International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 8, No.3 (May-2019)
http://dx.doi.org/10.12785/ijcds/080305

A Biology Inspired Algorithm to Mitigate the Local Minima Problem and Improve the Classification in Neural Networks

Nabil M. Hewahi[1] and Zahraa J. Jaber [2]

[1,2]*Department of Computer Science, University of Bahrain, Bahrain*

Received 14 Jan 2019, Revised 20 Mar. 2019, Accepted 9 Apr. 2019, Published 1 May 2019

Abstract: ANN is a very well-known approach used for classification based on supervised machine learning. This approach faces some issues, notably the local minima problem, which leads to diminished accuracy in the results. To solve the problem of local minima, a new algorithm called RPSOGAC has been proposed. The proposed algorithm combines the strengths of both optimization algorithms PSO and GA to improve the classification accuracy of ANNs. RPSOGAC starts by finding the best weights that lead to the best ANN classification result using the backpropagation algorithm and adds these weights to the initial population. The other individuals of the population are randomly generated. Based on randomness, the algorithm reciprocally and continually switches between applying the GA and PSO algorithms until it reaches to the best solution. Two major differences between RPSOGAC and other previous algorithms, firstly is the random selection of GA and PSO, which gives equal opportunities for them to improve the classification. Secondly, during PSO, a competition between two population sets are performed to come up with a new population having the best individuals, this gives a chance for expected to improve individuals to enhance in the future if possible. Various experiments on six different datasets related to four domains have been conducted to show the classification accuracy of RPSOGAC. Also, a comparative study has been performed to compare the accuracy performance of classification between RPSOGAC and other algorithms. The obtained results show that RPSOGAC outperforms other approaches in four datasets and in the other two the results are very close.

Keywords: Classification, Neural network, Genetic algorithm, Particle swarm optimization

1. INTRODUCTION

Machine Learning (ML) is a major field within Artificial Intelligence (AI) and computer science. Classification of a set of objects/patterns into a given categories depending on their properties is one of the main issues in ML. Classification is considered to be a supervised learning. Using ANN for classification is considered to be supervised learning, whereas using ANN for clustering is considered to be unsupervised learning [10].

In supervised learning, ANNs are trained using training datasets until they become mature enough, after which they are used for classifying unknown data. One well-known supervised training algorithm foe ANNs is called the backpropagation (BP) algorithm [8]. One of its major drawbacks is that it does not guarantee to find the global minima (the minimum error) and falls into what so-called local minima, which causes training classification accuracy to stop improving. In this paper we propose a new algorithm based on a hybrid approach of GA and PSO to get rid of the local minima problem and to improve the classification accuracy.

A. Artificial Neural networks

ANN is a mathematical model inspired from biological operations of the human brain. ANNs are used in machine learning for classification and clustering. Classification comes under supervised learning, whereas clustering comes under unsupervised learning [28]. Multi-Layer Perceptron (MLP) is a neural network that has at least three layers; the input layer, hidden layer, and output layer as shown in Fig. 1. In MLP there might be more than a single hidden layer. Each of the three layers of a neural network contains neurons. There are links connecting the neurons of each layer with the next layer. Each link is associated with a specific numerical value called a weight. These weights are adjusted numerically using a training algorithm until suitable weights capable of giving adequate classification accuracy are obtained.

E-mail address: nhewahi@uob.edu.bh, zahraajawadjaber@gmail.com

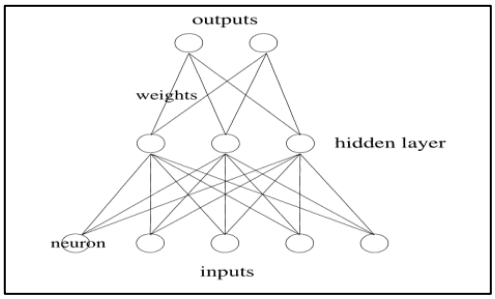

Figure 1. Architecture of a neural network

Weight adjustments are done using a well-known algorithm called BP algorithm. The process of training ANNs is based on providing them with training datasets. Initially, the neural network weights are generated at random. With each training instant, the BP algorithm updates the weights. This process is repeated for many epochs (one epoch is the process of parsing all the instants of the training dataset one time). The BP algorithm continues training until the required error rate is achieved or the number of iterations is reached [31].

B. The Problem of Local Minima

Since the PB algorithm uses the gradient descent method, it faces two issues. First, the convergence speed is very slow even when a termination error is given. Many training algorithms have been proposed to solve this problem, such as the quasi-Newton algorithm and the conjugate gradient algorithm [12] [13]. Second, it faces the local minima problem during the learning process, as shown in Fig. 2. It is especially common in nonlinear separable or complex function approximation problems [4]. This problem leads to BP algorithm failure in finding or reaching a globally optimal error. There are many reasons behind these issues. Chief among them are the initially selected weight values, the parameters of the activation function, learning rates, momentum or even the nature of the data and its size [22]. In other words, the local minima problem is stopping the process of convergence to reach the global minima which is the lowest required error. Fig. 2 depicts the local minima problem. Many attempts have been used to solve the problem of local minima relaying on biology-inspired algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) [9] and [2].

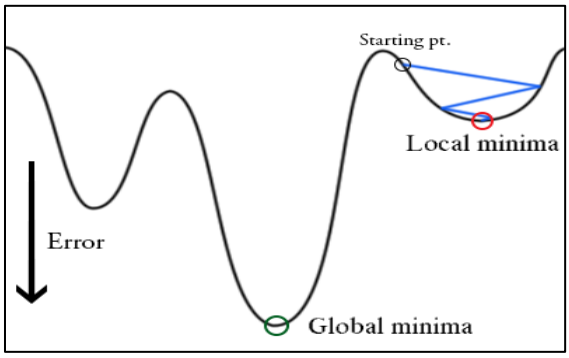

Figure 2. Gradient descent stuck at local minima [24]

C. Genetic Algorithms

GA is a method of heuristic optimization [18], it became popular after John Halland published his work in 1970 [19]. GA is an efficient algorithm for solving complex problems, especially when the number of parameters is large [25]. Moreover, it is commonly used to produce efficient solutions to search problems and for optimization purposes. It applies bio-inspired rules and is based on the concepts of natural selection and genetics. It works in an iterative fashion, making modifications on the individuals (chromosomes) within the population, thus perpetuating the evolutionary process. GA is applied in many fields, such as neural network classification [25], economics (e.g., modeling and game theory) [23], vehicle routing [27], DNA analysis [29]. The population is represented as individuals (chromosomes). Each chromosome is a group of genes. GA involves three main genetic operators; selection, crossover, and mutation. In selection operator, contributing individuals (parents) are selected to produce the population of the next generation. In crossover operator, two parents merge to produce two new children. In mutation operator, changes are applied on parents to produce new children. As shown in Fig. 3, GA consists of three main operations; encoding, fitness function computation, and applying genetic operators [31]. The operations of the GA are as follows: chromosomes are converted to binary code as a string of genes because the GA cannot work directly with the problem data. Then, the initial population is randomly generated to contain the individuals (chromosomes/solutions). The size of the population, the total number of generations, the probability of both crossover $P_{crossover}$ and mutation $P_{mutation}$ are defined and have to be given to GA. The $P_{crossover}$ and $P_{mutation}$ are the probability values for the possibility of applying crossover and mutation consecutively. The total number of iterations is the number of times the genetic operators are applied on the population. Next, the fitness function is defined and used to evaluate the chromosomes in order to select the best individuals and pass them down to the

next generation, thus measuring the efficacy of each instance is solution.

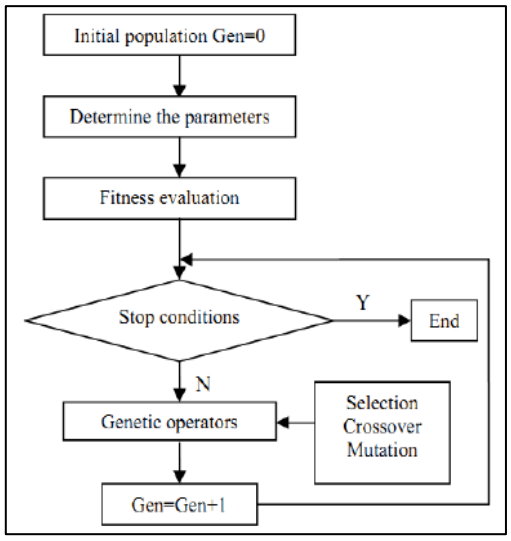

Figure 3. GA structure

D. Particale Swarm Optimization Algorithm

The PSO algorithm is a heuristic global and computational method that optimizes the problem by improving candidate solutions through an iterative technique. It was produced by Kennedy and Eberhart in 1995 [4]. This algorithm is based on the concept of swarm intelligence, i.e., the simulation of the social behavior of animals in groups. It was developed based on research on swarm behavior in foraging, such as in fish and bird species [4]. Unlike other algorithms with non-deterministic search functionality, the PSO algorithm has a fixed search space, with parameters that take their values within the scope of this fixed search space without exceeding it [26]. The basic PSO algorithm, consists of **n** number of particles in the swarm (a.k.a, the population of candidate solutions). Each particle has a position x and velocity v in the hyperspace of the D-dimension. The iteration period evaluates the particles in it. Furthermore, the fitness function measures the efficacy of a given candidate solution during the iteration. As illustrated in Figure 4, the process of the PSO algorithm is as follows: each particle in the population is assigned a random position and velocity. Afterwards, the fitness function for each particle in the population is calculated. For each particle the P_{best} initially is the fitness value of the particle, whereas the G_{best} is the fitness of the P_{best}. After changing the position and velocity of a particle, its fitness is calculated, if the new value of the fitness is better than P_{best}, the new fitness becomes the $P_{best,}$ otherwise, the P_{best} remains and the new one is ignored. After calculating the P_{best} for all particles, the new G_{best} is found by comparing the old G_{best} with the G_{best} of the all current P_{best}, the better of them will be the new G_{best}. Afterwards, position x and velocity v are updated for each particle in the population. Steps are repeated starting from calculating the fitness function for each particle, to updating the position and velocity for each particle until the maximum number of iterations is reached.

E. GA Versu PSO

Both GA and PSO algorithms possess strengths and advantages over each other. They are similar in many ways (e.g., they both employ multi-agent and probabilistic search) [11]. Also, they both start by establishing a random population, and use the fitness function to evaluate the population [5]. On the other hand, PSO algorithm utilizes individual memory, which saves the best solution for each particle and the best one among the swarm in each iteration. Meanwhile, in the GA, previous knowledge is lost as soon as the population changes [5].

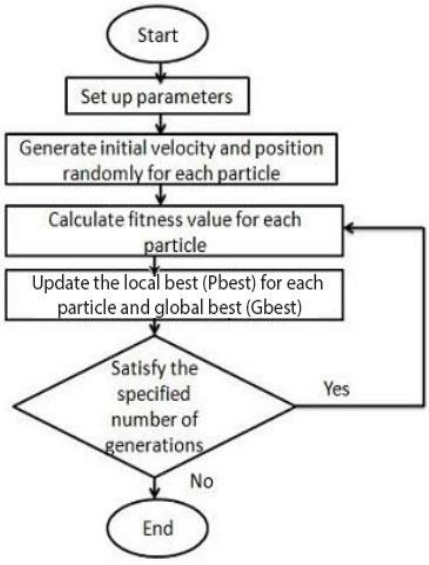

Figure 4. Structure of PSO algorithm

For individual operators, GA uses mutations, while the PSO algorithm uses P_{best} and velocity inertia. With regard to social operators, the GA uses the selection and crossover operators, while the PSO algorithm uses neighborhood P_{best} history. The search space in the GA is discrete, while it is continuous in the PSO algorithm (Goldberg and Holland, 1988). Also, they are different in how they represent populations, as the PSO algorithm can deal with real numbers, while the GA needs to convert those populations to binary encoding. Additionally, the PSO algorithm does not need to store the value of the fitness function like the GA. This can come in handy, especially when the size of the population is quite large. Based on the most recent research, the PSO algorithm outperforms the GA in most continuous

optimization problems [20]. The GA is still widely used in continuous and discrete optimization problems [5]. The GA can perform global searches in an efficient way and control active optimization. The PSO algorithm is a random search algorithm which enjoys simple calculations [3], strong local search and fast convergence velocity, but suffers from poor global search vis-a-vis the GA [20]. By capitalizing the strengths of each approach, we can reap the benefits of both the global search ability and high convergence speed, thus optimizing the outcome far more effectively [20].

2. RELATED WORK

In 2004, Juang [11] proposed a new algorithm called a Hybrid Genetic Algorithm Particle Swarm Optimization (HGAPSO). The technique relay on applying a certain procedure after calculating the fitness value for all the individuals. These individuals are sorted from the highest to lowest fitness value. The top half of the population results are selected as elites. These selected elites will be improved through the PSO algorithm. This group of elites is regarded as a swarm and each elite corresponds to a particle in it. These improved elites will form half of the population in the next generation. As for the other half, it will undergo the crossover and mutation operators in order to produce enhanced elites. This algorithm is implemented in temporal sequence production through fully connected recurrent neural networks and dynamic plant control problems with a Takagi-Sugeno-Kang-type recurrent fuzzy network. Moreover, testing proved the superiority of HGAPSO over GA and PSO on their own. In 2008, Chen, et al. [7] proposed a method that combines both the GA and the PSO algorithms. Their algorithm uses the PSO algorithm as an operator within the GA. So instead of the GA having three operators (selection, crossover, and mutation), now it has four. First, the selection, crossover, and mutation operators are run, and then the PSO is run for further improvement on the population. This algorithm was applied to the temperature prediction neural network in transverse flux induction heating. It was also implemented in electromagnetic engineering domains, and the results show unmistakable performance advantages over using the GA or the PSO algorithm on their own. In 2009, Kuo, et al. [17] proposed a new algorithm called the Hybrid of Particle Swarm and Genetic algorithm-based Optimization (HPSGO), aimed to improve the learning performance of radial-basis function neural networks (RBFNN). In this algorithm, the method followed is to first perform the PSO algorithm in one single iteration. Next, the chromosomes produced are duplicated, and the GA operators are run on them. After that, the two groups of chromosomes are represented as a new population (one of them comes from the GA and the PSO together, while the other comes from PSO only). This algorithm was tested on daily sales forecasts of papaya milk in actual industrial production, and it yielded far better results vis-à-vis other algorithms such as the GA, the PSO, and the Box-Jenkins model. In 2010, Caputo, et al. [6] proposed a new Genetic Swarm Optimization (GSO) algorithm. It is based randomly dividing the population into two batches in each iteration. The GA is applied to the first batch, while the PSO is applied to the second. The fitness function is evaluated for each algorithm's generated population. After that, both are combined once again. The same process happens for each iteration. This algorithm has been tested on engineering problems and has shown the potential for use in this field. In 2011, Xin-qiu and Yan-sheng [30] produced an algorithm named Genetic Particle Swarm Optimization (GPSO), combining the GA and PSO algorithms and based on the concept of adding the GA crossover and mutation operators inside the PSO algorithm. Firstly, the BP algorithm is applied on the tandem cold rolling force prediction ANN model. Second, adding GA operators inside the PSO in order to improve the results. Its performance was markedly advantageous when compared to the basic PSO and GA algorithms. In 2011, Abd-El-Wahed [1] produced a novel algorithm, embedding the GA within the PSO algorithm. This approach starts by applying the PSO, followed by the GA. Between the regular operators of either algorithm, there are additional operators, such as ranking, elitist strategy, and repairing operators. This algorithm has been tested on non-linear problems and has displayed superior results over the basic PSO algorithm. In 2012, Qian, et al. [21] proposed a new algorithm called Genetic Algorithm-Particle Swarm Optimization- Radial Basis Function (GA-PSO-RBF) and it is based on another algorithm proposed by Xin-qiu, and Yan-sheng [30] and tested it on a different type of ANN model and different domains. It combines both the GA and PSO algorithms to improve the Radial-Basis-Function Neural Network (RBFNN). The algorithm was applied for fault diagnosis in a generator unit. The motivation for establishing this algorithm was to avoid the drawbacks of the Particle Swarm Optimization-Radial Basis Function (PSO-RBF) algorithm. Since PSO algorithm has very slow convergence rate and early-maturing problems which effect training process. The newly proposed algorithm has shown clear performance benefits over PSO-RBF in training speed, convergence accuracy and diagnosis accuracy. In 2017, Hewahi and Abu Hamra [14] produced a new algorithm that applies the BP algorithm on the ANN, then improves the results by applying the GA followed by the PSO algorithm and so on in a loop. They started by taking **n** number of ANN best results and putting them in one set, and then the loop of GA followed by PSO starts. Iteration after iteration, the size of the set is reduced until the maximum number of iterations is

reached. At the end, the individuals with the highest fitness value is selected to be the best solution. The algorithm results have been compared with other results of previous research and have shown superior performance in various domains.

Based on the previous studies, it has been noticed that most of the attempts use a combination of GA and PSO algorithms for only certain domains except the work developed by Hewahi and Abu Hamra [14] and the work proposed by Chen, et al. [7], they are domain independent. Moreover, some trails were related to RBFNN applied on certain application domains. Our proposed approach is different from other approaches in the order of applying the GA and PSO algorithms, it is totally random which gives more opportunity to improve the classification. Another issue in which our proposed approach differs from others is that it has a competition between two sets of population during the PSO process forming at the end one robust set maintain the same set size and giving more opportunity to some individuals that have potential to improve.

3. THE PROPOSED APPROACH

Our proposed algorithm Random PSO and GA Classifier (RPSOGAC) is explained in a detailed and precise manner in this section. The main concept of our approach is to increase the classification of ANNs based on random application of GA and PSO algorithms to modify the ANN weights.

A) Methodology

The proposed algorithm RPSOGAC can be summarized as below:

1) The overall solution structure: The general structure of the proposed algorithm is shown in Fig.5, and explained as below:

a) Define and introduce the datasets to be used for classification.

b) Create ANN structure based on the dataset structure and apply BP algorithm on ANN until an acceptable solution is reached (target error) if possible, if an acceptable solution is reached, the algorithm quits and solution is found, otherwise, do as following:

c) Create a population set and call it set(1) with size ***n****, where each individual in the population is generated at random and represents random weights of the ANN. The last obtained weights from the BP algorithm are also included in the population set as an individual.*

d) Set the number of iterations to zero.

e) Repeat until number of iteration is equal to max number of iteration.

- Generate a random number **r** between 0 and 1.
- If **r** $\geq$ 0.5, apply GA

  Else apply PSO.

f) Select the best weights individual to be the winner individual used for any further classification.

2) GA Algorithm

The used GA is as shown in Fig. 6 and explained below:

Input: set(1).

Output: modified set-1 after applying GA.

Procedures:

a. Set the GA_iteration to zero.
b. Repeat until GA_iteration is equal to GA_Max_iteration.
   Apply GA on individuals and calculate the fitness value.
c. Set(1) will have at the end the best individuals obtained by GA.

3) PSO Algorithm

The used PSO algorithm is as shown in Fig. 7 and explained below:

Input: set(1).

Output: modified set(1) after applying PSO.

Procedures:

a. Set the PSO_iteration to zero.
b. Create set(2) based on random generations of weights for individuals to have the same number of individuals in set(1).
c. Repeat until PSO_iteration is equal to PSO_Max_iteration.
   1. Calculate fitness value for each particle in set(1) and set(2).
   2. Get P_{best} of each set (i.e., the best performing individual) and modify G_{best} (i.e., the best of the two P_{best}) if necessary.
   3. Update position and velocity of each particle in each set.
d. Modify set(1) to have the best individuals from set(1) and set(2).

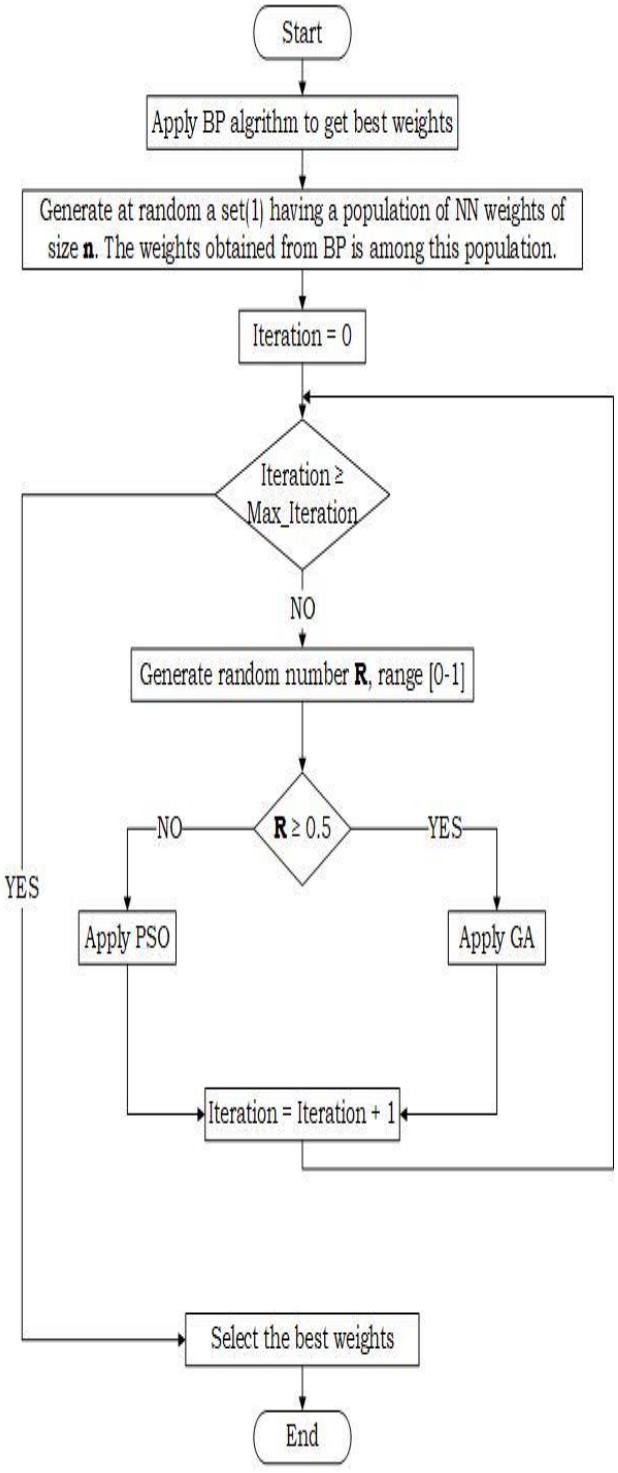

Figure 5. RPSOGAC algorithm flowchart

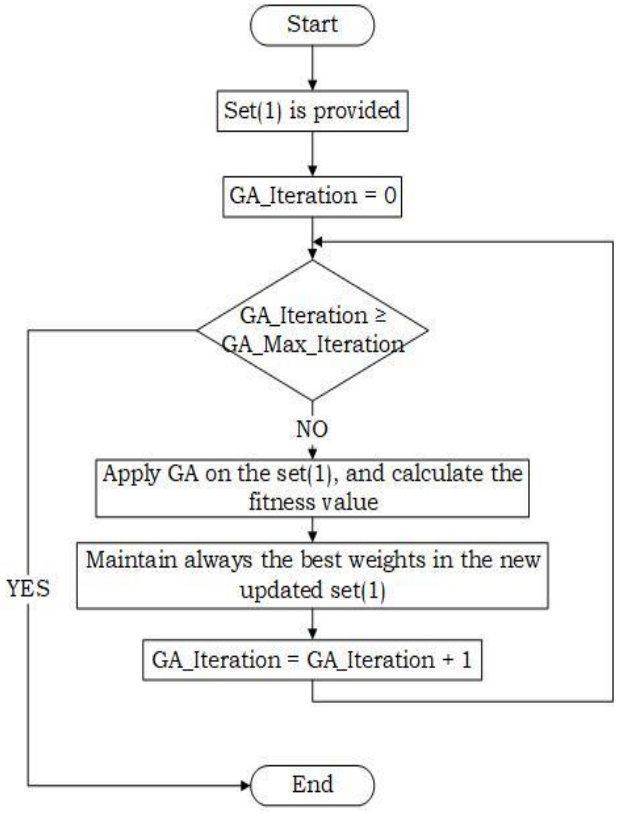

Figure 6. Genetic algorithm flowchart.

4. EXPERIEMENTAL RESULTS AND EVALUATIONS

To examine our proposed algorithm, the ANN is tested alone without the support of any optimization algorithm. In addition, ANN supported with our proposed algorithm is tested. In addition, **RPSOGAC** approach results are compared with the results obtained in [14] and in [7].

A. Choosing Problem Domain

The **RPSOGAC** algorithm is applied on six datasets obtained from UCI [2]. Selected datasets are from various domains with different specification and characteristics. Each dataset has a different number of attributes, attributes types, and number of instances. Table 1 represents each dataset with its characteristics.

Dataset name	Network structure (No. of input neurons – No. of hidden neurons – No. of output neurons).
Nursery	8-2-5
Balance Scale	4-2-3
Car Evaluation	6-2-4
Ecoli	7-2-8
Glass Identification	8-2-2
Contraceptive Method Choice	9-2-3

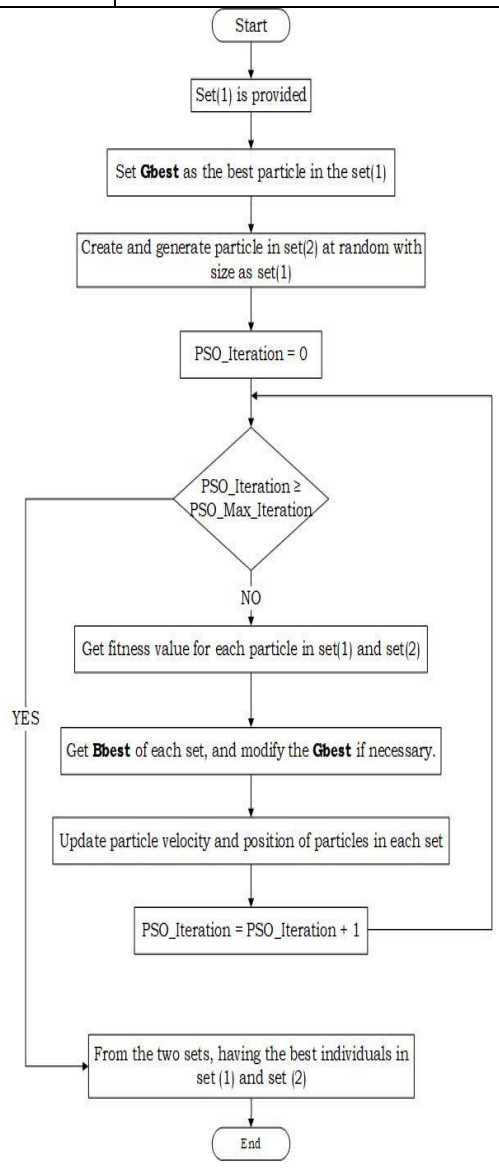

Figure 7. Particle swarm optimization algorithm flowchart.

TABLE 1. CHARACTERISTICS OF DATASETS.

Dataset name	# of instances	# of attributes without the class	Attributes type	Domain
Nursery	12960	8	Categorical	Social
Balance Scale	625	4	Integer	Social
Car Evaluation	1728	6	Categorical	Business
Ecoli	336	7	Real	Life
Glass Identification	214	9	Real	Physical
Contraceptive Method Choice	1473	7	Categorical, Integer	Life

The neural network structures used for the datasets are as shown in Table 2.

Table 2. Network structure for datasets (The used representation is No. of input of hidden neurons – No. of output neurons).

B. Experiementation Parameters

Table 3 shows all parameter values used in the **RPSOGAC** algorithm.

C. Experiemental Results

Under this section we present two types of experiments, the first one is comparing our RPSOGAC with only the classifier using BP algorithm and the second experiment is a comparison between RPSOGAC and other proposed algorithms.

TABLE 3. PARAMETERS OF PROPOSED ALGORITHM.

Parameter	Description
BP algorithm	
Learning rate = 0.3	Used to control the amount of weight change in each update iteration and represent how quickly a network changed old beliefs for new ones.
Momentum = 0.2	Used to avoid algorithm from getting stuck in a local minimum problem.
Minimum error (MSE) = 0.001	Stand for "Mean Squared Error" which estimator measure the average of the squares of the errors.

GA algorithm	
Mutate rate = 0.3	Control the number of genes to mutate in the children chromosome.
Crossover rate = 0.5	Control the number of parent chromosomes to crossover genres between them depending on the crossover points to produce new children.
Tournament selection value = 0.4	Used to help in the selection process for choosing the best two chromosomes in the population and they will be the parents for the new generation.
PSO algorithm	
Inertia coefficient (w) = 0.7	Used to determines the influence of the current velocity.
c1 and c2 = 1.4	Cognitive and social weights, they used for determine the influence of particle's best position and best position in the swarm.
r1 and r2 (random values in range [0,1])	They used to affect the particle's movement and prevent it to become stuck.
GA and PSO algorithms	
Population size = 50	Size of the population or the swarm.
BP, GA and PSO algorithms	
Maximum number of iterations = (changeable)	The maximum number of iterations in algorithms, which is not fixed number and changeable.

1) *RPSOGAC and BP*

Table 4 shows the results accuracy using only the BP algorithm without considering any of the optimization algorithms and the results obtained by RPSOGAC. Accuracy in Table 4 and next tables is computed as the number of correctly classified instances by the number of instances in testing portion of the dataset. As shown in Table 4, the RPSOGAC algorithm outperforms the classification accuracy of ANN alone, even when ANN alone results 's accuracy is high such as Nursery and Balance Scale datasets. Furthermore, in datasets like Glass Identification, Contraceptive Method Choice, and Ecoli which have low accuracy with only using BP algorithm, the RPSOGAC could make a significant jump in accuracy percentage. It is noticed that RPSOGAC could make a big jump in classification accuracy especially in Contraceptive Method Choice, but due to the small number of instances compared with the number of attributes, and each attribute has several values, the classifications wouldn't be very high.

TABLE 4. COMPARISON BETWEEN ANN AND OUR ALGORITHM RESULTS.

Dataset name	ANN Accuracy Percentage	RPSOGAC algorithm percentage
Nursery	91.82	100
Balance Scale	88.37	98
Car Evaluation	70.79	96.37
Ecoli	50.92	89.53
Glass Identification	31.63	74.73
Contraceptive Method Choice	43.57	50.16
Average	**62.85**	**84.79**

2) *RPSOGAC and other Approaches*

In this section, we present the results of other previous methods that have used the same datasets shown in Table 5. Also, we compare these results with our proposed approach. These methods are Hewahi and Abu Hamra's algorithm [14] and Chen, et al. algorithm that uses GA followed by PSO [7]. Table 5 shows the results of the three algorithms.

TABLE 5. COMPARISON BETWEEN HEWAHI AND ABU HAMRA, CHEN, ET AL. AND RPSOGAC ALGORITHMS.

Dataset name	RPSOGAC algorithm	Hewahi and Abu Hamra's algorithm	Chen, et al. algorithm
Nursery	100	100	97.49
Balance Scale	95.92	98.4	96
Car Evaluation	93.37	95.84	95.95
Ecoli	89.53	86.76	82.35
Glass Identification	74.73	62.79	58.14
Contraceptive Method Choice	50.16	21.02	18.98
Average	**83.95**	**77.47**	**74.81**

As illustrated in Table 5, RPSOGAC algorithm gives in general better results than the other two algorithms in most of the cases. Fig. 8 shows a comparison between the results of RPSOGAC, Hewahi and Abu Hamra, and Chen, et al. algorithms.

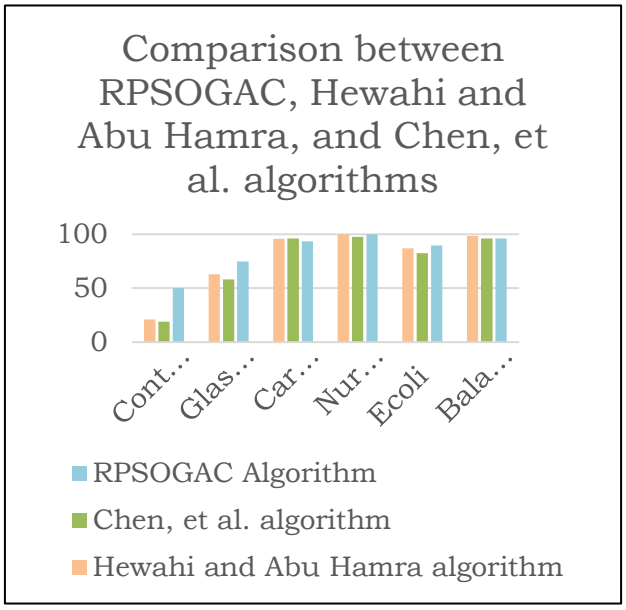

Figure 8. Comparison results between RPSOGAC, Hewahi and Abu Hamra, and Chen, et al. algorithms.

3) Results Discussion

In general, according to the obtained results, we can conclude the that the RPSOGAC algorithm provides better results and performance compared to other algorithms in most of the cases (four out of six datasets). In cases where the other algorithms obtain higher classification accuracy, the difference with the RPSOGAC classification results are minor. Using ANN alone, the classification accuracy was very good for some datasets and for others was low. This may happen for many reasons, among these could be the ratio of the number of attributes to the number of instances, in addition, does the nature of the data going to lead to a generalization or not? The RPSOGAC algorithm uses the same idea of Hewahi and Abu Hamra's algorithm [14], nevertheless, there are some major differences. In the RPSOGAC algorithm, the population is created by taking the best result of one ANN (as one individual) and fill the other individual in the population by random numbers. The algorithm then keeps switching between applying GA and PSO algorithms on the set based on a random number. In addition, the size of the set is kept as it is. While in Hewahi and Abu Hamra's algorithm, the set is created by taking the best results of **k** number of ANNs, and keep applying the GA followed by PSO algorithm on the set, then again after PSO, the GA is applied and so on. Furthermore, the size of the set is reduced to the half in each iteration. Hewahi and Abu Hamra's algorithm starts strongly by taking the best results of **k** number of ANNs but keep removing the weak individuals in the set (population or swarm) in each iteration. This process might not be always proper because the candidate solutions can improve at any moment and make a huge difference in the classification accuracy. In addition, one major difference between RPSOGAC and Hewahi and Abu Hamra's algorithm is that in RPSOGAC algorithm; a competition process between the two set (1) and set (2) are performed during the PSO portion to come up at the end with a one set that is having the best individuals. This process ensures always the best individuals will be maintained with also those which could be improved in close future. To show the capability of RPSOGAC, we need to notice that the result of classifying Ecoli, and Glass Identification, and Contraceptive Method Choice datasets were 50.92, 31.63, and 42.57 respectively. After applying the RPSOGAC algorithm a great jump happened as follows: 89.53, 74.73, and 50.16 for the same datasets respectively. In case of obtained high results using ANN with BP algorithm, RPSOGAC algorithm shows that it can still improve them very well. For example, using ANN with BP algorithm, the results for Nursery, Balance Scale, and Car Evaluation were 91.82, 88.37, and 70.79 respectively. After applying RPSOGAC algorithm the results of the same datasets have improved to 100, 92.68 and 84 respectively.

5. Conclusion

In this paper, a hybrid biology-inspired algorithm RPSOGAC has been proposed to improve the classification accuracy of ANN by solving the problem caused by local minima. The local minima problem affects the training stage by not allowing the classification accuracy to increase despite that the global maxima is still far. This will ultimately make the trained ANN does not perform well in the testing stage.

The proposed algorithm begins by applying the BP algorithm on the ANN to land at the best solution that provides adequate classification accuracy. The weights of this ANN are used as an individual in the initial population. The algorithm then generates random individuals and adds them to the population. After that, the algorithm keeps switching randomly between the GA and PSO algorithms until it reaches to an acceptable best solution. During GA, two-point crossover and mutation genetic operators are applied to get new weights individuals. In the PSO stage the algorithm uses a population set obtained from the GA algorithm and another population set generated at random. A competition between the two sets to obtain the G_{best} particle is performed. At the end of PSO stage, a new population set having the best particles from the two sets

are formed. The new set will also contain the particles that are expected to improve giving them an opportunity to be a possible solution.

To evaluate and examine RPSOGAC, it has been applied on six datasets related to various domains. These datasets have been obtained from the UCI datasets repository. In the first experiment, all the datasets have been used with only PB algorithm. In the second experiment, two tests have been conducted, one using the BP algorithm followed by PSO algorithm, and the other using BP followed by GA algorithm. In the third experiment, RPSOGAC algorithm is performed. At the end, a comparative study between three algorithms have been presented, these algorithms are RPSOGAC, Hewahi and Abu Hamra's algorithm and Chen, et al. 's algorithm.

The obtained results show that RPSOGAC could make a big classification jump compared with only using BP, BP with PSO or BP with GA even in cases where the obtained results are very high such as in Nursery and Balance Scale datasets where the results using only BP for example were 91.82 and 88.37 respectively and became 100 and 95.2 in the same order. In case the classification accuracy results of all the previous mentioned methods are low, RPSOGAC could also make a big jump and improved the classification accuracy such as in Ecoli, Class Identification, Contraceptive Method Choice datasets where the results using only BP for example were 50.92, 31.63 and 43.57 respectively and became 89.53, 74.73 and 50.16 in the same order.

The obtained results of RPSOGAC compared with the results of Hewahi and Abu Hamra's algorithm and Chen, et al. 's algorithm show that RPSOGAC outperforms the obtained results using Chen, et al. 's algorithm in all the data sets except in Car Evaluation dataset with a small difference. For RPSOGAC results compared with Hewahi and Abu Hamra 's algorithm, in four datasets out of six datasets, RPSOGAC outperforms Hewahi and Abu Hamra's algorithm, these datasets are Nursery, Ecoli, Glass Identification and Contraceptive Method Choice datasets. In case of the other two datasets where Hewahi and Abu Hamra's algorithm performs better than RPSOGAC, the classification accuracy difference is not much. These datasets are Balance Scale and Car Evaluation datasets where the results were 98.4 and 95.84 using Hewahi and Abu Hamra's algorithm, whereas using the RPSOGAC algorithm the results were 95.92 and 93.37 respectively. Some of the future directions would be testing RPSOGAC on more datasets and several domains and incorporating other biology inspired algorithms to improve the classification accuracy such as Ant Colony Optimization (ACO), Fish Swarm Optimization (FSO), Bee Colony Optimization (BCO), bat algorithm and island bat algorithm, and examining RPSOGAC with different ANN models such as deep Learning, Convolution Neural Networks (CNN), Recursive Neural Networks (RNN), and Recurrent Neural Networks (RNN).

REFERENCES

[1] W. Abd-El-Wahed, A. Mousa, and M. El-Shorbagy, "Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems", Journal of Computational and Applied Mathematics, 235(5), pp. 1446–1453, 2011.

[2] Archive.ics.uci.edu. UCI Machine learning repository: Data sets. [Online] Available at: https://archive.ics.uci.edu/ml/datasets.html [Accessed 18 March 2018].

[3] Q. Bai, "Analysis of particle swarm optimization algorithm", Computer and Information Science, 3(1), pp. 180–184, 2010.

[4] W. Bi, X. Wang, Z. Tang, and H. Tamura , "Avoiding the local minima problem in backpropagation algorithm with modified error function", IEICE Transactions on Fundamentals of Electronics, *Communications and Computer Sciences*, E88–A(12), pp. 3645–3653, 2005.

[5] K. Borna and R. Khezri, "A combination of genetic algorithm and particle swarm optimization method for solving traveling salesman problem", *Cogent Mathematics*. 2(1), pp. 1–13,2015.

[6] D. Caputo, F. Grimaccia,, M. Mussetta. and R. Zich, Photovoltaic plants predictive model by means of ANN trained by a hybrid evolutionary algorithm, in International Joint Conference on Neural Networks (IJCNN). IEEE, 2010, pp. 1–6.

[7] T. Chen, Y. Wang, L. Pang, J. Sun, and A. Jinlong,, A New Hybrid Genetic Algorithm and Its Application to the Temperature Neural Network Prediction in TFIH,. *Automation Congress Conference*, 2008, pp. 1-4.

[8] *S. Dreyfus "Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure", Journal of Guidance, Control, and Dynamics, 13(5), pp. 926–928, 1990.*

[9] *A. Escalante-B, and L. Wiskott, "How to Solve Classification and Regression Problems on High-Dimensional Data with a Supervised Extension of Slow Feature Analysis", Journal of Machine Learning Research, 14, pp. 3683–3719, 2013.*

[10] K. Fuchs,Machine Learning: Classification Models. [Online] Available at: https://medium.com/fuzz/machine-learning-classification-models-3040f71e2529 [Accessed: 29 February 2018],

[11] D. Goldberg, and J. Holland, "Genetic Algorithms and Machine Learning", *Machine Learning*, 3(2), pp. 95–99, 1988.

[12] R. Haelterman, Analytical study of the Least Squares Quasi-Newton method for interaction problems. [Online] Available at: https://biblio.ugent.be/publication/720660 [Accessed: 12 April 2018], 2009.

[13] M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems", Journal of Research of the Natural Bureau of Standards, 49(6), pp. 409–436, 1952.

[14] N. Hewahi and E. Abu Hamra, "A Hybrid Approach Based on Genetic Algorithm and Particle Swarm Optimization to Improve Neural Network Classification" , Journal of Information Technology Research, 10(3), pp. 48–68, 2017.

[15] C. Juang, "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design", IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 34(2), pp. 997–1006, 2004.

[16] N. Kang, *Multi-Layer Neural Networks with Sigmoid Function—Deep Learning for Rookies (2)*. [Online] Available at: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

[17] R. Kuo, T. Hu, and Z. Chen,(2009). Application of Radial Basis Function Neural Network for Sales Forecasting, 2009 International Asia Conference on Informatics in Control, Automation and Robotics. IEEE, 2009, pp. 325–328.

[18] B. Liu, C. Jin, J. Wan, and Y. Huanxi, "Modeling and optimizing an electrochemical oxidation process using artificial neural network, genetic algorithm, and particle swarm optimization", Journal of the Serbian Chemical Society, 83(03), pp. 379–390, 2018.

[19] A. Norouzi, M. Hamedi, and V. Adineh, "Strength modeling and optimizing ultrasonic welded parts of ABS-PMMA using artificial intelligence methods", International Journal of Advanced Manufacturing Technology, 61(1–4), pp. 135–147, 2012.

[20] C. Ou and W. Lin, Comparison between PSO and GA for Parameters Optimization of PID Controller, in *2006 International Conference on Mechatronics and Automation*. IEEE, 2006, pp. 2471–2475.

[21] Y. Qian, H. Zhang, D. Peng, and C. Huang, Fault diagnosis for generator unit based on RBF neural network optimized by GA-PSO, in *2012 8th International Conference on Natural Computation*. IEEE, 2012, pp. 233–236.

[22] P. Sharma, 'Optimizing Back-Propagation using PSO _ Hill _ A * and Genetic Algorithm', *International Journal of Computer Applications*, 71(17), pp. 35–41, 2013.

[23] K. Stanislawska, K. Krawiec, and Z. Kundzewicz, "Modeling global temperature changes with genetic programming", *Computers & Mathematics with Applications*, 64(12), pp. 3717–3728, 2012.

[24] Thinkingandcomputing.com., *Training neural networks: back-propagation vs. genetic algorithms | Thinking and Computing*. [Online] Available at:

https://thinkingandcomputing.com/posts/genetic-algorithms-neural-networks.html [Accessed: 16 March 2018].

[25] C. To and J. Vohradsky "A parallel genetic algorithm for single class pattern classification and its application for gene expression profiling in Streptomyces coelicolor"., *BMC genomics*. BioMed Central, 8:49, PP.1-13, .2007.

[26] B. Vallade and T. Nakashima, Improving the Performance of Particle Swarm Optimization Algorithm With a Dynamic Search Space, The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 43–48, 2013.

[27] T. Vidal, T. Crainic,, M. Gendreau, N. Lahrichi, and W. Rei, "A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems", Operations Research. INFORMS, 60(3), pp. 611–624, 2012.

28]S. Wang, Artificial Neural Network, Interdisciplinary Computing in Java Programming. Boston, MA: Springer US, pp. 81–100, 2003.

[29] K. Wong, C. Peng, M. Wong, and K. Leung, "Generalizing and learning protein-DNA binding sequence representations by an evolutionary algorithm", Soft Computing, 15(8), pp. 1631–1642, 2011.

[30] *Z. Xin-qiu, and W. Yan-sheng, BP neural network based GPSA used in tandem cold rolling force prediction, International Conference on Consumer Electronics, Communications and Networks (CECNet). IEEE, 2011, pp. 4829–4832.*

[31] X. Yu, M. Efe, and O. Kaynak, "A general backpropagation algorithm for feedforward neural networks learning", IEEE Transactions on Neural Networks, 13(1), pp. 251–254. 2002.

Nabil M. Hewahi obtained his PhD degree in Computer Science from Jawaharlal Nehru University, New Delhi, India in 1994, and M.Tech degree in Computer Science and Engineering from Indian Institute of Technology, Bombay, India in 1991. He is now with the university of Bahrain, Bahrain and the Islamic University of Gaza, Palestine. Dr. Hewahi is a full professor of Computer Science (Artificial Intelligence) since 2006. He published over 65 papers in well-known journals and conferences. His main research interest is intelligent systems including machine learning and knowledge representation.

Zahraa J. Jaber obtained her BSc degree in Computer Science from the University of Bahrain, Sakhir, Bahrain in 2018. She is passionate about machine learning, system analysis.