Pertanika J. Sci. & Technol. 25 (2): 453 - 468 (2017)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

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

Robust Artificial Bee Colony in the Hopfield Network for 2-Satisfiability Problem

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

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

ABSTRACT

Swarm intelligence is a research area that models the population of swarm that is able to self-organise effectively. Honey bees that gather around their hive with a distinctive behaviour is another example of swarm intelligence. In fact, the artificial bee colony (ABC) algorithm is a swarm-based meta-heuristic algorithm introduced by Karaboga in order to optimise numerical problems. 2SAT can be treated as a constrained optimisation problem which represents any problem by using clauses containing 2 literals each. Most of the current researchers represent their problem by using 2SAT. Meanwhile, the Hopfield neural network incorporated with the ABC has been utilised to perform randomised 2SAT. Hence, the aim of this study is to investigate the performance of the solutions produced by HNN2SAT-ABC and compared it with the traditional HNN2SAT-ES. The comparison of both algorithms has been examined by using Microsoft Visual Studio 2013 C++ Express Software. The detailed comparison on the performance of the ABC and ES in performing 2SAT is discussed based on global minima ratio, hamming distance, CPU time and fitness landscape. The results obtained from the computer simulation depict the beneficial features of ABC compared to ES. Moreover, the findings have led to a significant implication on the choice of determining an alternative method to perform 2SAT.

Keywords: Artificial Bee Colony Algorithm, exhaustive search, hopfield network, satisfiability, logic programming, 2SAT

INTRODUCTION

Artificial Neural Networks (ANNs) have been successfully applied in solving infinite applications such as classification, function approximation, optimisation and associative memory. Nonetheless, the success of neural networks in constraint optimisation problem largely depends on their architecture and

Article history:
Received: 04 April 2016
Accepted: 21 July 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

solution searching techniques (Haykin, 1999). In some approaches, the integration of neural network, logic programming, satisfiability problem and neuro-searching techniques has been proven to minimise the complexity of the network. One of the earliest neural networks that resembles how human brain actually works is the Hopfield neural network. The Hopfield neural network, which was introduced by Hopfield and Tank (1985), is a simple recurrent network that can serve as an efficient associative memory and store definite memories with exceptional retrieval power (Rojas, 1999; Sathasivam et al., 2013). Moreover, it is a branch of the neural networks that has been applied in vast combinatorial problems such as Travelling Salesperson problem (TSP) and hard satisfiability problem (Haykin, 1992). For instance, logic programming can be treated as a problem in combinatorial optimisation perspective (Kowalski, 1979). Previously, logic programming has been implemented and assimilated in a neural network to search desired solutions (Hamadneh et al., 2013). In this paper, the advantages of the Hopfield network, logic programming and neuro-searching methods are combined to solve satisfiability problem.

Moreover, some neuro-searching methods, such as Exhaustive Search (ES) and meta-heuristic, can be implemented as a mechanism in solving satisfiability problems. The most widely used technique is the ES because it is a simple algorithm (Hooker, 2005). Conventionally, this particular technique considers all possible search spaces in order to verify clauses satisfaction for satisfiability problems. However, the ES can be applied only if the problem size or the number of clause is limited (Mark & Lee, 1992). Another limitation is that the ES typically consumes more time to complete the whole searching process (Tobias & Walter, 2004; Kaushik, 2012). Over the past decade, the growing interest in computational swarm intelligence has caused the emergence of several new optimisation algorithms. Recently, an algorithm based on the model of bee foraging behaviour was developed. Artificial bee colony (ABC) was first introduced by Karaboga (2005). Since ABC is simple in concept, easy for implementation and has fewer parameters (Pan et al., 2011), it has attracted the attention of many researchers to solve constraint optimisation.

In this paper, a meta-heuristic approach from the ABC is proposed to obtain the possible satisfied assignments within acceptable timescales. More specifically, the motivation of this paper is to apply the ABC as a searching technique incorporated with the Hopfield neural network in performing logic programming. A good searching technique during neural network simulation is vital to improve the convergence of the algorithm (Siddique et al., 2013).

This paper has been organised as follows. In Section 2, the fundamental theory of random 2-satisfiability (2SAT) problems is discussed. Moving on, Section 3 presents the neuro-searching methods employed in this research, including ES and ABC. Next, Section 4 provides a brief discussion of the neuro-logic, which explains the Hopfield neural network, content addressable memory (CAM) and logic programming in neuro symbolic integration. Meanwhile, the theory implementation of the networks is discussed in Section 5. Finally, Sections 6 and 7 enclose the experimental results and the conclusion of this exploration.

SATISFIABILITY (SAT) PROBLEM

Satisfiability or SAT is a significant problem in the computer science field. This problem revolves in determining if a truth assignment to variables that appears in a Boolean formula ϕ is satisfied (Kowalski, 1979). The Boolean formula is said to be satisfiable if an assignment of true and false values renders the entire expression as true. One way to solve SAT would be by trying out every possible truth of the assignment. 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 the 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, $\neg$ refers to negations (logical NOT), $\vee$ means negations (logical OR), and $\wedge$ means negations (logical AND). Thus, the formula above is satisfied when $x_1 = true, x_2 = false, x_3 = false, x_4 = true$, where it takes on the value of true.

2-Satisfiability (2SAT)

2SAT is the problem of deciding the 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). Besides, the 2SAT paradigm can allow two choices for the value of each variable. Normally, 2SAT problem can be expressed as 2-CNF (2-Conjunctive Normal Form) or Krom formula (Fernandez, 2011). In contrast, randomised 2SAT problem is considered as an NP problem or a non-deterministic problem. The three components of 2SAT are summarised in the following:

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 $\{1, -1\}$. In fact, researchers have emphasised the True and False in the neural networks by 1 and -1. Due to this, the goal of the 2SAT problem is to determine if an assignment of truth values to variables does exist, which makes the following formula satisfiable.

$$P = \bigwedge_{i=1}^{n} C_i \quad [2]$$

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

$$C_i = \bigvee_{i=1}^{n} (x_i, y_i) \quad [3]$$

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

NEURO-SEARCH TECHNIQUES

Exhaustive Search (ES)

Generally, ES is the simplest algorithm, but it can be computationally super expensive. In this algorithm, it exhaustively searches for the entire possible clause even though the search space gets larger. The main advantage of this algorithm is the guarantee to obtain a solution (satisfied clause) by taking into consideration the entire search space (Rojas, 1999). Furthermore, the modification of ES (Ata & Coban, 2015) might increase the efficiency of the algorithm through numerous assumptions and conditions. Moreover, the ES has been proven to consume more computation time or CPU time to hunt for the satisfied interpretation completely (Kaushik, 2012; Asrar & Aiman, 2015). In ES, the clause satisfaction is determined directly for the randomised 2SAT problem until a satisfying one is found (Tobias & Walter, 2004). Meanwhile, in this particular case, the satisfied clause has been sought during the training phase for 2SAT.

The complexity of the network has a positive correlation to a number of neurons. As the number of candidate solution is increased, the complexity of the searching technique will increase. In some cases, it may lead to combinatorial explosion (Gagneur & Klamt, 2004). Generally, for randomised 2SAT problem, there are potentially 2^n candidate solution and run-time complexity, which are equivalent to $O(2^n)$. Additionally, the satisfied assignments are obtained after performing trial-and-error processes. If the algorithm successfully finds an incorrect assignment, the algorithm will reset the whole search space. The correct assignment will be stored as a graded pattern as CAM. In this paper, this algorithm, together with the Hopfield neural network, logic programming, and satisfiability problem, has been implemented.

Artificial Bee Colony (ABC)

The ABC algorithm is a swarm-based meta-heuristic algorithm that was popularised by Karaboga (2005) in order to optimise numerical problem. It was inspired by the intelligent foraging behaviour of honey bees (Muthiah & Rajkumar, 2014). In fact, some papers published by Civicioglu and Besdok (2013) and Karboga and Basturk (2008) discovered that ABC had outperformed some meta-heuristic methods such as cuckoo-search and particle swarm optimisation. The model consists of three essential components, namely, employed and unemployed bees, as well as scout bees. The first two components, employed and unemployed bees, search for rich food sources. These two components are necessary for self-organising

and collective intelligence. Besides, recruitment of foragers to rich sources that results in positive feedback can help us to achieve our desired solution. If the solution obtained from the first two components does not meet the criteria set, the algorithm will spawn scout bees. Scout bees will try to find alternative food source (solution). The act of scout bees will reset the whole search algorithm. This strategy prevents the algorithm from having local maxima (non-improving) solution.

General Artificial Bee Colony (ABC) Algorithm

1) Initially, food sources are produced for all employed bees.
2) Each employed bee group goes to a food source and checks a neighbour source. They will evaluate its nectar amount and dance in the hive.
3) Each onlooker bees group will watch the dance of the employed bees and choose one of their sources, depending on the dances and then, goes to that source. After choosing a neighbour around them, it will evaluate its nectar amount. The best food source will be registered.
4) If the food source is not the desired food source, it will be abandoned. The abandoned food sources are determined and replaced with a new food source. The scout bee will reset the food source. The new set of food sources will be discovered by the scout bees.
5) Repeat steps 2 until 4. The best output will be recorded.

The difference between employed bees, onlooker bees and scout bees is that the employed bees share their food source information with the onlooker bees, while waiting in the hive (Karboga & Basturk, 2007). From the information retrieved from the employed bees, the onlooker bees would probably choose their food sources depending on this information. Moreover, these onlooker bees choose a food source depending on the probability values calculated by using the fitness values provided by the employed bees. Meanwhile, the unemployed bees that choose their food sources randomly are called scouts. These employed bees, whose solution cannot improve through a pre-determined number of trials specified by user, are called "limit". These employed bees will be converted to bee scouts and their solutions are abandoned. The scout bees will search for a new set of solution randomly.

Binary Artificial Bee Colony (ABC) Algorithm

Binary ABC has been implemented by several researchers (Kashan et al., 2012) for binary optimisation. As for this study, ABC was utilised to hunt the fittest assignment or the highest satisfied clause given at any randomised 2SAT clause. Since only binary value (1 and -1) had been considered for this representation, the traditional ABC was improved to produce binary solutions. Therefore, the fittest assignment gave the maximum number of satisfied clauses, which depended on the number of satisfied clauses, and can be calculated as follows:

$$fit_i(x) = c_1(x) + c_2(x) + c_3(x) ..... + c_{total\ NC}(x) \quad [4]$$

Basically, the employed bees will find solution (assignments) in a search space. The journey to find the food source (solution) can be evaluated by using the following equation (Jia et al., 2014).

$$v_{ij} = x_{ij} \vee \left(\phi_{ij} \otimes \left(x_{ij} \wedge x_{kj} \right) \right) \quad [5]$$

where

ϕ_{ij} parameter where

$$\phi_{ij} = \begin{cases} 1, rand(0,1) < 0.5 \\ -1, rand(0,1) \geq 0.5 \end{cases}$$

$\otimes$ is a 'XOR' operator

$\wedge$ is an 'AND' operator

$\vee$ is an 'OR' operator

Once all the employed bees have returned to their hive, they will dance. The information transfer occurs during the dance. Each employed bee has its own fitness. Fitness is evaluated based on the number of (satisfied) clauses. The dancing, on the other hand, is observed by the onlooker bees. The onlooker bees will choose the information based on the following probability of roulette wheel selection (Goldberg, & Deb, 1991).

$$p_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \quad [6]$$

Where $\sum_{i=1}^{SN} fit_i$ is the desired fitness, while SN is denoted by the group size of the bees.

The onlooker bees will find the solution by using equation (5). Thus, the best solution (desired fitness) is generated until the number of trial is equal to the limit. Consequently, if the solution from the onlooker bees cannot be further improved, the onlooker bees will change to scout bees. As a result, the scout bees will abandon the search space (Singh & Alok, 2009). Note that, if the algorithm finds a solution with desired fitness, the solution will exit the algorithm easily and print the best solution. The correct assignments will be stored into Hopfield's brain as CAM.

Binary Artificial Bee Colony (ABC) Pseudocode

Step 1

Initialise parameters, size N. employed bees group size SN (Number of clauses), maximum allowed generations $g_{\max}$ (10 Generations) and trial number limit. Initialise all bees X .

Step 2

for $g = 1$ *to* $g_{\max}$ (Number of Generation)
Calculate the fitness for each bee X (one group of bee) and then evaluate them (we take the best two bee groups as x_i and x_k).

Step 3

{Employed bees phase}
for i=1 to SN (Group size)
Produce a new food source v_i using equation [5]
Check the fitness of v_i.
If v_i beats x_i,
then replace x_i with v_i in next generation and $trial_i = 0$
else $trial_i = trial_i + 1$
end for
Calculate the probability values p_i by using the equation [6] below.

Step 4

{Onlooker bees phase}
$t = 0, i = 1$
while $t < SN$
if $random < p_i$,
then produce a new food source v_i by using the following equation [5].
If v_i beats x_i,
then replace x_i with v_i in next generation and $trial_i = 0$,
else $trial_i = trial_i + 1$ $BL_{success,i} = BL$
end if
$t = t + 1$ (until $t = 100$)
end while

Step 5

{Scout bees phase}
if $\max(trial_i > \text{limit})$ then
Reset x_i within the search space
end if
Record the best solution founded so far
end for
Output the final solution

NEURO-LOGIC IN HOPFIELD NEURAL NETWORK

Hopfield Model

The Hopfield model was popularised by John Hopfield in 1982 (Hopfield, & Tank, 1985), which was used to solve vast pattern and combinatorial optimisation problems. The model comprised of interconnected units called neurons to form a network (Sathasivam, 2010). Thus, it is easy to hybridise the Hopfield network with other algorithms. The units in Hopfield nets are called the binary threshold unit (Haykin, 1992), which can only take binary values such as 1 and -1. The promising definitions for the activation of unit I, a_i , are:

$$a_i = \begin{cases} 1 \quad if \sum_j W_{ij} S_j > \xi_i \\ -1 \; Otherwise \end{cases} \qquad [7]$$

where W_{ij} is the connection strength from unit j to i. S_j is the state of unit j and ξ_i is the threshold of unit i. Normally, the connection in the Hopfield net has no connection with itself, but $W_{ii} = 0$ and some of the connections are symmetric or bidirectional (Sathasivam et al., 2013). The network consists of N recognised neurons, each is described by an Ising spin variable. The neurons are basically bipolar. Furthermore, $S_i \in \{1, -1\}$ follows the dynamics $S_i \to \text{sgn}(h_i)$, where the local field is h_i. The connection model can be generalised to embrace higher order connection. This modifies the field to:

$$h_i = \sum_j W_{ij}^{(2)} S_j + J_i^{(1)} \qquad [8]$$

The weight or the connection strength in the Hopfield network is always symmetrical. The updated rule is maintained as:

$$S_i(t+1) = \text{sgn}\left[h_i(t)\right] \qquad [9]$$

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

$$E = ... - \frac{1}{2} \sum_i \sum_j W_{ij}^{(2)} S_i S_j - \sum_i W_i^{(1)} S_j \qquad [10]$$

This energy function is significant because it establishes the degree of convergence of the network (Ioneschu et al., 2010). Thus, energy value is vital in order to reach the global solutions.

Content Addressable Memory (CAM)

The Hopfield model is a standard model for CAM (Holland, 1975). It is one of the remarkable features of Hopfield, which was inspired by the way the biological brain works. In layman's term, CAM can be defined as common memories where the retrieved data, from a given

address of the memory location, are stored (Ioneschu et al., 2010). In this paper, the satisfied randomised 2SAT assignment was stored in the CAM and retrieved after the training process. Basically, the CAM was used to the implementation of search algorithms such as ES and ABC.

Logic Programming in Hopfield Network

Strictly speaking, logic programming can be treated as a problem in combinatorial optimisation and it can be carried out on a Hopfield network (Sathasivam et al., 2013). The most celebrated logic programming model was the Wan Abdullah's logic paradigm, which implemented the Hopfield network with Horn clauses (Wan Abdullah, 1993). Therefore, Pinkas and Dechter (1995) emphasised on a bi-directional mapping between the prepositional logic formula and the Hopfield network by integrating the energy minimisation scheme to attain global convergence. Hence, the logic programming was implemented in Hopfield for 2SAT clause with some local search algorithms, namely, ES and ABC.

Implementation of 2SATABC in Hopfield Neural Network (HNN-2SATABC)

i. Translate all the random 2SAT clauses into Boolean algebra.
ii. Recognise a neuron to each ground neuron. Initialise all connection strengths to zero.
iii. Derive a cost function that is related with negation of all 2SAT 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 is represented conjunction and addition represents disjunction of clauses.

iv. The cost function is compared with energy, E in order to obtain the values of the connection strengths or weights (Sathasivam & Wan Abdulllah, 2008).
v. Check 2SAT clauses satisfaction by using ABC algorithm. Satisfied assignments will be stored in CAM.
vi. 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, it is considered as the final state.
vii. Find the corresponding final energy, E , of the final state by using lypunov equation. Verify whether the final energy obtained is global minimum energy or local minima. Calculate the corresponding hamming distance. The time taken to complete the process is recorded. The fitness value is computed by using the Kauffman model (Imada, & Araki, 1997):

$$f = \frac{1}{t_0 p}\sum_{t=1}^{t_0}\sum_{v=1}^{p} m^v(t) \quad \text{whereby, } m^v(t) = \frac{1}{N}\sum_{i=1}^{N} \xi_i^v S_i^v(t) \qquad [11]$$

THEORY IMPLEMENTATION

The simulations were performed on Microsoft Visual Studio 2013 for Windows. First of all, random 2SAT clauses were generated. After that, the initial states for the neurons were initialised in the 2SAT clauses. The network evolved until it reached the final state. Once the programme had reached the final state, the neuron state was updated via equation (8). As soon as the network relaxed to an equilibrium state, the final state obtained for the relaxed neuron was tested to determine if it was in a stable state. Additionally, if the state had remained unchanged for five runs, a stable state would have been considered. According to Pinkas and Dechter (1995), permitting an ANN to evolve will eventually lead to a stable state where the energy function obtained would not change further. Subsequently, the corresponding final energy for the stable state was computed. If the difference between the final energy and the global minimum energy had been within the tolerance value, then the solution would be considered as a global solution. Both algorithms were repeated 100 times with 100 neuron combinations. The tolerance value for the final energy was 0.001. In fact, according to Sathasivam et al. (2013), 0.001 was selected because it gave a better performance than other values did, besides successfully dodging statistical errors. Moreover, global minima ratio, Hamming distance, fitness landscape value and CPU time obtained from both ES (HNN-2SATES) and ABC (HNN-2SATABC) had also been compared.

RESULTS AND DISCUSSION

Global Minima Ratio and Hamming Distance

In this study, global minima ratio is defined as the ratio between the global solutions over the number of total runs. Meanwhile, Hamming distance is demarcated as the closeness of bits between the training state and the global state (retrieved state) of the neurons upon relaxation process (Sathasivam, 2010).

Table 1
Global minima ratio and hamming distance for HNN- 2SATES and HNN-2SATABC

Number of Neurons (NN)	Global Minima Ratio		Hamming Distance	
	HNN-2SATES	HNN-2SATABC	HNN-2SATES	HNN-2SATABC
10	0.9972	0.9985	0.00887	0.002980
20	0.9915	0.9922	0.01899	0.017852
30	0.9859	0.9897	0.02902	0.028978
40	0.9634	0.9742	0.03876	0.029685
50	0.9537	0.9713	0.04780	0.030669
60	0.9411	0.9699	0.06580	0.037976
70	-	0.9672	-	0.049651
80	-	0.9615	-	0.086186
90	-	0.9598	-	0.086991
100	-	0.9534	-	0.099981

Table 1 delineates the obvious variation in the global minima ratio and the Hamming distance obtained from HNN-2SATABC and HNN-2SATES. Thus, one can observe clearly from Table 1 that the global minima ratios obtained for HNN-2SATABC are nearly 1 from NN=10 until NN=100 compared to those of HNN-2SATES. Moreover, when the number of global minima ratio approached 1, the network produced more global solutions. Hence, the solutions were obtained by implementing HNN-2SATABC, which always converged towards global minima even though the complexity increased. The food source (solution) found by employed bee can be improved further via onlooker bees. When the search space reached local maxima for randomised 2-SAT fitness, scout bees would reset the search space. Through this point of view, the network always effectively finds optimum solutions. Nevertheless, the problem with HNN-2SATES has been the nature of ES that deploys tedious training process in searching the correct neuron states. Hence, the updated rule for HNN-2SATES generated more abrupt energy surface for the program to obtain more local minima instead of global minima. In addition, the neurons have to jump enormous energy barrier to reach the global solutions.

On top of that, the HNN-2SATABC consistently performed better than HNN-2SATES in terms of Hamming distance. When the value of Hamming distance is close to zero, the distance between the stable states and the global states will be almost zero. Thus, the accuracy of the output will be almost 100%. In this case, the solutions were nearly optimal and stable. Besides, Table 1 depicts that HNN-2SATABC outperformed HNN-2SATES in the Hamming distance standpoint. Furthermore, HNN-2SATABC was able to recall the correct states that contributed to the lower hamming distance compared to HNN-2SATES. Contrariwise, the ES algorithm emphasised the trial-and-error process during the clause satisfaction process.

Furthermore, the proposed paradigm, HNN-2SATES, was able to sustain up to 100 neurons. Hence, the ability to sustain a huge number of neurons was due to the special ability of ABC algorithms that reduced the computation burden in hunting the correct states. Thus, this enhanced the capability and the ability to control the energy relaxation process although the network got larger. Besides, providing more relaxation time for the network helped the network to retrieve the state more effectively. Thus, less relaxation time generated spurious minima, which caused the retrieved solution to achieve local minima energy (Sathasivam, & Sagir, 2014). The spurious minima, however, had been discovered in HNN-2SATES because it could only sustain up to 60 neurons.

Fitness Landscape Value

Figure 1 portrays the differences detected in the fitness landscape values obtained for HNN-2SATABC and HNN-2SATES. As observed, the differences in fitness value is zero. Hence, one can conclude that HNN-2SATABC is highly unlikely to get trapped in the local minima and always hunts for global solutions. The smoothness in the figure illustrates the consistency of the landscape fitness values, supported by the Hamming distance values discussed earlier on, which is also zero. The ruggedness of the fitness landscape for HNN-2SATES indicates the existence of error in obtaining global solutions.

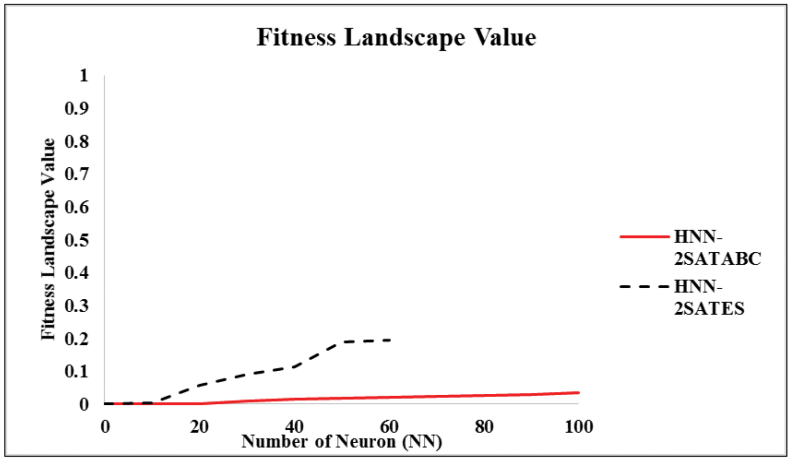

Figure 1. Fitness landscape value for HNN- 2SATES and HNN-2SATABC

Computation Time

The CPU time can be delineated as the time taken for the network to generate global solutions, including the training process. In this study, the analysis was limited up to 100 neurons.

Table 2
Computation time for HNN- 2SATES and HNN-2SATABC

Number of Neurons	CPU Time (in seconds)	
	HNN-2SATES	HNN-2SATABC
10	4	4.28
20	72	14.37
30	208	31.48
40	759	56.77
50	8036	90.16
60	75472	131.6
70	-	194.2
80	-	238
90	-	305.8
100	-	386

Table 2 depicts the CPU time for the proposed paradigm, HNN-2SATABC, in comparison to the conventional technique, HNN-2SATES. In terms of CPU time, the ES consumed has a rather relatively high CPU time (running time) compared to the ABC algorithm. Technically, the training process via ES frequently devoured more training time due to the trial-and-error procedure in achieving satisfied solutions. On the contrary, when the ABC algorithms were employed, the CPU time was faster due to the interaction between the onlooker and employed bees in hunting satisfied 2SAT assignments systematically. Additionally, this occurred as the

HNN-2SATABC incurred less computation burden during the training process, as compared to that of HNN-2SATES. For instance, the complexity of the network increased as the network began to grow massive. One could also note that the computational time increased when the number of neurons got higher. This trend had been consistent for HNN-2SATABC, even though the complexity of the network increased from NN=10 to NN=100, except for HNN-2SATES, which was only capable to sustain up to 60 neurons. This was because when the network became larger and more complex, the network devoured more computation time (Rojas, 1999). As a consequence of these arguments, extra time was needed to relax the global solution as the number of neurons increased.

CONCLUSION

Inspired by foraging intelligence behaviour displayed by honey bee swarm and engaging concept in artificial intelligence, a hybrid paradigm has been proposed; ABC algorithm incorporated with Hopfield neural network (HNN-2SATABC), in performing random 2SAT logic programming. Later, the proposed model was compared with a conventional technique; ES with the Hopfield neural network (HNN-2SATES). The work, as reported in this paper, reveals the tremendous differences in the performance of both the paradigms in terms of global minima ratio, Hamming distance, fitness landscape value and CPU time. Moreover, based on the experimental results, the proposed paradigm offers us a global minima ratio of approximately 1, faster computation time, as well as Hamming distance and fitness landscape values of approximately 0 compared to HNN-2SATES. Hence, the HNN-2SATABC has been unequivocally established to be more robust than the HNN-2SATES in certain aspects, which include better global minima ratio, lower Hamming distance, consistent fitness landscape value and faster CPU time, in performing random 2SAT logic programming. For future work, HNN-2SATABC can be extended to other satisfiability problems such as MAX-SAT, MIN-SAT and other SAT problems. This work can be utilised to solve traditional optimisation method such as travelling salesman and n-queen's problem.

REFERENCES

Abdullah, W. A. T. W. (1993). The logic of neural networks. *Physics Letters A, 176*(3-4), 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.

Ata, B., & Coban, R. (2015). Artificial Bee Colony Algorithm Based Linear Quadratic Optimal Controller Design for a Nonlinear Inverted Pendulum. *International Journal of Intelligent Systems and Applications in Engineering*, *3*(1), 1-6.

Brueggemann, T., & Kern, W. (2004). An improved deterministic local search algorithm for 3-SAT. *Theoretical Computer Science, 329*(1-3), 303-313.

Civicioglu, P., & Besdok, E. (2013). A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artificial Intelligence Review*, *39*(4), 315-346.

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

Gagneur, J., & Klamt, S. (2004). Computation of elementary modes: a unifying framework and the new binary approach. *BMC Bioinformatics*, *5*(1), 175-195.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms*, (pp. 69-93). California: Morgan Kaufmann Publishers.

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), 383-387.

Haykin, S. (1999). *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.), In *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.

Imada, A., & Araki, K. (1997, April). Application of an evolution strategy to the Hopfield model of associative memory. In *IEEE International Conference on Evolutionary Computation, 1997* (pp. 679-683). IEEE.

Ionescu, L. M., Mazare, A. G., & Serban, G. (2010). VLSI Implementation of an associative addressable memory based on Hopfield network model. *IEEE Semiconductor Conference*, *2*, 499-502.

Jia, D., Duan, X., & Khan, M. K. (2014). Binary Artificial Bee Colony optimization using bitwise operation. *Computers & Industrial Engineering*, *76*, 360-365.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization* (Vol. 200). Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459-471.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, *8*(1), 687-697.

Kashan, M. H., Nahavandi, N., & Kashan, A. H. (2012). DisABC: a new artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, *12*(1), 342-352.

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.

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.

Muthiah, A., & Rajkumar, R. (2014). A Comparison of Artificial Bee Colony algorithm and Genetic Algorithm to Minimize the Makespan for Job Shop Scheduling. *Procedia Engineering*, *97*, 1745-1754.

Pan, Q. K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences, 181*(12), 2455-2468.

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.

Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, *9*(2), 625-631.