Malaysian Journal of Mathematical Sciences 16(1): 37–53 (2022)
https://doi.org/10.47836/mjms.16.1.04

Malaysian Journal of Mathematical Sciences

Journal homepage: https://einspem.upm.edu.my/journal

Propositional Satisfiability Logic via Ant Colony Optimization in Hopfield Neural Network

Kho, L. C.[1], Kasihmuddin, M. S. M.*[1], Mansor, M. A.[2], and Sathasivam, S.[1]

[1]*School of Mathematical Sciences, Universiti Sains Malaysia, Malaysia*
[2]*School of Distance Education, Universiti Sains Malaysia, Malaysia*

E-mail: shareduwan@usm.my
**Corresponding author*

Received: 9 April 2020
Accepted: 2 November 2021

Abstract

Minimizing the cost function that corresponds to propositional logic is vital to ensure the learning phase of HNN can occur optimally. In that regard, optimal and non-biased algorithm is required to ensure HNN will always converge to global solution. Ant Colony Optimization (ACO) is a population-based and nature-inspired algorithm to solve various combinatorial optimization problems. ACO simulates the behaviour of the real ants that forage for food and communication of ants through pheromone density. In this work, ACO will be used to minimize the cost function that corresponds to the logical rule in Hopfield Neural Network. ACO will utilize pheromone density to find the optimal path that leads to zero cost function without consuming more learning iteration. Performance for all learning models will be evaluated based on various performance metrics. Results collected from computer simulation implies that ACO outperformed conventional learning model in minimizing the logical cost function.

Keywords: 2 satisfiability logic; ant colony optimization; propositional logic; hopfield neural network.

1 Introduction

Hopfield Neural Network (HNN) [8] is a single-layered neuron that is connected by synaptic weight. Each of the neurons in HNN will fire if the product of the synaptic weight and neuron state exceeded the threshold value. The main contribution of HNN [8] in this work serves as the best variant of recurrent neural network in logic programming and combinatorial problems. Despite the success of HNN in solving various combinatorial problems, HNN suffers several disadvantages such as capacity issues. As the number of neurons increased, the capacity of the associative memory decreases exponentially. This problem eventually led to the idea of assigning neurons in HNN with a set of logical rules. Abdullah [1] proposed logic programming in HNN by comparing the cost function with the final energy function. The work of [1] has been the core of the optimal synaptic weight computation which lead to a more feasible way of propositional logic programming in HNN. The optimal synaptic weight calculation proposed by Abdullah [1] also coined as Wan Abdullah method has contributed to an effective synaptic weight calculations by comparing the cost function of the propositional logic and the energy function in HNN. In this work, Wan Abdullah method will be integrated with a metaheuristic algorithm in optimizing the learning phase in HNN.

The proposed work has been further improved by Sathasivam [9] where a systematic form of propositional logic called 2 Satisfiability has been implemented in HNN. The proposed HNN managed to achieve an acceptable global minimum solution during the retrieval phase. This work is a starting point to the implementation of the systematic logical rule in HNN. The proposed method is managed to achieve the final neuron state that corresponds to the global minimum solution. The proposed hybrid HNN integrated with various activation function reduce the possible local minimum energy. Immediately afterwards, Alway et al. [2] proposed the first logical rule that considers the major proportion of the second-order logic with respect to other orders of logical rule. By increasing the number of second-order logic with respect to other logical rules, the proposed HNN obtained is reported to obtain more solution variation than the existing work. Unfortunately, all the mentioned study has a severe limitation during the learning phase of HNN. As the number of neurons increased, the storage capacity of the HNN reduced dramatically which leads to the more local minimum solution (suboptimal solution). Based on Alway et al. [2], the abundance of local minima solutions manifested the weaknesses in the learning phase that requires an optimization algorithm towards the synaptic weight management even though the computation has been correct via Wan Abdullah method. Thus, the work of [2] has confirmed the importance of having effective metaheuristics to reduce the learning error.

Finding the consistent interpretation that leads to zero cost function is important to ensure HNN always learn Satisfiable logic. Kasihmuddin et al. [9] proposed a Genetic Algorithm (GA) in minimizing the cost function associated with 2SAT logical rule. The proposed GA is compared with the conventional exhaustive search [9] and is reported to reduce the learning error. The work of Kasihmuddin et al. [9] has provided an insight on the potential of metaheuristic algorithm in minimizing the cost function and optimizing the learning phase of 2SAT logical rule. This shows the importance of exploitation in minimizing the cost function that leads to the optimal synaptic weight which will lead to an optimal final neuron state. The main problem with the proposed evolutionary algorithm is the difficulty of the solution to converge to the final neuron state. In this case, effective swarm metaheuristics (SI) algorithm is required to locate the optimal final neuron state with a smaller number of iterations.

Ant Colony Optimization (ACO) is proposed as a method to solve hard combinatorial optimization problems (COPs) [6]. Based on [6] ACO has been designed as an ideal algorithm for discrete optimization due to the existence of various discrete operator and optimizer that will im-

prove the learning phase of the computational model. Among the many strands of SI, the ACO algorithm is considered one of the most successful ones [3, 4]. Both [3, 4] has mapped the inspiration of natural and social swarm insect behaviour into a potential swarm intelligence system in terms of optimization algorithms with related formulations. For instance, by leaving behind a trail of pheromones, ants will locate the shortest trail connecting their nest and the food source. The social behaviour of ants has been a breakthrough in swarm metaheuristic development. Zhang and Crossley [15] showed that ACO can be utilized to effectively solve optimization problems and ACO produces an optimal solution. Based on [15], the work has been applied as a basis of the formulation of our proposed ACO algorithm in this work. Therefore, the reliability of ACO has affirmed the capability of ACO in optimizing the learning phase in HNN. ACO is efficient and useful since it can acquire satisfactory solutions in an acceptable computation time. The main difference between the conventional ACO with this paper is the use of binary output to represent the variable of the logical rule. To our best knowledge, there is no recent implementation of binary ACO during the learning phase of HNN. By capitalizing on the feature of the artificial ants that can exploit and explore, ACO will be able to minimize the cost function associated with the logical rule. In this case, the solution produced by ACO will be diversified during the global and local search. Hence, the contribution of this paper:

i) We proposed 2 Satisfiability logical rules in Hopfield Neural Network. The proposed logical rule can be implemented into Hopfield Neural Network by comparing the cost function with final the Lyapunov Energy function.

ii) Functional binary Ant Colony optimization that utilizes pheromone and visibility density model will be proposed. The proposed metaheuristics learn all the logical combinations of logic during the learning phase of the Hopfield Neural Network.

iii) Extensive analysis of the proposed hybrid network on the simulated datasets. The performance of the proposed hybrid network is observed to be competitive with the state-of-the-art Hopfield Neural network model.

In this paper, ACO will be embedded in HNN to do the 2SAT logical rule. Thus, HNN, 2SAT and ACO will be integrated as a single intelligent unit that produces the final neuron state corresponds to the learned logical rule. The rest of the paper is organized as follows: the formulation of the proposed logical rule namely 2 Satisfiability is given in the next section. The proposed 2SAT will be implemented in HNN in the next section. The detail of the proposed ACO will be explained and the detailed experimentation setup will be discussed. Finally, the final two sections will emphasize the result, discussion, and concluding remarks.

2 Materials and Methods

2.1 2 Satisfiability

The 2 Satisfiability (2SAT) is a logical rule that comprises of only 2 literals per clause. 2SAT is usually expressed as Boolean formulas called Conjunctive Normal Form (CNF) or Krom formulas. 2SAT consists of three components [9]:

i) A set of x variables, $V_1, V_2, ..., V_x$.

ii) A set of literals. A literal can be any variable or a negation of any variable.

iii) A set of y definite clauses, $C_1, C_2, C_3..., C_y$ linked by logical AND ($\wedge$). Each clause comprises of strictly 2 literals joined by just logical OR ($\vee$).

Each of the variable can only take bipolar value of 1 or -1 which represents true or false respectively. Explicit definition of the 2SAT formula P_{2SAT} is given by

$$P_{2SAT} = \bigwedge_{i=1}^{y} C_i, \tag{1}$$

where C_i is a list of clause with 2 variables each,

$$C_i = \bigvee_{i=1}^{y} (m_i, n_i), \tag{2}$$

where m_i and n_i are the possible variables of the clause C_i. The main objective of P_{2SAT} is to find the consistent interpretation that make formula P_{2SAT} become satisfied [9]. The selection of 2 literals per clause in P_{2SAT} reduce the logical complexity in discovering the relationship between the variables ANN.

2.2 2 Satisfiability in Hopfield Neural Network

HNN is one of the most commonly used neural network models. It is a simple neural network model that has feedback connections. HNN systematically stores patterns as a content addressable memory (CAM) [10]. The work of [10] has been tremendously contributed in elucidating the brief process of CAM storage and memory, as the main features of HNN. HNN is a network of N interconnected neurons where the output and input of each neuron is connected. The connection weight from neuron i to j is denoted by w_{ij}. In HNN, $w_{ij} = w_{ji}$, (symmetric networks), and $w_{ii} = w_{jj} = 0$ (no self-feedback connections). Let S_i be the state or output of the i th unit, θ_i is the pre-defined threshold of unit i. For bipolar networks, S_i is either $+1$ or -1. General updating rule in HNN is given by:

$$S_i = \begin{cases} 1 & \text{if } \sum_j^N w_{ij} S_j \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases} \tag{3}$$

P_{2SAT} can be implemented in HNN by assigning neuron to each variable in the formula. In this case, there is no redundant variable in P_{2SAT} when it is implemented in HNN. The implementation of P_{2SAT} in HNN is abbreviated as HNN-2SAT. Note that, the aim of the HNN-2SAT is to create an HNN that retrieve P_{2SAT} pattern during the retrieval phase. The cost function $E_{P_{2SAT}}$ that corresponds to P_{2SAT} is given as follows:

$$E_{P_{2SAT}} = \sum_{i=1}^{NC} \sum_{j=1}^{NV} L_{ij}, \tag{4}$$

where NC and NV represent the number of clause and variable respectively. The logical inconsistency L_{ij} is obtained by negating Equation (4):

$$L_{ij} = \begin{cases} \frac{1}{2}(1 - S_x) & \text{if } \neg x, \\ \frac{1}{2}(1 + S_x) & \text{otherwise.} \end{cases} \tag{5}$$

Similar to conventional HNN, synaptic weight of the HNN-2SAT is symmetrical and there is no self-connection between the neurons. HNN-2SAT employs Wan Abdullah method [1] where the synaptic weight is obtained by comparing $E_{P_{2SAT}}$ with $H_{P_{2SAT}}$. Thus, the local field $h_i(t)$ of the HNN-2SAT is given by:

$$h_i(t) = \sum_{j=1, i\neq j}^{N} w_{ij}^{(2)} S_j + w_i^{(1)}. \tag{6}$$

The output classification of the Equation (6) is given as follows:

$$S_i(t) = \begin{cases} 1, & \sum_{j=1,i\neq j}^{N} w_{ij}^{(2)} S_j + w_i^{(1)} \geq 0, \\ -1, & \sum_{j=1,i\neq j}^{N} w_{ij}^{(2)} S_j + w_i^{(1)} < 0. \end{cases} \tag{7}$$

The final state of neurons will be examined by using Lyapunov or energy function:

$$H_{P_{2SAT}} = -\frac{1}{2} \sum_{i=1,i\neq j}^{N} \sum_{j=1,i\neq j}^{N} w_{ij}^{(2)} S_i S_j - \sum_{i=1}^{N} w_i^{(1)} S_i. \tag{8}$$

Note that, Equations (7)-(8) ensure the final neuron state will always converge the nearest minimum energy. In this context, if the final neuron state is considered correct if it achieves global minimum energy. Another interesting perspective is, we can determine the absolute minimum energy $H_{P_{2SAT}}^{min}$ that corresponds to the number of clause in the HNN-2SAT based on the following formula:

$$H_{P_{2SAT}}^{min} = -\frac{\epsilon}{4}, \tag{9}$$

where ϵ is the number of clause C_i in HNN-2SAT. Hence, the quality of the solution produced by HNN-2SAT can be examined using the following condition:

$$|H_{P_{2SAT}} - H_{P_{2SAT}}^{min}| \leq \omega, \tag{10}$$

where ω is a tolerance value of the network. The main issue of this implementation is the logical verification during learning phase of HNN-2SAT. As the number of neuron increased, HNN-2SAT requires more feasible search space to obtain the optimal interpretation. In other word, Ant Colony Optimization is required to find the interpretation that minimize the inconsistency of the logical rule before the optimal synaptic weight can be assigned.

2.3 Ant Colony Optimization

Ant Colony Optimization (ACO) is simulated by the behaviour of foraging of real ants [6]. Real ants traverse the space surrounding their nest in random when searching for food. The ant will evaluate and carry back some of the food to the nest when it finds a food source. When travelling back to the nest, the ant will leave a trace of pheromone on the ground. Density of the pheromone it deposits is decided by the amount and value of the food. Pheromones deposited will lead the other ants to the food source. Through the pheromone trails, the ants are able to

find short paths from their nest to the food source [7]. The work of [7] relates the movements of Ants in finding the shortest path with the concept of optimization. Therefore, it has been a solid proof of the potential application of ACO in optimization problems including our propositional logic. ACO algorithm consists of artificial ants (agents) with distinctive functions and structures. The agents work with each other to accomplish a potential unified behaviour for the system as a whole, creating a vigorous system that has the ability to find high quality solutions for problems with a huge search space. Dorigo et al. [6] proposed this system as a metaheuristic to solve COPs. This metaheuristic has been proven to be vigorous and flexible since it has been put into use successfully to different COPs. This fact can also be supported in the recent application by Sreelaja [12], binary ACO can be implemented to solve a binary related problem if the problem is well presented. Thus, the work of Sreelaja [12] has approved the usage of binary ACO which relates to the binary propositional logic such as 2SAT. ACO can be implemented in HNN-2SAT by assigning the artificial ant as neurons with the potential state of 1 and -1. Hence, the aim of the of the ACO in HNN-2SAT is to find the interpretation that minimize the cost function corresponds to the P_{2SAT} logical rule. The work of Changdar et al. [5] has contributed to the main formulation of the proposed ACO in HNN-2SAT, with major modifications in the fitness calculation to align with 2SAT logical rule representation. The implementation of ACO algorithm in learning P_{2SAT} (HNN-2SATACO) is inspired by Changdar et al. [5]:

Step 1: Initialization. Initialize potential biploar neuron state , S_i where $S_i(t) \in [-1, 1]$.

Step 2: Fitness evaluation. Calculate the fitness of $S_i(t)$ using the following equation [9]:

$$f(S_i(t)) = \sum_{i=1}^{NC} C_i, \tag{11}$$

where NC is the number of clause in P_{2SAT} and C_i is given as follows:

$$C_i = \begin{cases} 1, & \text{true,} \\ 0, & \text{false.} \end{cases} \tag{12}$$

Step 3: Pheromone density initialization. Pheromone for each value of candidate group 1 or -1 is represented by a real vector $T_{ij}(1) = (T_{i1}, T_{i2}, ..., T_{iU})$ and $T_{ij}(-1) = (T_{i1}, T_{i2}, ..., T_{iU})$ where each T_{ij} is a random number between $[0, 1]$, $i = 1, 2, ..., V$; $j = 1, 2, ..., U$.

Step 4: Visibility density initialization. Visibility density for each value of candidate group 1 or -1 is represented by a real vector $\eta_{ij}(1) = (\eta_{i1}, \eta_{i2}, ..., \eta_{iU})$ and $\eta_{ij}(-1) = (\eta_{i1}, \eta_{i2}, ..., \eta_{iU})$ where each η_{ij} is a random number between $[0, 1]$, $i = 1, 2, ..., V$; $j = 1, 2, ..., U$.

Step 5: Ants searching phase. The movement probability of the ant "k" $(k = 1, 2, ..., M)$ is defined as follows:

$$p_{ij}^{k}(-1) = \frac{[T_{ij}(-1)]^{\alpha} . [\eta_{ij}(-1)]^{\beta}}{[T_{ij}(-1)]^{\alpha} . [\eta_{ij}(-1)]^{\beta} + [T_{ij}(1)]^{\alpha} . [\eta_{ij}(1)]^{\beta}}, \tag{13}$$

where α $(\alpha > 0)$ is the relative importance of the pheromone and β $(\beta > 0)$ is the relative importance of the visibility of the ants. Hence, the complementary of the movement is written as follows:

$$p_{ij}^{k}(-1) = 1 - p_{ij}^{k}(-1), \tag{14}$$

where p_{ij}^{k} is the probability of movement from the bit "i" to the state "j" at time t.

Step 6: Evaporation. The decrement of pheromone is based on the following equations:

$$T_{ij}(-1)(t+1) = (1-\rho)T_{ij}(-1)(t) + \Delta T_{ij}^{best}, \tag{15}$$

$$T_{ij}(1)(t+1) = (1-\rho)T_{ij}(1)(t) + \Delta T_{ij}^{best}, \tag{16}$$

$$\Delta T_{ij}^{best} = \frac{1}{f(S_i^{best})}, \tag{17}$$

where ρ is the coefficient representing evaporation rate and $\rho \in [0,1]$, $f(S_i^{best})$ is the number of clause for 2SAT.

Step 7: Refinement. Calculate the fitness of the solution $f(S_i)$ by using equation (11). The optimal state found so far will be recorded as $S_i^*(t+1)$. If $S_i^*(t+1)$ is superior than $S_i^*(t)$, then $S_i^*(t)$ is replaced by $S_i^*(t+1)$. Pheromone density will be updated by using equation (13). If $f(S_i) \neq NC$, repeat Step 4 to Step 7 until pre-determined trial, $Trial$ is achieved.

2.4 Experimental Design

In this paper, simulation for HNN-2SATACO will be implemented in Dev C++ Version 5.11 on a computer equipped with Intel Core i7 2.5GHz processor and 8GB RAM using Windows 8.1. All outputs that exceed the threshold CPU time, which is 24 hours are excluded. The effectiveness of our proposed method will be compared with several existing model such as:

a) Exhaustive Search (HNN-2SATES) deploys a simple random search during learning phase of HNN. In order to make a fair comparison, the logical rule of this method has been changed to P_{2SAT} instead of using HORNSAT.

b) Genetic Algorithm (HNN-2SATGA) proposed by Kasihmuddin et al. [9]. GA consist of chromosomes that serve as a potential neuron state during learning phase of HNN. This method consists of several operators such as selection, crossover, mutations. Note that, HNN-2SATGA represents the performance of the evolutionary algorithm.

c) Kernel Hopfield Neural Network (KHNN-2SAT) adds kernel function such as Gaussian into the local field of the HNN-2SAT. The kernel function increases the storage capacity of HNN-2SAT.

Simulated data set is used for this experiment by generating random neuron state during both learning and retrieval phase of HNN-2SAT. Note that, Hyperbolic Activation function (HTAF) will be implemented during retrieval phase due to avoid possible neuron oscillation that leads to suboptimal neuron state. The parameter involved in all HNN-2SAT model are given in Table 1 until Table 4. Note that, each HNN-2SAT model will generate 100 different logical combination ($COMBMAX$) to avoid possible biasness. To assess the quality of the retrieval phase, each $COMBMAX$ will produce 100 final neuron states ($Trial$) via local field and Hyperbolic Activation Function. By creating 100 final neuron states, we can assess the capability of the HNN-2SAT in producing global minimum solution. Most of the parameters in ACO is set based on the work

of Wan et al. [13]. [13] has been a great reference for the most feasible parameter of binary ACO which will be applied in our propposed HNN-2SATACO. Figure 1 shows the general implementation of HNN-2SAT model.

Table 1: List of parameter in HNN-2SATACO (Proposed work).

Parameter	Considered Parameter Value
Neuron Combination $COMBMAX$	100 [9]
Trial	100 [9]
Tolerance Value (ω)	0.001 [9]
Activation Function	Hyperbolic [9]
Number of Ant	50 [13]
Pheromone (α)	1 [13]
Visibility (β)	3 [13]
Exploration of the pheromone (ρ)	$[0, 1]$ [13]

Table 2: List of parameter in HNN-2SATGA [9].

Parameter	Considered Parameter Value
Neuron Combination $COMBMAX$	100 [9]
Trial	100 [9]
Tolerance Value (ω)	0.001 [9]
Activation Function	Hyperbolic [9]
Number of Chromosome	100 [9]
Number of generation	100 [9]
Selection Rate	0.1
Crossover Rate	1
Mutation Rate	0.1

Table 3: List of parameter in HNN-2SATES [9].

Parameter	Considered Parameter Value
Neuron Combination $COMBMAX$	100 [9]
Trial	100 [9]
Tolerance Value (ω)	0.001 [9]
Activation Function	Hyperbolic [9]

Table 4: List of parameter in KHNN-2SAT.

Parameter	Considered Parameter Value
Neuron Combination $COMBMAX$	100 [9]
Trial	100 [9]
Tolerance Value (ω)	0.001 [9]
Type of Kernel	Linear Kernel

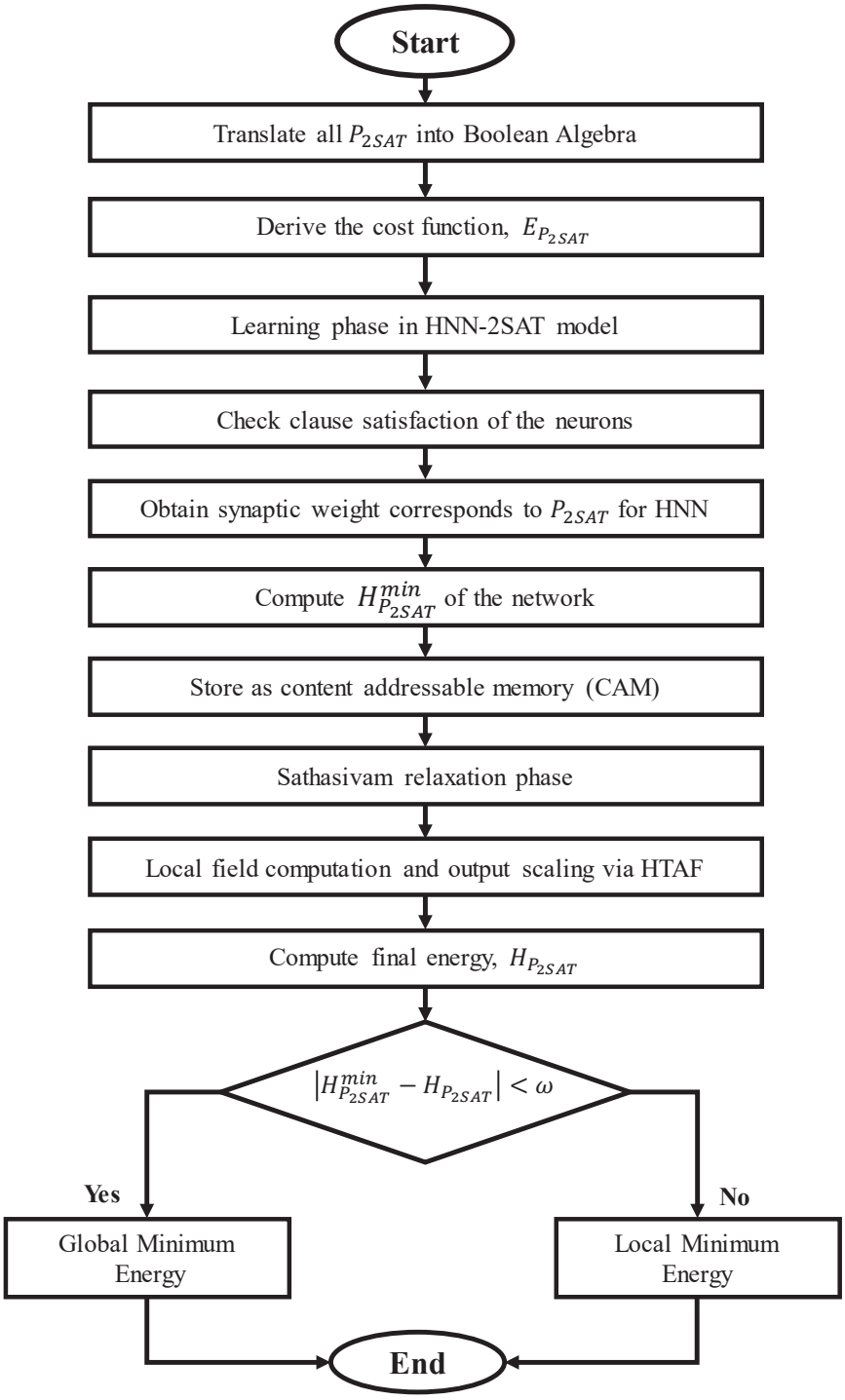

Figure 1: Flow diagram for algorithm of implementation of the HNN-2SAT models.

2.5 Performance Evaluation

To evaluate the efficiency of all HNN-2SAT models, a total of six performance evaluation metrics, namely root mean square error, mean absolute error, sum of squared error, mean absolute percentage error, global minima ratio and CPU time will be analysed.

2.5.1 Root Mean Square Error

Root mean square error (RMSE) is normally used to compute the differences between target value and the actual observed value of the model. RMSE measure the deviation of the error between current solution fitness with the desired fitness. The equation for RMSE is defined as [11]

$$RMSE = \sum_{i=1}^{n} \sqrt{\frac{1}{n}(f_{NC} - f_i)^2}, \tag{18}$$

where f_{NC} is the total number of 2SAT clauses, f_i is the fitness of the solution in HNN-2SAT model and n is the number of iteration before $f_i = f_{NC}$. The best HNN-2SAT model will have the smallest value of RMSE. The main contribution of [11] is the standard formulation of RMSE which will be used in this work.

2.5.2 Mean Absolute Error

Mean absolute error (MAE) is the mean of the absolute values of the errors. MAE is derived from the absolute difference of $f_{NC} - f_i$. MAE is defined by the following equation [14].

$$MAE = \sum_{i=1}^{n} \frac{1}{n}|f_{NC} - f_i|, \tag{19}$$

where f_{NC} is the total number of 2SAT clauses, f_i is the fitness of the solution in HNN-2SAT model and n is the number of iteration before $f_i = f_{NC}$. Similar to RMSE, the least value of MAE indicates the best HNN-2SAT model. The main contribution of [14] is the fundamental formulation of MAE to be applied in our work.

2.5.3 Sum of Squared Error

Sum of squared error (SSE) is the addition of the squared differences between the target value and observed value of the model. SSE evaluate the sensitivity of the error of the HNN-2SAT model. The equation for SSE is expressed as:

$$SSE = \sum_{i=1}^{n} (f_i - f_{NC})^2. \tag{20}$$

The best HNN-2SAT model will be determined by the smallest value of SSE.

2.5.4 Mean Absolute Percentage Error

Mean absolute percentage error (MAPE) is the mean of the absolute values of the errors in percentage terms. MAPE is a measure the error of HNN-2SAT in the percentage form. MAPE can be expressed as:

$$MAPE = \sum_{i=1}^{n} \frac{100}{n} \frac{|f_{NC} - f_i|}{|f_i|}. \tag{21}$$

The theory of MAPE is very simple, however, it has a crucial flaw. MAPE cannot be used if the observed value is zero as it will lead to division by zero. The best HNN-2SAT model will have the lowest percentage of MAPE.

2.5.5 Global and Local Minima Ratio

Global minima ratio, Z_m is defined as the ratio of the total global minimum energy to the total number of simulations. Each HNN-2SAT model will produce 10,000 solutions per execution, hence the ratio of global minimum energy will help to determine the accuracy of the model. Every computed final energy of the neurons in HNN-2SAT model is filtered by a specific value of tolerance, ω. The final energy is considered as global minimum energy if it is within the tolerance value. The equation of global minima ratio is defined as

$$Z_m = \frac{1}{tc} \sum_{i=1}^{n} N_{H_{P_{2SAT}}}. \tag{22}$$

The equation of local minima ratio is defined as:

$$L_m = 1 - Z_m, \tag{23}$$

where t is the number of trial, c is the neuron combination and $N_{H_{P_{2SAT}}}$ is the number of global minimum energy of the proposed model. HNN-2SAT model with higher value of Z_m has higher accuracy.

2.5.6 Computational Time

CPU time is defined as the time required by a particular HNN-2SAT model to finish one execution. CPU time denotes the stability and competency of the HNN-2SAT models. Each simulations will be run on identical processor to cancel off the effect of bad sector and memory build-up. Equation of the CPU time is given by

$$CPU\_Time = Learning\_Time + Retrieval\_Time. \tag{24}$$

The best HNN-SAT model has the least computation time during the learning phase. Hence, the best HNN-2SAT model will have the shortest CPU time.

3 Results and Discussion

The performance of simulated program for all HNN-2SAT models will be discussed based on efficiency, accuracy and stability. Efficiency of the model will be assessed based on the values of RMSE, MAE, SSE and MAPE. Accuracy of the model will be evaluated based on the global minima ratio, Z_m. Stability of the model will be determined by the CPU time.

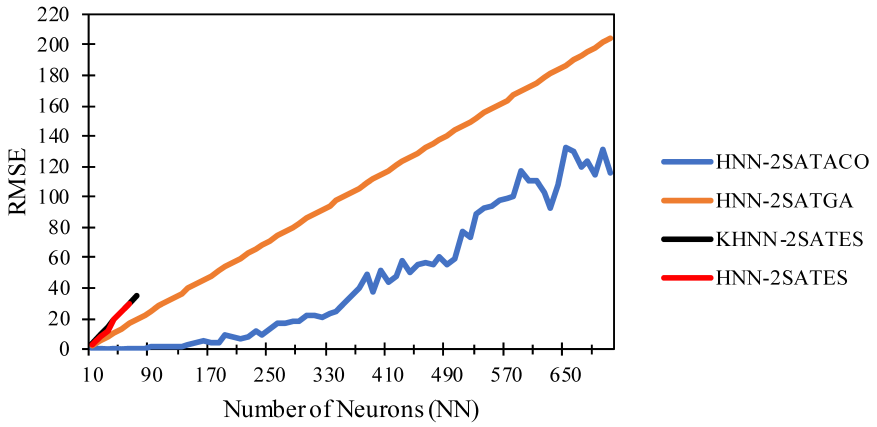

Figure 2: RMSE for HNN-2SAT models.

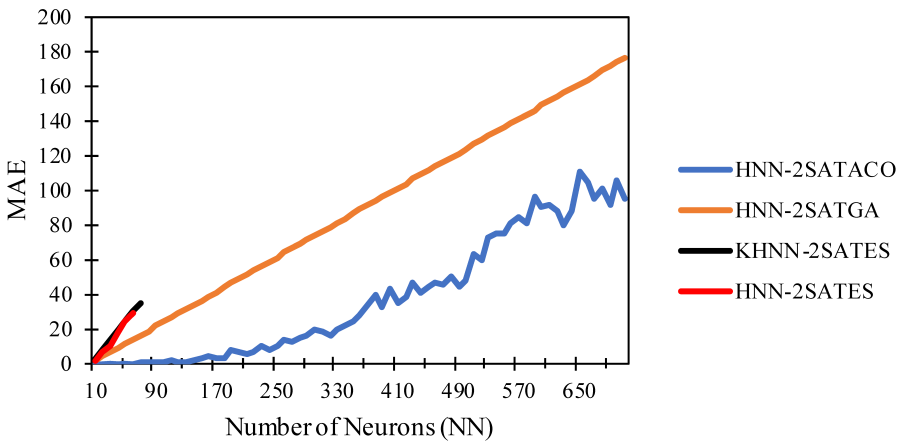

Figure 3: MAE for HNN-2SAT models.

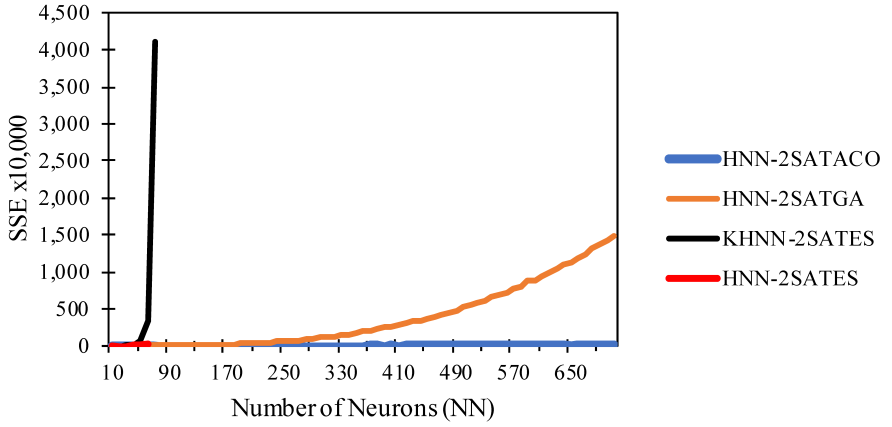

Figure 4: SSE for HNN-2SAT models.

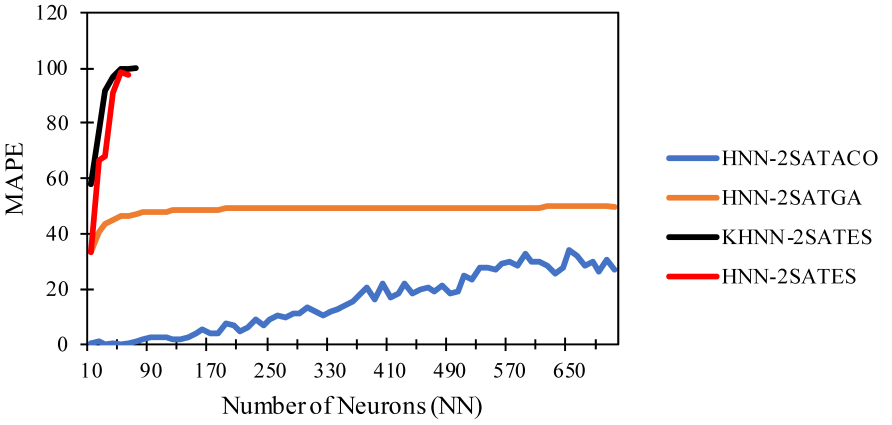

Figure 5: MAPE for HNN-2SAT models.

Figure 2, Figure 3 and Figure 4 shows the RMSE, MAE and SSE results for all HNN-2SAT models. As the results shown, HNN-2SATACO outperformed HNN-2SATES, KHNN-2SATES and HNN-2SATGA in terms of RMSE, MAE and SSE. There were no results reported for HNN-2SATES after NN $= 60$ and no results for KHNN-2SATES after NN $= 70$ since these 2 learning model exceeds the threshold CPU time. It can be seen that the solution in HNN-2SATES and KHNN-2SATES has deviated further from the optimal fitness by judging from the value of RMSE. The value of MAE has also increased due to this huge deviation because the probability of finding consistent P_{2SAT} interpretation decreases dramatically as the number of P_{2SAT} clause increases. HNN-2SATGA needs more initial iterations before chromosomes can reach global solution, hence leads to error accumulation and higher value of errors. On contrary, the effect of interaction between the ants and pheromone density helps HNN-2SATACO to diversify candidate solution in search space. Any non-fit solution after pheromone evaporation will be improved further by pheromone density initialization. Two layered optimization mechanism of pheromone initialization and pheromone evaporation reduces the deviation error of the network and results in positive results which is the minimal RMSE, MAE and SSE accumulation. Value of RMSE, MAE and SSE for all HNN-2SAT models have seen an increase in trend as the number of neurons

of P_{2SAT} clause increases. This is due to the high number of iterations required to satisfy high number of clauses during the learning phase.

Figure 5 shows the MAPE results for all HNN-2SAT models. Based on Figure 5, the value of MAPE for HNN-2SATACO is much lower compare to the other models. A large percentage of clause in HNN-2SATES and KHNN-2SATES from the total P_{2SAT} clause is not fully satisfied in the learning phase. The MAPE value rises as the number of P_{2SAT} inconsistencies increases and the probability of getting all satisfied clause decreases dramatically. Around 49% of the P_{2SAT} clauses is inconsistent before the learning arrive to the global solution because HNN-2SATGA encounters inadequate chromosomal crossover and needs more iterations to attain the correct clause. In contrary, HNN-2SATACO has achieved more consistent clause from the total clause since it has minimal value of MAPE. Interaction between ants through the pheromone density has reduced the possibility of HNN-2SAT model to meet inconsistent solution. According to [13], pheromone density plays an important role to reduce the probability of the ant to get trapped in the suboptimal path.

Table 5: Z_m and L_m for HNN-2SAT models.

	HNN-2SATACO		HNN-2SATGA		KHNN-2SATES		HNN-2SATES	
NN	Z_m	L_m	Z_m	L_m	Z_m	L_m	Z_m	L_m
10	1.0000	0	1.0000	0	1.0000	0	1.0000	0
20	1.0000	0	1.0000	0	1.0000	0	1.0000	0
30	1.0000	0	1.0000	0	1.0000	0	0.9992	0.0008
40	1.0000	0	1.0000	0	1.0000	0	0.9983	0.0017
50	1.0000	0	1.0000	0	0.9995	0.0005	0.9974	0.0026
60	1.0000	0	1.0000	0	0.9981	0.0019	0.9963	0.0037
70	1.0000	0	1.0000	0	0.9978	0.0022	–	–
80	1.0000	0	1.0000	0	–	–	–	–
90	1.0000	0	0.9994	0.0006	–	–	–	–
100	1.0000	0	0.9942	0.0058	–	–	–	–

Table 5 shows the global minima ratio, Z_m for all HNN-2SAT models. If the value of Z_m in HNN-2SAT model is close to 1, almost all solution of the HNN-2SAT model has achieved global minimum energy. In this analysis, we limit our comparison up to NN $= 100$ because several HNN-2SAT model such as HNN-2SATES and KHNN-2SAT were trapped in trial and error state. The complexity of the HNN-2SAT model will increase and the final state of the neurons tends to trap at suboptimal solution (local minimum energy) when the number of neuron increases [9]. It is shown in Table 5 that the value of Z_m for HNN-2SATACO is closer to 1 compared to the other HNN-2SAT models. Note that, HNN-2SATES and KHNN-2SATES were facing issue during learning phase because as the number of neurons increased, the probability of achieving the optimal interpretation reduce dramatically. This reduces the relaxation time of the network before all the neurons advance to retrieval phase. As a consequent, the neurons in HNN-2SATES and KHNN-2SATES are not fully relaxed and tend to retrieve the wrong final states. HNN-2SATGA lowers the convolution of the network to discover the correct states and unsatisfied P_{2SAT} clauses will be enhanced through crossover and mutation until it reaches the correct neuron state. In contrast, HNN-2SATACO utilizes an effective searching technique that can complete the learning phase with minimal complexity. As a result, the neuron state for HNN-2SATACO during retrieval phase has minimal state oscillation and can be updated correctly.

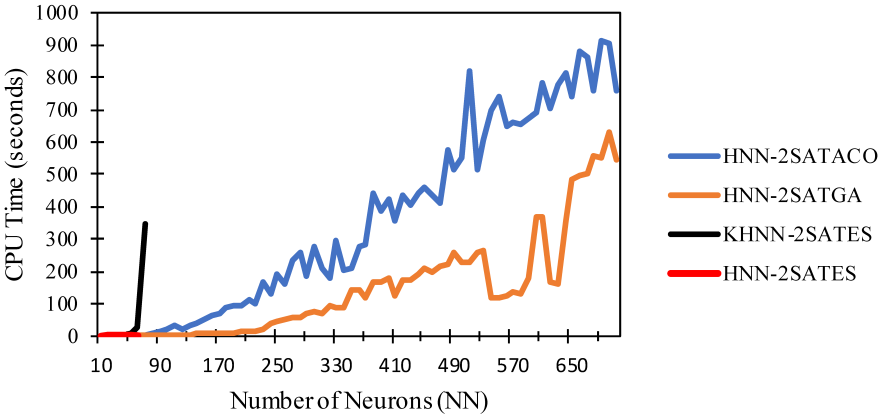

Figure 6: CPU time for HNN-2SAT models.

Figure 6 shows the CPU time for all HNN-2SAT models. Based on Figure 6, all models have similar CPU time up to NN $= 60$ because all the HNN-2SAT model achieved zero cost function at the same time. HNN-2SATES has reached its limits when NN > 60 and KHNN-2SATES when NN > 70 as the CPU time exceeds the threshold CPU time set for this research. On the contrary, HNN-2SATACO and HNN-2SATGA continues to produce results up to NN $= 710$. During learning phase of HNN-2SAT, optimal searching technique is required to drive the solution to maximum fitness in acceptable time range. HNN-2SATES and KHNN-2SATES requires a significant amount of time to reach the maximum fitness. At NN > 60, HNN-2SATES failed to accomodate the complexity of the learning phase and is observed to trap in trial and error state during learning phase. However, HNN-2SATACO and HNN-2SATGA are able to sustain up to NN $= 710$ and require less CPU time to complete a single execution. HNN-2SATACO accentuates on fitness improvement in every iteration. Solution fitness in HNN-2SATACO also significantly improves in every learning iteration and this reduces the CPU time. The indirect communication of ants through pheromone density has brought positive impact to the quality of the solutions in a shorter time. HNN-2SATGA is reported to complete the learning phase slightly faster than HNN-2SATACO. This shows that in the early learning phase, HNN-2SATACO takes more iterations to complete both global search and local search while HNN-2SATGA can perform the crossover between parent P_{2SAT} chromosomes straight away in the learning phase. Hence, more CPU time is required before the network can enter the relaxation phase for HNN-2SATACO.

4 Conclusions

Creating optimal HNN that is governed by logical rule is an important problem in many applications especially in logic mining. In this work, the P_{2SAT} is embbeded into HNN by comparing the cost function with Lypunov Energy function. Due to the complexity of the learning phase in the existing work, this work proposed a Hybrid ACO in finding the correct interpretation that minimizes the cost function if the HNN-2SAT. The proposed approach called HNN-2SATACO, has been evaluated across various performance evaluation such as RMSE, MAE, SSE, MAPE, Z_m and CPU time. According to the experimental results, the HNN-2SATACO outperformed the other HNN-2SAT models in almost all performance evaluations. The proposed hybrid HNN-2SATACO model gives minimal value for RMSE, MAE, SSE, and MAPE. All the solution retrieved from the HNN-2SATACO are global minimum solution which leads to the value of Z_m that is approach-

ing 1. The proposed model has a reasonable CPU time and can generate final neuron state up to NN = 710. For future work, we can aim to reduce the execution time by improving the initial neuron state of the ACO. In this context, evolutionary algorithm such GA can be used to filter the initial solution so that only optimal trail can be explored. In addition, ACO can be extended into non-Satisfiable logical rule such as Maximum Satisfiability and Major Satisfiability [2] by minimizing the cost function of the special logical rule.

Acknowledgement This research was funded by Fundamental Research Scheme Grant (FRGS), grant number FRGS/1/2019/STG06/USM/02/4. The authors would like to express special dedication to all the researchers from AI Research Development Group (AIRDG) for the continuous support.

Conflicts of Interest The authors declare no conflict of interest.

References

[1] W. A. T. W. Abdullah (1992). Logic programming on a neural network. *International Journal of Intelligent Systems*, *7*(6), 513–519.

[2] A. Alway, N. E. Zamri, S. A. Karim, M. A. Mansor, M. S. M. Kasihmuddin & M. M. Bazuhair (2021). Major 2 satisfiability logic in discrete Hopfield neural network. *International Journal of Computer Mathematics*, *1*, 1–45.

[3] E. Bonabeau, M. Dorigo & G. Théraulaz (1999). *Swarm intelligence: From natural to artificial systems*. Oxford University Press, England.

[4] E. Bonabeau, M. Dorigo & G. Théraulaz (2000). Inspiration for optimization from social insect behaviour. *Nature*, *406*(6791), 39–42.

[5] C. Changdar, G. Mahapatra & R. K. Pal (2013). An ant colony optimization approach for binary knapsack problem under fuzziness. *Applied Mathematics and Computation*, *223*, 243–253.

[6] M. Dorigo, G. Di Caro & L. M. Gambardella (1999). Ant algorithms for discrete optimization. *Artificial Life*, *5*(2), 137–172.

[7] S. Goss, S. Aron, J.-L. Deneubourg & J. M. Pasteels (1989). Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, *76*(12), 579–581.

[8] J. J. Hopfield & D. W. Tank (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, *52*(3), 141–152.

[9] 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.

[10] M. K. Muezzinoglu, C. Guzelis & J. M. Zurada (2003). A new design method for the complex-valued multistate Hopfield associative memory. *IEEE Transactions on Neural Networks*, *14*(4), 891–899.

[11] F. Schwenker, H. A. Kestler & G. Palm (2001). Three learning phases for radial-basis-function networks. *Neural Networks*, *14*(4-5), 439–458.

[12] N. Sreelaja (2021). Ant colony optimization based light weight binary search for efficient signature matching to filter ransomware. *Applied Soft Computing*, *111*, 107635. https://doi.org/10.1016/j.asoc.2021.107635.

[13] Y. Wan, M. Wang, Z. Ye & X. Lai (2016). A feature selection method based on modified binary coded ant colony optimization algorithm. *Applied Soft Computing*, *49*, 248–258.

[14] C. J. Willmott, S. G. Ackleson, R. E. Davis, J. J. Feddema, K. M. Klink, D. R. Legates, J. O'donnell & C. M. Rowe (1985). Statistics for the evaluation and comparison of models. *Journal of Geophysical Research: Oceans*, *90*(C5), 8995–9005.

[15] B. Zhang & P. Crossley (2017). Reliability improvement using ant colony optimization applied to placement of sectionalizing switches. *Energy Procedia*, *142*, 2604–2610.