mathematics

MDPI

Article

Random Maximum 2 Satisfiability Logic in Discrete Hopfield Neural Network Incorporating Improved Election Algorithm

Vikneswari Someetheram [1], Muhammad Fadhil Marsani [1,*], Mohd Shareduwan Mohd Kasihmuddin [1], Nur Ezlin Zamri [2], Siti Syatirah Muhammad Sidik [1], Siti Zulaikha Mohd Jamaludin [1] and Mohd. Asyraf Mansor [2]

[1] School of Mathematical Sciences, Universiti Sains Malaysia—USM, Gelugor 11800, Penang, Malaysia
[2] School of Distance Education, Universiti Sains Malaysia—USM, Gelugor 11800, Penang, Malaysia
* Correspondence: fadhilmarsani@usm.my; Tel.: +60-4-6533657

Abstract: Real life logical rule is not always satisfiable in nature due to the redundant variable that represents the logical formulation. Thus, the intelligence system must be optimally governed to ensure the system can behave according to non-satisfiable structure that finds practical applications particularly in knowledge discovery tasks. In this paper, we a propose non-satisfiability logical rule that combines two sub-logical rules, namely Maximum 2 Satisfiability and Random 2 Satisfiability, that play a vital role in creating explainable artificial intelligence. Interestingly, the combination will result in the negative logical outcome where the cost function of the proposed logic is always more than zero. The proposed logical rule is implemented into Discrete Hopfield Neural Network by computing the cost function associated with each variable in Random 2 Satisfiability. Since the proposed logical rule is difficult to be optimized during training phase of DHNN, Election Algorithm is implemented to find consistent interpretation that minimizes the cost function of the proposed logical rule. Election Algorithm has become the most popular optimization metaheuristic technique for resolving constraint optimization problems. The fundamental concepts of Election Algorithm are taken from socio-political phenomena which use new and efficient processes to produce the best outcome. The behavior of Random Maximum 2 Satisfiability in Discrete Hopfield Neural Network is investigated based on several performance metrics. The performance is compared between existing conventional methods with Genetic Algorithm and Election Algorithm. The results demonstrate that the proposed Random Maximum 2 Satisfiability can become the symbolic instruction in Discrete Hopfield Neural Network where Election Algorithm has performed as an effective training process of Discrete Hopfield Neural Network compared to Genetic Algorithm and Exhaustive Search.

Keywords: knowledge discovery; explainable artificial intelligence; Random Maximum 2 Satisfiability; artificial neural network; Election Algorithm; potential forecasting model

MSC: 68T05; 68T27; 68T50

Citation: Someetheram, V.; Marsani, M.F.; Mohd Kasihmuddin, M.S.; Zamri, N.E.; Muhammad Sidik, S.S.; Mohd Jamaludin, S.Z.; Mansor, M.A. Random Maximum 2 Satisfiability Logic in Discrete Hopfield Neural Network Incorporating Improved Election Algorithm. *Mathematics* **2022**, *10*, 4734. https://doi.org/10.3390/math10244734

Academic Editor: Marek Sikora

Received: 21 September 2022
Accepted: 18 October 2022
Published: 13 December 2022

1. Introduction

The research of artificial neural networks (ANNs) provides interesting ideas in understanding the way brain interprets data and offers near optimal solutions to optimization problems. The ANN model was inspired by the group of biological neurons that was efficiently modeled and fires according to the goal of the whole neuron system. Due to that reason, ANNs have gained attention from various researchers from different backgrounds to solve various potential optimization problems [1–5]. The high demand of the ANN is due to the nature of ANNs that can improve the solution through specific iterations which can be run easily by computer program. One of the earliest ANNs is Hopfield Neural Network (HNN) which was proposed by Hopfield and Tank [6] to provide potential solution for the travelling salesman problem through the connectionist model. HNNs

consist of interconnected neurons with input and output neuron without hidden neurons. Each neuron fires and updates iteratively until the final neuron state converges towards near-optimal solution. Interestingly, the neuron state of the HNN can be interpreted in terms of Lyapunov energy function which is always minimized by the network. In this context, HNN will update the neuron state until the network achieves global minimum energy to ensure the optimality of the solution for any given optimization problem. Despite having wide applicability [7–10], HNNs are prone to storage capacity issues. As proposed by various studies [11,12], the number of stored memory patterns is severely limited (about 14%) which indicates the need for optimal neuron modeling in HNNs.

One of the earliest efforts to represent the neuron in the form of symbolic logic was proposed by Abdullah [13]. In this work, the HNN was viewed as a computational paradigm and symbolic rule rather than a tool to solve optimization problem. Logic was chosen as symbolic rule in HNN because logic conventionally involves a database of declared knowledge, sequential procedure, and resolution, which help HNNs to prove/disprove the goal of the network. The introduction of logic as a symbolic rule in HNNs has attracted other representations of logic. This led to the introduction of the Wan Abdullah method [14] to find the optimal synaptic weight associated with the embedded logical rule. This development has attracted researchers to find other variants of logic to be embedded into HNNs. Kasihmuddin et al. [15] proposed 2 Satisfiability (2SAT) logic in HNNs by creating a cost function that capitalizes the symmetric neuron connection. In this paper, the proposed 2SAT in HNN was effectively optimized using the Estimated Distribution Algorithm during the retrieval phase. As a result, the proposed HNN achieved a high rate of global minima ratio. Next, Sathasivam et al. [16] proposed the first non-systematic logic namely Random 2 Satisfiability (RAN2SAT) by proposing the combination of first and second order clause to the logic formulation. Although the quality of the final neuron state deteriorates as the number of neuron increases, the synaptic weight of the logic shows higher number of variations compared to the other existing logic. Karim et al. [17] proposed the higher order non-systematic logic by introducing a third order clause. This paper shows an interesting logical variant where the first, second, and third order clause were proposed interchangeably. The direction of non-systematic logic was extended by Sidik et al. [18] and Zamri et al. [19] where Weighted Random 2 Satisfiability (*r*2SAT) were proposed in HNN. To create the correct *r*2SAT logic, logic phase was introduced to ensure the right amount of negated literal was imposed to RAN2SAT. The proposed *r*2SAT was reported to obtain final neuron state with high total neuron variation. In another development, Guo et al. [20] combined the beneficial feature of both systematic and non-systematic logic by proposing Y-Type Random 2 Satisfiability (YRAN2SAT). The proposed logic shows an interesting behavior because YRAN2SAT can be reduced to both 2SAT and RAN2SAT. On the other hand, Gao et al. [21] extended the order of the clause in the logic by adding third order clause. Although the final energy for [20,21] tends to converge to local minimum energy (due to high number of neuron), both logics offer a wide range of flexibility to represent symbolic rule in HNN. Despite rapid development in the field of logic in HNN, none of the mentioned studies consider the existence of a redundant variable in the logical rule. In this context, a redundant variable with the opposing literals will usually lead towards non-satisfiable logic.

Maximum Satisfiability (MAXSAT) is another variant of logical rule that is not satisfiable in nature. According to Bonet et al. [22], MAXSAT is to find the interpretation that maximize the number of satisfied clauses. In this context, MAXSAT logic will never be satisfied, and the logical outcome is always False. Kasihmuddin et al. [23] proposed the first non-satisfiability logic namely Maximum 2 Satisfiability (MAX2SAT) in HNN. The cost function of the proposed MAX2SAT only considers the logic that is satisfiable where the synaptic weight for the non-satisfiable logic is zero. The proposed MAX2SAT utilized the exhaustive search and was reported to achieve global minimum energy for lower number of neurons. To reduce learning error during learning phase, Sathasivam et al. [24] proposed genetic algorithm (GA) to find the correct interpretation that leads to zero cost function.

The proposed GA was reported to increase the storage capacity and successfully prevent HNN from obtaining sub-optimal synaptic weight. Although the proposed metaheuristics were reported to produce zero learning error, the capability of the algorithm in doing non-systematic MAX2SAT remains unknown. Throughout this process, the real parameters will undergo some adjustments and it take several trials to get a good result. For further developments, in order to find the optimal solution, researchers have suggested distributional robust optimization techniques to solve the non-convex non-linear structure of the high dimensional parameter space of the neural network [25]. Next, some combinatorial problems based on min–max and min–max regret version have been discussed in [26].

Election Algorithm (EA) was initially proposed by Emami and Derakhshan [27] to optimize the solution of the combinatorial problems. EA is a social-political algorithm that was inspired by the presidential election process of the majority in a particular country. The intelligent search of EA can cover a wide range of solutions in a large solution space. Other than that, the mechanism of EA partitions the solution space where effective partitioning will help in reducing complexity and allowing the searching process to be more accurate. EA consists of three-layered optimization operators that will help in improving the solution in every iteration. Researchers have utilized EA for real-life problems where the versatility of EA can cater to both continuous and discrete optimization problems. In terms of RAN2SAT, Sathasivam et al. [28] proposed the first binary EA to optimize the learning phase of the HNN. In this context, several functions in the EA were replaced by the binary operator to fit the fitness function of the HNN. The proposed network was reported to outperform most of the state-of-the-art algorithm in doing RAN2SAT. Next, Bazuhair et al. [29] utilized EA in finding the correct interpretation for higher order RAN3SAT. Similar to the previous study, the proposed EA was reported to achieve almost zero error and maximum total neuron variation for RAN3SAT. This shows the superiority of the EA in reducing the learning complexity of the HNN. However, the performance of the proposed EA in doing non-satisfiable logic remains unknown. In this context, the proposed EA must have the capability to reduce the fitness of the neuron to non-zero cost function. Thus, the contributions of the present paper are as follows:

1. To formulate a novel non-satisfiability logical rule by connecting Random 2 Satisfiability with Maximum 2 Satisfiability into one single formula called Random Maximum 2 Satisfiability. In this context, the logical outcome of the proposed logic is always False and allows the existence of redundant variable. Thus, the goal of the Random Maximum 2 Satisfiability to find the interpretation that maximize the number of satisfied clauses.
2. To implement the proposed Random Maximum 2 Satisfiability into Discrete Hopfield Neural Network by finding the cost function of the sub-logical rule that is satisfiable. Each of the variables will be represented in terms of neurons and the synaptic weight of the neurons can be found by comparing cost function with the Lyapunov energy function.
3. To propose Election Algorithm that consists of several operators such as positive advertisement, negative advertisement, and coalition to optimize the learning phase of the Discrete Hopfield Neural Network. In this context, the proposed EA will be utilized to find interpretation that maximize the number of the satisfied clause.
4. To evaluate the performance of the proposed hybrid network in doing simulated datasets. The hybrid network consisting of Random Maximum 2 Satisfiability, Election Algorithm, and Hopfield Neural Network will be evaluated based on various performance metrics. Note that the performance of the hybrid network will be compared with other state of the art metaheuristics algorithm.

By creating an effective and efficient hybrid network, the proposed network creates a new method to learn non-satisfiable logic which accounts for most of real-life problem. Thus, this paper is organized as follows: Section 2 provides the preliminary explanation on the Random Maximum 2 Satisfiability, how Random Maximum 2 Satisfiability studies in DHNN, Genetic Algorithm, and Election Algorithm. The methods and experimental

setup will be given in Section 3. The simulation of the study will be discussed in Section 4. Finally, concluding remarks are given in the final section, Section 5.

Table 1 below shows the list of related research.

Table 1. Summaries of related studies.

Author(s)	Detail of the Studies	Summary and Findings
Hopfield and Tank [6]	The authors proposed non-linear analog neurons.	The authors used DHNN to solve the optimization of the Traveling Salesman Problem (TSP).
Abdullah [13]	The author proposed the Wan Abdullah method to solve logic programming in DHNN.	This author proposed a new method in retrieving the synaptic weight of Horn clause. The new method has outperformed Hebbian learning.
Kasihmuddin et al. [23]	The authors proposed Restricted Maximum *k* Satisfiability in DHNN.	The performance of MAX*k*SAT in DHNN outperformed MAX*k*SAT in Kernel Hopfield Neural Network (KHNN).
Sathasivam et al. [16]	The authors proposed a RAN2SAT has been developed to represent the non-systematic logical rule in DHNN.	RAN2SAT is embedded in DHNN by retrieving maximum outcomes of global solutions.
Zamri et al. [19]	The author proposed Implemented Imperialist Competitive Algorithm (ICA) in 3 Satisfiability (3 SAT) and compared with ES and GA.	This paper shows the comparison of the proposed model done by two real data sets and ICA has outperformed ES and GA.
Bazuhair et al. [29]	This paper proposed EA in training phase of RAN3SAT.	Incorporation of RAN3SAT with EA in DHNN has the ability to achieve the optimal training phase.
Gao et al. [21]	This paper utilized DHNN in explaining G-Type Random *k* Satisfiable.	This paper emphasizes on the formulation of first, second, and third order of clauses in the logical structure. The proposed logic structure produces higher solution diversity.
Karim et al. [17]	The author proposed a new novel of multi-objective HEA for higher order Random Satisfiability in DHNN.	HEA achieves highest fitness value. The HEA is able to yield a high quality of global optimal solution with higher accuracy and outperformed GA, ABC, EA, and ES.

2. Related Works

2.1. Random Maximum 2 Satisfiability

Random Maximum 2 Satisfiability (RANMAX2SAT) belongs to the non-systematic satisfiability model presented in Conjunction Normal Form (CNF). The logical structure consists of two redundant variables per clause and at most two random variables in each clause connected by a logical operator OR operator. RANMAX2SAT defined in CNF formula that consists of p clauses of two redundant variables and positive integer g where $g \leq p$ [23] and m clauses containing two random variables in each clause and n clauses containing one random variables in each clause. RANMAX2SAT problem gives the best bit string assignments to the variables by satisfying at least g of p clauses for clauses with redundant variables and m clauses for second order random variables and n clauses for first order random variables. The formulation for Maximum 2 Satisfiability (MAX2SAT) is given below [23]:

$$P_{MAX2SAT} = (A \vee B) \wedge (\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee \neg B)\,, \tag{1}$$

where variables A and B are redundant literals in the logical structure. The formulation for RAN2SAT is given as [16]:

$$P_{RAN2SAT} = \bigwedge_{i=1}^{m} C_i^{(2)} \bigwedge_{i=1}^{n} C_i^{(1)}\,, \tag{2}$$

where m is total number clauses with two variables and n is total number of clauses with one variable. Thus, the general formulation of P_{RM2SAT} is a combination of $P_{MAX2SAT}$ and $P_{RAN2SAT}$ as follow:

$$P_{RM2SAT} = P_{MAX2SAT} \wedge P_{RAN2SAT}\,, \tag{3}$$

$$P_{RM2SAT} = (A \vee B) \wedge (\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee \neg B) \bigwedge_{i=1}^{m} C_i^{(2)} \bigwedge_{i=1}^{n} C_i^{(1)}\,, \tag{4}$$

The definition of the clauses in P_{RM2SAT} is given by:

$$C_i^{(k)} = \begin{cases} (X_i^* \vee Y_i^*) & ,k = 2\,, 1 \leq i \leq m \\ Z_i^* & ,k = 1\,, 1 \leq i \leq n \end{cases}, \tag{5}$$

where $X_i^* \in \{X_i, \neg X_i\}$, $Y_i^* \in \{Y_i, \neg Y_i\}$ and $Z_i^* \in \{Z_i, \neg Z_i\}$ is a variable in P_{RM2SAT}. $C_i^{(2)}$ and $C_i^{(1)}$ defines second order clauses and first order clauses, respectively. One of the examples for RANMAX2SAT formulation is given:

$$P_{RM2SAT} = (A \vee B) \wedge (\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee \neg B) \wedge (X_1 \vee Y_1) \wedge (X_2 \vee Y_2) \wedge Z_1 \wedge Z_2 \tag{6}$$

According to the above equation, if $P_{RM2SAT} = (A, B, X_1, Y_1, X_2, Y_2, Z_1, Z_2) = (1, 1, 1, 1, 1, 1, 1, 1)$, the outcome of above equation is $P_{RM2SAT} = -1$ with only seven clauses satisfied. Note that P_{RM2SAT} is unsatisfiable and there is no interpretation that make P_{RM2SAT} true. Based on the example, the highest number of causes that can be satisfied is seven out of eight. According to the study by [23], the best ratio of satisfied clause for any of MAXkSAT is between 0.7 and 1.

2.2. Discrete Hopfield Neural Network

The HNN developed by [6] is well known in artificial intelligence due to the content addressable memory (CAM) and energy minimization capability. The HNN stores patterns as a CAM systematically [23]. The HNN model consists of an interconnected unit called neurons that forms a network with no hidden layers. Hopfield network uses collections of interconnected neurons to carry out the computation based on [6]. The units are HNN represented in bipolar values of 1 and -1 [15]. The neuron is updated based on the given formulation below:

$$S_i = \begin{cases} 1, if \sum_j^n W_{ij} S_j \geq \theta_i \\ -1, otherwise \end{cases}, \tag{7}$$

where S_j is represents the state of unit j, W_{ij} is the synaptic weight from unit i to j, and θ is the threshold of unit i. There are two characteristics of synaptic weight. First, the synaptic weight of DHNN has no connection with itself $W_{ii} = W_{jj} = 0$ [28]. Secondly, the synaptic weight in this network is always symmetric $W_{ij} = W_{ji}$ which makes DHNN obtain the optimal solution by always converging to global minimum energy. $\theta = 0$ is to make sure the energy of DHNN reduces uniformly. RANMAX2SAT has been implemented in DHNN. In this case, RANMAX2SAT will consider at most two neurons per clause. DHNN has a useful CAM that can store and retrieve memory the way the human brain works. CAM can retrieve stored data with the presentation of partial information from that data. The main objective for implementing P_{RM2SAT} in DHNN is to minimize the cost functions to reduce the inconsistencies of logic of the network towards a minimum energy during the training phase. Generally, the cost function $E_{P_{RM2SAT}}$ is given by the following formula:

$$E_{P_{RM2SAT}} = \sum_{i=1}^{NC} \prod_{j=1}^{p+m+n} T_{ij}, \tag{8}$$

where NC is the number of clauses and $p + m + n$ are the number of variables in P_{RM2SAT}. The inconsistency of P_{RM2SAT} is:

$$T_{ij} = \begin{cases} \frac{1}{2}(1 - S_A)\,, if \neg A \\ \frac{1}{2}(1 + S_A)\,, otherwise \end{cases}, \tag{9}$$

The inconsistency of P_{RM2SAT} is defined by taking the negation of P_{RM2SAT}. Note that there is no consistent interpretation that results in $E_{P_{RM2SAT}} = 0$ because of the existence of redundant variables in the logical structure. Therefore, the network is shifted by determin-

ing the least value of cost function. In this work, Wan Abdullah's method is implemented to obtain the weights for DHNN-RANMAX2SAT based on logical inconsistencies [13]. Each neuron will be assigned by truth values. The minimized cost function is defined by finding the maximum number of satisfied clauses. During the testing phase, before finding the final state of neuron, the local field plays an important role in squashing the retrieved output. The state of the neuron will be updated asynchronously by the equation of local field h_i.

$$h_i(t) = \sum_{j=1, i\neq j}^{NC} W_{ij}^{(2)} S_j + W_i^{(1)} , \tag{10}$$

Local field determines the effectiveness of the final neuron states that are produced by DHNN. Then, the retrieved final states will be interpreted whether the final solution is over fit or not. Specifically, the updating equation is based on following equation:

$$S_i(t+1) = \begin{cases} 1 & , if \tanh(h_i) \geq 0 \\ -1 & , otherwise \end{cases} , \tag{11}$$

$$\tanh(h_i) = \frac{e^{h_i} - e^{-h_i}}{e^{h_i} + e^{-h_i}} , \tag{12}$$

where h_i is the local field of the network and $\tanh(h_i)$ implies the squashing function of Hyperbolic Activation function (HTAF). In the training phase, the cost function will be compared with the Lyapunov energy function of DHNN to obtain the synaptic weight. The equation of Lyapunov energy function is as follows:

$$H_{P_{RM2SAT}}(t) = -\frac{1}{2} \sum_{i=1, i\neq j}^{NC} \sum_{j=1, i\neq j}^{NC} W_{ij}^{(2)} S_i S_j - \sum_{i=1, i\neq j}^{NC} W_i^{(1)} S_i , \tag{13}$$

where $W_{ij}^{(2)}$ is the synaptic weight for second order clauses and $W_{ij}^{(1)}$ is the synaptic weight for first order clauses. The final energy that $H_{P_{RM2SAT}}$ at time $t+1$ is given as below:

$$H_{P_{RM2SAT}}(t+1) = -\frac{1}{2} \sum_{i=1, i\neq j}^{NC} \sum_{j=1, i\neq j}^{NC} W_{ij}^{(2)} S_i^u S_j - \sum_{i=1, i\neq j}^{NC} W_i^{(1)} S_i^u , \tag{14}$$

$H_{P_{RM2SAT}}(t+1)$ shows the energy after being updated by u that validates the final stated produced by training phase and retrieval phase. The differences in energy level are expressed as follows:

$$\Delta H_{P_{RM2SAT}} = H_{P_{RM2SAT}}(t) - H_{P_{RM2SAT}}(t+1) \tag{15}$$

By substituting Equations (13) and (14) in (15),

$$\Delta H_{P_{RM2SAT}} = -\frac{1}{2}(S_j - S_j^u) \left(\sum_{i=1, i\neq j}^{NC} W_{ij}^{(2)} S_i S_j - W_i^{(1)}) \right) , \tag{16}$$

Thus, simplified version of Equation (15) is as follows:

$$\Delta H_{P_{RM2SAT}} = -\frac{1}{2}(S_j - S_j^u)(h_i(t)) , \tag{17}$$

According to Equation (15), the similar states imply an optimized final state whereby $\Delta H_{P_{RM2SAT}} = 0$. Therefore, this equilibrium proves that DHNN will always converge to

a stable state. The minimum values from the energy function can be defined as the stable state of the neurons [30]. The formula to calculate $H^{\min}_{P_{RM2SAT}}$ is given by:

$$H^{\min}_{P_{RM2SAT}} = -\left(\frac{\theta + 2\eta}{4}\right), \tag{18}$$

where $\theta = n(C_i^{(2)})$ and $\eta = n(C_i^{(1)})$ that corresponds to P_{RM2SAT}.

After completing the training phase of DHNN-RANMAX2SAT, synaptic weight obtained in the training phase will be used in the testing phase. Note that network relaxation will help the network to get the correct final state. During firing and receiving information, more interconnected neurons are involved when the number of neurons increased. Note that, inefficiency of the relaxation mechanism produces more local minima solution. To be precise, to guarantee the network comes to a relaxation stage where it reaches a stable state, the neuron will be updated based on the Sathasivam Relaxation method [30]. The exchange of information between neurons will be computed by the given formula:

$$\frac{dh_i^{\text{new}}}{dt} = R\frac{dh_i}{dt}, \tag{19}$$

where h_i refers to the local field of the network and R denotes the relaxation rate. In this network, the relaxation rate used in our program is $R = 3$. The quality of the final neuron state is determined by using the following equation:

$$\left| H_{P_{RM2SAT}} - H^{\min}_{P_{RM2SAT}} \right| \leq tol\ , \tag{20}$$

where tol is the tolerance value. The final state of the neuron will be trapped in a local minima if Equation (16) is not satisfied. The energy function is crucial because the degree of convergence of the network is determined by the energy function. The energy value produced will be classified either as global minimum energy or local minimum energy. Figure 1 shows the schematic diagram for DHNN-RANMAX2SAT. Note that the blue dotted lines represent the second-order clauses of redundant variables and green dotted lines represent the second-order clauses of non-redundant variables with four possibilities of neurons in the second-order clauses. Meanwhile, the red dotted lines represent first-order clauses with two possibilities of neurons. Algorithm 1 illustrates the steps DHNN-RANMAX2SAT through the pseudocode below.

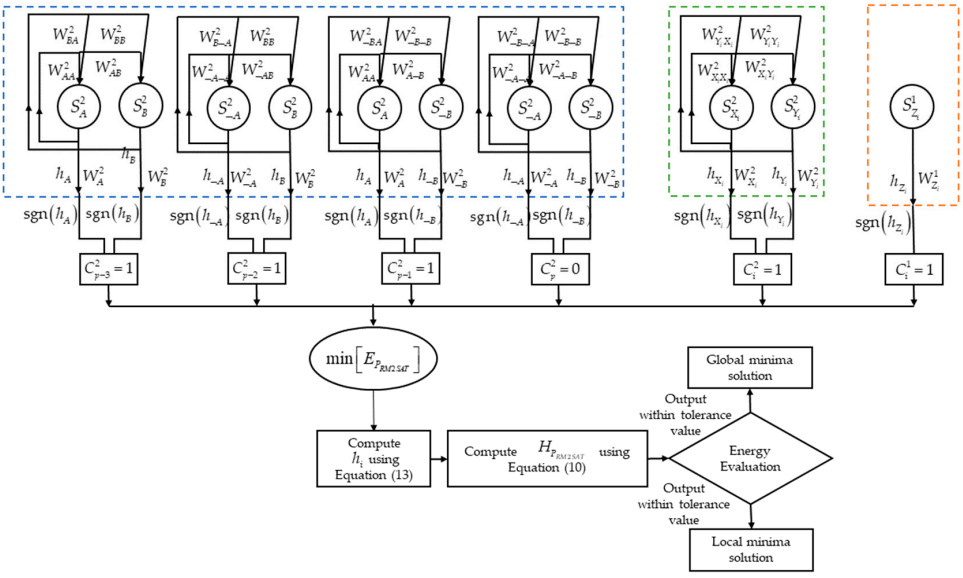


Figure 1. Schematic diagram for DHNN-RANMAX2SAT.

Algorithm 1: The pseudocode of DHNN-RANMAX2SAT

Start
Create the initial parameters, $COMBMAX$, trial number, relaxation rate, tolerance value;
Initialize the neuron to each variable consisting of $S_i \in [S_1, S_2, S_3, \ldots S_n]$;
while $(i \leq trial)$;
Form initial states by using Equation (7);
[TRAINING PHASE]
Apply Equation (8) to define cost function $E_{P_{RM2SAT}}$;
for $S_i \in [S_1, S_2, S_3, \ldots S_n]$ **do**
Apply Equation (8) to check clauses satisfaction of clauses;
if $E_{P_{RM2SAT}} = 0$
$S_i \rightarrow$ Satisfied;
Else
$S_i \rightarrow$ Unsatisfied;
end
Apply WA method to calculate synaptic weights;
Apply Equation (13) to compute the $H_{P_{RM2SAT}}$;
[TESTING PHASE]
Compute final state by using local field computation, Equation (10);
for $(i \leq trial)$;
End
Apply Equation (18) to compute the $H^{\min}_{P_{RM2SAT}}$;
Compute the final energy;
Assign global minimum energy or local minimum energy
if $|H_{P_{RM2SAT}} - H^{\min}_{P_{RM2SAT}}| \leq tol$
Allocate Global minimum energy;
Else
Allocate Local minimum energy;
end
return Output the final neuron state;

2.3. Genetic Algorithm

A Genetic Algorithm (GA) is an evolutionary algorithm inspired by the biological evolution process of natural selection, crossover, and mutation. According to [31], GA based on Darwin's evolutionary theory is to find an optimal solution. Initially, GA will randomly produce a population of chromosomes. The chromosomes represent the possible solutions of the optimization problem. Next, to evaluate the quality of each chromosome, the fitness function of these chromosomes will be calculated based on Equation (21).

$$f_{RM2SATGA} = \sum_{i=1}^{p+m} C_i^{(2)} + \sum_{i=1}^{n} C_i^{(1)}, \tag{21}$$

where $C_i^{(2)}$ is second order clauses RANMAX2SAT clauses and $C_i^{(1)}$ is the first order RANMAX2SAT clause and given as follows:

$$C_i^{(2)} = \begin{cases} 1, & Satisfied \\ 0, & otherwise \end{cases}, \tag{22}$$

$$C_i^{(1)} = \begin{cases} 1, & Satisfied \\ 0, & otherwise \end{cases}, \tag{23}$$

The objective function of proposed GA is to maximize the fitness of the S_i:

$$\max[f_{RM2SATGA}] \tag{24}$$

GA also has the ability to deal with a higher number of variables in the logical structure without affecting the overall computation. Previous work by [28] used GA in the training

phase of DHNN with the logical interpretation of RANkSAT structure being similar to the chromosomes of DNA. Inspired by this fascinating work, in this proposed study, GA will be implemented in the training phase of P_{RM2SAT}. The implementation of GA in DHNN of this proposed model is abbreviated as DHNN-RANMAX2SATGA. The involved stages in DHNN-RANMAX2SATGA are as follows.

2.3.1. Initialization

N_{PopGA} chromosome S_i where $S_i \in \{S_1, S_2, S_3, \ldots S_{N_{PopGA}}\}$ initialized. In each S_i, the state of neuron is represented by 1 (TRUE) and -1 (FALSE).

2.3.2. Fitness Evaluation

All the bit string will undergo fitness evaluation based on Equation (21). If $f_{RM2SATGA}$ reaches maximum fitness, the algorithm will be terminated.

2.3.3. Selection

N_S chromosomes with the highest fitness are selected from the randomized chromosomes by selection rate, λ.

$$N_S = \lambda N_{PopGA}\ , \tag{25}$$

where λ is ranging to $\lambda = [0, 1]$. This stage is vital because a chromosome with a low value of $f_{RM2SATGA}$ will not proceed to next stage.

2.3.4. Crossover

Two chromosomes are selected and separated during the crossover phase from selected chromosomes. The information between two sub-structures of the bit strings will be exchanged based on a crossover rate.

2.3.5. Mutation

Mutation involves flipping the state of the bit string from 1 to -1 or -1 to 1. However, if the wrong state is flipped during this stage, there is a chance for the fitness value to decrease. The current chromosomes will repeat the Sections 2.3.1–2.3.5 if the fitness value does not achieve maximum fitness.

Algorithm 2 below illustrates the process of Genetic Algorithm through pseudocode.

Algorithm 2: The pseudocode of GA in the training phase

Start
Create initial parameters including chromosomes population size N_{PopGA} consisting of $S_i \in \{S_1, S_2, S_3, \ldots S_{N_{PopGA}}\}$, trial number;
while $i \leq trial$
  Initialize $N_{PopGA} - N_S$ random S_i;
  [Selection]
  for $i \in \{1, 2, 3, \ldots, N_{PopGA}\}$ **do**
    Apply Equation (21) to compute the fitness of each S_i;
    Apply Equation (25) to compute N_S;
  end
  [Crossover]
  for $S_i \in \{1, 2, 3, \ldots, N_S\}$ **do**
    The states of the selected two S_i exchanged at a random point;
  End
  [Mutation]
  for $S_i \in \{1, 2, 3, \ldots, N_S\}$ **do**
    Flipping states of S_i at random location;
    Evaluate the fitness of the S_i based on Equation (21);
  End
end while
return Output the final neuron state.

2.4. Election Algorithm

EA is an iterative algorithm inspired by the social-politic mechanism of presidential elections [27]. Ref. [28] said that EA works with a set of population called solution where each individual is either a candidate or a voter. EA is a metaheuristic that is used to obtain the optimal solution in the logic that can minimize the cost function during the training phase of the DHNN. This training algorithm consist of stage that partitions in a solution space. Each partition will be assigned as party and organized by a candidate [29]. An illustrative example of EA in doing RANMAX2SAT will be explained in Appendix A. In general, the eligibility value for the candidate, L_i is as follows:

$$f_{RM2SATEA} = \sum_{i=1}^{p+m} C_i^{(2)} + \sum_{i=1}^{n} C_i^{(1)}, \tag{26}$$

where $C_i^{(2)}$ and $C_i^{(1)}$ are the second and first order RANMAX2SAT clause, respectively and were given as

$$C_i^{(2)} = \begin{cases} 1, & Satisfied \\ 0, & otherwise \end{cases}, \tag{27}$$

$$C_i^{(1)} = \begin{cases} 1, & Satisfied \\ 0, & otherwise \end{cases}, \tag{28}$$

The objective function of proposed EA is to obtain the maximum fitness of the S_i:

$$\max[f_{RM2SATEA}] \tag{29}$$

Note that EA possessed few operators that represent the real presidential election process. The operators are explained in Sections 2.4.1–2.4.5.

2.4.1. Initializing Population and Forming Initial Parties

The algorithm will start by initializing a random population N_{PopEA} that comprises of voters and candidates. Each individual is represented by $S_i \in \{S_1, S_2, S_3, \ldots S_{N_{PopEA}}\}$ where $S_i = \{-1, 1\}$. The partitioning of solution space into number of parties j occurs based on the given equation:

$$N_j = \frac{N_{PopEA}}{N_{Party}}, \text{where } j = 1, 2, 3, 4 \tag{30}$$

where N_{Party} is the number of parties j. The highest eligibility value will be selected as candidate, L_j of party j. The rest of the individuals will be represented as voters v_i^j that support the candidate. The similarity of belief between candidate, L_j, and the voter, v_i^j, is in the term of distance and computed by using equation below:

$$dist(f_{L_j}, f_{v_i^j}) = f_{L_j} - f_{v_i^j}, \tag{31}$$

where f_{L_j} and $f_{v_i^j}$ are the eligibility of the candidate and voters, respectively.

2.4.2. Positive Advertisement

In this stage, the candidate in each party tries to gain more support from the voter in their party by sharing their agendas and ideas. The number of voters that will be influenced by the candidate is as follows:

$$N_S = \sigma^P N_j, \text{ where } \sigma^p \in [0, 0.5] \tag{32}$$

where σ^p is the positive advertisement rate. The reasonable effect from the candidate to the voter is defined as the eligibility distance coefficient given by:

$$\omega_{v_i^j} = \frac{1}{dist(f_{L_j}, f_{v_i^j}) + 1}, \tag{33}$$

The voters then will undergo state flipping based on the following equation

$$S_{v_i^j} = NC\omega_{v_i^j}, \tag{34}$$

where NC is total number of clauses. The fitness of the voters is then updated by Equation (22). The candidate will be replaced if the voters have higher fitness than the candidate.

2.4.3. Negative Advertisement

The candidate will try to gain popularity from outside voters. The number of voters that is influenced is represented by following equation:

$$N_{v_i^*} = \sigma^n (N_j - N_s) \text{ where } \sigma^n \in [0, 0.5] \tag{35}$$

where σ^n is the negative advertisement rate, v_i^* is the voters from the other parties. The equation of similarity of belief between the candidate and voter as below:

$$dist(f_{L_j}, f_{v_i^*}) = f_{L_j} - f_{v_i^*}, \tag{36}$$

The reasonable effect from the candidate to the voter from other party is the eligibility distance coefficient stated in the equation below:

$$\omega_{v_i^*} = \frac{1}{dist(f_{L_j}, f_{v_i^*}) + 1}, \tag{37}$$

$$S_{v_i^*} = NC\omega_{v_i^*}, \tag{38}$$

where $NC = p + m + n$ is sum of second and first order clauses. The eligibility of voters is then updated by Equation (26) and the voters will replace the candidate if they have highest fitness.

2.4.4. Coalition

Candidates will form allies with individual (both candidate and voters) from other parties. The parties will collaborate dependently with each other. The effect from both candidates from both parties to the new coalition party is calculated by Equations (37) and (38). The state of the voters and candidate is then updated by Equation (26).

2.4.5. Election Day

Election day will happen when the condition of termination is met in Sections 2.4.3–2.4.5. In this stage, the final eligibility of all candidates will be evaluated. If the candidate L_j has the maximum fitness, the candidate will be elected as the solution. If not, Sections 2.4.3–2.4.5 will be repeated until a certain number of iterations that has been decided on.

Algorithm 3 below illustrates the process of Election Algorithm through pseudocode.

Algorithm 3: The pseudocode of EA in the training phase

Start
Create the initial parameters that includes the population size N_{PopEA} consisting of $S_i \in \{S_1, S_2, S_3, \ldots S_{N_{PopEA}}\}$, trial number;
while $i \leq trial$
  Forming initial parties by using Equation (30);
  for $J \in \{1, 2, 3, \ldots, N_{Party}\}$ **do**
    Apply Equation (31) to compute the similarity between the voters and the candidates
  end
  [Positive Advertisement]
  For $S_i \in \{1, 2, 3, \ldots N_{S_i}\}$ **do**
    Apply Equation (32) to evaluate the number of voters;
    Calculate the reasonable effect from the candidate $\omega_{v_i^j}$ by using Equation (33);
    Update the neuron state according to Equation (26);
    if $f_{v_i^j} > f_{L_j}$;
      Assign v_i^j as new L_j;
    Else
      Remain L_j;
  end
  [Negative Advertisement]
  for $S_i \in \{1, 2, 3, \ldots N_{v_j^*}\}$ **do**
    Calculate the similarity between the voters from the other party and the candidate from Equation (36);
    Compute the reasonable effect from the candidate $\omega_{v_i^*}$ and update the neuron state by using Equation (37);
    if $f_{v_j^*} > f_{L_j}$;
      Assign v_i^* as new L_j;
    else
      Remain L_j;
  end
  [Coalition]
  for $S_i \in \{1, 2, 3, \ldots N_{v_j^*}\}$ **do**
    Calculate the similarity between the voters from the other party and the candidate from Equation (36);
    Compute the reasonable effect from the candidate $\omega_{v_i^*}$ and update the neuron state by using Equation (37);
    If $f_{v_j^*} > f_{L_j}$;
      Assign v_i^* as new L_j;
    Else
      Remain L_j;
  End
end while
return Output the final neuron state;

3. Methodology

Figure 2 illustrates the general flow of the proposed study to ensure the readers gain a better understanding of this approach. In the training phase, training algorithms such as ES, GA, and EA will be implemented to Random Maximum 2 Satisfiability in DHNN to ensure the correct synaptic weight is obtained. ES operates based on random search to find the solution. However, GA and EA have optimization operators that will help in improving the solution in every iteration. Next, the computation of the local field of the neuron state and the final energy occurs in the testing phase. The differences between final energy and minimum energy are checked whether in the range of tolerance value in order to verify the final energy of the network. The final energy of the proposed model is considered a global minima solution if the differences are less than the tolerance value. Otherwise, it is trapped in the local minima solution.

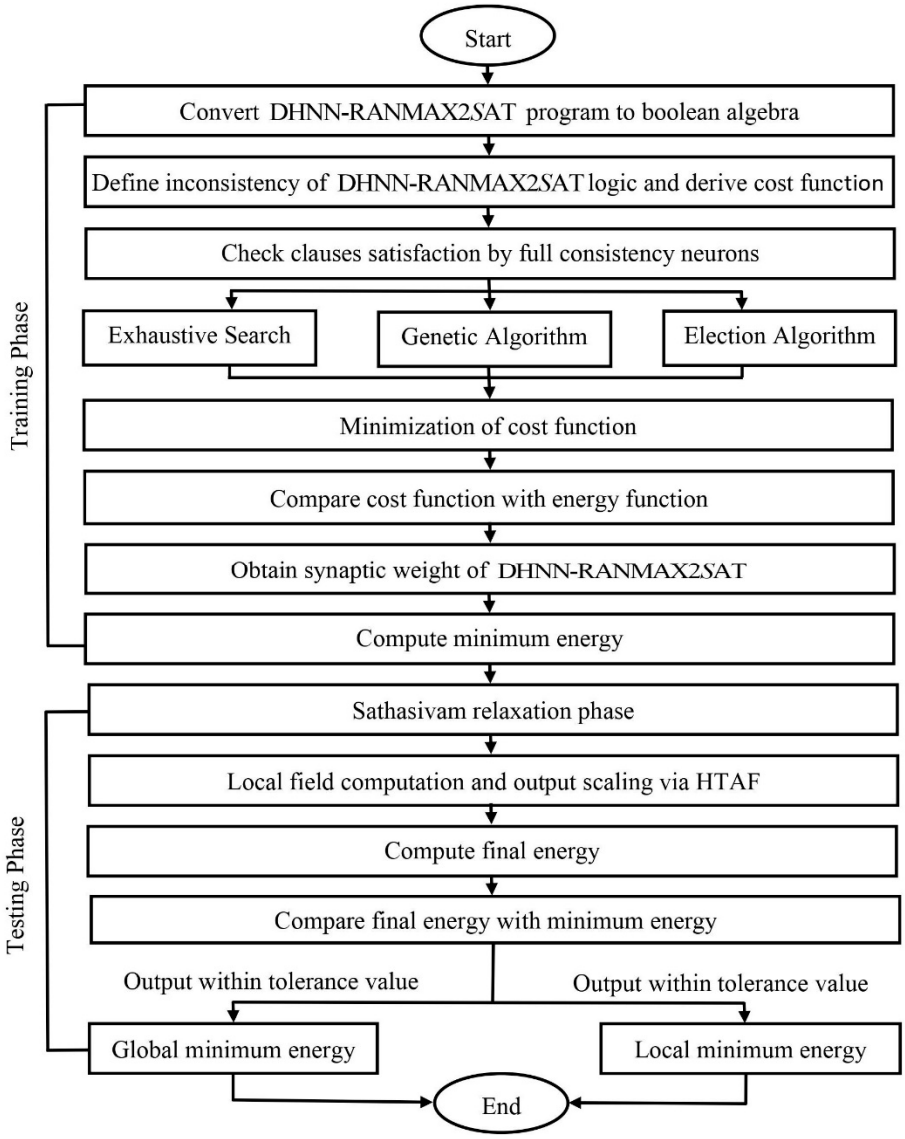


Figure 2. General flow of the proposed study.

3.1. Performance Metrices

In this section, the performance of DHNN-RANMAX2SATES, DHNN-RANMAX2SATGA, and DHNN-RANMAX2SATEA will be examined by various performance metrics. To examine the actual performance of the network, these performance metrics have been used by several researchers in neural network studies [18–21]. The purpose of the program is to obtain the best training model of DHNN-RANMAX2SAT.

3.1.1. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE)

RMSE and MAE measure the distance between the predicted value and observes the value of a model. RMSE and MAE are used to measure the accuracy of performance. In general, RMSE represents the data standard deviation of the differences between the target value and observed value. RMSE can be expressed as [19]:

$$\text{RMSE} = \sum_{i=1}^{\varepsilon} \sqrt{\frac{1}{\varepsilon}(P_i - O_i)^2}, \tag{39}$$

where P_i is the predicted value and O_i is the observed value. In this dissertation, RMSE for training error can be formulated as [19]:

$$\text{RMSE Training} = \sum_{i=1}^{\varepsilon} \sqrt{\frac{1}{\varepsilon}(f_{\max} - f_i)^2}\,, \tag{40}$$

where $f_{\max}$ is the total number of DHNN-RANMAX2SAT clause, f_i is the fitness of P_{RM2SAT} computed by the network and ε is the number of iterations before $f_i = f_{\max}$. RMSE for testing error is expressed by

$$\text{RMSE Testing} = \sum_{i=1}^{\varepsilon} \sqrt{\frac{1}{ab}(G_{P_{RM2SAT}} - L_{P_{RM2SAT}})^2}\,, \tag{41}$$

where $G_{P_{RM2SAT}}$ is the number of global minimum solution and $L_{P_{RM2SAT}}$ is the number of local minimum solution. a is number of combinations and b is number of trials. MAE is defined as the average absolute difference between the predicted value and observed value. The formula of MAE is given by [17]:

$$\text{MAE} = \sum_{i=1}^{\varepsilon} \frac{1}{\varepsilon}\left|P_i - O_i\right|\,, \tag{42}$$

The MAE training and testing used in this paper are

$$\text{MAE Training} = \sum_{i=1}^{\varepsilon} \frac{1}{\varepsilon}\left|f_{\max} - f_i\right|^2\,, \tag{43}$$

$$\text{MAE Testing} = \sum_{i=1}^{\varepsilon} \frac{1}{ab}\left|G_{P_{RM2SAT}} - L_{P_{RM2SAT}}\right|\,. \tag{44}$$

3.1.2. Mean Absolute Percentage Error (MAPE)

MAPE measures the size of the error in percentage. The formula of MAPE can be computed as [19]

$$\text{MAPE} = \sum_{i=1}^{\varepsilon} \frac{100}{\varepsilon}\frac{|P_i - O_i|}{|O_i|}\,, \tag{45}$$

MAPE formula for training error is given as:

$$\text{MAPE Training} = \sum_{i=1}^{\varepsilon} \frac{100}{\varepsilon}\frac{|f_{\max} - f_i|}{|f_i|}\,, \tag{46}$$

and for testing error, the formula of MAPE is

$$\text{MAPE Testing} = \sum_{i=1}^{\varepsilon} \frac{100}{\varepsilon}\frac{|G_{P_{RM2SAT}} - L_{P_{RM2SAT}}|}{|ab|}\,, \tag{47}$$

3.1.3. Global Minimum Ratio (Zm)

Global minimum ratio measures the ratio between total global minimum energy and the total number of runs. Global minimum energy can be obtained if the final energy is within the tolerance value [30]. The value of Zm can be obtained by the following formula [21]:

$$Zm = \frac{1}{ab}\sum_{i=1}^{\varepsilon} G_{P_{RM2SAT}} \tag{48}$$

3.1.4. Jaccard Similarity Index (JSI)

The final neuron state retrieved by the DHNN-RANMAX2SATES, DHNN-RANMAX2SATGA, and DHNN-RANMAX2SATEA models will be analyzed by using a similarity metric. The similarity metric that will be chosen in this paper is Jaccard similarity index utilized in [15]. The Jaccard similarity index is ratio of the similarity between two distinct datapoints. It also has been used in global evaluation. The Jaccard index for DHNN is as follows:

$$\text{JSI} = \frac{l}{l+m+n} \tag{49}$$

where

l is the number of $(f_{\max}, f_i)$ where both elements have the value of 1
m is the number of $(f_{\max}, f_i)$ where $f_{\max}$ is 1 and f_i is -1
n is the number of $(f_{\max}, f_i)$ where $f_{\max}$ is -1 and f_i is 1.

3.2. Baseline Methods

Note that this paper uses simulated data that generate randomly by computer program. The DHNN model is compatible to binary and bipolar representations. This paper utilizing bipolar representation in terms of logical structure that corresponds 1 defines as true and -1 defines as false. Moreover, bipolar neuron states used to evaluate the asynchronous neuron update in the DHNN model [30]. Furthermore, the bipolar representation converges faster than binary representations. This paper does not consider the binary structure that consists of 0 and 1 because the value of 0 that exists in the binary structure can eliminate important parameters. The use of bipolar and binary representative can be differentiated in the computation of finding synaptic weight. The 0 value in binary representation will lead to wrong synaptic weight or delete the synaptic weight. Thus, this helps the proposed model to converge faster.

The effective relaxation method and the activation function in DHNN can improve the stable final state of the neurons. In this paper, the Sathasivam relaxation method is used to retrieve correct neuron states and improve the proposed model. This is because the Sathasivam relaxation method helps neurons to hold or pause before resuming in exchange for information. This method also helps to reduce neuron oscillation and increases the efficiency of the network in finding stable neuron states. Earlier, the conventional model was Wan Abdullah's logic programming based on McCulloch–Pitts function. However, [30] stated that the results of McCulloch–Pitts function retrieve more local minimum energy and it consumes more time to retrieve global minimum energy. Therefore, hyperbolic tangent activation function (HTAF) is proved to be most stable activation function compared to McCulloch–Pitts and Elliot Symmetric Activation Function in [32]. Hence, HTAF was chosen because of its capability to train squashing neuron states before being classified into final neuron states in this study. Table 2 shows the parameters used for DHNN-RANMAX2SAT, Tables 3 and 4 show the parameters used in this study.

Table 2. List of parameters used in DHNN model.

Parameter	Parameter Value
Tolerance value (tol)	0.001 [28]
Number of combinations (a)	100 [28]
Number of learnings (ε)	10,000
Number of trials (b)	100 [28]
Order of clauses (k)	1,2
Number of neurons (NN)	(50, 300)
Threshold time simulation	24 h
Relaxation rate (R)	3 [30]
Activation function	Hyperbolic Tangent Activation Function (HTAF) [32]

Table 3. List of GA parameters used in training phase.

Parameter	Parameter Value
Number of generations	100
Selection rate	0.1 [19]
Crossover rate	1 [19]
Mutation rate	0.01 [19]
Type of selection	Random

Table 4. List of EA parameters used in training phase.

Parameter	Parameter Value
Number of populations	120 [28]
Number of parties	4 [28]
Positive advertisement rate	0.5 [27]
Negative advertisement rate	0.5 [27]
Candidate selection	Highest fitness
Type of voter's attraction	Random
Type of state flipping	Random
Number of strings on election day	2

3.3. Experimental Design

The proposed hybrid networks during training phase and testing phase of DHNN-RANMAX2SAT are DHNN-RANMAX2SATES, DHNN-RANMAX2SATEA, and DHNN-RANMAX2SATGA. All proposed DHNN-RANMAX2SAT models will be implemented in Dev C++ Version 5.11 coding software with a specification of a 3.1 GHz Intel Core i5 processor with 4 GB RAM in the Windows 10 operating system. The simulation will be carried out in only one device to avoid biases. The output is run by Dev C++ coding software and the graph is illustrated by MATLAB.

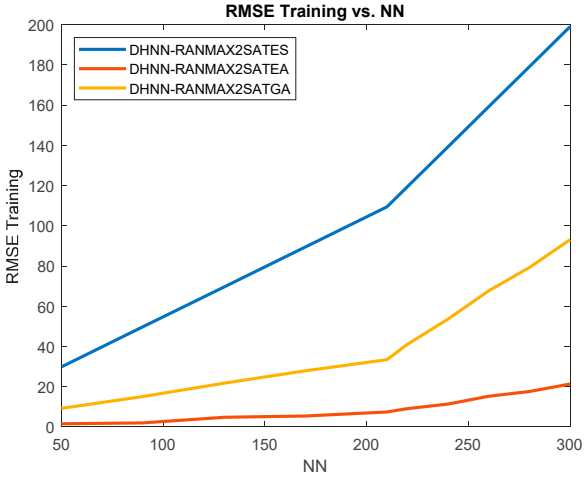
4. Results and Discussion

In this section, the performance of the three models, DHNN-RANMAX2SATES, DHNN-RANMAX2SATEA, and DHNN-RANMAX2SATGA, will be discussed. Note that the data will be divided into two phases. The first data phase will define the increment of 2SAT of non-redundant variables in Random Maximum 2 Satisfiability where $50 \leq NN \leq 210$ and the second data phase will define the increment of 1SAT of non-redundant variables in Random Maximum 2 Satisfiability where $210 < NN \leq 300$. This is important in order to observe the behavior of the 1SAT and 2SAT in Random Maximum 2 Satisfiability. In both phases, the clauses with two redundant variables are included to represent the maximum satisfiable part. The number of neurons is limited to 300 due to threshold time stimulation that is fixed to 24 h. An exhaustive search took more than 24 h in the stimulation for number of neurons more than 300. Therefore, number of neurons is limited to 300 for the Genetic Algorithm and Election Algorithm based on proposed logical structure to maintain the parallelism and produce comparable results. The result will be discussed according to training error, testing error, energy analysis, and similarity analysis.

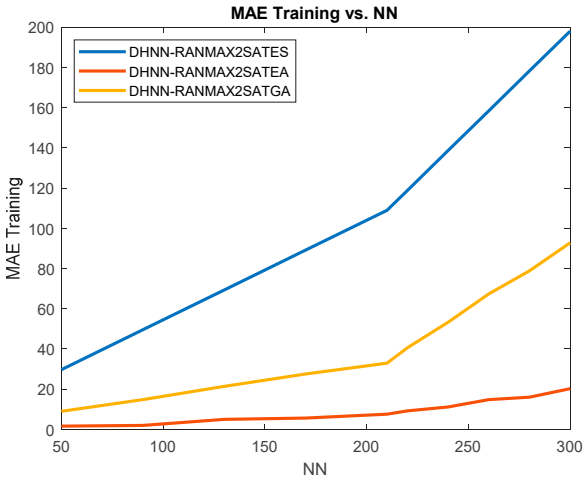
4.1. Training Error

ES, GA, and EA facilitate the training phase to check clause satisfaction. Algorithm approaches are utilized in this study to alter the parameters of the machine learning model and get an optimized solution. Thus, GA and EA play an important role in explainable artificial intelligence. The synaptic weight management by the proposed models is observed for all logical combinations of P_{RM2SAT} in this section. The optimal training phase is defined as the capability of the proposed model to minimize the cost function that can generate the optimal synaptic weight according to the Wan Abdullah (WA) method [13]. In order to

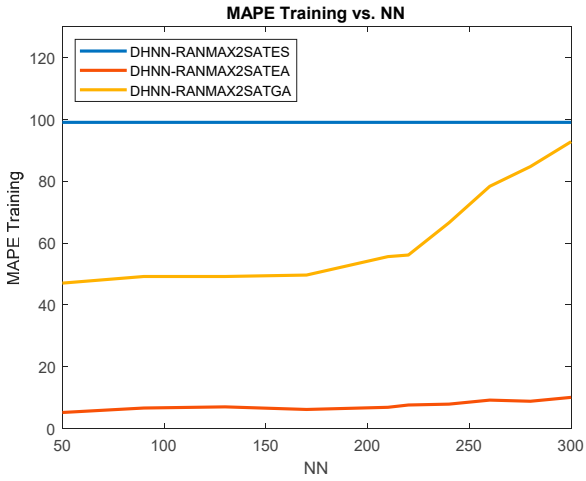
achieve minimized cost function, in this study, ES, GA, and EA will be implemented and compared. Figure 3 below illustrates the RMSE training, MAE training, and MAPE training of the proposed model.



(**a**)



(**b**)



(**c**)

Figure 3. (**a**) RMSE training for DHNN-RANMAX2SAT; (**b**) MAE training for DHNN-RANMAX2SAT; (**c**) MAPE training for DHNN-RANMAX2SAT.

(i) Observe that, as the total number of neurons increases, the value of RMSE training, MAE training, and MAPE training increases for DHNN-RANMAX2SATES, DHNN-RANMAX2SAEA, and DHNN-RANMAX2SAGA. We can also observe that there is a drastic increase in phase 2 compared to phase 1 in the graphs. This is due to the non-systematic logical structure that consists of first order clauses in phase 2 having the chances of getting a satisfied interpretation being low compared to second order clauses in phase 1 which makes the graph increase. Therefore, overall, when the number of neurons increases, the number of satisfied interpretations decrease which makes the training errors increase due to the complexity of the logical structure [16].

(ii) According to Figure 3, it is noticeable that the highest training error is shown at $NN = 300$ by DHNN-RANMAX2SATES compared to DHNN-RANMAX2SATEA and DHNN-RANMAX2SATGA. This is due to less stability of the neurons during the training phase and ES derives the wrong synaptic weight. Since ES is operated by a random search method, the complexity to get correct synaptic weight will increase as the number of neurons increases.

(iii) Observe that as the number of neurons increases, DHNN-RANMAX2SATGA manages to achieve low training error compared to DHNN-RANMAX2SATES. Note that the operator of crossover with crossover rate of 1 in the Genetic Algorithm is able to change the fitness of the population frequently by using the fitness function [19]. Moreover, the mutation rate of 0.01 based on [19] is able to obtain the optimum fitness. Therefore, it is easy for the chromosomes to achieve an optimal cost function to retrieve the correct synaptic weight.

(iv) However, based on the graph above, DHNN-RANMAX2SATEA outperformed DHNN-RANMAX2SATES and DHNN-RANMAX2SATGA as the number of neurons increased. Lower training error indicates the better accuracy of our model. This is due to proposed metaheuristic in which EA enhanced the training phase of DHNN. DHNN-RANMAX2SAEA is efficient in retrieving global minimum energy due to the global search and local search operators of EA [27]. This indicates that the optimization operators in EA enhanced the training phase of DHNN-RANMAX2SATEA. The highest rate of positive advertisement and negative advertisement chosen based on [27] that quickens the process of obtaining the candidate with maximum fitness. By diving the solution spaces during training phase, the synaptic weight management improved and the proposed model achieves the optimal training phase successfully.

4.2. Testing Error

An optimal testing phase is when the proposed model manages to retrieve the final neuron state that produces a global minimum solution. Good synaptic weight management of the proposed model will result in obtaining a global minimum solution. Therefore, the main focus on analyzing the testing error is to observe the quality of the solution whether the final neuron state produces the global minimum or local minimum solution by Equation (20). Figure 4 demonstrates the performance of DHNN-RANMAX2SATES, DHNN-RANMAX2SATEA, and DHNN-RANMAX2SATEA during the testing phase.

(i) According to Figure 4, the graphs show similar trends for DHNN-RANMAX2SATES, DHNN-RANMAX2SATEA, and DHNN-RANMAX2SATGA gives a constant graph for both phase 1 and phase 2 as the number of neurons increases. This is due to the logical structure becoming more complex as it contains a greater number of the neuron. In this case, as the number of neurons increases, the logical structure fails to retrieve more final states that lead to global minimum energy.

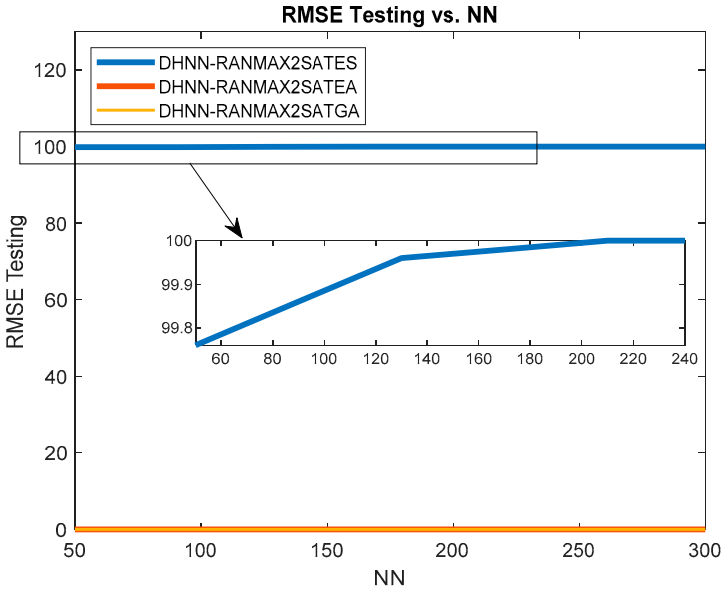
(ii) Based on Figure 4, RMSE testing, MAE testing, and MAPE testing of DHNN-RANMAX2SATES increases at $50 \leq NN \leq 210$. ES is a searching algorithm. The training phase could be affected by the nature of ES which will continuously affect the testing phase, thus resulting in a high value of testing error. Wrong synaptic weights retrieved during the testing phase due to the inefficiency of synaptic weight management. The complexity of the network increases when $NN > 210$ resulting in a constant

graph with maximum values of RMSE testing, MAE testing, and MAPE testing. Thus, the DHNN-RANMAX2SATES model starts retrieving the non-optimal states.

(iii) According to the graphs, the accumulated errors are mostly 0 for DHNN-RANMAX2SATGA for RMSE testing, MAE testing, and MAPE testing. This is due to metaheuristics GA consisting of an optimization operator which can help to improve the solution. It can be deduced that GA barely gets trapped in the local minima solutions. The operators of GA always search for optimal solutions which correspond to the global minimum energy. Moreover, mutation operator in GA reduced the chances for the bit string to retrieve local minima solutions. Thus, this resulted in a zero value of RMSE testing, MAE testing, and MAPE testing as the number of neurons increases.

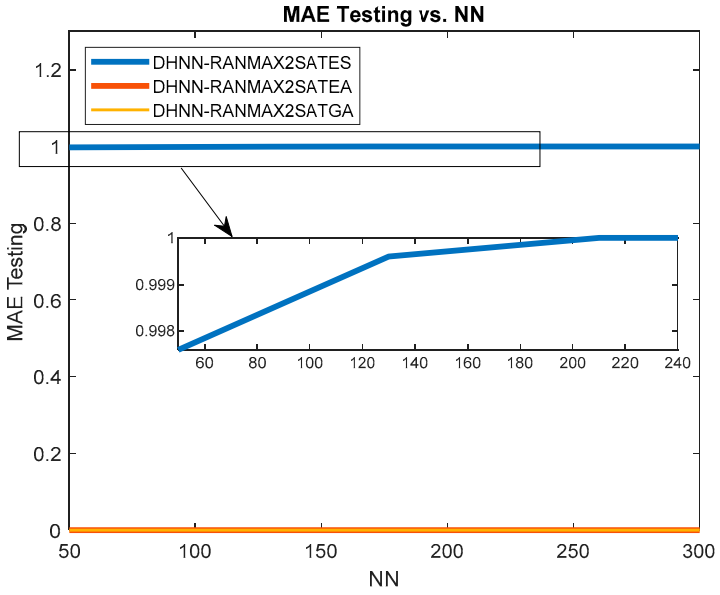
(iv) Notice that the graphs of RMSE testing, MAE testing, and MAPE testing of DHNN-RANMAX2SATEA also show a constant graph that achieves zero testing error as the number of neurons increases. Lower errors of RMSE testing, MAE testing, and MAPE testing define the effectiveness of proposed model to generates more global minimum energy. This is due to the effective synaptic weight management during training phase of DHNN-RANMAX2SATEA. The presence of local search and global search operator in EA that divides the solution spaces during training phase is the main reason that improves the synaptic weight management during retrieval phase. This leads DHNN-RANMAX2SATEA to produce global minimum energy in the testing phase.

(v) Generally, we can say that DHNN-RANMAX2SATEA and DHNN-RANMAX2SATGA outperformed DHNN-RANMAX2SATES in terms of RMSE testing, MAE testing, and MAPE testing. This indicates that ES failed to retrieve optimal synaptic weight during the training phase and consequently affected the testing phase, thus resulting in local minima solution. Meanwhile, GA and EA find more variation of the solution (more global solution). Therefore, DHNN-RANMAX2SATEA and DHNN-RANMAX2SATGA help the network to reduce generating local minimum energy by achieving zero for RMSE testing, MAE testing, and MAPE testing.

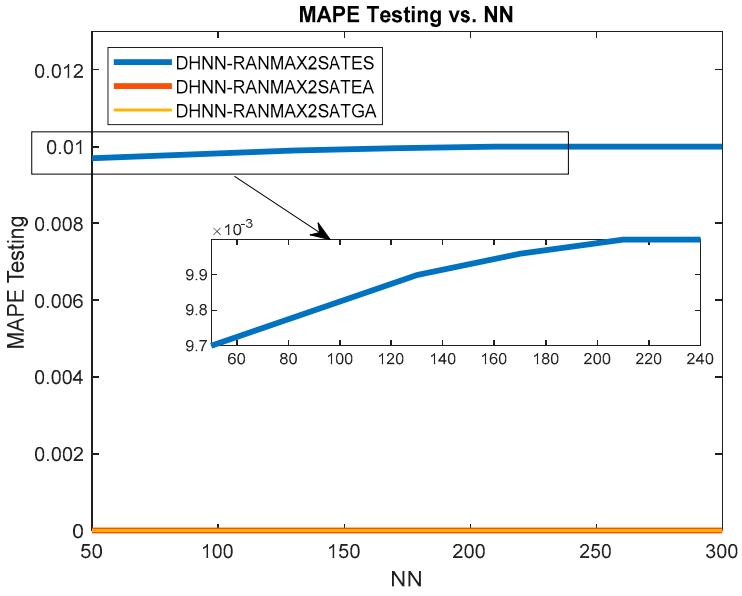


(a)

Figure 4. *Cont.*



(**b**)



(**c**)

Figure 4. (**a**) RMSE testing for DHNN-RANMAX2SAT; (**b**) MAE testing for DHNN-RANMAX2SAT; (**c**) MAPE testing for DHNN-RANMAX2SAT.

4.3. Energy Analysis (Global Minimum Ratio)

The Global Minimum Ratio (*Zm*) produced by DHNN-RANMAX2SATES, DHNN-RANMAX2SATEA, and DHNN-RANMAX2SATGA during the retrieval phase is shown in Figure 5. The amount of global minimum energy produced by the network can determine the efficiency of network. Therefore, if the global minimum ratio of the proposed network is close to 1, most of the solutions in the network reached correct final state during the retrieval phase. In the network, 10,000 bit strings solutions will be produced by each stimulation. For example, 0.9981 global minima ratio value defines 9981 bit strings are global minimum energy and only 19 bit strings are local minimum energy. In [20], it was discussed how the energy produced at the end of the process correlates with the global minima ratio.

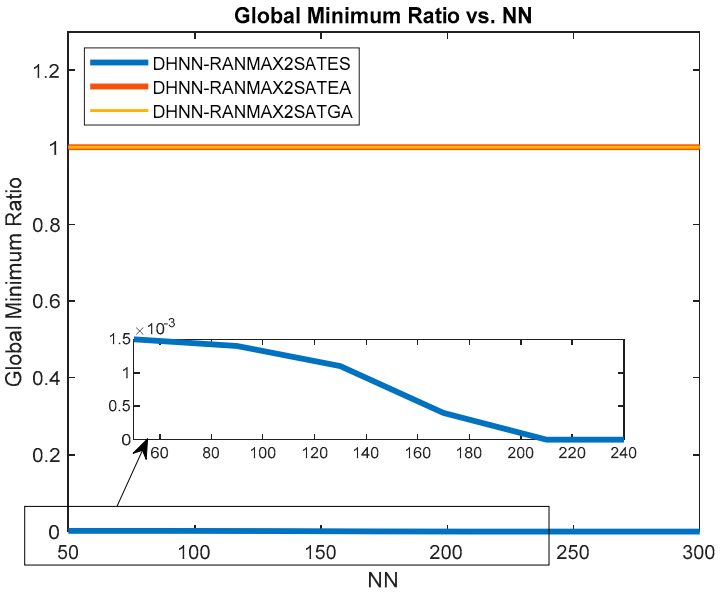


Figure 5. Global minimum ratio, Zm produced by DHNN-RANMAX2SAT.

(i) According to the graph, DHNN-RANMAX2SATES shows a decrease in the graph when $50 \leq NN \leq 210$ with Zm almost 0. At this stage, ES is only able to produce much less global minimum energy because most of the solutions are trapped at sub-optimal states. When the number of neurons increases in ES, the network becomes more complex. Thus, the local field is not able to generate the correct state of the neuron as the number of neuron increases. Hence, we can observe a constant graph at $Zm = 0$ when $NN > 210$.

(ii) However, DHNN-RANMAX2SATGA manages to achieve Zm almost 1 as the total number of neurons increases which indicates that most of the final neuron state in the solution space achieved global minimum energy [19]. The complexity of the searching technique has been reduced by implementing GA. The crossover stage improves the unsatisfied bit string with the highest fitness. The bit strings improved when it achieved the highest fitness as the number of generations increased. Therefore, GA produces many bit strings that achieved global minimum energy compared to the exhaustive search method.

(iii) Therefore, DHNN-RANMAX2SATEA also manages to achieve Zm almost 1 as the total number of neurons increases. This indicates that DHNN-RANMAX2SATEA manages to obtain stable final neuron states. The reason is due to the capability of DHNN-RANMAX2SATEA in achieving optimal training phase which results in an optimal testing phase where global minimum energy will be produced. Moreover, EA produces a bit string with less complexity by partitioning the solution space into 4 parties. The number of local solutions produced at the end of computation will be reduced by the effective relaxation method by choosing a relaxation rate of 3 [30].

(iv) Generally, based on the outcomes of DHNN-RANMAX2SATES, DHNN-RANMAX2SATEA, and DHNN-RANMAX2SATGA is able to withstand the complexity up to 300 neurons. It was observed that more than 99% of final state of the neuron in DHNN-RANMAX2SATGA and DHNN-RANMAX2SATEA achieved the global minimum solution. However, it was observed that 0.1% of final state of the neuron in DHNN-RANMAX2SATES achieved the global minimum solution. Therefore, DHNN-RANMAX2SATGA and DHNN-RANMAX2SATEA outperformed DHNN-MAX2SATES as the number of neurons increased in terms of global minimum ratio.

4.4. Similarity Analysis (Jaccard Similarity Index)

Figure 6 shows the JSI produced by DHNN-RANMAX2SATES, DHNN-RANMAX2SATEA, and DHNN-RANMAX2SATGA models. Similarity analysis was performed to analyze the final neuron state by comparing the retrieved neuron state with the benchmark neuron state. The JSI was chosen to investigate the quality of the solutions produced by DHNN-RANMAX2SATES, DHNN-RANMAX2SATGA, and DHNN-RANMAX2SATEA.

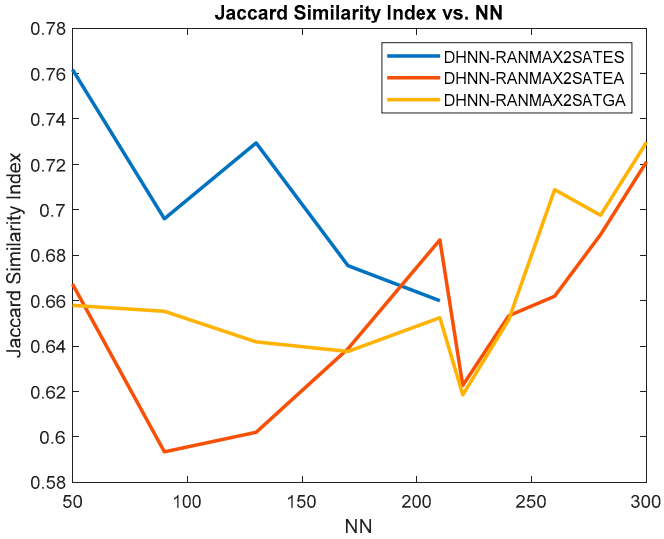


Figure 6. Jaccard Similarity Index, JSI produced by the DHNN-RANMAX2SAT.

(i) Based on Figure 6, DHNN-RANMAX2SATES shows the highest JSI at $NN = 50$. This indicates the major deviation and bias in the final states generated. The high value of JSI indicates that the model achieves overfitting as the DHNN-RANMAX2SATES model failed to produce differences in the final states of the neuron. However, there is a decrease in trend from $NN = 130$ to $NN = 170$. The JSI is decreasing, showing that the final neuron state generated is varied as the neuron increased [15]. This is due to the fewer benchmark neurons generated during the retrieval phase by the proposed model.

(ii) However, Jaccard has stopped getting any value when $NN > 210$ because all the solutions retrieved by the network are local solutions. This is because the nature of ES that operates based on trial and error could affect the minimization of the cost function. Since ES failed to produce optimal synaptic weight in training phase, it affects the final neuron states produced by the model at the end of computation.

(iii) According to Figure 6, we can see that the fluctuation for DHNN-RANMAX2SATGA and DHNN-RANMAX2SATEA increased. This is due to the total number of neurons increases. This increases the chances for the neuron to be trapped at the local minima. A higher number of total clauses imply more training error during the training phase which causes less variation of the final solution than the benchmark solution. Thus, this causes the trend of JSI for DHNN-RANMAX2SATGA and DHNN-RANMAX2SATEA to increase.

(iv) However, DHNN-RANMAX2SATEA has the lowest index value for Jaccard when $NN = 90$. In this case, the neuron retrieved from DHNN-RANMAX2SATEA has a lowest similarity with the benchmark state. The higher number of neuron variations produced by the network obtains lower value similarity index. This shows that the network produces less overfitting of the final states of the neuron.

4.5. Statistical Analysis

A Friedman Test was conducted for all DHNN-RANMAX2SAT models based on the results of RMSE Training. The analysis from the Friedman Test provided an insight whether the performance of the DHNN model in terms of RMSE Training is statistically significant or not. Initially, the null hypothesis, H_θ is defined, whereby H_θ = there is no significance in terms of RMSE Training between all models. The degree of freedom (df) considered is $df = 2$ with a significance level of $\alpha_0 = 0.05$ (95% confidence interval). Subsequently, the attained p-value was 0.000045 with a Chi square value of $\chi^2 = 20$. By observing that the value of p is much less than $\alpha_0 = 0.05$, the H_θ is rejected. This implies that the performance of each DHNN-RANMAX2SAT model in the training phase is not equal or statistically significant. Hence, the superiority of DHNN-RANMAX2SATEA as reported in Figure 3a is acknowledged. As DHNN-RANMAX2SATEA achieved the highest rank of 1 as compared to other algorithms, this highlights the importance of implementing an optimal training algorithm to minimize the satisfiable clauses of RANMAX2SAT.

5. Conclusions

One of the significant milestones in AI is to create DHNN that has the ability to learn optimally. This can be done by implementing flexible logic into DHNN. This paper serves as a benchmark to more implementation of non-satisfiability in DHNN. First, this study introduces a new logical rule, namely RANMAX2SAT, by combining two logical formulations, that is, Satisfiable and Non-Satisfiable. Note that, each clause in RANMAX2SAT contains redundant variables and this is the first attempt to introduce non-systematic logic into MAX2SAT (Refer Equation (3)). Second, the proposed RANMAX2SAT was implemented into DHNN or DHNN-RANMAX2SAT as a symbolic rule that governs the connection of the neurons. This can be done by comparing the cost function in Equation (8) with the energy function in Equation (13). It is worth mentioning that the ising spin of the neuron in DHNN-RANMAX2SAT is following the work of [33], where the dynamic is converged to the nearest local minimum energy. Third, the proposed model was optimized by using EA or DHNN-RANMAX2SATEA that was inspired by socio-political metaheuristics. The proposed EA was used to find the interpretation that leads to minimized cost function. In the perspective of Equation (3), the proposed EA will only learn the Satisfiable logic that formulates the whole logical formulation. Again, this is the first introduction of EA in optimizing the learning of DHNN that is not Satisfiable and has nonzero cost function. Finally, the quality of solution of DHNN-RANMAX2SAT model was tested in terms of various performance metrics. According to the experimental results, the proposed DHNN-RANMAX2SATEA outperformed other existing DHNN models in terms root mean square error, mean absolute error, mean absolute percentage error, global minima ratio, and Jaccard similarity analysis. It was observed that most of 99% of the final state of the neurons in DHNN-RANMAX2SATEA achieved global minimum solution. This shows that the proposed DHNN-RANMAX2SATEA managed to achieve the optimal training and testing phase which indicates the possibility of the RANMAX2SAT becoming an optimal symbolic rule for DHNN. As for future work, there are several interesting directions that are worth exploring. The proposed RANMAX2SAT can be implemented in another subset of ANN such as Boolean Neural Network [34], Graph Neural Network [35], or Kohonen Neural Network [36]. Due to the nature of RANMAX2SAT, it is interesting to observe the potential cost function of the mentioned ANN variants. In terms of learning phase, recent metaheuristics algorithm such as Black hole Algorithm [37], Driving Training-Based Optimization [38], Honey Badger Algorithm [39], Harmony Search-based Algorithm [40], and Gradient-based Optimizer [41] can also be implemented. The key here is to embed the feature of RANMAX2SAT into the objective function of the mentioned algorithms. Moreover, it would be worth exploring other effective algorithms to ensure the neuron in DHNN always converges to the global minimum energy. For instance, implementation of the Mutation operator [42] and memristor [43] were reported to increase the search space of the DHNN. Finally, the robust DHNN-RANMAX2SATEA has good potential to become

good forecasting model for various real-life modeling that is random in nature such as flood modeling, seismic modeling, and tsunami modeling. This can inspire the next implementation of large-scale logic mining design incorporated with DHNN-RANMAX2SATEA, which has the ability to classify and forecast.

Author Contributions: Conceptualization, project administration, formal analysis, and writing, V.S.; validation, M.F.M.; supervision, M.S.M.K.; writing—review and editing, N.E.Z.; writing—review and editing, S.S.M.S.; validation, S.Z.M.J.; validation and funding acquisition, M.A.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research is fully funded and supported by Universiti Sains Malaysia, Short Term Grant, 304/PMATHS/6315390.

Data Availability Statement: Not applicable.

Acknowledgments: All the authors gratefully acknowledged the financial support from the "Universiti Sains Malaysia, Short Term Grant, 304/PMATHS/6315390". We would like to express great appreciation to Revin Samuel James Selva Kumar for his helpful support in completion of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

To better understand the Algorithm 3, the process Election Algorithm will be explained with an illustrative example. EA is utilized to obtain the optimal solution for P_{RM2SAT} that minimizes the cost function during the training phase of the DHNN. Example of RANMAX2SAT formulation is taken from Equation (6).

(i) Initialization population and forming initial parties.

The population that consists of candidates and voters will be initialized. In this illustrative example, 20 individuals of a random population, N_{PopEA} will be initialized. The individuals consist of candidates and voters and can be represented as $S_i = [S_1, S_2, S_3, \ldots, S_{20}]$ where $S_i = \{-1, 1\}$. The solution space partitioned into 4 parties. Thus, the population of 20 are divided into 4 parties. Each individual will be calculated for their eligibility based on Equation (30). The individual with the highest eligibility will be the first candidate and will be highlighted as orange. Tables A1–A4 show the voters and candidates in Party 1, Party 2, Party 3, and Party 4, respectively.

Table A1. S_2 selected as candidate in Party 1.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_1	1	1	−1	−1	1	1	1	1	6
S_2	−1	1	1	1	−1	−1	1	1	6
S_3	1	−1	1	1	−1	−1	−1	−1	4
S_4	1	1	1	1	−1	−1	−1	−1	4
S_5	1	1	−1	1	−1	−1	−1	1	5

Table A2. S_6 selected as candidate in Party 2.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_6	1	1	1	1	−1	−1	1	1	6
S_7	−1	1	−1	1	−1	−1	−1	−1	4
S_8	1	−1	−1	−1	−1	1	−1	−1	4
S_9	1	−1	−1	−1	−1	1	−1	1	5
S_{10}	1	−1	1	1	−1	−1	−1	−1	4

Table A3. S_{14} selected as candidate in Party 3.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_{11}	1	1	1	1	1	1	−1	−1	5
S_{12}	1	1	−1	−1	1	1	−1	−1	4
S_{13}	1	1	1	−1	−1	−1	−1	1	4
S_{14}	−1	1	1	1	−1	−1	1	1	6
S_{15}	−1	−1	−1	−1	1	1	−1	−1	4

Table A4. S_{18} selected as candidate in Party 4.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_{16}	1	1	1	1	−1	−1	−1	−1	4
S_{17}	−1	−1	−1	−1	−1	1	−1	−1	4
S_{18}	−1	1	−1	−1	−1	1	1	1	5
S_{19}	−1	−1	−1	−1	1	1	−1	−1	4
S_{20}	−1	−1	−1	−1	−1	−1	−1	−1	3

(ii) Positive Advertisement

The number of voters, v_i^j that will be influenced by the candidate, L_j calculated by Equation (32) with $\sigma^p = 0.5$. Therefore, $N_S = 2$ influenced voters will be selected randomly. The number of neuron states that will be updated by the influenced will be determined based on the Equation (34). The candidate L_j will be replaced if a voter has a higher eligibility value than that candidate. Note that the individual highlighted with red is denoted as the new candidate, L_j. The individual highlighted green is denoted as an old candidate, L_j. The individual highlighted with blue is denoted as influenced voters. The neuron state that has been updated is highlighted with yellow. Tables A5–A8 show the process of positive advertisement in Party 1, Party 2, Party 3, and Party 4, respectively.

Table A5. S_2 remained as candidate in Party 1.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_1	1	1	−1	−1	1	1	1	1	6
S_2	−1	1	1	1	−1	−1	1	1	6
S_3	1	−1	1	1	1	1	−1	−1	5
S_4	1	1	1	1	1	1	−1	−1	5
S_5	1	1	−1	1	−1	−1	−1	1	5

S_3 and S_4 are the influenced voters and have undergone state flipping process. Since S_3 and S_4 have lower fitness than S_2; S_2 will remain as the candidate.

Table A6. S_6 remained as candidate in Party 2.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_6	1	1	1	1	−1	−1	1	1	6
S_7	−1	1	−1	1	1	1	−1	−1	5
S_8	1	−1	−1	−1	1	−1	−1	−1	4
S_9	1	−1	−1	−1	−1	1	−1	1	5
S_{10}	1	−1	1	1	−1	−1	−1	−1	4

S_7 and S_8 are the influenced voters and have undergone state flipping process. Since S_7 and S_8 have lower fitness than S_6 and S_6 will remain as the candidate.

Table A7. S_{13} selected as candidate in Party 3.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_{11}	1	1	1	1	1	1	−1	−1	5
S_{12}	1	1	−1	−1	1	−1	1	−1	5
S_{13}	1	1	1	−1	−1	1	1	1	7
S_{14}	−1	1	1	1	−1	−1	1	1	6
S_{15}	−1	−1	−1	−1	1	1	−1	−1	4

S_{12} and S_{13} are the influenced voters and have undergone state flipping process. Since S_{13} has higher fitness than S_{14}, S_{13} will be selected as the candidate.

Table A8. S_{19} selected as candidate in Party 4.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_{16}	1	1	1	1	−1	−1	−1	−1	4
S_{17}	−1	−1	−1	−1	−1	1	1	1	5
S_{18}	−1	1	−1	−1	−1	1	1	1	5
S_{19}	−1	−1	−1	−1	1	1	1	1	6
S_{20}	−1	−1	−1	−1	−1	−1	−1	−1	3

S_{17} and S_{19} are the influenced voters and have undergone state flipping process. Since S_{19} has higher fitness than S_{18}, S_{19} will be selected as the candidate.

(iii) Negative Advertisement

The number of voters v_i^* that will be attracted by the candidate L_j can be calculated by Equation (28) with $\sigma^n = 0.5$. Note that Party 1 will attract voters from Party 3 and Party 2 will attract voters from Party 4. The number of neuron states that will be updated by the influenced will be determined based on the Equation (38). The candidate L_j will be replaced if a voter has a higher eligibility value than that candidate. Note that Individual highlighted with red is denoted as the new candidate, L_j. The individual highlighted green is denoted as an old candidate, L_j. The individual highlighted with blue is denoted as attracted voters v_i^* in the new party and gray in the old party. The neuron state that has been updated is highlighted with yellow. Tables A9–A12 show the process of negative advertisement in Party 1, Party 2, Party 3, and Party 4, respectively.

Table A9. S_2 remained as candidate in Party 1.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_1	1	1	−1	−1	1	1	1	1	6
S_2	−1	1	1	1	−1	−1	1	1	6
S_3	1	−1	1	1	1	1	−1	−1	5
S_4	1	1	1	1	1	1	−1	−1	5
S_5	1	1	−1	1	−1	−1	−1	1	5
S_{15}	−1	−1	−1	−1	1	1	1	1	6

Party 1 gained S_{15} from Party 3. S_2 will remain as the candidate Party 1.

Table A10. S_{18} selected as candidate in Party 1.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_6	1	1	1	1	−1	−1	1	1	6
S_7	−1	1	−1	1	1	1	−1	−1	5
S_8	1	−1	−1	−1	1	−1	−1	−1	4
S_9	1	−1	−1	−1	−1	1	−1	1	5
S_{10}	1	−1	1	1	−1	−1	−1	−1	4
S_{18}	−1	1	1	1	1	−1	1	1	7

Party 2 gained S_{18} from Party 4. Since S_{18} has higher fitness than S_6, S_{18} will be selected as the candidate.

Table A11. Party 3 lost S_{15} to Party 1.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_{11}	1	1	1	1	1	1	−1	−1	5
S_{12}	1	1	−1	−1	1	−1	1	−1	5
S_{13}	1	1	1	−1	−1	1	1	1	7
S_{14}	−1	1	1	1	−1	−1	1	1	6
S_{15}	−1	−1	−1	−1	1	1	−1	−1	4

Table A12. Party 4 lost S_{18} to Party 2.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_{16}	1	1	1	1	−1	−1	−1	−1	4
S_{17}	−1	−1	−1	−1	−1	1	1	1	5
S_{18}	−1	1	−1	−1	−1	1	1	1	5
S_{19}	−1	−1	−1	−1	1	1	1	1	6
S_{20}	−1	−1	−1	−1	−1	−1	−1	−1	3

(iv) Coalition

Two parties will be grouped together where the individual with the highest eligibility value in the coalition party will be candidate L_j. The number of neuron states that will be updated by all voters v_i^* will be determined based on the Equation. The candidate L_j will be replaced if a voter has a higher eligibility value than the candidate. Note that the individual highlighted with red is denoted as the new candidate L_j. The neuron state that has been updated is highlighted with yellow. Table A13 shows the coalition of Party 1 and Party 4 and Table A14 shows coalition of Party 2 and Party 3.

Table A13. Coalition of Party 1 and Party 4.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_1	−1	−1	1	1	−1	−1	−1	−1	6
S_2	−1	1	1	1	−1	−1	1	1	6
S_3	1	1	−1	1	1	1	1	1	5
S_4	1	1	1	1	−1	−1	1	1	5
S_5	1	1	−1	−1	1	1	1	1	5
S_{15}	1	1	1	1	−1	−1	−1	−1	4
S_{16}	1	1	1	1	−1	−1	−1	−1	6
S_{17}	1	1	1	1	−1	−1	1	1	4
S_{19}	1	1	1	1	−1	−1	−1	−1	6
S_{20}	−1	−1	−1	−1	−1	−1	1	1	3

Note that Party 1 coalited with Party 4. The individual S_2 remained as candidate of this coalition party.

Table A14. Coalition of Party 2 and Party 3.

S_i	A	B	X_1	Y_1	X_2	Y_2	Z_1	Z_2	$f_{RM2SATEA}$
S_6	1	1	−1	−1	1	1	1	1	6
S_7	−1	1	−1	1	−1	−1	−1	−1	4
S_8	1	−1	−1	1	−1	−1	−1	−1	4
S_9	1	−1	−1	−1	−1	1	1	−1	5
S_{10}	1	−1	−1	−1	−1	−1	−1	−1	3
S_{18}	−1	1	1	1	1	−1	1	1	7
S_{11}	1	1	1	1	−1	−1	−1	−1	4
S_{12}	1	1	−1	−1	−1	−1	−1	−1	3
S_{13}	−1	−1	−1	1	1	−1	−1	−1	5
S_{14}	−1	1	1	−1	1	1	−1	1	6

Note that Party 2 coalited with Party 3. The individual S_{18} remained as candidate of this coalition party.

(v) Election Day

The final eligibility of all candidates from both coalition parties will be compared. If the eligibility value of the candidate is maximum ($f_{RM2SATEA} = 7$), the candidate will be elected. In this case, since S_{18} achieved the maximum eligibility value and higher eligibility value than S_2, S_{18} selected as the winner.

References

1. Liu, X.; Qu, X.; Ma, X. Improving flex-route transit services with modular autonomous vehicles. *Transp. Res. Part E Logist. Transp. Rev.* **2021**, *149*, 102331. [CrossRef]
2. Chen, X.; Wu, S.; Shi, C.; Huang, Y.; Yang, Y.; Ke, R.; Zhao, J. Sensing data supported traffic flow prediction via denoising schemes and ANN: A comparison. *IEEE Sens. J.* **2020**, *20*, 14317–14328. [CrossRef]
3. Chereda, H.; Bleckmann, A.; Menck, K.; Perera-Bel, J.; Stegmaier, P.; Auer, F.; Kramer, F.; Leha, A.; Beißbarth, T. Explaining decisions of graph convolutional neural networks: Patient-specific molecular subnetworks responsible for metastasis prediction in breast cancer. *Genome Med.* **2021**, *13*, 42. [CrossRef] [PubMed]
4. Lin, Y. Prediction of temperature distribution on piston crown surface of dual-fuel engines via a hybrid neural network. *Appl. Therm. Eng.* **2022**, *218*, 119269. [CrossRef]
5. Zhou, L.; Wang, P.; Zhang, C.; Qu, X.; Gao, C.; Xie, Y. Multi-mode fusion BP neural network model with vibration and acoustic emission signals for process pipeline crack location. *Ocean. Eng.* **2022**, *264*, 112384. [CrossRef]
6. Hopfield, J.J.; Tank, D.W. "Neural" computation of decisions in optimization problems. *Biol. Cybern.* **1985**, *52*, 141–152. [CrossRef]
7. Xu, S.; Wang, X.; Ye, X. A new fractional-order chaos system of Hopfield neural network and its application in image encryption. *Chaos Solitons Fractals* **2022**, *157*, 111889. [CrossRef]
8. Boykov, I.; Roudnev, V.; Boykova, A. Stability of Solutions to Systems of Nonlinear Differential Equations with Discontinuous Right-Hand Sides: Applications to Hopfield Artificial Neural Networks. *Mathematics* **2022**, *10*, 1524. [CrossRef]
9. Xu, X.; Chen, S. An Optical Image Encryption Method Using Hopfield Neural Network. *Entropy* **2022**, *24*, 521. [CrossRef]
10. Mai, W.; Lee, R.S. An Application of the Associate Hopfield Network for Pattern Matching in Chart Analysis. *Appl. Sci.* **2021**, *11*, 3876. [CrossRef]
11. Folli, V.; Leonetti, M.; Ruocco, G. On the maximum storage capacity of the Hopfield model. *Front. Comput. Neurosci.* **2017**, *10*, 144. [CrossRef]
12. Lee, D.L. Pattern sequence recognition using a time-varying Hopfield network. *IEEE Trans. Neural Netw.* **2002**, *13*, 330–342. [CrossRef]
13. Abdullah, W.A.T.W. Logic programming on a neural network. *Int. J. Intell. Syst.* **1992**, *7*, 513–519. [CrossRef]
14. Abdullah, W.A.T.W. Logic Programming in neural networks. *Malays. J. Comput. Sci.* **1996**, *9*, 1–5. Available online: https://ijps.um.edu.my/index.php/MJCS/article/view/2888 (accessed on 1 June 1996). [CrossRef]
15. Kasihmuddin, M.M.S.; Mansor, M.A.; Md Basir, M.F.; Sathasivam, S. Discrete mutation Hopfield neural network in propositional satisfiability. *Mathematics* **2019**, *7*, 1133. [CrossRef]
16. Sathasivam, S.; Mansor, M.A.; Ismail, A.I.M.; Jamaludin, S.Z.M.; Kasihmuddin, M.S.M.; Mamat, M. Novel Random k Satisfiability for k $\leq$ 2 in Hopfield Neural Network. *Sains Malays.* **2020**, *49*, 2847–2857. [CrossRef]
17. Karim, S.A.; Zamri, N.E.; Alway, A.; Kasihmuddin, M.S.M.; Ismail, A.I.M.; Mansor, M.A.; Hassan, N.F.A. Random satisfiability: A higher-order logical approach in discrete Hopfield Neural Network. *IEEE Access* **2021**, *9*, 50831–50845. [CrossRef]
18. Sidik, M.S.S.; Zamri, N.E.; Mohd Kasihmuddin, M.S.; Wahab, H.A.; Guo, Y.; Mansor, M.A. Non-Systematic Weighted Satisfiability in Discrete Hopfield Neural Network Using Binary Artificial Bee Colony Optimization. *Mathematics* **2022**, *10*, 1129. [CrossRef]
19. Zamri, N.E.; Azhar, S.A.; Mansor, M.A.; Alway, A.; Kasihmuddin, M.S.M. Weighted Random k Satisfiability for k = 1, 2 (r2SAT) in Discrete Hopfield Neural Network. *Appl. Soft Comput.* **2022**, *126*, 109312. [CrossRef]

20. Guo, Y.; Kasihmuddin, M.S.M.; Gao, Y.; Mansor, M.A.; Wahab, H.A.; Zamri, N.E.; Chen, J. YRAN2SAT: A novel flexible random satisfiability logical rule in discrete hopfield neural network. *Adv. Eng. Softw.* **2022**, *171*, 103169. [CrossRef]
21. Gao, Y.; Guo, Y.; Romli, N.A.; Kasihmuddin, M.S.M.; Chen, W.; Mansor, M.A.; Chen, J. GRAN3SAT: Creating Flexible Higher-Order Logic Satisfiability in the Discrete Hopfield Neural Network. *Mathematics* **2022**, *10*, 1899. [CrossRef]
22. Bonet, M.L.; Buss, S.; Ignatiev, A.; Morgado, A.; Marques-Silva, J. Propositional proof systems based on maximum satisfiability. *Artif. Intell.* **2021**, *300*, 103552. [CrossRef]
23. Kasihmuddin, M.S.M.; Mansor, M.A.; Sathasivam, S. Discrete Hopfield neural network in restricted maximum k-satisfiability logic programming. *Sains Malays.* **2018**, *47*, 1327–1335. [CrossRef]
24. Sathasivam, S.; Mamat, M.; Kasihmuddin, M.S.M.; Mansor, M.A. Metaheuristics approach for maximum k satisfiability in restricted neural symbolic integration. *Pertanika J. Sci. Technol.* **2020**, *28*, 545–564.
25. Tembine, H. Dynamic robust games in mimo systems. *IEEE Trans. Syst. Man Cybern. Part B* **2011**, *41*, 990–1002. [CrossRef]
26. Aissi, H.; Bazgan, C.; Vanderpooten, D. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.* **2009**, *197*, 427–438. [CrossRef]
27. Emami, H.; Derakhshan, F. Election algorithm: A new socio-politically inspired strategy. *AI Commun.* **2015**, *28*, 591–603. [CrossRef]
28. Sathasivam, S.; Mansor, M.; Kasihmuddin, M.S.M.; Abubakar, H. Election Algorithm for Random *k* Satisfiability in the Hopfield Neural Network. *Processes* **2020**, *8*, 568. [CrossRef]
29. Bazuhair, M.M.; Jamaludin, S.Z.M.; Zamri, N.E.; Kasihmuddin, M.S.M.; Mansor, M.A.; Alway, A.; Karim, S.A. Novel Hopfield neural network model with election algorithm for random 3 satisfiability. *Processes* **2021**, *9*, 1292. [CrossRef]
30. Sathasivam, S. Upgrading logic programming in Hopfield network. *Sains Malays.* **2010**, *39*, 115–118.
31. Zhi, H.; Liu, S. Face recognition based on genetic algorithm. *J. Vis. Commun. Image Represent.* **2019**, *58*, 495–502. [CrossRef]
32. Mansor, M.A.; Sathasivam, S. Accelerating activation function for 3-satisfiability logic programming. *Int. J. Intell. Syst. Appl.* **2016**, *8*, 44–50. [CrossRef]
33. Sherrington, D.; Kirkpatrick, S. Solvable model of a spin-glass. *Phys. Rev. Lett.* **1975**, *35*, 1792. [CrossRef]
34. Zhang, T.; Bai, H.; Sun, S. Intelligent Natural Gas and Hydrogen Pipeline Dispatching Using the Coupled Thermodynamics-Informed Neural Network and Compressor Boolean Neural Network. *Processes* **2022**, *10*, 428. [CrossRef]
35. Jiang, W.; Luo, J. Graph neural network for traffic forecasting: A survey. *Expert Syst. Appl.* **2022**, *207*, 117921. [CrossRef]
36. Yang, B.S.; Han, T.; Kim, Y.S. Integration of ART-Kohonen neural network and case-based reasoning for intelligent fault diagnosis. *Expert Syst. Appl.* **2004**, *26*, 387–395. [CrossRef]
37. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [CrossRef]
38. Dehghani, M.; Trojovská, E.; Trojovský, P. A new human-based metaheuristic algorithm for solving optimization problems on the base of simulation of driving training process. *Sci. Rep.* **2022**, *12*, 9924. [CrossRef]
39. Hashim, F.A.; Houssein, E.H.; Hussain, K.; Mabrouk, M.S.; Al-Atabany, W. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. *Math. Comput. Simul.* **2022**, *192*, 84–110. [CrossRef]
40. Zainuddin, Z.; Lai, K.H.; Ong, P. An enhanced harmony search based algorithm for feature selection: Applications in epileptic seizure detection and prediction. *Comput. Electr. Eng.* **2016**, *53*, 143–162. [CrossRef]
41. Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [CrossRef]
42. Hu, L.; Sun, F.; Xu, H.; Liu, H.; Zhang, X. Mutation Hopfield neural network and its applications. *Inf. Sci.* **2011**, *181*, 92–105. [CrossRef]
43. Wu, A.; Zhang, J.; Zeng, Z. Dynamic behaviors of a class of memristor-based Hopfield networks. *Phys. Lett. A* **2011**, *375*, 1661–1665. [CrossRef]