutation and crossover operators for real-valued weight vector Kennedy (2006). It was found that PSO guides a population of the FNN weight vectors towards an optimum population

Ismail and Engelbrecht (2000). Hence, many researchers resorted to working on swarm based metaheuristics for the FNN optimization.

3.2.2 Architecture plus weight optimization

The basic architecture optimization approach is a cascade correlation learning, which iteratively adds nodes to hidden layer to construct optimum architecture Fahlman and Lebiere (1989). Moreover, a constructive (add node iteratively) and destructive (prune nodes iteratively) method Frean (1990). However, the constructive and the destructive methods for optimizing architecture are no different from the manual trial-and-error method. Therefore, genetic representation of the FNN architecture as mentioned in Figs. (a) – (c) can be used for architecture optimization, which is equivalent to searching optimum architecture from a compact space of FNN topology.

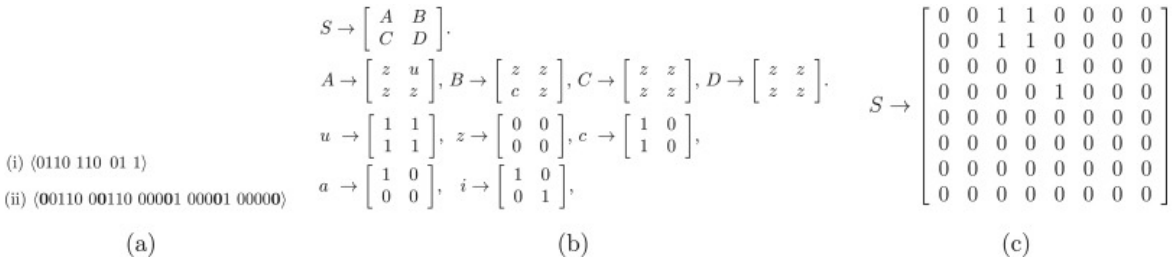


Fig 3.2.3 :Mapping of phenotype (Fig. 3.2.1 to genotype (for architecture). (a) Direct encoding to a vector of connectivity matrix (Fig. 3.2.1(b)): (i) upper right triangle; (ii) complete connectivity. (b) Indirect encoding schemes for architecture (Fig. 3.2.1(a)), where S is a start symbol, A, B, C, and D are the variables, and a, c, i, and u is the terminal. (c) Complete connectivity derived from rules operation shown in Fig.3.2.3

We describe the genetic representation of FNN architecture in detail . A direct encoding scheme was proposed by Whitley et al. (1990) and Schaffer et al. (1990), where the authors used an adjacency matrix (Fig. 3.2.1(b)) to represent

connections between nodes, where between any two nodes i and j, a presence of connection is indicated by " 1 ", and absence of connection is indicated by "0". Hence, they were able to encode complete structural information into a chromosome. However, it is disadvantageous because chromosome length increases with network size. Therefore, if only the network's structural information can be encoded into genotype, then, it will avoid chromosome length problem Harp et al. (1989). Additionally, the encoded network structural information can be accessed using rule-based recursive equation Mjolsness et al. (1989).

The indirect encoding scheme reduces chromosome length, where parametric information, such as the number of hidden layers, the number of nodes at hidden layers, the number of connection, etc., makes an archive s. The production rule (Fig. 3.2.3(b)) allow us to get access to complete structural information (Fig. 3.2.3(c)). Hence, a rule based encoding scheme allows a better FNN architecture optimization than a direct encoding scheme Siddiqi and Lucas (1998).

3.3 Meta-heuristic Algorithm for Optimization

Metaheuristic algorithm performs optimization by randomly defining the initial population of individuals in the search space according to some solution representation. As each individual represents a solution, the fitness of each individual in the population is then evaluated. Next, a new population is generated by perturbating solutions in the existing population through evolutionary operations, such as crossover and mutation. Hence, the search parameter is iteratively improved until it meets the stopping criteria in terms of the number of iterations or elapsed time. During iteration, the algorithm will attempt to obtain the final global minimum or maximum values which fulfil the respective objective function. A metaheuristic algorithm may not return the actual or exact solution but the near optimal solution.

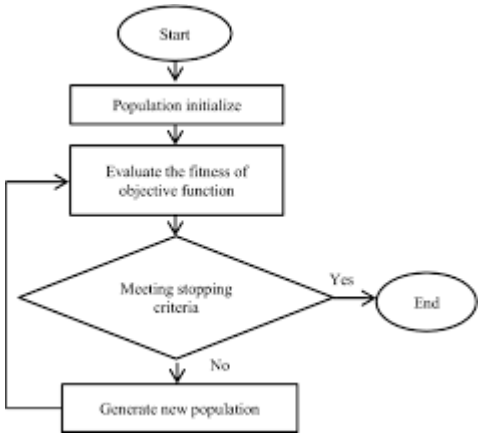


Fig 3.3 : Schematic Diagram Meta-Heuristics Optimization

3.3.1 Process of Optimizing Weight and Biases Using Meta-heuristic

A MLP is trained by supplying the training data , the metaheuristics formulate the components such as weights, bias, structure, nodes, etc.., into an optimization problem. The meta heuristics helps in enhancing the classification performance of neural networkOsman and Laporte (1996) . The fundamental motive behind combining a Meta-Heuristic and ANN(MLP) is to train the neural network to optimize the weights. The meta-heuristics are also used in conjunction with neural networks to pick the features to determine optimal training parameters, the training speed also improves. There are general steps that are followed when a Meta-Heuristics is used to train a Artificial Neural Network.

1. The metaheuristics determine the initialization of ANN with random Weights in the initial phase of training.

2. The training input samples are provided to the network for classification or the prediction dataset.

3. Then the output of the ANN is compared with predicted output. Also we calculate the error value using the error function such as MSE and others.

4. The Meta heuristics algorithm is then applied on random data in order to generate the next set of weights for the next iteration. Finally the target function uses input variable to access the potential population.

5. The step is repeated until all the requirements have been satisfied. The ideal solution is achieved if it has lowest error over all the iterations.

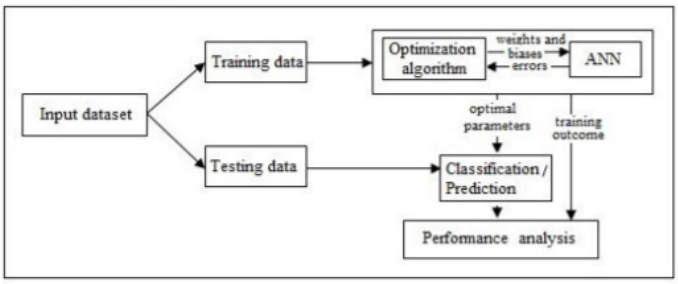


Fig 3.3.1 : Schematic Diagram For Training Neural Network with Meta-Heuristics

3.4 Conclusion:

It can be concluded that metaheuristic algorithms have provided various dimensions to the optimization of the FNNs. In this chapter, we have discussed the training problem of FNN/MLP and the role of the meta-heuristics. We can finally understand how meta-heuristic optimize the MLP, and how we can formulate the FNN with the help of the meta-heuristic algorithm. This chapter 3 has explained step by step how a meta-heuristic algorithm optimizes the weight and biases.However, the primary disadvantage of using metaheuristic algorithms is the training time consumption. Since the metaheuristic algorithms use a population (many solution candidates) during optimization, the time consumption becomes directly proportional to the number of candidates in a population.

Chapter 4

Introduction to Whale Optimization Algorithm

As we have previously discussed in 3 about Meta-Heuristic formulation to optimize weight and biases in MLP. In this Chapter we introduce one of the meta-heuristic swarm based algorithm known as Whale Optimization Algorithm . We will also introduce the method to train MLP with Whale Optimization Algorithm in the 4 , we will describe the process and the following steps that are to be followed to develop MLP-WOA trainer.

The motivation for the Whale Optimization method is discussed first in this chapter, followed by the explanation of the behaviour and its mathematical model.

4.1 What is Whale Optimization ?

WOA is a swarm intelligence algorithm proposed for continuous optimization problems. It is proven that this algorithm has better or comparable performance as compared to some existing algorithmic techniques Aljarah et al. (2018). The hunting behavior of the humpback whales has been the source of inspiration for WOA Aljarah et al. (2018). In WOA each solution is thought to be a whale. In this solution, a whale tries to replete a new place in the search space considered as a

reference the best element of the group. Two mechanisms are used by the whales to be used for search for the prey location, and attack. In the first one, the preys are encircled and the second creates bubble nets. Regarding optimization, when the whales look for a prey the exploration of the search space is performed and the exploitation occurs during the attack behavior.

4.1.1 Inspiration

Whales are fancy creatures. They are considered the biggest mammals in the world. Whales are mostly considered predators, as they never sleep because they have to breathe from the ocean's surface and their half brain sleeps. The whales are also considered as most emotionally intelligent creatures.

According to Hof and Van Der Gucht(Hof and Van der Gucht (2007)), whales have common cells in a certain area of the brain, like humans, known as the spindle cell. These cells are responsible for judgement, emotion and social behaviour, making us distinct from other creatures. Whales have twice the number of these cells than adult humans, making them smarter. It has also been proven that they can think, learn and judge.

Another unique factor of whale behaviour is their social behaviour. They either live alone or in small groups. They are, however, typically seen in groups. Some of their species, such as killer whales, can live as a family for their whole lives. Humpback whales are one of the giant baleen whales (Megaptera novaeangliae). The size of an adult humpback whale is comparable to that of a school bus. Their favourite preys are krill and fish herds.

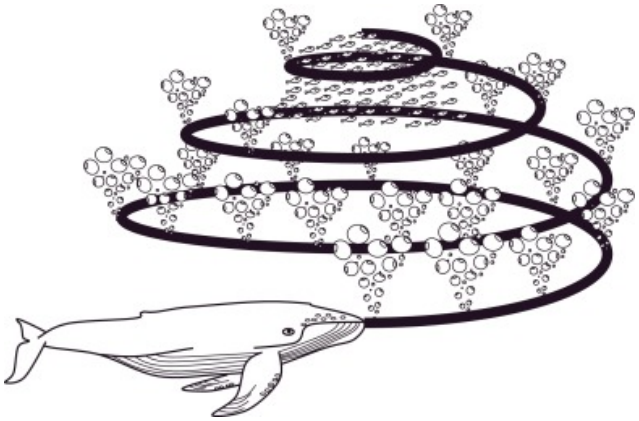


Fig 4.1 : Bubble Net Feeding Behaviour of Whales

The most interesting thing about humpback whales is their unique hunting method. This foraging behaviour is called the bubble-net feeding method. Humpback whales tend to hunt krill or small fish near the water's surface. It has been observed that foraging is done by creating distinctive bubbles along a circle or '9'-shaped path, as shown in Fig 4.1. Before 2011, the only way to study this activity was to see it from the surface. However,Goldbogen et al. (2013) used tag sensors to explore this behaviour. They captured 300 tag-derived bubble-net feeding events of 9 individual humpback whales. They discovered two bubble-related movements, which they termed 'upward-spirals' and 'double-loops'.Humpback whales dive to around 12 metres, then begin to build a spiral of bubbles around their prey and swim up to the surface. The later manoeuvre includes three different stages: coral loop, lobtail, and capture loop. Detailed information about these behaviours can be found in Goldbogen et al. (2013).

Bubble-net feeding is a unique behaviour seen only in humpback whales, and it's worth discussing here. The spiral bubble-net feeding technique is mathematically studied in this thesis to optimise it.

4.1.2 Significance of Whale Optimization

The process of obtaining a suitable equivalence between exploitation and exploration in improving any metaheuristic algorithm is the topmost challenge due to the arbitrary nature of the optimization algorithm. WOA has the highest significance compared to the different optimization approaches through the following

1. Exploitation ability
2. Exploration ability
3. Ability to get rid of the local minima

The WOA has an important capability of exploration due to the position updating mechanism of whales by using equation (4.5). Throughout the initial step of the algorithm, this equation forces the whales to move randomly around each other. In the following steps, equation (4.6) makes the whales update their positions rapidly and move along a spiral shaped route in the direction of the best path found so far. Since these two stages are done independently and in half iteration each, the WOA avoids local optima and achieves convergence speed simultaneously through the iterations. But most of the other optimization algorithms (like PSO and GSA) Tavazoei and Haeri (2007) do not have operators to consecrate a particular iteration to the exploration or the exploitation because they use only one format to update the position of search agents, so the probability of falling into local optima is more likely increased .

4.2 Mathematical Model and Optimization Algorithm

The mathematical model of encircling prey, spiral bubble-net feeding maneuver, and prey search are initially presented in this section. After that, the WOA algorithm is proposed.

4.2.1 Encircling prey

Humpback whales can recognize the location of prey and encircle them. Since the position of the optimal design in the search space is not known a prior, the WOA assumes that the current best candidate solution is the target prey or is close to the optimum. The effort is made to identify the best search agent, while the other search agents will update their positions near to the best search agent. The behavior is expressed by the following equations as follows:

$$\begin{aligned} \vec{D} &= \left| \vec{C} \cdot \overrightarrow{X^*} - \vec{X}(t) \right| \\ \vec{X}(t+1) &= \left| \vec{X}^*(t) - \vec{A} \cdot \vec{D} \right| \end{aligned} \tag{4.1}$$

where t indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, X^* is the position vector of the best solution obtained so far, $\vec{X}$ is the position vector, I is the absolute value, and . is an element-by-element multiplication. It is worth mentioning here that X^* should be updated in each iteration if there is a better solution.

The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$\begin{aligned} \vec{A} &= 2\vec{a} \cdot \vec{r} - \vec{a} \\ \overrightarrow{\mathrm{C}} &= 2.\overrightarrow{\mathrm{r}} \end{aligned} \tag{4.2}$$

Where $\vec{a}$ is linearly decreased from 2 to 0 over the range of iterations which plays the important role in order to control the exploration and exploitation phases. While $\overrightarrow{\boldsymbol{r}}$ denotes as a random vector in [0], [1], which has significant role to make each search agent reaching any position during its search space to ensure the exploration. In equation 4.1 solutions verify their locations in accordance with the site of the best solution (prey). In WOA to achieve the shrinking encircling behaviour , a is reduced using the following formula:

$$a = 2 - t\frac{2}{\mathrm{MaxIter}} \tag{4.3}$$

where t represents repetition of number and MaxIter is the maximum allowable iteration.

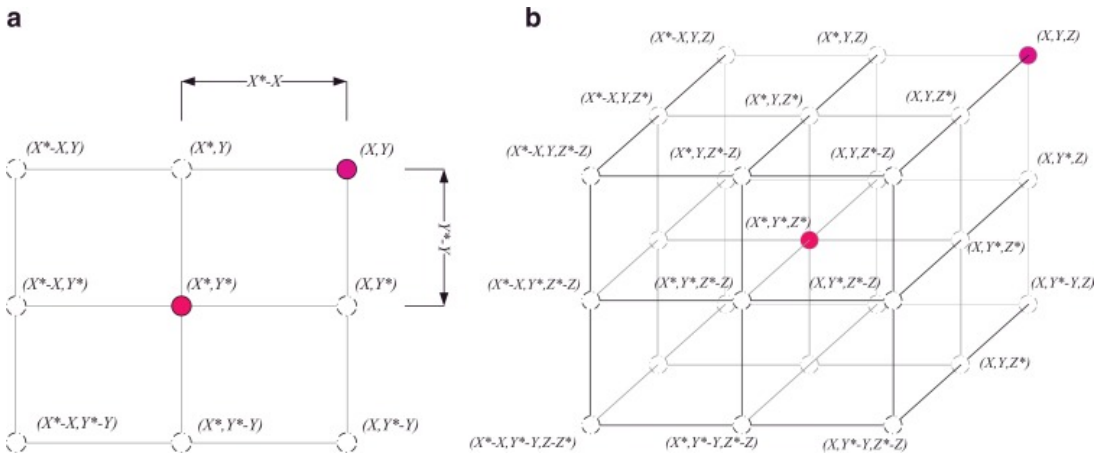


Fig 4.2: 2D and 3D position vectors and their possible next locations (X* is the best solution obtained so far).

The above image illustrates the rationale behind Eq.(4.1) for a 2D problem. The position (X, Y) of a search agent can be updated according to the position of the current best record $(X^{\star}, Y^{*})$. Different places around the best agent can be achieved with respect to the current position by adjusting the value of A and C vectors. The possible updating position of a search agent in 3D space is also depicted in Fig.4.2 (b). It should be noted that by defining the random vector $(\vec{r})$ it is possible to reach any position in the search space located between the key-points shown in Fig. 4.2. Therefore, Eq. (4.1) allows any search agent to update its position in the neighborhood of the current best solution and simulates encircling the prey.

The same idea can be used to a search space with **"n"** dimensions, with the search agents moving in hyper-cubes around the best answer found so far.The humpback whales, as indicated in the preceding section, use the bubble-net method to assault their prey. This strategy can be expressed numerically as follows:

4.2.2 Bubble-net attacking method (Exploitation phase)

In order to mathematically model the bubble-net behavior of humpback whales, two approaches are designed as follows:

4.2.2.1 Shrinking Circle Mechanism

This technique is modeled by minimizing $\overrightarrow{a}$ value as shown in Eq. (5.2).In other words A is a random value in the interval $[-a, a]$ where a is decreased from 2 to 0 over the course of iterations. Setting random values for $\vec{A}$ in $[-1, 1]$, the new position of a search agent can be defined anywhere in between the original position of the agent and the position of the current best agent. Fig.4.2.2 shows the possible positions from (X, Y) towards $(X^*, Y^\star)$ that can be achieved by $0 \leq A \leq 1$ in a 2D space.

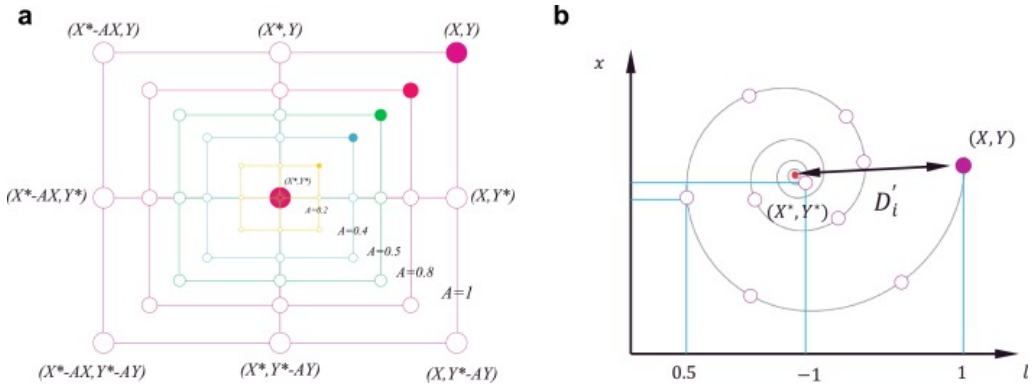


Fig 4.2.2: Bubble-net search mechanism implemented in WOA ($X^\star$ is the best solution obtained so far): (a) shrinking encircling mechanism and (b) spiral updating position.

Thus, the search agent updates its value during search space with a new value that is among the original location and the location of the current best.

4.2.2.2 Spiral Updating Position

This approach first calculates the distance between the whale located (X, Y) and prey located at$(X^*, Y^\star)$ as shown in Fig.4.2.2(b) .To imitate the helix-shaped movement of humpback whales, a spiral equation is established between the position of whale and prey as follows:

$$\vec{X}(t+1) = \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) \tag{4.4}$$

where $\overrightarrow{D'} = \left|\vec{X}^*(t) - \vec{X}(t)\right|$ and indicates the distance of the i th whale to the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, l is a random number in $[-1, 1]$, and . is an element-by-element multiplication.

To formulate the bubble-net behavior of humpback whale, a spiral mathematical formulation is applied between the position of whale and prey to imitate the helix-shaped movement of humpback whales . To represent this simultaneous behaviour, we assume that choosing between the shrinking encircling mechanism or the spiral model to update the position of whales during optimization has a 50% chance. The mathematical model is as follows:

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} & \text{if } p < 0.5 \\ \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) & \text{if } p \geq 0.5 \end{cases} \tag{4.5}$$

where p represents a random number in [0,1] for explaining the shape of the logarithmic spiral. In addition to using bubble nets, humpback whales hunt for prey at random.The mathematical model of the search is as follows.

4.2.3 Search for prey(exploration phase)

To find prey, the same approach is based on the modification of the $\vec{A}$ vector can be used (exploration). The humpback whales known fact is that they search at

random, based on their relative positions. As a result, we employ $\vec{A}$ to search for prey (exploration). We use $\vec{A}$ with values greater than **1** and **-1** to force search agents to move further away from the reference whale

While in exploitation phase we update the position of a search agent using a randomly picked search agent rather than the best search agent found this far. The mechanism and $|\vec{A}| > 1$ emphasizes exploration and allow the WOA algorithm to perform a global search. The mathematical model is as follows:

$$\begin{aligned} \vec{D} &= \left|\vec{C} \cdot \overrightarrow{X_{\text{rand}}} - \vec{X}\right| \\ \vec{X}(t+1) &= \overrightarrow{X_{\text{rand}}} - \vec{A} \cdot \vec{D} \end{aligned} \tag{4.6}$$

Where $\overrightarrow{X_{\text{rand}}}$ is a random position vector nominated arbitrarily from whales in the current iteration. Some of the possible solutions for a specific problem $\vec{A} > 1$ are illustrated in Fig 4.2.3.

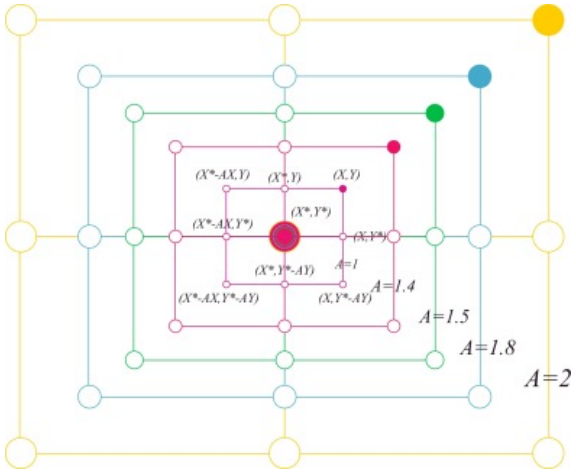


Fig 4.2.3: Exploration Implementation

A set of random solutions is used to start with the WOA algorithm. At the end of each iteration, search agents compare their positions to a randomly picked search agent or the best solution found so far. In order to facilitate exploration and exploitation, the a parameter is reduced from 2 to 0. When $|\overrightarrow{A}| > 1$, a random search agent is picked, and when $|\overrightarrow{A}| < 1$, the optimal solution is chosen for

updating the position of the search agents. WOA can flip between a spiral and a circular movement depending on the value of p. Finally, the WOA algorithm is ended when a termination requirement is satisfied.

Now we present the above discussed phenomenon in the form of the pseudo code:

Initialize the whale population $X_i (i = 1, 2, \cdots, n)$
Calculate the fitness function value
Randomly select the search agent X^*
while $t = 1$ *and* $t <$ *maximum iterations*
for *each search agent*
Update a, A, C, l, and p
if1 $(p < 0.5)$
if2 $(|A| < 1)$
Update the position of the current search agent by the Equation (8)
else if2 $(|A| \geq 1)$
Update the position of the current search agent by the Equation (9)
end if2
else if1 $(p \geq 0.5)$
Update the position of the current search by the Equation (10)
end if1
end for
Check if any search agent goes beyond the search space or else amend it
Calculate the fitness function value
Update X^* *if it is better*
$t = t + 1$
end while
return X^*

Fig 4.2.4: The pseudo code of the WOA algorithm

WOA can be considered a global optimizer from a theoretical standpoint because it has the potential to explore and exploit new areas. Furthermore, the suggested hyper-cube technique defines a search area in the vicinity of the best solution, allowing other search agents to take use of the current best record inside that domain. The WOA algorithm can easily transition between exploration and exploitation thanks to adaptive variation of the search vector A: by lowering A, certain iterations are dedicated to exploration ($|A| \geq 1$) and the rest to exploitation ($|A| < 1$).

Surprisingly, WOA only has two internal parameters that can be adjusted (A and C).

The first and most critical step is training an MLP using meta-heuristics. In other words, the problem of training MLPs should be formulated in a way that is suitable for meta-heuristics. The most important variables in training an MLP, as mentioned in the introduction, are weights and biases. A trainer should find a set of weight and bias values that provides the best classification/approximation/prediction accuracy. We apply Whale Optimization Algorithm to train MLP. The WOA will help in minimizing a cost function or maximizing a fitness function. We define the weights and biases as an optimization problem, then use WOA to solve the following problem and in other words, train the neural network.

4.3 Summarising Operation of Whale Optimization Algorithm

The WOA algorithm starts by assigning whales population with random solutions and assuming the best optimal value of the objective function is a minimum or maximum value (depending on the problem), then the objective function for each search agent is calculated.At each iteration, each search agent updates their location depending on either the best solution found so far when $|\vec{A}| < 1$ or on a randomly chosen search agent when $|\vec{A}| > 1$. In order to achieve exploration and exploitation phases, respectively, the value of a parameter is decreased from 2 to 0 .

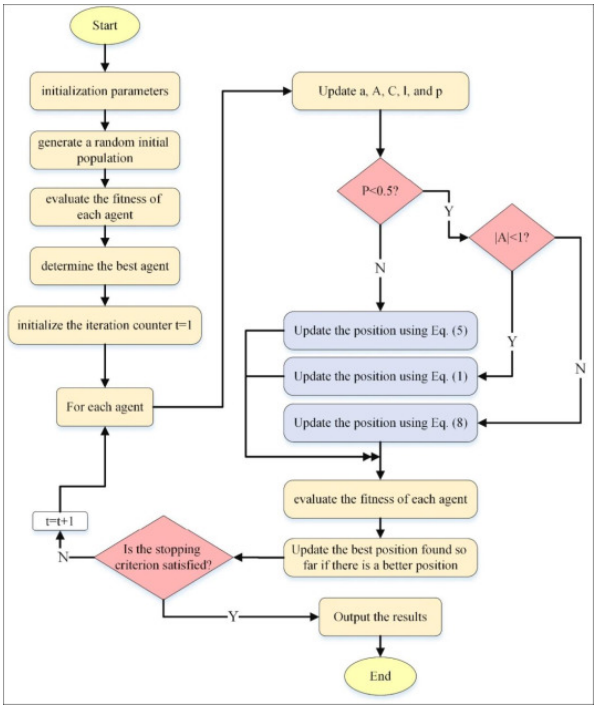


Fig 4.3: The Flowchart of the WOA algorithm

Also, WOA has the feature to select either a spiral or circular movement through the value of another parameter, which is p (a random number in $[0, 1]$) with a probability of 50% to select one of these two mechanisms, so if its value is greater than 0.5, then the search agents change their positions using Equation (4.5), otherwise they use Equation (4.1). Finally, the WOA algorithm ends by implementing of the termination condition .

4.4 Whale Optimization to train MLP

The first and most critical step is training an MLP using meta-heuristics. In other words, the problem of training MLPs should be formulated in a way that is suitable for meta-heuristics. The most important variables in training an MLP, as mentioned in the introduction, are weights and biases. A trainer should find a set of weight and bias values that provides the best classification/approximation/prediction accuracy. We apply Whale Optimization Algorithm to train MLP. The WOA will

help in minimizing a cost function or maximizing a fitness function. We define the weights and biases as an optimization problem, then use WOA to solve the following problem and in other words, train the neural network.

4.4.1 WOA based MLP trainer

In this section, we put forward the approach for using WOA for training the MLP network with one hidden layer which will be named as WOA-MLP.The MLP with initial settings is used to generate the first solution, and then WOA optimises the weights and biases to reduce the MLP's classification error rate.Two important aspects are taken into consideration when the approach is designed:the representation of the search agents in the WOA and the selection of the fitness function.

1. **Encoding Scheme :** Each whale in WOA is encoded as a one-dimensional vector to represent a candidate neural network. The vectors include three parts: The connection weights connecting the input layer and the hidden layer , the connection weights connecting the hidden layer and the output layer and the set of biases. The length of each vector is equal to the the total no of weights and biases in the neural network and it can be represented as follows using **equation(4.7)** where n is the no of inputs variable and m is the no of neurons in the hidden layer.

$$\text{Individual length } = (n \times m) + (2 \times m) + 1 \tag{4.7}$$

2. **Fitness Function:** Each whale is evaluated according to its fitness. This evaluation is done by passing the vector of weights and biases to MLP; then the MSE criterion is calculated based on the difference between the actual and predicted values by the generated agents (MLPs) for all training instances.The ideal solution is finally achieved after the maximum number of iterations is reached, and it is saved as the weights and biases of an MLP network. The aim is to minimise the value of MSE.

Since the WOA algorithm accepts the variables in the form of a vector, the WOA generates search agent inn the form of vectors and then assign it to the MLP as shown in Fig 4.4.1:

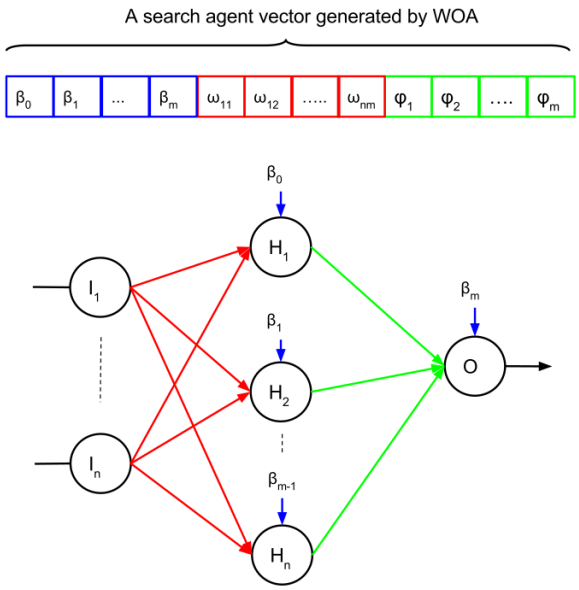


Fig 4.4.1: Vectors as Search Agents

Therefore to measure the fitness value of the generated WOA agents,we utilize the mean square error (MSE) fitness function which is based on calculating the difference between the actual and predicted values by the generated agents (MLPs) for all the training samples.

$$MSE = \sum_{i=1}^{m} \left(o_i^k - d_i^k\right)^2 \tag{4.8}$$

where m is the number of outputs, d_i^k is the desired output of the ith input unit when the k th training sample is used, and o_i^k is the actual output of the ith input unit when the kth training sample appears in the input.

The MLP adapts itself to the whole set of training samples in order to be effective. Therefore, the performance of an MLP is evaluated based on the

average of MSE over all the training samples as follows:

$$\overline{MSE} = \sum_{k=1}^{s} \frac{\sum_{i=1}^{m} \left(o_i^k - d_i^k\right)^2}{s} \tag{4.9}$$

where s is the number of training samples, m is the number of outputs, d_i^k is the desired output of the ith input unit when the kth training sample is used, and o_i^k is the actual output of the ith input unit when the kth training sample appears in the input.

Hence the problem of training an MLP with whale optimization can be formulated using the average MSE:

$$\text{Minimize } : F(\vec{V}) = \overline{MSE} \tag{4.10}$$

The WOA provides multi-layer perceptron with the weights and biases and calculates average mean square error of every training sample (Branch (2012)). By varying the weights and biases iteratively, the Whale Optimization algorithm gives reduced average mean square error for each training sample.

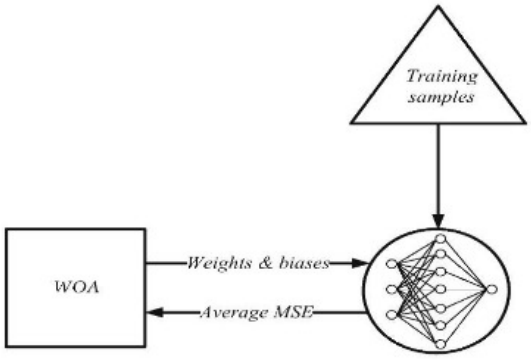
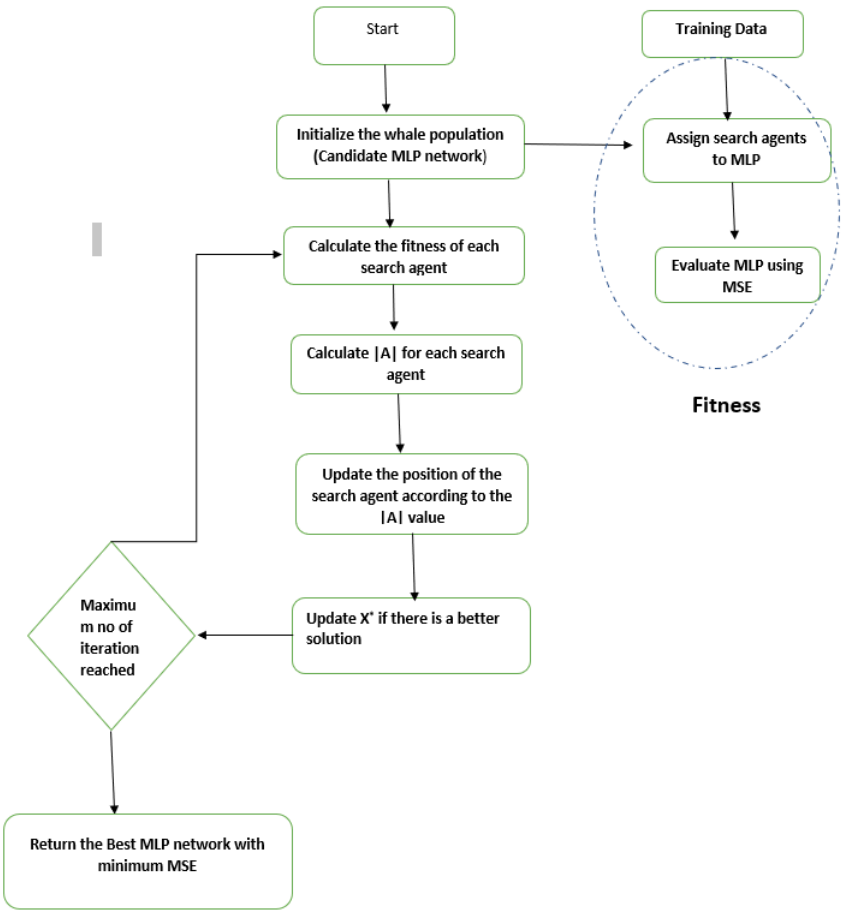


Fig 4.4.2: WOA presents MLP with weights and Biases to calculate average MSE

The following figure depicts training process of multi-layer perceptron with Whale Optimization Algorithm (WOA).

4.4.2 General Steps of WOA-MLP Approach:

The workflow of the WOA-based approach applied in this work for training the MLP network can be described in the following steps and flowchart:



Flowchart 4.4.3: Steps for WOA-MLP

1. Initialization: A predefined number of search agents are randomly generated. Each search agent represents a possible MLP network.

2. Fitness evaluation: The quality of the generated MLP networks is evaluated using a fitness function. To perform this step, the set of weights and biases that form the generated search agents vectors are first assigned to MLP networks, and then each network is evaluated. In this work, the MSE is selected,

which is commonly chosen as a fitness function in evolutionary neural networks. The goal of the training algorithm is to find the MLP network with the minimum MSE value based on the training samples in the dataset.

3. Update the position of the search agents.

4. Steps 2 to 3 are repeated until the maximum number of iterations is reached. Finally, the MLP network with the minimum MSE value is tested on unseen part of the dataset (test/validation samples).

4.5 Conclusion

We can finally say that proposed meta-heuristic algorithm includes three operator to simulate the search for the prey , encircling prey and bubble net foraging behaviour of the humpback whales . In this chapter we have introduced WOA-MLP trainer to optimize the weights and biases of MLP , and to also help in the training process of MLP .We have described the general steps followed during the training with the help of an flowchart.Further in the study we will analyze the exploration, exploitation , local optima avoidance and convergence behaviour and then compare them with other swarm and evolutionary algorithms .

Chapter 5

Experimental Setup and Discussion of Results

In the previous chapter we discussed the WOA-MLP trainer . We implement the WOA-MLP trainer and along with the other Meta-Heuristics on 20 classification dataset (Back Propagation ,Genetic Algorithm , Particle Swarm Optimization , Ant Colony Optimization, Differential Evolution , Evolution Strategy , Population Based Incremental Learning) to compare the results and establish if WOA (Whale Optimization) trainer is able to perform better than other meta-heuristic.

5.1 Experimental Setup

In this section, the proposed WOA approach for training MLP networks which is evaluated using twenty standard classification datasets, which are selected from the [†]University of California at Irvine (UCI) Machine Learning Repository and [‡]DELVE repository. Table 5.1.1 presents the above mentioned datasets in terms of the number of classes, features, training samples, and test samples. In the table it can be noticed that, the selected datasets have different numbers of features and

[†]http://archive.ics.uci.edu/ml/
[‡]http://www.cs.utoronto.ca/ delve/data/

instances to test the training algorithms in different conditions, which makes the problem more challenging.

5.1.1 Experimental Evaluations

For all experiments, we used Google Co-lab to implement the proposed WOA trainer and other algorithms. All datasets are divided into 66% for training and 34% for testing using stratified sampling in order to preserve class distribution as much as possible.

Normalization is an essential step for MLP when solving datasets with attributes in different ranges.Furthermore, to eliminate the effect of features that have different scales, all datasets are normalized using min-max normalization as given in the following equation:

$$v' = \frac{v_i - \min_A}{\max_A - \min_A} \tag{5.1}$$

where v' is normalized value of v in the range $[\min_A, \max_A]$.

All experiments are executed for ten different runs, and each run includes 250 iterations. In WOA, there are two main parameters to be adjusted A and C. These parameters depend on the values of a and r. In our experiments, we utilize a and r the same way as used in Mirjalili and Lewis (2016); a is set to linearly decrease from 2 to 0 over the course of iterations . In this section the results are compared to six metaheuristics namely PSO, DE, GA , ACO , ES and PBIL for verification. It is assumed that the optimization process starts generating random weights and biases in the range of [-10,10] for all the dataset .Other assumptions for algorithms are presented in Table 5.1.1

Further , In this section we discuss the classification of the dataset followed by tuning of parameters , MLP structure of each dataset and finally discuss results of the experiments.

DataSet	#Classes	#Features	#Training Samples	# Testing samples
Blood	2	4	493	255
Breast Cancer	2	8	461	238
Diabetes	2	8	506	262
Hepatitis	2	10	102	53
Verterbral	2	6	204	106
Diagnosis I	2	6	79	41
Diagnosis II	2	6	79	41
Parkinson	2	22	128	67
Liver	2	6	79	41
Australian	2	14	455	235
Credit	2	61	670	330
Monk	2	6	285	147
Tic Tac Toe	2	9	632	326
Titanic	2	3	1452	749
Ring	2	20	4884	2516
Twonoworm	2	33	4884	2516
Chess	2	36	2109	1087
Seed	2	7	138	72
Wine	2	13	117	61

Table 5.1.1: Classification of Datasets with their MLP structure,tuning parameters.

The above table shows the specification of the datasets. These classification datasets are chosen deliberately with different training/test samples and different tuning parameter to test the performance of the proposed WOA-MLP effectively.

Algorithm	Parameter	Value
GA	• Crossover Probability • Mutation Probability • Selection Mechanism	0.9 0.1 Roulette Wheel
PSO	• Acceleration Constants • Inertia weights	[2.1,2.1] [0.9,0.6]
DE	• Crossover Probability • Differential Weight	0.9 0.5
ACO	• Initial Phernomone (τ) • Phernomone Update Constant • Pherenomone Constant • Global pheronomone deacy rate(p_g) • Local pheromone decay rate (p_l) • Pheromone sensitivity (α) • Visibility sensitivity (β)	1e-06 20 1 0.9 0.5 1 5
ES	• λ • σ	10 1
PBIL	• Learning rate • Good population member • Bad population member • Elitism parameter • Mutational probability	0.05 1 0 1 0.1

Table 5.1.2: Initial Parameters of Meta-Heuristic Algorithm

The key factor in the experimental setup is the structure of MLPs. For MLP, researchers have proposed different approaches to select the number of neurons in the hidden layer. However, in the literature, there is no one standard method that is agreed about its superiority. In this work, we follow the same method proposed and used in Wdaa and Sttar (2008),Mirjalili et al. (2014) where the number of neurons in the hidden layer is selected based on the following formula: $2 \times N + 1$, where N is number of dataset features. By applying this method, the resulted MLP structure for each dataset is illustrated in Table 5.1.3.

Data Set	#Features	MLP Structure
Blood	4	4-9-1
Breast Cancer	8	8-17-1
Diabetes	8	8-17-1
Hepatitis	10	10-21-1
Verterbral	6	6-13-1
Diagnosis I	6	6-13-1
Diagnosis II	6	6-13-1
Parkinson	22	22-45-1
Liver	6	6-13-1
Australian	14	14-29-1
Credit	61	61-231-1
Monk	6	6-13-1
Tic Tac Toe	9	9-19-1
Titanic	3	3-7-1
Ring	20	20-41-1
Twonoworm	20	20-41-1
Ionosphere	33	33-67-1
Chess	36	36-73-1
Seed	7	7-15-1
Wine	13	13-27-1

Table 5.1.3: MLP Structure for each Dataset

The problem representation and objective function are to train MLPs with the algorithms in Table 5.1.1. The datasets are then trained 10 times using each algorithm to generate the results. The statistical results are then presented as average (AVG) and standard deviation (STD) of the obtained best MSEs in the last iteration by the algorithms. Obviously, lower average and standard deviation of MSE in the last iteration indicates the better performance.The statistical results are presented in the following in Tables 5.1.4; 5.1.5 and 5.1.6. All the highlighted values depicts the best results of the algorithms on the 20 classification datasets.

Datset\ Algorithm		WOA	BP	GA	PSO	ACO	DE	ES	PBIL
Blood	AVG	**0.7867**	0.6349	0.7827	0.7792	0.7651	0.7718	0.7835	0.7812
	STD	**0.0059**	0.2165	0.0089	0.0096	0.0119	0.0045	0.0090	0.0058
	Best	**0.7961**	**0.7804**	**0.7961**	0.7961	0.7765	0.7765	0.7922	0.7882
Breast Cancer	AVG	**0.9731**	0.8500	0.9706	0.9685	0.9206	0.9605	0.9605	0.9702
	STD	**0.0063**	0.1020	0.0079	0.0057	0.0391	0.0095	0.0095	0.0085
	Best	**0.9832**	0.9706	**0.9832**	0.9748	0.9622	0.9706	0.9748	0.9832
Diabetes	AVG	**0.7584**	0.5660	0.7504	0.7481	0.6679	0.7115	0.7156	0.7366
	STD	**0.0139**	0.1469	0.0169	0.0307	0.0385	0.0290	0.0233	0.0208
	Best	0.7786	0.6908	0.7748	**0.7977**	0.7557	0.7519	0.7519	0.7634
Hepatitis	AVG	**0.8717**	0.7509	0.8623	0.8434	0.8472	0.8528	0.8453	0.8491
	STD	**0.0318**	0.1996	0.0252	0.0378	0.0392	0.0318	0.0306	0.0267
	Best	**0.9057**	0.8491	0.9057	0.8868	0.8868	**0.9057**	0.8868	0.8868
Vertebral	AVG	**0.8802**	0.6858	0.8689	0.8443	0.7142	0.7821	0.8472	0.8623
	STD	**0.0141**	0.1465	0.0144	0.0256	0.0342	0.0582	0.0239	0.0214
	Best	**0.9057**	0.8113	0.8868	0.8774	0.7642	0.8679	0.8679	0.8962
Diagnosis I	AVG	**1.0000**	0.8195	**1.0000**	**1.0000**	0.8537	0.9976	**1.0000**	**1.0000**
	STD	**0.0000**	0.1357	**0.0000**	**0.0000**	0.1233	0.0077	**0.0000**	**0.0000**
	Best	**1.0000**	**1.0000**	**1.0000**	**1.0000**	1.0000	1.0000	**1.0000**	**1.0000**
Diagnosis II	AVG	**1.0000**	0.9073	**1.0000**	**1.0000**	0.8537	1.0000	**1.0000**	**1.0000**
	STD	**0.0000**	0.0994	**0.0000**	**0.0000**	0.1138	0.0000	**0.0000**	**0.0000**
	Best	**1.0000**	**1.0000**	**1.0000**	**1.0000**	1.0000	1.0000	**1.0000**	**1.0000**

Table 5.1.4: Accuracy Results for Blood, Breast Cancer , Diabetes , Hepatitis, Vertebral, Diagnosis I and Diagnosis II

Dataset/ Algorithm		WOA	BP	GA	PSO	ACO	DE	ES	PBIL
Parkinson	AVG	0.8358	0.7582	**0.8507**	0.8463	0.7642	0.8239	0.8254	0.8388
	STD	0.0392	0.1556	**0.0233**	0.0345	0.0421	0.0384	0.0264	0.0336
	Best	**0.8955**	0.8657	0.8806	**0.8955**	0.8209	0.8806	0.8657	0.8806
Liver	AVG	**0.6958**	0.5407	0.6780	0.6703	0.5525	0.5653	0.6347	0.6636
	STD	**0.0284**	0.0573	0.0524	0.0263	0.0639	0.0727	0.0535	0.0461
	Best	0.7373	0.6525	0.7373	0.7034	0.6356	0.6695	**0.7458**	0.7203
Australian	AVG	**0.8535**	0.7794	0.8289	0.8241	0.7724	0.8232	0.8096	0.8355
	STD	**0.0159**	0.0528	0.0228	0.0255	0.0474	0.0311	0.0297	0.0166
	Best	**0.8792**	0.8289	0.8553	0.8596	0.8342	0.8816	0.8509	0.8553
Credit	AVG	0.6991	0.6993	**0.7133**	0.7100	0.6918	0.7018	0.7061	0.7124
	STD	0.0212	0.0196	**0.0200**	0.0132	0.0219	0.0261	0.0263	0.0270
	Best	0.7364	0.7273	**0.7545**	0.7364	0.7121	0.7303	0.7303	0.7333
Monk	AVG	**0.8224**	0.6517	0.8109	0.7810	0.6646	0.7592	0.7884	0.7966
	STD	**0.0199**	0.0853	0.0300	0.0240	0.0659	0.0405	0.0320	0.0232
	Best	0.8571	0.7415	**0.8844**	0.8299	0.7551	0.8299	0.8299	0.8299
Tic-tac-toe	AVG	**0.6733**	0.5666	0.6353	0.6377	0.6077	0.8299	0.8299	0.6626
	STD	**0.0112**	0.0441	0.0271	0.0209	0.0347	0.6215	0.6383	0.0206
	Best	0.6840	0.6258	0.6687	0.6595	0.6810	0.0250	0.0313	**0.6902**
Titanic	AVG	0.7167	0.7377	0.7625	0.7605	0.7610	0.7622	0.7656	**0.7676**
	STD	0.0026	0.0441	0.0029	0.0049	0.0086	0.0072	0.0075	**0.0047**
	Best	0.7690	0.6258	0.7677	0.7677	0.7677	0.7690	0.7717	**0.7770**

Table 5.1.5: Accuracy Results for Parkinson, Liver , Australian, Credit , Monk , Tic-Tac-Toe,Titanic.

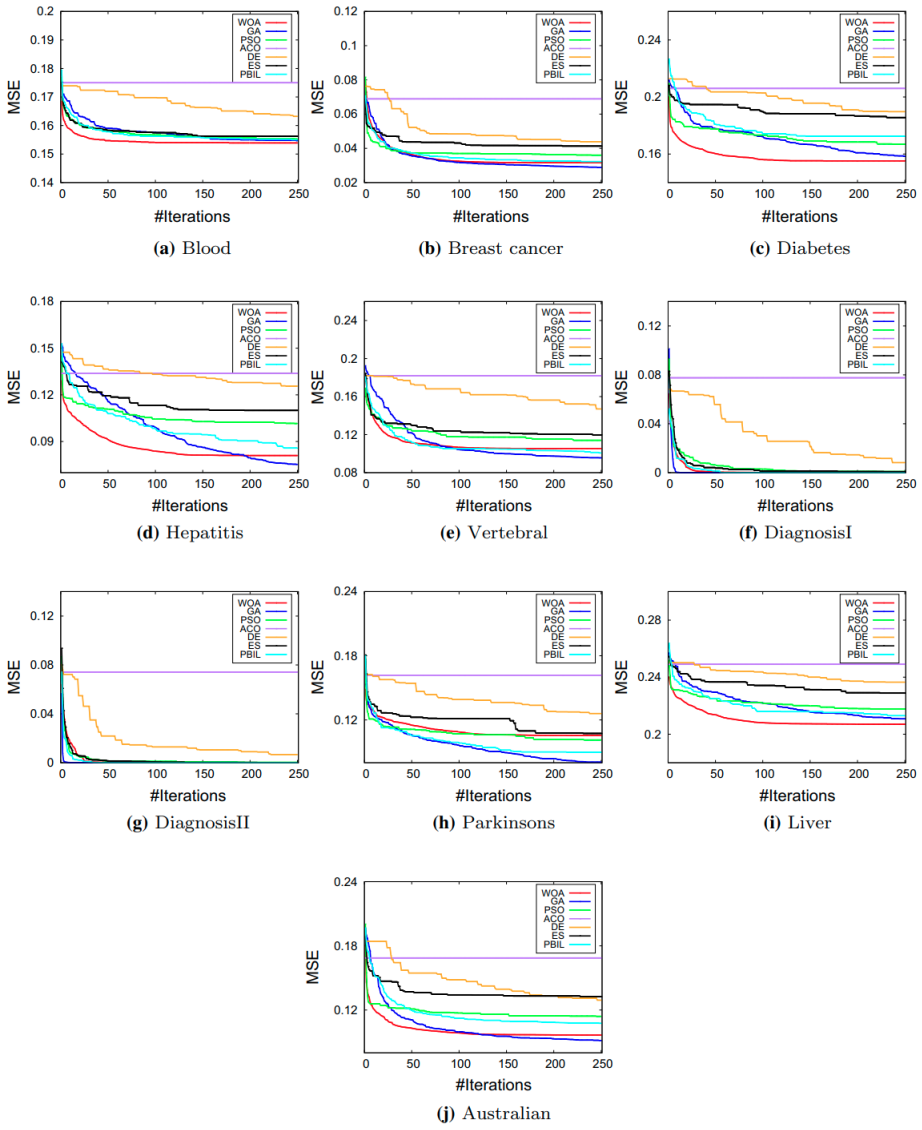
Dataset\ Algorithm		WOA	BP	GA	PSO	ACO	DE	ES	PBIL
Ring	AVG	**0.7729**	0.5502	0.7211	0.7091	0.6233	0.6719	0.6998	0.7393
	STD	**0.0084**	0.0513	0.0328	0.0118	0.0338	0.0230	0.0248	0.0176
	Best	**0.7830**	0.6526	0.7738	0.7266	0.6717	0.7075	0.7333	0.7655
Twono-worm	AVG	0.9744	0.6411	**0.9771**	0.9303	0.7572	0.8556	0.8993	0.9574
	STD	0.0033	0.1258	**0.0010**	0.0155	0.0398	0.0240	0.0131	0.0050
	Best	**0.9785**	0.9046	0.9781	0.9551	0.8275	0.8907	0.9134	0.9642
Ionosphere	AVG	0.7942	0.7367	**0.8205**	0.7600	0.7067	0.7767	0.7358	0.7825
	STD	0.0429	0.0385	**0.0157**	0.0242	0.0484	0.0340	0.0281	0.0240
	Best	**0.8667**	0.7833	0.8250	0.7197	0.8000	0.8417	0.7917	0.8333
Chess	AVG	0.7283	0.6713	**0.8088**	0.7160	0.6128	0.6695	0.6896	0.7733
	STD	0.0512	0.0471	**0.0238**	0.0218	0.0231	0.0289	0.0183	0.0196
	Best	0.8068	0.7259	**0.8500**	0.7479	0.6440	0.7259	0.7167	0.8040
Seed	AVG	**0.8986**	0.7986	0.8931	0.7903	0.5444	0.6347	0.7930	0.8583
	STD	**0.0208**	0.1277	0.0208	0.0580	0.1387	0.0747	0.0619	0.0375
	Best	**0.9306**	0.9167	0.9167	08889	0.7500	0.7361	0.8889	0.9028
Wine	AVG	**0.8894**	0.7697	**0.8894**	0.8227	0.6803	0.7576	0.7515	0.8667
	STD	**0.0335**	0.1153	0.0580	0.0474	0.1098	0.0763	0.0436	0.0557
	Best	**0.9545**	0.9091	0.9394	0.8939	0.8181	0.8788	0.8333	0.9091

Table 5.1.6: Accuracy Results for Ring, Twonoworm, Ionosphere, Chess, Seed and Wine.

The above tables shares the statistical results of the WOA-MLP trainer on the 20 classification dataset . The table determines how each algorithm has performed on each dataset. In the next section we discuss and analyse the results and summarise how WOA has performed when compared to other metaheuristics algorithm.

5.1.2 MSE Convergence Curve to Understand Which Algorithm Performed Faster

The following figures **5.1.2 , 5.1.3** show the convergence curves for the classification datasets employed using WOA, GA, PSO, ACO, DE, ES, and PBIL, based on averages of MSE for all training samples over ten independent runs. The figures show that WOA is the fastest algorithm for blood, diabetes, liver,monk, tic-tac-toe, titanic, and ring datasets. For other classification datasets, WOA shows very competitive performance compared to the best techniques in each case.



(a) Blood **(b)** Breast cancer **(c)** Diabetes

(d) Hepatitis **(e)** Vertebral **(f)** DiagnosisI

(g) DiagnosisII **(h)** Parkinsons **(i)** Liver

(j) Australian

Fig 5.1.2: MSE convergence curves of different classification datasets (a-j) MSE convergence curve for blood, breast cancer, diabetes, hepatitis, vertebral, diagnosis I, diagnosis II, Parkinson, liver, and Australian, respectively

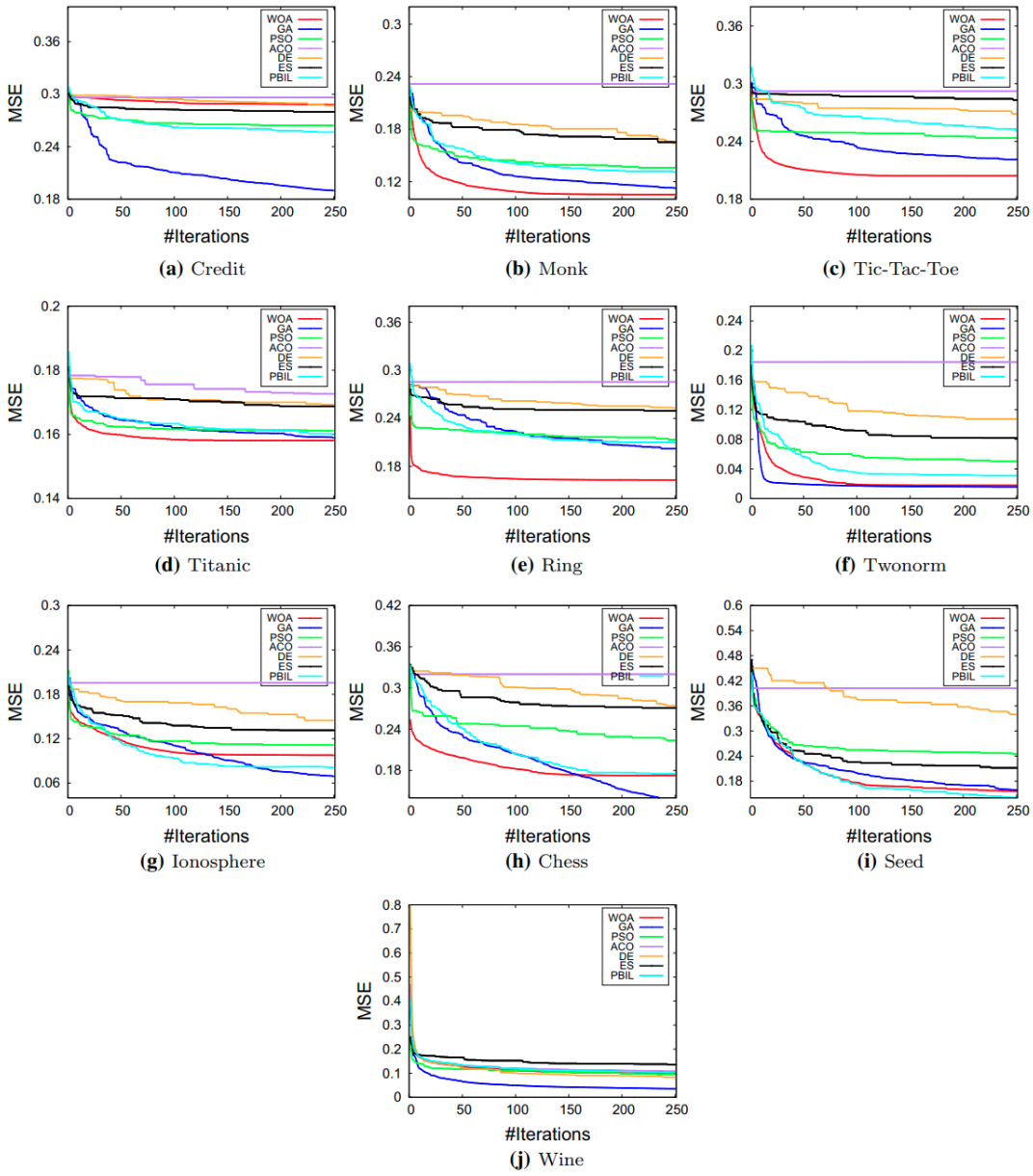


Fig 5.1.3: MSE convergence curves of different classification datasets (a-j) MSE convergence curve for credit, monk, tic-tac-toe, titanic, ring, twonorm, ionosphere, chess, seed, and wine, respectively

5.2 Discussion and Analysis of Results

The proposed WOA-MLP algorithm is compared with standard BP and other meta-heuristic trainers based on accuracy and MSE evaluation measures. Table **5.1.4, 5.1.5, 5.1.5** shows the statistical results, namely the average(AVG) and standard deviation of classification accuracy of the proposed WOA and other metaheuristics on the given dataset. As shown statistically, the algorithm provides superior local optima avoidance in twenty of the datasets and the best classification accuracy in all of the datasets. The WOA-MLP trainer outperforms all the other metaheuristics trainers for **Blood, Breast Cancer, Diabetes, Hepatitis, Vertebral, Liver, Diagnosis I, Diagnosis II, Australian, Monk, Tic-tac-toe, Ring, Wine, and Seeds Datasets** with an average accuracy of **0.7867, 0.9731, 0.7584, 0.8717, 0.8802, 1.000,1.000, 0.6958, 0.8535, 0.8224, 0.6733, 0.7729,0.8986, and 0.8894, respectively**

The reason for improved MSE is this algorithm's high local optima avoidance. The higher average and low standard deviation of the classification accuracy is achieved WOA-MLP trainer gives solid statistical evidence that this approach escapes the premature convergence towards local optima and finds the best optimum values for MLP's weights and biases. As we have seen statistically that WOA obtains the best accuracy compared to other algorithms employed.

Additionally, according to the mathematical formulation of the WOA algorithm, half of the iterations are devoted to the exploration of the search space when$|\mathbf{A}| > \mathbf{1}$, which promotes exploration of the search space. In addition, the C parameter always randomly pushes the search agents to take random steps toward the prey. This mechanism encourages resolving local optima stagnation even when the WOA algorithm is in the exploitation phase.

Further, the main reason for the high classification rate achieved by the WOA-MLP algorithm is that it is equipped with adaptive parameters to balance exploration

and exploitation smoothly. The random selection of prey in each selection is one of the primary strategies that helped the algorithm to avoid many local solutions. Another technique is WOA's encircling strategy, which compels search agents to search the area surrounding the prey. We should also focus on the fact that the high exploration ability of WOA comes from **Eq.4.6** due to the updating mechanism of whales. This equation makes the whales move randomly around each other. The **Eq.4.7** allows the whales to re-position themselves around or move in the spiral-shaped path towards the best optimal solution obtained so far. Since these two phases are done separately, and in almost half of the iterations each, the WOA shows high local optima avoidance and convergence speed during iterations.

The WOA-MLP trainer's higher convergence speed saves the best prey and adaptive search around it. The search agents in WOA tend to search more locally near the prey, proportional to the number of iterations. This feature of the WOA-MLP trainer helps outperform other algorithms in most datasets.

The most exciting finding in the results is the poor performance of PSO-MLP and ACO-MLP. These two algorithms belong to the class of swarm-based algorithms. This means that, unlike evolutionary algorithms, there is no mechanism for significant abrupt movements in the search space, leading to the poor performance of PSO-MLP and ACO-MLP. Although WOA is also a swarm-based algorithm, its mechanisms described in the preceding paragraph are why it is advantageous in training MLPs.

The most exciting finding among the results is the better result of algorithms **GA, PBIL and ES, respectively** when compared to PSO and ACO.

5.3 Friedman Test:

The Friedman Test is a non-parametric test developed by (Friedman et al. (2007)). This test is used for multiple comparisons of different results depending on two impact factors, namely the trainer method and the classification dataset.

We have used the Friedman Test to statistically evaluate the performance of each algorithm on all the classification datasets, we have used the Friedman test to establish the significance of the WOA-MLP trainer results against the other trainers.

Further, in table 5.3 below, we have shown the average rank of each trainer using the Friedman Test. The test depicts that there is no significant difference between the **"8"** presented algorithms, but WOA has proven to have the highest overall ranking in comparison with the other algorithms. This proves that WOA algorithms are suitable for training MLP's and FFNNs.

Algorithm	Ranking
WOA	2.05
BP	7.3
GA	2.2
PSO	4.269
ACO	7.2
DE	5.5
ES	4.6
PBIL	2.875

Table 5.3: Average Ranking of the Algorithms using Friedman Test

The results showed that the proposed WOA-MLP trainer is able to outperform the current algorithms in majority of the datasets, the results were achieving better accuracy but also convergence . The proposed trainer performed better than evolutionary algorithm due to high exploration and local optima avoidance, also we analysed that higher local optima avoidance does not degrade the convergence speed of the WOA. We can finally summarise that WOA is able to outperform due

high local optima avoidance and secondly the convergence speed of the proposed trainer is high. The WOA-MLP trainer is able to train Feed-Forward Neural Network effectively for the classified dataset with different level of difficulty. Hence , WOA-MLP trainer is able to train Feed-Forward Neural Network reliably without depending on the fact if there are small or large connection weights and biases . dealing with optimization issues in numerous domain of engineering. Several modifications, hybridizations, and multi-objective versions of WOA methods have been developed and applied by the researchers to optimize the constraints and issues in the engineering field . The WOA was used by (Touma (2016)) to solve the economic dispatch problem in engineering design , the algorithm was tested and verified using standard test system IEEE 30-Bus . The results depicted a reduction in the power output and provided optimum solution to the total loss. Also the algorithm performed better than other metaheuristics like PSO, ACO and GA.

5.4 Open Problems and Future Research Direction:

In this thesis we have discussed and studied about Whale Optimization , but still there some issues that can be studied and further investigated to improve the algorithm and achieve better results. The following section we will discuss few challenges faced by the Whale Optimization Algorithm.

1. **Generalization:** A number of the reviewed WOA literature's imply that some amount of change to the original WOA is required in order for it to fit into various contexts.This gives rise to many new challenges, each with its own set of criteria and parameters.As a result, several versions of the WOA must evolve to tackle the various difficulties in the various circumstances.This generalisation problem is not exclusive to the WOA; most heuristic and meta-heuristic algorithms suffered from it as well. In general, a more comprehensive research of generalization and standardization in WOA and meta-heuristic will greatly improve their applicability in other research areas.

2. **Sensitivity to Parameters:** Though the WOA is less sensitive to parameter settings than other nature-inspired algorithms, parameter setting remains a concern because the WOA requires user-defined parameters to be executed. As a result, in order to have optimal solutions for each individual problem, the most optimised parameters must be selected.We need more study to develop an operational algorithm that would converge to the optimal solution to a problem with minimal effort.Additionally, future study should attempt to make WOA a parameter-free algorithm.

3. **Hybridization:** Hybridization is now acknowledged as an important component of meta-heuristic algorithm research. A variety of traditional meta-heuristic algorithms, particularly those used in soft computing, include modules derived from other intelligence algorithms.Although the WOA derived some of its structures from previous nature-inspired algorithms, more study into integration with other algorithms is needed to increase its adaptability.Hybridizations usually reach their limit when significant problem instances with large search spaces become achievable solutions.

4. **Big Data Exploration:** According to the available literature's, there is a greater emphasis on the evaluation of WOA adjustments in small data sets. There is very little literature demonstrating the use of algorithms with massive data, which is the current reality of data science. This is despite the fact that big data has sparked huge interest in both scientific inquiry and industry.

In future we can also extend the Whale Optimization using other algorithms and then train ANN using hybrid algorithms . Also , we can try using different kind of neural networks to train using WOA and analyse the results by comparing with the presented work.

5.5 Conclusion

This thesis presented how to use the swarm-based metaheuristic Whale Optimization algorithm to train an MLP(Multi-Layer Perceptron); humpback whales inspire the algorithm. The proposed method (named as WOA-MLP trainer) includes three operators to initialize the search of the prey, encircling prey and bubble net foraging method. The study was conducted on 20 classification datasets to analyze exploration, exploration, local optima avoidance and convergence behaviour and compare with other swarm based and evolutionary algorithms. The WOA-MLP trainer is found to be competitive and perform better than the other metaheuristics based MLP trainers.

We can now summarise that this thesis highly recommends applying WOA to the problem of training MLPs as it overcomes the issue faced before by other metaheuristics such as local optima avoidance and fast convergence speed. The problem of training the MLP is formulated as minimization of MSE, where the parameters were connection, weights and bias. The reason for the minimum MSE of the WOA algorithm is the ability of whales to identify their prey and compute its movement by exploiting it in the search space. The Whale Optimization algorithm here was used to find the best weights and biases values to minimize the MSE. We use a set of 20 test functions with diverse features to benchmark the performance of the WOA-MLP trainer.

This study can be further improved by using chaos initialization which will help in improving the exploration capabilities and, at the same time, also maintains the balance between exploration and exploitation. In future work, if the convergence factor can be improved, it will help in enhancing the exploitation capabilities of the WOA-MLP trainer.

Bibliography

H. Abedinpourshotorban, S. M. Shamsuddin, Z. Beheshti, and D. N. Jawawi. Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm. *Swarm and Evolutionary Computation*, 26:8–22, 2016.

I. Aljarah, H. Faris, and S. Mirjalili. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, 22(1):1–15, 2018.

S. Baluja. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science, 1994.

A. Bhattacharya and P. K. Chattopadhyay. Solving complex economic load dispatch problems using biogeography-based optimization. *Expert Systems with Applications*, 37(5):3605–3615, 2010.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

C. Blum and K. Socha. Training feed-forward neural networks with ant colony optimization: An application to pattern classification. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pages 6–pp. IEEE, 2005.

C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied soft computing*, 11(6):4135–4151, 2011.

M. Branch. A multi layer perceptron neural network trained by invasive weed optimization for potato color image segmentation. *Trends in Applied Sciences Research*, 7(6):445–455, 2012.

M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, 45(3):1–33, 2013.

M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

J. Engel. Teaching feed-forward neural networks by simulated annealing. *Complex Systems*, 2(6):641–648, 1988.

S. Fahlman and C. Lebiere. The cascade-correlation learning architecture. *Advances in neural information processing systems*, 2, 1989.

H. Faris, I. Aljarah, and S. Mirjalili. Training feedforward neural networks using multi-verse optimizer for binary classification problems. *Applied Intelligence*, 45 (2):322–332, 2016.

D. B. Fogel, L. J. Fogel, and V. Porto. Evolving neural networks. *Biological cybernetics*, 63(6):487–493, 1990.

M. Frean. The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural computation*, 2(2):198–209, 1990.

M. Friedman, L. J. Savage, and G. S. Becker. *Milton Friedman on Economics: selected papers*. University of Chicago Press, 2007.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

D. E. Goldberg. Genetic algorithms in search. *Optimization, and MachineLearning*, 1989.

D. E. Goldberg. *Genetic algorithms*. Pearson Education India, 2006.

J. A. Goldbogen, A. S. Friedlaender, J. Calambokidis, M. F. McKenna, M. Simon, and D. P. Nowacek. Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology. *BioScience*, 63(2):90–100, 2013.

S. A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural network. In *Proceedings of the third international conference on Genetic algorithms*, pages 360–369, 1989.

P. R. Hof and E. Van der Gucht. Structure of the cerebral cortex of the humpback whale, megaptera novaeangliae (cetacea, mysticeti, balaenopteridae). *The Anatomical Record: Advances in Integrative Anatomy and Evolutionary Biology: Advances in Integrative Anatomy and Evolutionary Biology*, 290(1):1–31, 2007.

J. H. Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

L. Ingber. Very fast simulated re-annealing. *Mathematical and computer modelling*, 12(8):967–973, 1989.

A. Ismail and A. P. Engelbrecht. Global optimization algorithms for training product unit neural networks. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 1, pages 132–137. IEEE, 2000.

D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Citeseer, 2005.

D. Karaboga, B. Akay, and C. Ozturk. Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. In *International conference on modeling decisions for artificial intelligence*, pages 318–329. Springer, 2007.

D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57, 2014.

A. Kaveh and M. Khayatazad. A new meta-heuristic method: ray optimization. *Computers & structures*, 112:283–294, 2012.

J. Kennedy. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*, pages 187–219. Springer, 2006.

E. Kolay, T. Tunç, and E. Eğrioğlu. Classification with some artificial neural network classifiers trained a modified particle swarm optimization. *American Journal of Intelligent Systems*, 6(3):59–65, 2016.

J. Li, J.-h. Cheng, J.-y. Shi, and F. Huang. Brief introduction of back propagation (bp) neural network algorithm and its improvement. In *Advances in computer science and information engineering*, pages 553–558. Springer, 2012.

N. Lopes and B. Ribeiro. An efficient gradient-based learning algorithm applied to neural networks with selective actuation neurons. *Neural, Parallel & Scientific Computations*, 11(3):253–272, 2003.

R. Mendes, P. Cortez, M. Rocha, and J. Neves. Particle swarms for feedforward neural network training. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 2, pages 1895–1899. IEEE, 2002.

X. Meng, J. Li, B. Qian, M. Zhou, and X. Dai. Improved population-based incremental learning algorithm for vehicle routing problems with soft time windows. In *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*, pages 548–553. IEEE, 2014.

S. Mirjalili. How effective is the grey wolf optimizer in training multi-layer perceptrons. *Applied Intelligence*, 43(1):150–161, 2015.

S. Mirjalili and A. Lewis. The whale optimization algorithm. *Advances in engineering software*, 95:51–67, 2016.

S. Mirjalili, S. Z. M. Hashim, and H. M. Sardroudi. Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*, 218(22):11125–11137, 2012.

S. Mirjalili, S. M. Mirjalili, and A. Lewis. Let a biogeography-based optimizer train your multi-layer perceptron. *Information Sciences*, 269:188–209, 2014.

E. Mjolsness, D. H. Sharp, and B. K. Alpert. Scaling, machine learning, and genetic neural nets. *Advances in applied mathematics*, 10(2):137–163, 1989.

B. Mohammadi. Letter to the editor "generating electrical demand time series applying sra technique to complement nar and sarima models" by jorge l. tena garcía, erasmo cadenas calderón, eduardo rangel heras, christian morales ontiveros. *Energy Efficiency*, 13(1):157–158, 2020.

D. J. Montana, L. Davis, et al. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767, 1989.

K. R. Opara and J. Arabas. Differential evolution: A survey of theoretical analyses. *Swarm and evolutionary computation*, 44:546–558, 2019.

I. H. Osman and G. Laporte. Metaheuristics: A bibliography, 1996.

E. Pacini, C. Mateos, and C. G. Garino. Distributed job scheduling based on swarm intelligence: A survey. *Computers & Electrical Engineering*, 40(1):252–269, 2014.

C. Sammut and G. I. Webb. *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

K. Sastry, D. Goldberg, and G. Kendall. Genetic algorithms. In *Search methodologies*, pages 97–125. Springer, 2005.

J. D. Schaffer, R. A. Caruana, and L. J. Eshelman. Using genetic search to exploit the emergent behavior of neural networks. *Physica D: Nonlinear Phenomena*, 42 (1-3):244–248, 1990.

U. Seiffert. Multiple layer perceptron training using genetic algorithms. In *ESANN*, pages 159–164. Citeseer, 2001.

R. S. Sexton, R. E. Dorsey, and J. D. Johnson. Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. *Decision Support Systems*, 22(2):171–185, 1998.

Y. Shang and B. W. Wah. Global optimization for neural network training. *Computer*, 29(3):45–54, 1996.

A. A. Siddiqi and S. M. Lucas. A comparison of matrix rewriting versus direct encoding for evolving neural networks. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, pages 392–397. IEEE, 1998.

J. Sietsma and R. J. Dow. Creating artificial neural networks that generalize. *Neural networks*, 4(1):67–79, 1991.

M. S. Tavazoei and M. Haeri. Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. *Applied Mathematics and Computation*, 187(2):1076–1085, 2007.

H. J. Touma. Study of the economic dispatch problem on ieee 30-bus system using whale optimization algorithm. *International Journal of Engineering Technology and Sciences*, 3(1):11–18, 2016.

L. Wang, Y. Zeng, and T. Chen. Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Systems with Applications*, 42(2):855–863, 2015.

A. S. I. Wdaa and A. Sttar. *Differential evolution for neural networks learning enhancement*. PhD thesis, Universiti Teknologi Malaysia Johor Bahru, 2008.

D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel computing*, 14(3):347–361, 1990.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

X.-S. Yang. Random walks and optimization. *Nature-inspired optimization algorithms*, pages 45–65, 2014.

Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

M. Yin, Y. Hu, F. Yang, X. Li, and W. Gu. A novel hybrid k-harmonic means and gravitational search algorithm approach for clustering. *Expert Systems with Applications*, 38(8):9319–9324, 2011.

J. Yu, S. Wang, and L. Xi. Evolving artificial neural networks using an improved pso and dpso. *Neurocomputing*, 71(4-6):1054–1060, 2008.

Y. Zhang, S. Wang, and G. Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015, 2015.