Pertanika J. Sci. & Technol. 25 (1): 139 - 152 (2017)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Hybrid Genetic Algorithm in the Hopfield Network for Logic Satisfiability Problem

Mohd Shareduwan Mohd Kasihmuddin*, Mohd Asyraf Mansor and Saratha Sathasivam

School of Mathematical Sciences, University Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia

ABSTRACT

In this study, a hybrid approach that employs Hopfield neural network and a genetic algorithm in doing k-SAT problems was proposed. The Hopfield neural network was used to minimise logical inconsistency in interpreting logic clauses or programme. Hybrid optimisation made use of the global convergence advantage of the genetic algorithm to deal with learning complexity in the Hopfield network. The simulation incorporated with genetic algorithm and exhaustive search method with different k-Satisfiability (k-SAT) problems, namely, the Horn-Satisfiability (HORN-SAT), 2-Satisfiability (2-SAT) and 3-Satisfiability (3-SAT) will be developed by using Microsoft Visual C++ 2010 Express Software. The performance of both searching techniques was evaluated based on global minima ratio, hamming distance and computation time. Simulated results suggested that the genetic algorithm outperformed exhaustive search in doing k-SAT logic programming in the Hopfield network.

Keywords: Genetic Algorithm, Exhaustive Search, Hopfield network, Satisfiability, Logic Programming, HORN-SAT, 3-SAT, 2-SAT

INTRODUCTION

Artificial intelligence is one of the most eminent and staple fields in Mathematics, Physics and Computer Science (Bekir & Vehbi, 2010). Strictly speaking, the integration of neural network, logic programming, satisfiability problem and neuro-searching approach as a single hybrid network is a brand new paradigm in artificial intelligence. There are numerous types of neural networks, from modest to intricate, such as the Hopfield neural network introduced by Hopfield and Tank (1985). The Hopfield neural network is a simple recurrent network which can serve as an efficient associative memory and store definite

Article history:
Received: 26 January 2016
Accepted: 30 August 2016

E-mail addresses:
iwanmaidin@gmail.com (Mohd Shareduwan Mohd Kasihmuddin),
asyrafalvez@live.com (Mohd Asyraf Mansor),
saratha@usm.my (Saratha Sathasivam)
*Corresponding Author

ISSN: 0128-7680

memories in a manner rather similar to the brain (Rojas, 1999; Sathasivam et al., 2013). Moreover, it is a branch of neural network that has been applying a vast Mathematical problems such as Travelling Salesperson problem (TSP) and hard satisfiability problem (Haykin, 1992). Next, logic programming can be treated as a problem in combinatorial optimisation perspective (Kowalski, 1979). Hence, it can be implemented and assimilated in a neural network to hunt desired solutions (Hamadneh et al., 2013). Recently, the conventional and common model is Wan Abdullah's logic programming (Wan Abdullah, 1993). In this paper, we will combine the advantages of the Hopfield network, logic programming and neuro-searching methods to do the satisfiability problem.

Traditionally, neuro-searching methods, such as exhaustive search and metaheuristic, can be implemented as a mechanism of doing the k-SAT problems. The most widely used technique is exhaustive search as it is a simple algorithm (Hooker, 2005). Theoretically, this technique considers the whole search space in order to check the clause satisfaction for the k-SAT problems, namely, Horn-SAT, 2-SAT and 3-SAT. However, the exhaustive search can be applied only if the problem size or the number of clause is limited (Mark & Lee, 1992). Another limitation is that the exhaustive search typically consumes more time to complete the whole searching process (Tobias & Walter, 2004; Kaushik, 2012). In this paper, we proposed a metaheuristic approach, the genetic algorithm (GA), to obtain the satisfied interpretation within acceptable timescales. On the other hand, genetic algorithm is an evolutionary algorithm that involves iterative procedures combining the exploration and exploitation process within any search space (Holland, 1975; John, 2005; Aiman & Asrar, 2015). Hence, sense of balance in both processes is vital to improve the convergence of the algorithm (Siddique, & Adeli, 2013).

This paper is organised as follows. Section 2 introduces the important concept of the Hopfield neural network, logic programming, satisfiability problem, exhaustive search (ES) and genetic algorithm (GA). In section 3, the fundamental theory of k-satisfiability (k-SAT) problems, namely, Horn-Satisfiability (HORN-SAT), 2-Satisfiability (2-SAT) and 3-Satisfiability are discussed. Section 4 covers a brief discussion of neuro-logic, which basically revolves around the Hopfield neural network and logic programming. Meanwhile, section 5 presents the neuro-searching methods involved in this research including Exhaustive search (ES) and Genetic algorithm (GA). In section 6, the theory implementation of the networks is discussed. Finally, sections 7 and 8 enclose the experimental results and offer conclusion of this research.

SATISFIABILITY (SAT) PROBLEM

Satisfiability or SAT is a rudimentary problem in computer science. The problem is to determine whether a truth assignment to variables appearing in a Boolean formula ϕ is satisfied (Kowalski, 1979). Therefore, a Boolean formula is satisfiable if an assignment of true and false values renders the entire expression true. For a problem of size n, there will be 2^n such assignments and l literals to set for each assignment (Sathasivam & Sagir, 2014), in which such an approach requires $O(l.2^n)$ operations (Gu, 1999). Hence, SAT is an NP-complete problem in general. For instance, the satisfiability problem concerns Boolean variables or expressions in conjunctive normal form (CNF). CNF comprises of conjunction of clauses, where clauses are disjunctions of literal (Sathasivam *et al.,* 2013). Meanwhile, literal is a variable or its negation. For example:

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee x_4) \quad (1)$$

Here x_1, x_2, x_3, x_4 are Boolean variables to be assigned, $\neg$ which refer to negations (logical NOT), $\vee$ means negations (logical OR), $\wedge$ means negations (logical AND). One may note that the formula above is satisfiable when $x_1 = true, x_2 = false, x_3 = false, x_4 = true$, where it takes on the value of true. However, if a formula is not satisfiable, it is called unsatisfiable, which means that it takes on the value false on any value of its variable. The basic form of k-SAT is simplified, as follows:

$$\text{k-SAT} : P \rightarrow \{\text{YES}, \text{NO}\} \quad (2)$$

2-Satisfiability (2-SAT)

2-SAT is the problem of deciding satisfiability of sets of clauses with at most two literals per clause (2-CNF formulas). It is a special case of general Boolean satisfiability which can involve constraints on two variables (Kowalski, 1979). These variables can allow two choices for the value of each variable. 2-SAT problem can be expressed as 2-CNF (2-Conjunctive Normal Form) or Krom formula (Fernandez, 2011). In contrast, 2-SAT problem is considered as a NP problem or non-deterministic problem. The three components of 2-SAT are summarised as follows:

1. A set of m variables, $x_1, x_2, ..., x_m$
2. A set of literals. A literal is a variable or a negation of a variable.
3. A set of n distinct clauses: $C_1, C_2 ..... C_n$. Each clause consists of only literals combined by just logical OR ($\vee$). Each clause must consist of 2 variables.

The Boolean values are $\{1, -1\}$. Researchers have emphasised the True and False in the neural networks by 1 and -1. Due to this, the goal of the 2-SAT problem is to determine whether there exists an assignment of truth values to variables that makes the following formula satisfiable.

$$P = \bigwedge_{i=1}^{n} C_i \quad (3)$$

Where $\wedge$ is a logical AND connector, *P* denotes the entire Boolean formula for 2-SAT. C_i is a clausal form of DNF with 2 variables. Each clause in 2SAT has the following form:

$$C_i = \bigvee_{i=1}^{n} (x_i, y_i) \quad (4)$$

$x_i \in \{k_i, \neg k_i\}$ and $y_i \in \{r_i, \neg r_i\}$ $\neg k_i$ and $\neg r_i$ are negations of the literals.

3-Satisfiability (3-SAT)

In this paper, we emphasize a paradigmatic NP-complete problem namely 3-Satsifiability (3-SAT). Generally speaking, 3-SAT can be defined as a formula in conjunctive normal form, where each clause is limited to at most or strictly three literals (Vilhelm et al., 2005), thus

the problem is an example of a non-deterministic problem (Tobias & Walter, 2004). In our analysis, the following 3-SAT logic programme consisting of 3 clauses and 3 literals will be used. For instance:

$$P = \left(A \vee B \vee \overline{C}\right) \wedge \left(\overline{A} \vee \overline{B} \vee C\right) \wedge \left(\overline{A} \vee B \vee D\right) \tag{5}$$

We represent the above 3 CNF formula with P. The formula can be in any combination as the number of atoms can be varied, except for the literals that are strictly equal to 3, which is vital for the combinatorial optimisation problem. Thus, the higher number of literals in each clause will increase the possibilities or chances for a clause to be satisfied. The general formula of 3-SAT for conjunctive normal form (CNF):

$$P = \bigwedge_{i=1}^{n} Z_i \tag{6}$$

Hence, the value of k denotes the number of satisfiability. In our case, k-SAT is 3-SAT.

$$Z_i = \bigwedge_{j=1}^{k} (x_{ij}, y_{ij}, z_{ij}), k > 3 \tag{7}$$

Horn-Satisfiability (HORN-SAT)

Horn-Satisfiability can be defined as a clause with at most one positive literal. It comprises of any individual atom in the head and literals in their body. Therefore, the general form of horn clauses is illustrated as follows:

$$P = C_i \leftarrow \bigwedge_{j=1}^{m} D_j \tag{8}$$

For the HORN-SAT clause, we can construct the logic in a form of $P = C_1, C_2, \mathrm{C}_3 ...., \mathrm{C}_n \leftarrow D_1, D_2, \mathrm{D}_3 ...., \mathrm{D}_m$. In this study, P should be satisfied. The truth assignment is vital as in 3-SAT previously. Thus, the HORN-SAT formula comprised at most one positive literal is as follows:

$$P = \left(A \vee \neg B \vee \neg C\right) \wedge \left(D \vee \neg B\right) \wedge C \tag{9}$$

Works on HORN-SAT have been done by Wan Abdullah (1992) and Sathasivam (2010). One of the important features of HORN-SAT is that the formula is always satisfiable.

NEURO-LOGIC IN THE HOPFIELD NEURAL NETWORK

The Hopfield Model

The Hopfield model is a standard model for content addressable memory (CAM). The units in Hopfield nets are binary threshold unit (Haykin, 1992), which can only take binary values such as 1 and -1. The possible definitions for unit I's activation, a_i are:

$$a_i = \begin{cases} 1 \quad if \sum_j W_{ij} S_j > \xi_i \\ -1 \; Otherwise \end{cases} \tag{10}$$

Where, W_{ij} is the connection strength from units j to i. S_j is the state of unit j and ξ_i is the threshold of unit i. The connection in the Hopfield net typically has no connection with itself, $W_{ij} = 0$ and connections are symmetric or bidirectional (Sathasivam et al., 2013). The system consists of N formal neurons, each is described by an Ising variable. Neurons are bipolar. $S_i \in \{1, -1\}$ is obeying the dynamics $S_i \to \text{sgn}(h_i)$, where the local field h_i is. The connection model can be generalised to include higher order connection. This modifies the field to:

$$h_i = \sum_j W_{ij}^{(2)} S_j + J_i^{(1)} \tag{11}$$

The weight in the Hopfield network is always symmetrical. The updating rule maintains:

$$S_i(t+1) = \text{sgn}\left[h_i(t)\right] \tag{12}$$

This properties guarantee that the energy will decrease monotonically while following the activation system. The following equation represents the energy for the Hopfield network.

$$E = ... - \frac{1}{2}\sum_i \sum_j W_{ij}^{(2)} S_i S_j - \sum_i W_i^{(1)} S_j \tag{13}$$

Logic Programming in the Hopfield network

In essence, logic programming can be seen as a problem in combinatorial optimisation and it can be carried out on a Hopfield network (Sathasivam et al., 2013). Furthermore, this can be done by using the neurons to store the truth values of the literal and writing a cost function, which is minimised when all clauses are satisfied (Wan Abdullah, 1993). In other words, the main task is to find the 'models' corresponding to the given logic programme. The fundamental of the Hopfield network in doing logic programming was brought up due to its unique content on addressable memory properties.

Implementation of k-SATGA in the Hopfield neural network (HNN-kSATGA)

i. Translate all the k-SAT clauses into Boolean algebra. Identify a neuron to each ground neuron.

iii. Initialise all connection strengths to zero.

iv. Derive a cost function that is associated with the negation of all k-SAT clauses. For example, $X = \frac{1}{2}(1 + S_X)$ and $\overline{X} = \frac{1}{2}(1 - S_X)$. $S_X = 1$ (True) and $S_X = -1$ (False). Multiplication represents CNF and addition represents DNF.

v. Compare the cost function with energy, E, by obtaining the values of the connection strengths.

vi. Check clause satisfaction by using k-SATGA. Satisfied clauses will be stored.

vii. Randomise the states of the neurons. The network undergoes a series of network relaxation. Calculate the corresponding local field $h_i(t)$ of the state. If the final state is stable for 5 runs, we consider it as the final state.

viii. Find the corresponding final energy E of the final state by using lypunov equation. Verify whether the final energy obtained is a global minimum energy or local minima. Calculate the corresponding hamming distance and computation time.

THE NEURO-SEARCH TECHNIQUES

Exhaustive search algorithm

Strictly speaking, exhaustive search is the simplest algorithm but can be computationally super expensive. In this algorithm, it exhaustively searches for the entire possible clause even though the search space is getting tremendous. The main advantage of this algorithm is the guarantee to obtain a solution (satisfied clause) by taking into a consideration the entire search space (Rojas, 1999). However, the exhaustive search consumes more computation time or CPU time to hunt for the satisfied interpretation completely (Kaushik, 2012; Asrar & Aiman, 2015). In exhaustive search, we will check the clause satisfaction directly for the k-SAT problem until a satisfying one is found (Tobias & Walter, 2004). In our case, we are required to seek for the satisfied clause during the training phase for HORN-SAT, 2-SAT and 3-SAT problem.

In this paper, we emphasise the complexity of the network as we venture with a higher number of neurons. Hence, the computation time will be slower if the complexity of the network increases. Generally for k-SAT problem, there are potentially 2^n satisfying assignments. Thus, the run-time complexity is equivalent to $O(2^n)$. The correct interpretation will be stored into Hopfield's brain as content addressable memory (CAM). In this paper, we will implement this algorithm together with the Hopfield neural network, logic programming and satisfiability problem.

Genetic algorithm

Genetic algorithm is a staple computational paradigm inspired from the Darwin's model, namely, survival for the fittest model. Darwin stated that the survival of an organism can be maintained through the processes of reproduction, crossover and mutation (Holland, 1975). Hence, it can be implemented in the Mathematical model and become one of good heuristic methods. For instance, every generation is represented by an array of bit strings analogous to the chromosomes of DNA. The core impetus of our approach is to hunt for the fittest assignment or satisfied clause given any k-SAT clauses. Besides, the fittest assignment gives the maximum number of satisfied clauses, which depends on the number of satisfied clauses and can be calculated as follows:

$$f_{k-SAT}(x) = c_1(x) + c_2(x) + c_3(x) ..... + c_{total\,NC}(x) \tag{14}$$

Theoretically, HNN-kSATGA is able to scan and search for the satisfied interpretation with the highest fitness value systematically. On the other note, the procedures do not require complex Mathematics to execute and are easy to implement to solve any constrained optimisation

problem. In GA, the initial set of population will be generated by using bit string as a chromosome. Moreover, the fitness value will be computed according to the truth value of each chromosome. During the crossover process, two chromosomes were randomly selected and broken from a randomly selected crossover locus (John, 2005). In addition, a child chromosome is produced by linking the first part and second part of the parent chromosome. Correspondingly, the second child is formed by joining the second part of the first chromosome and the first part of the second chromosome (Luke, 2013). Thus, we can compute the fitness value for the newly formed chromosomes. The crossover operator mimics the biological combination between two single-chromosomes (haploid) in organisms (Rojas, 1999).

During mutation, a newly child chromosome is selected to form a set of bit strings. If the fitness value is still lower than the maximum fitness, the random bit will be improved during this stage. Besides that, the number of bits being complemented or flipped depends on the mutation rate. In this paper, we are using the crossover rate equal to 0.9. According to Aiman and Asrar (2015), the perfect mutation rate is usually less than 1. Then, the fitness will be computed for the new chromosome (bit strings). If the fitness value matches the maximum fitness, we print the output as our desired satisfied interpretation. Figure 1 shows the algorithm for this paradigm.

THEORY IMPLEMENTATION

For the implementation, firstly, random k-SAT clauses were generated. From there, the initial states were initiated for the neurons in the clauses. The network will evolve until the final state is reached. Once the programme has reached the final state, the neuron state is updated via equation (11). As soon as the network is relaxed to an equilibrium state, the final state obtained for the relaxed neuron is tested to determine whether it is a stable state. Furthermore, stable state will be considered, provided the state remains unchanged for five runs. According to Pinkas and Dechter (1995), letting an ANN to evolve will eventually lead to stable state where the energy function obtained does not change further. In this case, the corresponding final energy for the stable state will be calculated. If the difference between the final energy and the global minimum energy is within the given tolerance value, then the solution is considered as a global solution.

In addition, the simulations are performed on Microsoft Visual C++ 2010 Express Software. Each satisfiability problem will be repeated with 100 different combination neurons and each neuron combination will undergo 100 trials to reduce the statistical error, in which the selected tolerance value is 0.001. According to Sathasivam et al. (2013), 0.001 was selected because it gave a better performance than other values. Other than that, connection strength can be obtained by using either Sathasivam's method or Hebbian Rule. Both methods will result in similar strength (Sathasivam & Sagir, 2014) because both methods consider the same knowledge base (clauses).

RESULTS AND DISCUSSION

Global Minima Ratio

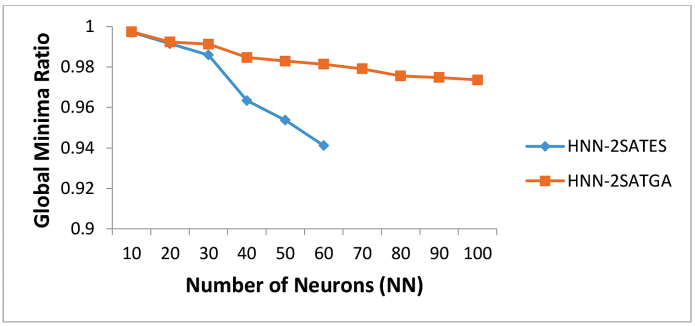

Figure 2. Global Minima Ratio for HNN-2SATES and HNN-2SATGA

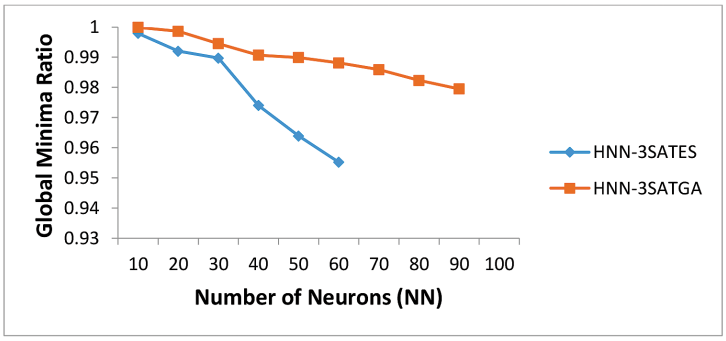

Figure 3. Global Minima Ratio for HNN-3SATES and HNN-3SATGA

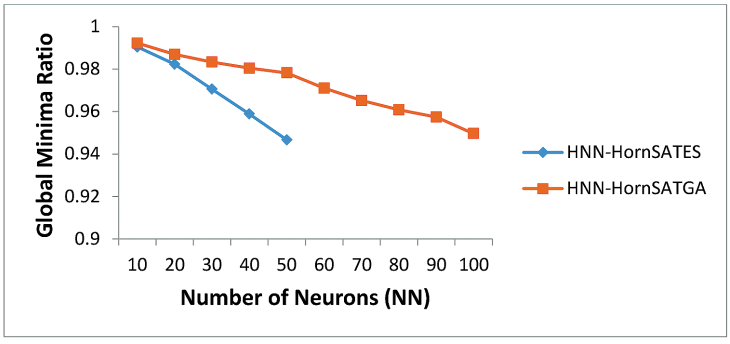

Figure 4. Global Minima Ratio for HNN-HornSATES and HNN-HornSATGe

The results of the HNN-kSATES and HNN-kSATGA are summarised in Figure 2 through Figure 4. Based on these figures, HNN-kSATGA outperforms HNN-kSATES in terms of global minima ratio. Also, HNN-kSATGA is able to recall more correct states compared to

HNN-kSATES. When the number of neurons increased, HNN-kSATGA is able to sustain more neurons. The limit for HNN-kSATES is 60 neurons. After 60 neurons, the network is stuck in trial and error state in a long period of time. The whole bitstring in exhaustive search method will be collapsed when one of the neuron-clauses is not satisfied. In addition, it might take a longer time for the network to search the correct neuron states and proceed with the relaxation state, as the network spends more time in training state. On the other hand, HNN-kSATGA can sustain more neurons because genetic algorithm reduces the complexity of the network to find the correct states. Apart from that, unsatisfied neuron clauses will be improved through crossover among the best offspring and undergo mutation until we found the correct neuron state. Besides, giving more relaxation time for the network will help the network to retrieve the state more effectively. Thus, less relaxation time will create spurious minima which will cause the retrieved solution to achieve local minima.

Hamming Distance

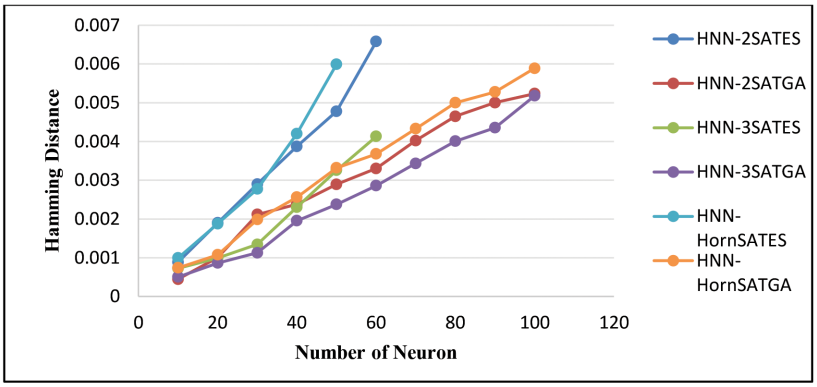

Figure 5. Hamming distance for different HNN- kSATGA and HNN-kSATES.

Figure 5 depicts the obvious variation in Hamming distance obtained by HNN-kSATGA and HNN-kSATES. In this study, Hamming distance is the closeness of bits between the training state and global state (retrieved state) of the neurons upon relaxation process. The plot shows that HNN-kSATGA consistently performed better than HNN-kSATES. Hence, the ability to recall accurate interpretations (training phase) and correct states (testing phase) was improved drastically by implementing Genetic algorithm (GA) in the Hopfield neural network for any k-SAT problem. This is due to the power of GA in ascertaining the satisfied clause, especially during the crossover stage, where the clause was being improved by certain rate to achieve the highest fitness value. Additionally, HNN-kSATGA would be able to recall the correct states that contributed to the lower hamming distance. Conversely, the exhaustive search algorithm emphasised the trial and error process during clause satisfaction process. When the complexity increased, HNN-2SATES and HNN-HORNSAT were able to sustain up to 50 neurons and HNN-3SATES with the limitation until 60 neurons. The main reason is due to the nature of exhaustive search that increases the computation burden in searching

for the correct neuron states. On the contrary, HNN-HornSATGA, HNN-2SATGA and HNN-3SATGA were able to sustain up to 100 neurons. Hence, the ability to sustain huge number of neurons is due to the special ability of GA that reduces the computation burden in hunting the correct states.

Computation Time

Table 1
Computation time for different HNN-kSATGA and HNN-kSATES.

Number of Neurons (NN)	Computation Time					
	HNN-2SATES	HNN-2SATGA	HNN-3SATES	HNN-3SATGA	HNN-HornSATES	HNN-HornSATGA
10	4	3	2	1	8	4
20	72	8	30	9	145	57
30	208	18	88	22	4566	149
40	759	33	202	46	33890	280
50	8036	53	4601	99	260953	372
60	75472	78	19455	134		455
70		111		268		652
80		143		371		899
90		181		404		1004
100		231		450		2345

The computation time or CPU time can be defined as the time taken for a logic programme to generate the global solutions including the training process. According to Table 1, the computation time for the HNN-Ksatga is faster than HNN-kSATES. For instance, the complexity of the network increased as the network got massive. We can see that the computational time increased when the number of neuron was getting higher. This is due to the condition when the network was getting larger and complex, the network was likely to get stuck in local minima and devour more computation time. As a consequence of these arguments, extra time was needed to relax to global solution as the number of neurons increased. Moreover, the neurons needed to jump enormous energy barrier to reach the global solutions. On a separate note, the training process by using exhaustive search usually consumes more computational time due to the trial and error processes in hunting for the satisfied interpretation. However, when genetic algorithm was implemented, the computation time was faster due to the crossover and mutation processes that turned the unsatisfied clause into satisfied clause systematically. The computation time obtained for HORN-SAT, 2-SAT and 3-SAT indicated that HNN-kSATGA outperformed HNN-kSATES in terms of computation time.

CONCLUSION

Inspired by the fundamental of artificial intelligent and nature inspired optimization, an optimal k-SAT model was developed in the Hopfield network. In general, genetic algorithm incorporated with the Hopfield network can sustain k-SAT patterns. Computer simulations were carried out to verify the ability of the Hopfield network doing k-SAT logic. HNN-kSATGA gives us global minima ratio of approximately 1 and hamming distance, which is approximately 0 compared to HNN-kSATES. On the basis of the above illustration, it has been shown in computer simulation that genetic algorithm successfully reduced the complexity of the network, produced more ideal solutions and had a smaller error compared to the traditional exhaustive search method in the Hopfield network.

REFERENCES

Abdullah, W. A. T. W. (1993). The logic of neural networks. *Physics Letters A*, *176*(3), 202-206.

Aiman, U., & Asrar, N. (2015). Genetic algorithm based solution to SAT-3 problem. *Journal of Computer Sciences and Applications*, *3*(2), 33-39.

Bekir, K., & Vehbi, A. O. (2010). Performance analysis of various activation functions in generalized MLP architectures of neural network. *International Journal of Artificial Intelligence and Expert Systems*, *1*(4), 111-122.

Fernandez, W. (2001). Random 2-SAT: Result and Problems. *Theoretical Computer Science*, *265*(1), 131-146.

Gu, J. (1999). The Multi-SAT algorithm. *Discrete Applied Mathematics*, *96*, 111-126.

Hamadneh, N., Sathasivam, S., Tilahun, S. L., & Ong, H. C. (2013). Prey-Predator Algorithm as a New Optimization Technique Using in Radial Basis Function Neural Network. *Research Journal of Applied Sciences*, *8*(7), 383-5.

Haykin, S. (1992). *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Michigan: The University of Michigan Press.

Hooker, J. N. (2005). Unifying local and exhaustive search. In L. Villasenor & A. I. Martinez (Eds.), *Proceeding of ENC 2005 - Sixth Mexican International Conference on Computer Science* (pp. 237-243). IEEE Press.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernatics*, *52*(3), 141-152.

Iwama, K. (1989). CNF Satisfiability Test by Counting and Polynomial Average Time. *SIAM, Journal of Computer, 4*(1), 385-391.

John, M. (2005). Genetic algorithm for modelling and optimization. *Journal of Computational and Applied Mathematics*, *184*, 205-222.

Kaushik, M. (2012). Comparative analysis of exhaustive search algorithm with ARPS algorithm for motion estimation. *International Journal of Applied Information Systems*, *1*(6), 16-19.

Kowalski, R. A. (1979). *Logic for Problem Solving*. New York: Elsevier Science Publishing.

Luke, S. (2013). *Essentials of metaheuristics* (2nd edition). United States: Lulu.

Mark, W. G., & Lee C. G. (1992). Routing in random multistage interconnections networks: Comparing exhaustive search, greedy, and neural network approaches. *International Journal of Neural System*, *2*(3), 125-142.

Pinkas, G., & Dechter, R. (1995). Improving energy connectionist energy minimization. *Journal of Artificial Intelligence Research*, *3*, 223-237.

Razgon, I., & O' Sullivan, B. (2009). Almost 2-SAT is Fixed Parameter Tractable. *Journal of Computer Sciences and System Sciences*, *75*(8), 435-450.

Rojas, R. (1999). *Neural Networks: A Systematic Introduction*. Berlin: Springer.

Sathasivam, S. (2010). Upgrading Logic Programming in Hopfield Network. *Sains Malaysiana*, *39*(1), 115-118.

Sathasivam, S., Ng, P. F., & Hamadneh, N. (2013). Developing agent based modelling for reverse analysis method. *Journal of Applied Sciences, Engineering and Technology*, *6*(22), 4281-4288.

Sathasivam, S., & Sagir, A. M. (2014). An Overview of Hopfield Network and Boltzmann Machine. *International Journal of Computational and Electronics Aspects in Engineering*, *1*(1), 20-26.

Siddique, N., & Adeli, H. (2013). *Computational Intelligence Synergies of Fuzzy Logic, Neural Network and Evolutionary Computing.* United Kingdom: John Wiley and Sons.

Tobias, B., & Walter, K. (2004). An improved deterministic local search algorithm for 3-SAT. *Theoretical Computer Science*, *329*(1), 303-313.

Vilhelm, D., Peter, J., & Magnus, W. (2005). Counting models for 2SAT and 3SAT formulae. *Theoretical Computer Science*, *332*(1), 265-291.

APPENDIX

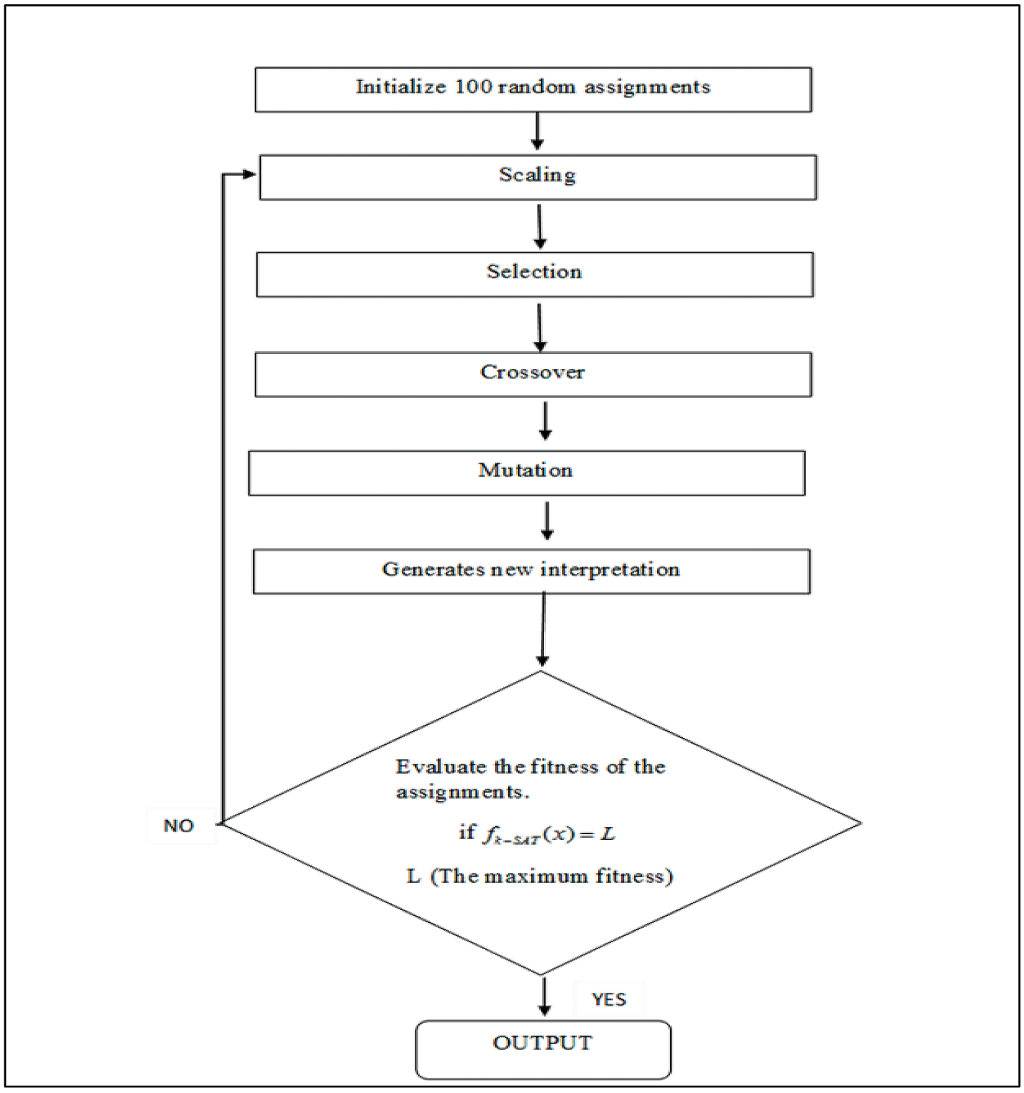

Figure 1. Flowchart for Genetic Algorithm