Satisfiability Logic Analysis Via Radial Basis Function Neural Network with Artificial Bee Colony Algorithm

Mohd Shareduwan Mohd Kasihmuddin[1], Mohd. Asyraf Mansor [2]*, Shehab Abdulhabib Alzaeemi[1], Saratha Sathasivam[1]

[1] School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang (Malaysia)
[2] School of Distance Education, Universiti Sains Malaysia, 11800 USM, Penang (Malaysia)

Received 6 October 2019 | Accepted 7 May 2020 | Published 25 June 2020

Abstract

Radial Basis Function Neural Network (RBFNN) is a variant of artificial neural network (ANN) paradigm, utilized in a plethora of fields of studies such as engineering, technology and science. 2 Satisfiability (2SAT) programming has been coined as a prominent logical rule that defines the identity of RBFNN. In this research, a swarm-based searching algorithm namely, the Artificial Bee Colony (ABC) will be introduced to facilitate the training of RBFNN. Worth mentioning that ABC is a new population-based metaheuristics algorithm inspired by the intelligent comportment of the honey bee hives. The optimization pattern in ABC was found fruitful in RBFNN since ABC reduces the complexity of the RBFNN in optimizing important parameters. The effectiveness of ABC in RBFNN has been examined in terms of various performance evaluations. Therefore, the simulation has proved that the ABC complied efficiently in tandem with the Radial Basis Neural Network with 2SAT according to various evaluations such as the Root Mean Square Error (RMSE), Sum of Squares Error (SSE), Mean Absolute Percentage Error (MAPE), and CPU Time. Overall, the experimental results have demonstrated the capability of ABC in enhancing the learning phase of RBFNN-2SAT as compared to the Genetic Algorithm (GA), Differential Evolution (DE) algorithm and Particle Swarm Optimization (PSO) algorithm.

Keywords

Artificial Bee Colony Algorithm, Radial Basis Function Neural Network, 2 Satisfiability, Logic.

DOI: 10.9781/ijimai.2020.06.002

I. Introduction

Generally, Artificial Intelligence (AI) can be encompassed in some functional graphical and mathematical models that act as a symbolic system [1]. Greater impact can be achieved when symbolic operations have been integrated inside the Artificial Neural Network (ANN). ANN known as the conductive system has received careful attention due to its ability to evaluate the complex nonlinear dataset [1]. ANN has been successfully used in solving non-limited applications such as classification and optimization of approximation functions. However, the functionality of ANN can be measured by embedding the correct symbolic rule to govern the whole neural system. Logic programming has been a language of ANN for decades. Wan Abdullah [2] successfully explored the neural network that has been governed by logic programming. In this work, logical rule embodied ANN and the characteristic of the network will be examined by using Lyapunov energy analysis. The minimization of energy as a solution to the combinatorial representation motivates the integration of logical rules in a neural network [3]-[6]. The question remains on how one can choose the best ANN model in order to embed logic programming.

The reliable ANN model typically has the least prediction and classification error analysis. In that regard, Radial Basis Function Neural Network (RBFNN) fascinated the researchers from sciences and engineering field because of simpler networks structure, faster learning speed and better approximation capabilities. As stated by Hamadneh *et al.* [7] in their paper, RBFNN can be used to develop separate models for the shear stress and heat transfer rate due to simpler networks structure. RBFNN is a feedforward neural network that contains 3 neuron layers (input, hidden and output layers). The input layer (containing input neurons) receives information being transferred to the hidden layer for data synthesis and training. The synthesized data will be used in the output layer (containing output neuron). The foundation of having 3 layers is to minimize the classification and prediction error in RBFNN [8]. Hamadneh *et al.* [4] initially implemented logic programming in RBFNN. Their proposed network explored the capability of HornSAT as a logical rule in RBFNN. In this case, logical structure of RBFNN is solely dependent on 3 parameters: the center of all input neurons, its widths, and its Gaussian activation function. Despite the fact that RBFNN can be applied effectively, the number of neurons in a hidden layer for RBFNN will determine the complexity of the network [9]. If the number of neurons in the hidden layers is not enough, the learning in RBFNN fails to achieve optimal convergence. However, if the number of neurons in the hidden layers is very high, the network will experience overlearning [10]. Since the complexity of RBFNN increases as the number of clauses increase, an optimization algorithm becomes crucial.

Yang and Ma [11] have successfully applied the Sparse Neural Network (SNN) algorithm for optimizing the number of hidden neurons. The core mechanism of SNN is in reducing the error via trial

* Corresponding author.
E-mail address: asyrafman@usm.my

and error approach for determining the number of hidden neurons explicitly from the set of neurons. The limitation of SNN paradigm can be seen in extensive computational time during the number of hidden neuron computation process. Inspired by several works of [12]-[14], the 2 Satisfiability (2SAT) logic representation will be utilized with RBFNN to determine the important parameters for the hidden layer that control the number of hidden neurons. In fact, 2SAT is selected since it is complying with RBFNN based on the structure and representations.

Another major component of 2SAT in RBFNN is the training method that has a significant influence on the performance of RBFNN. On this matter, a plethora of global optimization methods have been extensively applied due to their global search capability. Metaheuristics algorithm is a popular algorithm to search for a near optimal solution for RBFNN [15], [16]. There are various nature-inspired and recently developed optimization algorithms such as Genetic Algorithm, Differential Evolution algorithm, Particle Swarm Optimization algorithm, Artificial Bee Colony, etc. and many of these proved their suitability to many engineering optimization problems [17].

The theoretical basis of the Genetic Algorithm (GA) has been developed by Holland [18]. The first who used GA in a problem involving the control of gas-pipeline transmission were Goldberg and Holland [19]. Other studies have been made by Hamadneh *et al.* [4] who used GA to train the hybrid model RBFNN with higher-order SAT logic. In this study, they used the full training paradigm to train RBFNN with higher-order SAT logic using k-means cluster algorithm and GA. The quest of finding the optimal algorithm was continued by Pandey *et al.* [20] who compared Multiple Linear Regression (MLR) and genetic algorithm to predict temporal scour depth near-circular pier in non-cohesive sediment. This study utilized 1100 laboratory experimental data-sets to develop the generalized scour equation using MLR and GA. In recent publications, Jing and Li [21] developed a reliability analysis method by integrating GA with RBFNN. This paper adopted GA to find the "potential" most probable point (MPP) in the optimization problem by control the density of samples to refine the RBFNN.

Differential evolution (DE) was first introduced by Storn and Price [22] to solve the various global optimization problems. DE is a manageable yet powerful evolutionary algorithm with the advantages of less parameter, high simplicity, and fast convergence [22]. DE has been beneficial to various networks such as Hopfield Neural Network [23] and feed-forward neural networks [24]. Chauhan & Chandra [22] proposed the DE algorithm to train a wavelet neural network (WNN) by minimizing network error to obtain the proper relationship from the input vector in the input layer to the output vector in the output layer. Tao *et al.* [25] utilized the DE algorithm to improve RBFNN as the prediction model for the coking energy consumption process. Particle Swarm Optimization algorithm (PSO) is a nature-inspired evolutionary algorithm that imitates the influence of bird migration behavior [26]. PSO algorithm is one of the evolutionary algorithms proposed by Kennedy and Eberhart [27]. In some succeeding works, Qasem & Shamsuddin [28] proposed the PSO algorithm for enhancing RBFNN learning by optimizing the parameters of the hidden layer and output layer. Another study has been made by Alexandridis *et al.* [29], who used the PSO algorithm to optimize the construction of RBFNN. The proposed model was able to solve classification problems and solve function approximations with improved generalization capabilities and accuracy.

Karaboga and Basturk [30], [31] proposed the Artificial Bee Colony algorithm (ABC) to gain computational edge in optimizing the capability of both local search and global search. ABC was inspired by collective behaviors of bees gathering honey in an optimized pattern. ABC has been beneficial to various networks such as Hopfield Neural Network [14] and Hermite Neural Network [32]. Kurban & Besdok [33] utilized ABC to estimation the centers, width, and weights as the main parameters of RBFNN. Yu and Duan [34] proposed an optimized ABC in RBFNN integrated with Fuzzy C mean Clustering. In this paper, 2 layers of optimization in ABC were reported to increase the accuracy of the image fusion. Jafrasteh and Fathianpour [35] proposed hybrid RBFNN by introducing perturbation in ABC. The proposed system was reported to capture non-linear relationship in ore grade data. In another development, Satapathy *et al.* [36] combined the benefit of kernel trained ABC to further optimize the capability of RBFNN. The proposed RBFNN managed to increase the classification accuracy of EEG signal for epileptic seizure identification. The perspective has been expanded by Aljarah *et al.* [37] when they introduced hybrid ABC with RBFNN to solve well known datasets. On the perspective of logic programming in RBFNN, little studies have been done to optimize the parameter of RBFNN by using ABC. Kasihmuddin *et al.* [14] has demonstrated the ability of ABC to serve as an effective learning algorithm in Hopfield Neural Network (HNN). One of the notable use of ABC is proposed by Jiang *et al.* [30]. In this work, the ABC is employed for optimizing the parameters of RBFNN and predicting the ecological pressure. In another development, Menad *et al.* [38] have utilized the RBFNN framework with ABC algorithm (RBFNN-ABC) for predicting the carbon dioxide solubility and concentration in brine. The results manifested the capability of ABC in optimizing RBFNN that result in higher accuracy. By hybridizing RBFNN with 2SAT logic, here we examine the effects of ABC on the training phase as a single framework, RBFNN-2SATABC. Worth noting that the proposed model will be compared with the existing models. Thus, the main motivation of employing ABC in this research is due to:

1. According to Kasihmuddin *et al.* [14], [62], ABC has outperformed the other algorithm such as [5] and [6] in enhancing the training phase for bipolar 2SAT logical representation. We extended the non-binary representation for optimizing the parameter entrenched in the hidden layer of RBFNN as inspired by the binary operators consist of employed bees and onlooker bees' phase.

2. Several current studies such as Menad *et al.* [38] and Jiang *et al.* [39] utilize the ABC in optimizing the prediction capability of RBFNN. Both local search and global search capability reduce the chances for ABC to achieve sub-optimal fitness. Motivated by these recent works, ABC algorithm is applied in improving the output quality from the output weight thereby improving the performance of the structure RBFNN-2SAT.

To this end, the contributions of this paper are as follows:

1. This paper explores another perspective in approaching implicit knowledge by using an explicit learning model. Real-life problem (implicit representation) is learnable by using a set of explicit mathematical representation (2SAT logical rule).
2. This is the first attempt to embed 2SAT logical rule (knowledge) to the feed-forward neural networks (learner). In this study, the 2SAT logical rule has been embedded in RBFNN by systematically obtaining the optimal value of parameters (center and width). 2SAT logical rule is expected to optimize the structure of the RBFNN by fixing the number of hidden neurons involved.
3. Since the training of the proposed RBFNN always converges to suboptimal output weight, this paper will explore the capability of Artificial Bee Colony (ABC) compared to other existing established metaheuristics. The aim of the training model in RBFNN is to obtain the optimal output weight with the lowest iteration error. Extensive experimentation with various performance metrics has been conducted to reveal the effectiveness of ABC in the proposed RBFNN-2SAT.
4. The proposed RBFNN provides an interesting perspective. RBFNN obtained the output weight of 2SAT by minimizing the

objective function with the structurally systematic parameters. This approach is interestingly different from Sathasivam [40] that utilized the Wan Abdullah method in finding the correct synaptic weight (output weight). Although both paradigms utilized ABC in optimizing the proposed methods, the method proposed in this paper deals with non-binary optimization compared to the existing method. Therefore, the proposed method creates a new possible horizon for logic programming in the neural networks.

The rest of this paper is arranged as follows. The 2SAT logical rule is formulated in the first section. After the overview structure of the general RBFNN, the proposed hybrid model integrated with 2SAT is constructed. Accordingly, the proposed training model via metaheuristics algorithm namely GA, DE, PSO, and ABC will be discussed in detail. Finally, this paper presents numerical results to show the effectiveness of ABC in optimizing 2SAT in RBFNN and we conclude the paper with some remarks and future work.

II. Boolean 2 Satisfiability Representation

Satisfiability (SAT) is demarcated as a logic rule with an array of clauses composed of binary literals. SAT is effectively governed by positive [5] and negative outcomes. The main structure of SAT representation is shown as follows:

(a) Consists of a set of m variables of $v_1, v_2, v_3, \ldots\ldots, v_m$.

(b) Composes of a set of literals. A literal refers to the variable v or a negation of a variable, $\neg v$.

(c) A set of n discrete clauses, $l_1, l_2, l_3, \ldots, l_n$. Every single clause composes of literals strictly combined by only $\wedge$ logical operator.

Every variable can only take a bipolar value which is 1 or 0 that exemplifies the idea of true and false. Another variant of SAT representation is 2 Satisfiability. 2 Satisfiability (2SAT) consist of set of clauses that contain strictly 2 literals. The general formula for 2SAT logic is as follows:

$$P_{2SAT} = \underset{i=1}{\overset{n}{\wedge}} l_i, \text{where } l_i = \underset{i=1}{\overset{k}{\vee}} C_i \underset{j=1}{\overset{n}{\vee}} D_j, \; k=2 \tag{1}$$

where l_i refers to the clauses of 2SAT, meanwhile C_i and D_i denote the literals, $\vee$ refers to Disjunction (OR), and $\wedge$ is an logical operator of Conjunction (AND).

The goal of 2SAT logic is to establish the ideal logical model of RBFNN to calculate the parameters of the hidden layer which contribute in deciding the number of hidden neurons in the hidden layer. Ideally, a combinatorial problem is similar to an ordinary mathematical model with quantifiable rate of change. Unfortunately, that statement does not hold if the specific combinatorial problem is dynamical and appeared as non-linear or linearly distributed. There were several efforts to represent the combinatorial problem via 2SAT formulation [42], [43]. These combinatorial problems contain implicit knowledge and could not be represented in standard rate of changes [44]. From that perspective, 2SAT is the main representation because this logical rule has a huge flexibility in terms of state (1 or 0) compared to standard mathematical representation.

III. Radial Basis Function Neural Network

Radial Basis Function Neural Network (RBFNN) is a variant of feed forward neural network with hidden interconnected layer which was pioneered by Lowe and Moody [45], [46]. Compared to other network, RBFNN has a more integrated structure and architecture. In terms of structure, RBFNN contains three neuron layers for computation purposes (See Fig. 1) [47]. In the input layer, m neurons represent the input data that was transferred to the system. During the training phase, the parameters (center and width) will be calculated in the hidden layer. The parameters obtained will be used to calculate the output weight in the output layer. To reduce the dimensionality from the input to the output layer, a Gaussian activation function has been introduced. The Gaussian activation function, $\varphi_i(x)$ of the hidden neuron in RBFNN is as follows [48], [49]:

$$Q(x) = \frac{\left\| \sum_{j=1}^{N} w'_{ji} x_j - c_j \right\|^2}{2\sigma_j^2} \tag{2}$$

$$\varphi_i(x) = e^{-Q(x)} \tag{3}$$

where c_j, σ_i are the center and width of the hidden neuron, respectively. In this case, x_j is a input value for N input neurons and the Euclidean norm $\| \; \|$ from neuron i to j can be defined as follows:

$$\left\| \sum_{j=1}^{N} w'_{ji} x_j - c_i \right\| = \sqrt{\sum_{m=1}^{N} \left(\sum_{j=1}^{N} w'_{ji} x_j - c_i \right)^2} \tag{4}$$

where w'_{ji} is the input weight between the input neuron j and the hidden neuron *i*. Structurally, x_j is a input data in the training set and the hidden neuron *i*. c_i is the center of the hidden neuron. The final output of RBFNN $F(w_i)$ is given by the following:

$$F(w_i) = \sum_{i=1}^{j} w_i \varphi_i(x_k) \tag{5}$$

where $F(w_i) = (F(w_1), F(w_2), F(w_3), \ldots\ldots, F(w_N))$ is the output value of RBFNN and the output weight is given by $w_i = (w_1, w_2, \ldots, w_N)$.

The aim of RBFNN is to obtain the optimal weights w_i that satisfy the desired output value. In RBFNN, the hidden neuron provides a set of function that represents input pattern spanned by the hidden neuron [4], [47].

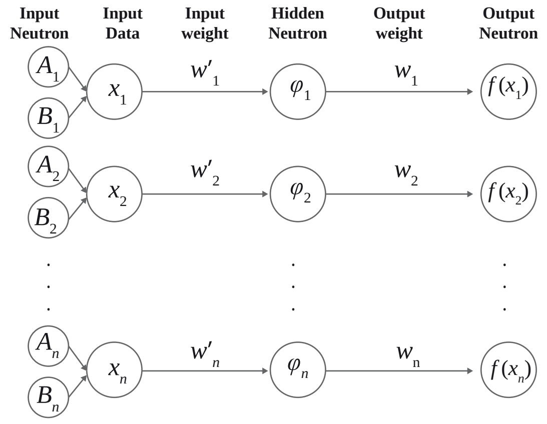

Fig. 1. Structure of RBFNN.

In this section, we will consider no training in conventional method Radial Basis Function Neural Network. Radial Basis Function Neural Network no-training paradigm was proposed by Vakil-Baghmisheh and Pavešić [50]. No training in Radial Basis Function is the simplest training because all the parameters were fixed. This method of training of RBFNN-2SAT does not have any practical value, because the number of prototype vectors should be equal to the number of input data, and consequently the network will be too complex. Fig. 2 shows the steps to integrate RBFNN no training with 2SAT, which can be abbreviated as RBFNN-2SATNT [9]:

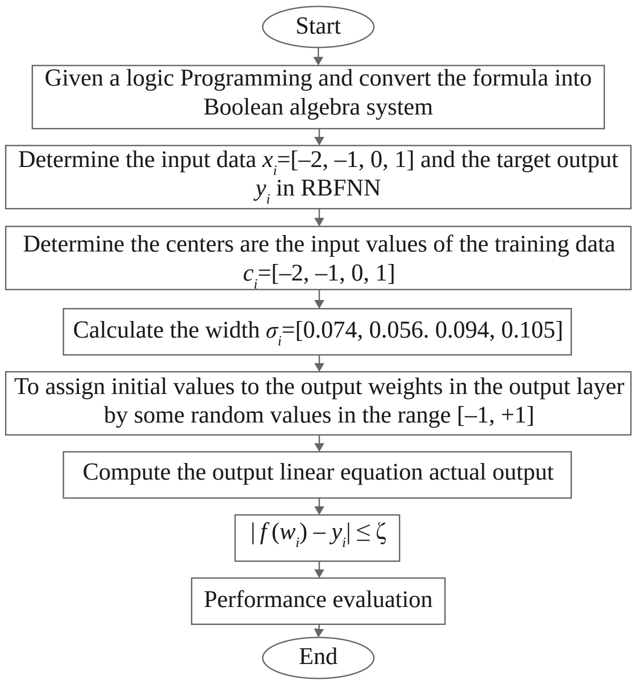
Start
Given a logic Programming and convert the formula into Boolean algebra system
Determine the input data x_i=[–2, –1, 0, 1] and the target output y_i in RBFNN
Determine the centers are the input values of the training data c_i=[–2, –1, 0, 1]
Calculate the width σ_i=[0.074, 0.056. 0.094, 0.105]
To assign initial values to the output weights in the output layer by some random values in the range [–1, +1]
Compute the output linear equation actual output
$|f(w_i) - y_i| \le \zeta$
Performance evaluation
End

Fig. 2. Flowchart of RBFNN-2SATNT.

The parameter x_i is the input data, whereas $c_i = x_i$ is the center, σ_i is the width, and ζ is the tolerance value.

IV. 2SAT Programming in RBFNN

Kasihmuddin *et al.* proposed logic programming by integrating 2SAT rule with neural network [14], [51]. The weight of the network was determined by Wan Abdullah method [2] where the inconsistencies of 2 Satisfiability logical rule have been minimized. The only problem of the proposed network is the rigidness of the weight calculation. 2SAT can be embedded to RBFNN by representing the variable as input neuron. Each input neuron x_j constitutes $\{0,1\}$ which signifies False and True. By using the value from input neuron, the parameters such as c_i and σ_i will be computed and the best number of hidden neuron will be obtained. In other words, embedding 2SAT as a logical rule makes RBFNN able to receive more input data with a fixed value of center and width. Hence the aim of the combination is to create a RBFNN model that classifies data based on 2SAT logical rule. Representation of 2SAT in RBFNN is given as the following formula:

$$P_{2SAT} = \overset{k}{\underset{i=1}{\vee}} C_i \overset{n}{\underset{j=1}{\vee}} D_j \tag{6}$$

where $k, n \in \mathbb{N}$. C_i and D_j are atoms. Applying embedding method of RBFNN, Eq. (6) will transform to:

$$x_i = \sum_{i=1}^{k} I(C_i) + \sum_{j=1}^{n} I(D_j) \tag{7}$$

$$I(C_i) or\, I(D_j) = \begin{cases} 1, & when\, C\, or\, D\, is\, True \\ 0, & when\, C\, or\, D\, is\, False \end{cases} \tag{8}$$

Eq. (7) and (8) are vital in calculating training data for each 2SAT clause. Hence the implementation of 2SAT in RBFNN is abbreviated as RBFNN-2SAT. Table I illustrates the input data of RBFNN-2SAT for:

$$P_{2SAT} = C, D \leftarrow, E \leftarrow F, K \leftarrow L \tag{9}$$

TABLE I. The Input Data and the Output Target data for $P_{2SAT} = C, D \leftarrow, E \leftarrow F, K \leftarrow L$

Clause	$C, D \leftarrow$			$E \leftarrow F$			$K \leftarrow L$		
DNF	$C \vee D$			$E \vee \neg F$			$K \vee \neg L$		
The Input Data Form	$x = C + D$			$x = E - F$			$x = K - L$		
Input Data in the Training Set x_i	0	1	2	-1	0	1	-1	0	1
The Target Output Data y_i	0	1	1	0	1	1	0	1	1

After finding the center and the width of the hidden layer, RBFNN will use the Gaussian function in Eq. (3) to calculate the output weight. As the number of clauses increase, RBFNN-2SAT requires more efficient learning method to find the correct output weight. In this paper, a metaheuristics algorithm will be implemented to find the optimal output weights that minimize the following objective function:

$$f(w_i) = \sum_{i=1}^{j} w_i \varphi_i(x) \tag{10}$$

where $f(w_i)$ is the final output classification of the RBFNN-2SAT.

V. Genetic Algorithm in RBFNN-2SAT

A Genetic Algorithm (GA) is a standard metaheuristic algorithm in solving various optimization problems. Given a finite solution space, the structure of a GA can be divided into local search and global search [52]. In a GA, the strings populations called chromosomes are represented in terms of solutions to the optimization problem [53]. The quality of the chromosome is denoted by the fitness value. At every generation, the fitness value of each chromosome is estimated, and the best fitness is selected as final solution. The chromosomes improve their fitness by implementing three (3) operators namely crossover, selection and mutation. Crossover promotes the exchange of information between chromosomes. Hamadneh *et al.* [4] used the GA to decide the centers of hidden neurons width and number of the hidden neuron by minimize the sum of absolute error of the actual outputs and the desired outputs. During selection, several chromosomes are selected from the current population depending on their fitness value. Mutation has been added to create genetic diversity of the chromosomes. In this paper, GA will be used to optimize the output weight of RBFNN-2SAT by reducing the training error. The implementation of GA in RBFNN is defined as RBFNN-2SATGA. In RBFNN-2SATGA, GA will calculate the output weight by using the centers, width in the hidden neuron. The steps involved in RBFNN-2SATGA are as follows:

Step 1

Population Initialization: The output weights represented by a chromosome will be initialized. The representations of chromosomes are as follows:

$$w_i = (w_1, w_2, w_3, \ldots, w_N) \tag{11}$$

The population has N_{pop} chromosomes containing N_N of random output weights. The aim is to minimize the objective function:

$$f_{GA}(w_i)=\begin{cases}1, & \sum_{i=1}^{j} w_i\varphi_i(x)\le 0\\ 0, & Otherwise\end{cases} \tag{12}$$

where $f_{GA}(w_i)$ is the objective function in the RBFNN-2SATGA model.

Step 2

Fitness Computation: The fitness of each individual chromosome is calculated via a basis function of RBFNN-2SAT. The basis function used in this paper is shown in the following equation:

$$fit_i=\frac{1}{\left(1+f_{GA}(w_i)\right)},\ 0\le fit_i\le 1 \tag{13}$$

where $f_{GA}(w_i)$ is the objective function and fit_i is the fitness value.

Step 3

Selection: The chromosomes are arranged in descending order based on the value of the fitness function. Only the best chromosomes (with the highest fitness value) are kept while others are discarded. The selection probability, p_i for each chromosome will be calculated by using the following equation:

$$p_i=\frac{fit_i}{\sum_{i=0}^{n} fit_i} \tag{14}$$

Step 4

Crossover: During the crossover phase, information from the parent will be randomly exchanged for creating offspring with different genetic composition. The location of the crossover will be randomly selected. Crossover phase will determine the number of cross-population according to the crossover rate. Given two parents w_k and w_m, the offspring w_i^{new} will be produced by the following equations [54], [55]:

$$w_i^{new}=\begin{cases}w_m+r(w_k-w_m), & p_i,\ i=1,2,3,\dots n\\ w_m, & 1-p_i\end{cases} \tag{15}$$

where p_i is the probability, r is the crossover rate, w_k is the chromosome with higher probability, w_m is the chromosome with lower probability and the parameter k is choosen by the following equation:

$$k=\begin{cases}rand(m,n), & p_i\\ m, & 1-p_i\end{cases} \tag{16}$$

where $k + m = n$ and $k > m$. The value of k is uniformly distributed between k and m.

Step 5

Mutation: During the mutation phase, the chromosome information will be randomly assigned within the pre-determined range (often determined by the user). The mutation is expected to create a newly breed of chromosome. The equation involved is as follows:

$$w_m^{new}=\begin{cases}rand(-5,5), & rand(0,1)<\tau\\ w_i, & rand(0,1)\ge\tau\end{cases} \tag{17}$$

where w_m^{new} is the new chromosome from mutation phase when $\tau\in[0,1]$.

Step 6

Termination: GA will iterate up to 10000 Generations. If a given solution termination criterion is met, the calculation of the algorithm is stopped or will go back to step 2 with $i=i+1$. The final output of RBFNN-2SAT is a chromosome that contains optimal output weights of RBFNN-2SATGA.

VI. Differential Evolution Algorithm in RBFNN-2SAT

Storn and Price [22] has fruitfully introduced a new evolutionary population-based algorithm called the Differential Evolutionary (DE) algorithm which typically is being used in numerical optimization. The fundamental framework of DE algorithm can be divided into local and global search with an adaptable function optimizer [56]. The core differences between GA and DE is that the selection operator in DE uses an equal probability to elect parents. Hence, the chance is independent towards the fitness value of the solutions. In the DE algorithm, every individual solution competes with its parent and the fittest one will win [57]. In this work, the DE algorithm will be adopted as a learning mechanism during the training phase. The purpose of the training is to compute the corresponding output weights that connect hidden neurons and output neurons of RBFNN-2SAT. The stages involved in RBFNN-2SATDE in optimizing the connection weights between the hidden layer and the output layer is represented in Fig. 3.

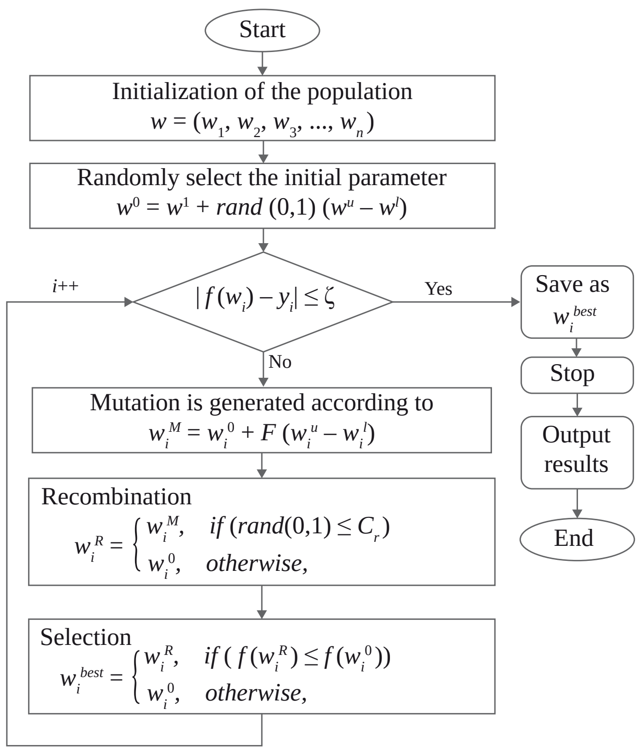
Start
Initialization of the population
$w=(w_1, w_2, w_3, \dots, w_n)$
Randomly select the initial parameter
$w^0=w^1+rand(0,1)(w^u-w^l)$
$|f(w_i)-y_i|\le\zeta$
i++
Yes
No
Save as w_i^{best}
Stop
Output results
End
Mutation is generated according to
$w_i^M=w_i^0+F(w_i^u-w_i^l)$
Recombination
$w_i^R=\begin{cases}w_i^M, & if\ (rand(0,1)\le C_r)\\ w_i^0, & otherwise,\end{cases}$
Selection
$w_i^{best}=\begin{cases}w_i^R, & if\ (f(w_i^R)\le f(w_i^0))\\ w_i^0, & otherwise,\end{cases}$

Fig. 3. Flowchart of RBFNN-2SATDE.

The real parameters w^l and w^u are lower and upper bounds, respectively. w^0 is the initial parameter value distributed uniformly on the intervals $[w_i^l, w_i^u]$. w_i^M is the mutation output weight, $F\in[0,2]$ is the mutation factor. w_i^R is the recombination output weight, $C_r\in[0,1]$ is the crossover probalitiy. ζ is the tolerance value.

VII. Particle Swarm Optimization Algorithm in RBFNN-2SAT

The PSO algorithm is a class of iterative swarm-based searching algorithm, deployed widely as the learning algorithm or universal optimization. The pioneer work of PSO was coined by Eberhart and Kennedy [26] by mathematically modelling the socio-behavioral feature of the bird flocking and fish schooling in their own population. The remarkable feature in PSO is the existence of adjustable free parameters, which makes it easy to implement and optimize. Specifically, PSO adopted a vigorous searching process by impending the best particle in a solution space [58]. Pursuing that, the potential solutions, named particles, fly over the searching space by succeeding the existing optimum particles. In addition, the changes in the position of the particles occur in PSO, where it is vital in searching for the best particle. This study adopts the PSO algorithm to optimize the output weight among the hidden neurons and the output neurons of RBFNN-2SAT. Therefore, the steps involved in RBFNN-2SATPSO are represented in Fig. 4.

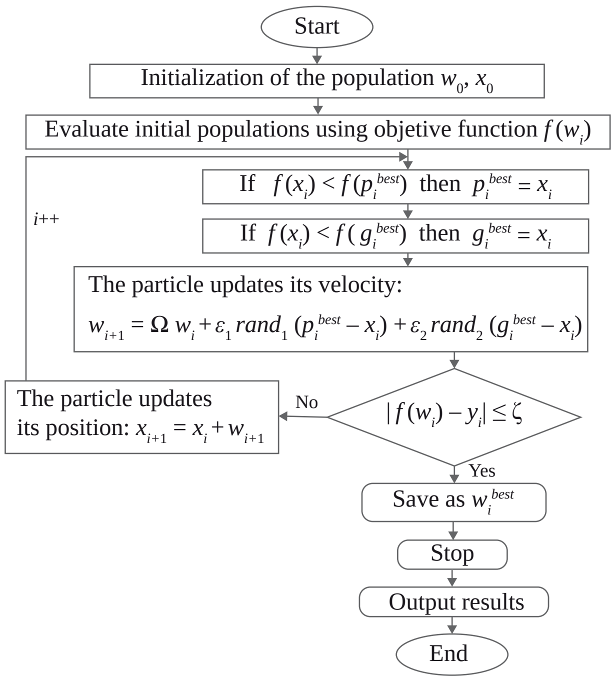
Start
Initialization of the population w_0, x_0
Evaluate initial populations using objetive function $f(w_i)$
If $f(x_i) < f(p_i^{best})$ then $p_i^{best} = x_i$
If $f(x_i) < f(g_i^{best})$ then $g_i^{best} = x_i$
i++
The particle updates its velocity:
$w_{i+1} = \Omega\, w_i + \varepsilon_1 rand_1\,(p_i^{best} - x_i) + \varepsilon_2 rand_2\,(g_i^{best} - x_i)$
The particle updates its position: $x_{i+1} = x_i + w_{i+1}$
No
$|f(w_i) - y_i| \le \zeta$
Yes
Save as w_i^{best}
Stop
Output results
End

Fig. 4. Flowchart of RBFNN-2SATPSO.

The parameter Ω is the inertia weight, whereas $\varepsilon_1 = \varepsilon_2 = 2$ are acceleration constants, $rand_1 = rand_2$ are experimented arbitrarily within [0, 1], p_i^{best} refers to the individual best position attained by the particle of the primary swarm, and g_i^{best} denotes the global best position completed by the particles of the sucessive swarm and the position of the new particle, x_i. Additionally, ζ is the tolerance value.

VIII. Artificial Bee Colony Algorithm in RBFNN-2SAT

Artificial bee colony (ABC) algorithm has been introduced by Karaboga [59] in resolving various mathematical optimization problems. In ABC, the colony of bees contains three groups called employed bees, onlooker bees, and scout bees. Generally, employed bees bring quantities of nectar from the resource food to the hive. They will share the information about the source of food with a certain probability by dancing inside the hive. Then, onlooker bees stay in the dancing areas and decide source of food depending on the prospect (the probability) provided by the employed bees [32]. The other type of bees is called the Scout Bee, which conducts the random search for new sources of food if the quality of the food source is not in a satisfactory state. In this paper, ABC will be used to optimize the output weight of RBFNN-2SAT by reducing the training error. The implementation of ABC in RBFNN is defined as RBFNN-2SATABC. In this context, the function to be optimized is:

$$f_{ABC}(w_i) = \begin{cases} 1, & \sum_{i=1}^{j} w_i \varphi_i(x) \le 0 \\ 0, & Otherwise \end{cases} \tag{18}$$

where $f_{ABC}(w_i)$ is the objective function of the RBFNN-2SATABC model. The algorithm involved in RBFNN-2SATABC is as follows:

Step 1

Population Initialization: Initialize all the bee that is:

$$w_{ji} = (w_{1,i}, w_{2,i}, \ldots, w_{ji}, \ldots, w_{di}) \tag{19}$$

in RBFNN-2SAT as:

$$w_{ji} = w_{j\min} + rand[0,1]\,(w_{j\max} - w_{j\min}) \tag{20}$$

where $w_{ji} \in [w_{j\min}, w_{j\max}]$, $w_{j\min}$ and $w_{j\max}$ are the minimum value and maximum value of the output weight with index of $i \in \{1,2,\ldots,n\}$ and $j \in \{1,2,\ldots,d\}$. n is the number of employed bees (the number of solutions), and d is the dimension of the solution space (number of hidden neurons).

Step 2

Employed Bee Phase: Employed bee will search for the food source. The new food source (solution) for employed bees, $w_{ji}^{employed}$ is given as follows:

$$w_{ji}^{employed} = w_{ji} + rand[0,1]\left(w_{ji} - w_{jk}\right) \tag{21}$$

where j, k are selected randomly and the w_{jk} is called the neighbor bee of w_{ji}. The value of $f_{ABC}\left(w_{ji}^{employed}\right)$ will be calculated as follows:

$$f_{ABC}\left(w_{ji}^{employed}\right) = \sum_{i=1}^{j} w_{ji}^{employed}\varphi_i(x) \tag{22}$$

$$fit_i = \frac{1}{1 + f_{ABC}\left(w_{ji}^{employed}\right)} \tag{23}$$

where fit_i is the fitness value of the bee.

Step 3

Onlooker Bee Phase: The probability value of the food sources will be calculated. Onlooker bee will perform exhange of information based on the following probability:

$$p_i^{\text{Onlooker}}{}_i = \frac{fit\left(w_i^{employed}\right)}{\sum_{i=1}^{SN} fit\left(w_i^{employed}\right)} \tag{24}$$

By using the above probability, the food source will be obtained by using equation (21).

Step 4

Scout Bee Phase: If the values of fitness of the employed bees are not improving by a number continuous predetermined of iterations, which is called (*Limit*) those food source are abandoned, and these employed bee become the scouts, and generate a new solution w_i^{new} for the employed bee by using the following equation:

$$w_i^{new} = \begin{cases} rand\,(-5,5), & limit > trial \\ w_i, & Otherwise \end{cases} \tag{25}$$

Step 5

Termination: If the stopping criterion is met, then it stops and the best food source is memorized, otherwise, the algorithm returns to Step 2.

IX. Experimental Setup

All the proposed RBFNN-2SAT model will be executed and coded in Microsoft Visual C # 2008 Express program in Microsoft Window 7, 64-bit, with 500 GB hard drive specification, 4096 MB RAM, and 3.40 GHz processor. The lists of parameters used in each RBFNN-2SAT model are summarized in Table II to Table V. Simulated data sets will be obtained by randomly generate the input data. The choice of data reduces the possible bias of the data which covers a wider range of search space. Next, the number of neurons NN used in the experiment varies from $6 \leq NN \leq 108$.

TABLE II. List of Parameters in RBFNN-2SATGA

Parameter	Value
Number of iteration	10000
Selection type	Wheel selection
Number of individuals	50
Mutation ratio	1
Mutation type	Uniform
Crossover ratio	1
Crossover type	Single point

TABLE III. List of Parameters in RBFNN-2SATDE

Parameter	Value
Number of iteration	10000
C_r	[0, 1]
F	[0, 2]
Population	50

TABLE IV. List of Parameters in RBFNN-2SATPSO

Parameter	Value
Ω	0.6
ε_1	2
ε_2	2
$rand_1 = rand_2$	[0,1]
Number of iteration	10000

TABLE V. List of Parameters in RBFNN-2SATABC

Parameter	Value
No_Employed_bees	50
No_Onlooker_bees	50
No_Scout_bees	1
Limit	1000
Trial	10000

X. Results and Discussion

Hamadneh *et al.* [60] use mean square error as a metric to appraise the performance of the trained RBFNN. In this paper, both proposed hybrid models will be compared by using four performance metrics such as Root Mean Square Error (RMSE), Sum of Squares Error (SSE), Mean Absolute Percentage Error (MAPE) and CPU Time. The equation for each performance metrics is as follows:

$$RMSE = \sum_{i=1}^{n} \sqrt{\frac{1}{n}\left(f(w_i) - y_i\right)^2} \tag{26}$$

$$SSE = \sum_{i=1}^{n} \left(f(w_i) - y_i\right)^2 \tag{27}$$

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{\left(f(w_i) - y_i\right)}{y_i} \right| \tag{28}$$

where $f(w_i)$ is the actual output value, y_i is the target output value and n is number of the iterations. In addition, computation time will be considered in order to evaluate the efficiency of the RBFNN model.

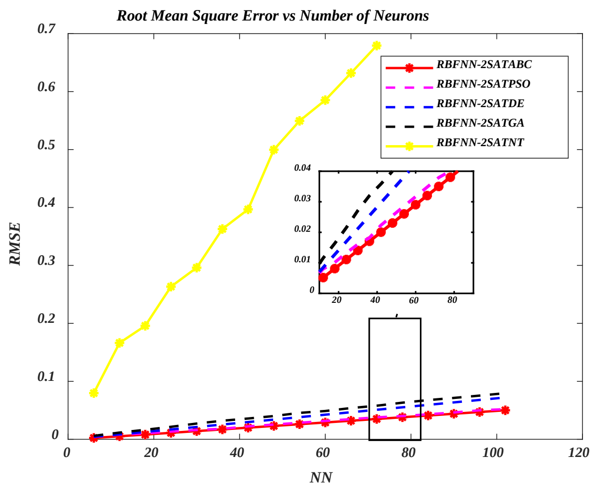

Fig. 5. RMSE value for all RBFNN-2SAT models.

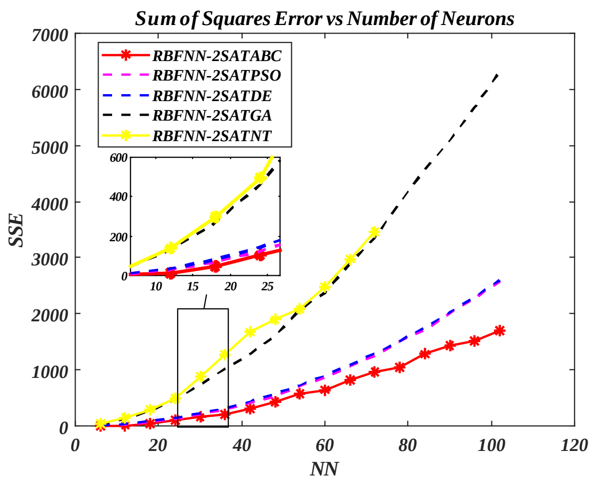

Fig. 6. SSE evaluation.

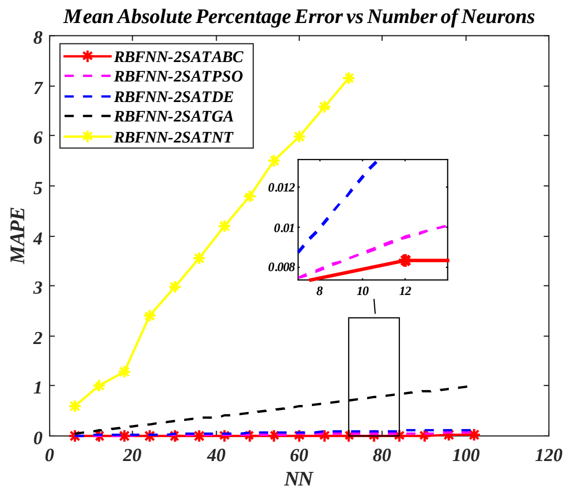

Fig. 7. MAPE evaluation.

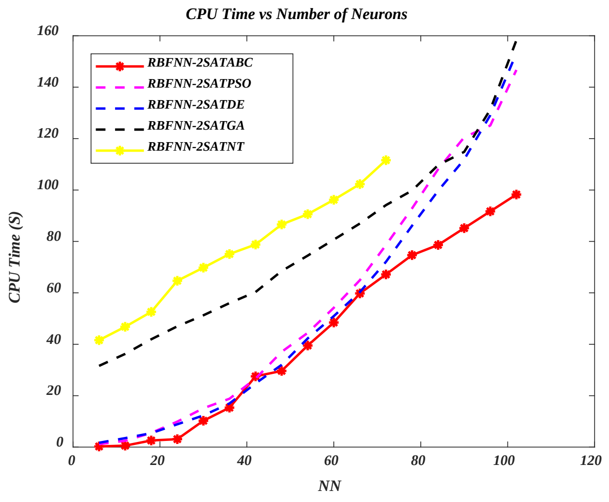

Fig. 8. Computation time evaluation.

In this study, 2SAT logical rule is expected to perform comparatively exceptional to other non-systematic logical rule such as [6], [29], [61], [62], [63]. This is due to the variation of the number of variables in each clause. This causes RBFNN-2SAT to alter the dimension of the hidden layer. Imbalance signal from the hidden layer to the output layer will lead to imbalance value of parameters (centre and width) and high computation error. The results in Fig. 5 until Fig. 8 allow to deduce the following findings:

1. RBFNN-2SAT can receive more input data with a fixed value of center and width. In this case, RBFNN-2SATABC creates a model that classifies data based on 2SAT logical rule with minimum value of RMSE, SSE and MAPE.
2. RBFNN-2SATABC has best performances in terms of errors as the number of neurons is increased. In the exploration front (employed bee), ABC locates the general range of the optimal output weight. The value of the output weight improves significantly during the exploitation phase (onlooker bees). Based on the result, the probability for RBFNN-2SATABC to reach the scout bee phase is approximately zero. In this case, RBFNN-2SATABC effectively explores different solution space in less iterations.
3. In terms of computation time, RBFNN-2SATABC was reported to be faster than the other RBFNN-2SAT model. At $NN > 20$, the possibility for the conventional method RBFNN-2SATNT to be trapped in trial and error state increases. Trial and error cause RBFNN-2SATNT to achieve pre-mature convergence.
4. On the other hand, RBFNN-2SATGA has a relatively larger learning error because of ineffective initial crossover. It requires several iterations for RBFNN-2SATGA to produce high quality output weight. During that time, the only operator that is effective is mutation. The problem is worsened when the suboptimal output weight is a floating number.
5. RBFNN-2SATDE is reported to illustrate some drawbacks such as tendency to be trapped at sub-optimal output weight and slow convergence rate. In this case, RBFNN-2SATDE requires more iterations to satisfy $\left|f(w_i)-y_i\right| \leq \zeta$ which results in the accumulation of error. In addition, the unbounded mutation operator in DE tends to create numerous alternate search space that reduces the probability of the RBFNN-2SATDE to achieve optimal output weight.
6. In another perspective, RBFNN-2SATPSO has a relatively lower learning error compared to another model. This is due to the use of the particle in this algorithm that mimics our proposed ABC algorithm. Although the result for RBFNN-2SATPSO seems quite promising, this algorithm lacks the control of the effective local search. In this case, as $t \rightarrow 10000$, the search space for each particle will magnify indefinitely and result in suboptimal output weight. Hence, RBFNN-2SATPSO will converge prematurely.

These experiments show that the ABC algorithm can be successfully applied to train RBFNN-2SAT. Another observation is that the effectiveness of ABC can be seen vividly when the number of neurons increases. Moreover, ABC algorithm in RBFNN achieves more promising performance based on RSME by 94.8%, SSE by 72.9%, MAPE by 99.1%, and CPU time by 39.8%. This concludes that ABC in RBFNN-2SAT could be used in practice to achieve better prediction results for the 2SAT logic programming.

XI. Conclusion

A hybrid paradigm, ABC algorithm incorporated with RBFNN and 2SAT (RBFNN-2SATABC) has been fruitfully developed to foster the learning phase with different number of neurons. Following that, the work as reported in this paper reveals the significant differences in the performance of RBFNN-2SATABC in terms of Root Mean Square Error (RMSE), Sum of Squares Error (SSE), Mean Absolute Percentage Error (MAPE), and process time (computation time in seconds). Furthermore, the proposed paradigm offers an error of approximately 2% of MAPE, and faster computation time compared to RBFNN-2SATGA. Henceforth, the RBFNN-2SATABC has been clearly recognized to be more robust than the RBFNN-2SATGA in certain aspects which include better lower error and faster process time in performing 2SAT logic programming. As future development, the RBFNN-2SATABC can be improved by using different classes of Satisfiability logic ranging from, Major Satisfiability (MAJ-SAT), Weighted SAT, Maximum Satisfiability (MAX-SAT) and Unsatisfiable Satisfiability (MIN-UNSAT). This work also can be applied as a traditional optimization method to solve problems such as travelling salesman and N-queen's problem.

Acknowledgment

This research was supported by Fundamental Research Grant Scheme (FRGS), Ministry of Education Malaysia, grant number 203/PMATH/6711804 and Universiti Sains Malaysia (USM).

References

[1] M. S. Alkhaawneh, (2019). Hybrid Cascade Forward Neural Network with Elman Neural Network for Disease Prediction. *Arabian Journal for Science and Engineering*, 44(11), 9209-9220.

[2] W. A. T. W. Abdullah, (1992). Logic programming on a neural network. *International journal of intelligent systems*, 7(6), 513-519.

[3] S. Sathasivam, (2010). Upgrading logic programming in Hopfield network. *Sains Malaysiana*, 39(1), 115-118.

[4] N. Hamadneh, , S. Sathasivam, S. L. Tilahun, & O. H. Choon, (2012). Learning logic programming in radial basis function network via genetic algorithm. *Journal of Applied Sciences(Faisalabad),* 12(9): 840-847.

[5] M. S. M. Kasihmuddin, M. A. Mansor, & S. Sathasivam, (2017). Hybrid Genetic Algorithm in the Hopfield Network for Logic Satisfiability Problem. *Pertanika Journal of Science & Technology*, 25(1), 139 - 152.

[6] M.A.B. Mansor, M.S.B.M. Kasihmuddin, and S. Sathasivam, 2017. Robust Artificial Immune System in the Hopfield network for Maximum k-Satisfiability. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(4), 63-71.

[7] N. Hamadneh, W. A. Khan, I. Khan, & A. S. Alsagri, (2019). Modeling and Optimization of Gaseous Thermal Slip Flow in Rectangular Microducts Using a Particle Swarm Optimization Algorithm. *Symmetry*, 11(4), 488-491.

[8] H. de Leon-Delgado, R. J. Praga-Alejo, D. S. Gonzalez-Gonzalez, & M. Cantú-Sifuentes, (2018). Multivariate statistical inference in a radial basis function neural network. *Expert Systems with Applications*, 93, 313-321.

[9] S. Alzaeemi, M.A. Mansor, M.S.M. Kasihmuddin, S. Sathasivam, and M. Mamat, (2020). Radial basis function neural network for 2 satisfiability programming. *Indonesian Journal of Electrical Engineering and Computer Science*, 18(1), 459-469.

[10] M. H. Horng, Y. X. Lee, M. C. Lee, & R. J. Liou, (2012). Firefly meta-heuristic algorithm for training the radial basis function network for data classification and disease diagnosis. In Theory and new applications of swarm intelligence. *IntechOpen*, 10(19), 7-28.

[11] J. Yang, & J. Ma, (2019). Feed-forward neural network training using sparse representation. *Expert Systems with Applications*, 116, 255-264.

[12] M. S. M. Kasihmuddin, M. A. Mansor, M. B. M. Faisal, & S. Sathasivam, (2019). Discrete Mutation Hopfield Neural Network in Propositional Satisfiability. *Mathematics*, 7(11), 1133-1154.

[13] M.S.M., Kasihmuddin, Mansor, M.A. and Sathasivam, S., 2018. Discrete Hopfield Neural Network in Restricted Maximum k-Satisfiability Logic Programming. *Sains Malaysiana*, 47(6), 1327-1335.

[14] M. S. M. Kasihmuddin, M. A. Mansor, & S. Sathasivam, (2017). Robust Artificial Bee Colony in the Hopfield Network for 2-Satisfiability Problem. *Pertanika Journal of Science & Technology*, 25(2), 453 - 468.

[15] N. Hamadneh, S. Sathasivam, and O.H. Choon, (2012). Higher order logic programming in radial basis function neural network. *Appl Math Sci*, 6(3), 115-127.

[16] H. V. H. Ayala, & L.dos Santos Coelho, (2016). Cascaded evolutionary algorithm for nonlinear system identification based on correlation functions and radial basis functions neural networks. *Mechanical Systems and Signal Processing,* 1(68), 378-393.

[17] R. D. Dandagwhal, & V. D. Kalyankar, (2019). Design Optimization of Rolling Element Bearings Using Advanced Optimization Technique. *Arabian Journal for Science and Engineering*, 44(9), 7407-7422.

[18] J. H. Holland, (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2), 88-105.

[19] D. E. Goldberg, & J. H. Holland, (1988). Genetic algorithms and machine learning, *Machine Learning*, 2(3), 95-99.

[20] M. Pandey, M. Zakwan, P. K. Sharma, & Z. Ahmad, (2020). Multiple linear regression and genetic algorithm approaches to predict temporal scour depth near circular pier in non-cohesive sediment. *ISH Journal of Hydraulic Engineering*, 26(1), 96-103.

[21] Z. Jing, J. Chen, & X. Li, (2019). RBF-GA: An adaptive radial basis function metamodeling with genetic algorithm for structural reliability analysis. *Reliability Engineering & System Safety*, 189, 42-57.

[22] R. Storn and K. Price, (1997). Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4), 341-359.

[23] A. Saha, A. Konar, P. Rakshit, A. L. Ralescu, & A. K. Nagar, (2013, August). Olfaction recognition by EEG analysis using differential evolution induced Hopfield neural net. *In The 2013 International Joint Conference on Neural Networks,* 4(9), 1-8.

[24] J. Ilonen, J. K. Kamarainen, & J. Lampinen, (2003). Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1), 93-105.

[25] W. Tao, J. Chen, Y. Gui, & P. Kong, (2019). Coking energy consumption radial basis function prediction model improved by differential evolution algorithm. *Measurement and Control*, 52(8), 1122-1130.

[26] R. Eberhart, & J. Kennedy, (1995, October). A new optimizer using particle swarm theory. *In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science,* 39-43.

[27] J. Kennedy, & R. Eberhart, (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-*International Conference on Neural Networks,* vol. 4, 1942-1948.

[28] S. N. Qasem, & S. M. H. Shamsuddin, (2009, May). Improving performance of radial basis function network based with particle swarm optimization. *In 2009 IEEE Congress on Evolutionary Computation*, *Man and Cybernetics*, 3149-3156.

[29] A. Alexandridis, E. Chondrodima, & H. Sarimveis, (2016). Cooperative learning for radial basis function networks using particle swarm optimization. *Applied Soft Computing,* 49, 485-497.

[30] D. Karaboga, & B. Basturk, (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization*, 39(3), 459-471.

[31] D. Karaboga, & E. Kaya, (2019). Training ANFIS by using an adaptive and hybrid artificial bee colony algorithm (aABC) for the identification of nonlinear static systems. *Arabian Journal for Science and Engineering,* 44(4), 3531-3547.

[32] G.E. Tsekouras, V. Trygonis, A. Maniatopoulos, A. Rigos, A. Chatzipavlis, J. Tsimikas, N. Mitianoudis, and A.F. Velegrakis, (2018). A Hermite neural network incorporating artificial bee colony optimization to model shoreline realignment at a reef-fronted beach. *Neurocomputing*, 280, 32-45.

[33] T. Kurban, & E. Beşdok, (2009). A comparison of RBF neural network training algorithms for inertial sensor based terrain classification. *Sensors*, 9(8), 6312-6329.

[34] J. Yu, & H. Duan, (2013). Artificial bee colony approach to information granulation-based fuzzy radial basis function neural networks for image fusion. *Optik-International Journal for Light and Electron Optics*, 124(17), 3103-3111.

[35] B. Jafrasteh, & N. Fathianpour, (2017). A hybrid simultaneous perturbation artificial bee colony and back-propagation algorithm for training a local linear radial basis neural network on ore grade estimation. *Neurocomputing*, 235, 217-227.

[36] S. K. Satapathy, S. Dehuri, & A. K. Jagadev, (2017). ABC optimized RBF network for classification of EEG signal for epileptic seizure identification. *Egyptian Informatics Journal,* 18(1), 55-66.

[37] I. Aljarah, H. Faris, S. Mirjalili, & N. Al-Madi (2018). Training radial basis function networks using biogeography-based optimizer. *Neural Computing and Applications*, 29(7), 529-553.

[38] N. A. Menad, A. Hemmati-Sarapardeh, A. Varamesh, & S. Shamshirband, (2019). Predicting solubility of CO2 in brine by advanced machine learning systems: Application to carbon capture and sequestration. *Journal of CO2 Utilization*, 33, 83-95.

[39] S. Jiang, C. Lu, S. Zhang, X. Lu, S. B. Tsai, C. K. Wang, & C. H. Lee, (2019). Prediction of Ecological Pressure on Resource-Based Cities Based on an RBF Neural Network Optimized by an Improved *ABC Algorithm. IEEE Access*, 7, 47423-47436.

[40] S. Sathasivam, (2010). Upgrading logic programming in Hopfield network. *Sains Malaysiana*, 39(1), 115-118.

[41] T. Hoeink, (2019). Boolean satisfiability problem for discrete fracture network connectivity. *Patent Application Publication*, 180(53), 1-12.

[42] S. Mukherjee, & S. Roy, (2015). Multi terminal net routing for island style FPGAs using nearly-2-SAT computation. In VLSI Design and Test (VDAT), 2015 *19th International Symposium on IEEE*, 10(1109), 1-6.

[43] R. Miyashiro, & T. Matsui, (2005). A polynomial-time algorithm to find an equitable home away assignment. *Operations Research Letters*, 33(3), 235-241.

[44] S. Even, A. Itai, & A. Shamir, (1976). On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM Journal on Computing*, 5(4), 691-703.

[45] J. Moody, & C. J. Darken, (1989). Fast learning in networks of locally-tuned processing units. *Neural computation*, 1(2), 281-294.

[46] D. Lowe, (1989, October). Adaptive radial basis function nonlinearities,

and the problem of generalisation. *In Artificial Neural Networks, First IEE International Conference,* 1(313), 171-175.

[47] A. K. Hassan, M. Moinuddin, U. M. Al-Saggaf, & M. S. Shaikh, (2018). On the kernel optimization of radial basis function using nelder mead simplex. *Arabian Journal for Science and Engineering*, 43(6), 2805-2816.

[48] A. Idri, A. Zakrani, & A. Zahi, (2010). Design of radial basis function neural networks for software effort estimation. *IJCSI International Journal of Computer Science Issues*, 7(4), 11-17.

[49] S. B. Roh, S. K. Oh, W. Pedrycz, K. Seo, & Z. Fu, (2019). Design methodology for Radial Basis Function Neural Networks classifier based on locally linear reconstruction and Conditional Fuzzy C-Means clustering. *International Journal of Approximate Reasoning*, 106, 228-243.

[50] M. T. Vakil-Baghmisheh, & N. Pavešić, (2004). Training RBF networks with selective backpropagation. *Neurocomputing*, 62, 39-64.

[51] L.C. Kho, M. S. M. Kasihmuddin, M. A. Mansor, & S. Sathasivam, (2020). Logic Mining in League of Legends. *Pertanika Journal of Science & Technology*, 28(1), 211 - 225.

[52] W. Jia, D. Zhao, T. Shen, C. Su, C. Hu, & Y. Zhao, (2014). A new optimized GA-RBF neural network algorithm. *Computational intelligence and neuroscience*, 1(4), 1-6.

[53] H. Marouani, K. Hergli, H. Dhahri, & Y. Fouad, (2019). Implementation and Identification of Preisach Parameters: Comparison Between Genetic Algorithm, Particle Swarm Optimization, and Levenberg–Marquardt Algorithm. *Arabian Journal for Science and Engineering*, 44(8), 6941-6949.

[54] M. Awad, (2010). Optimization RBFNNs parameters using genetic algorithms: applied on function approximation. *International Journal of Computer Science and Security (IJCSS),* 4(3), 295-307.

[55] L. J. Eshelman, & J. D. Schaffer, (1993). Real-coded genetic algorithms and interval-schemata. *In Foundations of genetic algorithms*, Vol. 2. Elsevier, 187-202.

[56] S. L. Wang, F., Ng, T. F. Morsidi, H. Budiman, & S. C. Neoh, (2020). Insights into the effects of control parameters and mutation strategy on self-adaptive ensemble-based differential evolution. *Information Sciences*, 514, 203-233.

[57] K. R. Opara, & J. Arabas, (2019). Differential Evolution: A survey of theoretical analyses. *Swarm and evolutionary computation*, 44, 546-558.

[58] Y. Fukuyama, & H. Yoshida, (2001, May). A particle swarm optimization for reactive power and voltage control in electric power systems. *In Proceedings of the 2001 Congress on Evolutionary Computation* (IEEE Cat. No. 01TH8546), vol. 1, 87-93.

[59] D. Karaboga, (2005). An idea based on honey bee swarm for numerical optimization. *Technical reporttr 06, Erciyes university, engineering faculty, computer engineering department*, Vol. 200, 1-10.

[60] N. Hamadneh, S. Sathasivam, S. L. Tilahun, & O. H. Choon, (2014, July). Satisfiability of logic programming based on radial basis function neural networks. *In AIP Conference Proceedings,* 1605(1), 547-550.

[61] M. S. M. Kasihmuddin, M. A. Mansor, & S. Sathasivam, (2016). Genetic Algorithm for Restricted Maximum k-Satisfiability in the Hopfield Network. *International Journal of Interactive Multimedia & Artificial Intelligence*, 4(2), 52-60.

[62] M. S. M. Kasihmuddin, M. A. Mansor, & S. Sathasivam, (2016). Artificial Bee Colony in the Hopfield Network for Maximum k-Satisfiability Problem. *Journal of Informatics and Mathematical Sciences*, 8(5), 317-334.

[63] C. Caleiro, F. Casal, & A. Mordido, (2019). Generalized probabilistic satisfiability and applications to modelling attackers with side-channel capabilities, *Theoretical Computer Science*, 781, 39-62.

Mohd Shareduwan Mohd Kasihmuddin

Mohd Shareduwan Mohd Kasihmuddin is a lecturer in School of Mathematical Sciences, Universiti Sains Malaysia. He received his Ph.D from Universiti Sains Malaysia. His current research interests include Metaheuristics method, neural network development, artificial intelligence and logic programming. He can be contacted via shareduwan@usm.my.

Mohd. Asyraf Mansor

Mohd. Asyraf Mansor is a lecturer in School of Distance Education, Universiti Sains Malaysia. He received his Ph.D from Universiti Sains Malaysia. His current research interests include evolutionary algorithm, satisfiability problem, neural networks, logic programming and heuristic method.

Shehab Abdulhabib Alzaeemi

Shehab Abdulhabib Alzaeemi received a Bachelor Degree of Education (Science) from Taiz Universiti in 2004, Master of Science (Mathematics) from Universiti Sains Malaysia in 2016 and an ongoing PhD student in Universiti Sains Malaysia. He was a fellow under the Academic Staff Training System of Sana'a Community College from 2005-2014. His research interests mainly focus on neural network, logic programming, and data mining. His email is shehab_alzaeemi@yahoo.com

Saratha Sathasivam

Saratha Sathasivam is an Associate Professor in the School of Mathematical Sciences, Universiti Sains Malaysia. She received her MSc and BSc(Ed) from Universiti Sains Malaysia. She received her Ph.D at Universiti Malaya, Malaysia. Her current research interest are neural networks, agent based modeling and constrained optimization problem. Her email is saratha@usm.my.