processes

MDPI

Article

Novel Hopfield Neural Network Model with Election Algorithm for Random 3 Satisfiability

Muna Mohammed Bazuhair [1], Siti Zulaikha Mohd Jamaludin [1], Nur Ezlin Zamri [2], Mohd Shareduwan Mohd Kasihmuddin [1,*], Mohd. Asyraf Mansor [2], Alyaa Alway [2] and Syed Anayet Karim [1]

[1] School of Mathematical Sciences, Universiti Sains Malaysia, Gelugor 11800, Penang, Malaysia; munabazuhair@student.usm.my (M.M.B.); szulaikha.szmj@usm.my (S.Z.M.J.); syedanayetkarim@student.usm.my (S.A.K.)

[2] School of Distance Education, Universiti Sains Malaysia, Gelugor 11800, Penang, Malaysia; ezlinzamri@student.usm.my (N.E.Z.); asyrafman@usm.my (M.A.M.); alyaalway@student.usm.my (A.A.)

* Correspondence: shareduwan@usm.my; Tel.: +60-46-534-769

Abstract: One of the influential models in the artificial neural network (ANN) research field for addressing the issue of knowledge in the non-systematic logical rule is Random k Satisfiability. In this context, knowledge structure representation is also the potential application of Random k Satisfiability. Despite many attempts to represent logical rules in a non-systematic structure, previous studies have failed to consider higher-order logical rules. As the amount of information in the logical rule increases, the proposed network is unable to proceed to the retrieval phase, where the behavior of the Random Satisfiability can be observed. This study approaches these issues by proposing higher-order Random k Satisfiability for $k \leq 3$ in the Hopfield Neural Network (HNN). In this regard, introducing the 3 Satisfiability logical rule to the existing network increases the synaptic weight dimensions in Lyapunov's energy function and local field. In this study, we proposed an Election Algorithm (EA) to optimize the learning phase of HNN to compensate for the high computational complexity during the learning phase. This research extensively evaluates the proposed model using various performance metrics. The main findings of this research indicated the compatibility and performance of Random 3 Satisfiability logical representation during the learning and retrieval phase via EA with HNN in terms of error evaluations, energy analysis, similarity indices, and variability measures. The results also emphasized that the proposed Random 3 Satisfiability representation incorporates with EA in HNN is capable to optimize the learning and retrieval phase as compared to the conventional model, which deployed Exhaustive Search (ES).

Keywords: Hopfield Neural Network; random 3 satisfiability; election algorithm; potential supervised learning

Citation: 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. https://doi.org/10.3390/pr9081292

Academic Editor: Giacomo Capizzi

Received: 10 May 2021
Accepted: 18 June 2021
Published: 26 July 2021

1. Introduction

A hallmark of any Artificial Neural Network (ANN) is the ability to behave according to the pre-determined output or decision. Without an "optimal" behavior, ANN will result in producing random outputs or decisions, which leads to useless information modelling. Although, ANNs can learn and model complex relationships, which is important to represent real-life problems, it lacks the interpretability of the results obtained and approximated. There is no simple correlation between the strength of the connection between neurons (the weights) and the results being approximated. Understanding the relationship between the ANN structure and the behavior of the output is a major goal in the field of Artificial Intelligence (AI). One of the notable ANN that has an association feature form of learning model is Hopfield Neural Network (HNN) [1]. The information is stored in the form bipolar representation and connected by synaptic weight. After the introduction of HNN, this network has been modified vigorously to solve optimization problems [1], medical diagnosis [2–4], electric power sector [5], investment [6], and many more. Despite achieving

extraordinary development in various performance metrics for any given problem, the optimal structure of HNN is still debatable among the ANN practitioners. To this end, the choice of the most optimal symbolic structure that governs HNN must be given fair share of attention.

The satisfiability (SAT) logical rule has a simple and flexible structure to represent the real-world problems. Satisfiability (SAT) is an NP-complete problem that was proven by [7] and plays a fundamental role in computational complexity theory [8–10]. One of the potential applications for SAT is the logical representation of the ANN. The goal of the logical representation is to represent the dataset and later filtered by the optimal logical rule. An interesting study conducted by [11] emphasized the complementary relationship between the logic and ANN in the field of AI. According to this study, computational logic is well-suited to represent structured objects and structure-sensitive processes for rational agents, while the ability to learn and to adapt to new environments is met by ANNs. One of the earliest efforts in implementing the logical rule in ANN was proposed by [12]. This work was inspired by the equivalence of the symmetrical mapping between the propositional logical rule with the energy function of the ANN [13,14]. After the emergence of logical rule in ANN, several logical rules have been proposed. The work by [15] proposed a higher-order Horn Satisfiability in Radial Basis Function Neural Network (RBFNN). The proposed work has been extended to another variant of logical rule namely 2 Satisfiability (2SAT) [16]. In this work, 2SAT became a logical structure to determine the parameters involved in RBFNN, thus fixing the number of hidden layers. The result from the performance evaluation in this work demonstrates the effectiveness of 2SAT in governing the structure of RBFNN. The first non-satisfiable logical rule namely Maximum k Satisfiability (MAXkSAT) in HNN was introduced by [17]. Despite achieving non-zero cost function during learning phase, the results showed a solid performance of the proposed model in obtaining global minimum energy and the highest value of the ratio of satisfied clause. Unfortunately, the mentioned studies focused on using the systematic logical rule where the number of variables in each clause is always constant. The prospect of the non-systematic logical rule in ANN is still poorly understood.

One of the challenges in embedding logical rule in HNN is the effectiveness of the learning phase. Output of the final state in HNN is structure-dependent and requires effective learning mechanism. Despite obtaining the optimal final state in previous studies such as in [17], finding logical assignments that lead to the optimal value of the cost function becomes convoluted as the number of logical clause increase. This is due to the probability of finding that the optimal cost function value of the HNN will reduce to zero. As a result, the final state of HNN governed by logical rule will be trapped into local minimum energy or suboptimal state. For instance, the work of [18] demonstrates the main weaknesses of conventional HNN in terms of retrieval capacity as the number of neurons increases. To overcome this issue, utilizing metaheuristic algorithm to find the optimal neuron assignment that leads to the minimization of the cost function has been proposed. [19] proposed Genetic Algorithm (GA) in learning 2SAT logical rule in HNN. The proposed metaheuristics is reported to obtain the optimal final state that corresponds to absolute minimum energy of the logical rule. The final state of the neuron exhibits higher value of global minima ratio, lower hamming distance and lower computational time. Ref. [20] proposed Artificial Immune System (AIS) to complement 3 Satisfiability (3SAT) in Elliot based HNN. The proposed method has been compared with other conventional metaheuristics algorithm such as Exhaustive Search (ES) and GA. The use of metaheuristics was extended to another logical variant that is not satisfiability. Ref. [21] proposed an interesting comparison among the prominent metaheuristics in learning Maximum 2 Satisfiability (MAX2SAT) logical rule. In this study, Artificial Bee Colony (ABC) integrated with MAX2SAT is observed to outperform other major metaheuristics. Despite the possibility of overfitting during learning phase of HNN, ABC remains competitive in finding the optimal assignment that minimizes the cost function of the MAX2SAT. In another development, [22] proposed Imperialist Competitive Algorithm (ICA) to opti-

mize 3 Satisfiability (3SAT) in HNN. The proposed method was implemented in logic mining paradigm where the proposed ICA is reported to optimize the logical extraction of the benchmark datasets. However, all the mentioned metaheuristics algorithms only learn systematic logical rule (either systematic second order or third-order logical rule). The proposed metaheuristics only discover single search space and no effective solution partitioning to locate the alternative neuron assignment that minimizes the cost function.

Currently, Election Algorithm (EA) proposed by [23] starts to gain popularity in solving various optimization problems. EA is inspired by the social behavior of individuals during electoral activity. EA simulates the actual political electoral systems that are organized by governments such as presidential election that starts with grouping the population into parties and selecting a representative of each party to convey their beliefs and ideas to the public. Parties compete against each other for seeking the most votes and the candidate who gets the most votes will be the leader. This strategy has a good resemblance to other papers which utilizes candidate as their potential solution of the objective function. In [24], the authors introduced Election Campaign Optimization (ECO) algorithm by considering the solution of the function as a candidate. In this case, the higher prestige of a particular candidate, the larger the cover range that results in smaller mean square deviation. Another resemblance of EA is reported in Evolutive Election Based Optimization (EVEBO) [25]. EVEBO is reported to utilize different types of electoral system to conduct series of optimization effort for both combinatorial and numerical problems. For this reason, EVEBO has more degree of freedom to be applied for a wide range of problems. Emami later proposed [26] the Chaotic Election Algorithms (CEA) by introducing migration operator to complement the existing EA. CEA enhances the EA by implementing chaotic positive advertisement to accelerate the convergence because the distance function that is used to find the eligibility consumes more computation time. In the development of logic programming in HNN, [27] has successfully implemented the first non-systematic logical rule namely Random 2 Satisfiability (RAN2SAT) in HNN. The proposed work utilized EA to find the correct assignment that leads to minimize the cost function. The proposed EA in this work was inspired by the work of [23] where there were three operators have been introduced to optimize the learning phase of HNN for RAN2SAT. They adopt an interesting distance function to complement the structure of RAN2SAT. The proposed method can perform the learning phase with minimal error and result in high retrieval capacity. However, despite their great potential, the computational ability of EA in doing higher order Random k Satisfiability (RANkSAT) has not been demonstrated. Higher order RANkSAT creates both upsides and downsides during learning phase of HNN. Third order logic in RANkSAT for $k = 3$ (RAN3SAT) can be easily satisfied but the computational capacity will increase dramatically. On the other hand, the first order logic is hard to satisfy and tend to consume more learning iteration. In this case, EA provides solution partitioning mechanism to reduce the impact of high-density capacity due to third order logic by increasing the probability of the network to obtain the correct assignment for first order logic. This view has been supported by the recent simulation work by [28]. This work showed the learning capacity of RAN3SAT in HNN reduced dramatically when the number of first order logic increases. As a result, the final neuron state tends to be trapped in local minimum solution. Thus, EA that is integrated with the third order logic is needed to complete the learning phase of RAN3SAT in HNN.

In this paper, a novel Random 3 Satisfiability logical rule will be embedded into HNN using EA as a learning mechanism. Different from the previous work, the proposed work in this paper will consider all combinational structure of $k = 1, 2, 3$ which creates 3-Dimensional (3D) decision system in HNN. This modification will provide flexibility and diversity of a logical structure in HNN. The compatibility of RAN3SAT as a symbolic rule in HNN is considered in this study creates a new perspective of assessing the variation of the final neuron state. The above merits are attributed to the following three objectives of the studies:

(i) To formulate a random logical rule that consist of first, second and third-order logical rule namely Random 3 Satisfiability in Hopfield Neural Network.
(ii) To construct a functional Election Algorithm that learns all the logical combination of Random 3 Satisfiability during the learning phase of Hopfield Neural Network.
(iii) To conduct a comprehensive analysis of the Random 3 Satisfiability incorporated with Election Algorithm for both learning and retrieval phase.

An effective HNN model incorporating the new logical rule was constructed and the proposed network is seen to be beneficial in knowledge extraction via logical rule. This paper has been organized as follows, Sections 2 and 3, Random 3 Satisfiability representation and the procedure of encoding the logical structure in HNN was explained. In Section 4, the mechanism of Election Algorithm will be discussed in detail. While Section 5 includes a brief introduction of Exhaustive Search. In Section 6, the implementation of the proposed model HNN-RAN3SAT model will be discussed. Furthermore, in Section 7, the experimental setup will be presented and the performance metrics for both HNN-RAN3SAT models will be presented in Section 8. Finally, Sections 9 and 10 will end the paper with discussion of the results obtained in learning and retrieval phase and the conclusion of the paper.

2. Random 3 Satisfiability Representation

Random k Satisfiability (RANkSAT) is a nonsystematic Boolean logic representation in which the total number of variables (literals or negated literals) in each clause is at most k. The logical rule consists of non-fixed clause length. In this section, we will be discussing the extended version of RANkSAT for $k = 3$ namely Random 3 Satisfiability (RAN3SAT). The general formulation for RAN3SAT or $P_{RAN3SAT}$ is given as follows:

$$P_{RAN3SAT} = \bigwedge_{i=1}^{NC} C_i^{(k)}, \text{ where } k = 1, 2, 3 \tag{1}$$

where $C_i^{(k)}$ refers to the clause i th with k variables which denoted as:

$$C_i^{(k)} = \begin{cases} A_i \vee B_i \vee D_i \,, & k = 3 \\ E_i \vee F_i \,, & k = 2 \\ G_i \,, & k = 1 \end{cases} \tag{2}$$

According to Equation (1), NC refers to the total number of the clauses in $P_{RAN3SAT}$ where $NC = n\left(C^{(3)}\right) + n\left(C^{(2)}\right) + n\left(C^{(1)}\right)$, and $n\left(C^{(k)}\right)$ indicates the number of clauses that contains k variables. A_i is a variable in $P_{RAN3SAT}$ that can be literal (A_i) or negated literal $(\neg A_i)$. Furthermore, $P_{RAN3SAT}$ consists of irredundant variables which means no repetition of the same variable (either in positive or negative literals) among $C_i^{(k)}$ in the logical structures [29]. Meanwhile, a variable is said to be a redundant if the variable exists more than one time in a logical rule.

$$P_{RAN3SAT} = (A \vee B \vee \neg C) \wedge (D \vee \neg E) \wedge (\neg F \vee \neg G) \wedge H \wedge \neg I \wedge \neg J \wedge K \wedge L \tag{3}$$

According to Equation (3), $P_{RAN3SAT}$ will be fully satisfied or $P_{RAN3SAT} = 1$ if the state of the variables reads $(A, B, C, D, E, F, G, H, I, J, K, L) = (1, 1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1)$. Note that, the mentioned state is not unique especially dealing with $C_i^{(2)}$ and $C_i^{(3)}$. One of the main motivations in representing the variable in the form Equation (3) is the ability to store the information in Bipolar state where each state represents the potential pattern for the dataset [30].

3. RAN3SAT Representation in Hopfield Neural Network

Hopfield Neural Network (HNN) was proposed by [1] to solve various optimization problems. HNN consists of N interconnecting neurons that is described by an Ising spin variable [31]. The conventional representation of HNN is given as follows:

$$S_i = \begin{cases} 1, \text{ if } W_{ij}S_j \geq \psi \\ -1, \text{ otherwise} \end{cases} \tag{4}$$

where W_{ij} is the synaptic weight from neuron S_i to S_j and ψ is a pre-defined threshold. As indicated by [32], the neuron does not permit any self-connection and employs a symmetrical neuron connection. In other words, the neuron connection in HNN can be represented in the form of symmetrical matrix with zero diagonal. This useful feature makes HNN an optimal platform to store important information. The final state of Equation (4) can be analyzed by computing the final energy and comparing it with the absolute minimum energy. As pointed out in [28,33], the main weakness of HNN is the lack of symbolic rule that governs the network. According to [28], the selection of $\psi = 0$ will ensure the network dynamic reach the nearest optimal solution. Lack of effective symbolic rule makes HNN, numerically difficult to obtain the absolute minimum energy. In this context, implementing $P_{RAN3SAT}$ as a logical rule in HNN (HNN-RAN3SAT) will effectively represents the neuron connections in the form of symbolic representation. To encode $P_{RAN3SAT}$ into HNN, a space $\hat{S}$ will be defined over a set of N variables $\{A, B, \ldots\}$. Each variable can be represented as a neuron and contains two states:

$$S_i = \begin{cases} 1 \text{ , if } A_i \text{ is TRUE} \\ -1 \text{ , if } A_i \text{ is FALSE} \end{cases} \tag{5}$$

One of the perspectives in defining the optimal implementation of $P_{RAN3SAT}$ in HNN is the minimization of the cost function. To find the cost function that corresponds to all the neuron where $E : \hat{S} \rightarrow R$, $P_{RAN3SAT}$ must be in Conjunctive Normal Form (CNF). By using De Morgan's Law, the inconsistency of $P_{RAN3SAT}$ can be obtained by converting the CNF $\left(\neg \wedge \left(\vee A_i^{(i)}\right)\right)$ to Disjunctive Normal Form (DNF) $\vee\left(\wedge \neg A_i^{(i)}\right)$. By complying with the convention, conjunction, and disjunction of $C_i^{(k)}$ were represented as arithmetic multiplication and arithmetic summation respectively. Therefore, the generalized cost function $E_{P_{RAN3SAT}}$ for $P_{RAN3SAT}$ is be given as:

$$E_{P_{RAN3SAT}} = \frac{1}{2^3} \sum_{j}^{n(C^{(3)})} \left(\prod_{i}^{3} A_i\right)_j + \frac{1}{2^2} \sum_{j}^{n(C^{(2)})} \left(\prod_{i}^{2} A_i\right)_j + \frac{1}{2} \sum_{j}^{n(C^{(1)})} A_j \tag{6}$$

$$\text{where } A_i \rightarrow \begin{cases} (1 - S_A) \text{ , if } \neg A_i \\ (1 + S_A) \text{ , otherwise} \end{cases} \tag{7}$$

Note that, $E_{P_{RAN3SAT}} \neq 0$, if at least $C_i^{(k)}$ is not fully satisfied and $E_{P_{RAN3SAT}} = 0$ signifies the state of $P_{RAN3SAT}$ were fully satisfied. On the other hand, the updating rule of HNN-RAN3SAT maintains the following dynamics.

The local field of each neuron can be found by:

$$h_i = \sum_{k=1, k\neq j}^{N} \sum_{j=1, j\neq k}^{N} W_{ijk}^{(3)} S_j S_k + \sum_{j=1, i\neq j}^{N} W_{ij}^{(2)} S_j + W_i^{(1)} \tag{8}$$

The updating rule is given by:

$$S_i(t+1) = \begin{cases} 1, \text{ if } \tan \mathrm{h}(\mathrm{h}_i) \geq 0 \\ -1, \text{ otherwise} \end{cases} \tag{9}$$

Note that, Equations (8) and (9) guarantee the network to decrease monotonically with dynamics. The final energy that corresponds to the final state of the neuron is given as follows [12]:

$$H_{P_{RAN3SAT}} = -\frac{1}{3}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} W_{ijk}^{(3)} S_i S_j S_k - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}^{(2)} S_i S_j - \sum_{i=1}^{N} W_i^{(1)} S_i \tag{10}$$

The synaptic weight of the HNN-RAN3SAT can be pre-calculated [34] by comparing Equations (10) and (6). The important assumptions when considering $P_{RAN3SAT}$ into Equations (8)–(10) are:

(i) The variables in $P_{RAN3SAT}$ are irredundant and if there is $m \in \{1, \cdots, N\}$ such that $i_m \notin C^{(k)} \wedge i_n, i_l \in C^{(k)} \rightarrow W_{i_m i_n i_j} = W_{i\ m i\ n} = 0$, $\forall n,\ l \in \{1, \cdots, N\},\ k \in \{1,2,3\}$. Hence all the clauses are independent to each other.

(ii) The no self-connection among all neurons in $C^{(k)}$ where $W_{ii} = 0,\ \forall i \in \{1, \ldots, N\}$ and the symmetric property of HNN leads to $W_{ij} = W_{ji}$ and W_{ijk} is equivalent to all permutation order of ijk such as $W_{jik},\ W_{kij}$ etc.

By using the above assumptions, let us simplify Equation (10) starting with the first term, we get:

$$-\frac{1}{3}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} W_{ijk}^{(3)} S_i S_j S_k = -\frac{1}{3}\left[\begin{array}{l} W_{111}S_1S_1S_1 + W_{112}S_1S_1S_2 + \ldots \\ +W_{123}S_1S_2S_3 + W_{213}S_2S_1S_3 + \ldots + W_{NNN}S_NS_NS_N \end{array}\right] \tag{11}$$

According to the assumption (ii), the synaptic weight associated with the self-connection is zero or $W_{111}S_1S_1S_1 = W_{112}S_1S_1S_2 = \cdots = 0$. Moreover, if there are at least two neurons belong to independent clauses, the synaptic weights between them will be zero as well. Therefore, the remainder of the above terms will be the terms to which neurons belong in the same clause. By considering assumption (ii) where the synaptic weights of HNN-RAN3SAT are symmetrical, the remaining synaptic weights are defined as follows:

$$W_{ijk}S_iS_jS_k = W_{jik}S_jS_iS_k = W_{ikj}S_iS_kS_j = W_{jki}S_jS_kS_i = W_{kij}S_kS_iS_j = W_{kji}S_kS_jS_i \tag{12}$$

By substituting Equation (12) to Equation (11), we obtain the first term as:

$$\begin{aligned} -\tfrac{1}{3}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} W_{ijk}^{(3)} S_iS_jS_k &= -\tfrac{3!}{3}\sum_{r=1}^{n(C^{(3)})}\left(W_{ijk}^{(3)} S_iS_jS_k\right)_r \\ &= -2\sum_{r=1}^{n(C^{(3)})}\left(W_{ijk}^{(3)} S_iS_jS_k\right)_r, i,j,k \in \{1, \cdots, N\} \end{aligned} \tag{13}$$

where the sub index r that corresponds to the number of $C^{(3)}$. Using the same process, the second term of the of Equation (10) is given as follows:

$$-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}^{(2)} S_iS_j = -\sum_{r=1}^{n(C^{(3)})+n(C^{(2)})}\left(W_{ij}^{(2)} S_iS_j\right)_r \tag{14}$$

The sub-index in Equation (14) is from one to the number of the clauses containing two and three variables. This is because the connection in $C^{(3)}$ will consist of connection with two variables as well. By substituting information in Equations (13) and (14) into Equation (10), the expanded representation of final energy for HNN-RAN3SAT is given as:

$$\begin{aligned} H_{P_{RAN3SAT}} &= -2\sum_{r}^{n(C^{(3)})}\left(W_{ijk}^{(3)} S_iS_jS_k\right)_r - \sum_{r}^{n(C^{(3)})+n(C^{(2)})}\left(W_{ij}^{(2)} S_iS_j\right)_r \\ &\quad -\sum_{i=1}^{N} W_i^{(1)} S_i, i,j,k \in \{1, \cdots, N\} \end{aligned} \tag{15}$$

Equation (15) provides an alternative energy representation for HNN-RAN3SAT because each term will identify the contribution for each clause. By comparing Equations (6) and (15), the generalized synaptic weight for HNN-RAN3SAT is given as follows:

$$W^{(k)} \in \left\{ -\frac{1}{2^i(k-1)!}, +\frac{1}{2^i(k-1)!} \right\} \tag{16}$$

where k is the dimension of the weights (the number of neurons that are being connected), i is the number of variables inside the clause. For example, for $k = 3$ with 2 variables inside a clause $i = 2$, the derived synaptic weight is given as $W_{ij}^{(2)} \in \left\{ -\frac{1}{4}, +\frac{1}{4} \right\}$. Next, the remaining constants of $E_{P_{RAN3SAT}}$ will be denoted as:

$$H_{P_{RAN3SAT}}^{\min} = -\frac{n(C^{(3)}) + 2n(C^{(2)}) + 4n(C^{(1)})}{8} \tag{17}$$

where $H_{P_{RAN3SAT}}^{\min}$ is the absolute minimum energy that HNN-RAN3SAT will achieve. Hence, the difference between the energy obtained by the final state and the absolute energy is denoted as follows:

$$E_{P_{RAN3SAT}} = \left| H_{P_{RAN3SAT}} - H_{P_{RAN3SAT}}^{\min} \right| \leq Tol \tag{18}$$

where $Tol = 0.001$ is taken as a tolerance value [31]. The aim of the HNN-RAN3SAT is to find the neuron state that corresponds to $E_{P_{RAN3SAT}} = 0$. Since Equation (8) consists of N variables (N_{var}) with only one logic of $P_{RAN3SAT}$, the number of free variables for the system is $N - 1$. The system has infinitely many final neuron states. Since all the clauses were independent to each other, the probability of getting $E_{P_{RAN3SAT}} = 0$ follows the Binomial distribution. Hence, the probability of getting the zero cost function in $P_{RAN3SAT}$ is given as follows:

$$P(E_{P_{RAN3SAT}} = 0) = \prod_{i=1}^{3} \left(1 - \frac{1}{2^i} \right)^{n(C^{(i)})} \tag{19}$$

where $n\left(C^{(i)}\right)$ is the number of clauses that contains i variables. Worth noting that, $P(E_{P_{RAN3SAT}} = 0) \to 0$, as $n\left(C^{(i)}\right) \to \infty$. In other words, the probability of finding satisfied neuron assignment approaches zero when the search space expands. Despite having a capability to pre-calculate $H_{P_{RAN3SAT}}^{\min}$, HNN-RAN3SAT requires neuron assignment that leads to $H_{P_{RAN3SAT}} \to H_{P_{RAN3SAT}}^{\min}$. In this context, implementing approximating algoritm during the learning phase of HNN-RAN3SAT will optimize Equation (18) in finding the optimal synaptic weights that correspond to lower value of $H_{P_{RAN3SAT}}$. Moreover, this system will have infinite solutions (due to the exponential growth in the search space), which justify the use of approximation algorithm such as Election Algorithm during learning phase of HNN.

4. Election Algorithm

Election algorithm (EA) is classified as an evolutionary algorithm by [35] and swarm intelligence algorithm by [23]. EA simulates the behavior of candidates in a presidential election process. Besides, EA is an iterative random population-based that starts with initializing the population randomly. EA works as a metaheuristic method depends on maximizing or minimizing the objective function of candidate solutions in the search space of the optimization problem. This method starts with a promising initial solution, whose objective value is used as an upper or a lower bound of objective function to generate better solutions and improve areas in search space. The integration of metaheuristic method with the other techniques from AI as ANNs and mathematical programming can improve a solution generated by ANN [36]. In this study, EA will be implemented for $P_{RAN3SAT}$ in HNN, which resulted in the proposed HNN-RAN3SATEA. The reason is that EA divides the search space area of the population into small ones. EA consists of three operators—positive advertisement, negative advertisement, and coalition. Through each operator, the population will be updated; the first operator will try to focus only on the small areas separately to improve them, and later next operators will try to explore wide areas in the search space. This will accelerate the convergence to the optimal solution. The mechanism of EA that implemented for $P_{RAN3SAT}$ in HNN consists of four stages:

1. Stage 1: Initialize Population

The population includes potential solutions of the search space of $P_{RAN3SAT}$ is generated randomly. Everyone $\boldsymbol{S}_i = (v_1, v_2, \ldots, v_n)$, $v_i \in \{-1, 1\}$ is a potential (candidate) solution of the

problem. Let the search space of $P_{RAN3SAT}$ be $\hat{\mathbf{S}}_{P_{RAN3SAT}} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_{2^n}\}$. Considering the eligibility or the fitness value of each $\boldsymbol{S}_i$ that qualifies it to be a candidate is the objective function $f : \hat{\mathbf{S}} \to \{0, 1, 2, \ldots\}$ of the optimization problem that was given by Equation (18).

$$f_{L_j} = \sum_{i=1}^{NC} C_i^{(k)},\ k = 1, 2, 3 \tag{20}$$

$$C_i^{(k)} = \begin{cases} 0, \text{ False} \\ 1, \text{ True} \end{cases} \tag{21}$$

The goal is maximizing the eligibility value or reducing the cost function value.

2. Stage 2: Forming Initial Parties

After initialization, the second stage will begin by dividing the search space of $P_{RAN3SAT}$ into N_{party} equal parts. Each party consists of N_j potential solutions given by the equation:

$$N_j = \frac{N_{pop}}{N_{party}} \tag{22}$$

where N_{pop} is the size of the population. The potential solution with the highest eligibility value in each party will be elected as a candidate.

3. Stage 3: Advertisement Campaign

In this stage, after formatting initial parties and choosing the initial candidate solution of each party, each candidate will start his own advertisement campaign that includes three steps positive advertisement, negative advertisement, and coalition.

(a) Positive Advertisement

The candidate will start their campaign by influencing the voters in the same party to increase his popularity. In general, people can be influenced by the one who has the same ideas and beliefs. In this algorithm and in discrete space, this process can be formalized by calculating the distance between the candidate $(\boldsymbol{S}_L)$ and the voters $(\boldsymbol{S}_v)$ of $P_{RAN3SAT}$ by using Equation (23). Meanwhile, the number of voters that support the candidate N_S is randomly selected by using Equation (24):

$$d\left(f_{L_j}, f_{v_j}\right) = \left|f_{L_j} - f_{v_j}\right| \tag{23}$$

where f_{L_j},f_{v_j} are the fitness values of the candidate and the voter in party j respectively.

$$N_{S_j} = N_j \sigma^p,\ j = 1, 2, 3, 4 \tag{24}$$

where σ^p is a positive advertisement rate $\sigma^P \in [0, 0.5]$. The candidates in each party will try to affect to their supporters. which means update the states of potential solutions (voters) randomly by flipping several variables of $S_{\ i}$ from 1 to -1 or vice-versa, by the given equation. Note that, $\omega_{v_i^j}$ is the eligibility distance coefficient and N_{var} is the number of the variables of the $P_{RAN3SAT}$ in party j as in Equation (27).

$$S_{v_i^j} = N_{\text{var}} \omega_{v_i^j} \tag{25}$$

$$\omega_{v_i^j} = \frac{1}{d\left(f_{L_j}, f_{v_i^j}\right) + 1} \tag{26}$$

$$N_{\text{var}} = 3n\left(C^{(3)}\right) + 2n\left(C^{(2)}\right) + n\left(C^{(1)}\right) \tag{27}$$

After the supporters are updated according to their candidate and their eligibility, there is a possibility for a voter to be a candidate (voter has higher eligibility than the candidate). Therefore, to increase the quality of the solutions in the parties, the candidate will be replaced by the solution with the highest fitness value (more qualified supporter). In other cases, if the voter and candidate share the same eligibility, the candidate position will remain (no replacement will be made). If the optimal solution is found in the positive advertisement, it would remain as a candidate through the next steps until the first iteration ends, then it will be announced as the best solution in the election stage. Figure 1 shows the process of updating the old candidate with new candidate after calculating the number of variables that need to be flipped based on Equations (25) and (26). Given, $d\left(f_{L_j}, f_{v_j}\right) = 1$ and $\omega_{v_i^j} = \frac{1}{2}$. Therefore, $S_{v_i^j} = 6$ which means six variables need to be flipped randomly. After the

updating process, the old candidate with three fitness value has been replaced with a new candidate with five fitness value.

	S_i	A	B	C	D	E	F	G	H	I	J	K	L	f
Before	Supporter	1	−1	−1	1	−1	1	1	−1	1	−1	−1	−1	3
	Candidate	−1	−1	−1	−1	−1	−1	−1	−1	1	1	−1	1	4

	S_i	A	B	C	D	E	F	G	H	I	J	K	L	f
After	Candidate	−1	−1	−1	1	−1	−1	−1	1	1	1	−1	1	5
	Supporter	−1	−1	−1	−1	−1	−1	−1	−1	1	1	−1	1	4

Figure 1. Updating and replacing the supporting voter as the new candidate.

(b) Negative Advertisement

In this stage, widening the area of search space is required. The candidates try to attract voters from other parties. We presented Equation (28) to decide the number of voters, the candidate can attract from other parties (parties with highest fitness candidate):

$$N_{S^*} = \sigma^n \left(N_j - N_{S_j} \right) \tag{28}$$

where $\sigma^n \in [0, 1]$ is the negative advertisement rate. After random selection, voters from other parties, the same steps, and equations that reflect the candidate's influence on the supporters from outside will be used. The eligibility distance coefficient is given as follows:

$$\omega_{v_i^*} = \frac{1}{d\left(f_{L_j}, f_{v_i^*} \right) + 1} \tag{29}$$

where $d\left(f_{L_j}, f_{v_i^*} \right)$ is the metric function of the eligibility of the candidate and voters. The formula of updating voters in parties is based on the following:

$$S_{v_i^*} = N_{\text{var}}\, \omega_{v_i^*} \tag{30}$$

By using Equation (20), we can calculate the eligibility of the new supporters (potential solutions of $P_{RAN3SAT}$). Again, if there exists a voter with fitness value greater than the candidate, the candidate will be replaced by this voter.

(c) Coalition

In this stage, different parties cooperate and form a coalition to explore more areas in search space of $P_{RAN3SAT}$. The coalition stage utilized the same steps and formulations as in negative advertisement. First, the two parties will be united randomly to decide who is going to be a new candidate after this union. First, find the eligibility distance function and distance coefficient by using Equation (29). After that, we decide the number of variables that needs to update in each voter from this combined party by using Equation (30). Now, all the voters' fitness value has been updated. Finally, we compare the fitness value of the voters and old candidates. If the old candidate still has the highest fitness value, then the old candidate will proceed to election day. But if other situations happened, for instance, a

voter has a higher fitness value compared to the old candidate. This fittest voter will be a new candidate to compete on election day with another party.

4. Stage 4: The Election Day

In this stage, the best solution (the candidate) in each party will be tested. If the solution has achieved the maximum fitness value $f_{NC} = n\left(C^{(3)}\right) + n\left(C^{(2)}\right) + n\left(C^{(1)}\right)$, this solution will be announced as the optimal solution. Otherwise, the second iteration will take place. The steps will be repeated until the conditions are met. Figure 2 and Algorithm 1 summarize the steps involved in EA during the learning phase of HNN-RAN3SAT and Pseudo code of EA respectively.

Algorithm 1: Pseudo Code of the Proposed HNN-RAN3SATEA

Generate initial population N_{POP}
while $(i < \max(r))$ or $(f_{L_i} < f_{NC})$
Forming Initial Parties by using Equation (22)
for $\left(j \leq N_{Party}\right)$ *do*
Calculate the similarity between the voter and the candidate utilizing Equation (23)
end
{Positive Advertisement}
Evaluate the number of voters by using Equation (24)
for $i \leq N_{S_j}$ *do*
Evaluate the reasonable effect from the candidate $\omega_{v_i^j}$ by using Equation (26)
Update the neuron state according to Equation (25)
if $f_{v_i^j} > f_{L_j}$
Assign v_i^j as a new L_j
Else
Remain L_j
End
{Negative advertisement}
Evaluate the number of voters $N_{v_j^*}$
for $i \leq N_{v_j^*}$ *do*
Evaluate the reasonable effect from the candidate $\omega_{v_i^j}$ by using Equation (29)
Update the neuron state according to Equation (30)
if $f_{v_i^*} > f_{L_j}$
Assign v_i^* as a new L_j
Else
Remain L_j
End
{Coalition}
for $i \leq N_j + N_k$ *do*
Evaluate the reasonable effect from the candidate $\omega_{v_i^j}$, by using Equation (29)
Update the neuron state according to Equation (30)
if $f_{v_i^*} > f_{L_j}$
Assign v_i^* as a new L_j
Else
Remain L_j
End
end while
return output the final neuron state

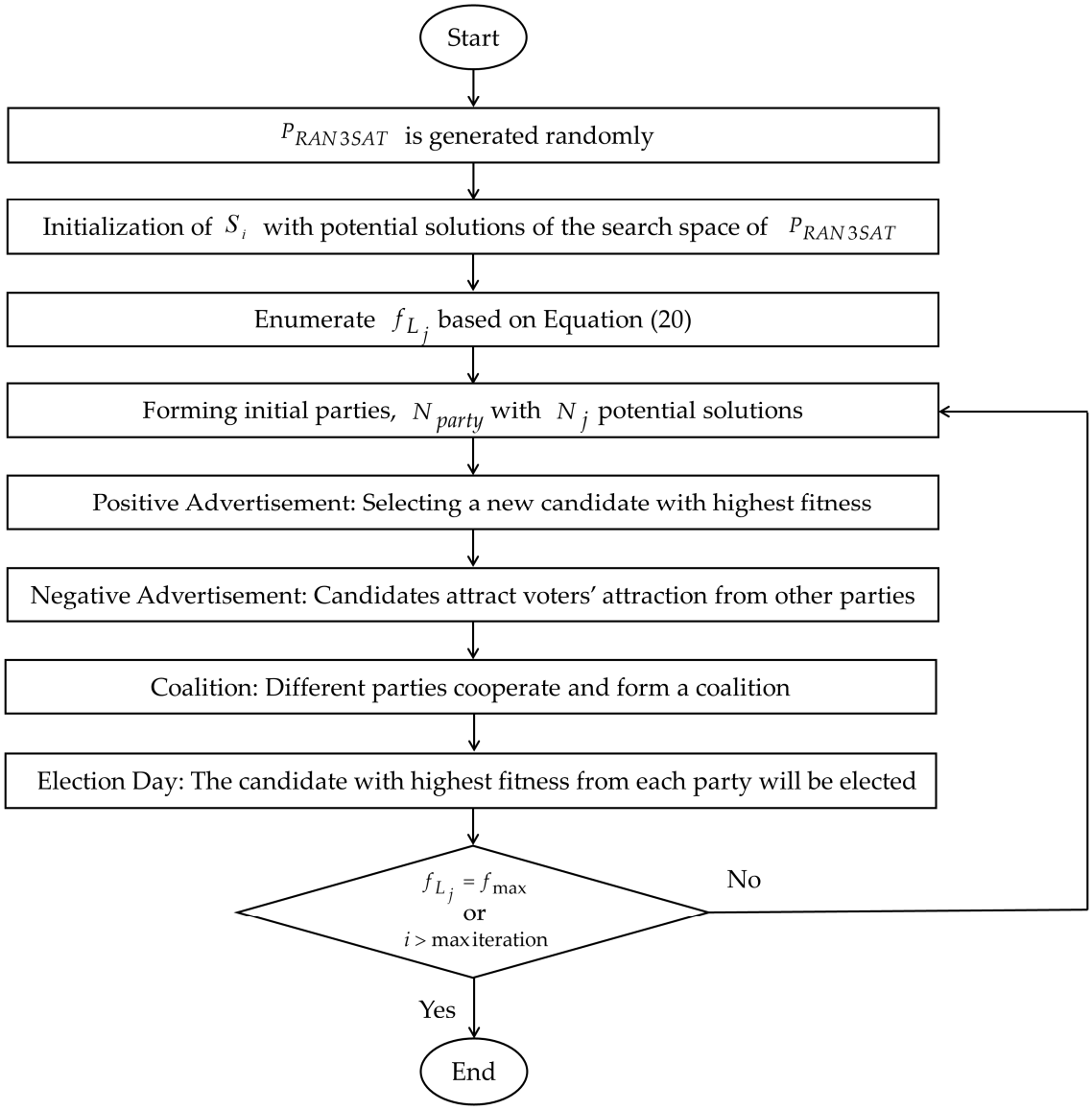

Figure 2. Flowchart of the mechanism of EA for $P_{RAN3SAT}$.

5. Exhaustive Search (ES)

Exhaustive search (ES) is one of the oldest techniques to enumerate and search for the solution. ES operates as a "generate and test" mechanism for all candidate solutions of the problem in search space until the optimal solution is obtained [37]. The main advantage of the ES algorithm is ES guarantees to obtain a solution (satisfied clause) by checking all candidate solutions in the search space. However, this algorithm consumes more computational time with intractable problems when the variables start increasing and the search space starts expanding exponentially with time [30]. Generally, ES algorithm consists of three main steps as stated below [38].

1. Step 1: Initialization.

   Generate the potential bit strings $\{S_1, S_2, \ldots, S_n\}$.

2. Step 2: Fitness Evaluation.

   Compute the number of satisfied clauses (fitness value of S_i) by using Equation (20).

3. Step 3: Test the solutions.

If the candidate has maximum fitness value, then the potential solution will return as output. Else, repeat the Step 1 and 2.

6. Summary of Learning and Retrieval Phase of HNN-RAN3SAT

This section demonstrates the full implementation of $P_{RAN3SAT}$ in HNN (or the HNN-RAN3SAT model). The main framework of the proposed HNN-RAN3SAT model is divided into two main phases. At first, the learning phase of the HNN-RAN3SAT model is introduced to check the satisfied assignment of $P_{RAN3SAT}$ by using learning algorithms to generate the optimal synaptic weights. In this study, Election Algorithm (EA) and Exhaustive Search (ES) are utilized as the learning algorithms. Then, the retrieval phase generates the correct final neuron states that lead to global minimum energy of the HNN-RAN3SAT model. The subsection below shows a detailed explanation of each phase in the HNN-RAN3SAT model.

6.1. Learning Phase in HNN-RAN3SAT

The purpose of embedding $P_{RAN3SAT}$ into the learning phase of HNN-RAN3SAT model is to create a network that will "behave" according to $P_{RAN3SAT}$ logical structure. $P_{RAN3SAT}$ has prominent characteristics such as bipolar representation and easy to represent in the form of neuron makes $P_{RAN3SAT}$ is compatible with HNN. The main concern in learning $P_{RAN3SAT}$ is the random structure of negated and positive literal in C_i^k which leads to learning complexity of HNN to achieve $E_{P_{RAN3SAT}} = 0$. To reduce possible combinatorial explosion, EA is implemented in HNN to find the optimal assignment for $P_{RAN3SAT}$ that leads to $E_{P_{RAN3SAT}} = 0$. In this work, EA will utilize effective operator that has been mentioned in the previous section to computation burden for HNN-RAN3SAT. Conventionally, HNN will utilize ES [17] to obtain the optimal assignment that leads to $E_{P_{RAN3SAT}} = 0$. If the proposed EA failed to obtain to get the correct assignment, HNN-RAN3SAT will retrieve a suboptimal final state which leads to local minima energy. In other words, HNN-RAN3SAT will only retrieve the random pattern. In summary, the learning phase of HNN-RAN3SAT will require EA to find the assignment that leads optimal synaptic weight. From this synaptic weight, HNN will operate as a system that "behave" according to the proposed $P_{RAN3SAT}$. The steps in executing learning phase of HNN-RAN3SAT model is summarized as follows:

1. Step 1: Convert $P_{RAN3SAT}$ into CNF type of Boolean Algebra.
2. Step 2: Assign neuron for each variable in $P_{RAN3SAT}$.
3. Step 3: Initialize the synaptic weights and the neuron state of HNN-RAN3SAT.
4. Step 4: Define the inconsistency of the $P_{RAN3SAT}$ logic by taking the negation of $P_{RAN3SAT}$.
5. Step 5: Derive the $E_{P_{RAN3SAT}}$ using Equation (6) that is associated with the defined inconsistencies in Step 4.
6. Step 6: Obtain the neuron assignments that leads to (using EA and ES), $E_{P_{RAN3SAT}} = 0$.
7. Step 7: Map the neuron assignment associated with the optimal synaptic weight. Precalculated synaptic weight can be obtained by comparing $E_{P_{RAN3SAT}}$ with the Final Energy function in Equation (15).
8. Step 8: Store synaptic weights as a Control Addressable Memory (CAM).
9. Step 9: Calculate the value of $H_{P_{RAN3SAT}}^{\min}$ by using Equation (17).

6.2. Retrieval Phase in HNN-RAN3SAT

After completing the learning phase of HNN-RAN3SAT, we will be using the synaptic weight obtained in the learning phase into the retrieval phase. This is a phase where the HNN demonstrate the behavior of $P_{RAN3SAT}$. According to [39], the neuron state produced by HNN should have non-linear relationship with the input state. Inspired by this motivation, we utilize Hyperbolic Tangent Activation Function (HTAF) [40] to squash the neuron output. The quality of the final neuron state will be checked based the differences between the calculated energy function $H_{P_{RAN3SAT}}$ and minimum energy

function $H^{\min}_{P_{RAN3SAT}}$. The following steps explain the calculation involved in retrieval phase of HNN-RAN3SAT model.

1. Step 1: Calculate the local field of each neuron in HNN-RAN3SAT model using Equation (8).
2. Step 2: Compute the neurons state value by using HTAF [40] and classify the final neuron state based on Equation (9).
3. Step 3: Calculate the final energy of the HNN-RAN3SAT model using Equation (15).
4. Step 4: Verify whether the final energy obtained satisfy the condition in Equation (18). If the difference in energy is within the tolerance value, we consider the final neuron state as global minimum solution.

Worth mentioning that, the effective learning phase of HNN-RAN3SAT will optimize the state produced by the retrieval phase. In this case, the choice of Metaheuristics Algorithm in obtaining $E_{P_{RAN3SAT}} = 0$ is paramount in ensuring the optimality of HNN-RAN3SAT.

7. Experimental Setup

The experimental setup for this study mostly follows the experiment set of [27]. The simulations for HNN-RAN3SATEA were executed on Dev C++ (Developed and Manufactured by Embarcadero Technologies, Inc, Texas, US) Version 5.11 running on the computer that has 4GB RAM and using Windows 8. The CPU threshold time for generating data will be 24 h and the simulation will be terminated if CPU time exceeds 24 h [41]. Notably, the models used a simulated dataset obtained by Dev C++ with 100,000 number of learning to generate $P_{RAN3SAT}$. Here, the probability of generating a literal is equal to the probability of generating a negated literal. Additionally, the logical structure of $P_{RAN3SAT}$ consists of first, second and third order logic. The selection of clauses in a logical structure is set by random. Besides that, the use of HTAF is important in this study since we want to generate correct final neuron states. The characteristics of HTAF that were differentiable and nonlinear is important in constructing the proposed model as HTAF is a mathematical function that attached to the neurons. The nonlinear HTAF helps the proposed model to learn complex patterns in data. Additionally, the proposed model still operates if no activation function is utilized. However, the proposed model unable to learn the pattern of classification problem which is the same as a linear classifier. The effectiveness of this paper will be tested by comparing two models: HNN-RAN3SATEA and HNN-RAN3SATES in terms of accuracy and efficiency. The parameter assignments in HNN-RAN3SATEA and HNN-RAN3SATES models are summarized in Tables 1 and 2.

Table 1. List of parameters utilized in HNN-RAN3SATEA model.

Parameter	Parameter Value
Neuron combination (c)	100
Number of Trials (t)	100
Maximum Number of Iterations	100,000
Size of population (N_{pop})	120
Number of parties (N_{party})	4
Positive advertisement rate (σ^p)	0.5
Negative advertisement rate (σ^n)	1
Tolerance value (Tol)	0.001
Threshold Time	1 day
Activation Function	Hyperbolic Tangent activation function (HTAF)
Initialization of neuron states	Random

Table 2. List of parameters utilized in HNN-RAN3SATES model.

Parameter	Parameter Value
Neuron combination (c)	100
Number of Trials (t)	100
Maximum Number of Iterations	100,000
Size of population (N_{pop})	100
Tolerance value (Tol)	0.001
Threshold Time	1 day

8. Performance Metric for HNN-RAN3SAT Models

Four types of matrices performance will be utilized to evaluate the efficiency of HNN-RAN3SAT models in terms of error analysis, energy analysis and similarity index. For error analysis, the evaluation for the learning and retrieval phase will be based on root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE). Meanwhile, global minima ratio (Z_m) is evaluated in energy analysis. Finally, Jaccard (J), Sokal Sneath (SS), Dice (D), and Kulczynski (K) are evaluated in similarity index. The purposes of evaluating each metric will be explained further below.

8.1. Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE)

Root mean square error (RMSE) and mean absolute error (MAE) has been widely used in evaluating the accuracy of the models by measuring the difference between the values predicted by the model and the target values. RMSE is sensitive to outliers, and it is more appropriate to use than the MAE when the model follows the normal distribution [42]. Note that, mean absolute percentage error (MAPE) is the average of absolute percentage errors. MAPE is highly adapted for predicting applications, especially in situations sufficient data is available [43]. In real-world applications, MAPE is frequently used when the amount to predict does not equal zero. This is due to the MAPE produces infinite values when the actual values are zero or close to zero, which is a common occurrence in some fields [44]. In the current study, the formula of calculating RMSE, MAE, and MAPE in the learning phase is modified as [27]:

$$RMSE_{\ Learning} = \sum_{i=1}^{n} \sqrt{\frac{1}{n}(f_{NC} - f_i)^2} \tag{31}$$

$$MAE_{\ Learning} = \sum_{i=1}^{n} \frac{1}{n}|f_{NC} - f_i| \tag{32}$$

$$MAPE_{\ Learning} = \sum_{i=1}^{n} \frac{100}{n}\frac{|f_{NC} - f_i|}{|f_{NC}|} \tag{33}$$

where f_{NC} is the highest fitness achieved in the network based on the HNN-RAN3SAT model, f_i is the fitness values that achieved during learning phase and n is the number of iterations before $f_{NC} = f_i$. Notably, i is the iterations generated by the simulation before achieved its full iteration, n or can be written as $i \leq n$. The RMSE, MAE, and MAPE in the testing phase will be expressed as [43]:

$$RMSE_{\ Testing} = \sum_{i=1}^{n} \sqrt{\frac{1}{n}(P_i - O_i)^2} \tag{34}$$

$$MAE_{\ Testing} = \sum_{i=1}^{n} \frac{1}{n}|P_i - O_i| \tag{35}$$

$$MAPE_{\ Testing} = \sum_{i=1}^{n} \frac{100}{n}\frac{|P_i - O_i|}{|P_i|} \tag{36}$$

where P_i is the target value (global energy) and O_i refers to the predicted value (calculated energy) by the proposed model.

8.2. Global Minima Ratio (Z_m)

Z_m is the ratio between the total global minimum energy and the total number of runs in the testing phase [22]. Z_m can be expressed as:

$$Z_m = \frac{1}{tc}\sum_{i=1}^{n} N_{H_i\ (P\ RAN3SAT)} \tag{37}$$

where t is the number of trials, c is the number of neuron combination and N_H is the number of global minimum energy of the proposed model. This measurement is good to compare the performance of the models since HNN-RAN3SATES when the number of variables increases the model fails to reach global minimum energy.

8.3. Similarity Index

A similarity index measures the relationship between final states of the neuron in the retrieval phase and benchmark neurons (the ideal neuron state) that can be defined as [18]:

$$S_i^{\max} = \begin{cases} 1, & if \quad A \\ -1, & if\ \neg A \end{cases} \tag{38}$$

Here, some similarity indices for neuron state that will be utilized in this study to distinguish global minimum solutions produced by HNN presented in Tables 3 and 4.

Table 3. Similarity indices used in the current study.

Similarity Index	The Formula
Jaccard (J)	$J = \frac{l}{l+m+n}$
Sokal Sneath (SS)	$SS = \frac{l}{l+2(m+n)}$
Dice (D)	$D = \frac{2l}{2l+m+n}$
Kulczynski (K)	$K = 0.5\left(\frac{l}{l+m} + \frac{l}{l+n}\right)$

Table 4. Neuron state for parameters in similarity index.

Parameter	$S_i^{\max}$	S_i
l	1	1
m	1	−1
n	−1	1
o	−1	−1

The neuron variation of HNN model is the number of solutions can be produced in each neuron combination and can be calculated by Equations (39) and (40):

$$V = \sum_{i=1}^{\lambda} F_i \tag{39}$$

$$F_{i+1} = \begin{cases} 1, & x_{i+1} \neq x_i \\ 0, & x_{i+1} = x_i \end{cases} \tag{40}$$

where λ is the total number of solutions produced by the HNN model. x_i is the solution produced in the i-th trial.

9. Result and Discussion

Firstly, to test the performance of the proposed HNN-RAN3SAT model, the comparison between the learning algorithms of EA and ES is analyzed in each phase. Various evaluation metrics utilized to study the influence of different learning algorithms in the HNN-RAN3SAT model. All compared learning algorithms are conducted independently on the simulated datasets generated randomly by Dev C++. Further analysis on the performance of HNN-RAN3SATEA and HNN-RAN3SATES revolves around the learning phase and retrieval phase performance analysis.

9.1. Learning Phase Performance

This section evaluates the capability of the proposed EA in doing $P_{RAN3SAT}$ in HNN (HNN-RAN3SATEA). The proposed model will be compared with the benchmark learning method proposed by [28] (HNN-RAN3SATES). The main emphasis of this comparison is to evaluate whether the proposed learning method can embed the behavior of $P_{RAN3SAT}$ into HNN model. Inspired by works such as [45,46], this section will utilize learning error metrics such as Root Mean Square Error $\left(RMSE_{Learning}\right)$, Mean Absolute error $\left(MAE_{Learning}\right)$ and Mean Absolute Percentage Error $\left(MAPE_{Learning}\right)$.

Figures 3–5 demonstrate the $RMSE_{Learning}$, $MAE_{Learning}$ and $MAPE_{Learning}$ results for HNN-RAN3SATES and HNN-RAN3SATEA respectively for $NN = 18$ until $NN = 300$ neurons. The graphs emphasize the accuracy of the learning phase of the proposed model as opposed to the conventional model, where lower error evaluations indicating higher accuracy. Generally, similar trends observed in Figures 3–5, where HNN-RAN3SATES recorded the highest measures as compared to HNN-RAN3SATEA for every NN. Based on the results, HNN-RAN3SATEA outperformed HNN-RAN3SATES when the number of neurons increased. According to Figure 3, the deviation of the error is much lower as the number of neurons increased. Note that the high value of $RMSE_{Learning}$ manifests the significant gap between the average fitness difference of the HNN-RAN3SAT model with the lowest fitness value. Despite acquiring single objective function $\left(E_{P_{RAN3SAT}} = 0\right)$ during learning phase, EA managed to complete the learning phase with lower value of $RMSE_{Learning}$ compared to the conventional method. Note that $P_{RAN3SAT}$ consists of C_i^k where $k = 1, 2$ and 3 that requires effective state update. By increasing $n\left(C_i^k \neq 1\right)$ in $P_{RAN3SAT}$ will create a fitness gap between optimal string fitness and the current fitness. In terms of probability of obtaining $E_{P_{RAN3SAT}} = 0$, $C_i^{(3)}$ is observed to achieve satisfied clause or $P\left(E_{P_{3SAT}} = 0\right) > P\left(E_{P_{2SAT}} = 0\right)$ but will result in more entry for synaptic weight structure. To make matters worse, probability for $P\left(C_i^{(1)} = 1\right)$ reduced exponentially as the number of $C_i^{(1)} \to \infty$. In this case, the high $RMSE_{Learning}$ value is due to high number of $n\left(C_i^{(1)}\right)$ in $P_{RAN3SAT}$. Next, to effectively update all the state combination in $C_i^{(k)}$, HNN-RAN3SATEA employed positive advertisement which accelerates the learning to achieve $E_{P_{RAN3SAT}} = 0$. The potential neuron space was divided into several parties where the eligibility distance coefficient ω increases both fitness of the most eligible candidate (highest fitness) and the least eligible candidate (lowest fitness). If the fitness gap is high (between the candidate and the voter), the next positive advertisement campaign will further reduce the gap until the best candidate is found. Another interesting observation is the most eligible candidate that achieve $\left(E_{P_{RAN3SAT}} = 0\right)$ will assist other candidates in the party via update of ω. A similar mechanism was employed in other parties that potentially reduce the fitness gap between all solution in HNN-RAN3SATEA. However, the proposed positive advertisement in HNN-RAN-3SATEA has the capacity to reduce the eligibility of the voters or candidate but this effect is not obvious because there is no $RMSE_{Learning}$ fluctuation for all values of NN. Unlike other metaheuristics algorithms that consist only 1 single space such as Genetic Algorithm, EA has additional parties with different updating rule. Hence, the fitness gap between the least eligible candidate with optimal fitness can be reduced which will result in lower value of $RMSE_{Learning}$. Based on Figure 3, $RMSE_{Learning}$

value for HNN-RAN3SATEA remains low although the number of neurons increases to $NN \geq 200$. This behavior has been reported in the work of [27] where EA effectively update the neurons state of lower order logical rule. Additionally, for HNN-RAN3SATES, the solution search space for this model is not well-defined and there is a high chance for the model to achieve non-improving solution or local maxima. This is due to very minimal effort by the model to improve the fitness of the current solution string. With the increase of number of neurons, HNN-RAN3SATES has irregular space that will increase the complexity of the method to 2^n. The increase of complexity will result in high value of $RMSE_{Learning}$. The high value of $RMSE_{Learning}$ demonstrates low effectiveness of the proposed method in completing the learning phase of HNN.

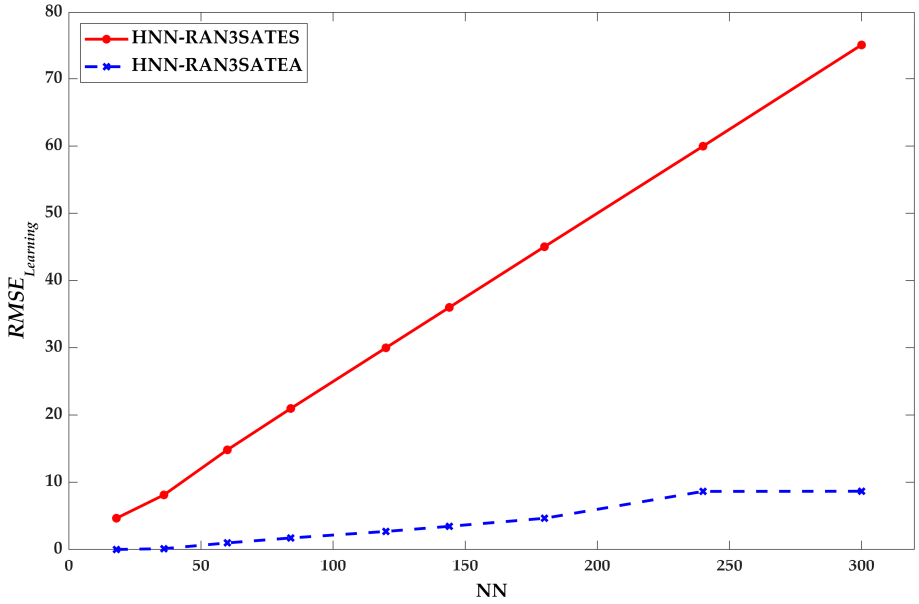

Figure 3. $RMSE_{Learning}$ for HNN-RAN3SAT model.

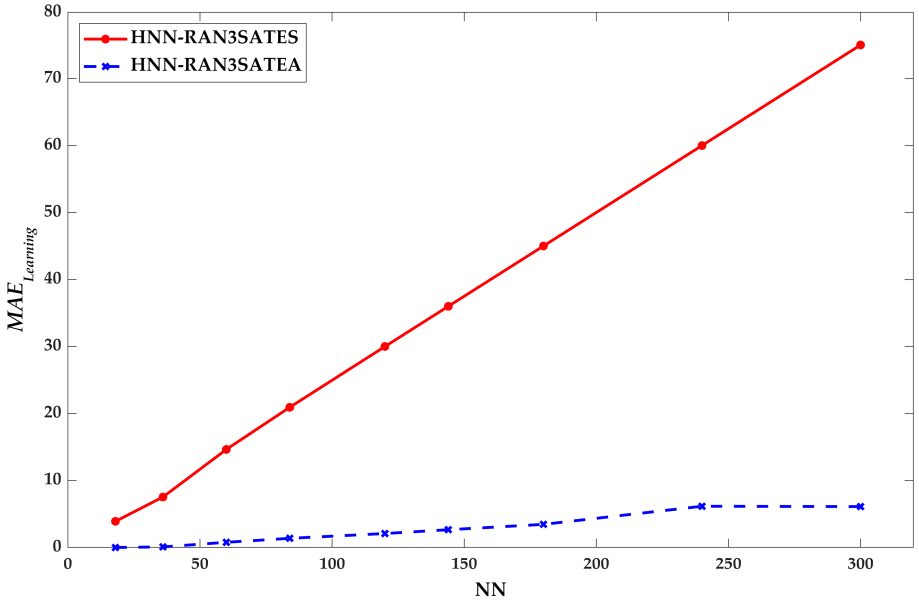

Figure 4. $MAE_{Learning}$ for HNN-RAN3SAT model.

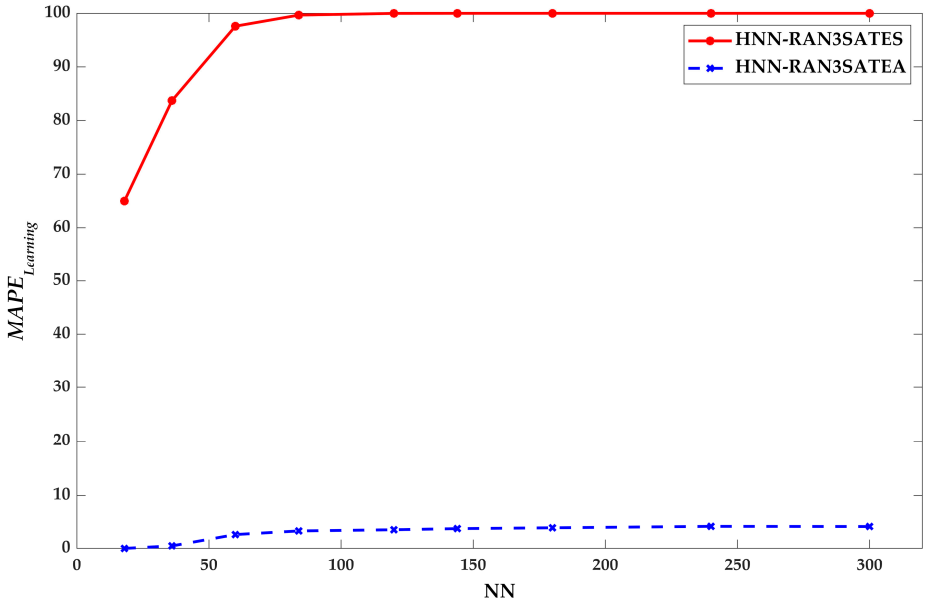

Figure 5. $MAPE_{Learning}$ for HNN-RAN3SAT model.

On the other hand, according to Figure 4, the absolute error for HNN-RAN3SATEA is much lower as the number of neurons increased. Note that, the high value of $MAE_{Learning}$ signifies the actual difference between the average fitness and the maximum fitness of the HNN-RAN3SAT model. The key strategy by HNN-RAN3SATEA to maintain a low $MAE_{Learning}$ value is the negative advertisement. Negative advertisement capitalizes the exploration capability to identify the potential optimal candidates. During negative advertisement, the composition of the voters in HNN-RAN3SATEA changed where eligible candidates have been attracted to the stronger party. The change of position of the voter will increase the chances for the voter to increase eligibility because the stronger party has a higher eligibility value. Thus, individual eligibility for all the voters and candidates will increase and the absolute error reduced dramatically. Indirectly, negative advertisement divides the composition of the strong party that has less of $n\left(C_i^{(1)} \neq 1\right)$ compared to weak party that has more of $n\left(C_i^{(1)} = 1\right)$. Although there is potential $n\left(C_i^{(2)} \neq 1\right)$ and $n\left(C_i^{(3)} \neq 1\right)$ during the change of position but the $MAE_{Learning}$ value suggested most of the error accumulated from the $C_i^{(1)}$ that is very sensitive to the state change. This finding has a good agreement with the work of [27,28] where $C_i^{(1)}$ is the main Satisfiability issue in Random Satisfiability. Note that, third order clause $C_i^{(3)}$ in $P_{RAN3SAT}$ was reported to be easily satisfied due to more state option, thus showing the potential $MAE_{Learning}$ value for $C_i^{(3)}$ is almost zero. It is worth mentioning that negative advertisement will improve the candidate of the party because the new positioning of the voters into a new party will change the solution space of the party. For example, if one of the voters that has $E_{P_{RAN3SAT}} = 4$ was attracted to other party in which the candidate of the party has $E_{P_{RAN3SAT}} = 2$, the change of candidate will occur. The new positioning of the candidate will improve the overall eligibility of the party. Another scenario to consider is when all the candidates from the weak party move to strong party and the overall eligibility of the candidate in the weak party decreases, thus increasing the $MAE_{Learning}$ value. In this case, the weak party will be slowly disregarded during the next trial of HNN-RAN3SATEA. As of state update, negative advertisement allows a similar update to positive advertisement. The update will increase the individual fitness in all parties and hence, reduce the value of $MAE_{Learning}$. Additionally, HNN-RAN3SATES does not acquire any systematic exploration method. The fitness value for each individual solution string varies from near-optimal fitness to very low fitness without any potential improvement. There is a high chance

that both $C_i^{(2)}$ and $C_i^{(3)}$ will not satisfied during the next iteration. The inefficient state exploration will result in high $MAE_{Learning}$ value. In short, the result in Figure 4 confirms that HNN-RAN3SATEA offers great optimization capability to diversify the candidates and provide more chances for the individual voter to improve with respect to the current candidate. The $MAE_{Learning}$ value demonstrates the compatibility of the $C_i^{(3)}$ as a logical rule in HNN as the number of neuron increase.

As shown in Figure 5, the percentage error for HNN-RAN3SATEA is lower compared to HNN-RAN3SATES as the number of neurons increased. One of the key hindrances of learning $P_{RAN3SAT}$ in HNN is the random assignment of the negated literal. Negated literal in $C_i^{(k)}$ mostly impact first and second order logic where the probability to obtain $C_i^{(k)} \neq 1$ is high. Note that, the high value of $MAPE_{Learning}$ signifies a high percentage of unsatisfied clause of $P_{RAN3SAT}$. HNN-RAN3SATEA managed obtain lower value of $MAPE_{Learning}$ compared to HNN-RAN3SATEA because the effectiveness of Coalition to reduce the potential local maxima during the learning phase. According to Figure 5, the $MAPE_{Learning}$ value when $NN \leq 50$ is almost zero compared to HNN-RAN3SATES that reaches 60–70% percentage of unsatisfied clauses. This implies the effectiveness of the HNN-RAN3SATEA in discovering other solution space if the HNN achieves the non-improving solution. During Coalition, eligibility of the candidate in the strong parties improved by combining the effect of distance coefficient from another strong party. In this case, the eligibility of the candidate will increase with respect to other eligible candidates from other parties. In other words, Coalition expedites the process of finding neuron state that corresponds to $E_{P_{RAN3SAT}} = 0$ during learning phase. In summary, HNN-RAN3SATEA managed to complete the learning phase for all values of NN compared to HNN-RAN3SATES that achieve 100% percentage error when $NN \leq 55$.

9.2. Retrieval Phase Performance

This section encompasses the analysis of the retrieval phase for the proposed model, HNN-RAN3SATEA with the benchmark model, HNN-RAN3SATES as proposed by [28] in terms of error evaluation, energy analysis, neuron variation and similarity analysis measure. The testing error evaluations are based on the RMSE, MAE and MAPE to comply with learning error evaluation. The energy evaluation will utilize the ratio of global solution as coined by the work of [18]. In terms of similarity index (SI), several performance metrics such as Jaccard (J), Sokal Sneath (SS), Dice (D) and Kulczynski Index (K) inspired by the work of [28] will be implemented in this study. Building from the similarity aspects, the total variation V also will be considered to assess the quality of the final neuron state. As of physical meaning for each figure, Figures 6–8 demonstrate the accuracy of the testing phase (retrieval phase) of our proposed model with respect to the benchmark model. Lower error measures the effectiveness of the HNN-RAN3SAT model to generate the final state that achieved global minimum energy. The numerical comparison of the number of global minimum energy in the form of ratio is shown in Figure 9. Figures 10–13 demonstrate the similarity indices of the final states obtained by the HNN-RAN3SAT model. The higher index depicts more overfitting behavior of the final state produced by HNN-RAN3SAT model. As of Figure 14, the value V in the graph indicates the effectiveness of the HNN-RAN3SAT model in retrieving different final state from each other. In other words, high value of V indicates HNN-RAN3SAT model produce non-repeating final neuron state which leads to more diversified neuron states.

According to Figures 6–8, the error values that were reported for HNN-RAN3SATEA are $RMSE_{testing} = 0$, $MAE_{testing} = 0$ and $MAPE_{testing} = 0$ for $18 \leq NN \leq 300$. This is due to the effective synaptic weight management that correspond to the "learned" $P_{RAN3SAT}$ during learning phase of HNN-RAN3SATEA. The capability of local search and global search operator in EA has successfully creates partitioning of the search spaces during learning phase, which improves the synaptic weight management during retrieval phase. However, the error measures for HNN-RAN3SATES model are growing rapidly, particularly when $NN \geq 84$. The non-effective synaptic weight management in ES reduces the

correctness of synaptic weights being retrieved during the retrieval phase. The outcomes will drive into the non-optimal states being retrieved and thus, HNN-RAN3SAT model starts retrieving the local minimum energy (refer Figure 9). Conversely, HNN-RAN3SATEA model consistently reaches the global minimum energy, and this is explained by the results in Figures 6–8 which means HNN-RAN3SAT recalls the correct final states.

The results in Figure 9 demonstrate the performance of the HNN-RAN3SATEA with respect to HNN-RAN3SATES based on the ratio of global minimum solutions, Z_m. In addition, the consistent trend of HNN-RAN3SATEA which exhibit $Z_m = 1$ has verified the performance of HNN-RAN3SAT model in term of energy analysis. According to [28], the optimal synaptic weight management in EA contributes to the optimal testing phase in retrieving the consistent final neuron states. Therefore, the conditions such as the optimal final neuron states and correct synaptic weight, were the effect of the optimization operators in EA. Based on Figure 9, the fluctuations can be seen throughout the simulations for HNN-RAN3SATES except for $NN \geq 180$, where the suboptimal states and non-systematic synaptic weight management have deteriorated the Z_m values. With this analysis, we can highlight the relationship between the ineffective learning by ES has influenced the learning phase, particularly when $NN \geq 180$. The mechanism of ES, which deploys the intensive explorations within a larger search space, causing an ineffective learning phase that eventually affects synaptic weight management [46]. There will be a the possibility of retrieving non-optimal synaptic weight during the retrieval phase. Also, it needs to mention that in terms of solution space, EA approached the most systematic partition solution space in the model, which improves global and local search and finds the solution in all defined space. In contrast, ES has no effort of improving the global-local search in its solution space.

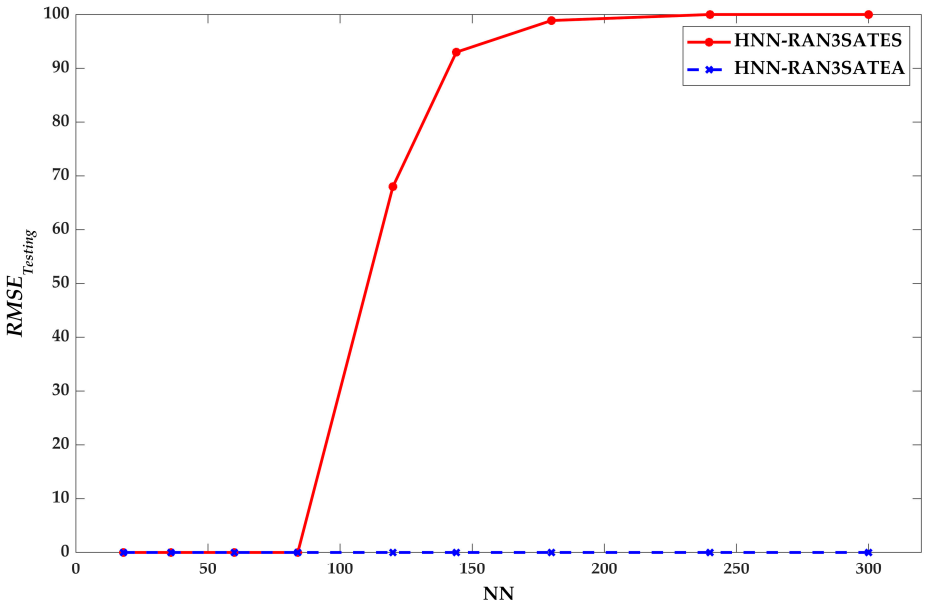

Figure 6. $RMSE_{Testing}$ for HNN-RAN3SAT model.

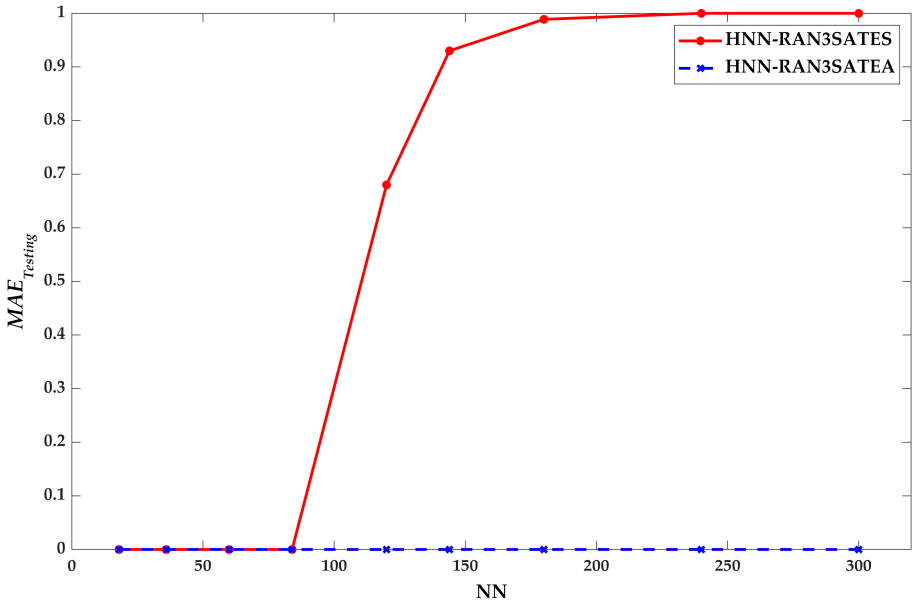

Figure 7. $MAE_{Testing}$ for HNN-RAN3SAT model.

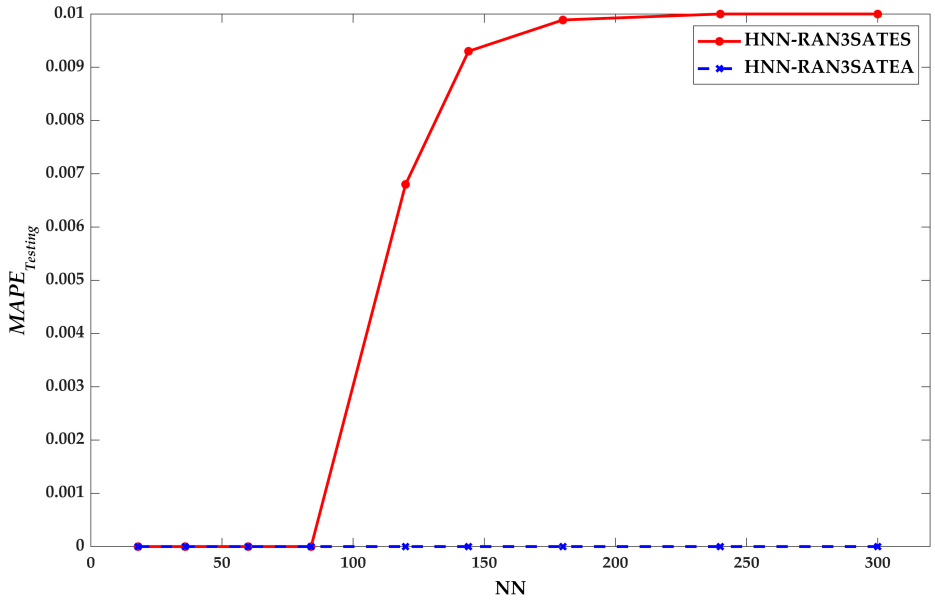

Figure 8. $MAPE_{Testing}$ for HNN-RAN3SAT model.

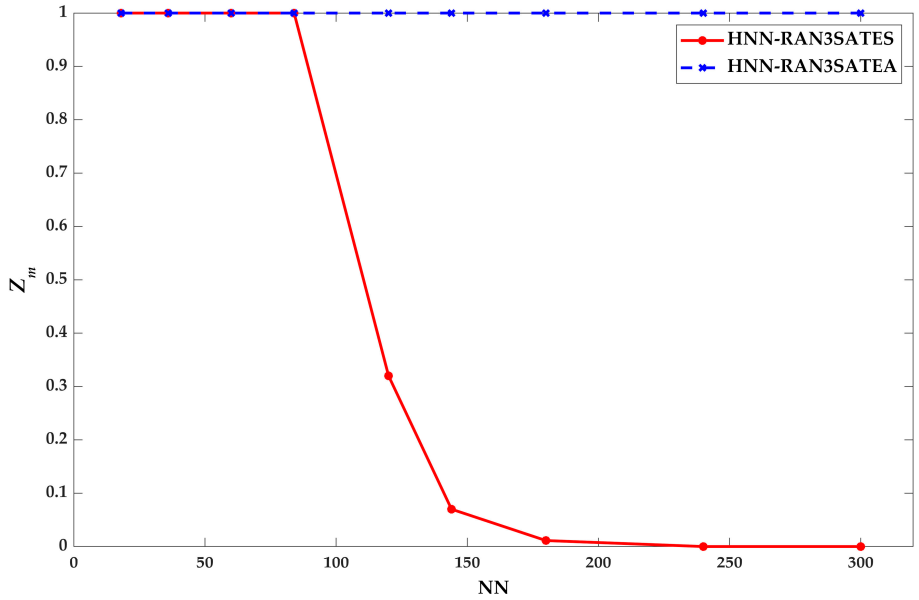

Figure 9. Z_m for HNN-RAN3SAT model.

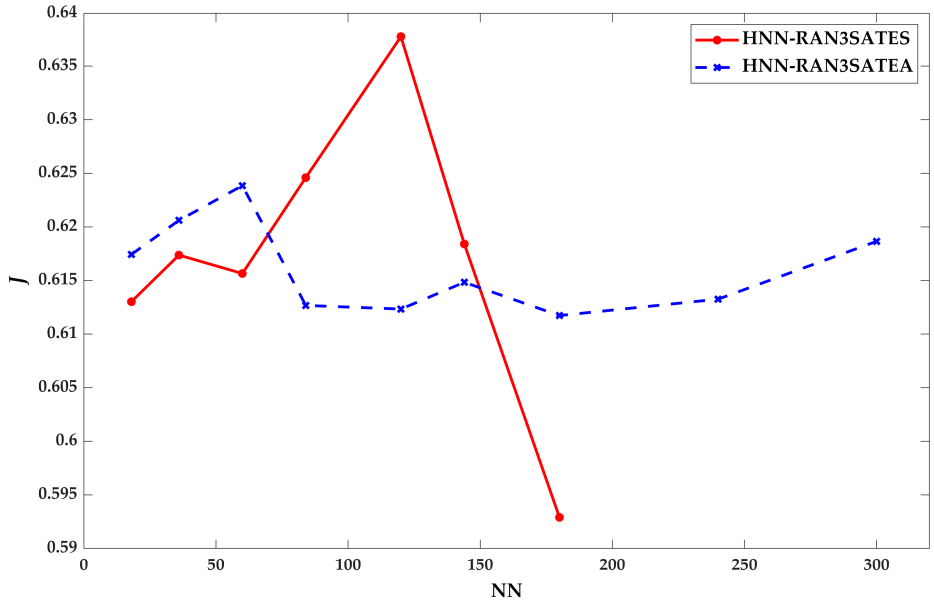

Figure 10. J (Jaccard) for HNN-RAN3SAT model.

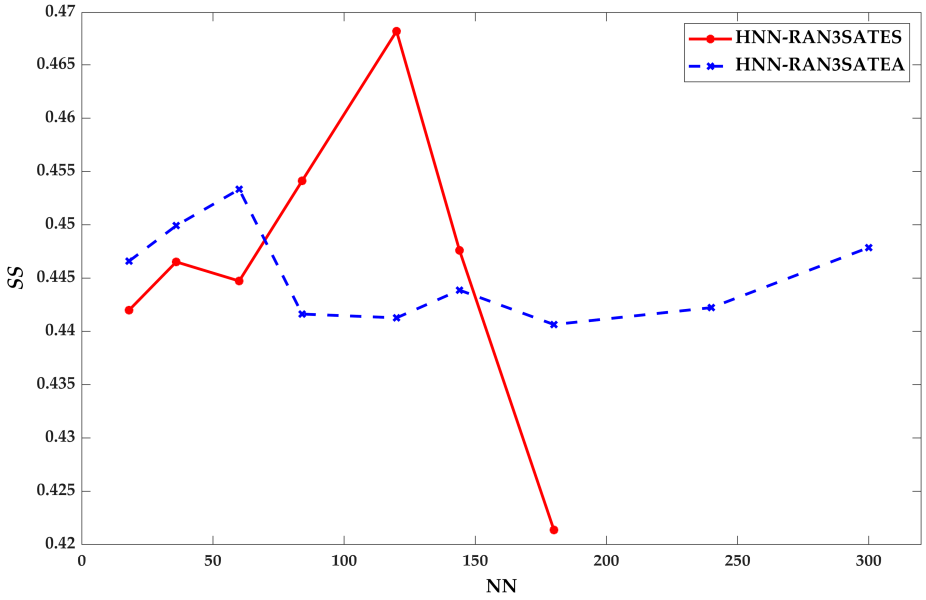

Figure 11. *SS* (Sokal Sneath) for HNN-RAN3SAT model.

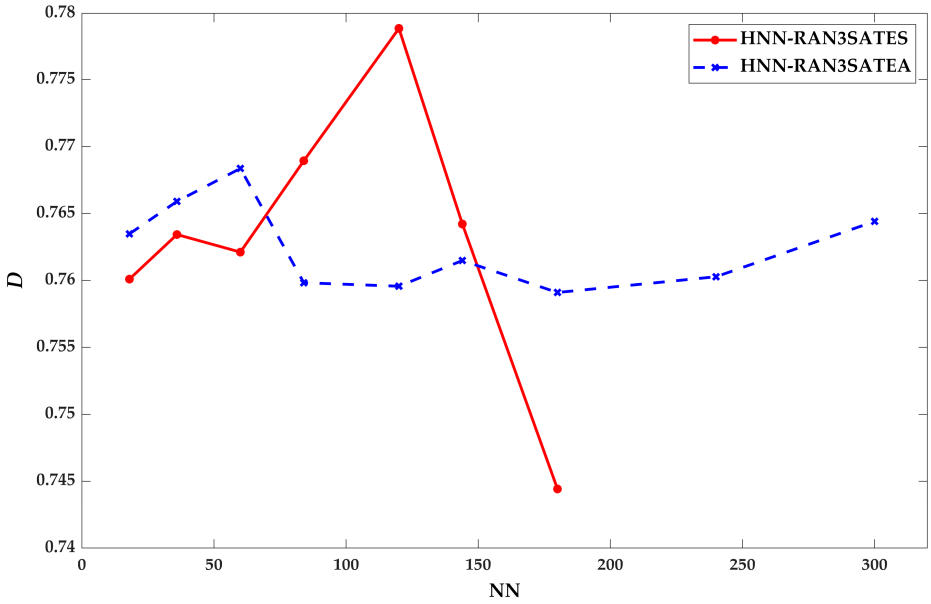

Figure 12. *D* (Dice) for HNN-RAN3SAT model.

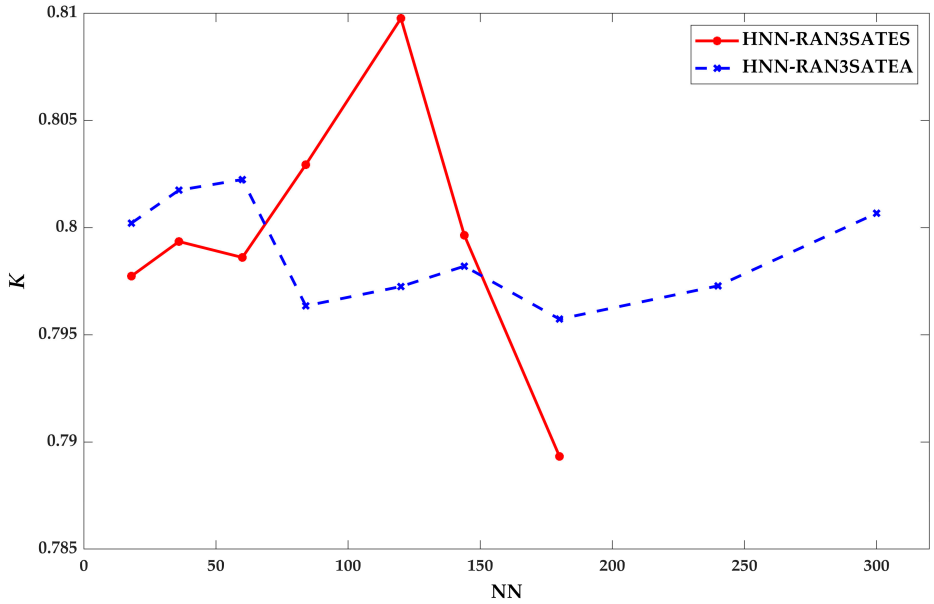

Figure 13. *K* (Kulczynski) for HNN-RAN3SAT model.

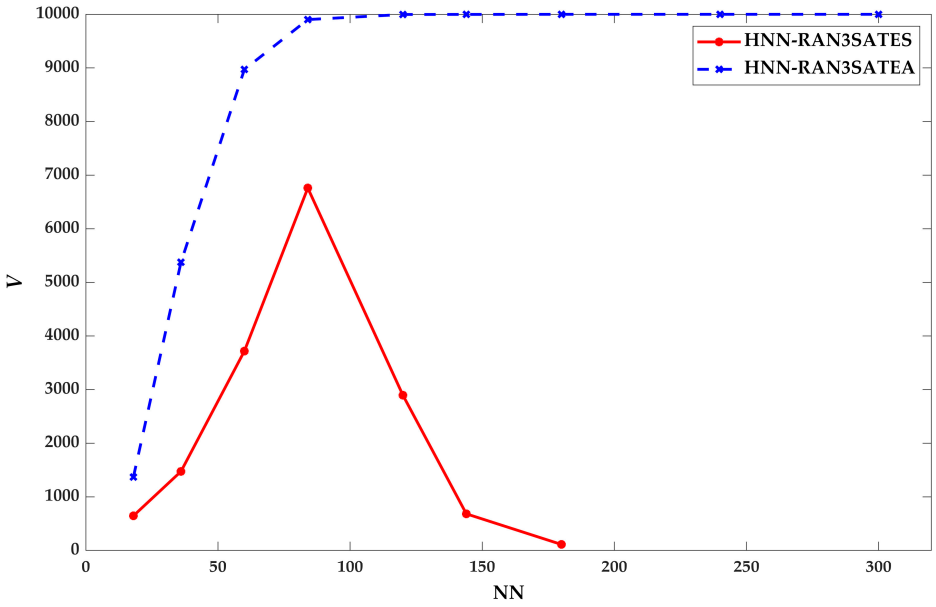

Figure 14. *V* (Neuron Variation) for HNN-RAN3SAT model.

Furthermore, to investigate the quality of the solutions produced by HNN-RAN3SAT, similarity indices were calculated to compare each neuron of the string with the benchmark neuron that is determined by Equation (38) [28]. The general trend for Jaccard, Sokal Sneath, Dice and Kulczynski recorded by HNN-RAN3SATES as shown in Figures 10–13 were increasing until the peak at $NN = 120$, and stop getting any value when $NN > 180$ due to the presence of more local solutions. To support the analysis, the value of the global minima ratio Z_m that approaches zero within the aforementioned range. This indicates the variables of each state for $P_{RAN3SAT}$ (neurons of the string) obtained are not all equal the ideal neuron. HNN-RAN3SATEA model has similarity index values ranging from 0.4–0.8 and managed to generate the results for $18 \leq NN \leq 300$. The most interesting to ponder here is the difference in the similarity index values achieved in a specific number of neurons. For example, when $NN = 120$, the similarity analysis were $J = 0.63775$, $SS = 0.46816$, $D = 0.778813$ and $K = 0.809763$. Lower similarity index was found to be recorded based on the J and SS due to lower similarity values of the positive state

as the positive benchmark states. This indicates the final states generated exhibits less overfitting, affecting the neuron variations. The effective synaptic weight management has been contributed due to the effectiveness of the local search and global search operators in EA as coined by [27]. The number of neuron variations as shown in Figure 14 also is very crucial to illustrate how HNN-RAN3SATEA can recall different solutions in each trial which means exploring different solutions and different areas in the search space. By taking a direct comparison towards V when $NN = 120$, where $V = 10,000$ has manifested the lower similarity index reflects the higher variability of the final solutions. On contrary, the fluctuation in trend of V can be seen in HNN-RAN3SATES where the model stops getting any value when $NN > 180$. This is due to the non-effective synaptic weight management and training via ES, where the exploration being done in a tremendous search space, without any intervention of optimization operators [46].

Overall, HNN-RAN3SATEA outperformed the HNN-RAN3SATES in terms of the solution quality based on similarity analysis and the variability according to total variation measures. The final neuron states attained by the proposed model are certified to be less over-fit based on the lower similarity index and higher neuron variation achieved at the end of the simulations. In addition, the error evaluation and energy analysis also demonstrated the capability of HNN-RAN3SATEA in generating the global minimum solutions that correspond to the global minimum energy. The analysis also has verified the compatibility of $P_{RAN3SAT}$ with EA in HNN logic programming, with a promising potential to be applied in the logic mining [45] paradigm in the next exploration.

10. Conclusions

The main findings of this research prove the compatibility of $P_{RAN3SAT}$ logical representation analysis via Election Algorithm (EA) with HNN in both learning and retrieval phase based on error evaluations, energy analysis, similarity measures, and variability. The results via computer simulation have extensively shown that the formulated propositional logical rule $P_{RAN3SAT}$ consists of first, second, and third order logical rule has been successfully embedded optimally in Hopfield Neural Network, indicating the flexibility of the logical representation. Another finding of this research is that EA is an effective method for tackling $P_{RAN3SAT}$ as well as RAN2SAT in work [27]. In the context of learning algorithm, the results have manifested the capability of EA during the learning phase of $P_{RAN3SAT}$ logic in HNN as opposed to the conventional algorithm namely, Exhaustive Search (ES) by the learning accuracy and efficiency. In addition, particularly the proposed model has demonstrated the capacity in optimizing the retrieval phase based on the performance indicators such as error measures, energy analysis, similarity index, and variability evaluation. Based on the analysis, all research objectives have been achieved after conducting the simulations.

The research raises important questions about the role of EA that has verified the capability of the higher-order propositional logic in HNN, specifically referring to the proposed non-systematic RAN3SAT logic. Based on the promising results obtained by encoding $P_{RAN3SAT}$ in HNN with EA, the new logical structure RAN3SAT can be further applied in logic mining due to its flexibility to represent non-systematic data. Additionally, the proposed logical structure can be extended as a main logic mining representation of various data ranging from medical, engineering, and finance data set. The potential performance metrics such as median absolute deviation (MAD) [47], area under curve (AUC) [48], F1 score [49], logarithmic loss [50], and dissimilarity index analysis can be potentially considered in future work to enhance the analysis.

Author Contributions: Conceptualization, methodology, writing—original draft preparation, M.M.B.; validation, S.Z.M.J.; formal analysis, N.E.Z.; investigation, resources, funding acquisition, M.S.M.K.; writing—review and editing, M.A.M.; visualization, A.A.; project administration, S.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Fundamental Research Scheme Grant (FRGS), grant number 203/PMATHS/6711804.

Acknowledgments: The authors would like to express special dedication to all of the researchers from AI Research Development Group (AIRDG) for the continuous support.

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

References

1. Hopfield, J.J.; Tank, D.W. "Neural" computation of decisions in optimization problems. *Biol. Cyber.* **1985**, *52*, 141–152.
2. Hemanth, D.J.; Anitha, J.; Mittal, M. Diabetic retinopathy diagnosis from retinal images using modified Hopfield neural network. *J. Med. Syst.* **2018**, *42*, 1–6. [CrossRef]
3. Channa, A.; Ifrim, R.C.; Popescu, D.; Popescu, N. A-WEAR Bracelet for detection of hand tremor and bradykinesia in parkinson's patients. *Sensors* **2021**, *21*, 981. [CrossRef]
4. Channa, A.; Popescu, N.; Ciobanu, V. Wearable solutions for patients with parkinson's disease and neurocognitive disorder: A systematic review. *Sensors* **2020**, *20*, 2713. [CrossRef]
5. Veerasamy, V.; Wahab, N.I.A.; Ramachandran, R.; Madasamy, B.; Mansoor, M.; Othman, M.L.; Hizam, H. A novel rk4-Hopfield neural network for power flow analysis of power system. *Appl. Soft Comput.* **2020**, *93*, 106346. [CrossRef]
6. Chen, H.; Lian, Q. Poverty/investment slow distribution effect analysis based on Hopfield neural network. *Future Gener. Comput. Syst.* **2021**, *122*, 63–68. [CrossRef]
7. Cook, S.A. The Complexity of Theorem-Proving Procedures. In Proceedings of the Third Annual ACM Symposium on Theory of Computing, New York, NY, USA, 3 May 1971; Association for Computing Machinery: New York, NY, USA, 1971; pp. 151–158.
8. Fu, H.; Xu, Y.; Wu, G.; Liu, J.; Chen, S.; He, X. Emphasis on the flipping variable: Towards effective local search for hard random satisfiability. *Inf. Sci.* **2021**, *566*, 118–139. [CrossRef]
9. Lagerkvist, V.; Roy, B. Complexity of inverse constraint problems and a dichotomy for the inverse satisfiability problem. *J. Comput. Syst. Sci.* **2021**, *117*, 23–39. [CrossRef]
10. Luo, J.; Hu, M.; Qin, K. Three-way decision with incomplete information based on similarity and satisfiability. *Int. J. Approx. Reason.* **2020**, *120*, 151–183. [CrossRef]
11. Hitzler, P.; Hölldobler, S.; Seda, A.K. Logic programs and connectionist networks. *J. Appl. Log.* **2004**, *2*, 245–272. [CrossRef]
12. Abdullah, W.A.T.W. Logic programming on a neural network. *Int. J. Intell. Syst.* **1992**, *7*, 513–519. [CrossRef]
13. Pinkas, G. Symmetric neural networks and propositional logic satisfiability. *Neural Comput.* **1991**, *3*, 282–291. [CrossRef] [PubMed]
14. Abdullah, W.A.T.W. Logic programming in neural networks. *Malays. J. Comput. Sci.* **1996**, *9*, 1–5. [CrossRef]
15. Hamadneh, N.; Sathasivam, S.; Choon, O.H. Higher order logic programming in radial basis function neural network. *Appl. Math. Sci.* **2012**, *6*, 115–127.
16. Mansor, M.A.; Jamaludin, S.Z.M.; Kasihmuddin, M.S.M.; Alzaeemi, S.A.; Basir, M.F.M.; Sathasivam, S. Systematic Boolean satisfiability programming in radial basis function neural network. *Processes* **2020**, *8*, 214. [CrossRef]
17. 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]
18. Kasihmuddin, M.S.M.; Mansor, M.A.; Basir, M.F.M.; Sathasivam, S. Discrete mutation Hopfield neural network in propositional satisfiability. *Mathematics* **2019**, *7*, 1133. [CrossRef]
19. Kasihmuddin, M.S.M.; Mansor, M.A.; Sathasivam, S. Hybrid genetic algorithm in the Hopfield network for logic satisfiability problem. *Pertanika J. Sci. Technol.* **2017**, *25*, 139–152.
20. Mansor, M.A.; Kasihmuddin, M.S.M.; Sathasivam, S. Modified artificial immune system algorithm with Elliot Hopfield neural network for 3-satisfiability programming. *J. Inform. Math. Sci.* **2019**, *11*, 81–98.
21. 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.
22. Zamri, N.E.; Alway, A.; Mansor, A.; Kasihmuddin, M.S.M.; Sathasivam, S. Modified imperialistic competitive algorithm in Hopfield neural network for boolean three satisfiability logic mining. *Pertanika J. Sci. Technol.* **2020**, *28*, 983–1008.
23. Emami, H.; Derakhshan, F. Election algorithm: A new socio-politically inspired strategy. *AI Commun.* **2015**, *28*, 591–603. [CrossRef]
24. Lv, W.; He, C.; Li, D.; Cheng, S.; Luo, S.; Zhang, X. Election campaign optimization algorithm. *Procedia Comput. Sci.* **2010**, *1*, 1377–1386. [CrossRef]
25. Pourghanbar, M.; Kelarestaghi, M.; Eshghi, F. EVEBO: A New Election Inspired Optimization Algorithm. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 916–924.
26. Emami, H. Chaotic election algorithm. *Comput. Inform.* **2019**, *38*, 1444–1478. [CrossRef]

27. Sathasivam, S.; Mansor, M.A.; Kasihmuddin, M.S.M.; Abubakar, H. Election algorithm for random k satisfiability in the Hopfield neural network. *Processes* **2020**, *8*, 568. [CrossRef]
28. 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.
29. Li, C.M.; Xiao, F.; Luo, M.; Manyà, F.; Lü, Z.; Li, Y. Clause vivification by unit propagation in CDCL SAT solvers. *Artif. Intell.* **2020**, *279*, 103197. [CrossRef]
30. Kasihmuddin, M.S.M.; Sathasivam, S.; Mansor, M.A. Hybrid Genetic Algorithm in the Hopfield Network for Maximum 2-Satisfiability Problem. In Proceedings of the 24th National Symposium on Mathematical Sciences: Mathematical Sciences Exploration for the Universal Preservation, Kuala Terengganu, Malaysia, 27–29 September 2016; AIP Publishing: Melville, NY, USA, 2017; p. 050001.
31. Sathasivam, S. Upgrading logic programming in Hopfield network. *Sains Malays.* **2010**, *39*, 115–118.
32. Alway, A.; Zamri, N.E.; Kasihmuddin, M.S.M.; Mansor, A.; Sathasivam, S. Palm oil trend analysis via logic mining with discrete Hopfield Neural Network. *Pertanika J. Sci. Technol.* **2020**, *28*, 967–981.
33. Jamaludin, S.Z.M.; Kasihmuddin, M.S.M.; Ismail, A.I.M.; Mansor, M.A.; Basir, M.F.M. Energy based logic mining analysis with Hopfield Neural Network for recruitment evaluation. *Entropy* **2021**, *23*, 40. [CrossRef]
34. Abdullah, W.A.T.W. The logic of neural networks. *Phys. Lett. A* **1993**, *176*, 202–206. [CrossRef]
35. Kumar, M.; Kulkarni, A.J. Socio-inspired optimization metaheuristics: A review. In *Socio-Culture. Inspired Metaheuristics*; Kulkarni, A.J., Singh, P.K., Satapathy, S.C., Kashan, A.H., Tai, K., Eds.; Springer: Singapore, 2019; Volume 828, pp. 241–266.
36. Blum, C.; Roli, A. Hybrid metaheuristics: An introduction. In *Hybrid Metaheuristics*; Blum, C., Aguilera, M.J.B., Roli, A., Sampels, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 114, pp. 1–30.
37. Nievergelt, J. Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In *International Conference on Current Trends in Theory and Practice of Computer Science*; Springer: Heidelberg/Berlin, Germany, 2000; pp. 18–35.
38. Mansor, M.A.; Kasihmuddin, M.S.M.; Sathasivam, S. Robust artificial immune system in the Hopfield network for maximum k-satisfiability. *Int. J. Interact. Multimed. Artif. Intell.* **2017**, *4*, 63–71. [CrossRef]
39. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef]
40. Mansor, M.A.; Sathasivam, S. Performance Analysis of Activation Function in Higher Order Logic Programming. In Proceedings of the 23rd Malaysian National Symposium of Mathematical Sciences (SKSM23), Johor Bahru, Malaysia, 24–26 November 2015; AIP Publishing: Melville, NY, USA, 2016; p. 030007.
41. Kho, L.C.; Kasihmuddin, M.S.M.; Mansor, M.A. Logic mining in league of legends. *Pertanika J. Sci. Technol.* **2020**, *28*, 211–225.
42. Chai, T.; Draxler, R.R. Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature. *Geosci. Model Dev.* **2014**, *7*, 1247–1250. [CrossRef]
43. Kim, S.; Kim, H. A new metric of absolute percentage error for intermittent demand forecasts. *Int. J. Forecast.* **2016**, *32*, 669–679. [CrossRef]
44. Myttenaere, A.D.; Golden, B.; Le Grand, B.; Rossi, F. Mean absolute percentage error for regression models. *Neurocomputing* **2016**, *192*, 38–48. [CrossRef]
45. Zamri, N.E.; Mansor, M.A.; Kasihmuddin, M.S.M.; Alway, A.; Jamaludin, S.Z.M.; Alzaeemi, S.A. Amazon employees resources access data extraction via clonal selection algorithm and logic mining approach. *Entropy* **2020**, *22*, 596. [CrossRef] [PubMed]
46. 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]
47. Miah, A.S.M.; Rahim, M.A.; Shin, J. Motor-imagery classification using riemannian geometry with median absolute deviation. *Electronics* **2020**, *9*, 1584. [CrossRef]
48. Liu, Y.P.; Li, Z.; Xu, C.; Li, J.; Liang, R. Referable diabetic retinopathy identification from eye fundus images with weighted path for convolutional neural network. *Artif. Intel. Med.* **2019**, *99*, 101694. [CrossRef] [PubMed]
49. Sedik, A.; Iliyasu, A.M.; El-Rahiem, A.; Abdel Samea, M.E.; Abdel-Raheem, A.; Hammad, M.; Peng, J.; El-Samie, F.A.; El-Latif, A.A.A. Deploying machine and deep learning models for efficient data-augmented detection of COVID-19 infections. *Viruses* **2020**, *12*, 769. [CrossRef] [PubMed]
50. Wells, J.R.; Aryal, S.; Ting, K.M. Simple supervised dissimilarity measure: Bolstering iForest-induced similarity with class information without learning. *Knowl. Inform. Syst.* **2020**, *62*, 3203–3216. [CrossRef]