Evolving Neural Networks using Ant Colony Optimization with Pheromone Trail Limits

Michalis Mavrovouniotis
Centre for Computational Intelligence (CCI)
School of Computer Science and Informatics
De Montfort University, The Gateway
Leicester LE1 9BH, United Kingdom
Email: mmavrovouniotis@dmu.ac.uk

Shengxiang Yang
Centre for Computational Intelligence (CCI)
School of Computer Science and Informatics
De Montfort University, The Gateway
Leicester LE1 9BH, United Kingdom
Email: syang@dmu.ac.uk

***Abstract*—The back-propagation (BP) technique is a widely used technique to train artificial neural networks (ANNs). However, BP often gets trapped in a local optimum. Hence, hybrid training was introduced, e.g., a global optimization algorithm with BP, to address this drawback. The key idea of hybrid training is to use global optimization algorithms to provide BP with good initial connection weights. In hybrid training, evolutionary algorithms are widely used, whereas ant colony optimization (ACO) algorithms are rarely used, as the global optimization algorithms. And so far, only the basic ACO algorithm has been used to evolve the connection weights of ANNs. In this paper, we hybridize one of the best performing variations of ACO with BP. The difference of the improved ACO variation from the basic ACO algorithm lies in that pheromone trail limits are imposed to avoid stagnation behaviour. The experimental results show that the proposed training method outperforms other peer training methods.**

I. Introduction

Artificial neural networks (ANNs) are commonly used for classification problems [1]. However, in order for an ANN to perform classification properly, a prior configuration is required, e.g., to decide the architecture of the ANN and the values of the connection weights.

In this paper, we focus on the selection of the optimal combination of the connection weights. Typically, gradient descent algorithms, such as back-propagation (BP) [2] are used to adjust the values of the connection weights. A drawback of BP is that it often gets trapped in a local optimum and is not capable of finding the global optimum [3], [4]. One way to overcome this drawback is to adopt evolutionary ANNs (EANNs), in which global optimization algorithms are used to find near to the global optimum combinations for the connection weights. Global optimization algorithms are less likely to get trapped in a local optimum solution than gradient descent algorithms.

Usually, evolutionary algorithms (EAs) [5] are used to evolve the connection weights in EANNs [6]. Ant colony optimization (ACO) is another global optimization algorithm, initially introduced for applications for discrete optimization problems [7], [8], which has attracted less attention in EANNs. ACO algorithms are inspired by the foraging behaviour of real ant colonies. Within the ACO framework, a population of self-organizing agents communicate via their pheromone trails to construct solutions for a given problem.

A hybrid framework was introduced in [9], where good initial weight values are obtained via basic ACO training and are passed to the BP training for further improvement. However, the pheromone trails generated may lead the algorithm into stagnation behaviour, where all ants select the same combination from the initial phase. In this paper, an improvement of the basic ACO training is proposed in which pheromone trail limits are imposed in order to eliminate high concentration of pheromone trails that may lead to stagnation behaviour. This way, the improved ACO training has even lower risk getting trapped in a local optimum than the basic ACO training. In fact, the experimental results on several datasets support this claim.

The rest of the paper is outlined as follows. Section II describes the concept of supervised learning and the evolution in ANNs. Section III describes the differences of the basic ACO training and the proposed ACO training, both of which are hybridized with the BP. In Section IV the experimental results and analysis are given. Finally, several concluding remarks and direction for future work are given in Section V.

II. Neural Networks

A. Architecture

An ANN consists of a number of units that are allocated in different layers, i.e., input, hidden, or output layers, which are interconnected. Typically, directed graphs are used to represent ANNs, where nodes represent the units and arcs represent the connections between them. Each arc holds a value which is the connection weight between the units. Each unit i performs a function which is defined as:

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right), \tag{1}$$

where f_i represents the activation function (usually sigmoid or Gaussian function) of unit i, y_i is the output of unit i, w_{ij} represents the connection weight between units i and j, x_j represents the j-th input of the unit, θ_i is the threshold (or bias) of unit i. Fig. 1 represents an ANN with four input units, two hidden units and three output units.

B. Supervised Learning

Supervised learning requires a training set that consists of several input parameters and a corresponding target parameter. Each input parameter is associated with a single unit from the input layer. The target parameter is used to calculate the network error, i.e., the difference between the actual and the target outputs. When the network error becomes sufficiently small it means that the network has generalized and the training process must stop to avoid possible over-fitting of the network. On the other hand, stopping the training process earlier may lead in under-fitting.

Generally, the aim of training is to minimize the error of the network by adjusting the connection weights. This process can be performed *on-line* in which the weights are adjusted after each training pattern (selected randomly) is processed, or *batch* in which the weights are adjusted when all training patterns are processed by the network.

C. Evolving Neural Networks

Once the ANN architecture is decided, then weight training needs to be performed before the network is used. Most training algorithms, e.g., the BP [2], are based on gradient descent and have been successfully applied to train ANNs [10], [11], [12]. However, BP has a drawback because it often gets trapped in a local optimum of the error function, since it is a local optimization algorithm [3], [4]. One way to avoid this drawback is to adopt EANNs, i.e., to evolve the connection weights. A comprehensive survey regarding EANNs is available in [6]. EAs are typically used to perform the evolution process. Different from BP, EAs are global optimization methods, and, thus, are less likely to get trapped in a local optimum.

There is a trade-off on whether training via evolution is more efficient than training via gradient descent [13]. Some researchers have demonstrated that a stand-alone evolutionary training is faster than BP training [14], whereas other researchers have demonstrated that there is no any significant difference between the two types of training and it really depends on the problem [15]. It was also discussed that simple training algorithms usually perform better than complex ones [16].

Kitano [17] proposed a hybrid genetic algorithm (GA) with BP (GA-BP) which shows that on small networks evolutionary training performs equally good as the optimized variances of BP. GA-BP first uses evolutionary training to find good initial weights. Then, BP uses these weights to increase accuracy. Yang-Peng *et al.* [9] proposed a similar hybrid but with an ACO algorithm instead of a GA. Their results show that the ACO with BP (ACO-BP) is more effective and efficient than the simple BP algorithm. Socha and Blum [15] have also showed that their ACO training algorithm performs better than a GA training algorithm, especially on large and complex problems.

Furthermore, it was suggested that instead of using a single large ANN to solve large and complex problems, it is better to use an ANN ensemble that adopts the divide-and-conquer strategy [18], [19]. ANN ensemble combines a set of ANNs that learn to decompose the problem into sub-problems and then solve them efficiently [20].

III. Hybrid ACO and BP

A. BP Training

The BP algorithm is a local optimization training algorithm and uses a gradient descent technique. More precisely, the actual outputs of the ANN for each training pattern are computed and then the network error is back-propagated. A gradient descent approach is used to adjust the connection weights and minimize the error as follows:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \tag{2}$$

where η is a constant that represents the learning rate, and E is the network error, in this case the squared error percentage (SEP), defined as follows:

$$E = 100 \frac{o_{max} - o_{min}}{n_o n_p} \sum_{p=1}^{n_p} \sum_{i=1}^{n_o} (t_i^p - o_i^p)^2, \tag{3}$$

where o_{max} and o_{min} are the maximum and minimum values of the output values from the output unit, respectively, n_p and n_o represents the number of patterns and the number of output units, respectively, t_i^p and o_i^p are the target and actual values of the output units, respectively.

B. ACO Training

ACO algorithms were initially proposed to solve discrete optimization problems [8], [21], [22]. Later on, an ACO continuous optimization framework ($ACO_{\mathbb{R}}$) was proposed [15], [23], and was applied to train feed-forward ANNs since the training process can be considered as a continuous optimization process. Also, other attempts were proposed to train ANNs that do not follow the original ACO framework [24], [25]. In contrast, Yang-Peng *et al.* [9] have proposed an ACO algorithm to train ANNs and their framework is very close to the original ACO framework developed for the travelling salesman problem (TSP) [8].

Considering that an ANN consists of l connection weights calculated as follows:

$$l = n_h(n_i + 1) + n_o(n_h + 1), \tag{4}$$

where n_h, n_i and n_o are the number of hidden, input, and output units, respectively. The additional units represents the bias inputs for the hidden and output layer. The optimal combination of connection weights values needs to be found by training. Therefore, ACO becomes a sufficient choice to select good combinations considering its performance on the TSP. The key idea is to split each connection weight w_z into v discrete points a_i^z, $i = 1, \ldots, v$, $z = 1, \ldots, l$, where each one represents a value of the corresponding connection weight w_z. The range of each discrete point is within the interval $[w_{min}, w_{max}]$, which is defined by the user.

In ACO-BP [9], each ant selects one and only one discrete point for each connection weight and stores the tag of the specific discrete point. When all ants choose a value for all connection weights, they deposit pheromone. Each connection weight w_z is assigned with a pheromone table τ_i^z, $i = 1, \ldots, v$, $z = 1, \ldots, l$.

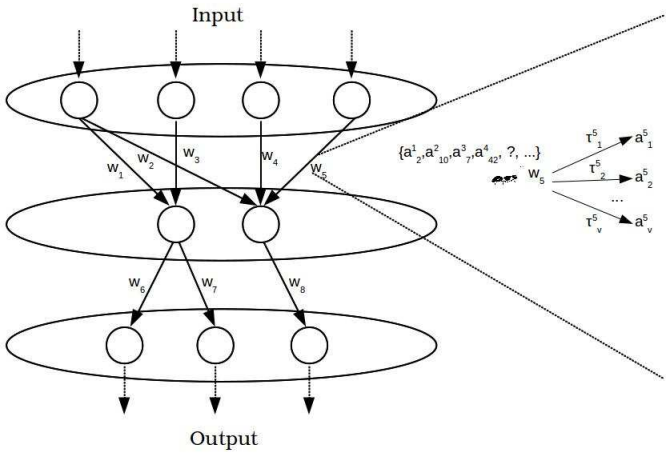

Fig. 1. Illustration of an ant selecting a value for connection weight w_5 in an ANN. After connection weight w_5 the ant moves to connection weight w_6, and so on.

More precisely, when an ant reaches a connection weight w_z, the probability of selecting a discrete point a_i^z is defined as follows:

$$p_i^z = \frac{\tau_i^z}{\sum_{1=j}^{v} \tau_j^z}, \tag{5}$$

where v represents the number of discrete points a_i^z, and τ_i^z represents the existing pheromone trail of connection weight w_i for discrete point a_i^z. For example, in Fig. 1, an ant has already selected values a_2^1, a_{10}^2, a_7^3 and a_{42}^4 for connection weights w_1, w_2, w_3 and w_4, respectively, and selects a value for w_5 according to the existing pheromone trails. The process is repeated until all ants select a value for all connection weights.

Then, each ant retraces the path according to the discrete points tags stored in the construction phase and deposits pheromone as follows:

$$\tau_i^z \leftarrow \tau_i^z + \Delta\tau_i^z, \forall\ a_i^z \in T^k, \tag{6}$$

where T^k is the combination of discrete points selected by the k-th ant, and $\Delta\tau_i^z$ is the amount of pheromone to be deposited in pheromone table z, which is defined as follows:

$$\Delta\tau_i^z = Q/E, \tag{7}$$

where Q is a constant (a good value is $Q = 1$) and E is a measurement to calculate the error between the actual output of the network and the target output defined in Eq. (3). The less network error the more pheromone is deposited. Furthermore, pheromone evaporation is applied in all pheromone tables, in which all trails are deducted as follows:

$$\tau_i^z \leftarrow (1-\rho)\tau_i^z, \forall\ a_i^z, \tag{8}$$

where $\rho \in (0, 1)$ is a constant that defines the pheromone evaporation rate. The pheromone evaporation helps to eliminate bad decisions made in the past.

C. Improved ACO Training

Heuristic information is not considered in the probabilistic rule in Eq. (5) as in the traditional ACO framework. For example, in the TSP, the distance between cities is considered as heuristic information, together with existing pheromone trails, when solutions are constructed. Dorigo and Stützle [7, pp. 97–98] stated that "when only pheromone amplification is at work without heuristic bias ACO leads in very poor results".

The use of heuristic information is very important for ACO and has a great impact to the overall performance of the algorithm. For example, at the initial stage of training an ANN, the pheromone trails in all tables assigned to each connection weight are initialized with an equal amount. Hence, the construction of the combinations at early stages is random since only pheromone trails are considered. As a result, high intensity of pheromone trails will be generated at early stages on a single combination that may have poor quality. Even if a higher pheromone evaporation rate is applied, it will not have an effect since all pheromone trails will be evaporated by the same factor and any previous knowledge may be destroyed.

According to the preliminary experiments on the TSP in [7], ACO without heuristic information but with the use of local search showed comparable results with an ACO with heuristic information. Therefore, the experiments give a good indication that ACO can provide good initial combination weights to the BP in order to perform a local search improvement. It was also suggested that in problems where heuristic information is not available, ACO needs to be applied with a local search scheme. In fact, training an ANN is one of these problems because it is not possible to consider the value of the connection weights as heuristic information. This is because the training of ANN is not a traditional minimization or maximization problem, e.g., the TSP where the less the distance the better the choice. In ANN, both negative and positive values might be the best choice for a connection weight, and, thus, it is impossible to distinguish if a higher or lower value is better in the combination selection process.

The existing ACO-BP [9] described in Section III-B satisfies the statements above, since heuristic information is not used, but BP is used as a local search operator. However, in ACO-BP the simplest case of ACO, i.e., ant system (AS) [26], is used, whereas in the experiments [7, pp. 97–98] other variations of the AS, e.g., $\mathcal{MAX}-\mathcal{MIN}$ AS ($\mathcal{MM}$AS) [27], are used. The main difference between AS and $\mathcal{MM}$AS

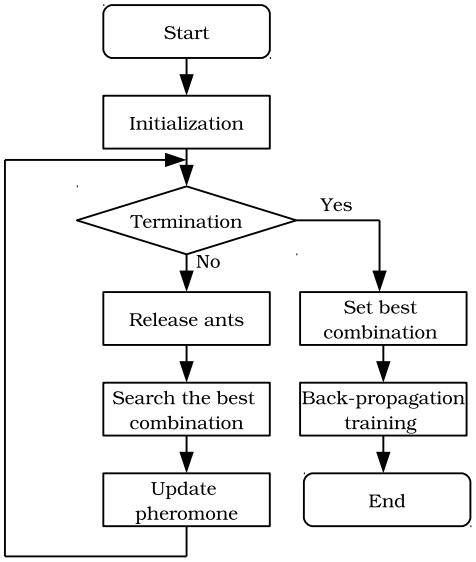

Fig. 2. Framework of hybrid training of ANNs via ACO and BP.

lies in that pheromone trail limits are imposed in the $\mathcal{MM}$AS. It is very important to keep the maximum and minimum pheromone trail values at a closer range in order to eliminate the high intensity of pheromone trails that may bias ants to search at non-promising areas. Therefore, our proposed neural network ACO-BP (NNACO-BP) inherits the $\mathcal{MM}$AS framework and the pheromone is updated as follows:

$$\tau_i^z \leftarrow \tau_i^z + \Delta\tau_i^z, \forall\ a_i^z \in T^{best}, \tag{9}$$

where T^{best} is the combination of discrete points selected by the *best* ant and $\Delta\tau_i^z$ is defined as in Eq. (7). In addition, pheromone evaporation is used as in Eq. (8). Note that in the proposed NNACO-BP only the best ant is allowed to deposit pheromone whereas in ACO-BP all the ants deposit pheromone. Within NNACO-BP, the range of possible range of pheromone trail values is limited to the interval $[\tau_{min}, \tau_{max}]$, where $\tau_{max} = 1/(\rho E^{best})$ is the maximum pheromone trail limit, ρ is defined in Eq. (8), E^{best} is the network error of the best combination, i.e., T^{best}, calculated in Eq. (7), $\tau_{min} = \tau_{max}/2l$ is the minimum pheromone trail limit and l is defined as in Eq. (4).

D. Hybrid Framework

Hybrid training algorithms, such as GA-BP [28] and ACO-BP [9], usually perform better than stand-alone evolutionary or traditional training algorithms. This is due to the fact that global optimization algorithms are less sensitive on the initial condition of training, whereas local optimization algorithms find the local optimum in the neighbourhood of the initial weights given. In many cases the initial weights selected for BP may lead to a very poor local optimum.

The general idea behind hybrid training is to use global optimization algorithms to find near-optimal initial connection weights and then use a local search operator to improve the accuracy of the specific near-optimal solution found; see Fig. 2.

TABLE I. OVERVIEW OF THE DATASETS FOR THE PROBLEMS USED

Dataset	Testing Set	Training Set
Cancer	174	525
Diabetes	192	576
Heart	230	690

TABLE II. OVERVIEW OF THE ANN'S ARCHITECTURE FOR DIFFERENT PROBLEMS.

Dataset	Input Layer	Hidden Layer	Output Layer	Connection Weights
Cancer	9	6	2	74
Diabetes	8	6	2	68
Heart	35	6	2	230

IV. EXPERIMENTAL STUDY

A. Experimental Setup

To evaluate the proposed NNACO-BP with other algorithms the training of feed-forward ANN for classification problems is used. The algorithms used in the experimental study are the following:

1) BP [2]: in this algorithm, the simple BP training, as described in Section III-A, is used on its own.
2) ACO-BP [9]: this hybrid training algorithm uses an ACO based on the basic AS framework as a global optimization algorithm and BP as a local optimizer, as described in Section III-B.
3) NNACO-BP: this is our proposed hybrid training algorithm, which uses an ACO based on the $\mathcal{MM}$AS framework as a global optimization algorithm and BP as a local optimizer, as described in Section III-C.

All the algorithms perform 1000 evaluations in order to have a fair comparison. A *four* cross-validation is used where a set of patterns is divided into four equal subsets. Then, four experiments are performed where one subset is used as the test dataset (patterns that the network has never seen before) and the remaining subsets are used as the training dataset. The classification error percentage (CEP) from all the four experiments is computed and the effect of the set division between the training and test dataset is averaged for 50 independent runs.

1) Problem Datasets: To evaluate the performance and effectiveness of NNACO-BP, three different benchmark datasets, i.e., Cancer, Diabetes and Heart, are used, which were taken from PROBEN1 [29]. All these datasets arise from the medical field. They consist of a pattern of real numbers which form a measurement and are classified as pathological or normal.

- Cancer: Taken from a database of diagnostics of breast cancer. It consists of 699 examples, where each one consists of 9 inputs in the interval of $[0, 1]$ and boolean target outputs [30], [31].
- Diabetes: Based on medical examinations which decides whether a Prima Indian diabetes is positive or not. It consists of 768 examples, where each one consists of 8 inputs in the interval of $[0, 1]$ and boolean target outputs [32].

TABLE III. PARAMETER SETTINGS FOR THE ALGORITHMS INVESTIGATED

	Cancer					Diabetes					Heart				
Algorithm	m	e	t	η	ρ	m	e	t	η	ρ	m	e	t	η	ρ
ACO-BP	50	500	10	0.002	0.9	50	500	10	0.01	0.8	50	500	10	0.001	0.8
BP	-	1000	-	0.002	-	-	1000	-	0.01	-	-	1000	-	0.001	-
NNACO-BP	50	500	10	0.002	0.009	50	500	10	0.01	0.008	50	500	10	0.001	0.008

TABLE IV. EXPERIMENTAL RESULTS REGARDING CEP FOR EACH ALGORITHM OVER ALL RUNS FOR ALL CROSS-VALIDATION EXPERIMENTS.

	Cancer		Diabetes		Heart	
Algorithm	Testing	Training	Testing	Training	Testing	Training
ACO-BP	3.90 $\pm$ 1.43	3.02 $\pm$ 0.76	26.17 $\pm$ 2.76	24.93 $\pm$ 1.28	20.56 $\pm$ 2.10	18.38 $\pm$ 0.76
BP	4.01 $\pm$ 1.30	3.27 $\pm$ 0.38	26.55 $\pm$ 2.82	25.70 $\pm$ 1.27	22.63 $\pm$ 1.67	19.68 $\pm$ 0.64
NNACO-BP	3.66 $\pm$ 1.65	2.88 $\pm$ 0.55	24.47 $\pm$ 2.27	23.95 $\pm$ 1.29	17.77 $\pm$ 1.46	15.42 $\pm$ 0.79

- Heart: Based on medical examinations which predicts heart disease. It consists of 920 examples, where each one consists of 35 inputs in the interval of $[0, 1]$ and boolean target outputs [33].

Since cross-validation is used, it means that the 75% of the dataset example is used for training dataset and the rest 25% is used for the test dataset. Table I indicates the division of the training and test dataset. For each experiment in cross-validation, the same sizes of the datasets are used.

2) Parameter Settings: The structure of the ANNs and the parameters of the algorithms are mainly inspired by the literature [15]. In Table II, the network architecture is given, which consists of one input layer, one hidden layer, and one output layer. The reason that the specific parameters are used, is to later on compare the results of NNACO-BP with the results of another ACO training [15] algorithm and a GA training [28] algorithm that use the same network architectures on the same problem datasets.

The parameters for ACO-BP, the stand-alone BP, and the proposed NNACO-BP are given in Table III. The parameters for BP and ACO-BP are the same with the ones used in [15], [9]. Hybrid training algorithms perform less e (epochs) in order to have equal number of evaluations with the basic BP. More precisely, each ant counts for a single evaluation, and, thus, m ants for t algorithmic iterations equal to $m \times t$ evaluations. And further e evaluations with BP equal to the same number of evaluations with the basic BP algorithm. Although algorithms perform the same number of evaluations, it does not mean that they use the same computational time. However, the same stopping criteria is used, i.e., when the maximum number of evaluations is reached. For ACO-BP and NNACO-BP $w_{min} = -1$, $w_{max} = 1$ and $v = 50$.

In the following sections, we first present the comparison between the investigated algorithms in detail with statistical results, and then compare the proposed algorithm with other hybrid training algorithms based on the experimental setup described above.

B. Analysis of the Results

The mean CEP with the standard deviation results, for both *testing* and *training* datasets, for each algorithm of all cross-validation experiments are presented in Table IV. The corresponding Wilcoxon rank sum statistical results (with Bonferroni correction) of the algorithms on the *testing* dataset are presented in Table V. Furthermore, the experimental results are presented in the form of box-plots in Figs. 3, 4, and 5, for the datasets Cancer, Diabetes, and Heart, respectively. Each figure contains two box-plots. The left box-plots illustrate the distribution of the CEP values, whereas the right box-plots the ranks of each algorithm. The corresponding ranks are calculated as follows. The CEP values are exchanged for their corresponding ranked value, i.e., having 3 algorithms and running 50 independent runs for each of the 4 cross-validation experiments; then the possible ranked results vary from 1 to 600. The box-plots based on the ranks of the algorithms usually give a clearer identification of the differences between the investigated algorithms. For example, in Fig. 3, from the CEP box-plots, it is difficult to identify which algorithm performs better, whereas from the corresponding ranked box-plots it is more clear.

Cancer (see Fig. 3) is the smallest dataset among the problems tested. All algorithms have good performance, but there is no any statistical significance between them (see Tables IV and V). It appears that the specific dataset is not noisy and it is easy for the training algorithms to classify it. However, none of the algorithms was able to classify all the patterns from the testing dataset correctly. This may be due to the limited number of patterns in the training dataset.

Diabetes (see Fig. 4) is again a small dataset but is slightly larger than Cancer. NNACO-BP significantly outperforms BP and ACO-BP (see Tables IV and V). In general, all algorithms have a relatively poor performance in terms of CEP. Different from Cancer, this dataset is harder for training algorithms to classify. This may be due to some noisy/missing patterns or the limited size of the training dataset, which similarly with Cancer, might not represent all the necessary patterns to classify more test patterns correctly.

Heart (see Fig. 5) is a large dataset and it is the largest problem among the other datasets. All algorithms are statistically different. The proposed NNACO-BP significantly outperforms ACO-BP and BP, whereas ACO-BP outperforms BP (see Tables IV and V). This may be due to the size of the dataset which is extremely large and it is easier for gradient descent algorithms, e.g., BP, to get stuck in local optima. On the other hand, ACO-BP and NNACO-BP can overcome this

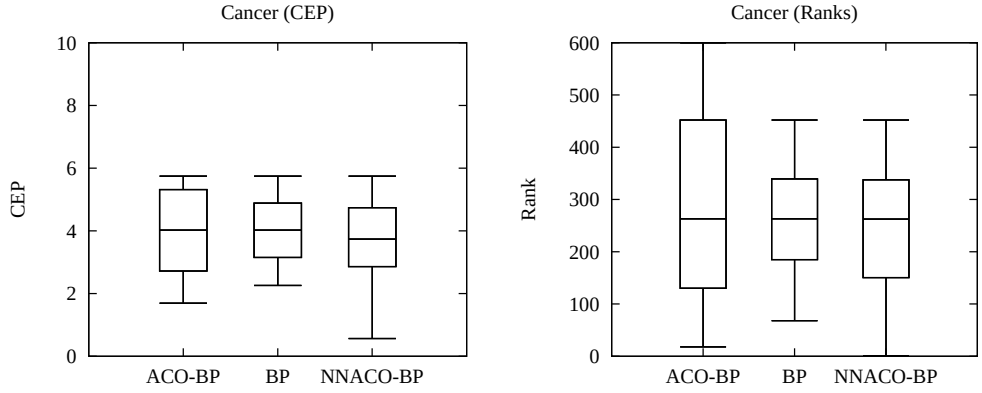

Fig. 3. Box-plots for Cancer drawn between the first and the third quartile of the distribution.

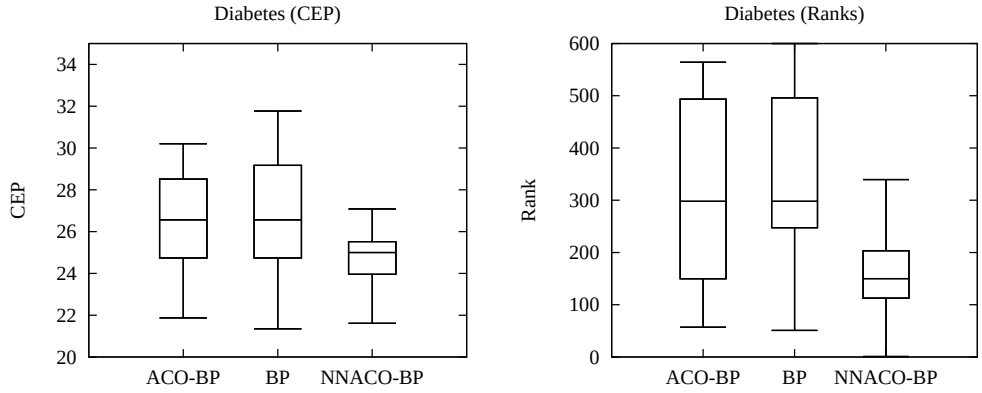

Fig. 4. Box-plots for Diabetes drawn between the first and the third quartile of the distribution.

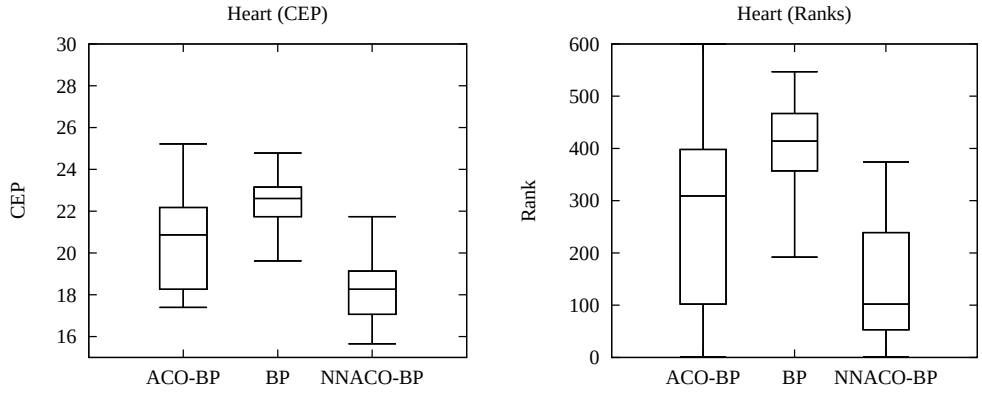

Fig. 5. Box-plots for Heart drawn between the first and the third quartile of the distribution.

problem and have better results on large problems.

Generally, the basic BP algorithm is usually outperformed by the other hybrid algorithms as the problem size increases. This is probably because the selected initial weights of the BP may lead to a poor local optimum. ACO-BP and NNACO-BP perform better since they are able to provide better initial weights for the BP. However, NNACO-BP significantly outperforms ACO-BP in Heart even if both algorithms use hybrid training. This shows the importance of imposing limits to the pheromone trails. The initial stages of NNACO-BP is more explorative than ACO-BP since the convergence is delayed. Hence, it has less risk, than ACO-BP, to get trapped in any local optima.

Finally, if the CEP results of the *testing* over the *training* datasets are compared, it shows that all the algorithms suffer from a slight over-fitting in all problems. Probably a higher

TABLE V. STATISTICAL TESTS OF THE CEP EXPERIMENTAL RESULTS ON THE TESTING DATASET.

Algorithm	ACO-BP	BP	NNACO-BP
Cancer			
ACO-BP		-	-
BP	-		-
NNACO-BP	-	-	
Diabetes			
ACO-BP		-	+
BP	+		+
NNACO-BP	+	+	
Heart			
ACO-BP		+	+
BP	+		+
NNACO-BP	+	+	

number of cross-validation experiments may be necessary. In fact, the performance regarding CEP may be further improved for all algorithms if more cross-validation experiments are performed.

C. Comparison with other hybrid algorithms

It is interesting to compare the proposed NNACO-BP with GA-BP [28], which is a hybrid training algorithm using a GA and BP. Additionally, it is more interesting to compare NNACO-BP with another hybrid ACO approach, that is $ACO_{\mathbb{R}}$-BP [34], which does not use the original framework of an ACO as the proposed NNACO-BP and the ACO-BP.

Both GA-BP and $ACO_{\mathbb{R}}$-BP were applied with the same stopping criteria based on the same training datasets used in our experimental setup. Table VI summarizes the CEP results obtained by GA-BP and ACO. Note that the results were not obtained by a *four* cross-validation and we only performed the first experiment of the *four* cross-validation experiments. Therefore, the results of NNACO-BP algorithm refer only to the first cross-validation experiment. All three algorithms performed 50 independent runs for each problem.

In the Diabetes and Heart problems, NNACO-BP outperforms both GA-BP and $ACO_{\mathbb{R}}$-BP. However, GA-BP is comparable with NNACO-BP in Cancer, whereas NNACO-BP performs better than $ACO_{\mathbb{R}}$-BP. The difference on Cancer between all three algorithms is small. It might need further investigation, or according to the *no free lunch* theory [35] it may be impossible to have an algorithm that performs best for all problems.

V. CONCLUSIONS

The connection weights in ANN are usually adjusted using the BP algorithm, which is a gradient descent method. Often the initial connection weights selected for BP may lead to the convergence to a local optimum. In this paper, the BP algorithm is hybridized with a global optimization algorithm, i.e., ACO. The basic idea of the proposed framework is to obtain good initial connection weight values via ACO and pass the results to the BP. The experimental results show that the hybrid training usually performs better than the basic BP training, especially in large problems. As the problem becomes larger, it may increase the probability of the basic BP algorithm to get stuck in local optima. Furthermore, the proposed hybrid ACO training with pheromone trail limits performs better than the basic hybrid ACO training and other hybrid training algorithms.

TABLE VI. COMPARISONS OF THE CEP VALUES OF THE PROPOSED ALGORITHM WITH OTHER HYBRID TRAINING ALGORITHMS

Algorithm	Cancer	Diabetes	Heart
NNACO-BP	1.44 ± 0.52	$\mathbf{20.83 \pm 0.71}$	$\mathbf{15.65 \pm 0.31}$
GA-BP	$\mathbf{1.43 \pm 4.87}$	36.36 ± 0.0	54.30 ± 20.03
$ACO_{\mathbb{R}}$-BP	2.14 ± 1.09	23.80 ± 1.73	18.29 ± 1.00

For future work, it will be interesting to apply other ACO improvements with the BP, or even use other local search approaches such as the Levenberg-Marquardt algorithm [36]. Moreover, it will be interesting to apply the algorithm on more complex benchmark problems, or even investigate NNACO-BP's adaptability to a dynamic environment [6].

ACKNOWLEDGMENT

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1.

REFERENCES

[1] G. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 30, no. 4, pp. 451–462, 2000.

[2] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by backpropagation errors," *Nature*, vol. 536, pp. 323–533, 1986.

[3] R. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proceedings of the 8th Annual Conference Cognitive Science Socitey*, 1986, pp. 823–831.

[4] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," *Parallel Computing*, vol. 14, no. 3, pp. 347–361, 1990.

[5] J. Holland, *Adaption in Natural and Artifical Systems*. Ann Arbor: University of Michigan Press, 1975.

[6] X. Yao, "Evolving artificial neural networks," in *Proceedings of the IEEE*, vol. 89, no. 9. IEEE Press, 1999, pp. 1423–1447.

[7] M. Dorigo and T. Stützle, Eds., *Ant colony optimization*. London, England: MIT Press, 2004.

[8] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[9] L. Yan-Peng, W. Ming-Guang, and Q. Ji-Xin, "Evolving neural networks using the hybrid of ant colony optimization and bp algorithm," in *Advances in Neural Networks - 3rd International Symposium on Neural Networks*, ser. LNCS, vol. 3971. Springer-Verlag, 2006, pp. 714–722.

[10] G. Hinton, "Connectionist learning approaches," *Artificial Intelligence*, vol. 40, no. 1-3, pp. 185–234, 1989.

[11] K. Lang, A. Waibel, and G. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Networks*, vol. 3, no. 1, pp. 33–43, 1990.

[12] S. Fels and G. Hinton, "Glove-talk: A neural network interface between a data-glove and a speech synthesizer," *IEEE Transactions in Neural Networks*, vol. 4, pp. 2–8, 1993.

[13] J. Bullinaria, "Evolving neural networks: Is it really worth the effort?" in *Proceedings of the European Symposium on Artificial Neural Networks*, 2005, pp. 267–272.

[14] D. Montana and L. Davis, "Training feedforward neural network using genetic algorithms," in *Proceedings of the 11th International Joint Conference Artificial Intelligence*. Morgan Kaufmann, 1989, pp. 762–767.

[15] K. Socha and C. Blum, "An ant colony optimization algorithm for continuous optimization: Application to feed-forward neural network training," in *Proceedings of Neural Computation and Applications*, vol. 16. Springer-Verlag, 2007, pp. 235–247.

[16] E. Cantu-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 35, no. 5, pp. 915–927, 2005.

[17] H. Kitano, "Empirical studies on the speed of convergence of neural network using genetic algorithms," in *Proceedings of the 8th National Conference on Artificial Intelligence*, vol. 2. Springer-Verlag, 1990, pp. 789–795.

[18] X. Yao and Y. Liu, "Ensemble structure of evolutionary artificial neural networks," in *Proceedings of 1996 International Conference on Evolutionary Computation*, 1996, pp. 659–664.

[19] ——, "Making use of population information in evolutionary artificial neural networks," *IEEE Transactions on Systems, Man, Cybernetics-Part B*, vol. 28, no. 3, pp. 417–425, 1998.

[20] X. Yao and M. M. Islam, "Evolving artificial neural network ensembles," *IEEE Computational Intelligence Magazine*, vol. 3, no. 1, pp. 31–42, 2008.

[21] B. Bullnheimer, R. Hartl, and C. Strauss., "A new rank-based version of the ant system: A computational study," *Central European Journal for Operations Research and Economics*, vol. 7, no. 1, pp. 25–38, 1999.

[22] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[23] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[24] B. Bilchev and I. Parmee, "The ant colony metaphor for searching continuous design spaces," in *Proceedings of the AISB Workshop on Evolutionary Computation*, ser. LNCS, vol. 993. Springer-Verlag, 1995, pp. 25–39.

[25] J. Dreo and P. Siarry, "A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions," in *Proceedings of the 3rd International Workshop on Ant Algorithms*, ser. LNCS, M. Dorigo, G. D. Caro, and M. Samples, Eds., vol. 2463. Springer-Verlag, 2002, pp. 216–221.

[26] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on System Man and Cybernetics-Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[27] T. Stützle and H. Hoos, "The max-min ant system and local search for the traveling salesman problem," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*. IEEE Press, 1997, pp. 309–314.

[28] E. Alba and J. Chicano, "Training neural networks with ga hybrid algorithms," in *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, ser. LNCS, K. Deb, Ed., vol. 3102. Springer-Verlag, 2004, pp. 852–863.

[29] L. Prechelt, "Proben1 - a set of neural network benchmark problems and benchmarking rules," University Karlsruhe, Germany, Tech. Rep. 21, 1994.

[30] O. Mangasarian and W. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, vol. 23, no. 5, pp. 1–18, 1990.

[31] W. Wolberg and O. Mangasarian, "Multisurface method for pattern separation for medical diagnosis applied to breast cytology," in *Proceedings of the National Academy of Sciences*, vol. 87, 1990, pp. 9193–9196.

[32] J. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Using the adap learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of the Twelfth Annual Symposium on Computer Applications in Medical Care*, vol. 9. IEEE Computer Society Press, 1988, pp. 261–265.

[33] R. Detrano, A. J. W., Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, and V. Froelicher, "International application of a new probability algorithm for the diagnosis of coronary artery disease," *American Journal of Cardiology*, vol. 64, no. 5, pp. 304–310, 1989.

[34] K. Socha, "ACO for continuous and mixed-variable optimization," in *Proceedings of the 4th International Workshop on Ant Algorithms and Swarm Intelligence*, ser. LNCS, M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stuzle, Eds., vol. 3172. Springer-Verlag, 2004, pp. 25–36.

[35] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.

[36] K. Levenberg, "A method for solution of certain problems in least squares," *The Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.