mathematics

Article

# Non-Systematic Weighted Satisfiability in Discrete Hopfield Neural Network Using Binary Artificial Bee Colony Optimization

Siti Syatirah Muhammad Sidik [1], Nur Ezlin Zamri [2], Mohd Shareduwan Mohd Kasihmuddin [1,*], Habibah A. Wahab [3], Yueling Guo [1] and Mohd. Asyraf Mansor [2]

1 School of Mathematical Sciences, Universiti Sains Malaysia, USM, Pulau Pinang 11800, Malaysia; syatirah@student.usm.my (S.S.M.S.); guoyueling@student.usm.my (Y.G.)
2 School of Distance Education, Universiti Sains Malaysia, USM, Pulau Pinang 11800, Malaysia; ezlinzamri@student.usm.my (N.E.Z.); asyrafman@usm.my (M.A.M.)
3 School of Pharmaceutical Sciences, Universiti Sains Malaysia, USM, Pulau Pinang 11800, Malaysia; habibahw@usm.my
* Correspondence: shareduwan@usm.my; Tel.: +60-4-6534769

**Abstract:** Recently, new variants of non-systematic satisfiability logic were proposed to govern Discrete Hopfield Neural Network. This new variant of satisfiability logical rule will provide flexibility and enhance the diversity of the neuron states in the Discrete Hopfield Neural Network. However, there is no systematic method to control and optimize the logical structure of non-systematic satisfiability. Additionally, the role of negative literals was neglected, reducing the expressivity of the information that the logical structure holds. This study proposed an additional optimization layer of Discrete Hopfield Neural Network called the logic phase that controls the distribution of negative literals in the logical structure. Hence, a new variant of non-systematic satisfiability named Weighted Random 2 Satisfiability was formulated. Thus, a proposed searching technique called the binary Artificial Bee Colony algorithm will ensure the correct distribution of the negative literals. It is worth mentioning that the binary Artificial Bee Colony has flexible and less free parameters where the modifications tackled on the objective function. Specifically, this study utilizes a binary Artificial Bee Colony algorithm by modifying the updating rule equation by using not and (NAND) logic gate operator. The performance of the binary Artificial Bee Colony will be compared with other variants of binary Artificial Bee Colony algorithms of different logic gate operators and conventional binary algorithms such as the Particle Swarm Optimization, Exhaustive Search, and Genetic Algorithm. The experimental results and comparison show that the proposed algorithm is compatible in finding the correct logical structure according to the initiate ratio of negative literal.

**Keywords:** weighted random 2 satisfiability; binary artificial bee colony; logic phase; discrete Hopfield neural network

check for updates

**Citation:** Muhammad Sidik, S.S.; Zamri, N.E.; Mohd Kasihmuddin, M.S.; Wahab, H.A.; Guo, Y.; Mansor, M.A. Non-Systematic Weighted Satisfiability in Discrete Hopfield Neural Network Using Binary Artificial Bee Colony Optimization. *Mathematics* **2022**, *10*, 1129. https://doi.org/10.3390/math10071129

Academic Editor: Petr Stodola

Received: 29 January 2022
Accepted: 7 March 2022
Published: 1 April 2022

## 1. Introduction

The Artificial Neural Network (ANN) has become the standard approach to solve a wide range of optimization problems due to the structural concepts of simulating the neurons' activity of the human brain. The solid feature of ANN is mimic ability on how the brain works with the hope that we can build a system or a model that can produce collective decision making. As a result, AI practitioners heavily rely on the features of ANN which results in a lack of in-depth understanding of the operations conducted by ANN models. According to [1], ANN is considered a non-interpretable model due to its "black-box" nature, which fails to determine the direction and magnitude of the neurons in the network. In addition, the regularization and training time of the ANN model is difficult to be set as a constant. Hence, Wan Abdullah [2] embedded the Satisfiability (SAT) logical structure in a class of feedback ANN model, the Hopfield Neural Network (HNN). The role of SAT in HNN is to represent the neurons as meaningful indicators (or meaningless); when using such auxiliary inputs, one can use it as the basis of reasoning to explain the predicted outputs. In layman's terms, SAT is the symbolic instruction to navigate the operations of HNN. The work capitalized the structure of SAT by proposing logical inference through the minimization of Horn Satisfiability (HornSAT) in HNN. The approach is capable to yield connection strengths (synaptic weight) between each neuron, resulting in an energy-minimizing dynamic network. However, HornSAT consists of redundant variables which deteriorate the practicality in representing real-life variables. Therefore, ref. [3] incorporates another variant of SAT in HNN called 2 Satisfiability (2SAT). 2SAT is a class of systematic SAT whereby the number of variables in each clause is restricted to only $k$ variables, in this case, $k = 2$. The performance of 2SAT in HNN can achieve 90% production of global minimum solutions, which indicates the credibility of 2SAT in representing the behavior of neurons in HNN.

Alternatively, ref. [4] counters the problem with the existing SAT structure by proposing Random 2 Satisfiability (RAN2SAT) in the same network. RAN2SAT is the non-systematic variant of SAT which considers a non-restrictive number of variables in each clause, whereby RAN2SAT consists of clauses with $k = 1, 2$. Such formulation is considered to promote logical variation, which is believed to widen the search space of global minimum solutions. The work can locate maximum production of global solutions which indicates RAN2SAT is successfully embedded in the operations of HNN. Despite acceptable results, little attention has been given to RAN2SAT that claimed has potential as optimal SAT in HNN. The repetitive final neuron state which results in overfitting solutions is still an issue to be solved. Recently, ref. [5] proposed a new novel of non-systematic SAT named Major 2 Satisfiability (MAJ2SAT). The user interface provided in MAJ2SAT controls the structure of the logic where 2SAT is a dominant clause. As a result, MAJ2SAT can increase the neuron variation as the ratio of 2SAT increases. Although a restrictive training environment, the synaptic weight attained leads to global minima solutions. However, overfitting occurs when the 2SAT ratio is increased, which explains the creation of a repeated pattern of neurons states that disrupted the variation process and may reduce the precision of HNN. Thus far, there has been no investigation to seek a new novel of SAT that can provide non-overfitting solutions in HNN.

A metaheuristic is an effective method for obtaining an approximately optimal solution for the problems. For decades, a variety of metaheuristics with different classes was proposed and improved. The work by Zamani [6] is worth exploring where it proposed a new metaheuristic inspired by migratory birds' behavior during their long-distance aerial migrations named QANA. The features of QANA are long-term and short-term memories, a V-echelon communication topology, and quantum-based navigation with two mutation strategies and a qubit-crossover operator. These features enhance the population diversity and avoid premature convergence. The effectiveness of QANA was evaluated using benchmark functions and four engineering functions. QANA can outperform state-of-the-art algorithms. However, the work shows the significant findings of QANA in solving single and continuous problems. Another interesting work that is worth exploring is by [7] improved Moth Flame Optimization (MFO) by introducing a migration strategy to enhance the exploration and maintain diversity. The performance of M-MFO was evaluated by conducting benchmark functions and compared with seven variants of MFO and eight state-of-the-art algorithms. It is worth mentioning that M-MFO managed to maintain while the size of the problem increased. Unfortunately, the work did not investigate the performance of M-MFO in the lowest population diversity. In 2019, Kasihmuddin [8] implemented Estimation of Distribution in the DHNN to predict the possible final neuron states with the lowest minimum energy which leads to a global minimum solution. EDA is able to provide superior performance in all the metrics. Despite successful results, this work did not analyze the proposed model with different logical rules. In another development, Sathasivam [9] was the first person to incorporate non-systematic SAT with metaheuristic. The work implementing Election Algorithm (EA) as a training algorithm in DHNN to optimize the structure of RAN2SAT. This work demonstrated EA and RAN2SAT's capability to promote more diverse interpretations that lead to global minima solutions. Additionally, the EA can outperform another state-of-the-art algorithm due to its effective partitioning of solution space that reduces the complexity. However, the work did not analyze the impact of the retrieval phase. Note that metaheuristics allow us to effectively obtain approximate solutions to problems. However, extensive investigation is required to determine suitable metaheuristics for the specific problem.

Various social creatures exist in nature, such as ants, bees, and fish, where individuals from a given population cannot survive without the other members of that population. However, when we analyze the entire population, we see that the individual is extraordinarily intelligent through cooperation. Consequently, researchers have developed a type of algorithm to solve optimization problems that are inspired by these natures, named swarm intelligence algorithm (SI). One of the most widely used SI algorithms in the studies is *ABC*, which simulates the foraging behavior of honeybees, initially proposed to optimize continuous optimization problems [10]. The work was extended by [11] that developed binary variants of *ABC* named bitABC. The model uses the same framework as the original *ABC* but uses bitwise operations to optimize the food source. A comprehensive comparison has been made with another variant binary *ABC* with 13 selected benchmarks problem. The performance of BitABC is crucial in terms of final solution accuracy, convergence speed, and robustness. Subsequently, Kasihmuddin [3] proposed *ABC* as a training algorithm in HNN with 2SAT. This work demonstrates that *ABC* is more competitive in HNN compared to the standard method in obtaining satisfactiry interpretation of 2SAT. However, the work lacks variation and diversification in the quality of the retrieval final neuron states. In another development, *ABC* was incorporated with Double Layer Neural Network (DLNN) to solve the Bi-Level Programming Problem (BLPP) [12]. *ABC* was improved by cooperating with two major components of the Genetic Algorithm (GA) which are crossover and mutation. The improved-*ABC* is able to yield a high quality of global optimal solution with higher accuracy in a smaller time. The work proves that *ABC* is flexible to co-operate with another metaheuristic. Currently, ref. [13] proposed modification of solution update rule of basic *ABC* replaced with a xor logic gate in binABC. The comprehensive comparison was conducted with six variants of binary *ABC* by solving a modern benchmark problem (CEC2015). binABC is compatible in solving discrete problems. This finding shows that the update rule equation plays an important role in binary *ABC*. However, there is no investigation on the performance of different logic gate operators in the update rule equation. Although many researchers show the stellar performance of binary *ABC* in solving a variety of problems, there has been much uncertainty of which basic logic gate operators work well in solving discrete optimization problems. Table 1 below summarizes the related research.

The important contributions of this study are listed as follows:

(i) This study introduced an additional layer of DHNN to generate a non-systematic logical structure with a consideration ratio of negative literals. The higher number of negative literals in the logical structure is considered to enhance diversified final neuron states retrieved.

(ii) This study explores the capability of the Artificial Bee Colony algorithm (*ABC*) with different logic gate operators in the update rule equation. The main purpose of *ABC* in this study is to find the correct logical structure according to a predetermined ratio of negative literal. A comprehensive comparison was conducted to reveal the effectiveness of *ABC* with the NAND operator. Next, the performance of *ABC* will be compared with three state-of-the-art metaheuristics. Statistical analysis was conducted to show that *ABC* is compatible in finding the correct logical structure.

The major contributions of the present study are as follows:

(i) A new novel of non-systematic Weighted Random $k$ Satisfiability ($rk$SAT) is proposed by combining the first and second-order Satisfiability logical rule with consideration of negative literals.

(ii) The implementation of Artificial Bee Colony in the logic phase to control the distribution of negative literals in the logical structure.
(iii) A comprehensive comparative analysis of five different logic gate operators in the food source equation in terms of producing a correct number of negative literals for a higher ratio.
(iv) The proposed Artificial Bee Colony in the logic phase will be compared with the state-of-the-art binary algorithm.

**Table 1.** Summary of the Related Studies.

| Author(s) | Detail of the Studies | Summary and Findings |
|---|---|---|
| Wan Abdullah [2] | The first work incorporates SAT with HNN. | The work capitalized the structure of SAT by proposing logical inference through the minimization of HornSAT in HNN. The approach is able to yield connection strengths (synaptic weight) between each neuron, resulting in an energy-minimizing dynamic network. |
| Kasihmuddin et al. [3] | HNN incorporates with *ABC* algorithm in minimizing the 2SAT structure. | The performance of 2SAT in HNN is able to achieve 90% production of global minimum solutions which indicates the credibility of 2SAT in representing the behavior of neurons in HNN. |
| Sathasivam et al. [4] | A non-systematic RAN2SAT has been developed to represent the symbolic output in HNN. | The work is able to locate maximum production of global solutions which indicates RAN2SAT is successfully embedded in the operations of HNN. |
| Alway et al. [5] | A new novel of non-systematic SAT was proposed and implemented in HNN. | MAJ2SAT can increase the neuron variation as the ratio of 2SAT increases. Although a restrictive training environment, the synaptic weight attained leads to global minima solutions. |
| Karaboga & Basturk [10] | *ABC* in solving multidimensional and multimodal numerical optimization problems. | The new SI algorithm named *ABC* was proposed with a few parameters and compared with existing algorithms. |
| Jia et al. [11] | Developed binary variants of *ABC* named bitABC. | A comprehensive comparison has been made with another variant binary *ABC* with 13 selected benchmarks problem. The performance of BitABC is crucial in terms of final solution accuracy, convergence speed, and robustness. |
| Watada et al. [12] | *ABC* was improved by cooperating with two major components of the Genetic Algorithm (GA) and implemented in DLNN to solve BLPP. | The improved-*ABC* is able to yield a high quality of global optimal solution with higher accuracy in a smaller time. The work proves that *ABC* is flexible to co-operate with another metaheuristic. |
| Kiran [13] | Proposed modification of solution update rule of basic *ABC* replaced with a xor logic gate in binABC. | The comprehensive comparison was conducted with six variants of binary *ABC* by solving a modern benchmark problem (CEC2015). binABC is compatible in solving discrete problems. |

Figure 1 illustrates the general flow of the proposed study to ensure the readers gain a better understanding of this approach. In the first place, the initialization of the logic phase occurs after the ratio of negative literal and number of literal is predetermined. *ABC* will be utilized in the logic phase as the search technique in order to generate the correct structure of Weighted Random 2 Satisfiability ($\Gamma_{r2SAT}$). Note that the objective of the logic phase is to distribute a correct number of negative literal in the $\Gamma_{r2SAT}$. Then, the first phase of DHNN will optimize $\Gamma_{r2SAT}$ to ensure the correct synaptic weight is obtained. The computation of the local field of the neuron state and the final energy happens in the testing phase. The quality of the output will be evaluated by comparing the final energy. Supposing the output achieved lies within the tolerance value, the final neuron states will achieve global minima solution, otherwise be trapped in local minima solution.

The outline of the current study includes Section 2, provides the preliminary explanation on the Weighted Random *k* Satisfiability. Section 3 briefly explains Weighted Random 2 Satisfiability study in DHNN. Then, the proposed binary Artificial Bee Colony and other binary algorithms are discussed in Section 4. The methods and experimental setup are given in Section 5. The simulation of the study is discussed in Section 6. Finally, concluding remarks are given in the last section.

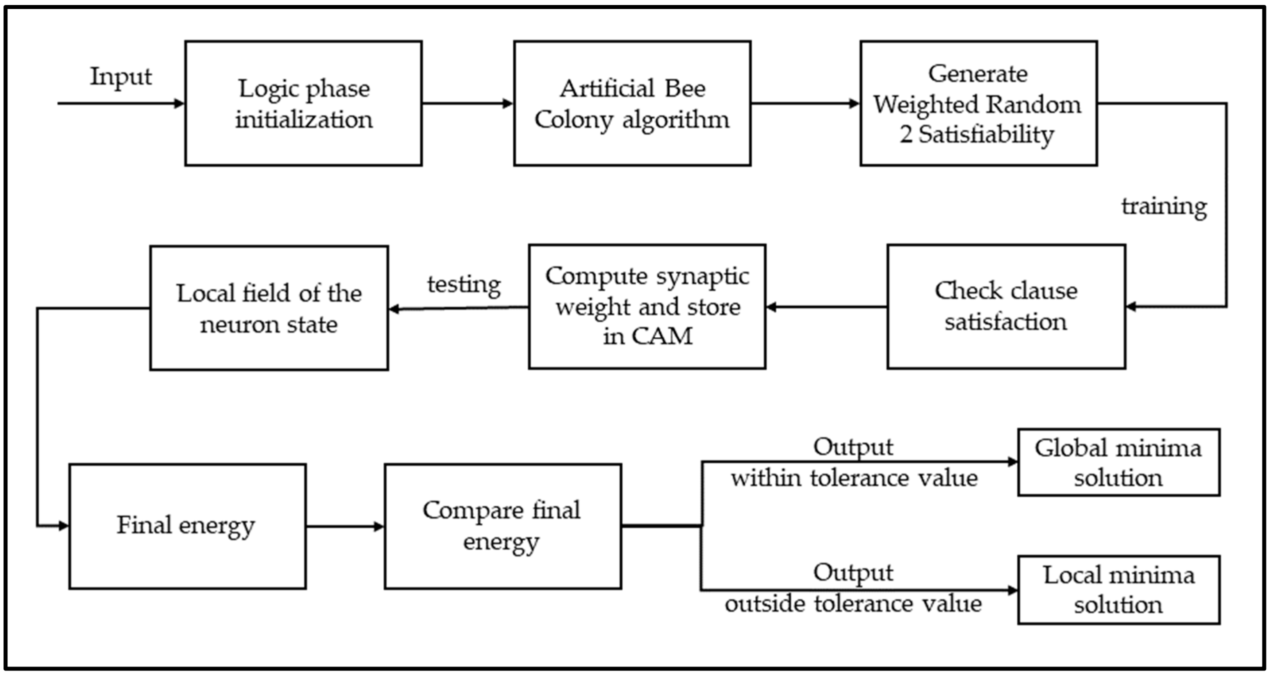

Input
Logic phase initialization
Artificial Bee Colony algorithm
Generate Weighted Random 2 Satisfiability
training
Check clause satisfaction
Compute synaptic weight and store in CAM
testing
Local field of the neuron state
Final energy
Compare final energy
Output within tolerance value
Global minima solution
Output outside tolerance value
Local minima solution

**Figure 1.** General Flow of the Proposed Study.

## 2. Weighted Random *k* Satisfiability

Satisfiability (SAT) is a Boolean formula that is said to be satisfiable if satisfied assignments of the literals exist, which makes the formula true. SAT is categorized as NP-complete [14] which inspires logic mining practitioners to utilize SAT as a form of symbolic instruction in representing the entries of a real dataset. SAT can be divided into two main structural classes which are systematic and non-systematic logical rule [15]. According to the findings in [4], non-systematic SAT provides more logical variation in terms of final neuron states. In this article, Weighted Random *k* Satisfiability is introduced as a new class of non-systematic SAT with consideration of the desired number of negative literals. The number of negative literals in the SAT can be predetermined with a randomized choice of literals that can be either positive or negative. The general equation for Weighted *k* Satisfiability can be formulated as follows:

$$\Gamma_{r2\mathrm{SAT}} = \wedge_{i=1}^{u} J_i^{(2)} \wedge_{i=1}^{v} J_i^{(1)} \tag{1}$$

whereby the value of 2 (second) and 1 (first) depicts the order of clauses $(k)$ in $\Gamma_{r2\mathrm{SAT}}$ where $k \in \{2, 1\}$. Note that $u$ and $v$ denote the total number of second- and first-order clauses, respectively. The total combination of clauses in $\Gamma_{r2\mathrm{SAT}}$ can also be defined as $m$ where $m = u + v$. Following Equation (1), the combination of orders in the formulation makes $\Gamma_{r2\mathrm{SAT}}$ a class of non-systematic SAT whereby the number of literals in each clause is not restricted to only one value of $k$. Note that $J_i^{(2)} \in \{(B_i \vee C_i), (B_i \vee \neg C_i), (\neg B_i \vee C_i), (\neg B_i \vee \neg C_i)\}$ and $J_i^{(1)} \in \{A_i, \neg A_i\}$ are possible clause for $\Gamma_{r2\mathrm{SAT}}$ formulation. In other words, $\Gamma_{r2\mathrm{SAT}}$ has non-redundant literals, whereby $i \neq i+1$ for any orders of $k$. By considering both order clauses, the total number of literal exists in $\Gamma_{r2\mathrm{SAT}}$ can be calculated by using Equation (2) as below:

$$\lambda = 2u + v \tag{2}$$

According to the study by [8], the assignments that correspond to $\Gamma_{r2\mathrm{SAT}}$ can be presented as $\{\mathrm{TRUE}, \mathrm{FALSE}\}^{\lambda} = \{1, -1\}^{\lambda}$. The main goal of the $\Gamma_{r2\mathrm{SAT}}$ is to find a set of assignments that make the whole formulation become TRUE. In comparison with the previous structure such as Karim et al. [15] and Alway et al. [5], the main structural comparison of $\Gamma_{r2\mathrm{SAT}}$ is the feature of controlling the distribution of negative (negation) literals in the logical structure. Thus, a systematic mechanism is introduced to only allow a specific total number of negative literals in $\Gamma_{r2\mathrm{SAT}}$. $N_v$ is defined as the desired number of negative literals in $\Gamma_{r2\mathrm{SAT}}$ which can be calculated by using Equation (3) below:

$$N_v = r\lambda \tag{3}$$

$r$ is the ratio of negative literals existing in $\Gamma_{r2\mathrm{SAT}}$ with a range of $r = [0.5, 0.9]$. The pre-determined value of $r$ depicts the percentage (%) of negative literals, whereby when $r = 0.5$, the percentage of negative literals in $\Gamma_{r2\mathrm{SAT}}$ is 50% out of all $\lambda$. The proposed step size in this article is $\Delta r = 0.1$. Conjointly, the value of $N_v$ can also is defined as $N_v = \lfloor r\lambda \rfloor$, as the number of negative literals that should be considered must be less than or equal to a given number in the form of $N_v \in \mathbb{N}$. For example, for $(\lambda, r) = (6, 0.5)$, the value of $N_v$ is given by $N_v = \lfloor (0.3)(6) \rfloor = \lfloor 1.8 \rfloor \approx 1$. Thus, using the value of $N_v$, one possible negative literal of $\Gamma_{r2\mathrm{SAT}}$ is given by $\Gamma_{r2\mathrm{SAT}} = (B_1 \vee C_1) \wedge (B_2 \vee \neg C_2) \wedge A_1 \wedge A_2$. This indicates that a specific method is required to filter the type of logical rule that satisfies the condition in Equation (3).

Unfortunately, without an appropriate mechanism, the distribution of negative literal in $\Gamma_{r2\mathrm{SAT}}$ will potentially be skewed to one side or biased to a specific type of clause. Therefore, in this article, an additional optimization layer of controlling the distribution of the negative literal in $\Gamma_{r2\mathrm{SAT}}$ is presented. The value of $N_v$ is set as the objective function with randomized literals selection which will be determined as negative or positive. In this context, a minimization task is introduced before generating the right structure of $\Gamma_{r2\mathrm{SAT}}$ which will be formulated based on Equation (4).

$$f_L = min[\kappa - N_v] \tag{4}$$

whereby the optimal value of the best fitness, $f_L$ is equal to 0. Note that $\kappa$ is the total number of weights in $\Gamma_{r2\mathrm{SAT}}$ which can be formulated in Equation (5) as follows:

$$\kappa = \sum_{i=1}^{\lambda} \eta_i \tag{5}$$

As $\kappa$ is the number of negative literals which should be achieved in order to satisfy Equation (6), $\kappa$ always accounts for a non-zero and non-negative value. Equation (6) presents the possible value of the weight, $\eta_i$:

$$\eta_i = \begin{cases} 1, & \text{if } \neg Z_i \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $Z_i$ can be any arbitrary negative literals in $k$ order clauses. Thus, to guarantee only the desired number of negative literals is achieved, without any bias towards a specific order of clauses, an additional phase named the logic phase is added to the model. This effort is to ensure $\Gamma_{r2\mathrm{SAT}}$ is optimally generated with respect to $N_v$. The role of the logic phase is to effectively generate $\Gamma_{r2\mathrm{SAT}}$ with the dynamic value of $r$ and $\lambda$ by successfully minimizing Equation (4). In summary, the structural feature of $\Gamma_{r2\mathrm{SAT}}$ formula are presented as follows:

- The structure combines a different order of clauses, $J_i^{(k)}$ with logical AND ($\wedge$).
- The variable in the $\Gamma_{r2\mathrm{SAT}}$ considers non-redundant with randomized selection to be negative or positive.
- Consideration of the desired number of negative literals without any biased to a specific order of clauses.

Consecutively, the minimization task in the logic phase will be carried out using an optimization algorithm which will be explained in the next section.

## 3. Discrete Hopfield Neural Network

DHNN consists of interconnected neurons containing input and output layers but without hidden layers. Several features of DHNN include associative memory, fault tolerance, and energy minimization as the neuron state changes. Generally, the neurons in DHNN will be updated asynchronously until the final neuron state reaches the equilibrium state which corresponds to the solution of the optimization problem. Thus, the general formulation of updating rule of $i$-th neuron $S_i$ in DHNN is given in Equation (7) as follows:

$$S_i = \begin{cases} 1, & \text{if } \sum_{j}^{n} W_{(i,j)} S_j \geq \rho_i \\ -1, & \text{otherwise} \end{cases} \tag{7}$$

where $W_{(i,j)}$ and $\rho_i$ denote the synaptic weight and threshold of the network. In order to ensure the energy of DHNN decreases monotonically, we set $\rho_i = 0$ which demonstrates effective neuron state classification. The $\Gamma_{r2\mathrm{SAT}}$ will be embedded by assigning each variable with neurons to the defined cost function, $E_{\Gamma_{r2SAT}}$. Equations (8) and (9) below represent the generalized cost function of $\Gamma_{r2\mathrm{SAT}}$:

$$E_{\Gamma_{r2SAT}} = \sum_{i=1}^{\lambda} \prod_{j=1}^{u+v} q_{ij} \tag{8}$$

$$q_{ij} = \begin{cases} \frac{1}{2}(1 - S_X), & \text{if } \neg X \\ \frac{1}{2}(1 + S_X), & \text{otherwise} \end{cases} \tag{9}$$

Note that the value of $E_{\Gamma_{r2SAT}}$ is proportional to the number of inconsistencies of the clauses [4]. In this context, $E_{\Gamma_{r2SAT}}$ will increase as the number of unsatisfied clauses increases. The lowest possible value that can be obtained for the cost function is $E_{\Gamma_{r2SAT}} = 0$. In order to ensure the minimization of $E_{\Gamma_{r2SAT}}$, the following local field is utilized by using Equation (10):

$$h_i(t) = \sum_{j=1,\, i=j}^{u+v} W_{(i,j)}^{(2)} S_j + W_{(i)}^{(1)} \tag{10}$$

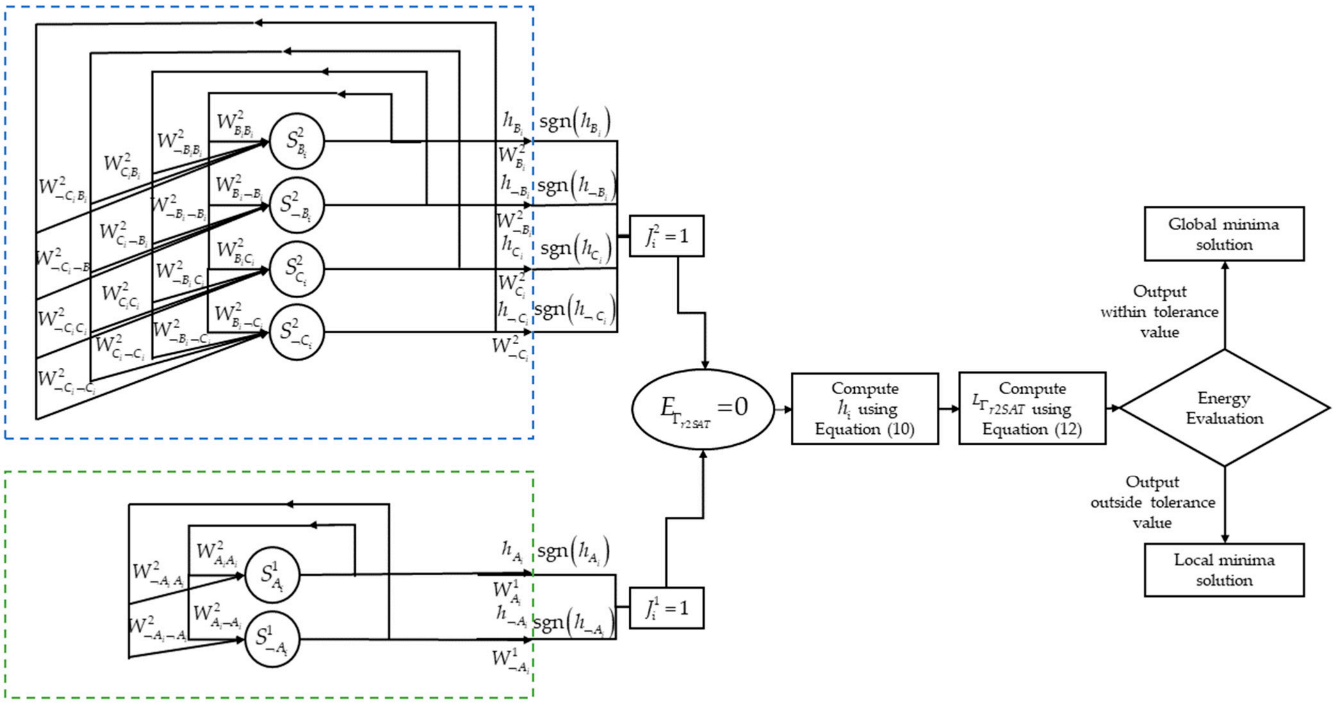
$S_i(t)$ is an updated neuron state given by Equation (11):

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

where $W_{(i,j)}^{(2)}$ and $W_{(i)}^{(1)}$ are second and first order synaptic weights, respectively. Hyperbolic Tangent Activation Function (HTAF) is utilized in Equation (11) to avoid the neuron state being updated linearly. In terms of neuron connection, synaptic weights can be obtained by using the Wan Abdullah method [2] which compares Equation (8) and Equation (12) as long as the proposed $\Gamma_{r2\mathrm{SAT}}$ has at least one satisfied interpretation. The Lyapunov energy function that corresponds to the $\Gamma_{r2\mathrm{SAT}}$ logic is given as in Equation (12):

$$L_{\Gamma_{r2SAT}} = -\frac{1}{2} \sum_{i=1, i\neq j}^{n} \sum_{j=1, j\neq i}^{n} W_{(i,j)}^{(2)} S_i S_j - \sum_{i=1}^{n} W_{(i)}^{(1)} S_i \tag{12}$$

The value of $L_{\Gamma_{r2SAT}}$ indicates the scalar quantity of the final neuron state which corresponds to the interpretation that models $\Gamma_{r2\mathrm{SAT}}$. Interestingly, Equation (12) informs the network whether the final neuron state reaches the absolute minimum energy that corresponds to the optimal final states for $\Gamma_{r2\mathrm{SAT}}$. Figure 2 shows the schematic diagram for DHNN-$r$2SAT. Note that the blue dotted lines represent the second-order clauses with four possibilities of neurons in the second-order clauses. Meanwhile, the green dotted lines represent first-order clauses with two possibilities of neurons.



$J_i^2 = 1$
$J_i^1 = 1$
$E_{\Gamma_{r2SAT}} = 0$
Compute $h_i$ using Equation (10)
Compute $L_{\Gamma_{r2SAT}}$ using Equation (12)
Energy Evaluation
Output within tolerance value
Global minima solution
Output outside tolerance value
Local minima solution

**Figure 2.** Schematic Diagram for DHNN-*r*2SAT.

## 4. Artificial Bee Colony

Bee colonies are like other insect communities which consist of distributed systems. Despite the simplicity of individuals, an insect colony has a highly structured social organization that allows bee colonies to undertake complicated tasks that would be impossible for a single bee to accomplish. Thus, recently, a new SI algorithm known as *ABC* has piqued the interest of researchers. *ABC* algorithm was inspired by the waggle dance and foraging behavior of honeybee colonies [10]. There are three main components in *ABC*: employed bees, onlooker bees, and scout bees. Each bee utilizes a herd mentality where its work and others' work are shared without any authority. As a result, bees have the ability to organize [16]. The employed bees will go out and search for a food source and memorize the location of the source. The onlooker waits in the hive to gain information from the employed bees by performing waggle dance in the dancing area of the hive. The onlooker bee acts as a decision-making process, deciding which food source is the most profitable [16]. Once the decision is made, the scout bee will be spawned to validate the new food source location whether is profitable. In this study, *ABC* is implemented in the logic phase to generate the desired number of negative literals in $\Gamma_{r2SAT}$. The profitable food source found by the scout bee indicates the correct number of negative literals.

### 4.1. Binary Artificial Bee Colony

Binary *ABC* has been implemented by several researchers ([11,17–19]) for binary optimization tasks. In the logic phase, the role of *ABC* is to minimize Equation (4), which corresponds to the desired number of negative literals in $\Gamma_{r2SAT}$ or $N_v$. Note that the binary values are {0, 1} where "0" denotes the positive literal, and "1" depicts the negative literal for each literal in $\Gamma_{r2SAT}$. The phases involved in *ABC* are elaborated in the next subsections:

#### 4.1.1. Initialization Phase

Suppose that the search space of the *ABC* algorithm is the hive's surroundings containing food sources. The algorithm starts with randomly producing food sources that correspond to the solutions in the search space. The initial food source is generated at random within the parameters' bounds. The whole population consists of $SN$ bees and a $D$ number of optimization parameters. Equation (13) is the initialization of the random food source, $x_{i,j}$, whereby,

$$x_{i,j} = x_j^{\min} + \text{rand}(0, 1)\left(x_j^{\max} - x_j^{\min}\right) \tag{13}$$

where $i = 1, \ldots, SN$ denotes the bees. Meanwhile, $j = 1, \ldots, D$ depicts the generation of bees. $x_j^{\min}$ and $x_j^{\max}$ are lower and upper bound of $x_{i,j}$, respectively. After initialization, the fitness of the food source will be evaluated based on Equation (4). Then, the population of food sources is susceptible to the cycle of the employed bees', onlooker bees', and scout bees' search operations.

### 4.1.2. Employed Bees

An employed bee's job is to look for a new food source in the vicinity of the current food source in their memory. Equations (14) and (15) is the update rule equation in *ABC*:

$$v_{i,j} = x_{i,j}\overline{\wedge}\left[\varphi\left(x_{i,j}\,\overline{\wedge}\,x_{k,j}\right)\right] \tag{14}$$

$$\varphi = \begin{cases} 1 & \text{rand}(0,1) < 0.5 \\ -1, & \text{rand}(0,1) \geq 0.5 \end{cases} \tag{15}$$

where $v_{i,j}$ is the new food source and $x_{k,j}$ is the observed food source. Note that $\overline{\wedge}$ represents the NAND logic gate operator. The fitness of each new food source will be evaluated. If the source at $v_{i,j}$ is superior to $x_{i,j}$, the employed bees have forgotten about the previous food source and have memorized the new one. Otherwise, the location of the previous food source is memorized. If $x_{i,j}$ cannot be improved, the employed bee will undergo five trials to improvise the food source.

### 4.1.3. Onlooker Bees

Once the employed bee finishes the search process, they will go to the hive to share the information related to the location of the food source and nectar amounts by the dance area. The feature of multiple interactions occurred in this stage. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to the nectar amount. A roulette wheel selection method in which proportions of the wheel are assigned to the bees based on their fitness value is introduced. Equation (16) formulated the probability of the fitness:

$$p_i = \frac{f_i}{\sum_{i=1}^{SN} f_i} \tag{16}$$

The employed bee that has a higher fitness value will have more chances to be selected. After an employed bee is chosen for an onlooker bee where the positive feedback occurred, a new food source will be produced using Equations (14) and (15) until the maximum limit is achieved.

### 4.1.4. Scout Bees

If the food source generated by the onlooker bee does not achieve $f_L = 0$, the scout bee will undergo a new iteration until the maximum limit is achieved. The feature of the negative feedback occurred when the food source does not achieve $f_L = 0$ after five iterations.

### 4.2. Optimizing Logic Phase via Binary Artificial Bee Colony

The logic phase is a systematic method that controls the distribution of the negative literals in $\Gamma_{r2SAT}$ according to the desired weighted ratio $r$, by minimizing Equation (4). In this paper, the logic phase employs *ABC* to minimize the solution. According to [6], the parameter in *ABC* is flexible to adjust and simple. This is due to the parameters involved in the food source equation that can be easily changed according to the objective function. Additionally, the solution space can be improvised using the food source equation in two phases, which are employed and onlooker bees. In the logic phase, $x_{i,j}$ represent the possible structure of $\Gamma_{r2SAT}$. The food source represents the number of negative literals, $\eta_i$ where $\eta_i \in \{0, 1\}$. Initially, the first generation will be initialized by generating ten random $x_{i,j}$ and the fitness (food source) will be evaluated. Then, the bees will explore new food sources in employed bees by using Equations (14) and (15). The profitable food source will be evaluated in the onlooker bees' phase, improvising the solution space using Equations (14) and (15). The scout bee will examine each food source to find a food source that carries a correct number of negative literals. Figure 3 and Algorithm 1 below illustrate the whole optimization process of *ABC* in the logic phase.

**Algorithm 1.** The pseudocode of *ABC* in the logic phase:

```
Set the initial parameters, including population size $SN$, employed bees' size, onlooker bees' size, scout bees' size, maximum allowed generations $g_{\max}$ (10 generations), trial number, and initialize all bees $X$.
for $g = 1$ to $g_{\max}$ (number of generation)
Calculate the fitness for each bee $X$ (one group of bees) and evaluate them (take the best two bee groups as $x_{i,j}$ and $x_{k,j}$)
{Employed bees' phase}
  for $i = 1$ to $SN$
    Produce new food source $v_{i,j}$ using Equation (14)
    Evaluate the fitness of $v_{i,j}$
    if $v_{i,j} < x_{i,j}$
    then replace $x_{i,j}$ with $v_{i,j}$ in next-generation and $trial_i = 0$
    else $trial_i = trial_i + 0$
    end if
    end for
    Computed the probability $p_i$ by using Equation (16)
{Onlooker bees' phase}
  $t = 0, \ i = 1$
  while $t < SN$
    if $rand_i < p_i$
    then produce a new food source $v_{i,j}$ using Equation (14)
    Evaluate the fitness of $v_{i,j}$
      if $v_{i,j} < x_{i,j}$
      then replace $x_{i,j}$ with $v_{i,j}$ 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
{Scout bees' phase}
  for $l = 0, \ i = 1$
  if $f_L = 0$
  then record the best solution founded
  else $l = l + 1$ (until $l = 5$)
  end if
  end for
Output for the solution
```

### 4.3. Benchmark Algorithms in the Logic Phase

To assess the efficiency of the proposed *ABC* in the logic phase, comparative analysis will be conducted versus state-of-the-art algorithms; Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Exhaustive Search (ES). Each algorithm has different baseline methods suitable to be compared with the proposed model in terms of minimizing Equation (4). GA is an evolutionary algorithm that simulated the natural selection process [20]. According to [21], GA search from a population of points to find a solution based on the desired fitness. Notably, GA consists of balanced global and local search operators of crossover and mutation. By involving the chromosomes exchange process and flipping genes in each chromosome, the termination criteria of GA is when a string of chromosomes that represent the structure of $\Gamma_{r2SAT}$ successfully corresponds to the desired $N_v$. As for PSO, the possible solutions are represented as particles where all the particles will form a set of solutions called the swarm. Additionally, PSO is known as a class of swarm intelligence algorithms that has a similar mechanism to *ABC*. Based on the previous study by [22], PSO is best utilized to solve discrete vector problems where the global best solutions concept is used to search for desired solutions. The position and velocity values will be initiated for each particle. The initial population will be initialized with random positions and velocities at the outset. The fitness of each particle will be evaluated by using Equation (4). The process is repeated within the determined number of trials. Although there is no optimizer involved, ES is reported to be efficient when handling a smaller sized optimization task [23]. ES is the benchmark searching technique that has the ability to search the entire search space with the random mechanism. The selected algorithms were chosen based on their potential to solve binary optimization tasks. Thus, they are suitable for evaluating the ability of algorithms in terms of local search and convergence speed in minimizing Equation (4).

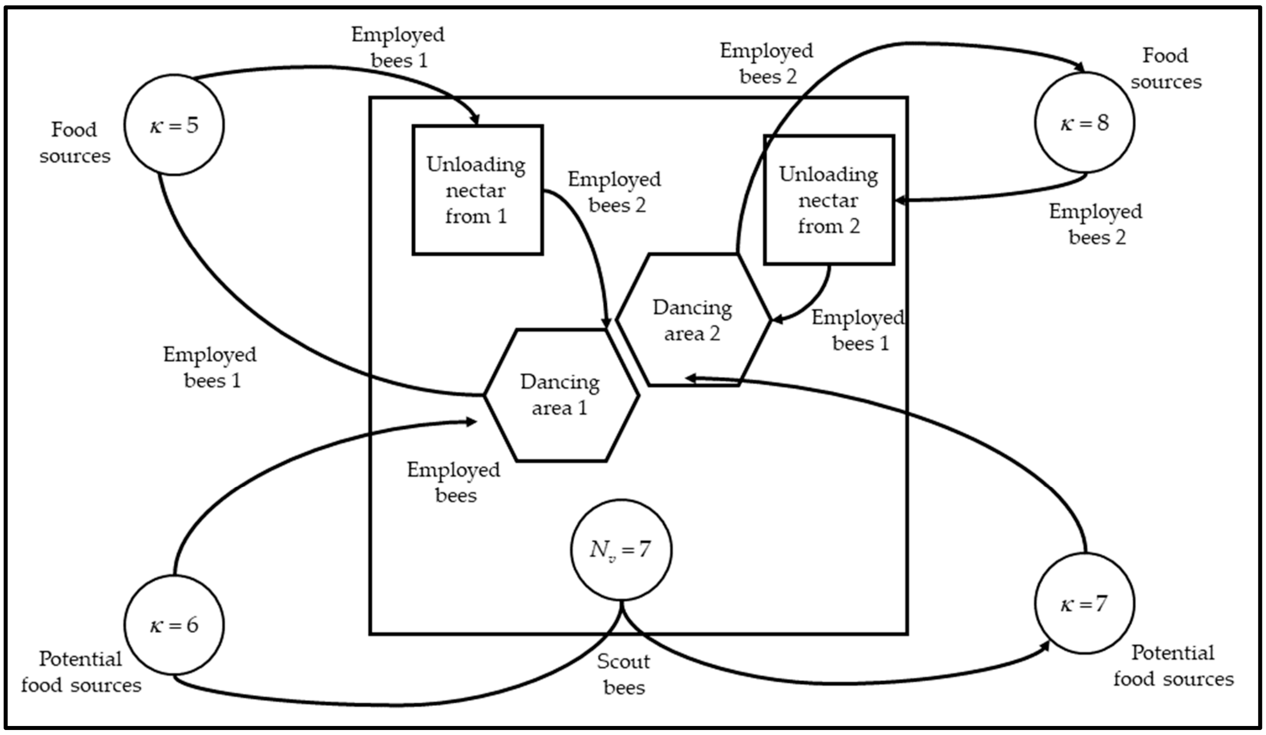


**Figure 3.** Schematic diagram of *ABC* optimizing logic phase.

## 5. Simulation Setup

In this section, we verify the effectiveness of the proposed model during the logic phase using different logic gate operators in Equation (14). This experiment only considers a single objective, which is to determine whether the proposed model generates the correct number of negative literals. To guarantee the reproducibility of the experiment, we set up our experiment as follows:

### *5.1. Simulation Design*

To avoid possible biases during experimentation, all the simulations will be conducted based on the following feature:

(i) Device Setting: In order to avoid any biases in evaluating the results, the proposed model was implemented and analyzed by using Dev C++ version 5.11. The simulation was run on the same device with a 2.70 GHz Intel Core i7 processor with a 64-bit Windows 10 Operating System where the threshold simulation time is set to 24 h.

(ii) SAT Configuration: The number of the second-order clauses is set twice from first-order clauses where $u = 2v$**.** For all models, each instance is associated with $\lambda \in \{10,\ 50,\ 90\}$.

### 5.2. Parameters Assignment

Tables 2–6 listed the parameters used in the logic phase for all the proposed DHNN models. The performance of all $r2SAT_{ABC}$ models is evaluated for different values of the weighted ratio, $r$, different number of neurons, $\lambda$, and constant neuron combination, $\beta$. The employed bee and the onlooker bee have a similar number of bees which indicates the population size used in the *ABC*. According to [24], a small number of scout bees (approximately 5%) with regard to the overall population can navigate all the bees to the optimal solution. Therefore, we set the maximum number of scout bees to five to ensure the proposed $r2SAT_{ABC}$ will not trap to possible local maxima solution. Similar to the work by Kasihmuddin et al. [3], the number of trials in employed and onlooker bees phases was set to five, because more trials are given to the scout bees to update their food source. On the other hand, GA will initialize a population of 500 that corresponds to $r2SAT_{GA}$ during the logic phase. The selection rate was set to 0.1 in order to enhance the searching process. RWS also utilized in the selection operator of GA. The rate of crossover and mutation are set to 1 in order to reduce the computation complexity. Note that each particle in PSO has a randomly selected position and velocity. The best particle size is set to 50 in order to obtain the best results for the majority of the PSO variants.

**Table 2.** List of Parameters used in DHNN model.

| Parameter | Parameter Value |
|---|---|
| Ratio of negative literals, $r$ | {0.5, 0.6, 0.7, 0.8, 0.9} |
| Number of neurons, $\lambda$ | {10, 50, 90} |
| Neuron combination, $\beta$ | 10 |
| Number of trials | 100 |
| Number of learning, $b$ | 100 |
| Learning iteration, $\varepsilon$ | $\varepsilon \geq b$ |
| CPU time | 24 h |
| Tolerance value, $Tol$ | 0.001 |
| Initialization of neuron states | Random |
| Training algorithm | Exhaustive Search [5] |

**Table 3.** List of *ABC* parameters used in Logic Phase.

| Parameter | Parameter Value |
|---|---|
| Neuron combination, $\beta$ | 10 |
| Number of employed bees | 50 [3] |
| Number of onlooker bee | 50 [3] |
| Number of the scout bee | 5 [23] |
| Number of trials | 5 [3] |

**Table 4.** List of GA parameters used in Logic Phase.

| Parameter | Parameter Value |
|---|---|
| Number of generations | 100 [25] |
| Number of populations | 500 |
| Selection rate | 0.1 [10] |
| Crossover rate | 1 [10] |
| Mutation rate | 1 [10] |

**Table 5.** List of ES parameters used in Logic Phase.

| Parameter | Parameter Value |
|---|---|
| Number of strings | 100 [10] |
| Selection rate | 0.1 [10] |

**Table 6.** List of PSO parameters used in Logic Phase.

| Parameter | Parameter Value |
|---|---|
| Particle size | 50 |
| Number of trials | 20 |
| Particle velocity | 10 |

### 5.3. Dataset

The experiment will utilize simulated data that will be generated randomly by the proposed model. The elements of the simulated data set are strings of bipolar values that are $\{-1, 1\}$ based on the structure of $\Gamma_{r2SAT}$. Existing research by Sathasivam et al. [4], Karim et al. [15], and Alway et al. [5] demonstrate that simulated data sets are commonly used in testing and evaluating the capabilities of a new proposed SAT in DHNN. As a result, the outcomes of the simulated data set will project and confirm the efficacy of the proposed model when applied to real-life data sets.

### 5.4. Error Analysis

The performance of all the models can be evaluated based on average mean absolute error (MAE) and best fitness, $f_{best}$. These metrics will measure the effectiveness of the model in generating the correct logical structure of $\Gamma_{r2SAT}$ according to initiate $r$. The value of $f_{best}$ and average MAE can be obtained according to Equations (17) and (18), respectively.

$$f_{best} = \sum_{i=1}^{n} \min\left|\kappa_i - (N_v)_i\right| \tag{17}$$

$$\text{MAE} = \sum_{j=1}^{\alpha} \frac{\frac{1}{n}\sum_{i=1}^{n}\left|f_{fit} - f_{best}\right|}{\alpha} \tag{18}$$

Note that $\kappa_i$ represents the cumulative weight of negative literals for $i$-th iteration and $N_v$ is the desired number of negative literals in $\Gamma_{r2SAT}$. On the other hand, $f_{fit}$ denoted the value of the ideal fitness ($f_{fit} = 0$) and $f_{best}$ is the current best fitness achieved for $i$-th iteration. In this context, $n$ and $\alpha$ is the number of iterations and the number of neuron combinations, respectively.

### 5.5. Statistical Analysis

Besides the error analysis, the overall performance of *ABC* was statistically analyzed using the Friedman test for all values of $r$. Friedman test is used to detect the significant differences between the results of two or more algorithms that participated in the logic phase that control the distribution of negative literals in the logical structure. The Friedman test compares several algorithms by computing the ranking ($R$) of the observed results for each algorithm. For instance, the first-best results and second-best results are considered rank 1 and 2, respectively; rank $n$ is given to the worst results. The formulation of the Friedman test is presented in Equation (19):

$$\psi = \frac{12}{\omega\delta(\omega+1)}\left[\sum_{m=1}^{\omega} R_j^2\right] - 3\delta(\omega+1) \tag{19}$$

where $\omega$ is the number of algorithms participated in this study and $\delta$ denotes the number of cases runs. Note that the distribution of $p$-value is based on chi-squared ($\chi^2$) distribution with $\omega - 1$ degree of freedom. The results will be significant if the $p$-value is less than 0.05.

### 5.6. Model Reproducibility

The same experiments should be conducted to reproduce a framework with DHNN-$r$2SAT repeatedly on certain data sets or obtain the same results. Specific environments or factors should be considered as follows:

(i) The logic phase must be conducted with a random $\beta$ logical combination to avoid biasedness to only one specific $\Gamma_{r2SAT}$ structure. In addition, the threshold iteration in the logic phase is set at 100 to distinguish the maximum capacity of a certain algorithm in successfully generating the desired $\Gamma_{r2SAT}$.

(ii) In order to retrieve the synaptic weight in the training phase, the Wan Abdullah method [2] was utilized. According to Bazuhair et al. [26], in representing the strength between the neurons in DHNN, the Wan Abdullah method is more stable than Hebbian learning.

(iii) It is also worth mentioning that different logic gate operators in food source equations have different capabilities in producing negative literal. The next subsection will explain the implication of different logic gate operators in Equation (13).

### 5.7. The Implication Truth Table for Food Source Equation

The configuration of the possible input and output variables for each logic gate can be determined based on the truth table in Tables 7–11. Note that Setup I indicates the condition where the random $\varphi$ value is less than 0.5 and Setup II indicates the condition where the random $\varphi$ value is equal and more than 0.5. These two setups are important to control the probability of generating '0' and '1' in *ABC* [10]. The control parameter of *ABC* in the logic phase has more probability to be a binary number '1' when $\varphi < 0.5$ because we want the food source to produce more negative literal as this study conducted a high value of $r$. Notice that different logic gate operators produce a different probability of producing negative literals.

**Table 7.** The implication truth table for NAND.

| | $r2SAT_{nand-ABC}$ | | Setup I | Setup II | NAND with Setup I | NAND with Setup II |
|---|---|---|---|---|---|---|
| $x_{i,j}$ | $x_{k,j}$ | $\overline{(x_{i,j} \wedge x_{k,j})}$ | $\varphi < 0.5$ | $\varphi \geq 0.5$ | $v_{i,j}$ | $v_{i,j}$ |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |

**Table 8.** The implication truth table for OR.

| | $r2SAT_{or-ABC}$ | | Setup I | Setup II | OR with Setup I | OR with Setup II |
|---|---|---|---|---|---|---|
| $x_{i,j}$ | $x_{k,j}$ | $(x_{i,j} \vee x_{k,j})$ | $\varphi < 0.5$ | $\varphi \geq 0.5$ | $v_{i,j}$ | $v_{i,j}$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |

**Table 9.** The implication truth table for AND.

| | **$r$2SAT$_{and-ABC}$** | | **Setup I** | **Setup II** | **AND with Setup I** | **AND with Setup II** |
|---|---|---|---|---|---|---|
| $x_{i,j}$ | $x_{k,j}$ | $(x_{i,j} \wedge x_{k,j})$ | $\varphi < 0.5$ | $\varphi \geq 0.5$ | $v_{i,j}$ | $v_{i,j}$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |

**Table 10.** The implication truth table for XOR.

| | **$r$2SAT$_{xor-ABC}$** | | **Setup I** | **Setup II** | **XOR with Setup I** | **XOR with Setup II** |
|---|---|---|---|---|---|---|
| $x_{i,j}$ | $x_{k,j}$ | $(x_{i,j} \underline{\vee} x_{k,j})$ | $\varphi < 0.5$ | $\varphi \geq 0.5$ | $v_{i,j}$ | $v_{i,j}$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |

**Table 11.** The implication truth table for NOR.

| | **$r$2SAT$_{nor-ABC}$** | | **Setup I** | **Setup II** | **NOR with Setup I** | **NOR with Setup II** |
|---|---|---|---|---|---|---|
| $x_{i,j}$ | $x_{k,j}$ | $(x_{i,j} \overline{\vee} x_{k,j})$ | $\varphi < 0.5$ | $\varphi \geq 0.5$ | $v_{i,j}$ | $v_{i,j}$ |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |

## 6. Results and Discussion

This section comprises two subsections: the comparison of the five different logic gates operators and the comparison of $r$2SAT$_{nand-ABC}$ with $r$2SAT$_{GA}$. The first subsection will compare the performance of five different logic gates in $r$2SAT$_{ABC}$ in terms of capability producing desired logic. Then, the performance of $r$2SAT$_{nand-ABC}$ will be compared with three state-of-the-art binary algorithms: $r$2SAT$_{GA}$, $r$2SAT$_{ES}$, and $r$2SAT$_{PSO}$. For clarity, the best performance has been shown in boldface. The '+', '$-$', and '=' indicates the performance of the model is better than, less than, and perform equally with all comparison models, respectively.

### *6.1. Comparison of Five Different Logic Gates Operators*

#### 6.1.1. The Performance of Five Different Logic Gates Operators in Terms of MAE

Tables 12–14 illustrate the performance of all models in terms of MAE. The presented errors show the closeness of the achieved solution to the desired solution. The acceptable value of MAE is 0, which indicates the success of generated $r$2SAT with a correct number of negative literals in one iteration. In this case, the results signify the *ABC* performance measure in the logic phase for random $n$ logical combinations. In this section, the mean, maximum, and minimum values of MAE are selected based on the fitness obtained by specific $r$2SAT combinations. The MAE value provides an intuitive overview of the impact of different operators in the logic phase. In order to investigate the accuracy of our proposed *ABC* ($r$2SAT$_{nand-ABC}$), a comparative analysis is conducted with those reported in previous studies; $r$2SAT$_{xor-ABC}$ [13], $r$2SAT$_{and-ABC}$, $r$2SAT$_{or-ABC}$, and $r$2SAT$_{nor-ABC}$. In regard to different $\lambda$ and $r$, the performance of $r$2SAT$_{nand-ABC}$ outperformed other logic gate operators. The overall outcome of MAE is based on Tables 12–14 resulting from the following observations.

**Table 12.** Comparison of $r2\mathrm{SAT}_{nand-ABC}$ against other logic gates in terms of MAE for $\lambda = 10$.

| $\lambda$=10 $r$ | Performance Indicator | $r$**2SAT**$_{nand-ABC}$ | $r$**2SAT**$_{xor-ABC}$ | $r$**2SAT**$_{and-ABC}$ | $r$**2SAT**$_{or-ABC}$ | $r$**2SAT**$_{nor-ABC}$ |
|---|---|---|---|---|---|---|
| 0.5 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.6 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.7 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.8 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.9 | Mean | **0.0000** | **0.0000** | 0.3000 | **0.0000** | 0.3267 |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | 1.0000 | **0.0000** | 1.0000 |
| | +/−/= | 0/0/3 | 0/0/3 | 0/2/1 | 0/0/3 | 0/2/1 |

**Table 13.** Comparison of $r2\mathrm{SAT}_{nand-ABC}$ against other logic gates in terms of MAE for $\lambda = 50$.

| $\lambda$=50 $r$ | Performance Indicator | $r$**2SAT**$_{nand-ABC}$ | $r$**2SAT**$_{xor-ABC}$ | $r$**2SAT**$_{and-ABC}$ | $r$**2SAT**$_{or-ABC}$ | $r$**2SAT**$_{nor-ABC}$ |
|---|---|---|---|---|---|---|
| 0.5 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.6 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.7 | Mean | **0.0000** | 0.7233 | 1.1567 | **0.0000** | 1.0450 |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | 1.6667 | 2.0000 | **0.0000** | 2.0000 |
| | +/−/= | 0/0/3 | 0/2/1 | 0/2/1 | 0/0/3 | 0/2/1 |
| 0.8 | Mean | **0.0000** | 4.1200 | 6.1000 | **0.0000** | 5.6400 |
| | Min | **0.0000** | 1.6000 | 4.6000 | **0.0000** | 4.6000 |
| | Max | **0.0000** | 5.8000 | 7.4000 | **0.0000** | 7.2000 |
| | +/−/= | 0/0/3 | 0/3/0 | 0/3/0 | 0/0/3 | 0/3/0 |
| 0.9 | Mean | 0.0500 | 8.9800 | 11.4400 | **0.0000** | 11.0200 |
| | Min | **0.0000** | 7.6000 | 9.2000 | **0.0000** | 9.6000 |
| | Max | 0.5000 | 10.2000 | 12.4000 | **0.0000** | 12.8000 |
| | +/−/= | 0/2/1 | 0/2/0 | 0/3/0 | 2/0/1 | 0/3/0 |

The proposed model by Jia et al. [11] introduced logic gates to convert the problem to binary space. The approach enabled Kiran [13] to capitalize logic gate operators in the *ABC* algorithm by modifying the updating food source equation to improve the search space of optimal solutions. However, there is uncertainty as to which operators work well for solving binary problems. Regardless, from Tables 7–11 from the previous section, we can observe the changes in the probability of the bit in the solution string based on the problem in the logic phase. With no regard to $\varphi$ (Setup I or Setup II), the probability to produce 1 is more than 75% for $r2\mathrm{SAT}_{nand-ABC}$. This explains why $r2\mathrm{SAT}_{nand-ABC}$ works well with a higher value of $r$. As for $r2\mathrm{SAT}_{or-ABC}$, the modified *ABC* algorithm has the second-best MAE values for higher $r$ due to the probability of returning 1 being higher compared to other logical operators. It is expected where the performance of $r2\mathrm{SAT}_{and-ABC}$ and $r2\mathrm{SAT}_{nor-ABC}$ are poor due to the solution bit in the candidate solution tending to be 0. On the other hand, $r2\mathrm{SAT}_{xor-ABC}$ works well in an environment where $r$ is balanced, which is $r \in \{0.5, 0.6\}$. This is due to the changing probability being 50% for both setups. Although the food source does not represent the final fitness in the logic phase, the selection of logic gate operators is crucial in improving the fitness of employed and onlooker bees iteratively to locate the optimal solutions. The reason for such modification is to aid the employed and onlooker bees in generating $r$2SAT with a higher number of negative literals. The exploration effort increases with the number of literals at the high state. Comparative analysis in Tables 12–14 highlights the reason why such an approach is considered. If this factor is not taken into account, the entire bit solutions will always trap in local optima, mapped to an undesired number of negative literals.

**Table 14.** Comparison of $r2\mathrm{SAT}_{nand-ABC}$ against other logic gates in terms of MAE for $\lambda = 90$.

| $\lambda$=90 $r$ | Performance Indicator | $r2\mathrm{SAT}_{nand-ABC}$ | $r2\mathrm{SAT}_{xor-ABC}$ | $r2\mathrm{SAT}_{and-ABC}$ | $r2\mathrm{SAT}_{or-ABC}$ | $r2\mathrm{SAT}_{nor-ABC}$ |
|---|---|---|---|---|---|---|
| 0.5 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.6 | Mean | **0.0000** | 0.0500 | 0.0500 | **0.0000** | 0.3000 |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | 0.5000 | 0.5000 | **0.0000** | 1.0000 |
| | +/−/= | 0/0/3 | 0/2/1 | 0/2/1 | 0/0/3 | 0/2/1 |
| 0.7 | Mean | **0.0000** | 2.1800 | 5.0200 | **0.0000** | 4.8600 |
| | Min | **0.0000** | **0.0000** | 3.0000 | **0.0000** | 2.6000 |
| | Max | **0.0000** | 3.4000 | 7.4000 | **0.0000** | 7.0000 |
| | +/−/= | 0/0/3 | 0/2/1 | 0/3/0 | 0/0/3 | 0/3/0 |
| 0.8 | Mean | 0.2000 | 12.5400 | 15.8000 | **0.0000** | 15.2200 |
| | Min | **0.0000** | 9.4000 | 14.0000 | **0.0000** | 13.6000 |
| | Max | 0.5000 | 14.6000 | 18.0000 | **0.0000** | 16.8000 |
| | +/−/= | 0/2/1 | 0/3/0 | 0/3/0 | 2/0/1 | 0/3/0 |
| 0.9 | Mean | **0.5000** | 21.6200 | 24.0400 | 0.6500 | 25.1200 |
| | Min | **0.0000** | 17.8000 | 22.4000 | **0.0000** | 22.8000 |
| | Max | **1.0000** | 23.8000 | 27.8000 | 1.5000 | 27.6000 |
| | +/−/= | 2/0/1 | 0/3/0 | 0/3/0 | 0/2/1 | 0/3/0 |

According to Table 15, the performance of $r2\mathrm{SAT}_{nand-ABC}$ for Setup I does not exhibit major differences in terms of returning '1'. However, the probability for Setup II to return 1 is lowered from 100% to 75%. This is also apparently why certain logic gate operators work well for selective $r$. In contrast to different updating food source equations, the value of $\lambda$ proportionally increases with MAE. According to [27], the complex structure of the bit solution will cause the performance of $ABC$ to deteriorate due to the possibility of locating an optimal solution having grown exponentially and requiring better control of free parameters in the algorithm. Therefore, the value of MAE achieved, especially for $\lambda = 90$, is relatively larger than others. This is because a larger $\lambda$ indicates an extreme number of negative literals is desired, which requires the additional effort of exploitation and exploration process. Modifications in the employed or onlookers can be considered to increase the local capability of the network. To avoid this issue in the future, the proposed algorithm may consider a greedy selection to retain sub-optimal solutions [27].

### 6.1.2. The Capability of Five Logic Gates Operator Producing Desired Logic

This section will discuss the capability of five different logic gate operators in generating desired logic structures that will be evaluated based on the best fitness. The best fitness is the solution that is near to the desired results. As for this study, the scout bee plays an important role to evaluate the quality of the food source (fitness). The rate of scout is set to five, which is the evaluation in which a food source happens five times. Note that at least 5–10% of the population is a scout bee [28]. According to [29], the contribution of the scout bee causes the number of function evaluations to increase, which reduces the time complexity. However, the computation time increases because the search space is high for $r2\mathrm{SAT}_{xor-ABC}$, $r2\mathrm{SAT}_{nor-ABC}$, and $r2\mathrm{SAT}_{and-ABC}$ with the highest $\lambda$ and $r$. Figure 4 below illustrates the $f_{best}$ of one logical combination, $\beta$ for these models in the logic phase.

**Table 15.** Position update in Equation (14) with NAND logic gate operator with equal probability (unbiased $\varphi$ ).

| $r2\mathrm{SAT}_{nand-ABC}$ | | | **Setup I** | **Setup II** | **NAND with Setup I** | **NAND with Setup II** |
|---|---|---|---|---|---|---|
| $x_{i,j}$ | $x_{k,j}$ | $\overline{(x_{i,j} \wedge x_{k,j})}$ | $\varphi < 0.5$ | $\varphi \geq 0.5$ | $v_{i,j}$ | $v_{i,j}$ |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |

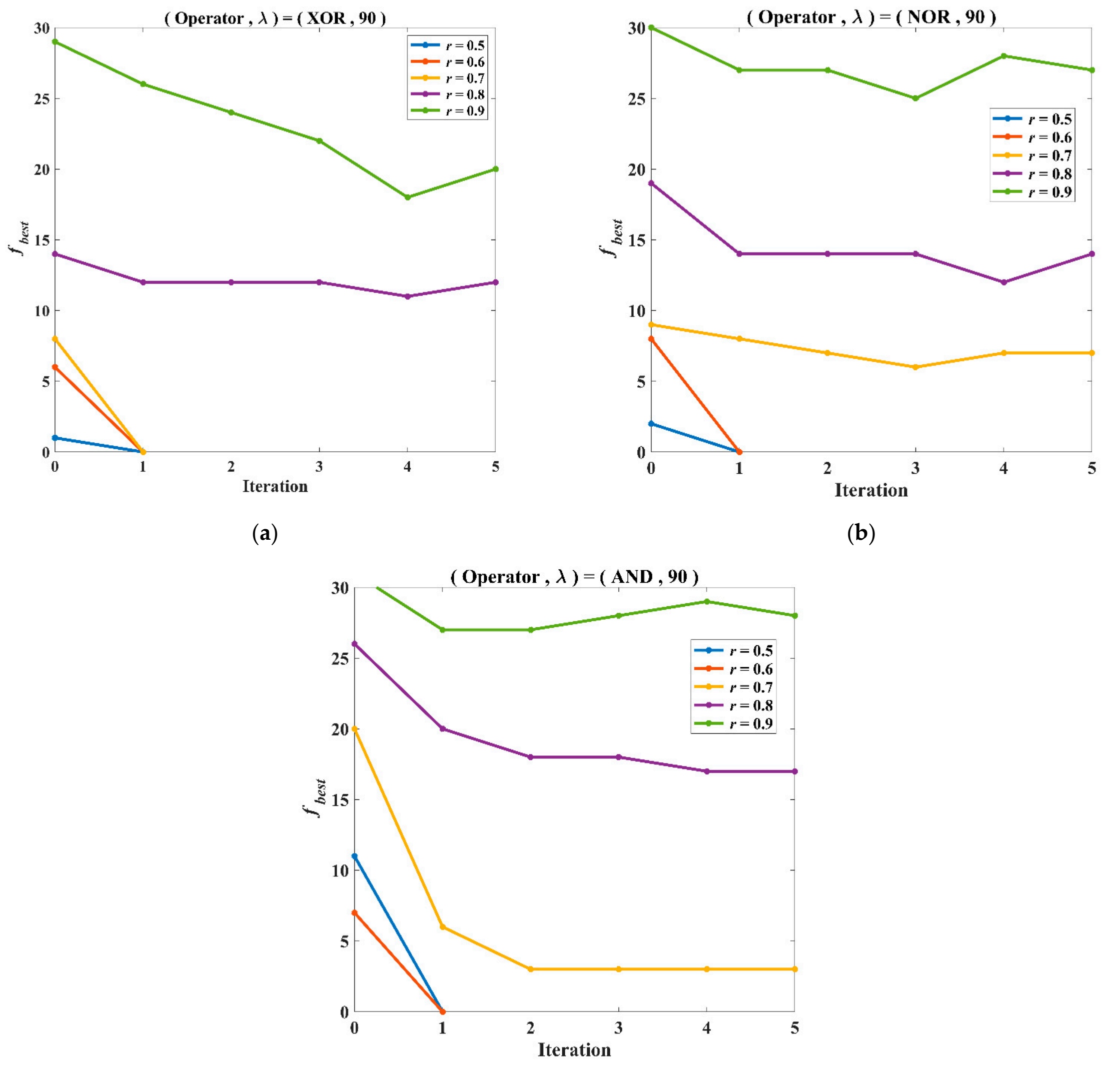


**Figure 4.** (**a**) $f_{best}$ for $r2\mathrm{SAT}_{xor-ABC}$; (**b**) $f_{best}$ for $r2\mathrm{SAT}_{nor-ABC}$; and (**c**) $f_{best}$ for $r2\mathrm{SAT}_{and-ABC}$.

Although the iterations have achieved the maximum number, the model is unable to generate the desired logical structure. Observe that premature convergence occurs due to the model converging toward a point where it was not supposed to converge [30]. According to [31], the model suffers from premature convergence due to the decrease in diversity. Despite being updated, the solution space has less diversity. For example, for $\lambda = 90$ with $r = 0.9$, the probability to obtain 90% of negative literals will be lower due to the probability to produce negative literals for $r2\text{SAT}_{xor-ABC}$, $r2\text{SAT}_{nor-ABC}$, and $r2\text{SAT}_{and-ABC}$ is less than 50%. Thus, there is a high tendency to see the same number of negative literal in the logic.

Figure 5 above illustrates the performance of one combination of logic, $\beta$ of $r2\text{SAT}_{nand-ABC}$ and $r2\text{SAT}_{or-ABC}$. With no regard, the performance of both models is the same. However, Table 14 shows that the maximum average of MAE of $r2\text{SAT}_{or-ABC}$ is higher compared to $r2\text{SAT}_{nand-ABC}$. In this study, ten $\beta$ will be generated. There is the possibility that others' $\beta$ require more iterations to obtain the desired logic. To summarize, $r2\text{SAT}_{nand-ABC}$ outperform other models in terms of the capability of converging to the desired fitness. Therefore, in the next section, the performance of $r2\text{SAT}_{nand-ABC}$ will be compared to other binary algorithms.

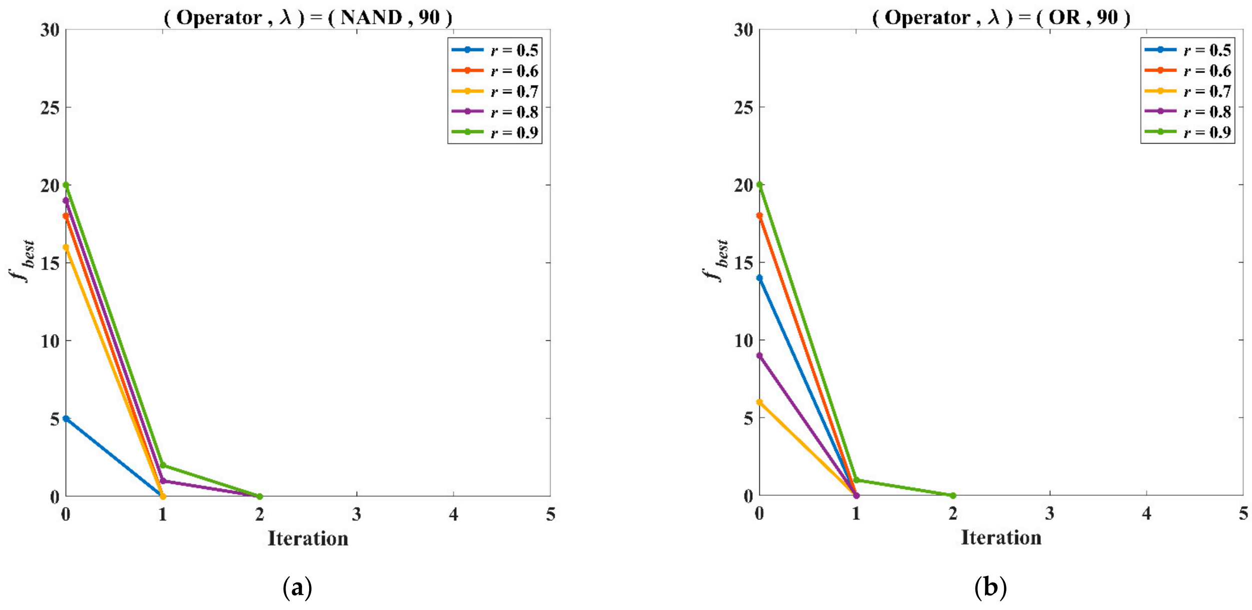


**Figure 5.** (**a**) $f_{best}$ for $r2\text{SAT}_{nand-ABC}$, (**b**) $f_{best}$ for $r2\text{SAT}_{or-ABC}$.

### *6.2. The Comparison of $r2SAT_{nand-ABC}$ with Other Binary Algorithms*

#### 6.2.1. The Comparison in Terms of Error Analysis

Tables 16–18 illustrate the comparative performance of $r2\text{SAT}_{nand-ABC}$ with $r2\text{SAT}_{GA}$, $r2\text{SAT}_{ES}$, and $r2\text{SAT}_{PSO}$ in terms of MAE.

Based on general observation of Tables 16–18, the performance of $r2\text{SAT}_{nand-ABC}$ when $r \in \{0.7, 0.8, 0.9\}$ for all $\lambda$ outperforms other state-of-the-art algorithms. Notice that $r2\text{SAT}_{ES}$ has a relatively larger average of MAE compared to other algorithms. This is due to the ES mechanism that is based on trial and error where it discovers all the possible solutions. According to [32], the trial-and-error mechanism managed to find the near-optimal solution, not the merely optimum. Figure 6 illustrated the convergence curve of one $\beta$ for each algorithm for all $r$ and $\lambda$. In the figure, observe that for $r \in \{0.8, 0.9\}$ with $\lambda = 90$, $r2\text{SAT}_{ES}$ managed to obtain lower $f_{best}$ for each iteration compared to $r2\text{SAT}_{PSO}$. As the size of $\lambda$ and $r$ increases, the number of possible solutions increases. Thus, it is impossible for ES to locate all the possible solutions in a few iterations only. Hence, ES requires more iterations as compared to PSO, which leads to an increment in MAE.

**Table 16.** Comparison of $r2\text{SAT}_{nand-ABC}$ against other binary algorithms in terms of MAE for $\lambda = 10$.

| $\lambda$=10 $r$ | Performance Indicator | $r$**2SAT**$_{nand-ABC}$ | $r$**2SAT** $_{GA}$ | $r$**2SAT** $_{ES}$ | $r$**2SAT** $_{PSO}$ |
|---|---|---|---|---|---|
| 0.5 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.6 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.7 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.8 | Mean | **0.0000** | **0.0000** | 0.1000 | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | 1.0000 | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/2/1 | 0/0/3 |
| 0.9 | Mean | **0.0000** | 0.1000 | 0.7284 | 0.1786 |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | 0.5000 | 1.8000 | 0.9524 |
| | +/−/= | 2/0/1 | 0/2/1 | 0/2/1 | 0/2/1 |

**Table 17.** Comparison of $r2\text{SAT}_{nand-ABC}$ against other binary algorithms in terms of MAE for $\lambda = 50$.

| $\lambda$=50 $r$ | Performance Indicator | $r$**2SAT**$_{nand-ABC}$ | $r$**2SAT** $_{GA}$ | $r$**2SAT** $_{ES}$ | $r$**2SAT** $_{PSO}$ |
|---|---|---|---|---|---|
| 0.5 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.6 | Mean | **0.0000** | 0.0500 | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | 0.5000 | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/2/1 | 0/0/3 | 0/0/3 |
| 0.7 | Mean | **0.0000** | 0.7333 | 2.1600 | 0.8250 |
| | Min | **0.0000** | **0.0000** | 1.0000 | **0.0000** |
| | Max | **0.0000** | 2.0000 | 4.0000 | 1.5238 |
| | +/−/= | 2/0/1 | 0/2/1 | 0/3/0 | 0/2/1 |
| 0.8 | Mean | **0.0000** | 2.8000 | 7.8200 | 4.3905 |
| | Min | **0.0000** | **0.0000** | 7.0000 | 3.0476 |
| | Max | **0.0000** | 4.0000 | 10.0000 | 5.3810 |
| | +/−/= | 2/0/1 | 0/2/1 | 0/3/0 | 0/3/0 |
| 0.9 | Mean | **0.0500** | 5.4008 | 31.8950 | 8.9714 |
| | Min | **0.0000** | 4.0000 | 10.0000 | 7.6191 |
| | Max | **0.5000** | 6.2308 | 209.0500 | 9.7620 |
| | +/−/= | 3/0/0 | 0/3/0 | 0/3/0 | 0/3/0 |

Despite the fact that PSO and ES were both initiated with the same objective function, their searching capabilities of optimal solutions are indeed different. Exploration and exploitation are the main features of every optimization algorithm. Exploration is the ability to discover unknown solution space, while exploitation is finding the better solution in a known region [33]. In *ABC*, the features are performed in a different way. Employed bees utilize exploration, in which the bees explore the environment of the hive (search space) to find a food source (solutions). Onlooker and scout bees utilize exploitation, which improves the current solutions if the new solution is better than the previous solution. Meanwhile, according to [34], the exploration features of PSO occur when new solutions in the search space can be added to the population without being compared to the current solution. The exploitation occurs when the new solution is better than its best solution. Consequently, $r2\text{SAT}_{PSO}$ requires more iteration to improve the solution compared to $r2\text{SAT}_{nand-ABC}$. It is worth mentioning that $r2\text{SAT}_{GA}$ is the second-best algorithm in the logic phase. Unfortunately, the mechanism of GA causes lower diversification of $\Gamma_{r2SAT}$ as the early solutions are usually nonfit and require optimization operators, such as crossover and mutation, before achieving the optimal solutions [4]. Additionally, the mutation in GA has a chance to reorder already existing solutions that will decrease the possibility of finding the optimal solution [35]. These features give a drawback to GA, which magnifies the errors in an average of MAE. From Figure 6, it is obvious that *ABC* has better convergence speed, as *ABC* is able to avoid being trapped in local minima solutions [10] where the solution space keeps improvising in two phases. The capability of $r2\text{SAT}_{nand-ABC}$ generating the desired number of negative literals in logic is related to the effectiveness of the food source equation. *ABC* improves the quality of the generated food sources (solution) in the early stages (employed bee) [36]. Although at the highest $\lambda$, $r2\text{SAT}_{nand-ABC}$ is able to obtain desired logic with few iterations that make the value of MAE lesser compared to other binary algorithms.

**Table 18.** Comparison of $r2\text{SAT}_{nand-ABC}$ against other binary algorithms in terms of MAE for $\lambda = 90$.

| **$\lambda$=90** <br> ***r*** | **Performance Indicator** | **$r2\text{SAT}_{nand-ABC}$** | **$r2\text{SAT}_{GA}$** | **$r2\text{SAT}_{ES}$** | **$r2\text{SAT}_{PSO}$** |
|---|---|---|---|---|---|
| 0.5 | Mean | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | +/−/= | 0/0/3 | 0/0/3 | 0/0/3 | 0/0/3 |
| 0.6 | Mean | **0.0000** | 0.2667 | 0.7221 | 0.2267 |
| | Min | **0.0000** | **0.0000** | **0.0000** | **0.0000** |
| | Max | **0.0000** | 0.6667 | 1.0000 | 0.6000 |
| | +/−/= | 2/0/1 | 0/2/1 | 0/2/1 | 0/2/1 |
| 0.7 | Mean | **0.0000** | 3.1822 | 8.9100 | 4.4476 |
| | Min | **0.0000** | 1.0000 | 5.0000 | 2.4286 |
| | Max | **0.0000** | 5.0000 | 12.6000 | 5.9048 |
| | +/−/= | 3/0/0 | 0/3/0 | 0/3/0 | 0/3/0 |
| 0.8 | Mean | **0.2000** | 8.3365 | 17.9400 | 15.0143 |
| | Min | **0.0000** | 6.0000 | 15.4000 | 13.3333 |
| | Max | **0.5000** | 9.5000 | 20.0000 | 16.1905 |
| | +/−/= | 3/0/0 | 0/3/0 | 0/3/0 | 0/3/0 |
| 0.9 | Mean | **0.5000** | 13.3613 | 28.5150 | 24.0238 |
| | Min | **0.0000** | 12.5000 | 26.0000 | 22.8571 |
| | Max | **1.0000** | 14.0690 | 34.0000 | 26.0476 |
| | +/−/= | 3/0/0 | 0/3/0 | 0/3/0 | 0/3/0 |

### 6.2.2. Statistical Analysis

A Friedman test was conducted for $r \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ with $\lambda \in \{10, 50, 90\}$. The degree of freedom, $df = 4$ and considers $a_0 = 0.05$ [37]. Suppose that $H_0$ have no significant results between $r2\text{SAT}_{nand-ABC}$ with all comparison algorithms. Table 19 is the average rank $(Av_R)$ of all algorithm and Table 20 is the result of the Friedman test.

According to the $Av_R$ obtained in Table 19, $r2\text{SAT}_{nand-ABC}$ has the best average rank among other algorithms. The closest competition is from $r2\text{SAT}_{GA}$ with slightly higher $Av_R$ for higher values of $r$. From Table 20, the $p$-value for all cases of $r$ is greater than $a_0 = 0.05$. Hence, the $H_0$ is accepted, which signifies that *ABC* is the powerful algorithm in the logic phase of DHNN in generating the desired number of negative literals in the $\Gamma_{r2SAT}$.

### 6.2.3. Impact Analysis

Thus far, the previous experimental and statistical results (Table 20) demonstrate that *ABC* is superior to another logic operator. This section presents the impact of the exploration and exploitation by $r2\text{SAT}_{nand-ABC}$, finding the optimal solution which corresponds to the desired value of $r$. As shown in the Pseudocode of *ABC* in the logic phase, the NAND gates were observed to explore more negated variable that corresponds to the given $r$. For the exploration front, the trajectory of the food obtained by the employed bee phase in Equation (14) was skewed towards 1's, which minimizes the objective function given in Equation (4). This is contrary to other gates which are more skewed towards zero, which increases the gap between the desired outcome with the current fitness of the bee. As the fitness value of the employed bees increases when $r2\text{SAT}_{nand-ABC}$ was implemented, the probability is given by Equation (16) for the onlooker bees to select the food source to increase dramatically. For different logical operators, such as XOR, AND, OR, and NOR, there is a high chance that the food selection will have equal probability to construct the desired $r2\text{SAT}$ logic. Hence, the effectiveness of Equation (16) will diminish completely, and more iteration is required to minimize the objective function. In terms of exploitation, onlooker bees acquire fast-track fitness because the exploitation to find an optimal food source (depending to the desired $N_v$) starts with high fitness food. Thus, onlooker bees are able to obtain zero objective function with almost zero iteration as shown in Tables 12–14. Unfortunately, other logical operators do not acquire similar benefit because low-fitness food was inherited from the employed bees via Equation (16). Thus, $r2\text{SAT}_{nand-ABC}$ was reported to provide a positive impact although the initial food source was started at random.

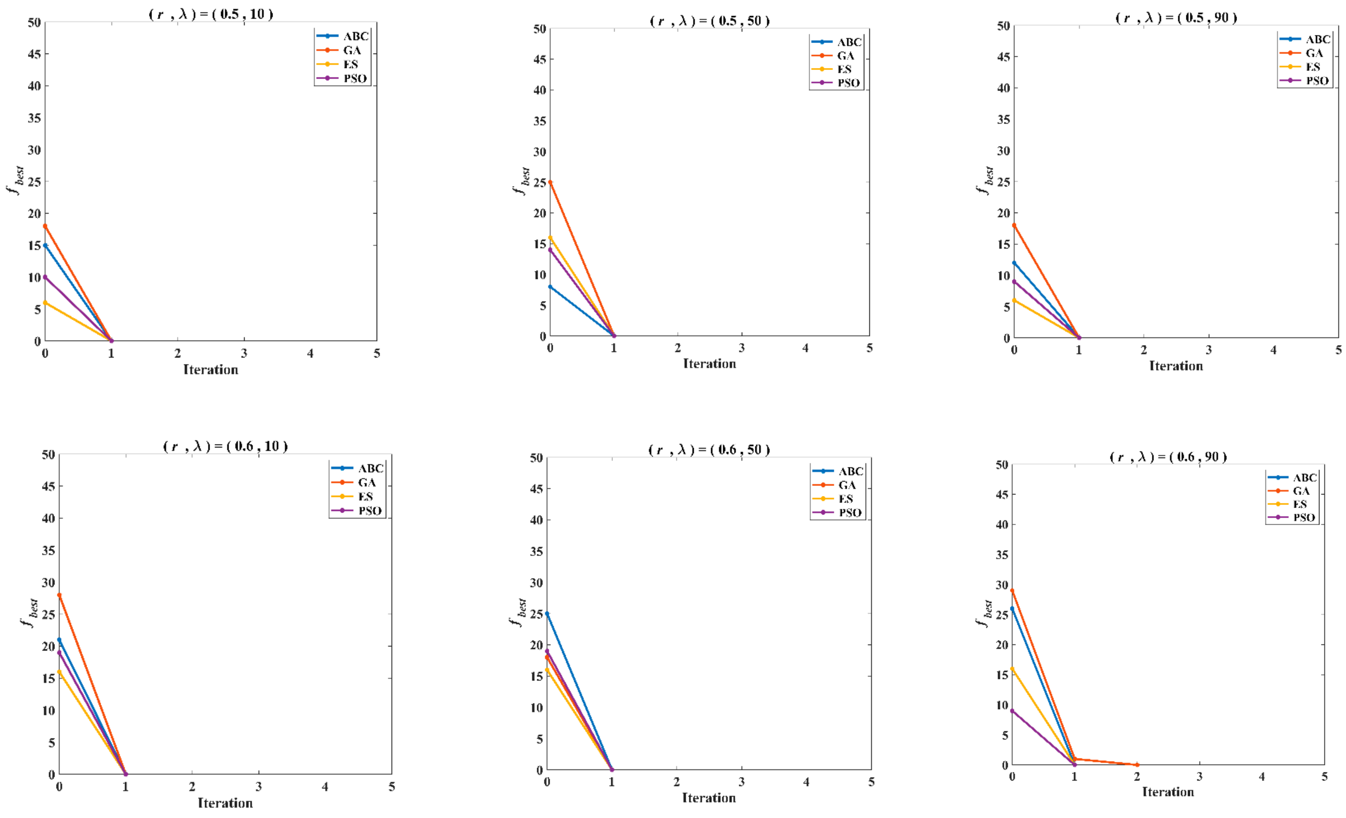


**Figure 6.** *Cont.*

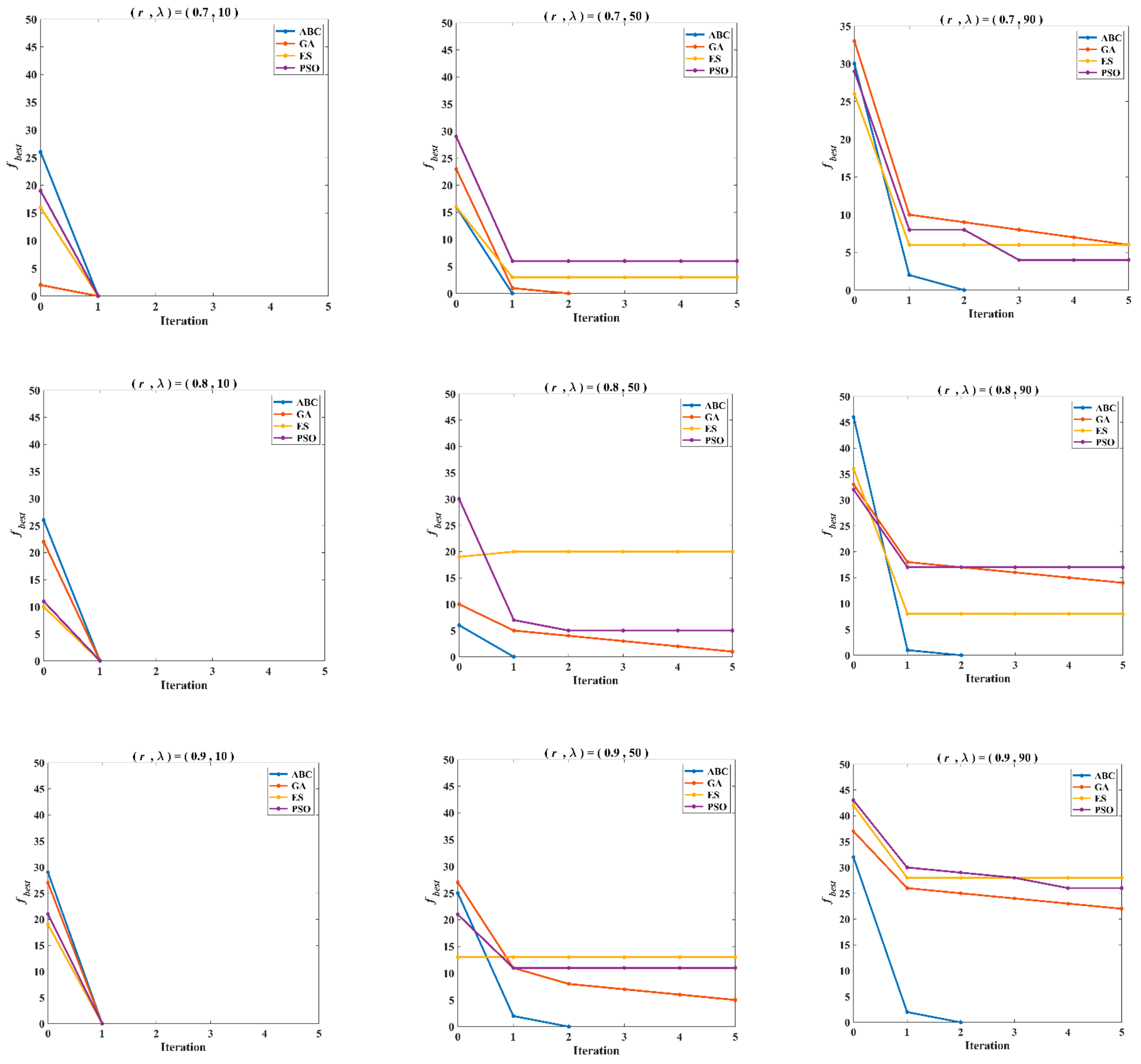


**Figure 6.** Convergence Curve for each Algorithms.

**Table 19.** $Av_R$ of all algorithms in the logic phase.

| $r$ | r2SAT$_{nand-ABC}$ | r2SAT $_{GA}$ | r2SAT $_{ES}$ | r2SAT $_{PSO}$ |
|---|---|---|---|---|
| 0.5 | 2.5000 | 2.5000 | 2.5000 | 2.5000 |
| 0.6 | 1.8333 | 3.0000 | 2.8333 | 2.3333 |
| 0.7 | 1.5000 | 2.1667 | 3.5000 | 2.8333 |
| 0.8 | 1.3333 | 2.0000 | 4.0000 | 2.6667 |
| 0.9 | 1.0000 | 2.0000 | 3.0000 | 4.0000 |

In comparison with another existing algorithm, the exploration and exploitation of the $r$2SAT$_{nand-ABC}$ were shown to be more effective. As shown in Figure 6, the convergence curve for $r$2SAT$_{nand-ABC}$ illustrated that *ABC* only requires a few iterations to obtain optimal solutions compared to another state-of-the-art algorithm. According to Su et al. [38], despite having more exploitation mechanisms, *ABC* managed to avoid being trapped in the local optimum solutions in the middle stage due to the exploitation mechanism occurring in the early stage (employed bees). Based on Figure 6, although $r2SAT_{GA}$ provides a competitive performance, the number of generations required to arrive at desired $\Gamma_{r2SAT}$ is large. Another obvious issue with $r2SAT_{GA}$ is high dependence towards the mutation operator during the initial stage due to the ineffective crossover phase. This demonstrates the primary weakness of $r2SAT_{GA}$ in terms of exploration and exploitation. As for $r2SAT_{PSO}$, the random positioning in PSO creates the same impact as another logic operator in $r2SAT_{ABC}$. This can be exhibited through the convergence curve in Figure 6, the particle positioning in PSO requires more iteration to explore the optimal $\Gamma_{r2SAT}$ structure. Meanwhile, the high error value in Table 17 and the convergence analysis shows that $r2SAT_{ES}$ is not impactful in terms of exploitation and exploration.

**Table 20.** Friedman test between $r2SAT_{nand-ABC}$ with all comparison algorithms.

| $r$ | Chi-Square Value, $\chi^2$ | $p$-Value | Accept/Reject $H_0$ |
|---|---|---|---|
| 0.5 | 0.0000 | 1.0000 | Accept $H_0$ |
| 0.6 | 1.5000 | 0.6823 | Accept $H_0$ |
| 0.7 | 4.0000 | 0.2615 | Accept $H_0$ |
| 0.8 | 7.0000 | 0.0719 | Accept $H_0$ |
| 0.9 | 9.0000 | 0.0611 | Accept $H_0$ |

## 7. Conclusions

The first contribution of this study introduces a systematic method in generating the logic with consideration of negative literal via Weighted Random 2 Satisfiability. The study shows that the logic phase is efficient to diversify the logical rule by controlling the dynamic of negative literal. The second contribution is that binary *ABC* was utilized in the logic phase to find the correct structure of $r2SAT$. Then, the performance of the proposed model was successfully compared with four variants of binary *ABC* with different logic gate operators in the food source equation. The experiments show the compatibility of $r2SAT_{nand-ABC}$ in the logic phase by generating a higher number of negative literals. Then, the proposed model is compared with three state-of-the-art algorithms: $r2SAT_{GA}$, $r2SAT_{ES}$, and $r2SAT_{PSO}$. *ABC* gives a solid performance as it requires few iterations to obtain the desired structure of logic. This study highlighted the capability of $r2SAT_{nand-ABC}$ in logic phase in generating the correct number of negative literals in a logical structure. In addition, this study also highlighted the capability of different logic gate operators in generating desired logical structure. The study can be further extended by considering the lowest values of $r$ which are 0.1, 0.2, and 0.3. Furthermore, different types of logic can generate in the logic phase, such as Random 2 Satisfiability [4], High order of Random $k$ Satisfiability [15], and Major 2 Satisfiability [5]. The robust architecture of $r2SAT$, integrated with an optimal operator for *ABC*, provides a view on the possible random dynamics for the application of real-life bioinformatics problem. For example, the proposed $r2SAT_{nand-ABC}$ can be embedded into logic mining which extract the best logical rule that classifies single-nucleotide polymorphisms (SNPs) inside known genes associated with Alzheimer's disease. This can lead to large-scale logic mining design incorporated with $r2SAT_{nand-ABC}$, which has the ability to classify and predict.

**Author Contributions:** Conceptualization and methodology, N.E.Z.; validation, H.A.W.; investigation and writing—original draft preparation, S.S.M.S.; writing—review and editing, M.A.M.; visualization, Y.G.; supervision and funding acquisition, M.S.M.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by Ministry of Higher Education Malaysia for Transdisciplinary Research Grant Scheme (TRGS) with Project Code: TRGS/1/2020/USM/02/3/2.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The authors would like to express special dedication to all of the researchers from AI Research Development Group (AIRDG) for the continuous support. The authors also would like to express a great appreciation to Vikneswari Someetheram and Revin Samuel James Selva Kumar for their helpful support that has been a great contributor to the completion of this paper.

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

## List of Notations

| | |
|---|---|
| ANN | Artificial Neural Network |
| HNN | Hopfield Neural Network |
| DHNN | Discrete Hopfield Neural Network |
| SAT | Boolean Satisfiability |
| 3SAT | 3 Satisfiability |
| 2SAT | 2 Satisfiability |
| RAN2SAT | Random 2 Satisfiability |
| RAN3SAT | Random 3 Satisfiability |
| MAJ2SAT | Major 2 Satisfiability |
| *ABC* | Artificial Bee Colony |
| GA | Genetic Algorithm |
| ES | Exhaustive Search |
| PSO | Particle Swarm Optimization |
| MAE | Mean Absolute Error |
| $r$ | Weighted ratio of negated literals |
| $\Gamma_{r2\mathrm{SAT}}$ | Logical structure of Weighted Random 2 Satisfiability |
| $\lambda$ | Total number of literals |
| $N_v$ | The desired number of negative literals |
| $f_L$ | Best fitness |
| $S_i$ | State of the *i*th neuron |
| $W_{(i,j)}$ | Synaptic weight from unit *i* to *j* |
| $\rho_i$ | Threshold constraints in DHNN |
| $E_{\Gamma_{r2SAT}}$ | Cost function of *r*2SAT |
| $h_i$ | Local field |
| $\wedge$ | Conjunction (AND) |
| $\vee$ | Disjunction (OR) |
| $\neg$ | Negation |
| $L_{\Gamma_{r2SAT}}$ | Lyapunov energy function |
| $x_{i,j}$ and $x_{k,j}$ | Two best bees |
| $v_{i,j}$ | New food source |
| $f$ | Control distance parameter |
| $f_i$ | Fitness of the bees |
| $p_i$ | Selection probability |
| $\overline{\vee}$ | Not or (NOR) |
| $\underline{\vee}$ | Exclusive or (XOR) |
| $\overline{\wedge}$ | Not and (NAND) |
| $r2\mathrm{SAT}_{nand-ABC}$ | Weighted Random 2 Satisfiability with Artificial Bee Colony using NAND operator |
| $r2\mathrm{SAT}_{or-ABC}$ | Weighted Random 2 Satisfiability with Artificial Bee Colony using OR operator |
| $r2\mathrm{SAT}_{and-ABC}$ | Weighted Random 2 Satisfiability with Artificial Bee Colony using AND operator |
| $r2\mathrm{SAT}_{xor-ABC}$ | Weighted Random 2 Satisfiability with Artificial Bee Colony using XOR operator |
| $r2\mathrm{SAT}_{nor-ABC}$ | Weighted Random 2 Satisfiability with Artificial Bee Colony using NOR operator |
| $r2\mathrm{SAT}_{GA}$ | Weighted Random 2 Satisfiability with Genetic Algorithm |
| $r2\mathrm{SAT}_{ES}$ | Weighted Random 2 Satisfiability with Exhaustive Search |
| $r2\mathrm{SAT}_{PSO}$ | Weighted Random 2 Satisfiability with Particle Swarm Optimization |

## References

1. Chereda, H.; Bleckmann, A.; Menck, K.; Perera-Bel, J.; Stegmaier, P.; Auer, F.; Kramer, F.; Leha, A.; Beißbarth, T. Explaining decisions of graph convolutional neural networks: Patient-specific molecular subnetworks responsible for metastasis prediction in breast cancer. *Genome Med.* **2021**, *13*, 42. [CrossRef] [PubMed]
2. Wan Abdullah, W.A.T. Logic programming on a neural network. *Int. J. Intell. Syst.* **1992**, *7*, 513–519. [CrossRef]
3. Kasihmuddin, M.S.M.; Mansor, M.; Sathasivam, S. Robust Artificial Bee Colony in the Hopfield Network for 2-Satisfiability Problem. *Pertanika J. Sci. Technol.* **2017**, *25*, 453–468.
4. Sathasivam, S.; Mansor, M.A.; Ismail, A.I.M.; Jamaludin, S.Z.M.; Kasihmuddin, M.S.M.; Mamat, M. Novel Random $k$ Satisfiability for $k \leq 2$ in Hopfield Neural Network. *Sains Malays.* **2020**, *49*, 2847–2857. [CrossRef]
5. Alway, A.; Zamri, N.E.; Karim, S.A.; Mansor, M.A.; Kasihmuddin, M.S.M.; Bazuhair, M.M. Major 2 Satisfiability Logic in Discrete Hopfield Neural Network. *Int. J. Comput. Math.* **2021**, 1–45. [CrossRef]
6. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. QANA: Quantum-based avian navigation optimizer algorithm. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104314. [CrossRef]
7. Nadimi-Shahraki, M.H.; Fatahi, A.; Zamani, H.; Mirjalili, S.; Abualigah, L.; Abd Elaziz, M. Migration-based moth-flame optimization algorithm. *Processes* **2021**, *9*, 2276. [CrossRef]
8. Kasihmuddin, M.S.; Mansor, M.; Md Basir, M.F.; Sathasivam, S. Discrete mutation Hopfield neural network in propositional satisfiability. *Mathematics* **2019**, *7*, 1133. [CrossRef]
9. Sathasivam, S.; Mansor, M.; Kasihmuddin, M.S.M.; Abubakar, H. Election algorithm for random $k$ satisfiability in the Hopfield neural network. *Processes* **2020**, *8*, 568. [CrossRef]
10. Karaboga, D.; Basturk, B. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In *International Fuzzy Systems Association World Congress*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 789–798.
11. Jia, D.; Duan, X.; Khan, M.K. Binary Artificial Bee Colony optimization using bitwise operation. *Comput. Ind. Eng.* **2014**, *76*, 360–365. [CrossRef]
12. Watada, J.; Roy, A.; Wang, B.; Tan, S.C.; Xu, B. An artificial bee colony-based double layered neural network approach for solving quadratic bi-level programming problems. *IEEE Access* **2020**, *8*, 21549–21564. [CrossRef]
13. Kiran, M.S. A binary artificial bee colony algorithm and its performance assessment. *Expert Syst. Appl.* **2021**, *175*, 114817. [CrossRef]
14. Ishtaiwi, A.; Alshahwan, F.; Jamal, N.; Hadi, W.; AbuArqoub, M. A Dynamic Clause Specific Initial Weight Assignment for Solving Satisfiability Problems Using Local Search. *Algorithms* **2021**, *14*, 12. [CrossRef]
15. Karim, S.A.; Zamri, N.E.; Alway, A.; Kasihmuddin, M.S.M.; Ismail, A.I.M.; Mansor, M.A.; Hassan, N.F.A. Random Satisfiability: A Higher-Order Logical Approach in Discrete Hopfield Neural Network. *IEEE Access* **2021**, *9*, 50831–50845. [CrossRef]
16. Roy, S.; Biswas, S.; Chaudhuri, S.S. Nature-Inspired Swarm Intelligence and Its Applications. *Int. J. Mod. Educ. Comput. Sci.* **2014**, *6*, 55–65. [CrossRef]
17. Kashan, M.H.; Nahavandi, N.; Kashan, A.H. DisABC: A new artificial bee colony algorithm for binary optimization. *Appl. Soft Comput.* **2012**, *12*, 342–352. [CrossRef]
18. Telikani, A.; Gandomi, A.H.; Shahbahrami, A.; Dehkordi, M.N. Privacy-preserving in association rule mining using an improved discrete binary artificial bee colony. *Expert Syst. Appl.* **2020**, *144*, 113097. [CrossRef]
19. Lin, G.; Xu, H.; Chen, X.; Guan, J. An effective binary artificial bee colony algorithm for maximum set k-covering problem. *Expert Syst. Appl.* **2020**, *161*, 113717. [CrossRef]
20. Wang, Z.; Cao, L.; Si, H. An improved genetic algorithm for determining the optimal operation strategy of thermal energy storage tank in combined heat and power units. *J. Energy Storage* **2021**, *43*, 103313. [CrossRef]
21. Muñoz, A.; Rubio, F. Evaluating genetic algorithms through the approximability hierarchy. *J. Comput. Sci.* **2021**, *53*, 101388. [CrossRef]
22. Hou, Y.; Hao, G.; Zhang, Y.; Gu, F.; Xu, W. A multi-objective discrete particle swarm optimization method for particle routing in distributed particle filters. *Knowl.-Based Syst.* **2022**, *240*, 108068. [CrossRef]
23. Mendler, A.; Döhler, M.; Ventura, C.E. Sensor placement with optimal damage detectability for statistical damage detection. *Mech. Syst. Signal Process.* **2022**, *170*, 108767. [CrossRef]
24. Janson, S.; Middendorf, M.; Beekman, M. Honeybee swarms: How do scouts guide a swarm of uninformed bees? *Anim. Behav.* **2005**, *70*, 349–358. [CrossRef]
25. Zamri, N.E.; Mansor, M.; Mohd Kasihmuddin, M.S.; Alway, A.; Mohd Jamaludin, S.Z.; Alzaeemi, S.A. Amazon employees resources access data extraction via clonal selection algorithm and logic mining approach. *Entropy* **2020**, *22*, 596. [CrossRef] [PubMed]
26. Bazuhair, M.M.; Jamaludin, S.Z.M.; Zamri, N.E.; Kasihmuddin, M.S.M.; Mansor, M.; Alway, A.; Karim, S.A. Novel Hopfield Neural Network Model with Election Algorithm for Random 3 Satisfiability. *Processes* **2021**, *9*, 1292. [CrossRef]
27. Gu, X. Application Research for Multiobjective Low-Carbon Flexible Job-Shop Scheduling Problem Based on Hybrid Artificial Bee Colony Algorithm. *IEEE Access* **2021**, *9*, 135899–135914. [CrossRef]
28. Li, J.Q.; Xie, S.X.; Pan, Q.K.; Wang, S. A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. *Int. J. Comput. Commun. Control* **2011**, *6*, 286–296. [CrossRef]

29. Anuar, S.; Selamat, A.; Sallehuddin, R. A modified scout bee for artificial bee colony algorithm and its performance on optimization problems. *J. King Saud Univ.-Comput. Inf. Sci.* **2016**, *28*, 395–406. [CrossRef]
30. Squillero, G.; Tonda, A. Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. *Inf. Sci.* **2016**, *329*, 782–799. [CrossRef]
31. Santos, R.; Borges, G.; Santos, A.; Silva, M.; Sales, C.; Costa, J.C. A semi-autonomous particle swarm optimizer based on gradient information and diversity control for global optimization. *Appl. Soft Comput.* **2018**, *69*, 330–343. [CrossRef]
32. Weiss, T.; Moser, C.; Venus, D.; Berggren, B.; Togerro, A. Parametric multi-objective energy and cost analysis in the life cycle of nearly zero energy buildings− an exhaustive search approach. *Sustain. Build.* **2019**, *4*, 5. [CrossRef]
33. Morales-Castañeda, B.; Zaldivar, D.; Cuevas, E.; Fausto, F.; Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? *Swarm Evol. Comput.* **2020**, *54*, 100671. [CrossRef]
34. Akay, B. A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. *Appl. Soft Comput.* **2013**, *13*, 3066–3091. [CrossRef]
35. Rashid, T.A.; Abdullah, S.M. A hybrid of artificial bee colony, genetic algorithm, and neural network for diabetic mellitus diagnosing. *ARO-Sci. J. Koya Univ.* **2018**, *6*, 55–64. [CrossRef]
36. Fairee, S.; Prom-On, S.; Sirinaovakul, B. Reinforcement learning for solution updating in Artificial Bee Colony. *PLoS ONE* **2018**, *13*, e0200738. [CrossRef] [PubMed]
37. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. CCSA: Conscious neighborhood-based crow search algorithm for solving global optimization problems. *Appl. Soft Comput.* **2019**, *85*, 105583. [CrossRef]
38. Su, H.; Zhao, D.; Yu, F.; Heidari, A.A.; Zhang, Y.; Chen, H.; Li, C.; Pan, J.; Quan, S. Horizontal and vertical search artificial bee colony for image segmentation of COVID-19 X-ray images. *Comput. Biol. Med.* **2022**, *142*, 105181. [CrossRef] [PubMed]