Levy Enhanced Cross Entropy-based Optimized Training of Feedforward Neural Networks

Kartik Pandya
M &V Patel Department of Electrical Engineering
FTE, CSPIT, CHARUSAT
Changa, India
kartikpandya.ee@charusat.ac.in

Dharmesh Dabhi
M &V Patel Department of Electrical Engineering
FTE, CSPIT, CHARUSAT
Changa, India
harmeshdabhi.ee@charusat.ac.in

Pratik Mochi
M &V Patel Department of Electrical Engineering
FTE, CSPIT, CHARUSAT
Changa, India
vipul.rajput@djmit.ac.in

Vipul Rajput
Department of Electrical Engineering
Dr. Jivraj Mehta Institute of Technology
Mogar, India
vipulrajput16986@gmail.com

Received: 11 July 2022 | Revised: 27 July 2022 | Accepted: 1 August 2022

Abstract-An Artificial Neural Network (ANN) is one of the most powerful tools to predict the behavior of a system with unforeseen data. The feedforward neural network is the simplest, yet most efficient topology that is widely used in computer industries. Training of feedforward ANNs is an integral part of an ANN-based system. Typically an ANN system has inherent non-linearity with multiple parameters like weights and biases that must be optimized simultaneously. To solve such a complex optimization problem, this paper proposes the Levy Enhanced Cross Entropy (LE-CE) method. It is a population-based meta-heuristic method. In each iteration, this method produces a "distribution" of prospective solutions and updates it by updating the parameters of the distribution to obtain the optimal solutions, unlike traditional meta-heuristic methods. As a result, it reduces the chances of getting trapped into local minima, which is the typical drawback of any AI method. To further improve the global exploration capability of the CE method, it is subjected to the Levy flight which consists of a large step length during intermediate iterations. The performance of the LE-CE method is compared with state-of-the-art optimization methods. The result shows the superiority of LE-CE. The statistical ANOVA test confirms that the proposed LE-CE is statistically superior to other algorithms.

Keywords-artificial neural networks; cross entropy method; feedforward neural networks; Levy step; training

I. INTRODUCTION

The Artificial Neural Network (ANN) [1] is the one of the most popular Artificial Intelligence methods. It is inspired from the communication and computation abilities of the human brain and it mimics the learning techniques of human brain to find out relationships between the input and the output (target) variables of a test system. The human brain consists of millions of neurons which are interconnected in a complex network which takes input signal to perform voice recognition, image classifications, etc. at remarkable speed and accuracy. Similarly, an ANN may consist of many neurons which are subjected to input signals through the connection links. Every connection link has an associated weight, which is multiplied with the signal. Weights are the control variables which are used to solve the optimization problem. Weights and biases are updated in every iteration to finally obtain their optimal values which will minimize the Mean Square Error (MSE) function. This procedure is known as the training (learning) of an ANN.

A. Related Work

Feedforward Neural Networks (FNNs) with two layers are a widely used ANN topology [2]. It has been proved that two-layer FNNs can approximate any nonlinear or linear function with a good accuracy [3]. Training is an integral part of FNNs. Various optimization methods have been suggested in the literature to train FNN. Back Propagation (BP) is a popular gradient-based classical optimization method to train FNNs. But it is susceptible to slow convergence [4] and may get trapped into local minima [5]. Newton's method [6] is another classical method which has quadratic convergence that largely depends upon the choice of initial starting point (guess). A wrong initial guess will lead to a local optima of the problem under consideration. The last two decades, population-based meta-heuristic methods that solve the FNN training problem effectively have emerged. Authors in [7] proposed the use of ANNs to detect the vibration of the rotor shaft of a gas turbine. Authors in [8] suggested the use of Genetic Algorithm (GA) to precisely recognize sign gesture using feature extraction, but mutation strategy makes GA very time consuming and it is preferred for binary solution sets. Also, it is vulnerable to premature convergence. Authors in [9] proposed a new approach to address the optimal design of a FNN using self-adaptive penalty functions. Authors in [10] proposed a Particle Swarm Optimization (PSO)-based neuro-fuzzy model to

Corresponding author: Kartik Pandya

enhance dynamic voltage stability of a wind connected grid. Authors in [11] proposed a PSO-powered back-propagation neural network load-shedding strategy in the post-fault condition in a microgrid. Another AI method is Gravitational Search Algorithm (GSA) [12], which has been inspired from the law of gravity and mass interactions. Even though it is a simple method, the unbalance between the application of gravitational law and mass interactions may create premature convergence. Differential Evolutionary (DE) method [13] is another powerful method which uses mutation, crossover, and selection operators on various vectors to get optimal solutions, but the selection of crossover rate greatly affects its convergence. PSO [14] is one of the most popular meta-heuristic methods, as it is easily implementable and it has less parameters to be tuned.

B. Aims and Objectives

The related work (literature survey) reveals that many classical optimization methods suffer from premature convergence, whereas artificial intelligence methods are population-based with generated randomly initial solutions, and as a result, in each simulation run it gives different optimal solutions. So, the aim and objective of this research is to suggest a proper solution methodology which trains an ANN more effectively and with better accuracy. This paper proposes the Levy Enhanced Cross Entropy (LE-CE) optimization method, which is an extension of the Cross Entropy (CE) method. The contributions of this paper are summarized below.

C. Research Outline

- The LE-CE method is proposed for the optimization of the weights and biases of the ANN with the aim to minimize the MSE function.
- The incorporation of Levy flight increases the global exploration capability of the CE method, which improves the quality of the solution.
- The LE-CE method has fewer parameters to be tuned, so it is a fast method.
- The practical Iris classification system is used to check the effectiveness of the LE-CE method.
- The performance of the LE-CE method is compared with the WCCI 2018 award winning EVDEPSO [15] and GECCO 2019 award winning HL_PS_VNSO [16] computational intelligence methods.
- The LE-CE method outperformed the compared methods in terms of optimal solutions.
- ANOVA statistical test and Tukey's HSD test also proved that the proposed method is statistically different from the other compared methods.

II. LEVY ENHANCED CROSS ENTROPY METHOD FOR TRAINING FNNS AND ENCODING STRATEGY

CE method was proposed by Rubinstein [17]. It is a population-based meta-heuristic optimization method similar to the Differential Evolutionary method. However, unlike traditional meta-heuristic methods which directly update prospective solutions (particles) to obtain sub-optimal solutions, this method produces a distribution of prospective solutions and updates it by updating the parameters like mean and standard deviation of each dimension to obtain sub-optimal solutions. As a result, it decreases the probability of getting stuck into local minima. It has a very few parameters to be tuned and it can be easily executed. The basics of LE-CE method are explained below.

The population of particles is randomly generated and they obey the probability distribution function (pdf) $f(.;\Phi)$, where Φ is the vector of parameters which are to be optimized. Generally, $f(.;\Phi)$ is the Gaussian distribution parameterized by its mean m and variance σ^2, i.e. $\Phi=\left(m,\sigma^2\right)$. Secondly, a threshold value (χ) of the fitness function is selected and only those particles whose fitness values are less than the set threshold value, i.e. $f(x)<\chi$ are considered in the subsequent iterations. Such particles are known as elite particles μ_e. Then, the new parameterized distribution function $f\left(.;\phi^n\right)$ with elite particles is updated in such a way that it coincides the target distribution function $f\left(.;\phi^*\right)$ by minimizing the Kullback-Leibler (KL) divergence. This procedure finishes one iteration. In the subsequent iterations, a family of distribution functions $f\left(.;\phi^{(1)}\right)...f\left(.;\phi^*\right)$ are produced in accordance with $\chi^{(1)}....\chi^{(*)}$ to reach the sub-optimal distribution function $f(.;\varphi^*)$. The following section shows the step-by-step implementation of LE-CE.

1) Step 1: Initialization of Mean and Standard Deviation for Each Dimension

Assume iteration *iter*=0, Total no. of iterations $iter_{max}=500$, No. of particles N, Dimension (control variables) of the problem D, elite particles μ_e (20% of N), smoothing parameters α_s and β_s , $m_e^{(iter)}\in\mathbb{R}^D$ is the mean value of the search distribution for each dimension at iteration *iter*, $\sigma^{(iter)}\in\mathbb{R}^+$ is the standard deviation at iteration *iter*. x_{min} and x_{max} are the minimum and maximum limits of the D^{th} dimension particle. Mean and standard deviation of population are initialized as:

$$m_e^{(0)}=0.5*\left(x_{min}+x_{max}\right) \quad (1)$$

$$\sigma^{(0)}=0.25*\left(x_{max}-x_{min}\right) \quad (2)$$

2) Step 2: Generate the Population of Particles

The generation of the population of particles from the sampling distribution $\text{Normal}\left(m_e^{(iter)},\sigma^{2(iter)}\right)$ occurs as:

$$x_i^{(iter+1)}=m_e^{(iter)}+\sigma^{(iter)}\text{randn}() \quad \text{for } i=1,\ldots\ldots,N \quad (3)$$

where N is the population size, $x_i^{(iter+1)}\in\mathbb{R}^D$ is the i^{th} particle obtained at iteration $\left(iter+1\right)$, $m_e^{(iter)}\in\mathbb{R}^D$ is the mean value of the search distribution for each dimension at iteration *iter*, $\sigma^{(g)}\in\mathbb{R}^+$ is step-size (standard deviation) at iteration *iter*, randn() is a normally distributed random variable with parameters Normal(0,1).

3) Step 3: Fitness Function Evaluations

The fitness values of the whole population are determined as follows:

The FNN consists of n_i input nodes, h_i hidden nodes, and o_t output nodes. In each iteration of learning, the output of each hidden nodes is calculated using (4):

$$f\left(y_k\right) = 1 / \left(1 + \exp\left(-\left(\sum_{j=1}^{n_i} w_{jk} * x_j - \theta_k\right)\right)\right), \quad (4)$$
$$k = 1, 2 \ldots\ldots h_i$$

where $y_k = \sum_{j=1}^{n_i} w_{jk} x_j - \theta_k$, w_{jk} is the connection weight from the j^{th} node from the input layer to the k^{th} node in the hidden layer, θ_k is the bias of the k^{th} hidden node, and x_j is the j^{th} input.

The final output is calculated following the output of hidden nodes as given by (5):

$$o_t = \sum_{j=1}^{h_i} w_{lk} * f\left(y_k\right) - \theta_{l,} \quad l = 1, 2, \ldots., o_t \quad (5)$$

where w_{lk} is the connection weight between the k^{th} hidden node to the l^{th} output node and θ_l is the bias of the l^{th} output node.

Eventually, the mean square error (e_l) is calculated from:

$$e_{\,l} = \sum_{j=1}^{h_i} \left(o_{tj}^{l} - d_j^{l}\right) \quad (6)$$

$$e = \sum_{l=1}^{t} \frac{e_l}{t} \quad (7)$$

where t represents the training samples and d_j^l is the desired output of the j^{th} input when the l^{th} training sample is considered.

So, the fitness function of the j^{th} training sample can be defined as follows:

$$\text{Fitness}(x_j) = e(x_j) \quad (8)$$

4) Step 4: Ranking of Fitness Functions

Sort (rank) all fitness function values in ascending order as given in (9):

$$f\left(x_1\right) < f\left(x_2\right) < \ldots < f(N) \quad (9)$$

where $f(x_j)$ is the fitness of the j^{th} particle and x_1 is a Global Best$\left(G_{Best}\right)$ particle having the best (minimum) fitness among all particles. Consider the top best 20%-30% of the particles as elite particles.

5) Step 5: Updating Mean and Standard Deviation of the Elite Particles

The mean $\left(m_{t\mu}^{(iter+1)}\right)$ of the selected elite particles is found by:

$$m_{t\mu}^{(iter+1)} = \frac{1}{\mu_e} \sum_{j=1}^{\mu_e} x_{j:N}^{(iter+1)} \quad (10)$$

where $x_{j:N}^{(iter+1)}$ is the j^{th} best particle among the whole population at *iter*+1iteration. The index *j:N* denotes the index of the j^{th} rank particle. Standard deviation $\left(\sigma_{t\mu}^{(iter+1)}\right)$(step-length) of elite particles is found by (11):

$$\sigma_{t\mu}^{(iter+1)} = \sqrt{\frac{1}{\mu_e - 1} \sum_{j=1}^{\mu_e} \left(x_{j:N}^{(iter+1)} - m_e^{(iter+1)}\right)^2} \quad (11)$$

6) Step 6: Apply Smoothing of Mean and Standard Deviation of the Whole Population

As elite particles are in the vicinity of optimal solutions, more smoothing (weightage) is applied to their mean value as compared to mean of whole population as per (12):

$$m_e^{(iter+1)} = \alpha m_{t\mu}^{(iter+1)} + (1-\alpha) m_e^{(iter+1)} \quad (12)$$

where $\alpha = 0.9, \beta = 0.1$ are the smoothing parameters.

Similarly, the standard deviation of the elite particles should be updated to a very small extent as they lie near the optimal solutions. So, less smoothing is applied to them as compared to the standard deviation of all particles which requires more exploration of the search space and thus more smoothing as given in (13):

$$\sigma^{(iter+1)} = \beta \sigma_{t\mu}^{(iter+1)} + (1-\beta) \sigma_{tu}^{(iter+1)} \quad (13)$$

7) Step 7: Apply Levy Flight for Global Exploration

Levy flight [18-19] is a random walk whose length is derived from the Levy distribution as described in (14), where u and v are obtained from the normal distribution. Many species (e.g. swordfish and silky sharks) use Levy flights to search for food. The function of Levy step is to efficiently explore the search space by taking a large step size during the intermediate iterations to obtain the global optimum solution.

$$\text{Levy-step} = \frac{u}{|v|^{1/\beta}} \quad (14)$$

where:

$$u = \text{rand}(0,1)\text{*Sigma} \quad (15)$$

$$v = \text{rand}(0,1) \quad (16)$$

$$\text{Sigma} = \left\{ \frac{\Gamma(1+\beta) * \sin(\Pi * \beta)}{\Gamma[(1+\beta)/2] * \beta * 2^{(\beta-3)}} \right\}^{1/\beta} \quad (17)$$

where β (=3/2) is a Levy coefficient and:

$$m_e^{(iter+1)} = m_e^{(iter+1)} + Levy - step \quad (18)$$

8) *Step 8: Increment of Iteration Count*

Set $iter := iter + 1$ and go to Step 2 until the maximum number of iterations is reached.

The detail flowchart of the LE-CE method to train FNNs is shown in Figure 1.

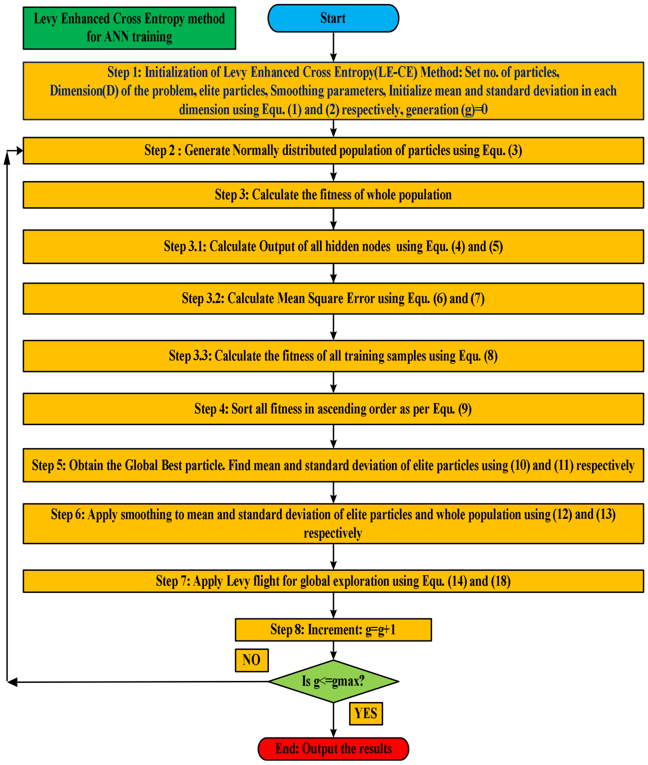
Levy Enhanced Cross Entropy method for ANN training
Start
Step 1: Initialization of Levy Enhanced Cross Entropy(LE-CE) Method: Set no. of particles, Dimension(D) of the problem, elite particles, Smoothing parameters, Initialize mean and standard deviation in each dimension using Equ. (1) and (2) respectively, generation (g)=0
Step 2 : Generate Normally distributed population of particles using Equ. (3)
Step 3: Calculate the fitness of whole population
Step 3.1: Calculate Output of all hidden nodes using Equ. (4) and (5)
Step 3.2: Calculate Mean Square Error using Equ. (6) and (7)
Step 3.3: Calculate the fitness of all training samples using Equ. (8)
Step 4: Sort all fitness in ascending order as per Equ. (9)
Step 5: Obtain the Global Best particle. Find mean and standard deviation of elite particles using (10) and (11) respectively
Step 6: Apply smoothing to mean and standard deviation of elite particles and whole population using (12) and (13) respectively
Step 7: Apply Levy flight for global exploration using Equ. (14) and (18)
Step 8: Increment: g=g+1
NO
Is g<=gmax?
YES
End: Output the results

Fig. 1. Flowchart of the LE-CE method for training FNNs.

III. ANN ENCODING STRATEGY

Figure 2 shows the typical structure of an ANN. It consists of 2 input nodes, 3 hidden nodes, and 1 output node. For the training of ANN, the most popular matrix encoding method is used in this paper as follows:

$$particle(:,i) = \left[w_{1,} \; B_1, w_2^T, B_2 \right] \quad (20)$$

$$w_1 = \begin{bmatrix} w_{13} & w_{23} \\ w_{14} & w_{24} \\ w_{15} & w_{25} \end{bmatrix}, B_1 = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}, w_2^T = \begin{bmatrix} w_{36} \\ w_{46} \\ w_{56} \end{bmatrix}, B_2 = \left[\theta_4 \right] \quad (21)$$

where w_1 is the weight matrix of the hidden layer, B_1 is the bias matrix of the hidden layer, w_2 is the weight matrix of the output layer, w_2^T is the transpose of the w_2 matrix, and B_2 is the bias matrix of the hidden layer.

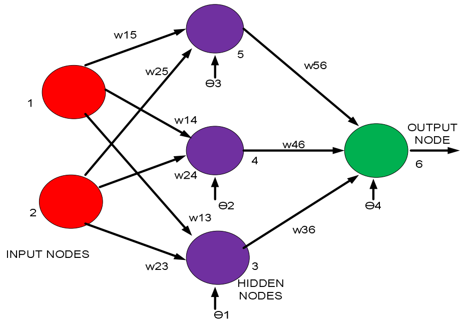

Fig. 2. ANN typical structure.

IV. SIMULATION RESULTS AND DISCUSSION

The performance of the proposed LE-CE algorithm is not tested on a small system because, as per the no-free lunch theorem [20], the average performance of all the optimization methods remains the same for small test systems which consist of a small number of control variables. So, in order to check the effectiveness of the proposed LE-CE algorithm, it is tested on the practical Iris classification [21] problem and its output results, convergence, and statistical results are compared with WCCI 2018 international award winning EVDEPSO [15] and GECCO 2019 international award winning HL_PS_VNSO [16] computational intelligence methods. Both these methods had secured 2nd ranks in the aforementioned conference competitions.

EVDEPSO (Enhanced Velocity Differential Evolution Particle Swarm Optimization), is a meta-heuristic iterative method which has been inspired from the behavior of bird flocking and fish schooling. Mathematically, each bird represents the prospective solution of the optimization problem and its position is adjusted depending upon the position of the best bird and their past best positions in every iteration to obtain optimal solutions. The process is continued until no significant improvement in the optimal solutions is observed. The detail theory of EVDEPSO is available in [15]. HL_PS_VNSO (Hybrid Levy Particle Swarm Variable Neighbourhood Search Optimization) algorithm is a hybridization of PSO and variable neighborhood search algorithm. Its key elements are Perception, Cooperation, and the Levy step. It consists of a Perception term for the local exploitation of search space in which a particle follows its personal best position, a Cooperation term in which the particle follows the global best particle with mutated weights, and the Levy step to globally explore the search space. Its detail theory is given in [16].

A. *Practical Iris Classification Problem*

The Iris classification problem was used in [21]. The Iris dataset contains 4 features (length and width of sepals and petals) of 3 species of Iris (Iris setosa, Iris virginica, and Iris versicolor) as shown in [22, 23]. The data contains 150 samples. These measures were used initially to create a linear model to classify the species in machine learning. Different hidden nodes such as *H*=4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15 were set to test the performance of the algorithms. The simulation results have been performed on Matlab 2017a environment, Intel CORE i5 with 16 GB RAM. All tested methods find the optimal combinations of the weights and biases which results into minimum error of the FNN. A total of 20 trials were executed as the LE-CE is a meta-heuristic method and as a result in every simulation run it yields different optimal solutions. Finally, the mean recognition rate was obtained from the results of the 20 trials. Table I shows that the proposed LE-CE outperforms the other methods in recognizing the output correctly with different hidden nodes in all cases, due to fact that LE-CE method's mean and standard deviation updating using smoothing parameters yield better solutions as the elite particles are in the vicinity of the optimal solution (as per (12) and (13)). Also, CE is powered by Levy flight, which has the ability to take larger step size during optimization. As a result, the search space is being explored more efficiently. So, LE-CE yields better solutions as compared to original CE method and other tested algorithms.

TABLE I. MEAN RECOGNITION RATE WITH DIFFERENT HIDDEN NODES

Hidden nodes (H)	LE-CE (%)	CE (%)	EVDEPSO (%)	HL_PS_VNSO (%)
4	**99.92**	98.32	90.12	89.14
5	**100**	97.25	91.18	87.17
6	**99.45**	95.36	92.78	89.35
7	**100**	96.65	90.07	90.14
8	**99.32**	93.45	91.95	87.88
9	**100**	92.32	90.45	83.41
10	**99.12**	93.85	88.35	87.12
11	**99.85**	92.65	91.98	83.45
12	**100**	96.36	89.45	76.3
13	**99.85**	94.36	90.42	86.32
14	**100**	93.95	88.01	83.25
15	**99.36**	94.15	91.05	87.05

It is cleared from Table II that the proposed LE-CE outperforms the other methods in all 12 cases for average MSE, std. dev MSE, and best MSE. Hidden nodes are increased from 4 to 15 in each case. As a result, the complexity of the ANN increases as more mathematical equations have to be solved by the algorithm. Also, the convergence characteristics of LE-CE for different hidden nodes are better than the compared algorithms' as shown in Figures 3-5, because, unlike other meta-heuristic methods, the LE-CE method is a tuning free algorithm and its adaptive mean and standard deviation updating strategy makes it quite suitable to obtain optimal solutions. Traditional CE has not a Levy flight step, so its performance remains inferior to LE-CE. EVDEPSO's and HLPSVNSO's performance is worse that LE-CE's because both these methods have a large number of parameters to be tuned which affects convergence, whereas LE-CE has a very small number of parameters to be tuned and it searches solutions by updating the mean and standard deviation of elite particles. As a result, it can achieve better solutions as compared to the other tested algorithms.

Figures 3-5 clearly show that the LE-CE has better convergence characteristics as compared to the other methods for the set termination criteria. The other methods have many step length updating operators, which increase the computational burden on the algorithm.

TABLE II. AVERAGE, STANDARD DEVIATION, AND BEST MSE IN 20 INDEPENDENT RUNS WITH DIFFERENT HIDDEN NODES

Hidden nodes (*H*)	Algorithm	Average MSE	Std dev MSE	Best MSE
4	LE-CE	1.9021e—02	1.1905e—03	1.3538e—02
	CE	2.1926e—02	2.0907e—03	1.4538e—02
	EVDEPSO	2.7048e—02	2.0825e—02	7.0291e—03
	HL_PS_VNSO	4.1936e—02	1.0413e—02	4.0013e—02
5	LE-CE	1.8127e—02	1.6252e—03	1.3085e—02
	CE	1.9457e—02	1.7267e—03	1.6525e—02
	EVDEPSO	2.4756e—02	1.7638e—02	1.3629e—02
	HL_PS_VNSO	4.4355e—02	1.1071e—02	4.0770e—02
6	LE-CE	1.3593e—02	2.2182e—03	1.2013e—02
	CE	1.6932e—02	2.0082e—03	1.5113e—02
	EVDEPSO	1.8453e—02	7.2653e—03	1.6253e—02
	HL_PS_VNSO	1.0441e—01	1.0374e—01	4.4510e—02
7	LE-CE	1.3682e—02	1.2446e—03	1.2128e—02
	CE	2.6122e—02	1.0746e—03	2.4358e—02
	EVDEPSO	1.7691e—02	4.0023e—03	1.4523e—02
	HL_PS_VNSO	5.5226e—02	6.6222e—03	5.0238e—02
8	LE-CE	1.7042e—02	3.2045e—03	1.2511e—02
	CE	1.8142e—02	6.2545e—03	1.3011e—02
	EVDEPSO	2.1454e—02	6.0198e—03	2.0378e—02
	HL_PS_VNSO	4.7541e—02	7.2863e—03	3.8607e—02
9	LE-CE	1.3294e—02	3.2993e—03	1.0070e—02
	CE	1.6302e—02	6.5093e—03	1.0670e—02
	EVDEPSO	5.0615e—02	1.2549e—02	5.0301e—02
	HL_PS_VNSO	3.8314e—02	1.3757e—02	3.0293e—02
10	LE-CE	1.2521e—02	1.0021e—03	1.1021e—02
	CE	1.8221e—02	1.1821e—03	1.4321e—02
	EVDEPSO	3.9265e—02	1.3342e—02	3.0427e—02
	HL_PS_VNSO	4.4245e—02	1.3204e—02	4.3230e—02
11	LE-CE	1.2646e—02	1.8009e—03	1.0840e—02
	CE	1.5246e—02	2.1709e—03	1.0880e—02
	EVDEPSO	1.4203e—02	1.2120e—02	1.0393e—03
	HL_PS_VNSO	4.1536e—02	6.7323e—03	3.7538e—02
12	LE-CE	1.2006e—02	2.2091e—03	1.0049e—02
	CE	1.6806e—02	2.9391e—03	1.3849e—02
	EVDEPSO	1.7889e—02	6.9515e—03	1.3409e—02
	HL_PS_VNSO	1.2098e—01	1.3702e—01	1.0316e—02
13	LE-CE	1.1753e—02	1.3090e—03	1.0021e—02
	CE	2.3453e—02	1.8740e—03	2.2422e—02
	EVDEPSO	1.8013e—02	5.0128e—03	1.5113e—02
	HL_PS_VNSO	3.1724e—01	1.3744e—01	3.0250e—02
14	LE-CE	1.1270e—02	1.0086e—03	1.0592e—02
	CE	2.6250e—02	1.0186e—03	2.3592e—02
	EVDEPSO	1.5241e—02	1.0171e—03	1.1297e—02
	HL_PS_VNSO	4.3257e—02	1.0214e—02	4.1200e—02
15	LE-CE	1.2189e—02	1.3530e—03	1.1509e—02
	CE	3.3589e—02	2.7530e—03	3.1725e—02
	EVDEPSO	2.145e—02	2.5846e—03	2.1153e—02
	HL_PS_VNSO	5.3539e—02	6.2441e—02	5.2546e—02

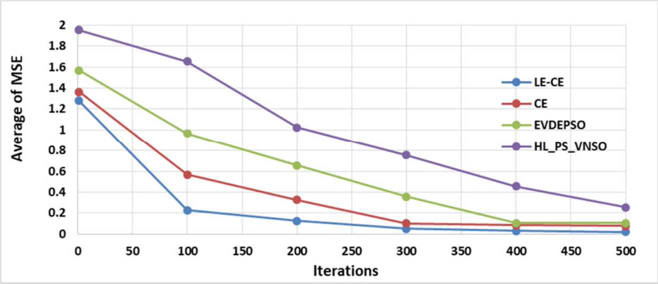

Fig. 3. Convergence curve with 5 hidden nodes.

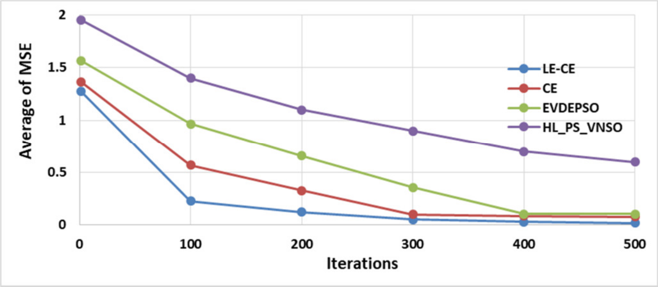

Fig. 4. Convergence curve with 11 hidden nodes.

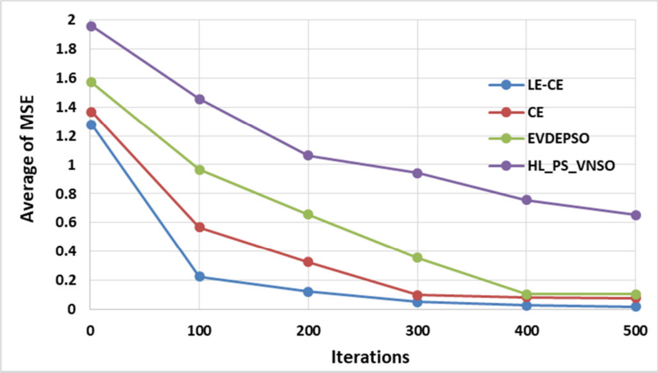

Fig. 5. Convergence curve with 15 hidden nodes.

V. Statistical Analysis

A. *One-Way ANOVA Statistical Test*

The ANOVA statistical test [24] is used to verify whether the MSE of all algorithms evaluated for each simulation run shows any significant difference. In this test, the degree of significance is set to 0.05 to check the statistical difference between the tested algorithms. During the comparison, if it is found that the P-value is less than 0.05, then it inferred that all tested algorithms are substantially different from one another. From Table III, it is seen that the F-ratio value is 6.75541. The P-value is 0.000758. So, the result is significant at $p<0.05$. This implies that at least one of the means of the groups is significantly different from the others. However, it is not known which group(s) contribute to this difference, hence Tukey's Honestly Significant Difference (HSD) test was carried out.

B. *Tukey's Hones Honestly Significant Difference Test*

To further check the statistical difference between two algorithms, the pairwise comparison test entitled Tukey's HSD test [25] was carried out. In this test, the first step is to find the critical value (Q_{crit}) from the studentized range distribution table [26] based on the a=4 treatments (algorithms) and DF=44 Degrees of Freedom. The critical value obtained from the studentized range distribution table is 3.777. Then, the $Q_{i,j}$ is calculated as per (22) for all the pairwise comparisons with the proposed algorithm. The result values are given in Table IV.

$$Q_{i,j} = \frac{\left|\overline{y_i} - \overline{y_j}\right|}{\sqrt{\frac{(MS)}{N_{Runs}}}} \qquad (22)$$

where i, j = 1,..a, $i \neq j$. is the difference between the optimal fitness values of the compared pair of algorithms.

TABLE III. RESULTS OF THE ONE-WAY ANOVA STATISTICAL TEST

Variation	Sum of square due to source (SS)	DF	Mean sum of square due to source (MS)	F ratio value	P-value	F crit
Between algorithms	0.03278	3	0.0109	6.7554	0.00075	2.816
Within algorithms	0.07117	44	0.0016			
Total	0.10395	47				

TABLE IV. TUKEY HSD TEST RESULTS

Pairwise comparisons	Q (Statistic)
LE-CE and CE	0.61
LE-CE and EVDEPSO	0.85
LE-CE and HL_PS_VNSO	5.64

Table IV reveals that for all pairwise treatments, the proposed algorithm is substantially statistically different from the other algorithms.

The algorithm has been developed in Matlab environment and the source codes can be provided to the enthusiastic learner by the main author upon request.

VI. Conlcusion

The Levy Enhanced Cross Entropy method to train feedforward neural network has been proposed in this paper. The LE-CE method has less parameters to be tuned and its adaptive updating for mean and standard deviation of the solutions make it quite powerful in terms of local exploitation and global exploration of the solution space. To further improve its global search exploration, the CE method is powered by the Levy flight. The simulation on the practical Iris test system reveals that the proposed LE-CE method outperforms the contemporary optimization methods in terms of solution quality, iterations, average MSE, standard deviation MSE, and best MSE. The proposed method is confirmed to be statistically different from the other compared optimization methods. The proposed method can be used in highly complex nonlinear ANN systems. In the future, the same algorithm will be applied to train more realistic training sets of neural networks with hundreds of variables and constraints.

REFERENCES

[1] C.-J. Lin, C.-H. Chen, and C.-Y. Lee, "A self-adaptive quantum radial basis function network for classification applications," in *International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, Budapest, Hungary, Jul. 2004, vol. 4, pp. 3263–3268, https://doi.org/10.1109/IJCNN.2004.1381202.

[2] S. Mirjalili, S. Z. Mohd Hashim, and H. Moradian Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11125–11137, Jul. 2012, https://doi.org/10.1016/j.amc.2012.04.069.

[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989, https://doi.org/10.1016/0893-6080(89)90020-8.

[4] J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1026–1037, Feb. 2007, https://doi.org/10.1016/j.amc.2006.07.025.

[5] M. Gori and A. Tesi, "On the Problem of Local Minima in Backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76–86, Jan. 1992, https://doi.org/10.1109/34.107014.

[6] L. V. Kantorovich, "Functional analysis and applied mathematics," *Uspekhi Matematicheskikh Nauk*, vol. 3, no. 6, pp. 89–185, 1948.

[7] E. A. Ogbonnaya, E. M. Adigio, H. U. Ugwu, and M. C. Anumiri, "Advanced Gas turbine rotor shaft fault diagnosis using artificial neural network," *International Journal of Engineering and Technology Innovation*, vol. 3, no. 1, pp. 58–69, 2013.

[8] R. Kaluri and P. Reddy CH, "Optimized feature extraction for precise sign gesture recognition using self-improved genetic algorithm," *International Journal of Engineering and Technology Innovation*, vol. 8, no. 1, pp. 25–37, 2018.

[9] M. Njah and R. E. Hamdi, "A Constrained Multi-Objective Learning Algorithm for Feed-Forward Neural Network Classifiers," *Engineering, Technology & Applied Science Research*, vol. 7, no. 3, pp. 1685–1693, Jun. 2017, https://doi.org/10.48084/etasr.968.

[10] D. N. Truong and V. T. Bui, "Hybrid PSO-Optimized ANFIS-Based Model to Improve Dynamic Voltage Stability," *Engineering, Technology & Applied Science Research*, vol. 9, no. 4, pp. 4384–4388, Aug. 2019, https://doi.org/10.48084/etasr.2833.

[11] L. T. H. Nhung, T. T. Phung, H. M. V. Nguyen, T. N. Le, T. A. Nguyen, and T. D. Vo, "Load Shedding in Microgrids with Dual Neural Networks and AHP Algorithm," *Engineering, Technology & Applied Science Research*, vol. 12, no. 1, pp. 8090–8095, Feb. 2022, https://doi.org/10.48084/etasr.4652.

[12] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, Jun. 2009, https://doi.org/10.1016/j.ins.2009.03.004.

[13] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997, https://doi.org/10.1023/A:1008202821328.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, Perth, WA, Australia, Dec. 1995, vol. 4, pp. 1942–1948, https://doi.org/10.1109/ICNN.1995.488968.

[15] D. Dabhi and K. Pandya, "Enhanced Velocity Differential Evolutionary Particle Swarm Optimization for Optimal Scheduling of a Distributed Energy Resources With Uncertain Scenarios," *IEEE Access*, vol. 8, pp. 27001–27017, 2020, https://doi.org/10.1109/ACCESS.2020.2970236.

[16] D. Dabhi and K. Pandya, "Uncertain Scenario Based MicroGrid Optimization via Hybrid Levy Particle Swarm Variable Neighborhood Search Optimization (HL_PS_VNSO)," *IEEE Access*, vol. 8, pp. 108782–108797, 2020, https://doi.org/10.1109/ACCESS.2020.2999935.

[17] R. Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, May 1997, https://doi.org/10.1016/S0377-2217(96)00385-2.

[18] C. T. Brown, L. S. Liebovitch, and R. Glendon, "Levy Flights in Dobe Ju/'hoansi Foraging Patterns," *Human Ecology*, vol. 35, no. 1, pp. 129–138, Feb. 2007, https://doi.org/10.1007/s10745-006-9083-4.

[19] X. S. Yang, "Random Walks and Levy Flights," in *Nature-Inspired Metaheuristic Algorithms*, 2nd ed., Cambridge, UK: Luniver Press, 2010, pp. 11–19.

[20] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997, https://doi.org/10.1109/4235.585893.

[21] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936, https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

[22] "The Iris Dataset," *Gist*. https://gist.github.com/curran/a08a1080b88344b0c8a7.

[23] K. Thirunavukkarasu, A. S. Singh, P. Rai, and S. Gupta, "Classification of IRIS Dataset using Classification Based KNN Algorithm in Supervised Learning," in *4th International Conference on Computing Communication and Automation*, Greater Noida, India, Dec. 2018, pp. 1–4, https://doi.org/10.1109/CCAA.2018.8777643.

[24] E. Ostertagova and O. Ostertag, "Methodology and Application of Oneway ANOVA," *American Journal of Mechanical Engineering*, vol. 1, no. 7, pp. 256–261, Jan. 2013, https://doi.org/10.12691/ajme-1-7-21.

[25] H. Abdi and L. J. Williams, "Tukey's Honestly Significant Difference (HSD) Test," in *Encyclopedia of Research Design*, Thousand Oaks, CA, USA: SAGE, 2010.

[26] H. L. Harter, "Critical Values for Duncan's New Multiple Range Test," *Biometrics*, vol. 16, no. 4, pp. 671–685, 1960, https://doi.org/10.2307/2527770.