A Clustering-based Differential Evolution Boosted by a Regularisation-based Objective Function and a Local Refinement for Neural Network Training

Seyed Jalaleddin Mousavirad
Computer Engineering Department
Hakim Sabzevari University
Sabzevar, Iran

Amir H. Gandomi
Faculty of Engineering and IT
University of Technology Sydney
Ultimo, NSW 2007, Australia

Hassan Homayoun
Quantitative MR Imaging and Spectroscopy Group
Research Center for Cellular and Molecular Imaging
Tehran University of Medical Sciences
Tehran, Iran

***Abstract*—The performance of feed-forward neural networks (FFNN) is directly dependant on the training algorithm. Conventional training algorithms such as gradient-based approaches are so popular for FFNN training, but they are susceptible to get stuck in local optimum. To overcome this, population-based metaheuristic algorithms such as differential evolution (DE) are a reliable alternative. In this paper, we propose a novel training algorithm, Reg-IDE, based on an improved DE algorithm. Weight regularisation in conventional algorithms is an approach to reduce the likelihood of over-fitting and enhance generalisation. However, to the best of our knowledge, the current DE-based trainers do not employ regularisation. This paper, first, proposes a regularisation-based objective function to improve the generalisation of the algorithm by adding a new term to the objective function. Then, a region-based strategy determines some regions in search space using a clustering algorithm and updates the population based on the information available in each region. In addition, quasi opposition-based learning enhances the exploration of the algorithm. The best candidate solution found by improved DE is then used as the initial network weights for the Levenberg-Marquardt (LM) algorithm, as a local refinement. Experimental results on different benchmarks and in comparison with 26 conventional and population-based approaches apparently demonstrate the excellent performance of Reg-IDE.**

***Index Terms*—Neural networks, differential evolution, regularisation, Levenberg-Marquardt algorithm, clustering, opposition-based learning.**

I. Introduction

Feed-forward neural networks (FFNN) are one of the most popular architectures in artificial neural networks (ANN) to tackle complex classification and regression problems. FFNNs consist of simple components called neurons as well as connections among them. In FFNN, inputs move in one direction and pass, through hidden layers, to the output layer. Each connection benefits from one weight, representing its strength. Training in FFNNs is to find proper weights so that the error between the actual and predicted outputs is minimised. Gradient-based approaches such as back-propagation algorithm are so popular in the literature, while they have a tendency towards local optimum [1].

Population-based metaheuristic (PBMH) algorithms such as differential evolution(DE) [2] and particle swarm optimisation (PSO) [3] are a reliable alternative to tackle the problems of conventional algorithms. Evolutionary algorithms (EA), as a category of PBMHs, has been extensively used for FFNN training. [4] compared BP and genetic algorithm (GA) for FFNN training and indicated that GA is superior in terms of effectiveness, ease-of-use and efficiency. In another study, [5] employed a modified GA for fast training FFNNs. Their results based on computational time indicate that the proposed algorithm is more efficient than the conventional GA-based training algorithm. [6] proposed a combination of GA and BP for finding the weights in FFNN and showed that it can outperform both GA and BP.

Another category of PBMHs is swarm intelligence algorithms. [7] proposed a hybrid approach based on accelerated PSO using Levenberg Marquardt to obtain a faster convergence rate. [8] suggested an opposition PSO-based training for medical datasets. Their experiments on several clinical datasets verified its performance. Other PBMHs which have been employed for FFNN training include artificial bee colony [9], imperialist competitive algorithm [10], [11], firefly algorithm [12], grey wolf optimiser (GWO) [13], [14], ant lion optimiser [15], dragonfly algorithm (DA) [16], sine cosine algorithm [17], whale optimisation algorithm (WOA) [18], grasshopper optimisation algorithm [19], and salp swarm algorithm (SSA) [20], among others.

Differential evolution(DE) [2] is a well-established and effective PBMH that has indicated excellent performance in solving complex optimisation problems [21]–[25], and many studies have been done on improving DE algorithm in recent years [26], [27]. It takes advantage of three main operators, including, mutation, crossover, and selection. Mutation combines information among different candidate solutions, while the responsibility of crossover is the integration of mutant vector and target vector. Finally, the selection operator chooses a better candidate solution between the old and new candidate

This work is done within 2020-2021. On the date of publication of final version, Seyed Jalaleddin Mousavirad is affiliated with Universidade da Beira Interior, Covilhã, Portugal, and is working on GreenStamp project.

solutions to include the current population. [28] proposed a DE algorithm for FFNN training, and indicated that it can provide better performance than gradient-based methods. In another work, [29] employed opposition-based learning in combination with DE (QODE) and indicated that QODE has a satisfactory performance compared to the competitors on different classification problems. In one of the most recent works, [30] employed a DE algorithm that is improved by opposition-based learning and a region-based strategy (RDE-OP).

One of the main features of FFNN is the ability to generalise. It has indicated that the generalisation ability of conventional neural networks can be enhanced by regularisation [31], a method to penalise network complexity by adding a penalty term to the loss function. Despite the efficiency of regularisation in improving the performance of conventional FFNNs, the applicability of the same technique to PBMHs is less explored [32], [33]. To the best of our knowledge, there is no research on the use of regularisation in DE-based FFNN trainers.

In this paper, we propose a novel DE algorithm, Reg-IDE, for FFNN training. The main characteristics of Reg-IDE are as follows.

1) Reg-IDE introduces a new objective function for DE-based trainer. It adds a regularisation term to increase the generalisation.
2) Reg-IDE employs the Levenberg-Marquardt algorithm as a local refinement to enhance the exploitation of the algorithm.
3) Reg-IDE uses a clustering algorithm to update the population. It acts like a multi-parent crossover.
4) Reg-IDE adds quasi opposition-based learning to the trainer for enhancing exploration.

Experimental results on different datasets and in comparison with 26 conventional and population-based approaches apparently demonstrate the supreme performance of Reg-IDE.

The remainder of this paper is organised as follows. Section II briefly explains some background knowledge on neural networks, regularisation, and DE. Section III first introduces the main components of Reg-IDE and then the general structure. Section IV evaluates Reg-IDE on different benchmark problems. Finally, Section V concludes the paper.

II. Background knowledge

A. Neural networks

Feedforward neural networks (FFNN), as a popular family of ANNs, are a supervised pattern recognition approach which have extensively used in a wide range of applications [34], [35]. They consist of three main layers, including, input layer, output layer, and hidden layers. The general architecture of an FFNN is shown in Figure 1. In this figure, $(x_1, x_2, ..., x_n)$ are input features, while $(O_1, O_2, ..., O_m)$ are outputs. In addition, each node has an activation function, which shows how the weighted sum of the inputs should transfer into the output. This paper employs the Sigmoid function, one of the most popular ones, defined as

$$\delta(net) = \frac{1}{1 + e^{-net}}. \quad (1)$$

where net is the input. Each link between different layers has a weight, representing strength between two nodes. Weights play a crucial role in the performance of a FFNN. Therefore, finding proper values for weights, or training, is considered as one of the most important, yet challenging, tasks in FFNNs. Gradient descent-based approaches (GD) are the most popular method for training the weights.

B. Weight regularisation

Training in an FFNN that can generalise well to new data is a complex task. An FFNN with too few neurons, and consequently connections, can not learn the relationship between inputs and outputs well, while too many neurons can lead to over-fitting. In both cases, FFNN can not generalise well. Broadly speaking, regularisation points out to any alteration in the training algorithm to reduce generalisation error. L1 and L2 regularisation techniques [36] are among the most common types of approaches. They add a regularisation term in the loss function of gradient-based approaches. It leads to a decrease in the connection weights, and consequently, a simpler model. Therefore, it can reduce over-fitting in a typical model.

C. Differential evolution

Differential Evolution (DE) [2] is a simple, yet effective, population-based metaheuristic algorithms, which indicates a superior performance in solving complex optimisation problems. DE begins with N_P candidate solutions generated randomly drawn from a uniform distribution and takes advantages of three primary operators for updating its population, including, mutation, crossover, and selection. Mutation generates a mutant vector, $v_i = (v_{i,1}, v_{i,2}, ..., v_{i,D})$, as

$$v_i = x_{r1} + F * (x_{r2} - x_{r3}), \quad (2)$$

where x_{r1}, x_{r2}, and x_{r3} are three distinct randomly selected candidate solutions from the current population, whereas F is the scale factor.

Crossover is responsible to incorporate the mutant vector with the target vector. Binomial crossover is used in this paper, which is defined as

$$u_{i,j} = \begin{cases} v_{i,j} & rand(0,1) \leq CR \text{ or } j == j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}, \quad (3)$$

where $i = 1, ..., NP$, $j = 1, ..., D$, CR means the crossover rate, and j_{rand} is a random number between 1 and N_P. the last operator is called selection, which select the better individual from the trial and target vectors.

III. Reg-IDE algorithm

This paper proposes a novel training algorithm, Reg-IDE, for finding the optimal weights in an FFNN. To this end, first, a regularisation term is added to the objective function to improve generalisation. Also, Reg-IDE benefits from a local refinement as a local search in the final phase, a region-based

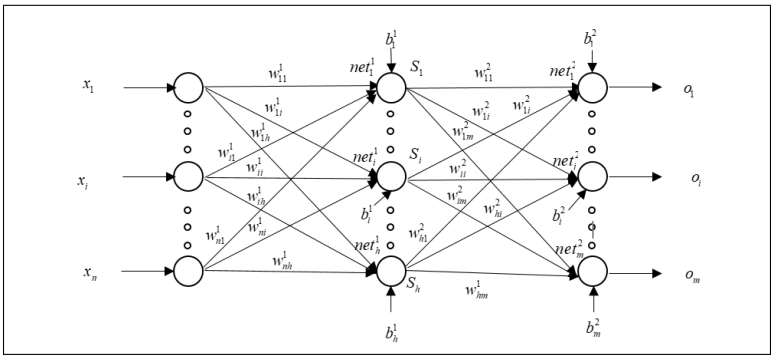

Fig. 1. General architecture of an FFNN network.

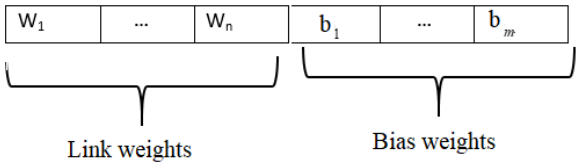

Fig. 2. Representation of a candidate solution in Reg-IDE algorithm.

strategy to increase the exploitation of the algorithm, and opposition-based learning for enhancing the exploration. In the following, first, we explain the main components of the Reg-IDE algorithm. Then, we detail how it proceeds.

A. Representation

Our approach benefits from a one-dimensional array to encode the weights and biases of FFNN as indicated in Figure 2. The length of the array is equal to the total number of weights and biases.

B. Regularisation-based objective function

This paper introduces a novel objective function based on regularisation for FFNN training. To the best of our knowledge, all objective functions for DE-based FFNN training has used error between the actual and predicted outputs [9], [14], [37]. In this paper, we propose to add a regularisation term to the objective function as

$$f = \frac{100}{P}\sum_{p=1}^{P}\xi(x_p) + \frac{\lambda}{2m}\sum||W||^2, \tag{4}$$

with

$$\xi(x_p) = \begin{cases} 1 & \text{if } o_p \neq d_p \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where d_p and o_p means the actual and predicted outputs, respectively, and m is the total number of samples. λ is called regularisation parameter which can be considered as a hyper-parameter. Regularisation term can penalize large value of the weights and biases.

If λ is too large to be selected, a lot of weights will be close to zeros which will make the FFNN simpler and prone to under-fitting. In contrast, If λ is selected too small, in practice, the role of regularisation term will be eliminated. If λ is selected good enough, it will deteriorate some weights that can overcome over-fitting.

C. Levenberg-Marquardt algorithm

Reg-IDE employs Levenberg-Marquardt(LM) algorithm [38], [39] as a local refinement in the last phase. The LM algorithm is one of the most effective conventional training algorithms for FFNNs. It commences with random weights and aims to mitigate the error function by modifying the network weights as

$$w_{t+1} = w_t - (J_t^T J_t + \mu I)^{-1} J_t^k E_t, \tag{6}$$

with

$$E_t = \sum_{i=1}^{N}(d_i - y_i)^2, \tag{7}$$

where J means the Jacobian matrix of the error vector E_t, J^T shows its transpose, I is the identity matrix of the same dimensions as the Hessian $J^p J$, N signifies the number of instances, d_i is the actual output, and y_i the predicted output. Also, $J^k E$ is the gradient of the error function E, and μ indicates a dumping factor that is altered during the optimisation process. It is worthwile to mention that the LM method reaches the the optimum faster in comparison to other conventional algorithms such as back-propagation with momentum [40], [41].

D. Region-based strategy

Reg-IDE benefits from a clustering algorithm to construct the regions. To this end, we employed k-means algorithm [42] as a popular clustering algorithm. The number of clusters is determined as a random number between 2 and $\sqrt{N_P}$. Cluster centres can be considered as a multi-parent crossover since the cluster centre is the total of candidate solutions located in one cluster. Reg-IDE employs cluster centres for updating the current population using a generic population-based algorithm (GPBA) proposed in [43]. Population-updating in Reg-IDE is based on a GPBA strategy as:

- **Selection:** randomly choose some candidate solutions from the current population. This corresponds to choosing initial points in the k-means algorithm.
- **Generation**: generate m candidate solutions (set A). Reg-IDE creates the new candidate solutions using the k-means algorithm, and each cluster centre determines a new candidate solution.
- **Substitution**: select m candidate solutions (set B) from the current population for substitution. Reg-IDE chooses m randomly selected candidate solution.
- **Update**: from the combination set $A \cup B$, choose the m best candidate solutions as $\bar{B}$. The new population is then achieved as $(P - B) \cup \bar{B}$.

Reg-IDE does not employ clustering algorithm in each iteration. In other words, clustering algorithms periodically is done, based on a parameter called clustering period, similar to the scheme proposed in [30], [44].

E. Opposition-based strategy

To further improvement, Reg-IDE employs an opposition-based learning (OBL) [45] for initialisation and updating. The opposition number of x is defined as

$$\bar{x_i} = a_i + b_i - x_i, \tag{8}$$

where a and b is the lower and upper bounds, respectively, and i means the i-th dimension of x. This paper uses quasi OBL(QOBL) strategy [46], rather than standard OBL, since quasi-opposition numbers are more likely to converge to the solution [46]. Quasi-opposition number is defined as

$$\breve{x_i} = rand\left[\frac{a_i + b_i}{2}, a_i + b_i - x_i\right], \tag{9}$$

where $rand[m, n]$ creates a uniform random number in the range of m and n.

In the initialisation step, Reg-IDE, first, creates a population Pop randomly, and then, a corresponding population, $Opop$ is generated using the QOBL strategy. Finally, the best candidate solutions are selected based on the combination of Pop and $OPop$.

A similar mechanism is employed in the updating step. After region-based strategy, a new population $Opop$ is generated using QOBL strategy and with a probability (jumping rate) J_r between 0 and 0.4 [46]. Afterwards, the best candidate solutions among the combination of Pop and $OPop$ form the new population.

F. Algorithm

Our proposed algorithm, Reg-IDE, proposes weight regularisation for DE-based FFNN training, which is booted by a local refinement, a region-based strategy, and quasi-opposition based learning. Reg-IDE proceeds in the following steps:

1) Initialise the parameters including population size N_{pop}, the maximum number of function evaluations $NFE_{\max}$, the maximum number of iterations for the LM algorithm $iter_{\max}$, the jumping rate J_r, clustering period C_P, and regularisation parameter λ . Set the current number of function evaluations $NFE = 0$, and the current iteration $iter = 1$.
2) Generate the initial population (Pop) using uniformly distributed random numbers.
3) Calculate the objective function of each candidate solution in Pop using Eq. (4).
4) Generate a new population ($OPop$) using QOBL strategy.
5) Calculate the objective function of each candidate solution in $OPop$ using Eq. (4).
6) Select N_P best candidate solutions from $Pop \cup OPop$ as initial population.
7) Set $NFE = NFE + 2 \times N_{pop}$
8) For each candidate solution, perform Steps 9 to 12.
9) Apply mutation operator.
10) Apply mutation operator.
11) Calculate the objective function using Eq. (4)
12) Apply selection operator.
13) Set $NFE = NFE + N_{pop}$.
14) If $rem(iter, C_P == 0)$, go to Step 15, otherwise go to Step 20.
15) Randomly generate k as random number between 2 and $\sqrt{N_P}$.
16) Perform k-means clustering and select cluster centres as set A.
17) Select k individuals randomly from current population as set B.
18) From $A \cup B$, select best k individuals as $\bar{B}$.
19) Select new population as $(P - B) \cup \bar{B}$.
20) If $rand(0, 1) < J_r$, go to Step 21, otherwise go to Step 25.
21) Generate a new population ($OPop$) using QOBL strategy.
22) Calculate the objective function of each candidate solution in $OPop$ using Eq. (4).
23) Select N_P best candidate solutions from $Pop \cup OPop$ as the current population.
24) Set $NFE = NFE + N_{pop}$.
25) Set $iter = iter + 1$.
26) If $NFE > NFE_{\max}$ go to Step 27, otherwise go to Step 8.
27) Initialise ω (initial weights for LM algorithm) as the best candidate solution in the current population and set current iteration $iter = 0$.
28) Compute the Jacobian J, the approximated Hessian $J^T J$, and the error E_t.
29) Update the weights using Equ. (6).
30) Recalculate E_t.
31) if $iter < iter_{\max}$ go to Step 27, otherwise the algorithm has terminated.

IV. Experimental results

To evaluate our proposed Reg-IDE algorithm, we fulfil some experiments on different datasets with diverse characteristics

from the UCI machine learning repository[1], namely

- *Iris*: this dataset is one of the most popular classification datasets in the literature. It has 150 samples and 4 features, placed in 3 classes.
- *Breast Cancer*: this dataset includes 699 samples, 9 features, and 2 classes.
- *Liver*: this binary clinical dataset from BUPA Medical Research Ltd. includes 345 instances and 7 features
- *Pima*: this clinical classification dataset is considered a challenging problem with 768 samples, 2 classes, and 8 features.
- *Seed*: this agricultural dataset has seven geometrical features of wheat kernels such as area, perimeter, asymmetry coefficient, length of kernel groove, placed in three different categories with 210 samples.
- *Vertebral*: this clinical dataset has bio-mechanical features such as lumbar lordosis angle and sacral slope, belonging to 3 classes with 310 samples.

This paper does not focus on the optimal FFNN architecture. Therefore, we follow [47], [48] and assign the number of neurons in the hidden layer to $2 \times N + 1$ where N is the number of inputs. For evaluation, we employed k-fold cross-validation (k is set to 10), where the dataset is partitioned to k sections, one section for testing and $k-1$ sections for training. This process is repeated k times so that each section is used once as a test dataset.

We compare Reg-IDE with a diverse range of conventional and PBMH algorithms. The number of function evaluations for all PBMHs is set to 25,000 [49], similar to the number of iterations for all conventional algorithms. The population size for all PBMHs is set to 50. For Reg-IDE, the crossover probability, scaling factor, and jumping rate are assumed 0.9, 0.5, and 0.3, respectively. Also, the clustering period and regularisation parameter are selected as 10 and 0.1, receptively. For the other algorithms, we used default parameters values borrowed from the cited publications.

In the first experiment, we compare Reg-IDE with 12 conventional algorithms including gradient descent with momentum backpropagation (GDM) [50], gradient descent with adaptive learning rate backpropagation (GDA) [36], gradient descent with momentum and adaptive learning rate backpropagation (GDMA) [51], conjugate gradient backpropagation with Fletcher-Reeves updates (CG-FR) [52], conjugate gradient backpropagation with Polak-Ribiere updates (CG-PR) [53], [54], conjugate gradient backpropagation with Powell-Beale restarts (CG-PBR) [55], BFGS quasi-Newton backpropagation (BFGS) [56], Levenberg-Marquardt backpropagation (LM) [38], [39], one-step secant backpropagation (OSS) [57], resilient backpropagation (RP) [58], scaled conjugate gradient backpropagation (SCG) [59], and Bayesian regularisation backpropagation(BR) [60]. Table I compares the results based on mean and standard deviation as well as the average rank. It is clear that in all datasets, Reg-IDE achieved the highest classification accuracy, and consequently, the first rank. Therefore, it outperforms other conventional algorithms with a wide margin.

The next experiment compares Reg-IDE with 14 population-based training algorithms, including DE [28], QODE [29], RDE-OP [30], PSO [61], ABC [9], ICA [10], FA [12], GWO [13], ALO [15], DA [16], SCA [62], WOA [18], GOA [19], and SSA [63]. We selected DE for comparison since our algorithm is based on this algorithm. Also, QODE and RDE-OP are selected since both are among the most recent DE-based trainers. In addition, algorithms such as PSO and ABC are among well-established trainers, whilst some others such as WOA and SSA are more recent.

Table II shows the results of Reg-IDE compared to other PBMH-based training algorithms. As it can be seen, Reg-IDE gives the best accuracy in 3 cases , and the second-best performance in 2 cases. Overall, Reg-IDE achieves the best average rank (1.67), while the second average rank is 4.67, indicating a wide margin in accuracy improvement. On the Iris dataset, Reg-IDE achieves the first rank with a classification accuracy enhancement of 1.33% compared to the second-best algorithms. On the Pima dataset, Reg-IDE improves accuracy slightly, more than 0.35%. Also, on seed dataset, Reg-IDE could improve accuracy dramatically, more than 8%. Finally, on Cancer and Liver datasets, Reg-IDE achieves slightly worse results.

In the last experiment, we assess the effect of regularisation on the weights. To this end, a frequency distribution of the weights for the Iris dataset, as a representative, on all folds are provided in Figure 3 and 4. It can be seen that the regularisation term significantly reduces the weight range.

Overall, our proposed RDE-OP algorithm thus provides superlative performance compared to the other 26 training algorithms.

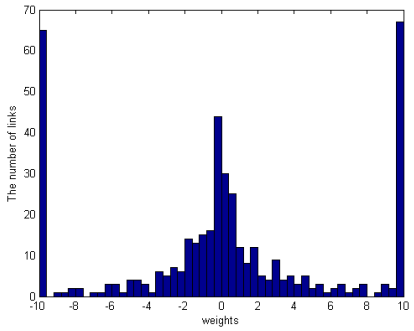

Fig. 3. Frequency distribution of the weights in the proposed algorithm without regularisation term.

V. Conclusion

Training is one of the most important parts of feed-forward neural networks (FFNN) since the performance of FFNNs is directly dependent on the training algorithms. Conventional algorithms such as gradient-based algorithms are extensively used in the literature, but they are prone to some difficulties such as getting stuck in the local optimum. In this paper, we

[1]https://archive.ics.uci.edu/ml/index.php

TABLE I
10CV CLASSIFICATION ACCURACY FOR ALL DATASETS FOR CONVENTIONAL ALGORITHMS IN COMPARISON TO REG-IDE

Algorithms		Iris	Cancer	Liver	Pima	Seed	Vertebral	avg. rank
GDM	mean	92.00	92.99	59.47	67.98	47.62	76.13	
	stddev	8.20	7.60	15.05	13.03	29.95	9.15	
	rank	12	11	12	13	13	12	12.17
GDA	mean	94.67	95.90	58.24	75.91	82.38	80.65	
	stddev	5.26	2.04	6.53	4.52	6.75	5.27	
	rank	10	10	13	9	9.5	10	10.25
GDMA	mean	82.67	90.47	60.66	73.20	80.00	72.90	
	stddev	16.98	5.94	13.99	6.74	16.47	17.62	
	rank	13	13	11	12	12	13	12.33
CG-FR	mean	95.33	96.19	60.86	76.69	86.67	83.87	
	stddev	4.50	1.72	8.29	6.20	9.73	7.60	
	rank	7.5	7	10	5	5.5	4	6.50
CG-PR	mean	96.00	97.07	65.18	75.12	85.24	80.97	
	stddev	6.44	1.39	8.47	3.57	9.90	4.92	
	rank	4	2	8	11	8	8.5	6.92
CG-PBR	mean	94.00	96.49	67.18	76.04	88.10	82.90	
	stddev	5.84	2.95	9.09	5.06	7.86	6.09	
	rank	11	5	3	7	3.5	6	5.92
BFGS	mean	95.33	96.92	65.22	76.70	86.67	84.19	
	stddev	4.50	1.28	7.06	3.11	9.73	5.98	
	rank	7.5	3	7	4	5.5	3	5.00
LM	mean	96.67	96.04	65.55	76.04	88.10	83.55	
	stddev	4.71	2.51	10.44	4.66	7.53	7.04	
	rank	2	9	6	8	3.5	5	5.58
OSS	mean	95.33	96.34	64.89	76.58	86.67	80.97	
	stddev	4.50	2.21	8.01	4.30	5.85	8.93	
	rank	7.5	6	9	6	7	8.5	7.33
RP	mean	95.33	96.05	65.82	76.83	80.48	78.39	
	stddev	5.49	2.47	5.21	4.50	9.38	5.05	
	rank	7.5	8	5	3	11	11	7.58
SCG	mean	96.00	96.63	66.97	78.52	82.38	82.26	
	stddev	8.43	2.09	9.60	3.05	9.54	8.50	
	rank	4	4	4	2	9.5	7	5.08
BR	mean	96.00	91.81	70.71	75.89	90.95	84.52	
	stddev	7.17	3.82	6.93	5.37	7.60	6.94	
	rank	4	12	2	10	2	2	5.33
Reg-IDE	mean	99.33	98.39	76.27	80.60	92.86	86.77	
	stddev	2.11	2.24	4.03	4.15	4.05	5.37	
	rank	1	1	1	1	1	1	1.00

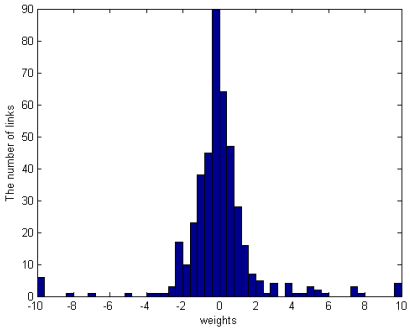

Fig. 4. Frequency distribution of the weights in the proposed algorithm with regularisation term.

have proposed a novel differential evolution (DE) algorithm for FFNN training. To this end, first, we introduce a regularisation-based objective function for the DE-based training algorithm. Then, we employ a Levenberg-Marquardt algorithm, as a local refinement, for enhancing the exploitation. In addition, our proposed algorithm benefits from two other strategies: 1) a region-based strategy to improve exploitation using a clustering algorithm, and 2) quasi opposition-based learning to enhance exploration. Our extensive experiments on different classification problems and in comparison to 26 training algorithms indicate that Reg-IDE is a competitive training algorithm. In the future, we intend to extend the proposed algorithm so that it can find the proper architecture as well as hyper-parameters, simultaneously. Also, extending the algorithm to deep neural networks and solving more complex problems are under investigation.

REFERENCES

[1] S. J. Mousavirad, G. Schaefer, S. M. J. Jalali, and I. Korovin, "A benchmark of recent population-based metaheuristic algorithms for multi-layer neural network training," in *Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 1402–1408.

[2] 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, 1997.

[3] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE International Conference on Evolutionary Computation*, 1998, pp. 69–73.

[4] R. S. Sexton and J. N. Gupta, "Comparative evaluation of genetic algorithm and backpropagation for training neural networks," *Information sciences*, vol. 129, no. 1-4, pp. 45–59, 2000.

[5] D. Kim, H. Kim, and D. Chung, "A modified genetic algorithm for fast training neural networks," in *International Symposium on Neural Networks*. Springer, 2005, pp. 660–665.

[6] S. Ding, C. Su, and J. Yu, "An optimizing bp neural network algorithm based on genetic algorithm," *Artificial intelligence review*, vol. 36, no. 2, pp. 153–162, 2011.

TABLE II
10CV CLASSIFICATION ACCURACY FOR ALL DATASETS FOR POPULATION-BASED TRAINING ALGORITHMS IN COMPARISON TO REG-IDE

Algorithms		Iris	Cancer	Liver	Pima	Seed	Vertebral	avg. rank
DE	mean	92	97.36	67.81	76.94	70.00	85.16	
	stddev	5.26	2.06	8.21	4.97	11.01	5.31	
	rank	11.5	12	13	13	13	9	11.92
QODE	mean	95.33	98.10	76.82	79.55	67.62	88.39	
	stddev	6.32	0.99	9.46	4.95	3.01	8.76	
	rank	7	4.5	1	3	14.5	1	5.17
RDE-OP	mean	96.67	98.82	75.63	80.21	67.62	86.77	
	stddev	6.48	1.67	6.45	5.73	4.92	4.42	
	rank	4.5	1	3	2	14.5	3	4.67
PSO	mean	96.00	97.95	73.36	77.60	78.10	86.45	
	stddev	5.62	1.72	6.28	3.24	11.92	8.02	
	rank	6	8	5	10	6	5	6.67
ABC	mean	84.67	97.95	70.75	78.26	72.38	82.90	
	stddev	9.45	1.03	6.47	4.45	8.03	5.70	
	rank	15	7	9	7	8.5	10	9.41
ICA	mean	96.67	97.22	72.39	79.42	84.76	86.77	
	stddev	4.71	1.46	11.99	5.77	10.24	4.67	
	rank	4.5	13	7	4	2	3	5.58
FA	mean	92.00	97.66	73.55	78.90	72.38	85.81	
	stddev	5.26	1.97	12.64	4.35	14.69	6.12	
	rank	11.5	10	4	6	8.5	6.5	7.75
GWO	mean	93.33	98.10	73.01	67.45	78.10	81.94	
	stddev	4.44	1.39	9.74	2.79	10.09	7.93	
	rank	9	3	6	15	5	13.5	8.58
ALO	mean	94.67	98.10	71.06	78.12	80.48	85.48	
	stddev	2.81	0.99	6.20	5.89	8.53	4.37	
	rank	8	4.5	8	8	4	8	6.75
DA	mean	92.67	97.51	70.42	77.85	70.48	81.94	
	stddev	5.84	1.83	7.01	5.40	7.03	5.31	
	rank	10	11	11	9	11.5	13.5	11.00
SCA	mean	90.67	97.08	65.50	74.47	71.43	82.26	
	stddev	7.83	1.82	5.96	4.20	8.98	10.67	
	rank	13	14	14	14	10	12	12.83
WOA	mean	87.33	97.07	62.87	76.95	70.48	79.03	
	stddev	8.58	1.96	6.40	3.65	8.92	10.99	
	rank	14	15	15	12	11.5	15	13.75
GOA	mean	98.00	98.09	70.73	79.03	84.29	82.58	
	stddev	3.22	1.84	6.45	3.72	12.10	6.49	
	rank	2.5	6	10	5	3	11	6.25
SSA	mean	98.00	97.80	69.85	77.34	77.62	85.81	
	stddev	3.22	2.42	7.78	6.50	9.27	7.16	
	rank	2.5	9	12	11	7	6.5	8.00
Reg-IDE	mean	99.33	98.39	76.27	80.60	92.86	86.77	
	stddev	2.11	2.24	4.03	4.15	4.05	5.37	
	rank	1	2	2	1	1	3	1.67

[7] N. M. Nawi, M. Rehman, M. A. Aziz, T. Herawan, J. H. Abawajy *et al.*, "An accelerated particle swarm optimization based levenberg marquardt back propagation algorithm," in *International Conference on Neural Information Processing*. Springer, 2014, pp. 245–253.

[8] T. Si and R. Dutta, "Partial opposition-based particle swarm optimizer in artificial neural network training for medical data classification," *International Journal of Information Technology & Decision Making*, vol. 18, no. 5, pp. 1717–1750, 2019.

[9] 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*, 2007, pp. 318–329.

[10] H. Duan and L. Huang, "Imperialist competitive algorithm optimized artificial neural networks for ucav global path planning," *Neurocomputing*, vol. 125, pp. 166–171, 2014.

[11] S. J. Mousavirad, A. A. Bidgoli, H. Ebrahimpour-Komleh, and G. Schaefer, "A memetic imperialist competitive algorithm with chaotic maps for multi-layer neural network training," *International Journal of Bio-Inspired Computation*, vol. 14, no. 4, pp. 227–236, 2019.

[12] S. Mandal, G. Saha, and R. K. Pal, "Neural network training using firefly algorithm," *Global Journal on Advancement in Engineering and Science (GJAES)*, vol. 1, no. 1, pp. 7–11, 2015.

[13] S. Mirjalili, "How effective is the grey wolf optimizer in training multi-layer perceptrons," *Applied Intelligence*, vol. 43, no. 1, pp. 150–161, 2015.

[14] S. Amirsadri, S. J. Mousavirad, and H. Ebrahimpour-Komleh, "A Levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training," *Neural Computing and Applications*, vol. 30, no. 12, pp. 3707–3720, 2018.

[15] W. Yamany, A. Tharwat, M. F. Hassanin, T. Gaber, A. E. Hassanien, and T.-H. Kim, "A new multi-layer perceptrons trainer based on ant lion optimization algorithm," in *Fourth international conference on information science and industrial applications*. IEEE, 2015, pp. 40–45.

[16] M. Khishe and A. Safari, "Classification of sonar targets using an MLP neural network trained by dragonfly algorithm," *Wireless Personal Communications*, vol. 108, no. 4, pp. 2241–2260, 2019.

[17] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[18] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm," *Soft Computing*, vol. 22, no. 1, pp. 1–15, 2018.

[19] A. A. Heidari, H. Faris, I. Aljarah, and S. Mirjalili, "An efficient hybrid multilayer perceptron neural network with grasshopper optimization," *Soft Computing*, vol. 23, no. 17, pp. 7941–7958, 2019.

[20] A. A. Abusnaina, S. Ahmad, R. Jarrar, and M. Mafarja, "Training neural networks using salp swarm algorithm for pattern classification," in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, 2018, pp. 1–6.

[21] S. J. Mousavirad, D. Zabihzadeh, D. Oliva, M. Perez-Cisneros, and G. Schaefer, "A grouping differential evolution algorithm boosted by attraction and repulsion strategies for masi entropy-based multi-level

image segmentation," *Entropy*, vol. 24, no. 1, p. 8, 2021.

[22] S. J. Mousavirad, A. Asilian Bidgoli, and S. Rahnamayan, "Tackling deceptive optimization problems using opposition-based DE with center-based Latin hypercube initialization," in *14th International Conference on Computer Science and Education*, 2019.

[23] A. Ara, N. A. Khan, O. A. Razzaq, T. Hameed, and M. A. Z. Raja, "Wavelets optimization method for evaluation of fractional partial differential equations: an application to financial modelling," *Advances in Difference Equations*, vol. 2018, no. 1, p. 8, 2018.

[24] S. V. Moravvej, S. J. Mousavirad, M. H. Moghadam, and M. Saadatmand, "An lstm-based plagiarism detection via attention mechanism and a population-based approach for pre-training parameters with imbalanced classes," in *International Conference on Neural Information Processing*. Springer, 2021, pp. 690–701.

[25] S. J. Mousavirad, D. Oliva, R. K. Chakrabortty, D. Zabihzadeh, and S. Hinojosa, "Population-based self-adaptive generalised masi entropy for image segmentation: A novel representation," *Knowledge-Based Systems*, vol. 245, p. 108610, 2022.

[26] S. J. Mousavirad, M. H. Moghadam, M. Saadatmand, R. Chakrabortty, G. Schaefer, and D. Oliva, "Rws-l-shade: An effective l-shade algorithm incorporation roulette wheel selection strategy for numerical optimisation," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2022, pp. 255–268.

[27] S. J. Mousavirad, G. Schaefer, I. Korovin, M. H. Moghadam, M. Saadatmand, and M. Pedram, "An enhanced differential evolution algorithm using a novel clustering-based mutation operator," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2021, pp. 176–181.

[28] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, vol. 17, no. 1, pp. 93–105, 2003.

[29] S. J. Mousavirad and S. Rahnamayan, "Evolving feedforward neural networks using a quasi-opposition-based differential evolution for data classification," in *IEEE Symposium Series on Computational Intelligence*, 2020.

[30] S. J. Mousavirad, G. Schaefer, I. Korovin, and D. Oliva, "RDE-OP: A region-based differential evolution algorithm incorporation opposition-based learning for optimising the learning process of multi-layer neural networks," in *24th International Conference on the Applications of Evolutionary Computation*, 2021.

[31] S. Mc Loone and G. Irwin, "Improving neural network training solutions using regularisation," *Neurocomputing*, vol. 37, no. 1-4, pp. 71–90, 2001.

[32] M. Carvalho and T. B. Ludermir, "Particle swarm optimization of feed-forward neural networks with weight decay," in *2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*. IEEE, 2006, pp. 5–5.

[33] A. Rakitianskaia and A. Engelbrecht, "Weight regularisation in particle swarm optimisation neural network training," in *2014 IEEE symposium on swarm intelligence*. IEEE, 2014, pp. 1–8.

[34] L. Munkhdalai, J. Y. Lee, and K. H. Ryu, "A hybrid credit scoring model using neural networks and logistic regression," in *Advances in Intelligent Information Hiding and Multimedia Signal Processing*. Springer, 2020, pp. 251–258.

[35] N. Abrishami, A. R. Sepaskhah, and M. H. Shahrokhnia, "Estimating wheat and maize daily evapotranspiration using artificial neural network," *Theoretical and Applied Climatology*, vol. 135, no. 3-4, pp. 945–958, 2019.

[36] H. D. Beale, H. B. Demuth, and M. Hagan, "Neural network design," *Pws, Boston*, 1996.

[37] S. J. Mousavirad, G. Schaefer, and I. Korovin, "An effective approach for neural network training based on comprehensive learning," in *International Conference on Pattern Recognition*, 2020.

[38] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[39] D. W. Marquardt, "An algorithm for least-squares estimation of non-linear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[40] G. Lera and M. Pinzolas, "Neighborhood based levenberg-marquardt algorithm for neural network training," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1200–1203, 2002.

[41] M. El-Bakry, E.-S. A. El-Dahshan, and E. Abd El-Hamied, "Charged particle pseudorapidity distributions for Pb–Pb and Au–Au collisions using neural network model," *Ukrainian Journal of Physics*, vol. 58, no. 8, pp. 709–709, 2013.

[42] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[43] K. Deb, "A population-based algorithm-generator for real-parameter optimization," *Soft Computing*, vol. 9, no. 4, pp. 236–253, 2005.

[44] Z. Cai, W. Gong, C. X. Ling, and H. Zhang, "A clustering-based differential evolution for global optimization," *Applied Soft Computing*, vol. 11, no. 1, pp. 1363–1379, 2011.

[45] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 1, 2005, pp. 695–701.

[46] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Quasi-oppositional differential evolution," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 2229–2236.

[47] S. J. Mousavirad, A. A. Bidgoli, H. Ebrahimpour-Komleh, G. Schaefer, and I. Korovin, "An effective hybrid approach for optimising the learning process of multi-layer neural networks," in *International Symposium on Neural Networks*, 2019, pp. 309–317.

[48] S. J. Mousavirad, S. M. J. Jalali, A. Sajad, K. Abbas, G. Schaefer, and S. Nahavandi, "Neural network training using a biogeography-based learning strategy," in *International Conference on Neural Information Processing*, 2020.

[49] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, "Differential evolution-based neural network training incorporating a centroid-based strategy and dynamic opposition-based learning," in *IEEE congress on evolutionary computation*. IEEE, 2016, pp. 2958–2965.

[50] V. Phansalkar and P. Sastry, "Analysis of the back-propagation algorithm with momentum," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 505–506, 1994.

[51] L. Scales, *Introduction to non-linear optimization*. Macmillan International Higher Education, 1985.

[52] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964.

[53] G. H. Golub and Q. Ye, "Inexact preconditioned conjugate gradient method with inner-outer iteration," *SIAM Journal on Scientific Computing*, vol. 21, no. 4, pp. 1305–1320, 1999.

[54] Y. Notay, "Flexible conjugate gradients," *SIAM Journal on Scientific Computing*, vol. 22, no. 4, pp. 1444–1460, 2000.

[55] M. J. D. Powell, "Restart procedures for the conjugate gradient method," *Mathematical Programming*, vol. 12, no. 1, pp. 241–254, 1977.

[56] R. L. Watrous, "Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization," 1988.

[57] R. Battiti, "First-and second-order methods for learning: between steepest descent and newton's method," *Neural Computation*, vol. 4, no. 2, pp. 141–166, 1992.

[58] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *IEEE International Conference on Neural Networks*. IEEE, 1993, pp. 586–591.

[59] M. F. Møller, *A scaled conjugate gradient algorithm for fast supervised learning*. Aarhus University, Computer Science Department, 1990.

[60] F. D. Foresee and M. T. Hagan, "Gauss-newton approximation to bayesian learning," in *International Conference on Neural Networks*, vol. 3, 1997, pp. 1930–1935.

[61] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks," in *IEEE Swarm Intelligence Symposium*, 2003, pp. 110–117.

[62] A. T. Sahlol, A. A. Ewees, A. M. Hemdan, and A. E. Hassanien, "Training feedforward neural networks using sine-cosine algorithm to improve the prediction of liver enzymes on fish farmed on nano-selenite," in *2016 12th International Computer Engineering Conference*. IEEE, 2016, pp. 35–40.

[63] D. Bairathi and D. Gopalani, "Salp swarm algorithm (SSA) for training feed-forward neural networks," in *Soft Computing for Problem Solving*. Springer, 2019, pp. 521–534.