Article,

MODIFIED ELECTION ALGORITHM IN ACCELERATING THE PERFORMANCE OF HOPFIELD NEURAL NETWORK FOR RANDOM *k*SATISFIABILITY

Hamza Abubakar[1], Saratha Sathasivam[2*], Mohd. Asyraf Mansor[3], and Mohd Shareduwan Mohd Kasihmuddin[4]

1. School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia; zeeham4u2c@yahoo.com;
2. School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia; saratha@usm.my
3. School of Distance Education, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia; asyrafman@usm.my;
4. School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia;shareduwan@usm.my;

* Correspondence: saratha@usm.my; Tel.: +6046532428

Abstract: Election Algorithm (EA) is a powerful metaheuristics model motivated by phenomena of the socio-political mechanism of the presidential election conducted in many countries. EA is selected as a topic of discussion due to its capability and robustness to carry out complex problems in the random-2SAT logic program. This paper utilizes a hybridized EA assimilated with the Hopfield neural network (HNN) in carrying out random logic program (HNN-R2SATEA). The efficiency of the proposed method was compared with the existing traditional exhaustive search (HNN-R2SATES) model and the recently introduced HNN-R2SATICA model. From the result obtained, clearly proven that based on our proposed hybrid model outperformed other existing model based on the Global Minima Ratio (ZM), Mean Absolute Error (MAE), Bayesian Information Criterion (BIC) and Execution Time (ET). The expected outcome portrays that the EA algorithm outperformed the other two algorithms in doing random-*k*SAT logic program. The results proved the robustness, effectiveness, and compatibility of the HNN-R2SATEA model.

Keywords: Hopfield Neural Networks; Election Algorithm; Imperialistic Competitive Algorithm; Exhaustive Search; Random Satisfiability; Logic Programming.

1. Introduction

Artificial neural networks (ANNs) belong to the family of the computational architectural-based model, viewed as equivalent to a brains' programming by imitating its design and attempts to mimic nervous system activity through which information is handled by the brains [1]. It consists of many basic processing components (called artificial neurons) that are loosely based on biological neurons. This learns about the relations between the processing elements and the system parameters through a cycle of adjustment. It contains a number of interconnected neurons that has a specific synaptic weight that essentially influences how much the neuron output can influence the input to the next neurons. Usually, neurons also have their own weight called “a bias” term that defines the neurons' effects on the network itself. The information is preserved in the weight and learned by the network. This process is done by several known algorithms [2]. There has been an enormous development in computational technology and increase interest in the possible application and use of artificial intelligence in various domains of applications. Recently, ANN is rated as one of the most significant and widespread branches of study in the field of computational sciences and artificial intelligence (AI). Essentially, ANN is computer-generated mathematical algorithms that

learn from regular data and extract the information from such data for meaningful and intelligence decision making. Trained ANN takes a very fundamental approach to the functioning of a small biological neural network. These are the digitalized biological brains prototype and can detect complex non-linear interactions between dependent as well as independent variables in data where human brains may not be detected. It can stimulate any function if the appropriate information is given. It is now commonly used in various medical and health disciplines, especially radiology and cardiology for detection purposes. Many scholars have applied ANN in medicine and clinical study for modelling the phermacoepidemiology and medical data mining [3].

It is very crucial to choose an appropriate framework for neural network training. Most predominantly Backpropagation (BP) algorithm is a gradient descent algorithm in error spaces that are most likely to be locally stuck and converged very slowly. There are many studies in the field of ANN training methods; this includes the work in [4] which studied the pattern retrieval Hopfield design analysis via genetic algorithm. The learning algorithms were considered as normal, robust and batch BP algorithms [5-6]. The improvement of the neural network BP algorithm as studied in [5]. ANNs "learn" several arrays of input-output mappings from observations by optimizing branch weights that connect the ANN nodes. In [6] the study on how to predict monthly streamflow assessed various training functions on network process has been presented. [7] proposed a learning approach and comparison with learning algorithms based on genetic algorithm fused in particle swarm optimization. In [8] a hybrid approach based on imperialist competition optimization incorporated in ANN for Fuzzy logic program algorithm for forecasting of reservoirs was proposed.[9] focused his research on Quantized neural networks: low accuracy weights and activations learning neural networks. The quantized weights and activations are used at train time to determine the gradients of the parameters. A novel hybrid solution has been suggested for global optimum path planning based on an ICA-trained neural network [10]. An examination of the fundamental insights into how the ICA evolved and how it extended to industrial engineering disciplines in [11]. In [12], a new method has been described for determining the learning rate of ANN which named as cyclical learning rates, which virtually eliminate the need to determine the best values and schedule for optimal learning rates experimentally. One of the major breakthroughs for AI and computational sciences is the neuro-symbolic computation that combines the benefit metaheuristics, Hopfield network and logic programming in finding an optimal solution of various optimization problems [4,12-14]. Neural-symbolic computing strives to incorporate the two most fundamental cognitive abilities, as projected by various scholars. The ability to learn and reason from what has been learned [14-16]. Neural-symbolic computation has consistently been an active research area, internalizing the benefits of novel and robust learning and the logical reasoning and interpretability of symbolic simplification of ANN [17-18]

Recent advances in neural-symbolic computation as a fundamental approach for applied machine learning and reasoning have been explored in [14]. The integration explains the usefulness of the technique by outlining the main characteristics of the methodology, the main convergence of neural processing with the symbolic representation of intelligence and reasoning that allows for the development of explainable AI systems. Neural-symbolic computation perspectives shed new light on the increasing need for interpretable and transparent AI systems [13-15].

The main idea of logic as a programming language in neural networks was introduced in [15] to serve, represent and interpret a problem. The motivating force behind it is the notion that a single formalism is adequate both for logic and computing, and that it subsumes computation. An algorithm may be known to consist of a logic component that defines the intelligence to be used in optimizing a given problems, and perhaps a control component that decides the problem-solving techniques by which that knowledge is utilized. So from the perspective of combinatorial optimization, it can be viewed as a problem. In [15], the logic programming idea was extended incorporating the competent logical mapping system or propositional knowledge through an asymmetric network of connectionists.

The proposed new symmetric connectionist network (SCN) has therefore attracted the interest of AI and computational scientist communities to combine the advantage of both logic program and neural network as a single network [16-18]. In [17], Wan Abdullah developed a method which

named as Wan Abdullah Method. It is a method used in calculating the synaptic weight of the Hopfield network corresponds to the suggested logical system, and the technique is still applicable particularly when working with a recurrent neural network. In [18] Wan Abdullah method was used for determining the synaptic weight of the Hopfield neural network for Horn logic program of the network. [19] upgraded the Horn logic programming by integrating the efficient relaxation method to generate the optimum final neuron states. The stochastic approach for Hopfield network model programming has been extended further by [20], which reduced neurons oscillations during the Hopfield network model recovery phase. The notion of logic programming merged with radial basis function neural network as a single model for computation has been achieved in [21]. The result reveals that RBFNN in logical programming inevitably worked well. Consequently, in Mean Field Theory [22] depict HNN logic programming flexibility. Furthermore, [23] launched Horn logical rules for the application of Hopfield neural network activation function. In [23] successfully implemented the logical rule *k*SAT which observed to be very closely linked to the HNN. The benefit of metaheuristics algorithms, such as GA, ICA, EA, etc, is that the neural network can be inferred at different stages such as weight training and adjustment, system adaptation for determining the number of layers, node transfer functions, retrieval phase and learning rules [24].

Recently, EA has emerged a new evolutionary metaheuristics algorithm that has been applied in finding an optimal solution to computational optimization and engineering application [25-26]. Unlike many other metaheuristics that are mainly inspired by swam or natural evolutionary process. EA is adopting the socio-political process of presidential elections conducted in many countries. The synthesis of metaheuristics and the Hopfield model have been proven effective in carrying out satisfiability as a logic programming [27-28]. Therefore, this research incorporates an election algorithm to complement the Hopfield neural network to accelerate the random-*k*SAT solution search process. HNN-R*k*SATEA stipulates the fusion of the Hopfield network and the Election algorithm in finding the optimal solution R-*k*SAT logic programming.

2. Boolean Satisfiability

The problem of whether a given propositional theorem has a satisfying statement of truth (SAT) is regarded as one of the first problems to be proved as an NP-complete problem. The SAT problem involves figuring a set of binary mapping that satisfy a set of constraints or prove that there is no such mapping [29]. SAT is a key issue in computational sciences and mathematical optimization problem as well as in many other areas of engineering and electronic design automation including other well known NP-complete problems such as timetabling problem, graph colourability, independent sets, circuit design verification vertex cover and Hamiltonian path [29-31]. Recently, the real-world application of SAT algorithm has improved dramatically due to its ability to solve large industrial SAT instances in a relative short instance of time [32]. On the one hand, local search algorithm has been used to solve a large random instance of SAT as well as some classes of industrial and practical instances of SAT [33-34]. Notwithstanding the SAT problem being NP-Complete [35], the SAT solver technology has been significantly improved over the past decade. This has culminated in the building of several successful SAT algorithms worthy of optimizing thousands of variables with many constraints. Such algorithms include discrete mutation [28], conflict-driven clause learning [36], Membrane computing [37], MiniSAT [38], branch and bound algorithm[39-40]. One of the primary goals of SAT algorithm is focus on reducing the computational complexity in the network. In the meantime, NP problems can be converted in polynomial time, and so the previous, potential and future research efforts to optimize NP problems. In this study, random -*k*SAT will be embedded as a logical rules in the HNN.

In term of HNN-R*k*SATEA, is a brand new model as there is no effort to apply the benefits of Election algorithm in accelerating the performance of HNN in finding the optimal solution to random-*k*SAT logic programming. The proposed hybrid model developed based on random-*k*SAT clauses. The main focus of this study is, therefore, to explore the feasibility of the hybrid election

algorithm incorporated in the neural Hopfield network model, based on the random-*k*SAT logical formula, therefore, HNN-R*k*SATEA will equate the efficacy of the proposed model with other current Hopfield models (HNN-R*k*SATES and HNN-R*k*SATICA).

3. Discrete Hopfield Neural Network

Hopfield, in an effort to model biological memory, proposed the so-called, Hopfield Associative Memory (HAM). Such a HNN model is based on the Mc-Culloch-Pitts paradigm of the artificial neuron [41]. In such an ANN, the state space constitutes the symmetric unit hypercube (i.e. the components of state vectors are {+1 or -1}. The associated convergence theorem ensures that in the serial mode of operation, the initial state converges to a stable state and in the fully parallel mode of operation [42]. In essence, the stable states are realized as the "memory states" of the associative memory. HNN was naturally interested in synthesizing a CAM, with certain "desired stable states" (that are preselected). It is a novel neural computational framework through the implementation of an auto-associative memory. They belong to recurrent or fully interconnected categories of neural networks. All neurons are connected to each other, but there is no self-recurrent connection between the neurons. HNN consists of *N* interconnected nodes; each of the nodes can be expressed by a simplified *Ising* variable of spin glass in statistical mechanics [43]. The model dynamical equation includes of the state-evolving equation calculated based on the following:

$$L_j = \begin{cases} 1, \ if \ \sum_k^N U_{jk} L_k(t) > \tau_j \\ -1, \quad otherwise \end{cases} \tag{1}$$

Where U_{jk} is the weight matrix going between *j* and *k* neurons, S_j defines the unit condition *k* and τ_j described the threshold function of neurons *j*. Several studies [28, 42-48] defined $\tau_j = 0$ to verify that the HNN always leads to a decrease in energy monotonically. Each time neuron was connected with U_{jk} ,the value of the connection will be preserved as a stored pattern in an interconnected vector where $U^{(1)} = [U_{jk}^{(1)}]_{n\times n}$ and $U^{(2)} = [U_{jk}^{(2)}]_{n\times n}$ for*N*-dimensional variable vectors $\tau = (\tau_1, \tau_2, ..., \tau_N)^T$ as observed in [36, 60] that the constraint of synaptic weight matrix $U^{(1)}$ and does not allow self-loop neuron connection $U_{jj}^{(2)} = U_{kk}^{(2)}, ..., = U_{ii}^{(2)} = 0$ and symmetrical neuron synaptic weight matrix $U_{kj}^{(2)} = U_{jk}^{(2)}$. HNN's energy dynamics and content-addressable memory function offers a versatile system with high capacity, error tolerance, rapid memory recovery and partial inputs [47-49]. To make it ideal for incorporation with combinatorial optimization such as SAT. HNN used the logical rule to instruct the network's activity based on the synaptic strength matrix. The logical formulation, in this scenario, consisting of variables vectors, is framed in form of *N* neurons. The implementation of the random-*k*SAT logical rule in HNN is translated in abbreviated terminology as HNN-R*k*SAT and the main objective is to minimize the logical discrepancies by reducing network cost function which can be represented eq. (2) as follows;

$$E_{P_{R-kSAT}} = \sum_{j=1}^{NC} \prod_{k=1}^{NV} C_{jk} \tag{2}$$

where NC & NV described the number of clauses generated and the number of variable vector in the solution space respectively. The inconsistencies of logical clause C_{jk} is provided in eq. (3) as follows,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

$$C_{jk} = \begin{cases} \frac{1}{2}(1-L_y), if \neg y \\ \frac{1}{2}(1-L_y), if \quad y \end{cases} \tag{3}$$

The weight matrix will represent the synaptic connection matrix between the clauses and variables in a given logical formula. A simplified approach for computing the synaptic weight matrix values of HNN which named as Wan Abdullah method has been outlined in [17-18]. This is done by equating the cost function of HNN $(E_{P_{R-kSAT}})$ to the final energy dynamics of the network $(H_{P_{R-kSAT}})$. Symbolizing the neural state of HNN in which each variable vector L_j at time t is $L_j(t)$. The local field of HNN can be represented in eq. (4) as follows;

$$h_j(t) = \sum_{j=1,k\neq j}^{N} U_{jk}^{(2)} L_k + U_j^{(1)} \tag{4}$$

$$L_j(t+1) = \begin{cases} 1, & \sum_{j=1,i\neq k}^{N} U_{jk}^{(2)} L_k + U_j^{(1)} \geq 0 \\ -1, & \sum_{j=1,i\neq k}^{N} U_{jk}^{(2)} L_k + U_j^{(1)} < 0 \end{cases} \tag{5}$$

Based on eq. (5) and (6) ensures the energy dynamics for the network decrease monotonically. The final energy dynamics of HNN is provided in eq. (6) as follows;

$$H_{R-2SAT} = -\frac{1}{2} \sum_{j=1,j\neq k}^{N} \sum_{k=1,i\neq k}^{N} U_{jk}^{(2)} L_j L_k - \sum_{j=1}^{N} U_j^{(1)} L_k \tag{6}$$

The convergence toward minimum energy is will be considered as optimal to an optimization search problem. The energy dynamics of an HNN has many local minima. As a consequence, the network is likely to reach an equilibrium state which does not satisfy a problem solution. A major task in this "field" is to search for evolutionary strategies like EA to move the network out of local minima.

4. Logic Program

The logic program has been used widely to represent connections and has both a declarative and functional meaning. It made up of a set program clause that is triggered by an initial goal declarative statement. It offers a simple way of solving problems [14]. The architecture of the logic program is easier to develop, modify and understand relative to the neuronal structure of the black-box network. Logic program is particularly appealing to novice database programmers and developers who don't want to be concerned with the complexities of monitoring the program's actions. A logic program (normal logic) is described in eq. (7) as a finite collection of logical clauses in the following expression:

$$\forall(\mathbb{C} \leftarrow B_1 \wedge ... \wedge B_n) \tag{7}$$

where $n \in N$ for each clause, they may differ, $\mathbb{C}$ described as an atom in some first-order language and $B_1, ..., B_n$ represent literals in the clause, that is, atoms or its negation. As is a common practice in logic programming, a logical clause may be as follows,

$$\mathbb{C} \leftarrow B_1 \wedge ... \wedge B_n \tag{8}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

Where the universal quantifier is known, and then $\mathbb{C}$ described the head of the program clause, B_i is referred to as body literal of the clause and their conjunction $B_1 \wedge ... \wedge B_n$ is referred to the body of the program clause.

5. Satisfiability logic program (SAT)

An instance of it is generated from three parameters (p, q, r), where *p*, *q* and *r* represent the number of a proposition, clauses, and literals per clause respectively. Each instance comprises of *q* random clauses involving exactly *r* literals each. Each is independently and randomly selected from the set of $2^r \begin{pmatrix} p \\ r \end{pmatrix}$ all possible cause of length *r*. The focus of the SAT problem, therefore, is to find out whether there exists a mapping of truth assignment to variables making the formula in eq. (9) achievable.

$$\mathbb{Z} = \bigwedge_{i=1}^{k} \mathbb{C}_i \tag{9}$$

The three components of general random-*k*SAT can be outlined as follows:

i. Consisting of a collection of I variables, $x_1, x_2, x_3, ..., x_I$ where $x_i \in \{1, -1\}$. Variables in the local clause are linked by logical symbol OR ($\vee$)

ii. A collection of literals. literal is a defined as variable x_i or its negation $\neg x_i$.

iii. A collection of n distinct logical clauses;. Each logical clause consists of literal linked by a logical notation AND ($\wedge$). Where $\mathbb{Z}$ describes the Boolean formula for *k*SAT. $\mathbb{C}_i$ designated a clausal form of DNF with *k* number of Boolean variables, *k*-CNF is a special case of CNF which contains at most *k* literals in each clause. It has also been shown that a given Boolean formula given can be translated into a 2-CNF formula for which the NP-complete satisfaction problem remains [29]. In case of *k*=2 for satisfiability problem where the logical clause in 2SAT has the represented in eq. (10) as follows,

$$\mathbb{C}_i = \bigvee_{j=1}^{k} (x_{ij}, h_{ij}), k = 2 \tag{10}$$

Where $\mathbb{Z}$ defined a satisfiability logical rule which contains of logical clause $\mathbb{C}_i$ given in eq. (11) as follows:

$$\mathbb{C}_i = \bigvee_{j=i}^{k} l_{ij} \tag{11}$$

where each of them l_{ij} represents a propositional variable and $\neg l_{ij}$ its negation; The first and most obvious application of SAT has been the well-known NP-complete which include vertex cover, TSP, independent set, Hamiltonian paths, etc.

6. Random 2-CNF formulas

The random 2-CNF Model was studied widely for a number of reasons. Firstly, it is an essentially reasonable framework, comparable to the random graph prototype, that focuses on the fundamental structural properties of the of satisfiability. Second, randomly selected formulations are experimentally difficult for SAT to choose appropriate parameters and are a frequently utilized benchmark for testing SAT algorithms. An instance of random-*k*SAT comes in the form of a collection of random clauses $M=\alpha N$ over *N* Boolean variables. Every logical clause usually contains exactly *k* variables that are associated by logical notation OR operations and tend to be negated with probability ½. In the scale-free model, given *n*, *m* and β, to formulate a random-*k*SAT, we

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

generate m clauses independently at random from the set of $2^k\binom{n}{k}$ logical clauses, sampling every valid clause with a probability of 0.5 [50-51].

Let $(\mathbb{C}_1,\mathbb{C}_1^{'}),(\mathbb{C}_2,\mathbb{C}_2^{'}),...,(\mathbb{C}_m,\mathbb{C}_m^{'})$ be a pattern of ordered pairs of 2CNF clauses selected randomly (with replacement) from a collection of all $4\binom{n}{2}$ clauses on n variables. Describe a probability distribution over 2CNF formulas by selecting exactly one logical clause from each pair [52]. For SAT or UNSAT of random 2SAT.Let $F_2(n,\ m)$ be the distribution of n variables, m clauses over 2-CNF and each clause are selected randomly from all possible 2-clauses. Let r be a positive constant. Then:

$$\lim_{x\to\infty}\Pr[F_2(n,r.n)\ \ is\ \ \text{SAT}]=\begin{cases}1, & if \quad r<1\\ 0, & if \quad r>1\end{cases} \tag{12}$$

The random 2SATformula can be summarized in eq. (13) as follows:

$$\mathbb{Q}_{R-2SAT}=(\neg F_1\vee F_2)\wedge(\mathrm{E}_1\vee\neg\mathrm{E}_2)\wedge\neg D \tag{13}$$

Where $\mathbb{Q}_{R-2SAT}$ described the random 2SAT logical rules that consist of literals $F_1,F_2,\ E_1,\ E_2\ \&\ D$ and logical clauses $(\neg F_1\vee F_2),(E_1\vee\neg E_2)\ \&\ \neg D$ generated at random. In general, the formulation can be generated in different combinations of neurons (atoms) as the number of neurons (atoms) fluctuated. Comparatively, a high number of neurons (atoms) per number of clause would increase the probability of a number of neurons beings satisfied [18-19, 28].

The costs function of the negated random *2SAT* for bipolar follows:

$$E_{\mathbb{Q}_{R-2SAT}}=\frac{1}{2}\left(1+S_{F_1}\right)\frac{1}{2}\left(1-S_{F_2}\right)+\frac{1}{2}\left(1-S_{E_1}\right)\frac{1}{2}\left(1+S_{E_2}\right)+\frac{1}{2}\left(1+S_D\right) \tag{14}$$

Random 2-SAT can be regarded as one of the constrained optimizations in the Hopfield model. This can be implemented in the network by storing atom truth values and generating a optimize cost function when maximum clauses are fulfilled. Eq. (14) can be written as in eq. (15) as follows,

$$E_{\mathbb{Q}_{R-2SAT}}=\frac{1}{4}(S_{F_1}-S_{F_1}S_{F_2}+S_{E_2}-S_{F_2}-S_{E_1}-S_{E_1}S_{E_2})+\frac{1}{2}S_D+1 \tag{15}$$

Since consistent interpretation found leading to $E_{\mathbb{Q}_{R-2SAT}}=0$ as the minimum value correlating to the fact that all random 2SAT clause is satisfied. The value $E_{\mathbb{Q}_{R-2SAT}}$ described as proportional to the number of unsatisfied clause. Applying the cost function eq. (16) to eq. (2), the respective synaptic weight of *HNN-R2SAT* can be calculated based on eq. (16).

$$\begin{aligned}H_{\mathbb{Q}_{R-2SAT}}=&-\frac{1}{2}(2L^{(2)}_{[F_1F_2]}S_{F_1}S_{F_2}+2L^{(2)}_{[F_1E_1]}S_KS_{E_1}+2L^{(2)}_{[F_1E_2]}S_{F_1}S_{E_2}+2L^{(2)}_{[F_1D]}S_{F_1}S_D\\&+2L^{(2)}_{[F_2E_1]}S_{F_2}S_{E_1}+2L^{(2)}_{[F_2E_2]}S_{F_2}S_{E_2}+2L^{(2)}_{[F_2D]}S_{F_2}S_D+2L^{(2)}_{[E_1E_2]}S_{E_1}S_{E_2}\\&+2L^{(2)}_{[E_1D]}S_{E_1}S_D)-(L^{(1)}_{F_1}S_{F_1}+L^{(1)}_{F_2}S_{F_2}+L^{(1)}_{E_1}S_{E_1}+L^{(1)}_{E_2}S_{E_2}+L^{(1)}_DS_D)\end{aligned} \tag{16}$$

The satisfied interpretation such as $S_K=S_Q=S_F=S_H=1$ is implemented into eq. (16). The global minimum energy is obtained as $H^{\min}_{\mathbb{Q}_{R-2SAT}}=-1$.The accuracy of the neuron state created by the network during the recovery stage will be used to be separated.

7. Exhaustive Search Algorithm

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

The aim of selecting the traditional searching framework is to determine the degree of efficacy of HNN-R2SATES model in carrying out random 2SAT logic programming. Apart from that, there are feasible satisfiable assignments given for any random 2SA logical representation [53]. The satisfied clause for the ES heuristic is extracted after a brutal " trial and error" procedure is carried out. The efficiency of the ES as searching algorithm received attention in [53-54]. The exhaustive search does the survey required to produce a precise topographical map. This approach requires that an extremely large but limited solution space be checked with the number of combinations of various variable values

The objectives function of ES is expressed as follows:

$$max\left| f_{ES} \right| \tag{17}$$

Where,

$$f = \prod_{j=1}^{NC} C_j \tag{18}$$

Stage 1: Initialization

The random population of an individual in the form of a variable vector S_j where, $S_j(t) \in [-1,1]$ is initiated.

Stage 2: present input vector

Present the input pattern $S_j = (S_{j1}, S_{j2}, ..., S_{jN})$ to the network and store it, where $j = (1,2,...,m)$.

Stage 3: Fitness Evaluation

The variable vector is tested based on the fitness $S_j(t)$ to a quantified variable position in the solution space,

$$f_j(p_j(t)) = \sum_{j=1}^{NC} C_j \tag{19}$$

Where f_j designates the number of different neurons combination, *NC* designates the number of a logical clauses in the random 2SAT and C_j the number of different values of logical clauses passed through the eligibility stage as follows;

$$C_j = \begin{cases} 1, & eligible \\ 0, & otherwise \end{cases} \tag{20}$$

Stage 4: clause evaluation

Preserve the clause with the highest possible fitness. Otherwise, identify a new candidate variable vector. Exhaustive searches do not get trapped in local minima with fine enough sampling and worth for both continuous and discontinuous variables function [29, 53]. Nonetheless, achieving a global minimum requires an extremely long time. Another drawback of this strategy is that the global minimum may be skipped because of undersampling. When measuring the cost function it is simple to use the sample whenever it takes a long time. For this reason, exhaustive searches are only practical for a small number of variables. For this reason, exhaustive searches in a minimal search area are only practicable for a small number of variables [12, 18, 28, 54].

8. Election Algorithm (EA)

Election algorithm is a simple computational system inspired by the socio-political phenomenon of a presidential election. The EA begins by describing the optimization variables, the cost function, and the cost. It ends by checking for convergence like other optimization algorithms. It is considered a robust evolutionary methodology attracting a talented amount of research in optimization research. If an environment is considered as a function, EAs serve as a function optimizer. In this case, each individual in a population is a sample point in the function space [25-26].

In this simulation, a weight matrix (population) is generated at random at the beginning of the EA. In every phase, the vector will be updated through the campaign and coalition mechanism, and their eligibility values will be evaluated to ascertain the best candidates or voters' position. The cycle of reorganizing the new weight vector variable with repeat the best individuals vector and continue the searching until an appropriate solution is reached (best solution) [25-26]. The HNN energy dynamics is used as the second eligibility assessment method to pick the most appropriate weight matrix to solve the problem of random-*k*SAT. On the contrary, the basic motivation of the election algorithm is to discover the variables that optimize the number of random clauses optimize before joining the HNN. The election algorithm in random-*k*SAT is specifically made up of distinctive stages 1-5.

In general, the global optimization can be demonstrated as below (without the loss of generality minimization problem being considered as;

$$max\left|f_{MEA}\right| \tag{21}$$

$$f_{MEA}=\prod_{j=1}^{NC}C_j \tag{22}$$

Where f_{MEA} is the eligibility function (Objectives), *NC* describes as the number of a clause in random -*k*SAT and C_i designates the number of clauses tested by the EA given as follows,

$$C_j=\begin{cases}1, & eligible\\ 0, & otherwise\end{cases} \tag{23}$$

Stage 1: Initialization

A random population of individuals are initialized based on certain features and the size of a population in form decision variables, p_j, where $p_j(t)\in[1,-1]$.Each interpretation is a potential solution to the random-*k*SAT clause randomly generated. The EA procedure begins by splitting the entire population into P parties. Initially, the number of individuals will be denoted by N_p in each party, in which, $N_1,...,=N_p,...,=N_P$, where, $p=1,2,...P$.

Stage 2: Eligibility evaluation

All randomized variable vector will undergo eligibility function assessment f_{MEA}. Each of the correct variable vectors yields in the satisfied random-*k*SAT clause will be "awarded". The number of achieved clauses constitutes the eligibility of the individuals (variable) during the eligibility assessment. The objective function $S_j(t)$ of the election algorithm is as follows:

$$f_j(p_j(t))=\sum_{j=1}^{NC}C_j \tag{24}$$

The role of the eligibility function is to measure the goodness of each individual variable.

Stage 3: Create an initial parties and their supporters

Split the sampling space connected with each variable vector x_j, $j=1,...,N$ into *P*party's space i.e

$$\forall x_j N_c^p = [N_c^{l,p},\ N_c^{u,p}],\ p=1,...,P,\ j=1,...N \tag{25}$$

From each party $p=1,...,P$ subspace, $N_c^p=[N_c^p, N_c^p]$ associated with each variable vector x_j, $j=1,...,N$, N_P values are selected and assessed based on the associated objective functions. Then, the individuals connected with each party *p*. $p=1,...,P$ could be designated as follows: $\forall pf(v_{1,m_p},...,v_{j,m_p},...,v_{N,m_p}), m_p=1,...,M_P$

Or

$$p=\begin{bmatrix} v_{1,1},...,v_{j,1},...,v_{N,1} \\ \vdots \\ v_{1,m_p},...,v_{j,m_p},...,v_{N,m_p} \\ \vdots \\ v_{1,M_P},...,v_{j,M_P},...,v_{N,M_P} \end{bmatrix}, p=1,...,P \tag{26}$$

Select the best value to serve as the initial number of candidate N_C in forming the initial parties [32]. Thus, the number of individuals sampled as initial candidates represented in eq. (27) as follows,

$$N_C=[C_r \times N_{Np}] \tag{27}$$

Where N_C described the total number of candidates, C_r describe the candidate rate, N_{Np} defined as the total population. The remaining is the total number of such candidates ' advocates (voters) is represented in eq. (28) as follows,

$$N_v = N_{Np} - N_C \tag{28}$$

Where N_v described the total number of voters, N_{Np} is the total population and N_C designated as the total number of candidates. All supporters are divided among the candidates based on their similarity. The Euclidian distance matrix is used as a similarity measure is used as follows.

$$d(v_i,c_j)=\sqrt{\sum_{j=1}^{N_c}(E_{v_i}-E_{c_j})^2} \tag{29}$$

Where E_{v_i} indicates the voter's eligibility and E_{C_i} indicates the *j*th candidate's eligibility. The party whose candidate is closest to it is assigned to each voter. In other words, if the following predicate holds: a person v_p; is considered as a candidate advocate c_i

$$P_i=\{v_p : \|E_{v_p}-E_{c_i}\| \le \|E_{v_p}-E_{c_j}\| \forall 1 \le j \le N_c\} \tag{30}$$

Based on eq. (30), each voter v_p is assigned precisely to one party c_i, where P_i defines the *i*th party, E_{c_i} and E_{v_p} indicate the eligibility of a voter v_p and candidate P_i.

Stage 4 Advertising Campaign

Positive advertisement

The process of modelling EA positive mechanism, the vector variable of a candidate in the solution space is randomly selected. Random numbers are sampled to pick the position of vector variables (voters) to be substituted. The selection rate is donated by $X_s = rand \in [0,1]$. The number of variable vector values that are transferred from a candidate toward its supporters is represented in eq. (31) as follows [25].

$$N_S = [X_S \times S_C] \tag{31}$$

Where N_S is defined as the number of sampled vector variables to be replaced X_S define the selection rate and S_C described the total number of vector variables of the candidate in the solution space. It is clear that, in an EA party based on the eligibility Euclidean metric between a candidate and its relevant supporters, the effectiveness of advertisement varies. Positive advertisement happens, whereby the candidates that seem to have an excellent plan and idea if the decision was taken by the voters is to be influenced, the number of its supporters will increase and the chances of increasing the quality of the party's plans will increase.

Tomodel this goal,we represented eligibility distance coefficient (ψ_e) as follows,

$$\psi_e = \frac{1}{\|M_e - M_v\| + 1} \tag{32}$$

Where M_e and M_v designated the eligibility of candidate *e* and voter *s*, respectively. EA applied Euclidean metric to measures the distance between the vector variable M_e and M_v in the solution space [32]. In EA, the advertising mechanism, after selecting N_S and measured ψ_e the vector values of identified vector variables from the candidate are multiplied in coefficient ψ_e and the replaced with the identified vector of the associated voters. In other words, given e_{iold} be the value of *i*th chosen vector variables of the voters before advertising advances, then after a campaign, the updated value of the identified vector is given as follows:

$$e_{inew} = \psi_e \times e_{iold} \tag{33}$$

On the basis of the eq. (33) in the campaign process, the near supporters are much more influenced by their associated candidate than by other followers.

Negative advertisement

In the implementations of EA, contrast advertisement is used among different negative campaigning strategy. Candidates, by their campaign of resistance, seek to fascinate the members of other parties towards themselves. This leads to an upsurge in support of the popular parties and also to a decline in popularity of the marginalized parties. The difference in eligibility between the voters and the candidates is measured at first in the defender group by applying the Euclidean metric as follows,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

$$dist(r_i, t_j) = \left\| r_i - t_j \right\| = \sqrt{\sum_{j=1}^{M_d} (E_{r_i} - E_{t_j})^2} \quad (34)$$

Where M_d described the number of all voters in the defender party, t_i defined the candidates of defender party and r_i defined the *i*th supporter of the defender party. E_{r_i} & E_{t_j} are the eligibility of candidate t_i and *i*th supporter, respectively.

Coalition

Candidates confederate if they shared the same ideas; In EA, sometimes two or more parties with the same ideas and goals in solution space can come together to create a new party. So some candidates are leaving the campaign and joining another one called "leader." The candidate leaving the arena of the election is called "follower" The candidate leaving the election arena is referred to as "follower." The candidates of the followers collate with the leader and encourage their supporters to follow the leader. All the followers ' supporters become the leader's supporters. In our applications, among the candidates who wish to unite, a candidate is randomly selected as the successor candidate and the remaining candidates are considered as followers [25-26].

Stage 5: Election Day (Stopping condition)

Until a condition of termination is met, three different operators, positive advertising, negative advertising and coalition will be applied to update the population. Ultimately a candidate who gets the most votes will declare himself the winner and is equal to the best solution found for the problem of optimization and search [25]. Modified election algorithm incorporated in the Hopfield network model to carry out random-*k*SAT is expected to be feasible. It the vector variable of does not achieve the desired eligibility; the present vector variable bits will continue to improve during the negative campaign and coalition strategies. In order to boost the solution space, 100 to 10000 iterations usually set in most of the metaheuristics is also considered in our case [18]. A bipolar search involving on 1 and -1 is used since it is simple for a variable to converge to global maxima. In this research, a modified election algorithm is proposed to accelerate the learning phase of the Hopfield network model as a single model to carry out random 2SAT logic program.

Election algorithm

```
BEGIN
Election Algorithm
BEGIN
  1. Generate initial population;
  2. Compute eligibility of each individual;
  3. Create initial parties: initial candidates and their advocators
  REPEAT/ /* advertising campaign (ad days) //*
      For candidate size do
      // positive advertisement mechanism
      Candidates advertise their plans and improve
      their position by learning new ideas
      // negative advertisement mechanism
      Candidates battle with one another to increase their advocators;
      // coalition mechanism
  4. Candidates unit if they have same ideas;
  5. Compute eligibility of candidates;
  UNTIL population has converged//*(Election Day)*//
  END
```

Figure 1: Pseudocode of the Election algorithm

9. Imperialist Competitive Algorithm (ICA)

Imperialist competition algorithm (ICA) is a metaheuristics optimization build on the basis of a socio-politically motivated strategy. It belongs to classes of evolutionary algorithm extracted from the imperialist competition. It begins with an initial population known as countries; a country includes groups of colonies and imperialist that together from empires [55-57]. In reality, imperialist countries are always trying to conquer other countries to turn them into their own colonies and complete among themselves actively for colonization of other nation; a proposed evolutionary heuristic is the competition that arises between empires in the solution space. During this competition, the weakened empires will collapse, and hence the stronger ones will become more powerful [50]. ICA converges towards a scenario where there exists only one empire in the solution space which is equal to the optimal solution. In this situation, the colonists use the same strength (fitness) and control function as the imperialists, as a portion of the colonies can be absorbed by imperialists [55-57].

The ICA consists of two primary phases, the movement of the colonies and imperialist competi tion. The ICA begins by generating a random population of countries in the solution space in the form of a variable vector S_i where $S_i(t) \in [-1,1]$ is initiated. Select best countries in the solution space to serve as imperialists and the remaining becomes the colonies of the imperialist. The country is described as $1 \times N$ variable vector in the N-dimensional matrix. This matrix vector is represented based on eq. (35).

$$country = [l_1, l_2, l_3, ..., l_{Np}] \tag{35}$$

The cost of each country is ascertained by calculating its objectives function $f(l_1, l_2, ..., l_{Np})$. Then,

$$\cos t = f(country) = f(l_1, l_2, l_3, ..., l_{Np}) \tag{36}$$

In order to share the colonies proportionally among the imperialists, the normalized cost of an imperialist is simply defined as follows,

$$M_n = m_n - \max_j \{m_i\} \tag{37}$$

Where M_n defines as the cost of nth imperialist and m_n is their normalized cost. Any imperialist with a higher cost value will have a lower normalized cost value. The power of each imperialist, having the normalized cost is measured using eq. (38) and it is based on the distribution of the colonies countries among the imperialist countries,

$$C_n = \left| \frac{m_n}{\sum_{i=1}^{N_{imp}} m_i} \right| \tag{38}$$

On the other hand, the colonies establish the normalized power of an empire. Then there will be the initial number of colonies of the empire which is defined as in eq. (39);

$$NM_n = rand\{C_n.(N_{col})\} \tag{39}$$

where NM_n represent the initial number of *nth* Imperial colonies and N_{col} represent the number of colonies in the population. NM_n are randomly selected from the colonies and assigned to their imperialists to disperse them among the imperialists. Using the absorption scheme, the imperialist countries incorporate the colonies within themselves. The absorption strategy makes the main core of the ICA, it empowers countries to move towards their optimal solution. Such colonies are absorbed by the imperialists in the light of their power as defined in eq. (39). The total strength

of each imperialist country is measured by the fitness power of both parts; the power of the empire plus the power of the average colonists as follows;

$$TM_n = \cos_t(f_{imperialist_n}) + \xi mean\{cos_t(f_{colonies\ of\ empire_n}\} \quad (40)$$

Where TM_n define the overall cost of the nth empire and ξ donate a positive number that is defined to be below one [49] as in eq. (41),

$$y \sim U(0, \delta \times d) \quad (41)$$

Throughout the absorption strategy, the colony countries forwarded towards the imperialist by a unit (y). The dimension of movements, the vector between the colony and the imperialist. The difference that d and y have shown between the imperialist and their colony is uniformly distributed random variable. Where δ is less than 2 but greater than 1. So, a reasonable choice can be δ = 2. In our execution γ is $\frac{1}{4}\prod$ (rad) respectively.

$$\theta \sim U(-\gamma, \gamma) \quad (42)$$

A colony could be better positioned as it moves towards the imperialist countries so that the colony's position changes as per the imperialist's position. In ICA competition plays by Imperialist impacts strongly on convergent of the algorithm. Throughout colonial rivalries, the weakened empires will give up their power and colonies. To model a competition process in ICA; compute the probability that each empire would own all the colonies, taking into consideration the empire's total cost.

$$NTM_n = \max_i\{TM_i\} - TM_n \quad (43)$$

Where TM_n refer to the total cost of nth empire and NTM_n descried the normalized total cost of nth empires. The ownership probability of each empire is measure in eq. (44) with the normalized total cost of empire.

$$Cc_n = \left| \frac{NTM_n}{\sum_{i=1}^{N_{imp}} NTM_i} \right| \quad (44)$$

To split the mention colonies respectively empires on the basis of their ownership probability P, matrix is constructed as in eq. (45) as follows,

$$P = [P_{p_1}, P_{p_2}, ...P_{p_{N_{imp}}}] \quad (45)$$

Form a vector matrix of the same size with a P with uniformly distributed random number of elements,

$$J = [j_1, j_2, ..., j_{N_{imp}}] \quad (46)$$

$$j_1, j_2, ..., j_{N_{imp}} \sim U(0,1) \quad (47)$$

Then vector D is form by simplify subtracting vector H from vector P as in eq. (48) follows,

$$Z = P - J = [z_1, z_2, ..., z_{N_{imp}}] \quad (48)$$

Eq. (49) can written as in eq. (50) as follows,

$$= [P_{p_1} - j_1, P_{p_2} - j_2, ..., P_{p_{N_{imp}}} - j_{N_{imp}}] \quad (49)$$

Applying to matrix Z, in ICA handover the colonies listed to an empire with a maximum corresponding index in Z. After a while, all the empires will crumble with the exception of the most powerful country and all the colonies remain under one single empire (best solution) [55-58].

In this paper, the hybridisation of ICA and HNN for random-SAT is represented as HNN-RSATICA. The main stages in HNN-RkSATICA are as follows:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

Stage 1. The random-*k*SAT problem is computed into a Hopfield Network, which is continuous and non-constrained.

Stage 2. The countries in ICA are initialized at random.

Stage 3. The HNN will obtain a feasible for every single run.

Stage 4. The fitness of the countries is calculated.

$$f_{country} = C_1(x) + C_2(x) + C_3(x) +,...,C_{NC}(x) = \sum_{i=1}^{NC} C_i \quad (50)$$

Where $f_{country}$ is the fitness of the countries,

Stage 5. New individuals are created, where the input vectors (neurons) are updated, and the output vectors (neurons) are calculated.

Stage 6. The new colonies will move towards the imperialist and the cycle continues.

IMPERIALIST COMPETATIVE ALGORITHM

1. Initialize the empires

$$country = [l_1, l_2, l_3, ..., l_{Np}]$$

$$\cos t = f(country) = (l_1, l_2, l_3, ..., l_{Np})$$

$$N_{col} = N_{pop} - N_{imp}$$

$$M_n = m_n - \max_j \{m_i\}$$

$$C_n = \left| \frac{m_n}{\sum_{i=1}^{N_{imp}} m_i} \right|$$

$$NM_n = rand\{C_n.(N_{col})\}$$

2. Move all colonies toward their imperialist country (Assimilation)

$$y \sim U(0, \delta \times d)$$

$$\theta \sim U(-\gamma, \gamma)$$

3. If there is a colony with has less cost than the imperialist country in an empire, exchange the locations of the colony and the imperialist country
4. Calculate total cost of each empire.

$$TM_n = \cos t(imperilaist_n) + \xi mean\{cost(colonies\ of\ empire_n\}$$

5. Select the weaker colony from the weaker empire and pass it to the empire that has more possibility to possess it. (The imperialistic competition)

$$NTM_n = \max_i \{TM_i\} - TM_n$$

$$Cc_n = \left| \frac{NTM_n}{\sum_{i=1}^{N_{imp}} NTM_i} \right|$$

6. Delete the weakest empires.
7. If only one empire remains, stop, if not return to step 2.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

Figure 2: Pseudocode of the Election algorithm

10. Model Performance Evaluation Criteria

The performance of our proposed HNN-R2SATEA model is compared with two existing models HNN-R*k*SATES [18] and HNN-R2SATICA [58] in term of global minimum ratio (*Zm*), Means Absolute Error (*MAE*), Bayesian Information Criterion (*BIC*) and Execution time (*ET*) to ascertain its efficiency, accuracy, robustness and model selection to make a fair comparison, both experiments are implemented in same computer with the same processor.

Table 1. List of parameters used HNN-R2SATES [18]

Parameter	Value
Neuron Combination	100
Number of Trials	100
Tolerance Value	0.001
Number of String	100
Selection_Rate	0.1

Table 2. List of parameters used in HNN-R2SATICA [58]

Parameter	Value
Neuron Combination	100
Number of Trials	100
Tolerance Value	0.001
Initial Empires	10
Parameter	0.05
Termination Value	0.05

Table 3. List of parameters used in HNN-R2SATEA

Parameter	Value
Neuron Combination	100
Number of Trials	100
Tolerance Value	0.001
Number of Learning	100
Positiovecampaign_Rate	0.2
Coalition_Rate	0.3
Candidate_rate	0.07

10.1 Global Minima Ratio (Zm)

This describes as the ratio of global minimum energy combined to the number of neurons in the solution space [18]. The *Zm* formula is given in eq. (51) as follows,

$$Z_m = \frac{1}{(NTr)(COMBMAX)} \sum_{i=1}^{n} N_{E\min} \tag{51}$$

*NTr*represents the amount of neurons trials, $N_{E\min}$ described the global minimum solutions achieved and *COMBMAX* described the maximum neurons combination generated in the solution space.

10.2 Mean Absolute Error (MAE)

This described as the average difference found between the expected and the actual values in the solution space in a given data. It is used to ascertain the proximity of forecasts to potential outcomes in a given distribution. It is commonly used because it has the capacity to estimate the error in the data. It is identified as one of the effective metrics used to identify the accumulation of uniformly distributed error in a given sample [18, 59]. The MAE equation is presented as the follows,

$$MAE = \sum_{i=1}^{n} \frac{1}{n} \left| f_{\max} - f_i \right| \tag{52}$$

where f_i and $f_{\max}$ describe the fitness value observedand the maximum fitness respectively.

10.3 Bayesian Information Criterion(BIC)

This described a criterion for the selection of models amongst a finite set of models in a given distribution. It is used to assess the computational efficiency of a model. The Mean square error (MSE) is taken into consideration when computing the BIC values. It is important to articulate the relationship between these two units [55]. In general, when the MSE is lower, the BIC tends to be lower. The model which has the minimum BIC value is regarded as the model of interest [60]. Cumulatively, the MSE will be used to evaluate the BIC value during the training and recovery process. The BIC formula is described as follows:

$$\text{BIC} = nIn(MSE) + pa.In(n) \tag{53}$$

Where *n*, *pa*, and *MSE* indicate the number of solutions obtained, its parameters and the mean square error used in the model respectively.

10.4 Execution Time(ET)

Execution time or commonly known as a computational time,represent the time taken to complete an implementation cycle [18]. It is represented in eq. (54) as follows,

$$\text{E\_Time (s)} = Training\ \_\ Time\ (s) + \text{Recovery\_Time}(s) \tag{54}$$

11. METHODOLOGY/IMPLEMENTATION/EXPERIMENTAL SETUP,

Implementation of Neuro-Heuristic Method random-*k*SAT in the HNN(HNN-R2SATEA, HNN-R2SATES & HNN-R2SATICA) were executed conducted with an Intel® Celeron(R) CPU B800@ 1.80GHz 4.00Gb (2.85GB usable), via MICROSOFT VISUAL WINDOW 8 DEV C++. The program's main task is to find the best "model" that find the optimal occurrences of random-*k*SAT. Both parameters and clauses were initially randomized. Simulations performed with a different number of complex neurons from *NN*=5 until *NN*=100. The execution of these models is carried out on random 2SAT logical rule as presented according to the following steps;

i. Translate all the random-*k*SAT logical clauses into Boolean algebra.
ii. Designate neuron to each variable in a random-*k*SAT formula.
iii. Randomize the state of the neurons and initialize all connection strengths to zero.
iv. Derive a cost function of HNN with the negation of all random 2SAT clauses by assigning $L = \frac{1}{2}(1 + S_L)$ and $\bar{L} = \frac{1}{2}(1 - S_L)$. The State of the Neuron has found out the truth as $S_L = 1$ and false when $S_L = -1$
v. Equate the cost function to energy dynamics, and obtain the values of the synaptic weight vector. Check clause satisfaction by applying EA, ES and ICA method. The satisfied assignment will be stored into the Hopfield network.

vi. Apply Sathasivam's relaxation method via equation.

vii. Randomise neural states. Compute the respective local field $h_i(t)$ of the state. If it remains unchanged after five loops, it will be considered as a stable state.

viii. Find the corresponding final state of the network by using the Lyapunov energy dynamics equation. Check whether the final energy derived is a global or a local minima. Find the terms *Zm, MAE, BIC* and *ET* of the models to ascertain their efficiency, accuracy, robustness and model selection. Sathasivam [40], proved that 0.001 tolerance value for Lyapunov energy dynamics is appropriate since it yields a better sorting mechanism for the network.

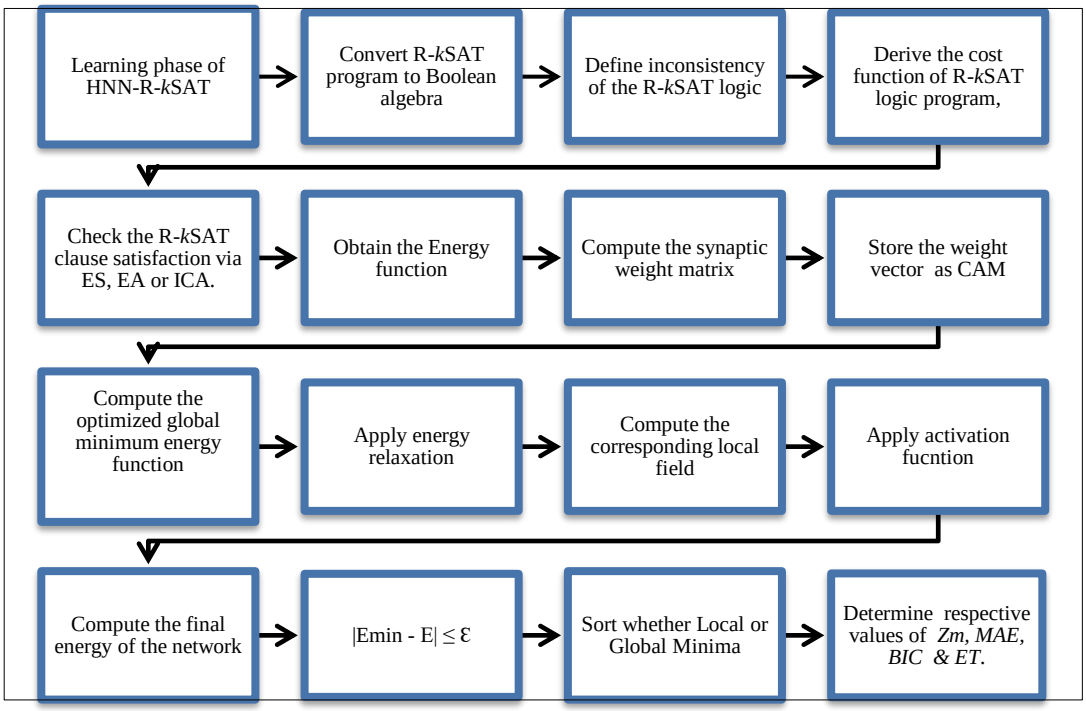

Figure 4. The implementation method of HNN-R2SATES, HNN-R2SATEA & HNN-R2SATICA

12. RESULTS AND DISCUSSIONS

The outcomes are reported after simulation of the random 2SAT logic system using various performance evaluation metrics. The *Zm, MAE, BIC* and *ET* of the models are plotted and presented in Figures 5-8. The objective is to review the performance of three HNN models in doing random 2SAT logic programming via simulated data.

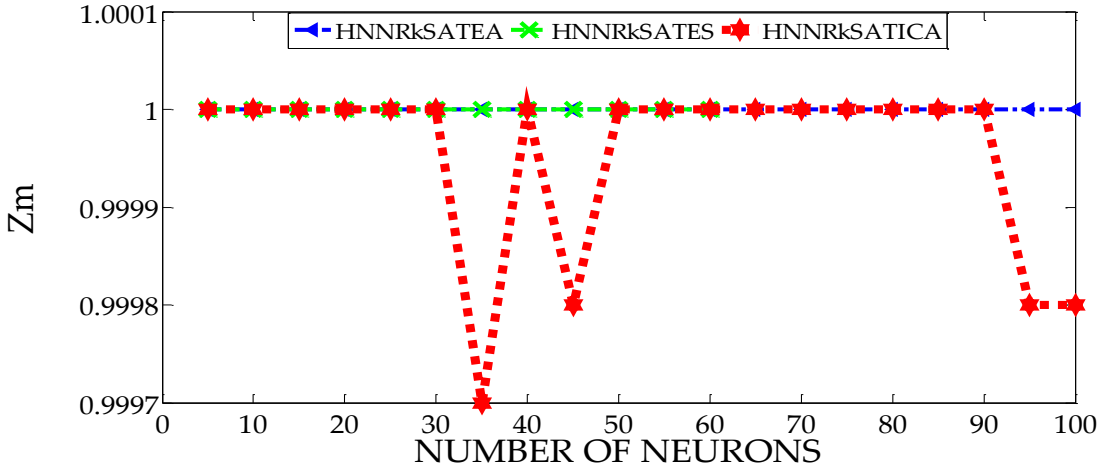

Figure 5. *Zm* for HNN-R2SATEA, HNN-R2SATES & HNN-R2SATICA

Figure 5, Shows the apparent success of the election algorithm (EA) in comparison to the exhaustive search (ES).and Imperialist competitive algorithm (ICA) in generating the optimally satisfied clauses. Meanwhile, the ES stressed the 'exhaustive' trial and error searching techniques during clause compliance. Once the complexity boosted, HNN-R2SATES were capable of sustaining only 60 neurons. This can be due to the nature of a thorough search, which raised the pressure of computing in order to achieve the correct neuron status. On the other hand, the ability of EA to control a high number of neurons form *NN*=5 up to *NN*=100 may be due to the sheer potential of EA's campaign and coalition mechanism and ICA's revolutionary and competition operator, that reduce the computation burden in finding the optimal states. However, some neurons states get trapped at NN=35, 45 and 95 in HNN-R2SATICA model as seen from the Figure 4. It indicates a greater efficiency in the technique of neuro-searching generated by HNN-R2SATEA carrying out random 2SAT logic programming. As the constraints expand forever, the network becomes more difficult as the NN rises in terms of the program's complexity.

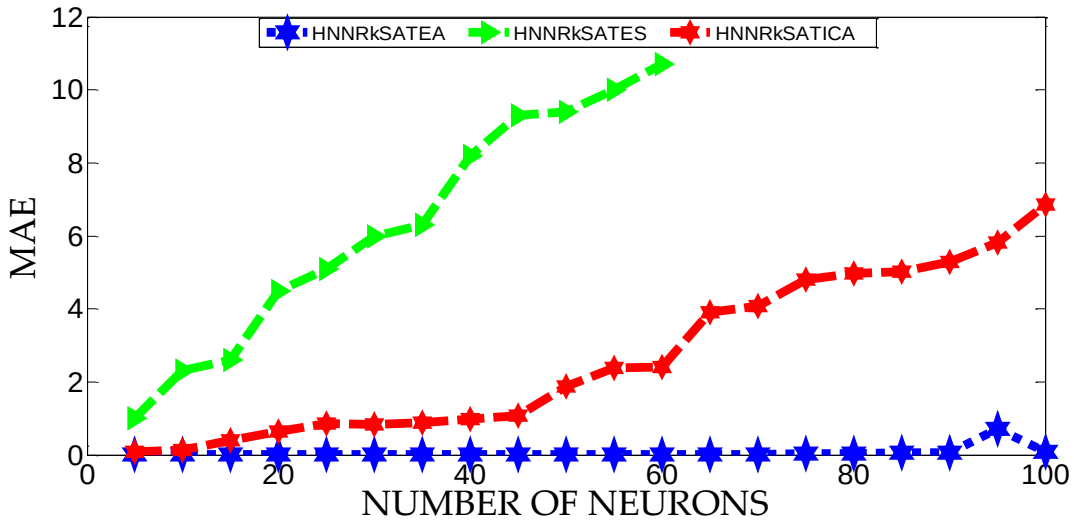

Figure 6. MEA for HNN-R2SATEA, HNN-R2SATES & HNN-R2SATICA.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 January 2020 doi:10.20944/preprints202001.0365.v1

Peer-reviewed version available at Processes 2020, 8, 568; doi:10.3390/pr8050568

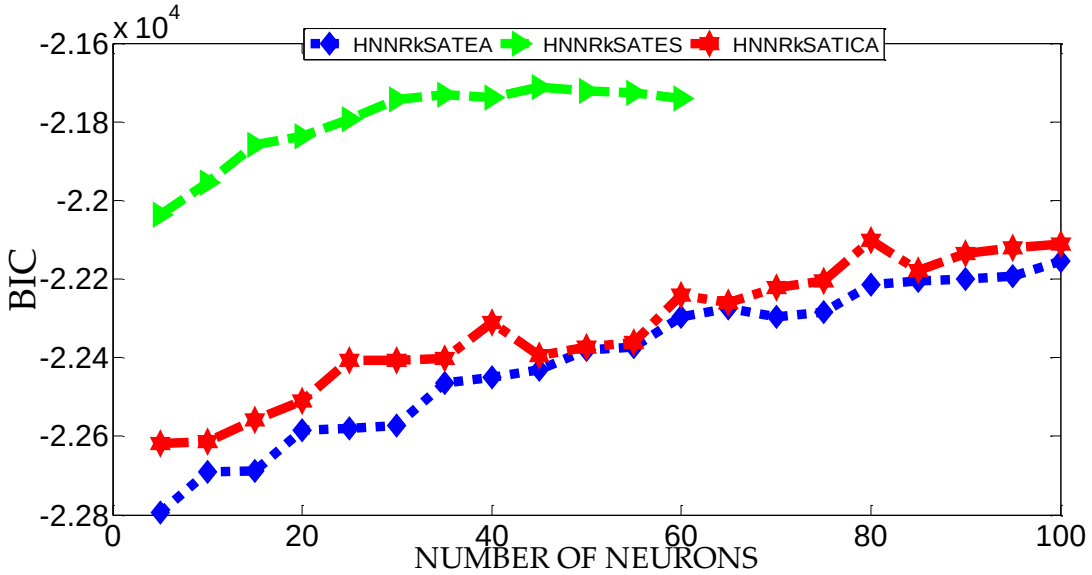

Figure 7. BIC for HNN-R2SATEA, HNN-R2SATES & HNN-R2SATICA.

Figures 6 and 7 the *MAE* and *BIC* evaluation between HNN-R2SATEA, HNN-R2SATES and HNN-R2SATICA on carrying out random 2SAT logic programming. It recorded the models' performance throughout the system learning phase from NN=5 to NN=100. It is explicitly shown that HNN-R2SATEA outshines the HNN-R2SATES and HNN-R2SATICA models on the basis of the MAE and BIC assessments. The HNN-R2SATES show a steady rise in errors due to the brute-force hunting for satifiability mapping from *NN*=5 up to *NN*=100. The ICA displays a growing pattern of angularity, but most of it lower than the ES. The colonial operator's efficiency decreases the iterations resulted in finding global solutions better than ES. However, HNN-R2SATEA outclasses the HNN-R2SATES and HNN-R2SATICA based on *MAE* and *BIC* measures. This is due to the fact that the optimization mechanism, such as a coalition in the EA searching process, is much simpler without requiring additional iterations to reach a satisfying assignment. In fact, non-improving solution will be enhanced by a coalition strategy during the HNN learning phase. The HNN-R2SATES model observed to have reported an accumulation of *MSE* during the learning stage, as a result, more iteration needed to achieve global convergence. The accumulation of MSE tends to penalize the values of BIC. The *BIC* for HNN-R2SATES is, therefore, the highest compared to the other two models. In terms of *MAE* and *BIC* assessment, EA is an acceptable approach in Hopfield network in doing random 2SAT logic programming compared to ES and ICA.

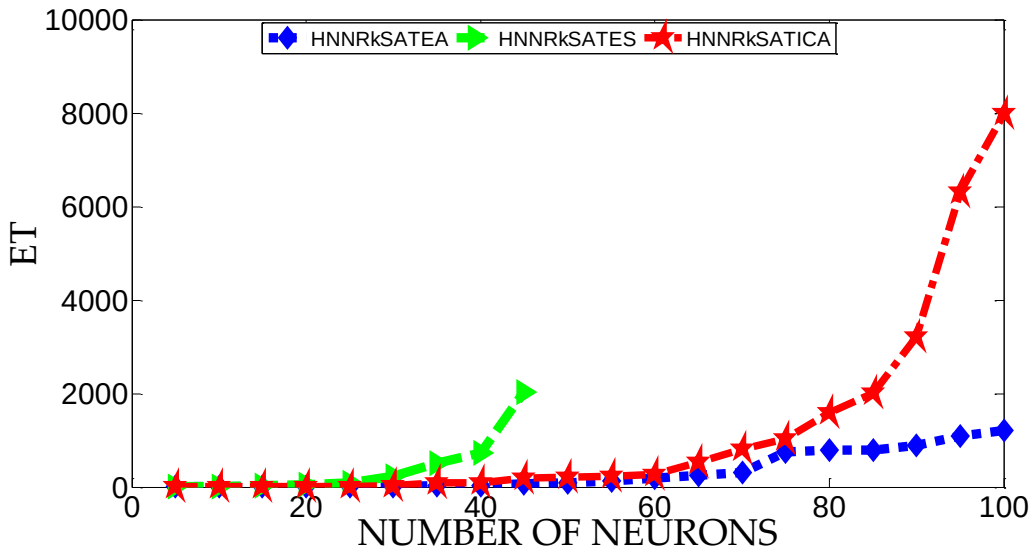

Figure 8. ET for HNN-R2SATEA, HNN-R2SATES & HNN-R2SATICA

Figure 8 demonstrates the Execution time for our proposed model, HNN-R*k*SATEA in comparison with HNN-R2SATICA and conventional HNN-R2SATES models. A quick glance at the running time shows that the program is becoming more complex, that it takes more effort to find global solutions. According to Figure 8, our proposed model HNN-R2SATEA requires less execution time compared to HNN-R2SATES and HNN-R2SATICA. Due to the fact that more neurons are required to cross the energy barrier to relax in global solutions through out the training phases of HNN [24-25]. In addition, the training process employing ES consumes more execution time due to the obvious trial and error procedure in locating the optimum number of randomly satisfying assignments. On the contrary, the implementation time was quicker with the incorporation of EA into HNN due to advocacy and coalition structures that speed up the training cycle to converge to global optimality. The coalition process can avoid the individual to trap in local minima (unsatisfied clause). Hence, the individual created by EA achieved global minima swiftly compared to ICA and ES searchin methods.

13. Conclusion

From the results obtained from simulations, it can be ascertained that our proposed HNN-R2SATEA model is the more improved and robust heuristic compared to HNN-R2SATICA and HNN-2SATES in accelerating the learning phase of HNN in carrying out a random-2SAT logic program. Our proposed model outperformed HNN-R2SATICA and HNN-R2SATES model. This has been established from the reported results in term of *Zm, MAE, BIC* and *ET* and more interestingly, a *Zm* of 1 is generated throughout the runs even with the complexity of the network. This also leads to the conclusion that EA is much more robust to boost the training phase of HNN for random 2SAT logic program and it is closest towards the global minimum, irrespective of the *NN* release into the HNN. Finally, in accelerating the computational phase of Hopfield neural network model, other metaheuristics approaches can be implemented in future.

Acknowledgements

This research is supported by the Fundamental Research Grant Scheme by Ministry of Higher Education Malaysia (203/PMATHS/6711689) and Universiti Sains Malaysia.

References

1. YahayaPudza, M.; ZainalAbidin, Z.; Abdul Rashid, S.; MdYasin, F.; Noor, A.S.M.; Issa, M.A. Sustainable Synthesis Processes for Carbon Dots through Response Surface Methodology and Artificial Neural Network. *Processes*, **2019**, *7*, 704
2. Shang, Y.; Wah, B.W. Global Optimization for Neural Networks Training. *IEEE Computer*, ***1996***, *29, 45-54.*
3. Abraham, A.; Nath, B. *Artificial neural networks for intelligent real-time power quality monitoring systems,* in M. Israel (Ed.), Proceedings of First International Power & Energy Conference, Australia, December, **2000,** ISBN 0732 620 945
4. Some, K.; Manu, P.S. Pattern recall analysis of the Hopfield neural network with genetic algorithm. *Computers and Mathematics with applications, **2010**, 60, 1049-1057.*
5. Leonard, J.; M. A. Improvement of the Backpropagation algorithm for training neural networks. *Computers& Chemical Engineering, **1990**, 14, 3, 337-341*
6. Mirjalili, S.; Safa-Sadiq, A. "*Magnetic optimization algorithm for training multilayer perceptron*", in IEEE International Conference on Industrial and Intelligent Information, Indonesia, **2011**, 42-46.
7. Mohammad, A. A.; Mohammad, E.; Amin, S.; Seyed, M.; Javad, M. Evolving artificial neural network and imperialist competitive algorithm for prediction oil flow rate of thereservoir. *Applied Soft Computin, **2013**, 13, 1085-1095.*

8. Itay, H.; Matthieu, C.; Daniel S.; Ran, E.; Yoshua, B.Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *Journal of Machine Learning Research,* ***2018,*** *18, 1-30.*
9. Haibin, D.; Linzhi, H. Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning, *Neurocomputing*, **2014**, 125, 11, 166-171.
10. Hosseini, S.M.; Khaed, A. A survey on the Imperialist Competitive Algorithm metaheuristic, Implementation in engineering domain and directions for future research. *Applied Soft Computing*, **2014**, 24, 1078-1094.
11. Leslie, N. S.*Cyclical Learning Rates for Training Neural Networks*.IEEE Winter Conference on Applications of Computer Vision (WACV), **2017**, 58
12. Mansor, MA.; Kasihmuddin, M.S.; Sathasivam, S. VLSI circuit configuration using satisfiability logic in the Hopfield network. *International Journal of Intelligent Systems and Applications (IJISA), 2****016****, 1, 8, 22-9.*
13. Kzar, A.; Jafri, M.; Mutter, K.; Syahreza, S. A Modified Hopfield Neural Network Algorithm (MHNNA) Using ALOS Image for Water Quality Mapping. *International journal of environmental research and public health*, **2016,** *13*, 1-92.
14. Kowalski, R.A. *The logic for Problem Solving.* New York: Elsevier Science Publishing, **1979**.
15. Pinkas G. Symmetric neural networks and propositional logic SAT. *Neural Computation, **1991**; 3, 2, 282-91.*
16. Nikola, K. Evolving Connectionist Systems for Adaptive Learning and Pattern Recognition: From Neuro-Fuzzy- to Spiking- and Neuro genetic. *Handbook on Computational Intelligence,* **2016**, 385-400.
17. Wan Abdullah, A.T.W. Logic programming on a neural network. *International Journal of Intelligent Systems,* **1992,** 7, 6, 513-519.
18. Sathasivam, S. Learning in the recurrent Hopfield network. *Computer Graphics, Imaging and Visualisation, CGIV'08. Fifth International Conference on IEEE,* **2008**, 323-328.
19. Sathasivam, S. Upgrading logic programming in Hopfield nets. *Sains Malaysiana, **2010***, 39 ,1, 115-118.
20. Sathasivam, S. Boltzmann machine and new activation function comparison. *Applied Mathematical Sciences,* **2011**, 5, 78, 3853-3860.
21. Hamadneh, N.; Sathasivam, S.; Tilahun, S.L.; Choon, O.H. Learning logic programming in radial basis function network via genetic algorithm. *Journal of Applied Sciences,* **2012,** 12, 9, 840-847.
22. Velavan, M.; Yahya, Z.R.; Abdul Halif, M.N.; Sathasivam, S. Mean-field theory in doing logic programming using a Hopfield network. *Modern Applied Science,* **2016**,10, 1, 154-160.
23. Mansor, M.A.; Sathasivam, S. Performance analysis of activation function in higher-order logic programming. *Advance in Industrial and Applied Mathematics: Proceedings of 23rd Malaysian National Symposium of Mathematical Sciences (SKSM23),* **2016,**1750:
24. Kasihmuddin, M.S.M. Bezier Curves Satisfiability Model in Enhanced Hopfield Network. *I.J. Intelligent Systems and Applications,* **2016,** 12, 9-17.
25. Emami, H.; Derakhshan, F. Election algorithm: A new socio-politically inspired strategy. *AI Communications*, 2015, 28, 3, 591- 603.
26. Kumar, M.; Kulkarni, A. J. Socio-inspired Optimization Metaheuristics: A Review. In *Socio-cultural Inspired Metaheuristics, Springer*, Singapore, *2019,* 241-265.
27. Gosti, G.; Folli, V.; Leonetti, M.; Ruocco, G. Beyond the Maximum Storage Capacity Limit in Hopfield Recurrent Neural Networks. *Entropy*, 2019,*21*, 8, 726
28. MohdKasihmuddin, M.S.;Mansor, M.A.; MdBasir, M.F.; Sathasivam, S. Discrete Mutation Hopfield Neural Network in Propositional Satisfiability. *Mathematics, **2019**, 7,*
29. Du, D.;Gu, J.; Pardalos, P.M. Satisfiability Problem: *Theory and Applications. American Mathematical Society,* **1997,** 35.
30. Kautz, H. ; Selman. B. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the National Conference on Artificial Intelligence.* (AAAI-96), Portland, OR,1996

31. Aloul, F. A. Symmetry in Boolean satisfiability. *Symmetry*, **2010**, *2*, 2, 1121-1134.
32. Zhang, H.; Bonacina, M.; Hsiang, J.; PSATO: a distributed propositional prove and its application to quasigroup problems. *Journal of Symbolic Computation*. **1996**, 21, 4, 543–560
33. Selman, B. and Kautz. H.Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 290–295, *1993*.
34. Franco, J.; Paull, M. Probabilistic analysis of the Davis Putman procedure for solving the Satisfiability problem. *Discrete Applied Mathematics,* **1983,** 5:77, 87.
35. Cook, S. *The Complexity of Theorem Proving Procedures.* In Proceedings of the Annual ACM Symposium on the Theory of Computing, Shaker Heights, OH, USA, **1971**, 151–158.
36. Marques-Silva, J., Lynce, I., Malik, S. Conflict-driven clause learning SAT solvers. In Handbook of Satisfiability; Biere, A., Heule, M., Van Maaren, H., Walsh, T., Eds.; IOS Press: Amsterdam, The Netherlands, **2009,** 185, 131–153
37. Hao, L.; Liu, J. Enhanced Membrane Computing Algorithm for SAT Problems Based on the Splitting Rule. *Symmetry*, **2019**, *11*, 1412
38. Biere, A. MiniSat, cadical, lingeling, plingeling, treengeling and yalsat Entering the SAT Competition. In Proceedings of the SAT Competition, **2018,** Solver and Benchmark Descriptions;
39. Zavala-Díaz, J. C., Cruz-Chávez, M. A., López-Calderón, J., Hernández-Aguilar, J. A., & Luna-Ortíz, M. E. A Multi-Branch-and-Bound Binary Parallel Algorithm to Solve the Knapsack Problem 0–1 in a Multicore Cluster. *Applied Sciences*, **2019**, *9*, 24, 5368
40. Davis, M.; Longman, G.; Loveland, D.A. Machine Program for Theorem Proving. *J. ACM,* **1962**, 5, 394–397.
41. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, **1982**, *79*, 8, 2554-2558.
42. Gosti, G., Folli, V., Leonetti, M., & Ruocco, G. Beyond the Maximum Storage Capacity Limit in Hopfield Recurrent Neural Networks. *Entropy*, **2019**,*21*, 8, 726
43. Edwards, S. F.; Anderson, P. W. Theory of spin glasses. *Journal of Physics F: Metal Physics*. **1976,** *6*, 10, 1927
44. Duong, T. L.; Nguyen, P. D.; Phan, V. D.; Vo, D. N.; Nguyen, T. T. Optimal Load Dispatch in Competitive Electricity Market by Using Different Models of Hopfield Lagrange Network. *Energies*. **2019,** *12*, 15, 2932.
45. Barra, A.; Beccaria, M.; Fachechi, A. A new mechanical approach to handle generalized Hopfield neural networks. *Neural Netw*, **2018,** 106, 205–222.
46. Bag, S.; Kumar, S.K.; Tiwari, M.K. An efficient recommendation generation using relevant Jaccard similarity. *Inf. Sci.,* **2019,** 483, 53–64
47. Peng, M.; Gupta, N.K.; Armitage, A.F.. An investigation into the improvement of local minima of the Hopfield Network. *Neural Netw*. **1996**, 90, 207–212.
48. Sulehria, H.K.; Y. Zhang, Y. *Hopfield Neural Networks. A Survey. Proceedings of WSEAS 6th International Conference on Artificial Intelligence, Knowledge Engineering and Databases* (AIKED'07), February 16-19, **2007**, Corfu, Greece, 125-130.
49. Hopfield, J.J.; Tank, D.W. Neural computation of decisions in optimization problems. *Biological Cybernetics.* ***1985,*** 52, 141-152.
50. Luzhi W., Shuli, H., Mingyang, L. and Junping, Z.An Exact Algorithm for Minimum Vertex Cover Problem.*Mathematics*, **2019,** *7*, 7, 603.
51. Creignou, N.; Kanna, S.; Sudan, M. Complexity Classifications of Boolean Constraint Satisfaction Problems; *Society for Industrial Mathematics,* **2001**, Philadelphia, PA, USA
52. Fernandez W. de la Vega. Random 2-SAT: results and problems. *Theoretical Computer Science,* **2001**, 265, 1–2, 28, 131-146.
53. Riddle P, Segal R, Etzioni O. Representation design and brute-force induction in a Boeing manufacturing domain. *Applied Artificial Intelligence an International Journal,* 1994, 1, 8, 1, 125-47.
54. Kaushik, M. Comparative analysis of exhaustive search algorithm with ARPS algorithm for motion estimation. *International Journal of Appl. Info. Systems*, **2012,** 1, 6, 16-19.

55. Abdechiri, M.; Meybodi, M.R. *A hybrid Hopfield network-imperialist competitive algorithm for solving the SAT problem*. In 3rd International Conference on Signal Acquisition and Processing (ICSAP 2011), **2011**, 2, 37-41.
56. Bernal, E.; Castillo, O.; Soria, J.; Valdez, F. Imperialist competitive algorithm with dynamic parameter adaptation using fuzzy logic applied to the optimization of mathematical functions. *Algorithms*. **2017**, *10*, 1, 18.
57. Lucas, C.; Nasiri-Gheidari, Z.; Tootoonchian, F. Application of an imperialist competitive algorithm to the design of a linear induction motor. *Energy conversion and management*, **2010**, *51*, 7, 1407-1411.
58. Vigneshwer K.; Mansor, M.A.; Kasihmuddin, M.S.M and Sathasivam, S. Hybrid imperialistic competitive algorithm incorporated with Hopfield neural network for robust 3 satisfiability logic programming. *IAES International Journal of Artificial Intelligence (IJ-AI). **2019,** 8, 2, 144~155.*
59. Willmott, C.J.; Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, **2015,** 30, 1, 79-82.
60. Chen, J.; Chen, Z., Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, **2008,** 95, 3, 759-771.