AIMS Mathematics

AIMS Mathematics, 10(6): 14840−14892.
DOI: 10.3934/math.2025667
Received: 23 January 2025
Revised: 15 April 2025
Accepted: 21 April 2025
Published: 27 June 2025

https://www.aimspress.com/journal/Math

Research article

A hybrid firefly algorithm for synaptic weight optimization in discrete Hopfield neural networks applied to random 3-satisfiability problems

Xiaoya Chen and Saratha Sathasivam*

School of Mathematical Sciences, Universiti Sains Malaysia (USM), Penang 11800, Malaysia

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

Abstract: The training phase of the random 3-satisfiability problem in discrete Hopfield neural networks aims to identify more satisfying clauses, enhancing synaptic weight for energy function computation and minimizing network energy. The primary challenge lies in designing synaptic weights that can be dynamically optimized to adapt to various formula structures while ensuring complete clause satisfaction and avoiding local optima. This enables an optimal balance between clause satisfaction and network convergence performance. To address this challenge, this paper first proposes a method for determining synaptic weights during the training phase based on the logical relationships between clauses and variables, simplifying the computation process and improving solution efficiency. Second, the hybrid firefly algorithm is employed during the training phase to optimize the number of satisfied clauses. This is achieved through a balance of global and local search mechanisms and a diversity maintenance strategy, facilitating the identification of more satisfied clauses, thereby leading to the generation of high-quality synaptic weights. Consequently, during the retrieval phase of the network, local field updates are executed based on these synaptic weights to find the optimal neuron states, thereby minimizing the energy function and improving global convergence performance. To evaluate the effectiveness of the hybrid firefly algorithm and the simplification of synaptic weight computation, we employed a comprehensive performance evaluation framework composed of maximum fitness, fitness ratio, entropy-adjusted diversity metrics, weight error, global minima ratio, energy error, similarity indices, and runtime. Experimental results indicate that the proposed model outperforms both traditional discrete Hopfield neural network random 3-satisfiability models and those that combine election algorithms with discrete Hopfield neural network random 3-satisfiability models across multiple performance metrics.

Keywords: random 3-satisfiability; discrete Hopfield neural networks; synaptic weight; energy

function; hybrid firefly algorithm
Mathematics Subject Classification: 03B52, 68T27, 68N17, 68W99

1. Introduction

Artificial neural networks, also referred to as neural networks, are mainly mathematical models that simulate the human brain's neural system in processing complex information. In particular, Hopfield neural networks (HNNs) have been widely studied [1]. In 1982, Hopfield proposed a single-layer feedback fully connected neural network, which strongly promoted the research of neural networks. He introduced the Lyapunov function and proved the stability of the network [2]. In 1992, Wan Abdullah introduced the satisfiability problem (SAT) to Hopfield neural networks using a logic programming approach [3], by leveraging symmetric weight matrices to effectively encode logical constraints. This breakthrough opened new avenues for combinatorial optimization using neural networks.

In recent years, various metaheuristic optimization techniques have been extensively explored to enhance the performance of discrete Hopfield neural networks (DHNN) in solving SAT problems. The election algorithm (EA) has been widely adopted in training discrete Hopfield neural networks for satisfiability problems, where it optimizes neuron states through heuristic selection processes to improve clause satisfaction rates [4,5]. Other heuristic methods, including differential evolution (DE), ant colony optimization algorithm (ACO), and the firefly algorithm (FA), have demonstrated remarkable performance in discrete optimization tasks [6−8].

Firefly algorithm (FA), initially proposed by Xin-She Yang in 2008 [9], has attracted significant attention due to its dynamic balance between global exploration and local exploitation. Its binary variant, binary firefly algorithm (BFA), has shown promise in solving complex discrete optimization problems, including feature selection, network optimization, and combinatorial optimization [10,11]. However, when BFA is directly employed in the training phase of random 3-SAT within DHNN, it exhibits a tendency to prioritize local search driven by variations in light intensity. As a result, its capacity for effective global exploration is significantly constrained. This issue becomes even more evident in high-dimensional search spaces, where the intricate interplay between neuron states and logical constraints necessitates a more adaptable optimization approach.

To address these challenges, this paper proposes an improved hybrid firefly algorithm (HFA) that integrates an adaptive neighborhood mechanism and weight adjustment strategy to optimize logical satisfiability and enhance the training efficiency of DHNNs. Building on these recent advancements in Hopfield network optimization [12−14], the proposed HFA emphasizes improved adaptability to maintain a balance between global exploration and local exploitation. The adaptive mechanisms in our approach share conceptual similarities with Feng et al.'s fuzzy clustering approach [12] and Alsadie and Alsulami's modified firefly algorithm [14] but are specifically tailored to address the challenges posed by random 3-SAT problems. Furthermore, this study incorporates the WAN method with a random 3-SAT weight computation approach (WAN-LC), ensuring an efficient and scalable solution for higher-order logic constraints in neural networks.

The paper is organized as follows: Section 2 introduces related works. Section 3 introduces the basic formulation of the random 3-satisfiability interpretation. Section 4 describes the logic programming for the random 3-satisfiability problem in discrete Hopfield neural networks, and Section 5 details the weight computation mechanism of the logic clauses-based WAN method (WAN-LC). Section 6 presents the hybrid firefly algorithm developed for the training phase. Section 7 outlines the steps and procedures for implementing the proposed model, and Section 8 discusses the experimental environment and setup. Section 9 explains the comparison metrics used for performance evaluation, while Section 10 focuses on the experimental results and analysis. Finally, Section 11 concludes the paper by summarizing the study's findings and offering future research perspectives.

2. Related works

2.1. Hopfield neural networks in SAT problems

In 1992, Wan Abdullah utilized symmetric weight matrices to address logical constraints and combinatorial optimization, offering innovative insights into neural network learning and demonstrating excellent computational efficiency [3]. In 1993, Wan Abdullah further demonstrated the implementation method of logic programming in neural networks in his book "Logic Programming on a Neural Network" [15]. This research was a breakthrough of historical significance to the field of artificial neural networks and provided a new perspective for solving logical constraint problems. Following Wan Abdullah's work, many researchers have explored and expanded the subject. In 2008, Saratha used computer simulation experiments to compare the logical equivalence between the same corresponding literal and clause, and supported the method proposed by Wan Abdullah [16]. In 2011, Sathasivam introduced the reverse analysis method in combination with conjunctive normal form (CNF), officially designating it as the Wan Abdullah method. This approach significantly advanced the logical mining process within neural networks, enhancing its computational efficiency and reinforcing its academic significance [17]. Through the study of logic programming in Hopfield neural networks, Kasihmuddin et al. (2016) applied the Wan Abdullah method to convert mathematical models into 2-SAT representations, facilitating the reconstruction of Bezier curve models [18]. In 2016, Mansor et al. pioneered the integration of Hopfield networks with 3-SAT, enabling the optimization of pattern satisfiability problems [19]. In 2020, Sathasivam et al. demonstrated the potential of implementing random k-satisfiability ($k \leq 2$) logical rules within Hopfield neural networks. Their study highlighted the expanded applicability of Hopfield networks, particularly in addressing logical constraints and optimization tasks, providing both theoretical and practical insights [20]. In 2021, Karim et al. introduced a novel approach to addressing random satisfiability problems by integrating higher-order logical methods with discrete Hopfield neural networks. This study expanded the scope of HNN applications and provided a robust framework for tackling complex logical and optimization tasks. The findings carry both theoretical and practical significance, marking a pivotal advancement in the integration of HNNs and logical programming [21].

2.2. Metaheuristic algorithms for optimizing random k-satisfiability problems in discrete Hopfield neural networks

Metaheuristic algorithms can be used to solve various discrete optimization problems, including the optimization of random k-satisfiability problems in discrete Hopfield neural networks (DHNN-RANkSAT). For random k-satisfiability (RANkSAT) problems, different configurations exist, including $k = \{1,2,3\}$, $k = \{1,2\}$, and $k = \{2,3\}$. This study specifically focuses on the $k = \{1,2,3\}$ scenario within discrete Hopfield neural networks (DHNNs). In 2021, several studies investigated the election algorithm (EA) as a heuristic search method in DHNNs for solving random k-SAT problems. Bazuhair et al. (2021) demonstrated that EA significantly improves clause satisfaction rates in DHNNs, outperforming other search methods such as exhaustive search (ES) [5].

Similarly, Abubakar & Danrimi (2021) evaluated EA in DHNN-based Boolean satisfiability problems, comparing it with ant colony optimization (ACO) and ES, and concluded that EA provides superior global search capability and clause satisfaction performance [22]. Furthermore, Abubakar et al. (2021) explored higher-order RANkSAT scenarios, comparing EA against ES and artificial bee colony (ABC) algorithms. Their findings confirmed that EA consistently outperforms these methods in optimizing neuron states and accelerating the learning process in DHNNs [23]. In 2022, H. Abubakar et al. integrated the ant colony optimization (ACO) algorithm into the training process of Hopfield neural networks (HNNs) to address the random 3-satisfiability (RAN3SAT) problem. The model's performance was assessed using an agricultural soil fertility dataset, achieving a classification accuracy of 87.5% [24]. For other cases of random k-satisfiability, the election algorithm (EA) has also been widely employed in the training phase of discrete Hopfield neural networks (DHNNs) to enhance logical clause satisfaction. Sathasivam et al. (2020) introduced an EA-based approach, optimizing neuron states by evaluating the satisfaction rates of candidate solutions, thereby enhancing both network training and clause satisfaction [4]. The election algorithm (EA) has been widely applied in the training phase of discrete Hopfield neural networks (DHNNs) to solve higher-order stochastic satisfiability problems [25,26]. These methods utilize socially inspired election mechanisms to compare candidate solutions based on their clause satisfaction rates, thereby selecting optimal neuron states and improving both network learning efficiency and scalability. Abubakar et al. (2020) employed EA in DHNN training, demonstrating its effectiveness in accelerating random k-satisfiability (RANkSAT) problem-solving [27]. Similarly, Karim (2023) applied EA to enhance DHNN's efficiency and scalability in handling RANkSAT formulations [28]. While the election algorithm (EA) has been widely applied in training discrete Hopfield neural networks (DHNNs) for various k-satisfiability (k-SAT) problems, recent research has also explored alternative optimization techniques. In 2024, Yueling Guo et al. introduced a hybrid differential evolution algorithm (HDEA) into the training phase of random 2-satisfiability (R2SAT), markedly improving the global search capability and convergence speed. Compared with the traditional election algorithm (EA), HDEA demonstrated superior performance in terms of search ability and optimization efficiency [6]. Due to its simplicity, efficiency, and powerful global search capabilities, differential evolution has been extensively applied to complex problems such as combinatorial optimization and neural network training [7], providing a fresh perspective for investigating new algorithms aimed at solving satisfiability problems in discrete Hopfield neural networks. According to Zhang et al., the firefly algorithm also exhibits outstanding global optimization performance, as its light intensity–based attraction mechanism effectively balances global exploration and local exploitation [8]. Notably, this characteristic aligns with the search strategy of HDEA, suggesting its potential applicability to satisfiability problems in discrete Hopfield neural networks.

The firefly algorithm was proposed by Xin-She Yang in 2008, inspired by how fireflies attract each other by glowing. In this algorithm, each firefly is regarded as a candidate solution, whose search direction is guided by its luminance (i.e., the objective function value). A stochastic perturbation mechanism is introduced to maintain a dynamic balance between global search and local optimization [9]. Researchers have comprehensively reviewed its application to discrete optimization problems and proposed various improvements, such as the binary firefly algorithm (BFA), which significantly enhances its performance in discrete binary data optimization tasks [10,11]. These studies show that the firefly algorithm and its variants can effectively address complex optimization problems involving binary or discrete variables [29]. Through diverse improvements and extensions [e.g., the binary firefly algorithm (BFA) and the hybrid firefly algorithm (HFA)], the firefly algorithm has demonstrated remarkable performance in processing discrete binary data, particularly in feature selection, network optimization, and classification tasks [30−36]. Compared with other classical algorithms such as the genetic algorithm (GA), particle swarm optimization (PSO), and differential evolution (DE), the firefly algorithm exhibits strong competitiveness and adaptability in balancing global exploration with local exploitation. It has been widely used in machine learning and data mining domains [37−42]. The binary firefly algorithm (BFA), for instance, shows powerful solving capabilities in various NP-hard optimization tasks—including hardware/software partitioning [43] and vehicle routing [44], while its hybrid variants (e.g., HAFA) have excelled in data clustering and module assignment by achieving a dynamic balance between global and local searches in both binary and discrete environments [45]. Inspired by these applications and theoretical advances, the firefly algorithm and its variants show outstanding global search capabilities in optimization tasks involving discrete binary data [10].

2.3. Limitations of existing approaches and the need for improvement

In recent years, metaheuristic algorithms such as election algorithms (EA) and ant colony optimization (ACO) have made significant progress in solving random 3-SAT problems using discrete Hopfield neural networks (DHNN). However, they still suffer from some key limitations. Election algorithms, while providing reliable local search capabilities, still face challenges when dealing with large-scale data and complex SAT problems. Their positive-negative propagation strategy lacks a balanced global search, making it difficult to reconcile global and local optimization needs [25]. Ant colony optimization (ACO) has a slow convergence rate and is prone to getting trapped in local optima [22]. For the case of random k, where k = 1,2,3, EA is usually preferred [5,22,23]. However, traditional methods based on EA [5] and non-heuristic search still face significant limitations under complex constraints. Specifically, when dealing with mixed logical expressions containing both 2-SAT and 3-SAT clauses, existing EA-based approaches tend to fall into suboptimal solutions due to their limited search scope. As the number of neurons increases and the complexity of logical constraints grows, the solution efficiency further decreases [25]. Additionally, in cases involving 1-SAT and 2-SAT clauses, EA often suffers from insufficient exploration during the search process. It frequently terminates prematurely upon identifying an initial feasible solution, overlooking other potential solutions within the search space [6].

The firefly algorithm (FA) exhibits strong global and local search capabilities in continuous optimization problems by iteratively updating solutions based on physical distance-based attraction, light intensity, and random perturbations [9]. However, when applying FA directly to the random 3-SAT training of discrete Hopfield neural networks (DHNN), its original continuous update mechanism cannot be directly applied. On the one hand, the discrete search space requires modifications in terms of distance metrics and movement patterns to accommodate discretization. On the other hand, without such modifications, FA tends to rely excessively on light intensity for local search, which limits its ability to conduct sufficient global exploration in highly complex discrete search spaces [30,32−35]. Recent research also shows the effectiveness of hybrid firefly algorithms in enhancing global exploration across challenging optimization landscapes [46]. Therefore, it is necessary to introduce discrete modifications or adopt hybrid strategies to enhance FA's applicability to discrete optimization problems such as stochastic 3-SAT.

Based on these challenges, existing metaheuristic methods continue to demonstrate limitations in solving complex SAT problems. Therefore, this paper proposes a novel hybrid firefly algorithm (HFA), aiming to improve the efficiency of solving DHNNs in random SAT problems by introducing an enhanced search strategy and an adaptive perturbation mechanism. The HFA achieves superior solution space exploration through a dynamic balancing mechanism between global and local search strategies [35–41,45]. This approach enables the algorithm to maintain high solution stability and efficiency, even when dealing with more complex higher-order clauses and an increasing number of 1-SAT clauses. In this study, HFA is introduced to optimize neuron states in DHNN-RAN3SAT by improving solution space exploration rather than relying solely on deterministic neuron state updates. The main objectives of HFA are to maximize the number of satisfied logical clauses, improve network stability, and enhance the energy minimization process of DHNN. By increasing the number of satisfied clauses, HFA indirectly affects synaptic weight determination, thereby enhancing network stability. In this study, we experimentally observed that as the number of satisfied clauses increases, the Hopfield network is more likely to escape local optima (see Section 10). Furthermore, at this stage, the synaptic weights remain primarily determined by the logical clause structure rather than by neuron state transitions, further improving the overall stability of the network. The HFA achieves this by integrating a global search strategy with a local tuning mechanism [8,9]. Specifically, it dynamically adjusts neuron state updates based on the relative brightness of fireflies to guide solutions toward the global optimum [10,11]. Additionally, HFA incorporates an adaptive stochastic perturbation technique to enhance diversity, preventing premature convergence and avoiding local optima [29].

In terms of weight computation, this paper follows the core principle of the Wan Abdullah method, which states that synaptic weights should be directly determined by the structure of logical clauses [3,15]. To this end, we propose a direct weight computation scheme (WAN-LC) for random 3-SAT and higher-order logical rules. Specifically, random 3-SAT formulas are first converted into conjunctive normal form (CNF), and then the corresponding weights are determined based on the logical clause structure of the CNF. Compared with traditional WAN methods, WAN-LC does not require explicit application of De Morgan's laws or additional cost function calculations. It also avoids coupling weight computation with the dynamic update process of the Hopfield network. This approach not only preserves the theoretical essence of the WAN method but also simplifies the computational process, ensuring a direct correspondence between synaptic weights and logical clause structures. Moreover, WAN-LC enables a more accurate representation of the energy function mapping between neurons and logical clauses, thereby reducing the risk of the network becoming trapped in local minimum energy states. Consequently, WAN-LC significantly improves solution efficiency and global convergence performance. In complex scenarios, such as higher-order logical rules and random 3-SAT, WAN-LC leverages the synaptic information from DHNN regarding the original SAT formula to precisely guide network state updates, demonstrating significant advantages and adaptability in solving complex logical optimization problems. Thus, both WAN-LC and HFA comprehensively enhance the problem-solving capability of the discrete Hopfield neural network (DHNN) in random 3-SAT problems by optimizing the initial neuron states and improving the overall solution process, laying a solid foundation for addressing more complex logical optimization scenarios.

3. Random 3-satisfiability

The randomized k-satisfiability problem is a non-systematic logical formula in which each clause contains at most k literals (where each literal is either a positive or negative occurrence of a variable) [47]. This logic rule is characterized by variable clause lengths, and the formula can be constructed by randomly generating different numbers of clauses [48]. As k increases, the complexity of solving the k-SAT problem grows significantly, since a larger variable space must be explored to find an assignment that satisfies all clauses [49].

In a related study, Saratha et al. (2020) investigated the random k-SAT problem, focusing on randomly selected k values of 1 and 2 [20]. Subsequently, in 2021, Bazuhair MM et al. extended the study to k = 3, making the formula structure more complex [5]. In this paper, we restrict $k \leq 3$ to further explore its performance in the randomized 3-satisfiability (RAN3SAT) problem.

Building on the aforementioned studies, in this paper, we restrict the range of k to $k \leq 3$ to further explore its performance in the random 3-satisfiability problem. Consequently, the general formula expression of the RAN3SAT problem can be defined as follows:

$$\vartheta_{RAN3SAT} = \wedge_{i=1}^{h} Z_i^{(1)} \wedge_{i=1}^{n} Z_i^{(2)} \wedge_{i=1}^{m} Z_i^{(3)}. \tag{1}$$

Where h, m, and n are all positive integers, and 1, 2, and 3 represent first-order, second-order, and third-order clauses, respectively. The total number of clauses (Num$C$) in $\vartheta_{RANKSAT}$ is given by Num$C$=m+n+h. The defined form of the clause in $\vartheta_{RANKSAT}$ is as follows:

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

Here, literal F_i, negated literal $\neg F_i$. For 3-SAT formulas, examples of first-order, second-order, and third-order formulas are provided below:

$$\vartheta_{RAN3SAT} = F_1 \wedge F_2 \wedge (A_1 \vee \neg B_1) \wedge (\neg A_2 \vee B_2) \wedge (C_1 \vee \neg D_1 \vee E_1) \wedge (\neg C_2 \vee D_2 \vee E_2). \tag{3}$$

From the above equations, we have $Z_1^{(1)} = F_1$, $Z_2^{(1)} = F_2$, $Z_1^{(2)} = (A_1 \vee \neg B_1)$, $Z_2^{(2)} = (\neg A_2 \vee B_2)$, $Z_1^{(3)} = (C_1 \vee \neg D_1 \vee E_1)$, and $Z_2^{(3)}$=$(C_1 \vee \neg D_1 \vee E_1)$. The satisfiability of the formula is determined by substituting the neuron states' Boolean values {1,–1} for the variable values in each clause. If all the formula's clauses are satisfied, then the formula is considered satisfied. Thus, if the formula is satisfiable, $\vartheta_{RAN3SAT} = 1$; otherwise, it is considered unsatisfiable, $\vartheta_{RAN3SAT} = -1$. Due to the diverse combinations of clauses, $\vartheta_{RANkSAT}$ can take on various forms. In this study, we focus on cases where $k \leq 3$.

4. Random 3-satisfiability logic programming on discrete Hopfield neural networks

The Hopfield neural network (HNN) can function as a content-addressable memory, capable of storing and retrieving information, while also supporting asynchronous parallel processing [17]. In a discrete Hopfield neural network (DHNN), the states of neurons are bipolar, taking values from the set {−1,1} [50]. The state update function is defined as follows:

$$S_i = \begin{cases} 1 & \sum_j W_{ij} S_j > \xi_i \\ -1 & Otherwise \end{cases}. \tag{4}$$

In the equation, $S_i = 1$ represents the activation state of a neuron, while $S_i = -1$ indicates its inhibitory state. S_j denotes the state of the j-th neuron, W_{ij} is the connection weight between neuron i and neuron j, and ξ_i represents the corresponding threshold. The weight matrix W_{ij} is symmetric, with diagonal elements set to zero, such that $W_{ii} = W_{jj} = 0$ and $W_{ij} = W_{ji}$. This symmetry ensures that the energy function decreases monotonically, guaranteeing convergence [2,51]. The Hopfield neural network typically adopts a fully connected topology, where every pair of neurons is interconnected.

In the random 3-SAT problem, the number of literals in each clause is restricted to 3. For 3-clauses, Wan Abdullah's method begins by defining a given random satisfiable clause, negating it, and then calculating its cost function [3,5]. The cost function formula for random 3-SAT is as follows:

$$E_{\vartheta_{RAN3SAT}} = \frac{1}{2^3}\sum_j^m\left(\prod_i^3 V_i\right)_j + \frac{1}{2^2}\sum_j^n\left(\prod_i^2 V_i\right)_j + \frac{1}{2}\sum_i^h V_j. \tag{5}$$

Here, m, n, and h represent the number of clauses in $E_{\vartheta_{3sat}}$. V_{ij} is used to map the literal states as follows:

$$V_{ij} = \begin{cases}(1-S_x) & if\ \neg x_1\\ (1+S_x) & if\ \ \ x_1\end{cases}. \tag{6}$$

The objective is to minimize the cost function $E_{\vartheta_{3sat}}$ by verifying the consistency of the RAN3SAT logical rules.

$$min[E_{\vartheta_{RAN3SAT}}]. \tag{7}$$

The minimal cost function ensures that the obtained weight values W_{ij} are optimal. The smallest cost function corresponds to the "most consistent" selection of S_x. When the cost function equals 0, $\vartheta_{RANkSAT} = 1$.

Additionally, the local field for each neuron is calculated using the following formula:

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

The update rule for the local field is as follows:

$$S_i(t+1) = \begin{cases}1\ ,if\ \ sign\,(h_i) \geq 0\\ \ \ -1\ ,\ otherwise\end{cases}. \tag{9}$$

Through the update of the local field, the network's dynamic process ensures that the energy function continuously decreases, guaranteeing a monotonic reduction in energy as the neuron states evolve. This process continues until the minimum point corresponding to the final neuron state is reached. The minimum energy function can be represented using the Lyapunov energy function [2], expressed as follows:

$$F_{\vartheta_{3sat}} = -\frac{1}{3}\sum_i^{nC}\sum_j^{nC}\sum_k^{nC} W_{ijk}^{(3)} S_i - \frac{1}{2}\sum_i^{nC}\sum_j^{nC} W_{ij}^{(2)} S_i S_j - \sum_{i=1}^{nC} W_i^{(1)} S_i. \tag{10}$$

As the dynamics evolve, the energy function $F_{\vartheta_{3sat}}$ gradually decreases and reaches its minimum value $F_{P_{3sat}}^{min}$. The formula for calculating this minimum energy function is given as:

$$F_{\vartheta_{3sat}}^{min} = -\frac{n(Z_i^3)+2n(Z_i^2)+4n(Z_i^1)}{8}. \tag{11}$$

Here, $n(Z_i^3)$ and $n(Z_i^2)$ represent the number of three-literal and two-literal clauses, respectively, corresponding to $\vartheta_{RAN3SAT}$. Thus, the quality of the final neuron state can be evaluated using the following conditions:

$$\left|F_{\vartheta_{3sat}} - F_{\vartheta_{3sat}}^{min}\right| < \delta. \tag{12}$$

Here, δ represents a user-defined tolerance value.

In this study, we implement $\vartheta_{RAN3SAT}$ within the Hopfield neural network, referring to it as the DHNN-RAN3SAT model.

5. WAN method for logical clause-based weight calculation

In a discrete Hopfield neural network, computing synaptic weights plays a pivotal role in reducing energy to address logical constraint problems. Following the Wan Abdullah approach, the weights are derived by analyzing the relationship between the energy function and the cost function, ensuring that the network state progresses toward the optimal solution of the logical problem. The WAN method first constructs logical clauses, then applies negation to them, and subsequently determines the cost function based on the negated clauses. Ultimately, synaptic weights are obtained by evaluating the correlation between the cost function and the energy function, leading to an increase in computational complexity.

5.1. WAN method weight calculation

Step 1: Randomly generate logical formulas containing 1SAT, 2SAT, and 3SAT sentences

$$\vartheta_{RAN3SAT} = F_1 \wedge \neg F_2 \wedge (A_1 \vee B_1) \wedge (\neg A_2 \vee B_2) \wedge (C_1 \vee D_1 \vee E_1) \wedge (C_2 \vee \neg D_2 \vee \neg E_2). \quad (13)$$

Step 2: Negate the logical word formula

$$\neg\vartheta_{RAN3SAT} = \neg F_1 \vee F_2 \vee (\neg A_1 \wedge \neg B_1) \vee (A_2 \wedge \neg B_2) \\ \vee (\neg C_1 \wedge \neg D_1 \wedge \neg E_1) \vee (\neg C_2 \wedge D_2 \wedge E_2). \quad (14)$$

Step 3: Calculate the cost function of the logical sentence

Based on the logical formula $\neg\vartheta_{RAN3SAT}$ obtained in Step 2, calculate the cost function $E_{\vartheta_{RAN3SAT}}$ of the logical sentence. The cost function of each sentence is as follows:

Cost function for 1SAT sentence:

$$E_{z_i^{(1)}} = \frac{1}{2}(1 - S_{F_1}) + \frac{1}{2}(1 + S_{F_2}). \quad (15)$$

Cost function for 2SAT sentence:

$$E_{z_i^{(2)}} = \frac{1}{2}(1 - S_{A_1})\frac{1}{2}(1 - S_{B_1}) + \frac{1}{2}(1 + S_{A_2})\frac{1}{2}(1 - S_{B_2}). \quad (16)$$

Cost function for 3SAT sentence:

$$E_{z_i^{(3)}} = \frac{1}{2}(1 - S_{C_1})\frac{1}{2}(1 - S_{D_1})\frac{1}{2}(1 - S_{E_1}) + \frac{1}{2}(1 - S_{C_2})\frac{1}{2}(1 + S_{D_2})\frac{1}{2}(1 + S_{E_2}). \quad (17)$$

The overall cost function is:

$$E_{\vartheta_{RAN3SAT}} = \sum_{i=1}^{h} E_{z_i^{(1)}} + \sum_{i=1}^{n} E_{z_i^{(2)}} + \sum_{i=1}^{m} E_{z_i^{(3)}}, \quad (18)$$

where h, m, n is 2. The cost function calculates the degree of satisfaction of each sentence through the variable state. The larger the value, the more unsatisfied sentences there are. The more logical constraints are satisfied, the lower the value of the cost function.

Step 4: Weight calculation of Wan Abdullah method

Based on the cost function $E_{\vartheta_{RAN3SAT}}$ calculated in Step 3, it is mapped to the energy function $F_{\vartheta_{3sat}}$ to solve the corresponding network synaptic weights. Muna Mohammed Bazuhair et al. expanded the final energy function of HNN-RAN3SAT [5], and its energy function expression is:

$$F_{\vartheta_{3sat}} = -2\sum_{r}^{m}\left(W_{ijk}^{(3)} S_i S_j S_k\right)_r - \sum_{r}^{m+n}\left(W_{ij}^{(2)} S_i S_j\right)_r - \sum_{i=1}^{nC} W_i^{(1)} S_i \quad , i, j, k \in \{1, \cdots, nC\}. \quad (19)$$

According to Wan Abdullah's method, the specific form of the weight is mapped from the cost function in Step 3. The weight of the SAT sentence is shown in the following Tables 1–3. The weights obtained by comparing the cost function with the energy function are shown in Figure 1.

Table 1. 1SAT weight.

Weights	$Z_1^{(1)}$	$Z_2^{(1)}$
W_i	$\frac{1}{2}$	$-\frac{1}{2}$

Table 2. 2SAT weight.

Weights	$Z_1^{(2)}$	$Z_2^{(2)}$
W_i	$\frac{1}{4}$	$-\frac{1}{4}$
W_j	$\frac{1}{4}$	$\frac{1}{4}$
W_{ij}	$-\frac{1}{4}$	$\frac{1}{4}$

Table 3. 3SAT weight.

Weights	W_i	W_j	W_k	W_{ij}	W_{ik}	W_{jk}	W_{ijk}
$Z_1^{(3)}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$\frac{1}{16}$
$Z_2^{(3)}$	$-\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{16}$

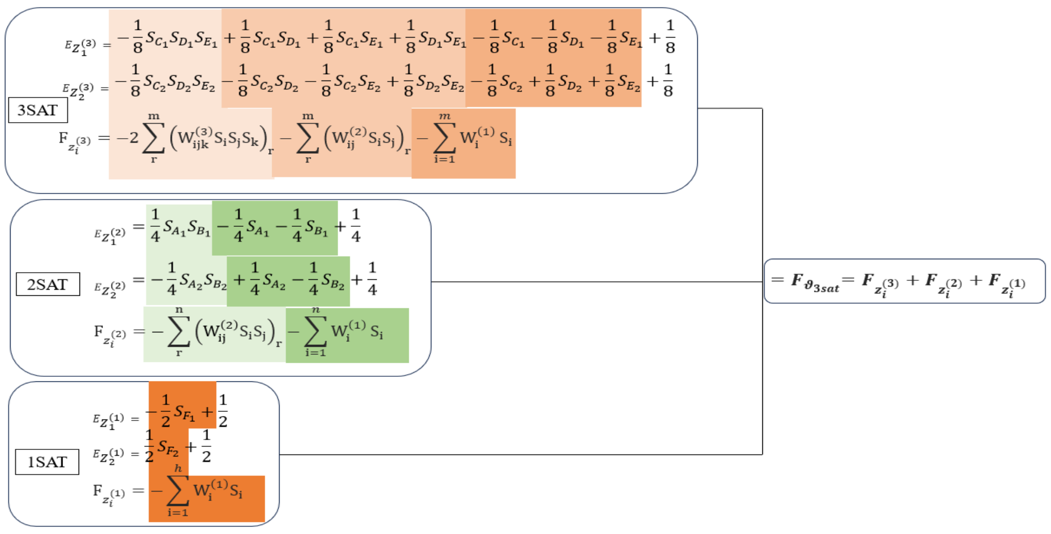

3SAT
$E_{Z_1^{(3)}} = -\frac{1}{8}S_{C_1}S_{D_1}S_{E_1} + \frac{1}{8}S_{C_1}S_{D_1} + \frac{1}{8}S_{C_1}S_{E_1} + \frac{1}{8}S_{D_1}S_{E_1} - \frac{1}{8}S_{C_1} - \frac{1}{8}S_{D_1} - \frac{1}{8}S_{E_1} + \frac{1}{8}$
$E_{Z_2^{(3)}} = -\frac{1}{8}S_{C_2}S_{D_2}S_{E_2} - \frac{1}{8}S_{C_2}S_{D_2} - \frac{1}{8}S_{C_2}S_{E_2} + \frac{1}{8}S_{D_2}S_{E_2} - \frac{1}{8}S_{C_2} + \frac{1}{8}S_{D_2} + \frac{1}{8}S_{E_2} + \frac{1}{8}$
$F_{z_i^{(3)}} = -2\sum_r^m \left(W_{ijk}^{(3)}S_iS_jS_k\right)_r - \sum_r^m \left(W_{ij}^{(2)}S_iS_j\right)_r - \sum_{i=1}^m W_i^{(1)} S_i$
2SAT
$E_{Z_1^{(2)}} = \frac{1}{4}S_{A_1}S_{B_1} - \frac{1}{4}S_{A_1} - \frac{1}{4}S_{B_1} + \frac{1}{4}$
$E_{Z_2^{(2)}} = -\frac{1}{4}S_{A_2}S_{B_2} + \frac{1}{4}S_{A_2} - \frac{1}{4}S_{B_2} + \frac{1}{4}$
$F_{z_i^{(2)}} = -\sum_r^n \left(W_{ij}^{(2)}S_iS_j\right)_r - \sum_{i=1}^n W_i^{(1)} S_i$
1SAT
$E_{Z_1^{(1)}} = -\frac{1}{2}S_{F_1} + \frac{1}{2}$
$E_{Z_2^{(1)}} = \frac{1}{2}S_{F_2} + \frac{1}{2}$
$F_{z_i^{(1)}} = -\sum_{i=1}^h W_i^{(1)} S_i$
$= F_{\vartheta_{3sat}} = F_{z_i^{(3)}} + F_{z_i^{(2)}} + F_{z_i^{(1)}}$

Figure 1. Traditional synaptic weight calculation.

5.2. WAN method for calculating the weight of a logic clause (WAN-LC)

In the discrete Hopfield neural network stochastic satisfiability problem (DHNN-RANSAT), the computation of synaptic weights is based on the structure of logical clauses. The traditional Wan

Abdullah method employs logical clauses in CNF (conjunctive normal form), where De Morgan's transformation is first applied to invert the clauses. The cost function is then computed and compared with the energy function to determine the synaptic weights.

Inspired by this approach, we propose a direct method for computing synaptic weights by directly utilizing CNF-form logical clauses and representing variable relationships using $\{1,-1\}$. This allows for weight computation without explicitly performing inverse transformations or cost function calculations. Our method preserves the core idea of the Wan Abdullah method while reducing computational steps and improving efficiency.

The contribution of each logical clause (1-SAT, 2-SAT, 3-SAT) to the synaptic weights is entirely determined by the structure of the logical clauses, with variable relationships represented by $\{1,-1\}$.

For 3-SAT clauses, the weight calculation involves the three-variable interaction term W_{ijk}, the two-variable interaction terms W_{ij} ,W_{ik},W_{jk} and the univariate weight terms W_i, W_j, W_k. For 2-SAT clauses, the weight calculation involves the two-variable interaction terms W_{ij} and the univariate weight terms W_i, W_j. For 1-SAT clauses, the weight calculation only involves the univariate weight term W_i.

In the initial CNF representation, logical variables are represented as $\mathrm{S}_{\mathrm{i}}^{(0)} = \{1, -1\}$, where negation follows $\bar{\mathrm{S}}_{\mathrm{i}}^{(0)} = -\mathrm{S}_{\mathrm{i}}^{(0)}$. These variable representations are used directly for weight computation, ensuring that the weight matrix is determined by logical clause structures rather than neuron state transitions. Neurons in the Hopfield network store these logical variable values, but the weight computation itself is based solely on logical clause structures. It should be noted that $\mathrm{S}_{\mathrm{i}}^{(0)}$ represents logical variables in the CNF clauses and is not to be interpreted as the initial states of neurons.

The characteristics of the weight matrix include the following:

Symmetry: The weight matrix is symmetric, satisfying $W_{ij} = W_{ji}$. For any permutation of i, j, k, W_{ijk} remains equal.

No self-feedback: Neurons are not self-connected, meaning $W_{ii} = W_{jj} = 0, W_{iii} = W_{jjj} = W_{kkk} = W_{iij} = W_{ijj} = 0$.

Using the Wan Abdullah method as outlined in Section 5.2, the general formula for calculating weights based on logical clauses is as follows. An example calculation is shown in Figure 2.

3-SAT: Three-variable interaction weights:

$$W_{ijk}^{(3)} = -\frac{1}{16}\left(\bar{S}_i^{(0)}\right)\left(\bar{S}_j^{(0)}\right)\left(\bar{S}_k^{(0)}\right) \text{ Where } W_{ijk}^{(3)} = W_{ikj}^{(3)} = W_{jki}^{(3)}. \tag{20}$$

Two-variable interaction weight:

$$W_{ij}^{(2)} = -\frac{1}{8}\left(\bar{S}_i^{(0)}\right)\left(\bar{S}_j^{(0)}\right), W_{ik}^{(2)} = -\frac{1}{8}\left(\bar{S}_i^{(0)}\right)\left(\bar{S}_k^{(0)}\right), W_{jk}^{(2)} = -\frac{1}{8}\left(\bar{S}_j^{(0)}\right)\left(\bar{S}_k^{(0)}\right) \text{ Where } W_{ij}^{(2)} = W_{ji}^{(2)}, W_{ik}^{(2)} = W_{ki}^{(2)}, W_{jk}^{(2)} = W_{kj}^{(2)}. \tag{21}$$

Univariate weights:

$$W_i^{(1)} = -\frac{1}{8}\left(\bar{S}_i^{(0)}\right),\ W_j^{(1)} = -\frac{1}{8}\left(\bar{S}_j^{(0)}\right),\ W_k^{(1)} = -\frac{1}{8}\left(\bar{S}_k^{(0)}\right). \tag{22}$$

2-SAT: Two-variable interaction weight:

$$W_{ij}^{(2)} = -\frac{1}{4}\left(\bar{S}_i^{(0)}\right)\left(\bar{S}_j^{(0)}\right) \text{ Where } W_{ij}^{(2)} = W_{ji}^{(2)}. \tag{23}$$

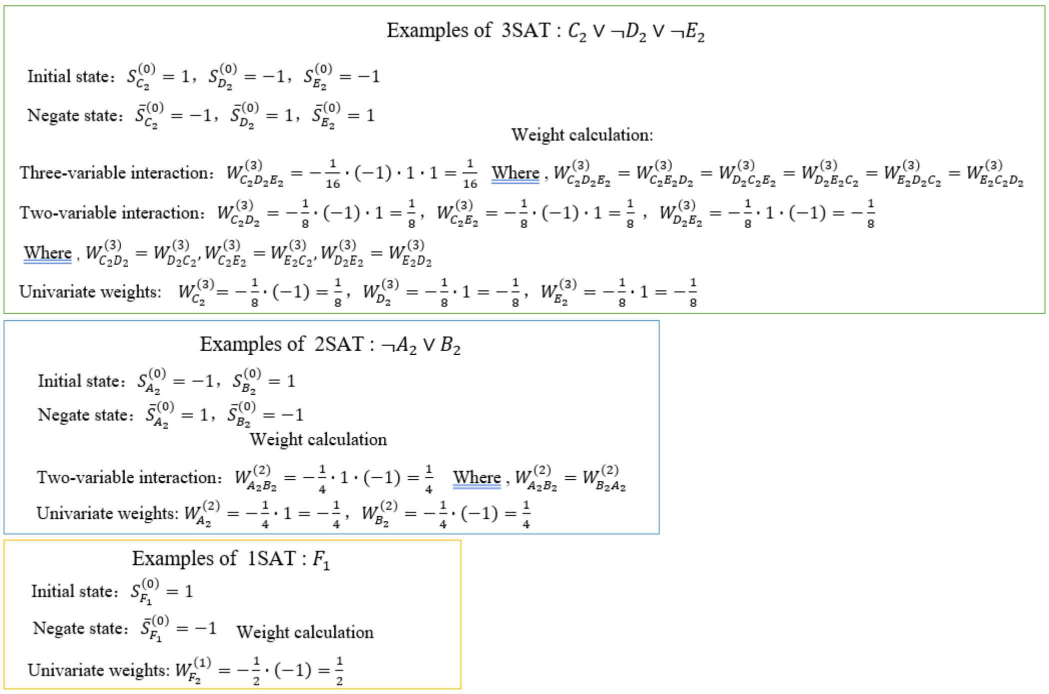
Univariate weights:

$$W_i^{(1)} = -\frac{1}{4}\left(\bar{S}_i^{(0)}\right), W_j^{(1)} = -\frac{1}{4}\left(\bar{S}_j^{(0)}\right). \tag{24}$$

1-SAT: Univariate weights:

$$W_i^{(1)} = -\frac{1}{2}\left(\bar{S}_i^{(0)}\right). \tag{25}$$

Directly computing synaptic weights based on the structure of logical clauses simplifies the calculation process, as illustrated in Figure 2 with examples of 1-SAT, 2-SAT, and 3-SAT. Figure 3 illustrates the process of obtaining synaptic weights using the WAN method in a discrete Hopfield neural network for the RAN3SAT satisfiability problem. Figure 4 demonstrates the improved WAN-LC method (Wan Abdullah with logical clauses), which directly utilizes logical clause structures to compute synaptic weights, where variable relationships are represented by $\{1,-1\}$, ensuring that weight determination relies solely on logical constraints rather than neuron states.



Examples of 3SAT : $C_2 \vee \neg D_2 \vee \neg E_2$

Initial state: $S_{C_2}^{(0)} = 1$, $S_{D_2}^{(0)} = -1$, $S_{E_2}^{(0)} = -1$

Negate state: $\bar{S}_{C_2}^{(0)} = -1$, $\bar{S}_{D_2}^{(0)} = 1$, $\bar{S}_{E_2}^{(0)} = 1$

Weight calculation:

Three-variable interaction: $W_{C_2D_2E_2}^{(3)} = -\frac{1}{16}\cdot(-1)\cdot 1\cdot 1 = \frac{1}{16}$ Where, $W_{C_2D_2E_2}^{(3)} = W_{C_2E_2D_2}^{(3)} = W_{D_2C_2E_2}^{(3)} = W_{D_2E_2C_2}^{(3)} = W_{E_2D_2C_2}^{(3)} = W_{E_2C_2D_2}^{(3)}$

Two-variable interaction: $W_{C_2D_2}^{(3)} = -\frac{1}{8}\cdot(-1)\cdot 1 = \frac{1}{8}$, $W_{C_2E_2}^{(3)} = -\frac{1}{8}\cdot(-1)\cdot 1 = \frac{1}{8}$, $W_{D_2E_2}^{(3)} = -\frac{1}{8}\cdot 1\cdot(-1) = -\frac{1}{8}$

Where, $W_{C_2D_2}^{(3)} = W_{D_2C_2}^{(3)}, W_{C_2E_2}^{(3)} = W_{E_2C_2}^{(3)}, W_{D_2E_2}^{(3)} = W_{E_2D_2}^{(3)}$

Univariate weights: $W_{C_2}^{(3)} = -\frac{1}{8}\cdot(-1) = \frac{1}{8}$, $W_{D_2}^{(3)} = -\frac{1}{8}\cdot 1 = -\frac{1}{8}$, $W_{E_2}^{(3)} = -\frac{1}{8}\cdot 1 = -\frac{1}{8}$

Examples of 2SAT : $\neg A_2 \vee B_2$

Initial state: $S_{A_2}^{(0)} = -1$, $S_{B_2}^{(0)} = 1$

Negate state: $\bar{S}_{A_2}^{(0)} = 1$, $\bar{S}_{B_2}^{(0)} = -1$

Weight calculation

Two-variable interaction: $W_{A_2B_2}^{(2)} = -\frac{1}{4}\cdot 1\cdot(-1) = \frac{1}{4}$ Where, $W_{A_2B_2}^{(2)} = W_{B_2A_2}^{(2)}$

Univariate weights: $W_{A_2}^{(2)} = -\frac{1}{4}\cdot 1 = -\frac{1}{4}$, $W_{B_2}^{(2)} = -\frac{1}{4}\cdot(-1) = \frac{1}{4}$

Examples of 1SAT : F_1

Initial state: $S_{F_1}^{(0)} = 1$

Negate state: $\bar{S}_{F_1}^{(0)} = -1$ Weight calculation

Univariate weights: $W_{F_2}^{(1)} = -\frac{1}{2}\cdot(-1) = \frac{1}{2}$

Figure 2. Example calculation.

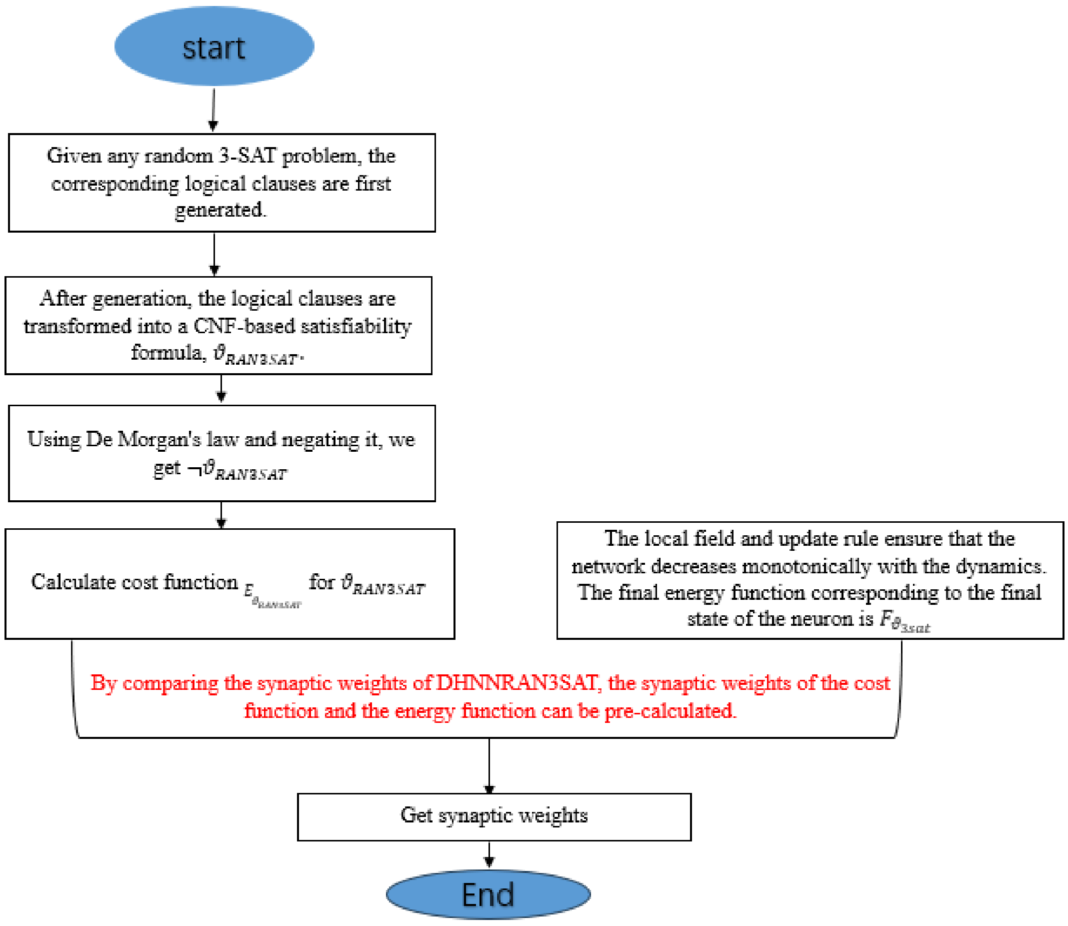

start
Given any random 3-SAT problem, the corresponding logical clauses are first generated.
After generation, the logical clauses are transformed into a CNF-based satisfiability formula, $\vartheta_{RAN3SAT}$.
Using De Morgan's law and negating it, we get $\neg\vartheta_{RAN3SAT}$
Calculate cost function $E_{\vartheta_{RAN3SAT}}$ for $\vartheta_{RAN3SAT}$
The local field and update rule ensure that the network decreases monotonically with the dynamics. The final energy function corresponding to the final state of the neuron is $F_{\vartheta_{3sat}}$
By comparing the synaptic weights of DHNNRAN3SAT, the synaptic weights of the cost function and the energy function can be pre-calculated.
Get synaptic weights
End

Figure 3. Synaptic weight calculation using the WAN method in DHNN for RAN3SAT.

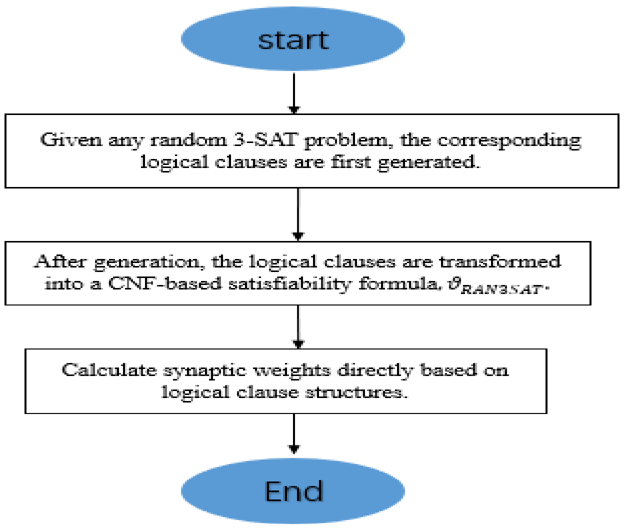

start
Given any random 3-SAT problem, the corresponding logical clauses are first generated.
After generation, the logical clauses are transformed into a CNF-based satisfiability formula, $\vartheta_{RAN3SAT}$.
Calculate synaptic weights directly based on logical clause structures.
End

Figure 4. Synaptic weight calculation using the improved WAN-LC method in DHNN for RAN3SAT.

6. The proposed method in the training phase

The primary objective of the proposed hybrid firefly algorithm (HFA) is to maximize the number of satisfied clauses in RAN3SAT during the training phase of the discrete Hopfield neural network (DHNN). Additionally, it aims to derive the optimal weight matrix based on the logical structure of the clauses. Compared to traditional binary firefly algorithms used for solving NP-hard problems [44,45], the HFA introduces significant improvements in balancing global exploration and local exploitation.

Recent advancements in hybrid firefly models, such as the HFASSON algorithm proposed by Idris et al. (2025), have shown that incorporating multiple intelligent strategies can significantly enhance convergence and solution quality in complex optimization tasks [46]. Inspired by this trend, our HFA integrates adaptive neighborhood adjustment and stochastic perturbation mechanisms specifically tailored for DHNN-RAN3SAT training. The detailed procedural steps of the HFA are outlined as follows (see Figure 5).

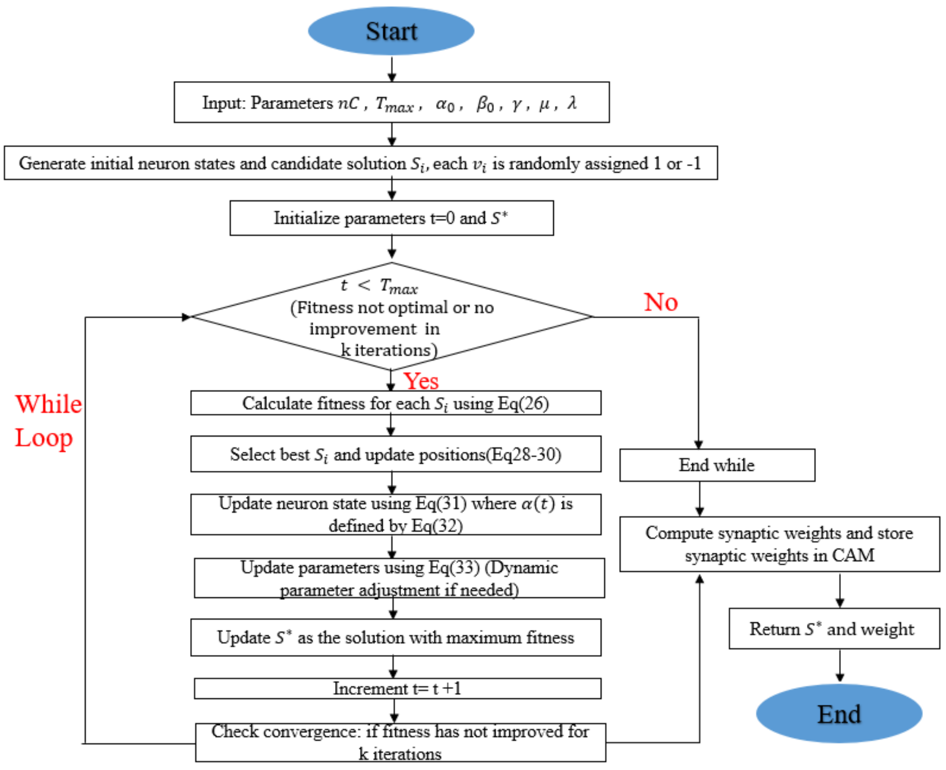

Start
Input: Parameters nC, T_{max}, α_0, β_0, γ, μ, λ
Generate initial neuron states and candidate solution S_i, each v_i is randomly assigned 1 or -1
Initialize parameters t=0 and S^*
$t < T_{max}$ (Fitness not optimal or no improvement in k iterations)
No
Yes
While Loop
Calculate fitness for each S_i using Eq(26)
Select best S_i and update positions(Eq28-30)
Update neuron state using Eq(31) where $\alpha(t)$ is defined by Eq(32)
Update parameters using Eq(33) (Dynamic parameter adjustment if needed)
Update S^* as the solution with maximum fitness
Increment t= t +1
Check convergence: if fitness has not improved for k iterations
End while
Compute synaptic weights and store synaptic weights in CAM
Return S^* and weight
End

Figure 5. Flowchart of the hybrid firefly algorithm (HFA) for DHNN-RAN3SAT training phase.

6.1. Initial population generation

Randomly generate combinations of neuron states, each consisting of a set of states $S_i = (v_1, v_2, \cdots, v_{nC})$ and a state $v_i \in \{-1,1\}$. Each individual represents a candidate solution in the search space. The search space for nC neurons is 2^{nC}, covering all possible combinations of neuron states. Each neuron state is assigned either -1 or 1 with equal probability (50%). This random initialization approach ensures extensive coverage of the initial population and mitigates the risk of premature convergence.

6.2. Fitness evaluation

The objective of the fitness function is to maximize the number of satisfied clauses, ensuring that the weight computation is determined directly by the structure of logical clauses rather than neuron state transitions. The algorithm evaluates the satisfaction of each logical clause and determines synaptic weights accordingly, without explicitly computing the cost function. This approach enhances the quality of candidate solutions by prioritizing clause satisfaction directly. The fitness value L_i for each candidate solution S_i is determined by the number of satisfied clauses, calculated as follows:

$$L_i = \sum_i^h f_i^{(1)} + \sum_i^n f_i^{(2)} + \sum_i^m f_i^{(3)}, \tag{26}$$

where the satisfaction status of clauses is defined by:

$$f_i^{(1)} = \begin{cases} 1\,, if \;\; Z_i^{(1)} \; is\; satisfied \\ 0. \qquad\qquad otherwise \end{cases}$$
$$f_i^{(2)} = \begin{cases} 1\,, if \;\; Z_i^{(2)} \; is\; satisfied \\ 0. \qquad\qquad otherwise \end{cases} \tag{27}$$
$$f_i^{(3)} = \begin{cases} 1\,, if \;\; Z_i^{(3)} \; is\; satisfied \\ 0. \qquad\qquad otherwise \end{cases}.$$

The fitness value of a candidate solution reflects the number of satisfied clauses, indicating its proximity to the global optimum. A cost function value of 0 signifies that all clauses in the formula are satisfied, meaning the candidate solution has achieved the global optimum. By evaluating fitness, the algorithm effectively sifts through candidate solutions, prioritizing the best options and directing the neural network toward its minimum energy state.

6.3. Position update

The position update mechanism effectively balances global and local search by encouraging individuals to gravitate toward those with higher fitness levels. The distance between individuals is determined using the Euclidean distance formula, and the strength of attraction diminishes as this distance increases, mirroring the behavior of fireflies in nature.

During the position update phase, the algorithm refines the positions of candidate solutions by assessing the differences in firefly brightness, which facilitates the optimization of neuron states. A higher fitness level is associated with greater brightness, leading to a stronger attraction force among individuals. The following update is applied to the neuron state:

$$S_i(t+1) = S_i(t) + \beta_0 e^{-\gamma r_{ij}^2}\left(S_j - S_i\right). \tag{28}$$

The notation $S_i(t)$ refers to the neuron-state vector of individual i at iteration t. Meanwhile, S_j signifies the neuron-state vector of individual j, who displays the highest level of responsiveness. β_0 represents the initial attraction strength, while γ is the coefficient for light intensity attenuation. Additionally, r_{ij} denotes the distance between individuals i and j.

$$r_{ij} = \left\|S_i - S_j\right\|. \tag{29}$$

A measure of the differences in neuron states within the solution space assists in computing the attraction strength. An adaptive light intensity adjustment mechanism dynamically modulates attractiveness based on both inter-individual distance and iteration count. This approach ensures a broad global search during the early stages and a refined local search in later stages, helping to prevent the algorithm from becoming trapped in a local optimum.

$$\beta(t) = \beta_0 e^{-\lambda t}. \tag{30}$$

The dynamic decay rate, denoted by λ, is responsible for the gradual decrease in attractiveness over time, while t represents the current iteration index.

6.4. Random perturbation

To prevent premature convergence of individuals to local optima, the algorithm incorporates random perturbations that modify neuron states through uniformly distributed noise. This strategy serves two critical functions at distinct stages of the search process. Initially, it employs a broader search radius to promote global exploration. As the intensity of the perturbation decreases, fine-tuning is implemented to avoid local minima, thereby enhancing overall convergence accuracy.

The random perturbation mechanism introduces a controlled degree of randomness, aiding candidate solutions in escaping local optima and improving global exploration. Throughout the search process, the algorithm adjusts neuron states by applying uniformly distributed noise to mitigate the risk of premature convergence. This mechanism functions in two phases: first, by expanding the search radius to facilitate global exploration, and subsequently by focusing on fine-tuning as perturbation intensity diminishes, ultimately improving convergence accuracy. The random perturbation process effectively enhances the ability of solutions to evade local optima and boosts global exploration.

$$S_i(t+1) = S_i(t) + \alpha \cdot RandomNoise. \tag{31}$$

The notation $S_i(t)$ signifies the neuron-state vector of an individual i at iteration t. The variable α serves as the perturbation coefficient, which governs the magnitude of the perturbation, whereas $RandomNoise$ represents a random noise component that adheres to a uniform distribution within the interval $[-1,1]$ This strategy effectively introduces a moderate degree of uncertainty into the solution space, thereby enhancing both diversity and the robustness of the search process.

Furthermore, an adaptive perturbation mechanism contributes to population diversity and adaptability by dynamically adjusting the perturbation coefficient $\alpha(t)$ over time. In the initial phases of the search, a higher value of $\alpha(t)$ fosters wider exploration; conversely, as the process advances, the value of $\alpha(t)$ diminishes to allow for a more concentrated focus on local refinements.

$$\alpha(t) = \alpha_0 e^{-\mu t} \tag{32}$$

where α_0 is the initial (often larger) perturbation coefficient, μ is the decay rate, and t is the current iteration number.

6.5. Fitness update and selection

The fitness update and selection mechanism employ an elitist retention strategy to guarantee that the highest-fitness candidate solution remains intact during updates, thus preserving the optimal solution identified so far. In each iteration, the entire population of individuals is assessed and ranked according to their fitness values. The individual with the highest fitness, denoted as S^* is:

$$S^* = arg\max_i L_i. \tag{33}$$

The fitness of individual S_i is denoted by L_i. The expression S^* signifies the candidate solution that satisfies the maximum number of clauses. The termination condition of the algorithm is adaptive, allowing for a flexible adjustment based on the observed trend in fitness improvement and the iteration count, thereby balancing search efficiency with solution quality. If the fitness value does not significantly improve over k consecutive iterations, the algorithm may terminate early to prevent unnecessary computations. Furthermore, if the maximum iteration T_{max} is reached, the algorithm will stop and return the solution with the highest fitness value.

6.6. Generation of synaptic weights

In the state of the selected optimal neuron S^* (the individual with the highest fitness), we confidently evaluate the satisfaction of each clause based on whether it is a 1-SAT, 2-SAT, or 3-SAT clause. Our objective is to determine the relevant weights for these clauses using the WAN-LC method, as clearly outlined in Section 4.2, which will yield the optimal synaptic weights. These synaptic weights will be efficiently stored in the control memory (CAM) for the retrieval phase, laying a strong foundation for subsequent local-field updates and energy-function optimization.

Pseudo-code of the hybrid firefly algorithm

```
1   Function [$S^*$, W] = HybridFireflyTraining( $nC$, $T_{max}$, $\alpha_0$, $\beta_0$, $\gamma$, $\mu$, $\lambda$)
    % Hybrid Firefly Algorithm for Training Phase in DHNN-RAN3SAT
    % Input: Parameters $nC$ , $T_{max}$ , $\alpha_0$ , $\beta_0$ , $\gamma$ , $\mu$ , $\lambda$ ;
    % Output: Global optimum $S^*$ and corresponding synaptic weight W（Stored
    in control memory CAM）
2   Generate initial neuron states
3   For each candidate solution $S_i$
4   For i=1 to nC
5   Generate $S_i = (v_1, v_2, \cdots, v_{nC})$, $v_i \in \{-1,1\}$ %Each $v_i$ is randomly assigned
    50% probability of being -1 or 1
6   End
7   End
    % Initialize parameters
8   Initialize iteration counter t = 0
9   Initialize global best solution $S^*$
10  While (Maximum fitness has not been reached or $t < T_{max}$)
11  For each $S_i$
12  Calculate fitness using Eq (26) where Eq (27) are defined by logic satisfaction
13  End
14  % Evaluate fitness for each candidate solution
15  For each $S_i$
```

```
Select $S_i$ with the highest fitness from the population and update position using Eq (28) where $r_{ij}$ is calculated using Eqs (29) and (30)
End
% Position update
For each $S_i$
Update neuron state using Eq (31) where $\alpha(t)$ is defined by Eq (32)
End
% Apply random perturbation
Update parameters using Eq (33)
% Dynamic parameter adjustment
Compare all $S_i$ based on fitness $L_i$
Update $S^*$ as the solution with maximum fitness
% Select global best solution
If（$t < T_{max}$）or (fitness has not improved for k iterations)
exit the loop.
Increment t= t +1
% Check convergence
End % end of while
For each logic clause in 1-SAT, 2-SAT, and 3-SAT
Compute synaptic weights using the WAN-LC method
End
% Compute synaptic weights and store synaptic weights in CAM (Content Addressable Memory)
Return $S^*$ (the global best solution) and the corresponding synaptic weights
% Output the results
End of algorithm
```

7. Steps and processes for the realization of the proposed model

This section shows the implementation of the DHNN-RAN3SAT-HFA model in two phases:

Training phase: The hybrid firefly algorithm (HFA) is used to minimize the cost function and obtain the optimal synaptic weights to ensure that the neural network is accurately mapped to the target logical structure.

Retrieval phase: Using the synaptic weights generated in the training phase, the neurons are dynamically updated based on the local field and activated by a sign function to find the global optimal solution.

In order to demonstrate the innovation and performance effect of the proposed model, this paper integrates the implementation steps of RAN3SAT in DHNN, as well as the implementation flow of DHNN-RAN3SAT-HFA, DHNN-RAN3SAT-EA, DHNN-RAN3SAT-ACO, and the traditional DHNN-RAN3SAT model into the flowchart shown in Figure 6. In addition, Figure 7 demonstrates some of the steps of the HFA algorithm in the training phase and the application of weights in the retrieval phase.

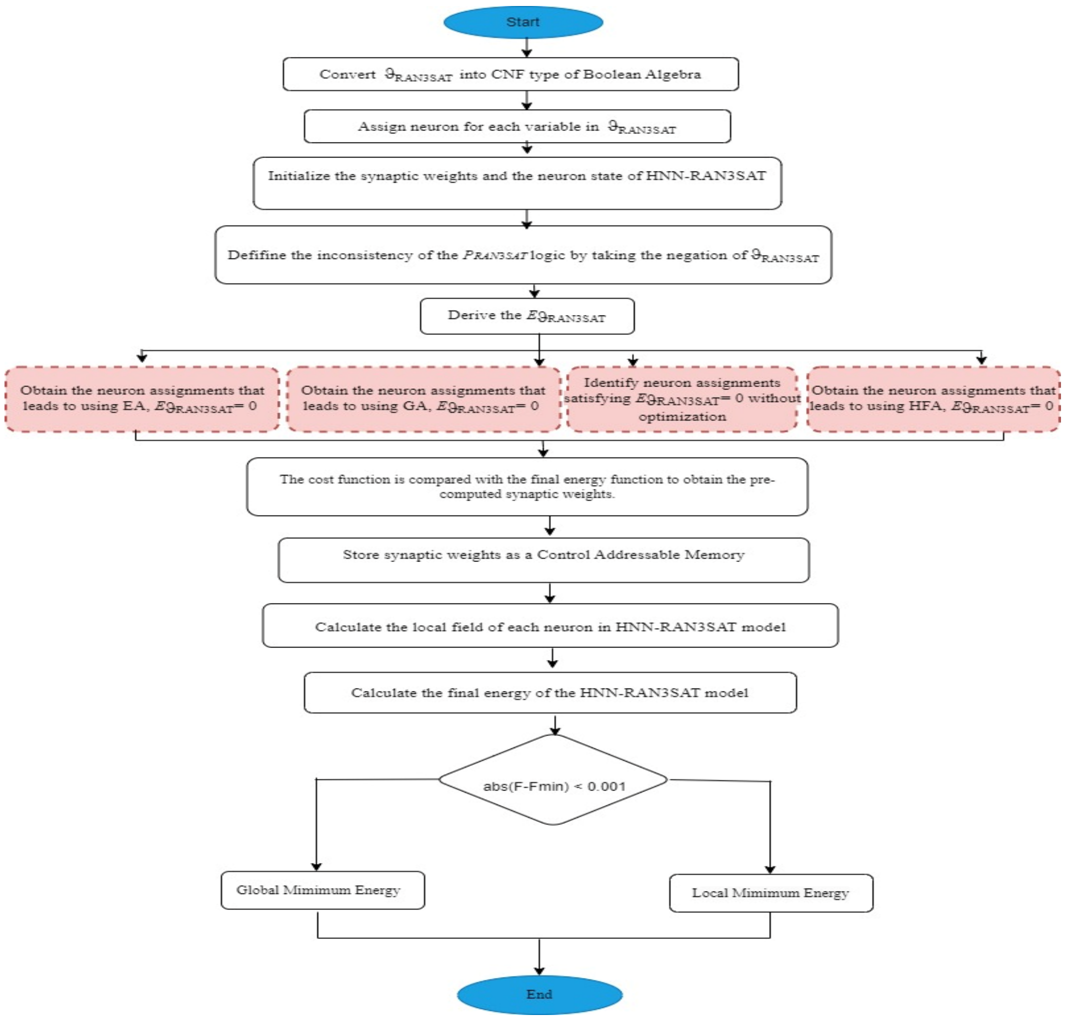

Start
Convert $\vartheta_{RAN3SAT}$ into CNF type of Boolean Algebra
Assign neuron for each variable in $\vartheta_{RAN3SAT}$
Initialize the synaptic weights and the neuron state of HNN-RAN3SAT
Defifine the inconsistency of the $P_{RAN3SAT}$ logic by taking the negation of $\vartheta_{RAN3SAT}$
Derive the $E\vartheta_{RAN3SAT}$
Obtain the neuron assignments that leads to using EA, $E\vartheta_{RAN3SAT}=0$
Obtain the neuron assignments that leads to using GA, $E\vartheta_{RAN3SAT}=0$
Identify neuron assignments satisfying $E\vartheta_{RAN3SAT}=0$ without optimization
Obtain the neuron assignments that leads to using HFA, $E\vartheta_{RAN3SAT}=0$
The cost function is compared with the final energy function to obtain the pre-computed synaptic weights.
Store synaptic weights as a Control Addressable Memory
Calculate the local field of each neuron in HNN-RAN3SAT model
Calculate the final energy of the HNN-RAN3SAT model
abs(F-Fmin) < 0.001
Global Mimimum Energy
Local Mimimum Energy
End

Figure 6. Flowchart of DHNN-RAN3SAT, DHNN-RAN3SAT-EA, DHNN-RAN3SAT-ACO, and DHNN-RAN3SAT-HFA.

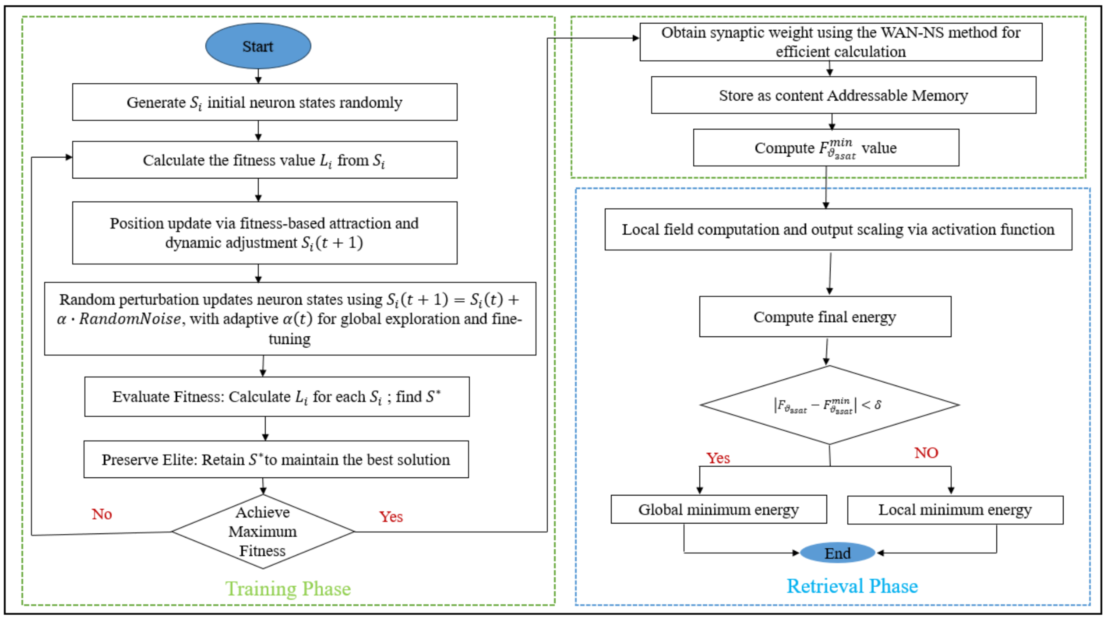

Start
Generate S_i initial neuron states randomly
Calculate the fitness value L_i from S_i
Position update via fitness-based attraction and dynamic adjustment $S_i(t+1)$
Random perturbation updates neuron states using $S_i(t+1) = S_i(t) + \alpha \cdot RandomNoise$, with adaptive $\alpha(t)$ for global exploration and fine-tuning
Evaluate Fitness: Calculate L_i for each S_i ; find S^*
Preserve Elite: Retain $S^*$to maintain the best solution
No
Achieve Maximum Fitness
Yes
Training Phase
Obtain synaptic weight using the WAN-NS method for efficient calculation
Store as content Addressable Memory
Compute $F_{\theta_{3sat}}^{min}$ value
Local field computation and output scaling via activation function
Compute final energy
$|F_{\theta_{3sat}} - F_{\theta_{3sat}}^{min}| < \delta$
Yes
NO
Global minimum energy
Local minimum energy
End
Retrieval Phase

Figure 7. Flowchart of HFA in DHNN-RAN3SAT.

8. Experimental settings

8.1. Overview of the experimental setup

This study's experimental setup draws on a previously proposed approach [5] to validate the performance and effectiveness of the proposed hybrid firefly algorithm on discrete Hopfield neural networks with stochastic 3-satisfiability during the training phase. At the same time, appropriate adjustments were made according to the research objectives and optimization needs to ensure the comprehensiveness and reliability of the experimental environment and the model evaluation. All experiments were conducted in MATLAB R202b on a laptop with 16 GB of RAM and an Intel® Core™ i9 processor running Windows 10. To ensure the reliability of the results, the maximum runtime for each experiment was capped at 24 hours, after which the experiment would be automatically terminated. Unlike the study in [5], the maximum number of iterations in this work was reduced from the original 10,000 to 100. This modification aims to strike a balance between algorithmic performance and computational resource requirements, while also ensuring that the experimental conditions are comparable to the election algorithm in terms of the dataset, the number of neurons, and the distribution of logical clauses, thus maintaining fairness and scientific validity. To further ensure consistency and reliability, all experimental setups, including neuron state initialization, maximum number of iterations, and other key parameters, were kept identical across all models, including HFA and the baseline models (DHNN-RAN3SAT, DHNN-RAN3SAT-ACO, DHNN-RAN3SAT-EA, DHNN-RAN3SAT). The simulated dataset is based on the RAN3SAT logic rule with a structure of $k = 1,2,3$. All experimental data in this paper are randomly generated using bipolar strings $\{1,-1\}$, containing first-order, second-order, and third-order logic clauses with a fixed number of neurons and an equal probability of generation. The total number of neurons is nC, while the number of clauses is $3m$+$2n$+h, where m, n, and h represent the numbers of 3SAT, 2SAT, and 1SAT clauses, respectively. According to the information entropy theory [52], a ratio of 1:3:6 is adopted to optimize both the generation of logical clauses and the coverage of the solution space.

This ratio-based allocation strategy is superior to purely random allocation, as it more effectively covers the solution space, enhances global search capability, and reduces the risk of local optima. Moreover, the ratio can be flexibly adjusted based on practical problem requirements [53], further reinforcing the broad applicability of the proposed method.

Previous studies [6] have shown that the sign function (Sign) performs well in neural network activation and state updating, especially for discrete optimization problems. The concise activation rule $S_i(t+1) = sign(h_i)$ enables the Sign function to efficiently and dynamically evaluate and optimize the network's energy function H, thereby handling complex clause satisfaction challenges. By contrast, the HTAF function proposed in [5] performs adequately in certain scenarios but is biased toward extreme states, which may lead to overfitting [6]. To address this issue, the model directly applies the sign function as the activation rule during the retrieval phase. This not only simplifies the neuron state updates but also significantly improves computational efficiency and logical interpretability. Additionally, by incorporating the optimization mechanism of the hybrid firefly algorithm (HFA), the model generates high-quality synaptic weight matrices in the training phase, thereby offering strong support for performance optimization in the subsequent retrieval phase.

8.2. Experimental parameter configuration

To validate the performance of the proposed discrete Hopfield neural network random 3SAT model based on the hybrid firefly algorithm (DHNN-RAN3SAT-HFA), we compare our experimental results with those of the traditional discrete Hopfield neural network random 3-SAT model (DHNN-RAN3SAT), the discrete Hopfield neural network random 3-SAT model using the ant colony optimization algorithm (DHNN-RAN3SAT-ACO), and the discrete Hopfield neural network random 3-SAT model using the election algorithm (DHNN-RAN3SAT-EA). The experimental parameter configurations are summarized in Tables 4–6.

Table 4. Parameter list for the DHNN-RAN3SAT-HFA model.

Parameter	Parameter value
Neuron combination (NC)	100 [4]
Number of trials (NT)	100 [4]
Maximum number of iterations (T_{max})	100 [4]
Number of experiments (NE)	100 [5]
Tolerance value (δ)	0.001 [5]
Threshold time (H)	24 h [5]
Activation function	Sign [6]
Neuron state initialization–training phase	Random
Neuron state initialization–retrieval phase	Random
Consecutive iterations (k)	50 [8-9]
Number of neurons (nC)	$20 \leq nC \leq 300$ [5]
Total number of clauses (NumC)	$9 \leq \text{NumC} \leq 135$ [52-57]
Initial random perturbation coefficient (α_0)	0.5 [8-9]
Initial attraction strength (β_0)	0.98 [8-9]
Light absorption coefficient (γ)	0.995 [8-9]
Random perturbation decay rate (μ)	0.95 [8-9]
Attraction decay rate (λ)	0.95 [8-9]

Table 5. Parameter list for the DHNN-RAN3SAT model.

Parameter	Parameter value
Neuron combination (NC)	100 [4]
Number of trials (NT)	100 [4]
Maximum number of iterations (T_{max})	100 [4]
Number of experiments (NE)	100 [5]
Tolerance value (δ)	0.001[4]
Threshold time (H)	24 h [5]
Activation function	Sign [6]
Neuron state initialization–training phase	Random
Neuron state initialization–retrieval phase	Random

Table 6. Parameter list for the DHNN-RAN3SAT-ACO, DHNN-RAN3SAT-EA model.

Parameter	Parameter value
Neuron combination	100 [4]
Number of trials	100 [4]
Maximum number of iterations (T_{max})	100 [4]
Number of experiments (NE)	100 [5]
Tolerance value (δ)	0.001 [5]
Threshold time (H)	24 h [5]
Activation function	Sign [6]
Neuron state initialization–training phase	Random
Neuron state initialization–retrieval phase	Random
Size of population (N_{pop})	120 [5]
Number of parties (N_{party})	4 [5]
Positive advertisement rate (σ^p)	0.5 [5]
Negative advertisement rate (σ^n)	0.5 [5]

9. Performance metrics

To thoroughly evaluate the optimization performance of the DHNN-RAN3SAT model in both the training and retrieval phases, this paper identifies appropriate performance metrics aligned with the model's optimization objectives. These metrics encompass critical aspects such as fitness evaluation, error analysis, and solution space coverage, ensuring a well-rounded assessment of the model's global search and local exploitation capabilities.

9.1. Training phase

9.1.1. Maximum fitness

Maximum fitness indicates the total count of unique solutions generated during the model's training phase that fulfill the maximum number of logical clauses. This metric assesses the model's search capabilities and its extent of coverage across the solution space during optimization. According to [6], it is calculated as follows:

$$F_{max} = \sum_{i=1}^{N_{iter}} \left(f_i - f_i^{\circ}\right). \tag{34}$$

The term F_{max} represents the value of the maximum fitness, reflecting the total count of unique maximum-fitness solutions identified across all iterations. The variable f_i indicates the number of maximum-fitness solutions found in iteration i, including duplicates. Additionally, f_i° denotes the number of duplicate solutions discovered in iteration i. The notation N_{iter} refers to the total number of iterations.

During each iteration, the model assesses neuron states in accordance with the fitness function f_{nC} and calculates the number of satisfied clauses. When the count of satisfied clauses reaches its maximum, the number of unique solutions is recorded. To enable comparisons under varying experimental conditions, F_{max} is normalized using the following approach:

$$F_{norm} = \frac{F_{exp} - F_{min}}{F_{max} - F_{min}}. \tag{35}$$

Here, F_{norm} is the normalized maximum fitness, and F_{exp} is the maximum fitness in the current experiment. This normalization effectively eliminates differences in the range of maximum-fitness values caused by different experimental settings, making the results more comparable.

9.1.2. Fitness ratio

The fitness ratio measures the proportion of maximum-fitness solutions retrieved by the model in the training phase, i.e., the effective search coverage relative to the theoretical maximum of the solution space. It is calculated as:

$$R = \frac{F_{max}}{F} \tag{36}$$

where R is the fitness ratio, indicating the fraction of maximum-fitness (non-repetitive) solutions out of the total solution space. F_{max} is the total number of maximum-fitness solutions, and F is the total number of neuron-state combinations. This formula evaluates the model's search efficiency and how well it covers the solution space.

9.1.3. Diversity metrics

To quantify the model's performance in solution space coverage and search capability, this paper employs information entropy [52] as the theoretical basis for diversity metrics. Information entropy measures system uncertainty and logical complexity, making it well-suited for evaluating the distribution and complexity of 1-SAT, 2-SAT, and 3-SAT clauses in combinational logic structures [54,55]. Its introduction provides a scientific basis for allocating clause proportions and ensures the statistical reliability of the model's search strategy [53]. Drawing on existing diversity-assessment methods and the theoretical study of multi-objective optimization algorithms [56,57], we optimize our diversity metrics in three ways:

- Measuring complexity and uncertainty of logical clauses: Entropy is calculated from probability distributions to quantify the complexity of different clauses.
- Providing a theoretical basis for clause-ratio settings: Entropy guides the allocation of logical clause proportions, ensuring a more scientific approach rather than relying on subjective assumptions.
- Quantifying global search capability via diversity: By examining entropy values, the model's

ability to balance global search and local exploitation can be evaluated, reflecting how extensively the solution space is covered.

To compute the complexity of logical clause combinations, information entropy is used to correct the clause allocation ratio [53]:

$$H = -\sum p_i \log_2(p_i). \tag{37}$$

Here, p_i is the generation probability of each clause type, based on the number of feasible combinations that satisfy the logical constraints. Table 7 below shows an example of deriving the number of satisfied combinations and the entropy values.

Table 7. The number of satisfactory combinations corresponds to the entropy value.

Clause type	Possible combinations	Satisfied combinations	Probability distribution p_i	Entropy H
1SAT	2	1	$\frac{1}{2}$	$H_{1SAT} = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)$
2SAT	2^2	3	$\frac{3}{4}$	$H_{2SAT} = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right)$
3SAT	2^3	7	$\frac{7}{8}$	$H_{3SAT} = -\left(\frac{7}{8}\log_2\frac{7}{8} + \frac{1}{8}\log_2\frac{1}{8}\right)$

From Table 7, the total number of satisfied combinations is 11, yielding proportions:

$$p_{1SAT} = \frac{1}{11}, p_{2SAT} = \frac{3}{11}, p_{3SAT} = \frac{7}{11}. \tag{38}$$

Following the multi-objective optimization ratio-correction method proposed in [53,56,57] and using a 70:30 weighting strategy, the clause distribution ratio can be adjusted to:

$$\text{1SAT:2SAT:3SAT} = 1:3:6. \tag{39}$$

Integrating the logical satisfiability ratio and entropy-complexity measure ensures an optimal balance between model accuracy and computational efficiency. After introducing entropy-based corrections, the diversity metric is:

$$D_{max} = \xi(3m + 2n + h), \tag{40}$$

where, according to [25,58], ξ=0.4 is the total diversity-adjustment factor, and m, n, and h are the counts of 3SAT, 2SAT, and 1SAT clauses, respectively. By combining clause-proportion allocation with entropy-based corrections, this metric accurately reflects the complexity of logical combinations and solution space coverage, bolstering the model's global search ability and multi-objective optimization performance.

9.1.4. Weight mean absolute error

In discrete Hopfield neural networks, the initial neuron states are randomly generated. To evaluate how effectively the model optimizes the salient (synaptic) weights during the training phase, we employ the mean absolute error (MAE) as a performance metric. MAE effectively measures the difference between the randomly initialized weights and their post-training values, reflecting the model's learning and optimization capacity. The WAN-NS method is directly used for weight calculation. The error from initial random weights to the final training results is computed as:

$$MAE_{weight} = \frac{1}{N_{total}} \sum_{i=1}^{N_{total}} \left| W_i^{optimal} - W_i^{trained} \right|, \tag{41}$$

where $W_i^{optimal}$ is the i-th target weight, $W_i^{trained}$ is the actual weight obtained through training, and N_{total} is the total number of synaptic weights (determined by the number of logical clauses and connected neurons). Compared to other metrics, MAE is robust and intuitive for measuring average errors, thereby more reliably reflecting the model's optimization effectiveness. It also avoids the high sensitivity to large-error terms seen in RMSE, making MAE widely adopted in model performance evaluations [59].

9.2. Retrieval phase

9.2.1. Global minima ratio

In the retrieval phase of a discrete Hopfield neural network (DHNN), the final neuron state typically converges to a global minimum-energy state, which corresponds to a zero-cost solution under certain logical constraints [6,60]. When the network satisfies the logical rules, its performance can be measured by the global minima ratio (GMR), which integrates the computational approach in [25] with the logic-rule embedding from [6] and extends the application of information entropy to further assess solution diversity. GMR thus evaluates the network's ability to locate global minimum-energy solutions under logical constraints, reflecting global search capability, convergence performance, and logic-rule satisfaction [6,60]. It is calculated as:

$$GMR = \frac{1}{NC*NT} \sum_{i=1}^{NC*NT} N_H^{logic}(i), \tag{42}$$

where NT is the total number of experiments, NC is the number of neuron combinations, and $N_H^{logic}(i)$ denotes—during the i-th experiment—the count of solutions satisfying Eq (1) [see also Eq (12)] that achieve global minimum energy while respecting the information-entropy ratio.

9.2.2. Mean absolute error of energy

The mean absolute error of energy (MAE) measures the gap between the final energy value computed by the model and the theoretical global minimum energy, indicating the model's convergence accuracy and error behavior [6]. This metric effectively assesses retrieval-phase performance by comparing the actual output against its theoretical expectation, thereby verifying the model's robustness and reliability:

$$MAE_{energy} = \frac{1}{NC*NT} \sum_{i=1}^{NC*NT} \left| E_{optimal}^{(i)} - E_{final}^{(i)} \right|, \tag{43}$$

where $E_{optimal}^{(i)}$ is the theoretical minimum energy for the i-th experiment [computed via Eq (11)] and $E_{optimal}^{(i)}$ is the final energy obtained by the i-th experiment. $NC * NT$ denotes the total number of experiments.

9.2.3. Comprehensive similarity metrics

To comprehensively evaluate solution diversity and the discovery of global optima in the retrieval phase, this paper adopts a similarity index (SI) as an assessment criterion, following the designs of multiple metrics in [4,6,25]. The SI reflects the global search capability of the model and the degree of coverage within the solution space.

9.2.3.1. Similarity index

To quantify the similarity between the neuron states generated by the model and the ideal solution states, we employ two similarity indices. The first, the Jaccard similarity index (JI) [21], measures the overlap between the model-generated neuron states and the ideal states:

$$JI = \frac{\text{ÅÅ}}{\text{ÅÅ}+\mathcal{B}\text{Å}+\text{Å}\mathcal{B}}. \tag{44}$$

The Gower–Legendre similarity index (GLI) [61] further accounts for negatively correlated solutions and emphasizes consistency in positive and negative clauses:

$$GLI = \frac{\text{ÅÅ}+\mathcal{B}\mathcal{B}}{\text{ÅÅ}+0.5(\mathcal{B}\text{Å}+\text{Å}\mathcal{B})+\mathcal{B}\mathcal{B}}. \tag{45}$$

The similarity metrics are defined as follows (as shown in Table 8).

Table 8. Benchmark information in the similarity index.

Parameter	S_i^{ideal}	S_i
ÅÅ	1	1
$\mathcal{B}$Å	-1	1
Å$\mathcal{B}$	1	-1
$\mathcal{B}\mathcal{B}$	-1	1

9.2.3.2. Neuron variation rate (NV): theoretical support metric

To more accurately interpret the diversity and global search capability of the model in the retrieval phase, we introduce the neuron variation (NV). NV measures the diversity and frequency of state changes among neurons across different experimental settings:

$$NV = \lambda \sum_{V=1}^{N} G_V, \tag{46}$$

where λ is the total number of neuron-state combinations generated by the model, and

$$G_{V+1} = \begin{cases} 1, & if\ x_{i+1} \neq x_i \\ 0, & if\ x_{i+1} = x_i \end{cases}. \tag{47}$$

A higher NV indicates stronger exploration capability, helping explain similarity metrics (e.g., the Jaccard index or Gower–Legendre index). However, NV is not used as a standalone performance measure, and its results are not displayed separately.

9.2.4. Runtime

The total runtime of both training and retrieval phases is:

$$Runtime = T_{train} + T_{retrieve} \tag{48}$$

where T_{train} denotes the training duration and $T_{retrieve}$ denotes the retrieval duration. A shorter total runtime implies higher overall efficiency in model optimization.

9.2.5. State coverage (diversity rate)

The diversity rate (DR) measures how extensively the model covers the solution space during retrieval, expressed as the ratio of distinct neuron states generated to the theoretical target states:

$$DR = \frac{\nu_A}{\nu_T} * 100\% \tag{49}$$

where ν_A is the number of unique neuron states actually generated in the retrieval phase, and ν_T is the theoretically possible number of target neuron states. According to [25], when DR ≥ 40%, the model is considered to have sufficient diversity and global search capability to cover enough target states, ensuring comprehensive exploration. In this study, since diversity distribution is already computed based on entropy during training and the 1:3:6 ratio is adopted for the clauses to optimize solution space coverage, we do not recalculate the state coverage in the retrieval phase. Instead, DR serves as an indirect supportive metric to verify the model's global search ability and solution space coverage under varying neuron counts.

10. Experimental results and analysis

To comprehensively evaluate the performance of the hybrid firefly algorithm (HFA) across varying neuron counts, we designed a series of experiments to analyze how different neuron quantities affect the model's optimization capability, solution space coverage, and computational efficiency. The experimental evaluation consists of two phases—training and retrieval—to thoroughly assess the model's performance across different aspects. In the training phase, we focus on optimizing the generation of initial neuron states and maximizing the search capabilities for solutions with maximum fitness. This phase specifically monitors the number of non-repeated maximum fitness solutions in each iteration while analyzing the distribution and complexity of logical clauses to ensure broader solution space coverage. The retrieval phase builds upon the training phase by conducting further global searches to validate the model's convergence properties and solution space coverage. This phase leverages the optimized weights and neuron states generated during training to evaluate the model's ability to locate global optimal solutions efficiently. By integrating the experimental designs of both phases, this study compared the performance of three models (the traditional DHNN-RAN3SAT model, the DHNN-RAN3SAT-EA model, and the DHNN-RAN3SAT-ACO model) with the proposed DHNN-RAN3SAT-HFA model to ensure a comprehensive performance evaluation.

10.1. Training phase

During the training phase, the model randomly initializes neuron states and employs a fitness function to compute the number of satisfied clauses, thereby enhancing the search for the solution with the highest fitness. This stage leverages information-entropy calculations and a 1:3:6 clause-ratio allocation strategy [52], in which 1-SAT, 2-SAT, and 3-SAT clauses are generated and assigned in a 1:3:6 proportion, respectively. This allocation ensures both statistical reliability and a diverse distribution of clauses, mitigating restricted exploration and local optima risk while bolstering the global search capability. Although state coverage is not directly computed in the training phase, its diversity performance can be indirectly evaluated via several metrics [25], as follows:

Fitness ratio: Assesses the model's coverage of the solution space, indicating whether a high ratio truly reflects comprehensive exploration instead of getting stuck in local regions.

Maximum fitness: Examines the distribution rationality of maximum-fitness solutions and whether they genuinely capture global search ability.

Mean absolute error (MAE) of the weights: Interpreted through diversity theory; lower MAE values reflect stable global performance during training.

In this phase, WAN-LC is used to directly compute the weights, combined with proportion-based clause allocations to strengthen the model's solution-space exploration and overall performance. By introducing information-entropy calculations and a 1:3:6 ratio adjustment, the model balances clause complexity with distribution diversity, ultimately enhancing coverage of the solution space.

10.1.1. Maximum fitness

Figure 8 plots the normalized maximum fitness values for four DHNN-RAN3SAT models across neuron counts ranging from 20 to 300. Overall, all algorithms show declining fitness as the neuron count increases, reflecting the exponential expansion of the solution space and the increasing difficulty in identifying optimal neuron states. At small scales ($nC \leq 50$), DHNN-RAN3SAT-HFA achieves the highest fitness (0.6892), benefiting from its brightness-attraction mechanism and dynamic weight adjustment, which enable efficient exploration. EA also performs reasonably well (fitness 0.4956), owing to its simple yet effective global selection strategy. In contrast, ACO starts lower (0.336), limited by its slow pheromone convergence and high computational cost, which are less suited for rapidly navigating discrete Hopfield networks. The baseline DHNN-RAN3SAT performs the weakest (0.0789), lacking sufficient diversity mechanisms to escape poor local optima. At medium scales ($50 < nC \leq 150$), HFA maintains superior performance (~0.4), thanks to its adaptive control strategies that sustain exploration. EA holds around 0.3, but its fixed local strategy shows signs of stagnation. ACO (~0.2), while slightly better than the baseline, declines steadily, hindered by its sequential solution-building process and inefficiency in discrete logical representations. DHNN-RAN3SAT continues to drop sharply, underscoring its limited scalability. At large scales ($nC > 150$), HFA demonstrates strong resilience, retaining 0.2568 at 300 neurons. EA drops to 0.1095, revealing its limitations in high-dimensional search. ACO converges to 0.0823, confirming its unsuitability for large-scale discrete optimization due to pheromone update delays and poor adaptation to non-path-based problems like random satisfiability. The original DHNN-RAN3SAT model effectively fails at this stage, with fitness near zero.

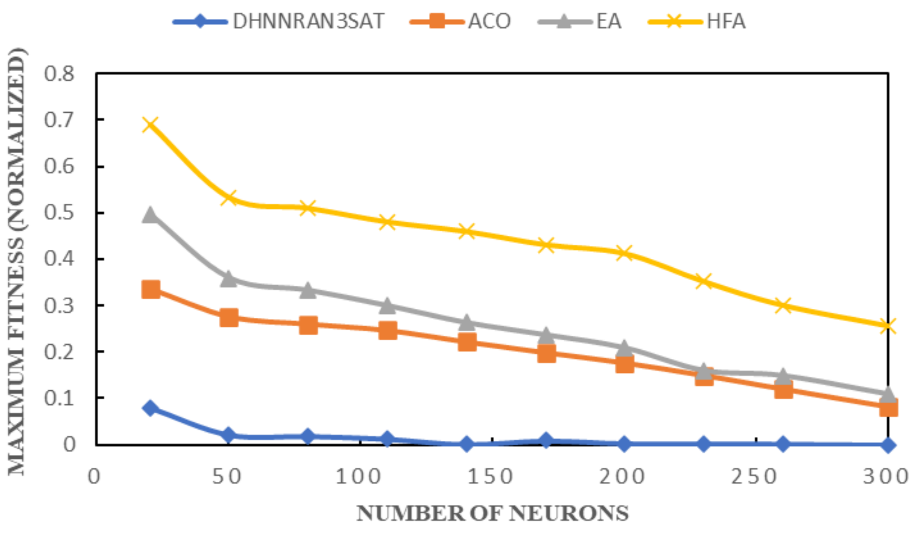


Figure 8. Maximum fitness for different DHNN-RAN3SAT models.

In sum, DHNN-RAN3SAT-HFA delivers the best performance across all neuron counts, enabled by its adaptive brightness mechanisms and diversity control, sustaining higher fitness even in large-scale settings. DHNN-RAN3SAT-EA remains competitive in smaller networks but loses effectiveness as problem size grows due to its limited local search strategy. DHNN-RAN3SAT-ACO, though theoretically promising, performs worse than EA and HFA in all settings. Its pheromone-based search suffers from high computational complexity, slow convergence, and an inability to represent discrete, non-path-based solution spaces like those in Hopfield networks. These drawbacks lead to early stagnation and poor scalability in random k-SAT optimization. Meanwhile, the base DHNN-RAN3SAT model shows the sharpest decline, lacking any effective heuristic mechanism to cope with increasing problem complexity, making it unsuitable for large-scale satisfiability challenges.

10.1.2. Fitness ratio

Figure 9 illustrates the variation in fitness ratio for four algorithms—DHNN-RAN3SAT, DHNN-RAN3SAT-EA, DHNN-RAN3SAT-ACO, and DHNN-RAN3SAT-HFA—as the neuron count increases from 20 to 300. The fitness ratio reflects each algorithm's effectiveness in locating near-optimal solutions relative to the theoretical best across different problem sizes. In small-scale problems ($nC \leq 50$), DHNN-RAN3SAT-HFA achieves the highest fitness ratio (0.9654), benefiting from brightness attraction, stochastic perturbation, and entropy-driven diversity control (based on a 1:3:6 clause ratio). These mechanisms promote efficient exploration and help avoid local optima. DHNN-RAN3SAT-EA performs reasonably well (0.8856), owing to partition-based search and solution refinement. However, its lack of global clause diversity limits further improvement. DHNN-RAN3SAT-ACO starts at a moderate fitness (~0.75), but its performance is hindered by pheromone-based path construction, which is less effective in discrete state spaces. The algorithm's high computational cost and slow pheromone convergence reduce its responsiveness in small-scale Hopfield networks. Baseline DHNN-RAN3SAT shows the lowest fitness (0.19), as expected due to the absence of adaptive or exploratory strategies. In medium-scale problems ($50 < nC \leq 150$), HFA retains strong performance (~0.7), sustained by adaptive diversity and entropy mechanisms that ensure robust clause distribution. EA drops to ~0.6 as the lack of dynamic global search affects exploration. ACO shows a continued decline (~0.45), attributed to its limited adaptability and incompatibility with discrete solution spaces, where no clear path sequence exists. DHNN-RAN3SAT drops below 0.1, reflecting poor coverage of the solution space. In large-scale problems ($nC > 150$), HFA remains the most robust (~0.4), preserving solution diversity through adaptive brightness and stochastic controls. EA declines (~0.2), constrained by its fixed local search strategy. ACO converges to ~0.1, further exposing its inefficiency in high-dimensional discrete landscapes, where pheromone trails become unstable and misleading. DHNN-RAN3SAT effectively collapses (fitness approximates 0), confirming its inability to scale.

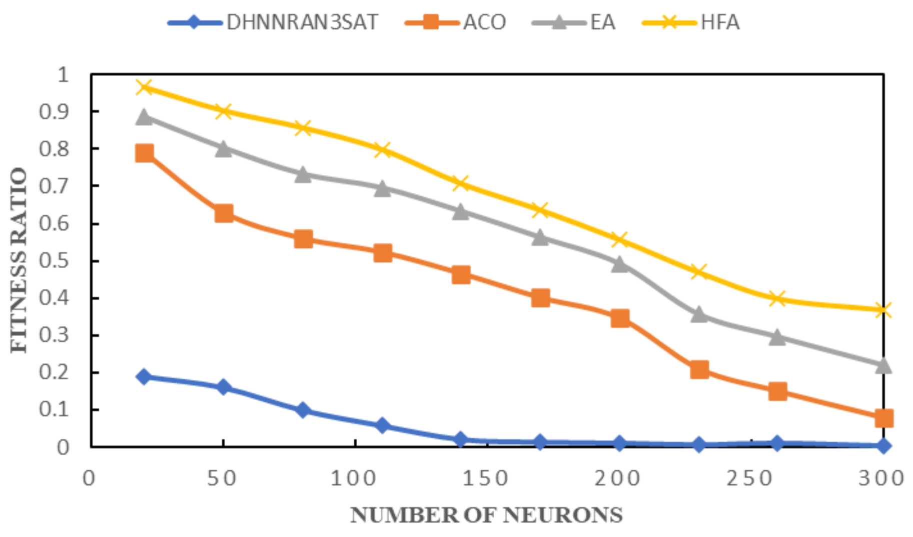


Figure 9. Fitness ratio for different DHNN-RAN3SAT models.

In summary, DHNN-RAN3SAT-HFA outperforms all other models across problem scales, owing to its brightness attraction, adaptive perturbation, and clause-based diversity mechanisms. These features ensure consistent exploration and balanced solution space coverage. DHNN-RAN3SAT-EA performs well for small and medium-sized problems but declines under larger-scale tasks due to the absence of global clause diversity. DHNN-RAN3SAT-ACO, despite its theoretical potential, underperforms because its pheromone mechanism is ill-suited for discrete, non-sequential Hopfield network states. The algorithm also suffers from high computational complexity and slow convergence, and lacks a well-defined fitness heuristic for random SAT representations. Finally, DHNN-RAN3SAT maintains a consistently low fitness ratio, confirming that traditional static strategies are insufficient for solving complex, high-dimensional k-SAT problems.

10.1.3. Mean absolute error of the weights

Figure 10 depicts the variation in MAE weight across four DHNN-RAN3SAT models as neuron count increases. The MAE reflects the models' effectiveness in adjusting synaptic weights from random initialization to convergence, directly indicating their optimization capability. For small-scale problems ($nC \leq 50$), HFA achieves the lowest MAE (~0), leveraging brightness attraction and entropy-driven diversity (via a 1:3:6 clause ratio) to refine weights with precision. EA shows competitive results (MAE ≈ 1), benefiting from partitioned search and moderate diversity, though lacking adaptive stochastic tuning. ACO begins with moderate performance (MAE ≈ 1–3) but is hindered by its pheromone mechanism, which is not well-suited for the discrete states of Hopfield networks, and lacks real-time weight adjustment capacity. DHNN-RAN3SAT shows poor performance (MAE < 7) due to its static update strategy and lack of clause-space coverage.

In medium-scale problems ($50 < nC \leq 150$), HFA maintains low MAE (<1), with its dynamic adjustment strategies ensuring balance between exploration and exploitation. EA rises moderately (MAE 1–5), remaining relatively stable. ACO shows a noticeable MAE increase (~3–23), as its path-construction mechanism fails to scale, and its search becomes inefficient in larger, non-sequential solution spaces. DHNN-RAN3SAT continues deteriorating (MAE ~7–35), lacking any compensatory mechanism.

In large-scale problems ($nC > 150$), HFA continues to perform best, with MAE remaining below 2, thanks to adaptive clause-driven tuning and entropy-controlled exploration. EA rises significantly (MAE ~6–13), as its fixed search structure struggles with large-scale complexity. ACO climbs rapidly (MAE > 23), due to slow convergence, high computational cost, and ill-defined fitness heuristics for weight adjustment in discrete Hopfield contexts. DHNN-RAN3SAT fails to converge (MAE > 37), confirming poor scalability.

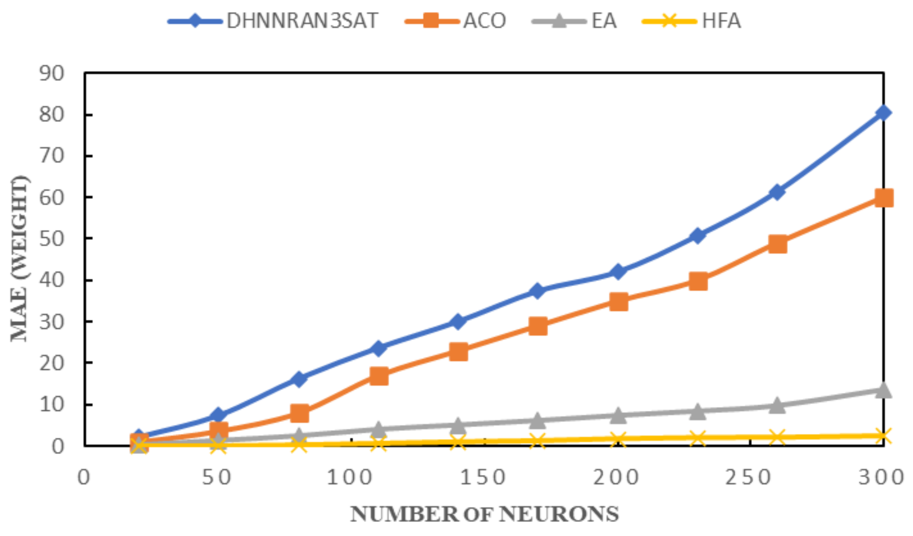


Figure 10. MAE weight for different DHNN-RAN3SAT models.

In summary, DHNN-RAN3SAT-HFA demonstrates the lowest MAE across all problem sizes, supported by adaptive brightness attraction, stochastic perturbation, and clause-based entropy diversity. These mechanisms enable efficient weight optimization even in large-scale networks. DHNN-RAN3SAT-EA performs adequately in smaller settings, but lacks the adaptive components needed for large-scale generalization. DHNN-RAN3SAT-ACO, although conceptually capable of global search, performs poorly due to incompatibility between pheromone-based dynamics and discrete random weight states. It suffers from local stagnation and high computational burden and lacks clear heuristic fitness modeling for weight adjustment, making it unsuitable for Hopfield-based SAT weight learning. Finally, DHNN-RAN3SAT fails to optimize across all scales due to its fixed, non-adaptive structure, highlighting the need for intelligent mechanisms such as diversity control and global search strategies in random k-satisfiability learning within discrete Hopfield networks.

10.1.4. Convergence analysis

Convergence analysis is a critical technique for assessing the effectiveness and performance of optimization algorithms during the training phase. It also serves as a fundamental metric for evaluating the maximum fitness, the adaptation ratio, and the mean absolute error (MAE) of synaptic weights throughout the training process. In this study, the vertical axis in the convergence graphs represents the deviation between the maximum number of satisfiable clauses (i.e., the maximum fitness value) and the theoretical optimal fitness value. The horizontal axis corresponds to the number of iterations. Ideally, under optimal optimization conditions, as the number of iterations increases, the error in the (k+1)-th iteration should be lower than in the k-th iteration, indicating that the algorithm is progressively converging toward the optimal solution. Figures 11–20 illustrate the progression of the maximum fitness error over multiple iterations for different algorithms applied to varying neuron counts (nC = 20 to nC = 300). These graphs reveal the impact of neuron count on the convergence speed and overall effectiveness of the algorithms.

HFA: For neuron counts below 80, the HFA algorithm consistently reaches the optimal solution in a single iteration. When the neuron count exceeds 80, although the algorithm requires additional iterations, its convergence speed remains significantly faster than the other algorithms. It effectively balances global exploration and local exploitation, contributing to superior optimization performance.

EA: In smaller neural networks (nC = 20, 50), the EA algorithm demonstrates faster error reduction and favorable convergence performance. However, as the neuron count increases (nC = 200 and above), the algorithm struggles to reduce the fitness error significantly, often leading to local optima stagnation. The DHNN-RAN3SAT algorithm shows the slowest convergence rate across all experiments. Particularly in larger neuron networks ($nC \geq 100$), the gap between the maximum fitness and the current fitness error becomes more pronounced, making it difficult to achieve the optimal solution within a limited number of iterations.

ACO: Shows slower and less stable convergence across all scales. Its pheromone-based mechanism is poorly suited to the random-state structure of Hopfield networks, resulting in inefficient weight updates. ACO is also prone to premature convergence, lacks strong global escape mechanisms, and involves high computational overhead. These limitations prevent it from keeping pace with EA or HFA, especially as nC increases.

Convergence speed: HFA demonstrates the fastest convergence rate across all neuron counts and often reaches the optimal solution within a single iteration for smaller networks. EA performs well in small-scale networks but experiences significant convergence lag as the network size increases. ACO converges more slowly and inconsistently, especially in large-scale networks, due to its dependence on pheromone trail accumulation, which is not well suited for discrete-state dynamics. DHNN-RAN3SAT consistently shows the slowest convergence speed and struggles to reduce error in larger neuron networks.

Global and local search balance: HFA maintains an effective balance between global exploration and local exploitation, enabling it to reach near-optimal solutions even in more complex network scenarios (e.g., nC = 300). In contrast, EA and DHNN-RAN3SAT face challenges with local stagnation, especially in larger neural networks, which prevents further optimization progress. ACO also lacks a strong global escape mechanism and tends to prematurely converge, particularly when neuron counts are high. Additionally, its pheromone-based guidance becomes ineffective in complex random-state landscapes, further limiting its optimization performance.

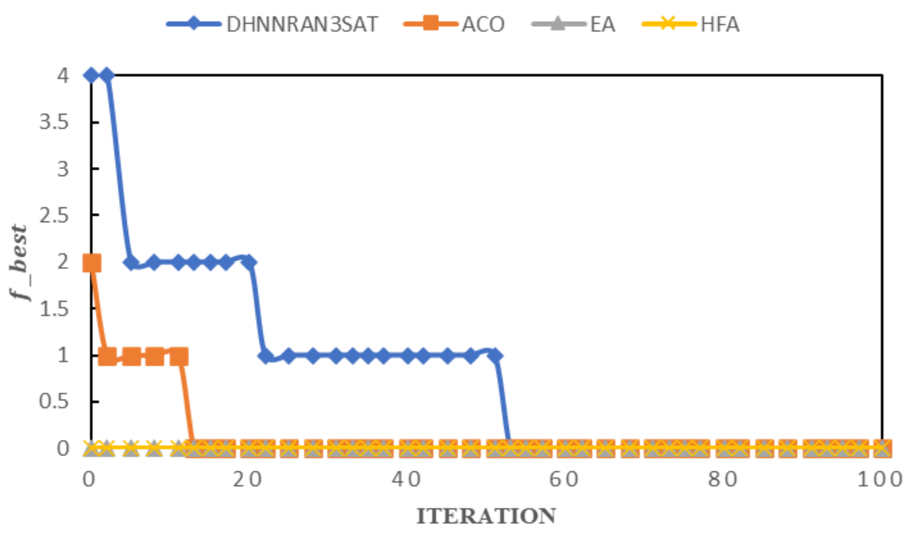


Figure 11. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 20) in DHNN-RANkSAT models.

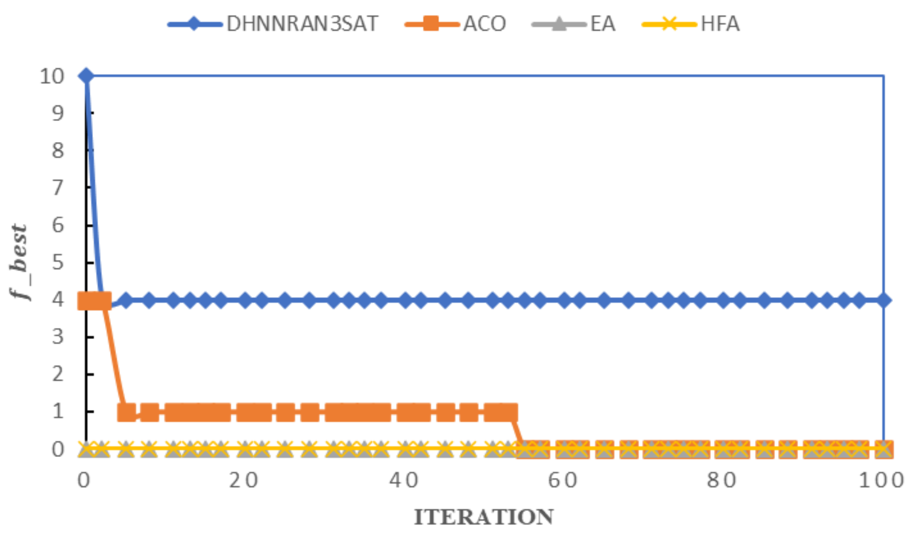


Figure 12. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 50) in DHNN-RANkSAT models.

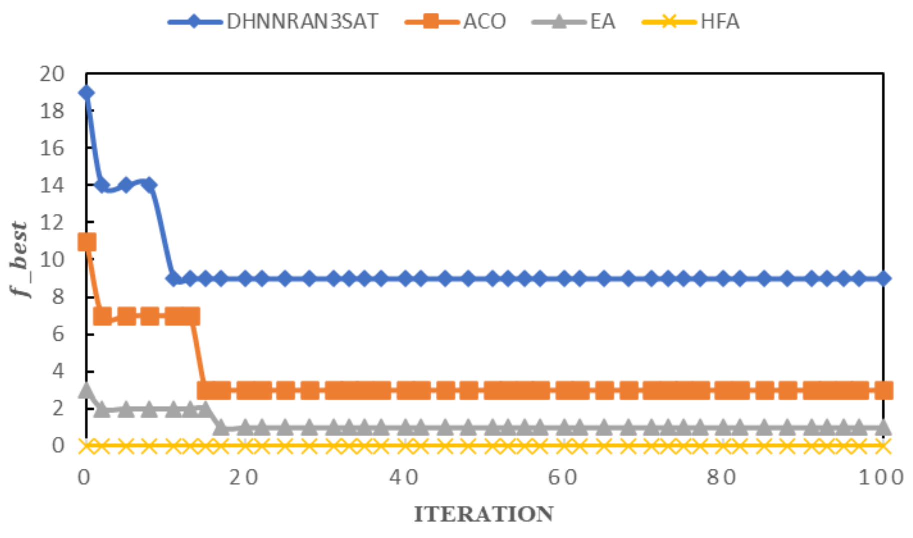


Figure 13. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 80) in DHNN-RANkSAT models.

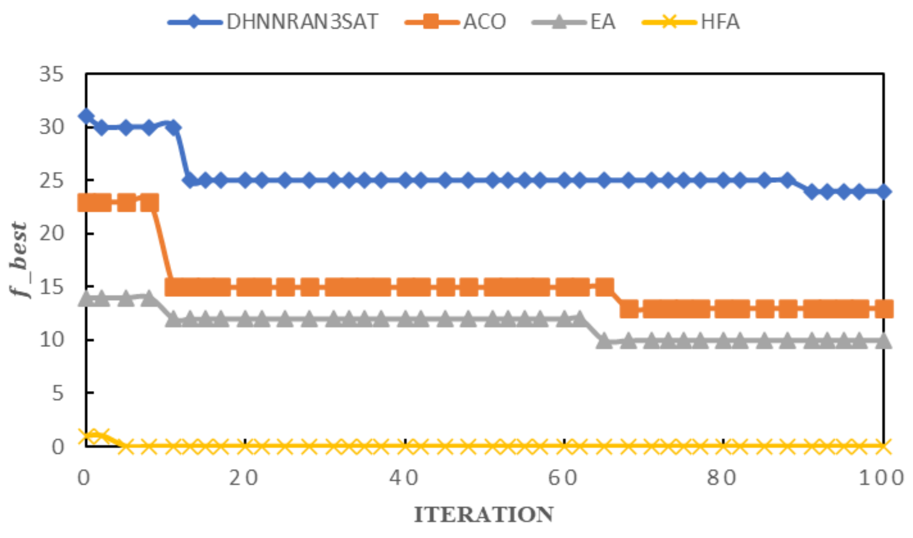


Figure 14. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 110) in DHNN-RANkSAT models.

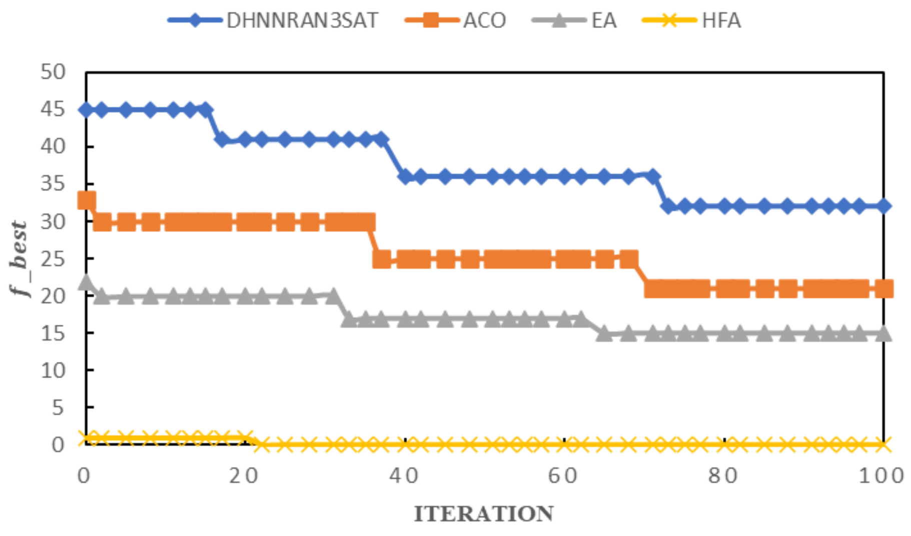


Figure 15. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 140) in DHNN-RANkSAT models.

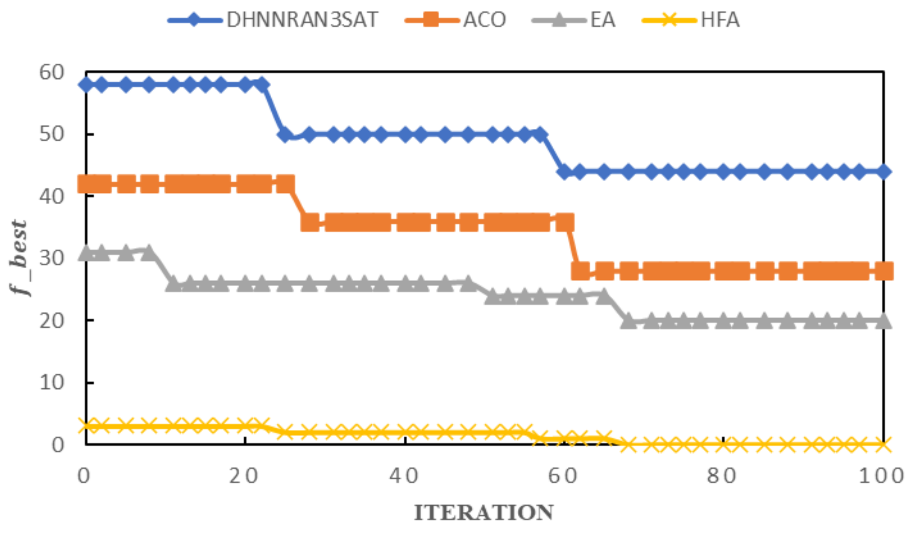


Figure 16. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 170) in DHNN-RANkSAT models.

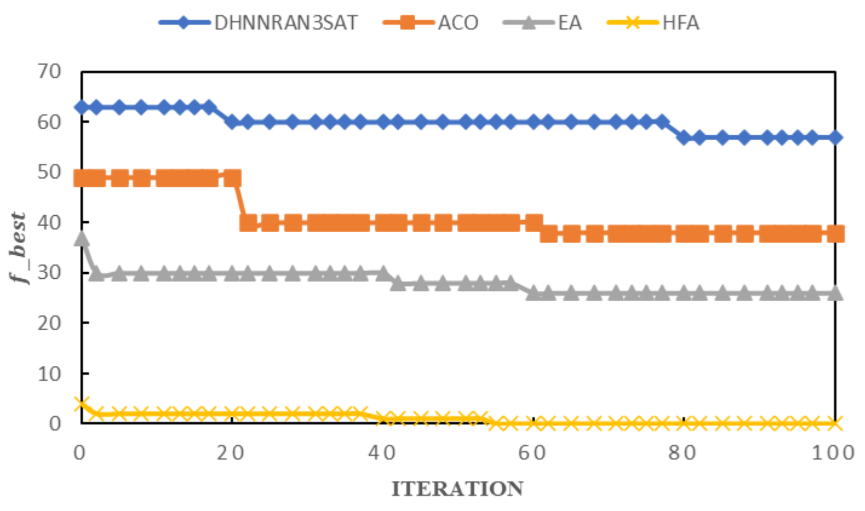


Figure 17. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 200) in DHNN-RANkSAT models.

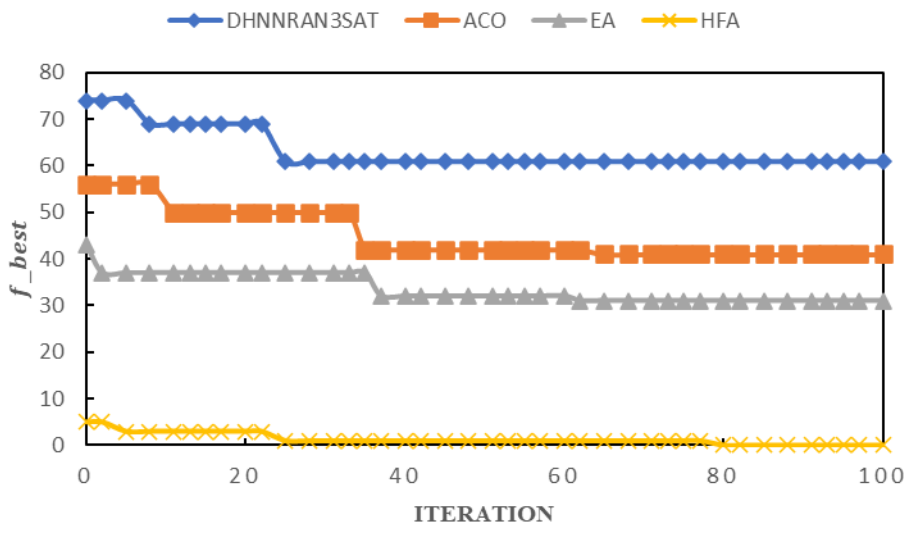


Figure 18. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 230) in DHNN-RANkSAT models.

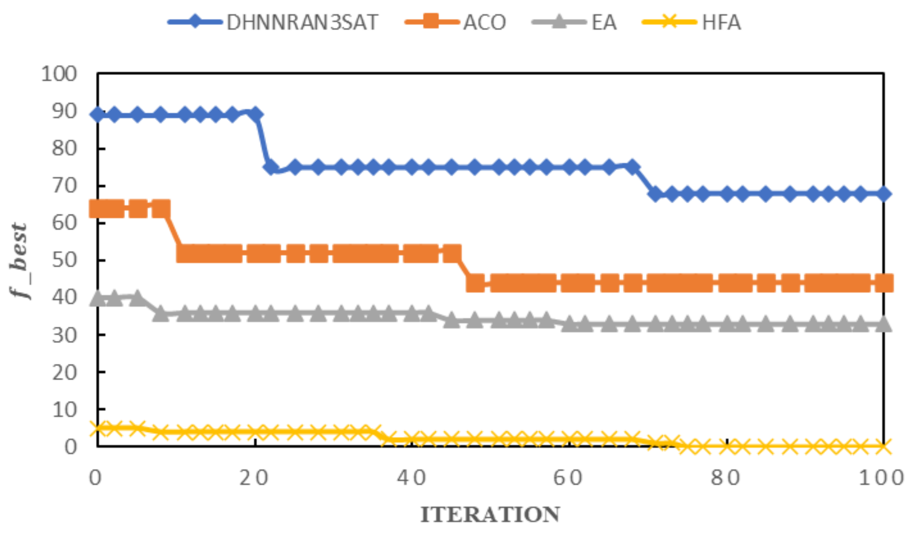


Figure 19. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 260) in DHNN-RANkSAT models.

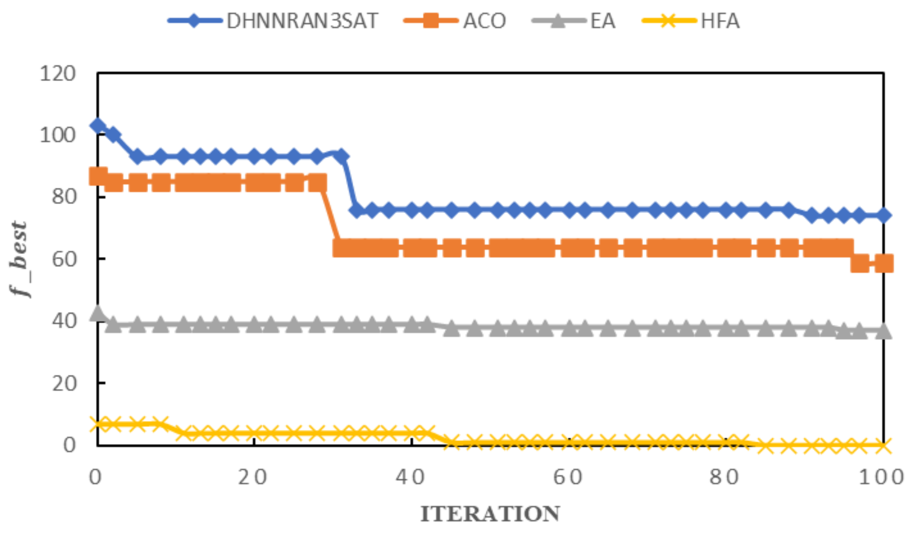


Figure 20. Convergence curve of maximum fitness error for neuron counts and clause sizes (nC = 300) in DHNN-RANkSAT models.

Among the tested algorithms, HFA consistently outperforms EA, ACO, and DHNN-RAN3SAT across most cases, particularly in large-scale neuron networks. Its superior convergence speed and effective balance between global and local search make it a suitable candidate for global optimization tasks in complex networks.

10.2. Retrieval phase

During the retrieval phase, the core objectives are to evaluate the model's global search capability and convergence performance and to validate the effectiveness of the optimizations achieved during training. To comprehensively assess retrieval outcomes, this study employs several key metrics. The global minimum ratio measures the frequency with which the model identifies a global minimum energy solution; the mean absolute error gauges the discrepancy between computed results and the theoretical optimum; the similarity index reflects how closely generated solutions match the ideal target state; and the runtime analyzes the computational efficiency of the model. Moreover, to further investigate the model's diversity and solution space coverage during retrieval, we introduce diversity rate (DR) and neuron variation as indirect supportive indicators. According to [25], when DR reaches or surpasses 40%, the model is considered to have sufficient diversity and global search strength to explore the solution space more thoroughly, thereby supporting the resolution of more complex problems.

10.2.1. Global minimum ratio

Global minimum ratio (GMR) serves as a key performance indicator to assess whether the neuron-state combinations, under the optimized weights generated in the training phase, can converge to a global minimum energy state during retrieval. Figure 21 shows that, for DHNN-RAN3SAT-HFA, the GMR remains near 1 across all neuron scales in logical combinations containing 1-SAT, 2-SAT, and 3-SAT clauses, demonstrating exceptional robustness. This superior performance mainly stems from the brightness attraction mechanism, which produces an adaptive weight matrix in the training phase. During retrieval, the network uses the sign function to dynamically update local fields, effectively guiding randomly generated neuron states to converge rapidly on the global optimum. In retrieval, the sign function recalculates and adjusts neuron states according to local-field values to determine whether the network has reached or closely approached a stable or globally optimal state. Additionally, higher neuron variation (NV) and diversity rate (DR) further reinforce the algorithm's effectiveness. NV indicates the extent of neuron-state fluctuations; its higher value fosters a broader exploration of the solution space, thereby boosting GMR. DR measures how extensively the solution space is covered; when DR $\geq$ 40%, the network's exploration becomes sufficiently comprehensive to more quickly approach the global optimum, further stabilizing the GMR. By contrast, DHNN-RAN3SAT-EA, although somewhat competitive on small-scale problems, shows a pronounced drop in GMR as the neuron count increases. This shortfall arises from its positive–negative propagation strategy in training, which focuses primarily on local weight adjustments, leading to low NV and thus constraining the solution-space exploration. Meanwhile, an inadequate DR (solution space coverage) limits the network's ability to achieve a global optimum, even when local fields are updated via the sign function in retrieval. DHNN-RAN3SAT-ACO, while maintaining modest GMR at small neuron counts, declines rapidly as network size increases. The pheromone mechanism, inherently suited to sequential path construction, proves ineffective in discrete Hopfield energy landscapes. As a result, ACO suffers from limited NV and low exploration diversity and lacks adaptability during retrieval, thereby failing to drive neuron states toward the global minimum. DHNN-RAN3SAT exhibits the poorest GMR overall, nearly failing to converge at any problem scale. The root cause lies in its fixed-strategy approach during training, which yields a static weight matrix lacking adaptability or dynamic revisions, resulting in extremely low NV and severely limited DR. Consequently, even if the sign function is applied for local-field updates in retrieval, the randomly generated neuron states cannot effectively converge to the global optimum.

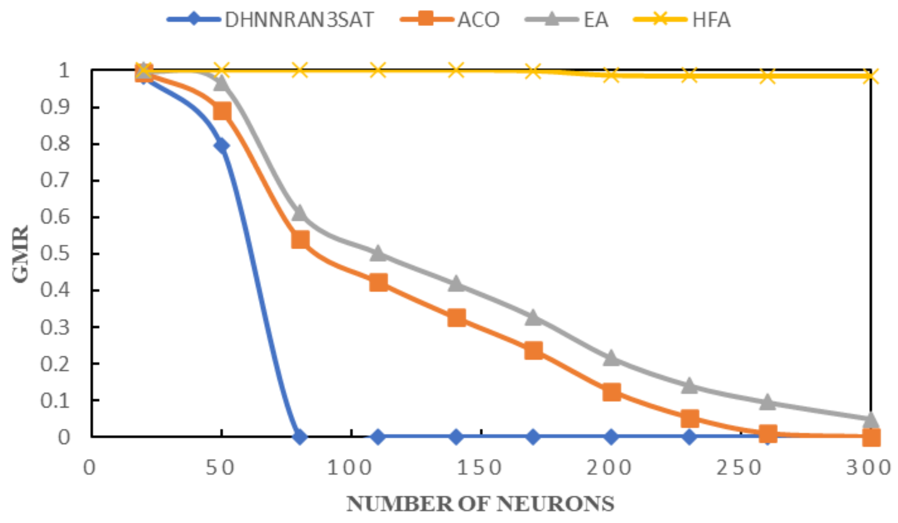


Figure 21. Global minimum ratio for different DHNN-RAN3SAT models.

Hence, by combining the brightness attraction mechanism with higher NV and DR and leveraging the sign function for dynamic adjustments, DHNN-RAN3SAT-HFA achieves a more favorable GMR. Although DHNN-RAN3SAT-EA performs reasonably well for smaller-scale problems, its lower NV and DR restrict global search capability, making stable convergence to the global optimum difficult, even with the sign function. ACO suffers from limited NV, poor scalability, and a lack of efficient guidance in discrete random-state networks, making its GMR performance notably weaker in large-scale settings. Finally, DHNN-RAN3SAT, hampered by a fixed-weight strategy and extremely low NV and DR, cannot effectively guide neuron states toward the global optimum during retrieval.

10.2.2. Mean absolute error of the energy function

During the retrieval phase, the model utilizes the weights generated during training, along with randomly initialized neuron states and dynamically updated local fields, to compute the energy value. The mean absolute error (MAE) quantifies the discrepancy between the computed energy and the theoretical global minimum energy solution, serving as a measure of the model's convergence accuracy and optimization performance in the retrieval stage. To more thoroughly interpret variations in MAE, this study introduces two indirect supportive indicators as follows:

Neuron variation (NV): A higher NV indicates more frequent neuron-state changes, facilitating exploration of a larger solution space, thereby reducing MAE.

Diversity rate (DR): When DR $\geq$ 40%, the solution space is deemed sufficiently covered, allowing randomly generated neuron states to lie closer to the global optimum and further lowering MAE.

Figure 22 illustrates the MAE performance of four algorithms under different neuron counts, reflecting their respective robustness and adaptability. DHNN-RAN3SAT-HFA maintains an MAE near zero across all problem scales, evidencing extremely high convergence accuracy. This outcome arises from the adaptive weight matrix produced by the hybrid firefly algorithm in the training phase, enabling the sign function in the retrieval phase to precisely guide randomly generated neuron states toward the global optimum. A high NV (frequent neuron-state changes) promotes broader solution-space exploration, while DR $\geq$ 40% (adequate coverage) ensures stable discovery of the global optimum. Moreover, the stochastic perturbation mechanism helps avert local minima, making solution-space exploration more comprehensive and effectively reducing MAE. By contrast, the MAE of DHNN-RAN3SAT-EA gradually increases with the neuron count, becoming more pronounced in medium- and large-scale problems. This behavior primarily arises from two factors, namely the lower NV and lower DR. For the former, the positive–negative propagation strategy adopted during training leans toward local optimization, limiting neuron-state changes and impeding solution-space diversity. For the lower DR, the solution space is insufficiently covered, causing randomly generated neuron states to remain relatively distant from the global optimum. Meanwhile, the absence of a dynamic perturbation mechanism restricts solution diversity, further driving up MAE. DHNN-RAN3SAT-ACO exhibits a moderate MAE increase with neuron scale but fails to maintain accuracy in larger problems. The pheromone mechanism, which favors path construction tasks, lacks adaptability for the random-state dynamics of Hopfield networks. Consequently, ACO suffers from insufficient NV and suboptimal DR, limiting solution space coverage and leading to subpar convergence during retrieval. DHNN-RAN3SAT experiences a rapid escalation in MAE as the neuron count grows and exhibits the worst overall performance across all problem scales. The main reasons include the extremely low NV, in which a fixed-strategy approach yields a static weight matrix, resulting in minimal neuron-state changes and inadequate exploration of the solution space; and the DR far below 40%, in which insufficient coverage hinders effective convergence to the global optimum, even with local-field updates via the sign function. Additionally, a fixed-weight strategy prevents the network from adapting to random neuron states during retrieval, leading to the poorest MAE performance.

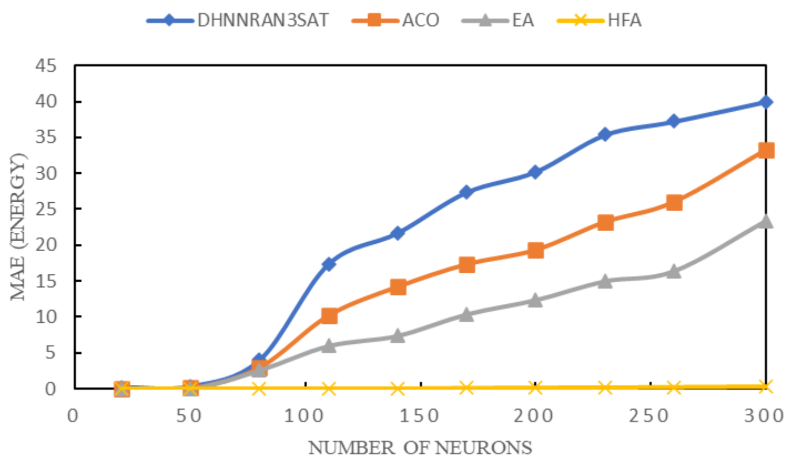


Figure 22. MAE energy for different DHNN-RAN3SAT models.

Underpinned by high NV and DR, DHNN-RAN3SAT-HFA—in conjunction with the brightness-attraction mechanism and stochastic perturbation—achieves a superior MAE that remains nearly zero. DHNN-RAN3SAT-EA, hampered by lower NV and DR, shows acceptable MAE performance at small scales but deteriorates markedly in larger-scale problems. Meanwhile, DHNN-RAN3SAT-ACO displays moderate MAE at first, but rapidly worsens as neuron count increases, due to limited diversity and poor adaptability in its pheromone-driven optimization. Lastly, DHNN-RAN3SAT, constrained by a fixed-weight strategy and extremely low NV and DR, attains the worst MAE among the three, further underscoring the importance of solution-space exploration and diversity for convergence accuracy.

10.2.3. Similarity index

The Jaccard similarity index (JI) and the Gower–Legendre similarity index (GLI) are used to measure the consistency between the solutions generated during retrieval and the ideal target solution. These indices reflect both the model's matching degree within the solution space and its ability to maintain coherence across positive and negative logical clauses. Figure 23 shows that, for DHNN-RAN3SAT-HFA, the JI remains around 0.6 for all problem scales, indicating a high level of consistency. This advantage primarily stems from the adaptive weight matrix produced in the training phase, which allows the sign function in the retrieval phase to effectively guide randomly generated neuron states to converge rapidly toward the ideal solution. In addition, a higher NV leads to more frequent state changes in the retrieval phase, fostering broader exploration of the solution space and thereby enhancing overall consistency. A higher DR (DR ≥ 40%) indicates more comprehensive solution space coverage, enabling the network to more stably find states close to the ideal solution, thus further boosting the JI. By contrast, DHNN-RAN3SAT-EA shows its JI decreasing from 0.6 to 0.4 as the neuron count grows. Major reasons for this decline include: (1) lower NV, since the positive–negative propagation strategy in the training phase focuses on local weight optimization, leading to insufficient neuron-state changes and restricting solution-space diversity; and (2) lower DR, meaning inadequate coverage of the solution space, making it difficult for randomly generated neuron states to match the ideal solution. DHNN-RAN3SAT-ACO also exhibits a noticeable decline in JI as the neuron count increases, with its curve dropping below 0.3 at larger scales. In random satisfiability contexts, the ACO's solution construction strategy—originally designed for sequential path-based problems—struggles to maintain consistency between neuron states and randomly assigned clause structures. The absence of dynamic feedback during retrieval leads to lower NV, while limited coverage of diverse neuron states results in reduced DR, jointly weakening the model's ability to converge toward consistent, satisfiable states. DHNN-RAN3SAT performs the worst in JI across all problem scales, dropping below 0.3. Its extremely low NV—caused by a fixed-weight strategy that rarely updates neuron states—limits exploration of the solution space. Moreover, low DR means insufficient coverage, leaving randomly generated neuron states unable to match the ideal solution.

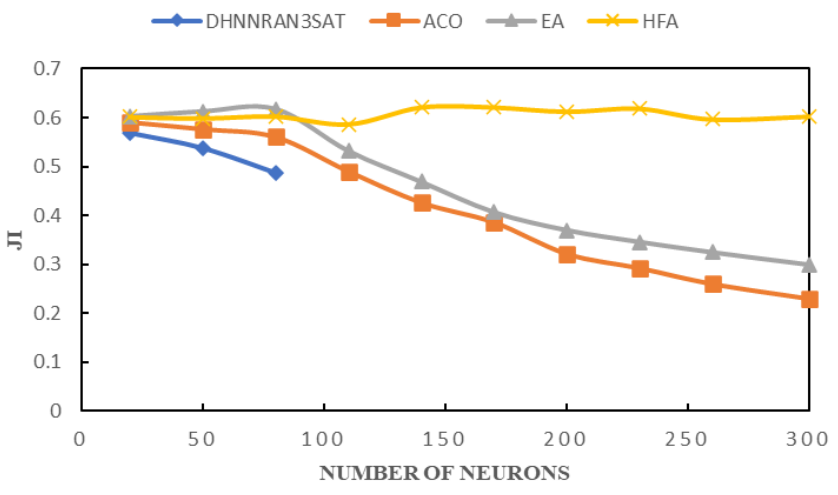


Figure 23. JI for different DHNN-RAN3SAT models.

The Gower–Legendre similarity index (GLI) places more emphasis on the model's ability to handle positive and negative logical clauses consistently. Figure 24 shows that DHNN-RAN3SAT-HFA maintains a GLI of approximately 0.7 across all problem scales, demonstrating excellent logical consistency. This strength derives from the joint support of a higher NV and DR. A higher NV signals more frequent neuron-state changes, facilitating broader exploration and reducing the risk of solution stagnation in localized regions. A DR of at least 40% (DR $\geq$ 40%) suggests sufficient coverage of the solution space, enabling the network to sustain strong consistency between positive and negative logical clauses and thereby further improving GLI performance. In addition, the adaptive weight matrix generated in the training phase, together with the brightness-attraction mechanism, effectively guides neuron states to quickly converge to the global optimum during the retrieval phase via the sign function. The interplay between the stochastic perturbation mechanism and NV further increases the depth and diversity of exploration, thus enhancing logical consistency. In contrast, DHNN-RAN3SAT-EA sees its GLI drop from 0.7 to 0.3 as the neuron count increases, indicating significant performance degradation. Its low NV and DR levels imply insufficient neuron-state changes and a limited search scope, causing solutions to cluster in local regions without effectively adapting to negatively correlated solutions in the global space. Additionally, the partitioning strategy's limited coverage further exacerbates this decline. DHNN-RAN3SAT-ACO shows a GLI decline similar to EA, dropping from approximately 0.65 to below 0.1 as neuron count increases. In random satisfiability problems, the pheromone-driven strategy of ACO does not adapt well to the stochastic clause composition and sign-based retrieval update mechanism. Its lower NV reflects limited neuron-state transitions, and a DR below 40% results in insufficient coverage of negatively correlated clause regions. These factors reduce the model's ability to maintain logical clause consistency during retrieval, thereby limiting GLI performance at larger scales. DHNN-RAN3SAT performs the worst in GLI, reaching only around 0.2, and exhibits a notably low NV as well. The fixed-weight strategy entirely fails in the retrieval phase, making it impossible for randomly generated neuron states to adapt to negatively correlated clauses. Due to the low NV, neuron-state updates lack both frequency and diversity, causing solutions to cluster in a small subset of the space and severely restricting solution space coverage.

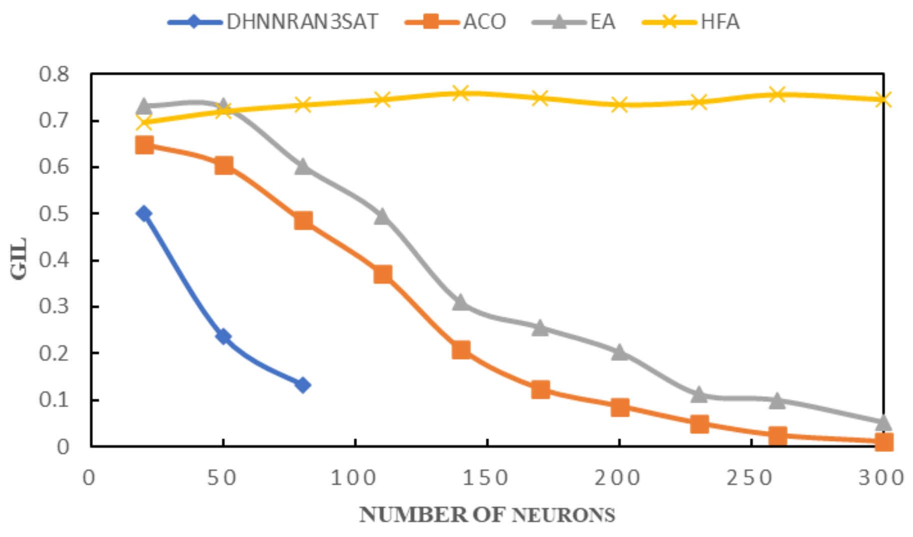


Figure 24. GLI for different DHNN-RAN3SAT models.

Overall, DHNN-RAN3SAT-HFA demonstrates excellent logical adaptability and global search capability during the retrieval phase. This strength results from the optimized weights obtained in the training phase, together with maintaining high NV and DR, ensuring broader solution space coverage and logical consistency. Consequently, it achieves the best performance in both JI and GLI, fully exploring the solution space with strong consistency. By contrast, DHNN-RAN3SAT-EA, whose training-phase weights lean toward local optimization and whose NV and DR are relatively low, exhibits insufficient adaptability in more complex logical combinations and large-scale problems. Similarly, DHNN-RAN3SAT-ACO fails to maintain consistency at scale, due to low neuron-state variation and poor clause-space coverage. While competitive at smaller sizes, its performance deteriorates in large networks, limiting its utility for global consistency in random satisfiability problems. Meanwhile, DHNN-RAN3SAT, burdened by a fixed-weight strategy and extremely low NV and DR, performs the worst in both JI and GLI due to insufficient coverage and consistency within the solution space.

10.2.4. Runtime

Runtime (CPU time) serves as a crucial metric for assessing the overall computational performance of a model during both the training and retrieval phases. Figure 25 depicts the runtime trends of three algorithms under varying neuron counts, highlighting notable differences in computational efficiency.

DHNN-RAN3SAT-HFA consistently achieves the shortest runtime across all problem scales and displays an approximately linear growth trend, indicating exceptional computational efficiency. This advantage is primarily due to the following factors: In the training phase, the brightness attraction mechanism (BAM) and a random perturbation mechanism adaptively adjust weights, reducing redundant iterations and expediting the calculation of optimal weights. The training goal is to find the optimal neuron state that satisfies the largest number of logical clauses, thereby minimizing the cost function and producing an optimal weight matrix. In the retrieval phase, the sign function–based local-field update further streamlines the computational process, making neuron-state updates more efficient and shortening runtime. In addition, the information entropy–based clause ratio (1:3:6) employed during training mitigates imbalances in the solution space, thus further lowering computational overhead.

DHNN-RAN3SAT-EA exhibits intermediate runtime values between those of HFA and DHNN-RAN3SAT and grows exponentially with increasing neuron counts. The key reasons are as follows: In the training phase, the positive–negative propagation strategy emphasizes weight adjustments in localized regions, necessitating frequent matrix updates and introducing greater computational complexity when encountering more intricate logical combinations. In the retrieval phase, because the solution space was insufficiently covered during training, additional local-field updates and energy computations are required to attempt convergence toward the optimal solution, thereby adding to the computational load.

DHNN-RAN3SAT-ACO exhibits a runtime curve close to DHNN-RAN3SAT, particularly in large-scale problems. Although its pheromone mechanism supports some degree of guided exploration, it lacks real-time adaptability and incurs extra overhead due to repeated probability matrix updates. In the context of randomly satisfiable logic, where clause composition and neuron states change stochastically, ACO's fixed path-construction approach struggles to cope efficiently. Consequently, both the training and retrieval phases consume more time as the network grows, reflecting limited scalability.

DHNN-RAN3SAT displays the highest runtime at all problem scales, with a pronounced exponential growth trend that severely undermines performance for large-scale problems. Several underlying causes contribute to this outcome. In he training phase, a fixed-strategy approach generates a static weight matrix lacking adaptive optimization, forcing the model to spend more time searching for solutions that satisfy more logical clauses. Furthermore, the absence of stochastic perturbation and weight adaptation exacerbates this inefficiency. In the retrieval phase, owing to low diversity in the training phase, the network's state coverage (DR) remains inadequate, constraining neuron-state variations. Consequently, additional iterations are needed to attempt discovery of the global optimum, further increasing complexity.

Although runtime primarily reflects computational efficiency, NV and DR indirectly influence complexity. A higher NV suggests more frequent neuron-state changes, potentially increasing state updates and energy calculations. Conversely, a higher DR indicates broad solution space coverage, which helps reduce redundant computations during training. However, if DR is too low, more iterations are required in the retrieval phase to explore states and compute updates, prolonging runtime.

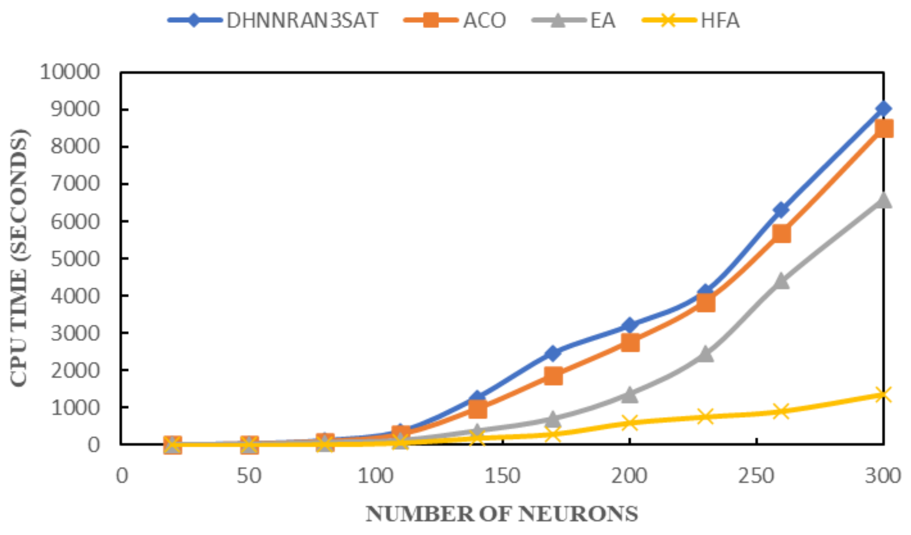


Figure 25. CPU time for different DHNN-RAN3SAT models.

As a key performance indicator, runtime reveals each algorithm's overall efficiency during training and retrieval. While DHNN-RAN3SAT-EA retains some local optimization capacity, its lower NV and DR cause runtime to escalate notably with larger neuron counts. DHNN-RAN3SAT-ACO, although partially guided by pheromone mechanisms, lacks adaptive clause handling and incurs higher computational overhead, especially in large-scale random satisfiability settings. DHNN-RAN3SAT, constrained by a fixed-weight strategy and lacking adaptive adjustments or diversity support, yields the highest runtime and scales poorly as neuron numbers increase.

10.3. Diversity analysis based on information entropy optimization

To quantitatively assess the model's performance regarding the distribution of logical clauses and the coverage of the solution space, this section introduces information entropy as the theoretical basis for our diversity metric. By calculating the complexity and proportional distribution of various types of logical clauses, information entropy effectively measures both the coverage and the global search capability of the solution space. Figures 26 and 27 respectively illustrate how information entropy contributes to proportional allocation and diversity improvements, as well as the corresponding calculation process.

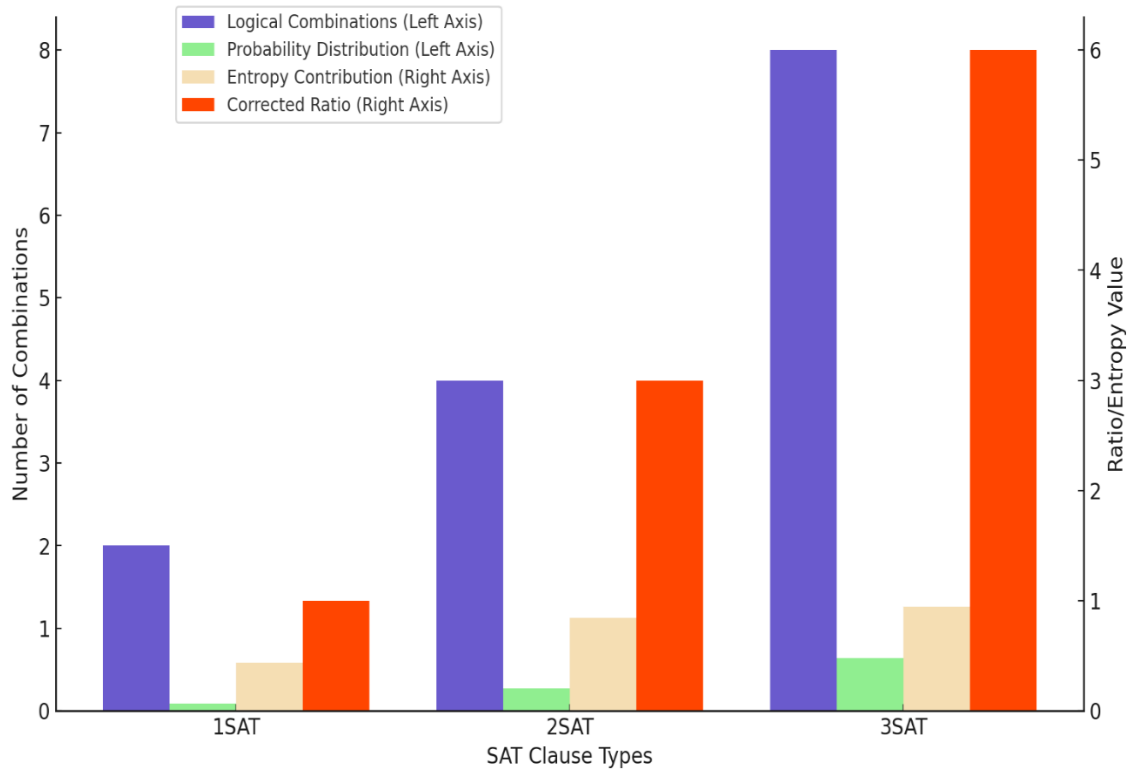


Figure 26. Bar chart of logical combinations, entropy, and correction ratio for 1SAT, 2SAT, and 3SAT clauses.

Figure 26 presents the final calculation results for 1SAT, 2SAT, and 3SAT clauses in a bar chart, depicting the changing trends of the number of logical combinations, probability distributions, entropy contributions, and the correction ratio. The left Y-axis displays the number of logical combinations, their probability distributions, and their entropy contributions, whereas the right Y-axis plots the correction ratio. As SAT complexity increases, both the number of logical combinations and the entropy contribution grow significantly. Notably, 3-SAT exhibits the highest entropy contribution, indicating greater logical complexity and uncertainty in more complex clauses. Consequently, the correction ratio is raised to balance global search capability with the diversity of solution space distributions.

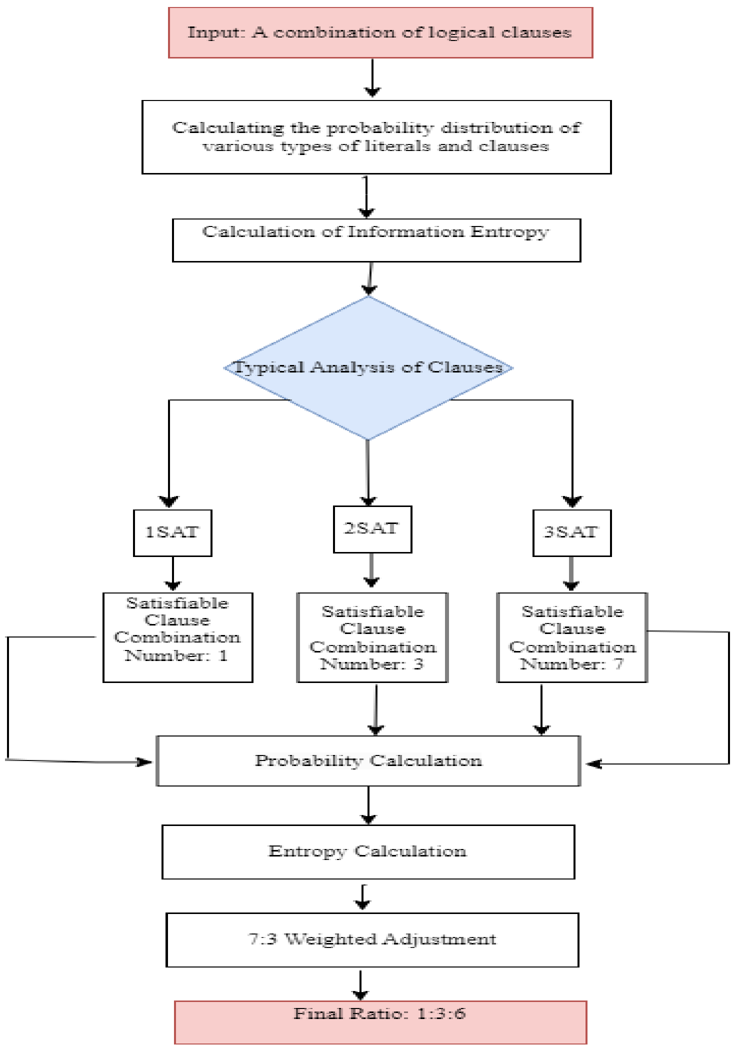

Input: A combination of logical clauses
Calculating the probability distribution of various types of literals and clauses
Calculation of Information Entropy
Typical Analysis of Clauses
1SAT
2SAT
3SAT
Satisfiable Clause Combination Number: 1
Satisfiable Clause Combination Number: 3
Satisfiable Clause Combination Number: 7
Probability Calculation
Entropy Calculation
7:3 Weighted Adjustment
Final Ratio: 1:3:6

Figure 27. Flowchart of the proportional allocation process for logical clauses.

Figure 27 illustrates the detailed proportional-allocation methodology for logical clauses in a comprehensive flowchart format. The process begins with an input set of logical clauses and progresses through distinct computational stages: first, calculating probability distributions across 1SAT, 2SAT, and 3SAT clause types, then, determining information entropy values for each type. The analysis phase explicitly identifies satisfiable clause combinations (with specific combination counts of 1, 3, and 7 for 1SAT, 2SAT, and 3SAT, respectively), feeding these values into subsequent probability and entropy calculations. The final stages implement a precise 70:30 weighted adjustment mechanism that transforms the preliminary values into the final 1:3:6 distribution ratio. Figures 26 and 27 complement each other: Figure 27 maps the sequential computational workflow with explicit intermediate values, while Figure 26 quantitatively visualizes the relationships between logical combinations, probability distributions, entropy contributions, and correction ratios for each SAT clause type throughout the allocation process.

In summary, information entropy provides a scientifically grounded approach to clause proportional allocation. By applying a well-reasoned distribution correction (1:3:6), the model significantly enhances both the coverage of the solution space and its global search capability. This ensures that the model maintains higher stability and adaptability under complex logical conditions.

11. Conclusions and future work

In this study, we propose a random 3-SAT model that realizes a non-systematic logical representation by randomly generating first-, second-, and third-order clauses. Current research on random 3-SAT in discrete Hopfield neural networks lacks clear standards for the proportion and quantity of clauses [4,5,22,25,26,62]. To address this gap, we introduce a clause-ratio setting method based on information entropy during the training phase. This method has significant theoretical advantages, effectively balancing logical complexity with computational efficiency, while flexibly adjusting clause proportions to accommodate real-world requirements.

Moreover, we observe that as the number of neurons grows—especially when the counts of 2-SAT and 3-SAT clauses increase—the complexity of the learning phase rises exponentially. To tackle this issue, we propose a hybrid firefly algorithm (HFA), whose primary goal is to optimize neuron states and compute better weights by identifying additional logic-satisfying clauses. HFA achieves a balance between global and local search via position updates and introduces an elitist strategy in the fitness-update and selection steps to efficiently optimize both logical clauses and weight calculations. Experimental results indicate that HFA substantially improves the discovery of global optima and demonstrates higher efficiency in addressing complex logical problems. Compared to traditional methods and the election algorithm, HFA exhibits marked advantages in fitness metrics—including maximum fitness, fitness ratio, and similarity indices—surpassing existing approaches. In addition, its accuracy in the energy function is notably enhanced, reducing absolute error and improving the global minimum ratio. On larger-scale neuron sets, HFA shows excellent computational efficiency, thus offering a more effective solution path for complex logical tasks. Recent research on hybrid firefly models further supports the effectiveness of combining multiple search strategies for enhanced performance in complex optimization scenarios [46].

Regarding weight computation, we adopt the WAN-LC method. Unlike traditional weight-calculation methods that require comparing cost functions with energy functions, WAN-LC calculates weights directly based on the structure of logical clauses. This not only streamlines the computation but also boosts efficiency.

In summary, the hybrid firefly algorithm and the information-entropy-based clause-ratio setting proposed in this study provide not only a novel theoretical framework for solving complex logical problems but also exhibit substantial optimization potential in practical applications. Although still at an experimental stage, the random 3-SAT model—especially with a higher proportion of 3-SAT clauses—demonstrates great applicability in complex logic optimization, circuit design, and task scheduling. In particular, 3-SAT clauses effectively capture multi-variable logical constraints, addressing the modeling demands of multi-level logical relationships in real scenarios. Experimental validation shows outstanding performance in discovering global optima, enhancing energy-function precision, and boosting computational efficiency. These findings suggest that HFA is suitable for both theoretical exploration and future applications in large-scale logic systems.

For future research, we plan to further explore the application and optimization of random 3-SAT within discrete Hopfield neural networks in practical contexts. Unlike traditional studies, our model contains mixed 1-SAT, 2-SAT, and 3-SAT clauses, reflecting real-world needs. For instance, 1-SAT clauses usually specify simple single-variable constraints, 2-SAT clauses are suitable for modeling binary relations or conflict-optimization tasks, and 3-SAT clauses find broad application in complex logic optimization and circuit design [63–65]. Such a hybrid approach allows for more flexible representation of multi-level logical constraints, including scheduling, resource allocation, and hardware circuit optimization. Additionally, prior work [66] on random 3-SAT highlights its significant theoretical and practical value for optimization, including logic planning, circuit design, and algorithm performance testing. Combined with logical mining techniques, we intend to further broaden the model's applicability—such as analyzing palm-oil trends [67] and other optimization problems [12,68].

Given that existing research has largely focused on 2-SAT during the retrieval phase [6], we aim to design an intelligent algorithm consistent with random 3-SAT in the retrieval stage to enhance the diversity and global efficacy of solutions. The complexity and potential phase transitions of the random 3-SAT model provide an important theoretical foundation for optimization studies. Building on [66], which further validates the pivotal role of random 3-SAT in exploring large solution spaces, we will conduct future experiments to examine this approach's global search capacity and solution space coverage, as well as explore how ratio adjustments apply to diverse problem types. Specifically, we plan a series of experiments adjusting the proportions of 1-SAT, 2-SAT, and 3-SAT clauses, observing variations in model performance. One limitation of the current theoretical research on the satisfiability problem in DHNN is that the largest scale studied so far involves only 300 neurons [4–6,22–29]. Future work could explore extending the scale further to examine its impact on optimization performance. Through these refined parameter-tuning strategies, we aim to address optimization problems of differing complexity more effectively. Such directions are also consistent with emerging research trends that emphasize hybridized swarm-based algorithms for solving large-scale NP-hard problems [46].

Author contributions

Xiaoya Chen: Writing − original draft, review & editing, visualization, software, validation; Saratha Sathasivam: Writing − review & editing, funding acquisition, software, validation, Supervision. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research was funded by the Ministry of Higher Education Malaysia (MOHE) under the Fundamental Research Grant Scheme (FRGS) (Grant No.: FRGS/1/2022/STG06/USM/02/11) and supported by Universiti Sains Malaysia.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. S. Haykin, *Neural networks and learning machines*, Pearson Education. Inc., New Jersey, 2009. Available from: http://dai.fmph.uniba.sk/courses/NN/haykin.neural-networks.3ed.2009.
2. J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *P. Natl. Acad. Sci.*, **79** (1982), 2554−2558. https://doi.org/10.1073/PNAS.79.8.2554

3. W. A. T. W. Abdullah, Logic programming on a neural network, *Int. J. Intell. Syst.*, **7** (1992), 513−519.
4. S. Sathasivam, M. A. Mansor, M. S. M. Kasihmuddin, H. Abubakar, Election algorithm for random k satisfiability in the Hopfield neural network, *Processes*, **8** (2020), 568. https://doi.org/10.3390/pr8050568
5. M. M. Bazuhair, S. Z. M. Jamaludin, N. E. Zamri, M. S. M. Kasihmuddin, M. A. Mansor, A. Alway, et al., Novel Hopfield neural network model with election algorithm for random 3 satisfiability, *Processes*, **9** (2021), 1292. https://doi.org/10.3390/pr9081292
6. Y. Guo, N. E. Zamri, M. S. M. Kasihmuddin, A. Alway, M. A. Mansor, J. Li, et al., Dual optimization approach in discrete Hopfield neural network, *Appl. Soft Comput.*, **164** (2024), 111929. https://doi.org/10.1016/j.asoc.2024.111929
7. R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997), 341−359. https://doi.org/10.1023/A:1008202821328
8. L. Zhang, L. Liu, X. S. Yang, Y. Dai, A novel hybrid firefly algorithm for global optimization, *PloS One*, **11** (2016), e0163230. https://doi.org/10.1371/journal.pone.0163230
9. X. S. Yang, *Nature-inspired metaheuristic algorithms*, Luniver press, 2010.
10. I. Fister, I. Fister Jr, X. S. Yang, J. Brest, A comprehensive review of firefly algorithms, *Swarm Evol. Comput.*, **13** (2013), 34−46. https://doi.org/10.1016/j.swevo.2013.06.001
11. S. L. Tilahun, J. M. T. Ngnotchouye, Firefly algorithm for discrete optimization problems: A survey, *KSCE J. Civil Eng.*, **21** (2017), 535−545. https://doi.org/10.1007/s12205-017-1501-1
12. C. Feng, S. Sathasivam, N. Roslan, M. Velavan, 2-SAT discrete Hopfield neural networks optimization via Crow search and fuzzy dynamical clustering approach, *AIMS Math.*, **9** (2024), 9232−9266. https://doi.org/10.3934/math.2024450
13. L. C. Lujano-Hernandez, J. M. Munoz-Pacheco, V. T. Pham, A fully piecewise linear Hopfield neural network with simplified mixed-mode activation function: dynamic analysis and analog implementation, *Nonlinear Dynam.*, 2025, 1−22. https://doi.org/10.1007/s11071-025-11099-y
14. D. Alsadie, M. Alsulami, Enhancing workflow efficiency with a modified firefly algorithm for hybrid cloud edge environments, *Sci. Rep.*, **14** (2024), 24675. https://doi.org/10.1038/s41598-024-75859-3
15. W. A. T. W. Abdullah, The logic of neural networks, *Phys. Lett. A*, **176** (1993), 202−206.
16. S. Sathasivam, *Learning in the recurrent Hopfield network*, In: 2008 fifth international conference on computer graphics, imaging and visualization, IEEE, 2008, 323−328. https://doi.org/10.1109/CGIV.2008.14
17. S. Sathasivam, W. A. T. W. Abdullah, Logic mining in neural network: Reverse analysis method, *Computing*, **91** (2011), 119−133. https://doi.org/10.1007/s00607-010-0117-9
18. M. S. M. Kasihmuddin, M. A. Mansor, S. Sathasivam, Bezier curves satisfiability model in enhanced Hopfield network, *Int. J. Intell. Syst. Appl.*, **8** (2016), 9.
19. M. A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Enhanced Hopfield network for pattern satisfiability optimization, *Int. J. Intell. Syst. Appl.*, **8** (2016), 27. http://dx.doi.org/10.5815/ijisa.2016.11.04
20. S. Sathasivam, M. A. Mansor, A. I. M. Ismail, S. Z. M. Jamaludin, M. S. M. Kasihmuddin, M. Mamat, Novel random k satisfiability for k≤ 2 in Hopfield neural network, *Sains Malays.*, **49** (2020), 2847−2857. http://dx.doi.org/10.17576/jsm-2020-4911-23

21. S. A. Karim, N. E. Zamri, A. Alway, M. S. M. Kasihmuddin, A. I. M. Ismail, M. A. Mansor, et al., Random satisfiability: A higher-order logical approach in discrete Hopfield neural network, *IEEE Access*, **9** (2021), 50831−50845. https://doi.org/10.1109/ACCESS.2021.3068998
22. H. Abubakar, M. L. Danrimi, Hopfield type of artificial neural network via election algorithm as heuristic search method for random Boolean k satisfiability, *Int. J. Comput. Digital Syst.*, 2021. https://doi. org/10.12785/ijcds/100163
23. H. Abubakar, A. S. Masanawa, S. Yusuf, G. I. Boaku, Optimal representation to high order random Boolean k satisability via election algorithm as heuristic search approach in Hopeld neural networks, *J. Nigerian Soc. Phys. Sci.*, 2021, 201−208.
24. H. Abubakar, A. Muhammad, S. Bello, Ants colony optimization algorithm in the Hopfield neural network for agricultural soil fertility reverse analysis, *Iraqi J. Comput. Sci. Math.*, **3** (2022), 32−42.
25. S. A. Karim, M. S. M. Kasihmuddin, S. Sathasivam, M. A. Mansor, S. Z. M. Jamaludin, M. R. Amin, A novel multi-objective hybrid election algorithm for higher-order random satisfiability in discrete Hopfield neural network, *Mathematics*, **10** (2022), 1963. https://doi.org/10.3390/math10121963
26. S. A. Karim, M. S. M. Kasihmuddin, M. Mansor, S. Z. M. Jamaludin, N. E. Zamri, M. R. Amin, *A socio-inspired hybrid election algorithm for random k satisfiability in discrete Hopfield neural network*, In: AIP Conference Proceedings, **2895** (2024). https://doi.org/10.1063/5.0194531
27. H. Abubakar, S. Sathasivam, M. A. Mansor, M. S. M. Kasihmuddin, Modified election algorithm in accelerating the performance of Hopfield neural network for random k-satisfiability, *Preprints*, 2020. https://doi.org/10.20944/preprints202001.0365.v1
28. S. A. Karim, *Multi-objective hybrid election algorithm for random K satisfiability in discrete hopfield neural network*, Doctoral dissertation, 2023.
29. V. Kumar, D. D. Kumar, A systematic review on firefly algorithm: Past, present, and future, *Arch. Comput. Method. E.*, **28** (2021), 3269−3291. https://doi.org/10.1007/s11831-020-09498-y
30. Y. K. Saheed, *A binary firefly algorithm based feature selection method on high dimensional intrusion detection data*, In: Illumination of artificial intelligence in cybersecurity and forensics, Cham: Springer International Publishing, 2022, 273−288. https://doi.org/10.1007/978-3-030-93453-8_12
31. K. Kumari, J. P. Singh, Multi-modal cyber-aggression detection with feature optimization by firefly algorithm, *Multimedia Syst.*, **28** (2022), 1951−1962. https://doi.org/10.1007/s00530-021-00785-7
32. Y. Zhang, X. F. Song, D. W. Gong, A return-cost-based binary firefly algorithm for feature selection, *Inform. Sci.*, **418** (2017), 561−574. https://doi.org/10.1016/j.ins.2017.08.047
33. R. F. Najeeb, B. N. Dhannoon, A feature selection approach using binary firefly algorithm for network intrusion detection system, *ARPN J. Eng. Appl. Sci.*, **13** (2018), 2347−2352.
34. K. Chandrasekaran, S. P. Simon, N. P. Padhy, Binary real coded firefly algorithm for solving unit commitment problem, *Inform. Sci.*, **249** (2013), 67−84. https://doi.org/10.1016/j.ins.2013.06.022
35. R. O. N. O. H. Kennedy, G. Kamucha, Comparison of hybrid firefly algorithms for binary and continuous optimization problems in a TV white space network, *WSEAS Trans. Commun.*, **19** (2020), 155−172. https://doi.org/10.37394/23204.2020.19.18

36. K. M. Alwan, A. H. AbuEl-Atta, H. H. Zayed, Feature selection models based on hybrid firefly algorithm with mutation operator for network intrusion detection, *Int. J. Intell. Eng. Syst.*, **14** (2021). https://doi.org/ 10.22266/ijies2021.0228.19
37. R. Z. Al-Abdallah, A. S. Jaradat, I. A. Doush, Y. A. Jaradat, A binary classifier based on firefly algorithm, *Jordanian J. Comput. Info.*, **3** (2017). https://doi.org/10.5455/jjcit.71-1501152301
38. M. A. Z. A. Sofiane, D. Zouache, *Binary firefly algorithm for feature selection in classification*, In: 2019 international conference on theoretical and applicative aspects of computer science (ICTAACS), IEEE, **1** (2019). https://doi.org/10.1109/ICTAACS48474.2019.8988137
39. O. S. Qasim, N. M. Noori, A new hybrid algorithm based on binary gray wolf optimization and firefly algorithm for features selection, *Asian-Eur. J. Math.*, **14** (2021), 2150172. https://doi.org/10.1142/S1793557121501722
40. W. Xie, L. Wang, K. Yu, T. Shi, W. Li, Improved multi-layer binary firefly algorithm for optimizing feature selection and classification of microarray data, *Biomed. Signal Proces.*, **79** (2023), 104080. https://doi.org/10.1016/j.bspc.2022.104080
41. S. G. Devi, M. Sabrigiriraj, A hybrid multi-objective firefly and simulated annealing based algorithm for big data classification, *Concurr. Comp.-Pract. E.*, **31** (2019), e4985. https://doi.org/10.1002/cpe.4985
42. M. F. P. Costa, A. M. A. Rocha, R. B. Francisco, E. M. Fernandes, Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization, *Adv. Oper. Res.*, **2014** (2014), 215182. https://doi.org/10.1155/2014/215182
43. K. Mourad, R. Boudour, A modified binary firefly algorithm to solve hardware/software partitioning problem, *Informatica*, **45** (2021).
44. R. Goel, R. Maini, A hybrid of ant colony and firefly algorithms (HAFA) for solving vehicle routing problems, *J. Comput. Sci.*, **25** (2018), 28−37. https://doi.org/10.1016/j.jocs.2017.12.012
45. A. E. S. Ezugwu, M. B. Agbaje, N. Aljojo, R. Els, H. Chiroma, M. A. Elaziz, A comparative performance study of hybrid firefly algorithms for automatic data clustering, *IEEE Access*, **8** (2020), 121089−121118. https://doi.org/10.1109/ACCESS.2020.3006173
46. M. Y. I. Idris, M. N. B. Ayub, H. A. Shehadeh, U. Ali, Hybrid firefly algorithm and sperm swarm optimization algorithm using Newton-Raphson method (HFASSON) and its application in CR-VANET, *arXiv preprint*, 2025. https://doi.org/10.48550/arXiv.2502.01053
47. R. G. Jeroslow, J. Wang, Solving propositional satisfiability problems, *Ann. Math. Artif. Intel.*, **1** (1990), 167−87. https://doi.org/10.1007/BF01531077
48. J. Marques-Silva, *Practical applications of Boolean satisfiability*, In: 2008 9th International Workshop on Discrete Event Systems, IEEE, 2008, 74−80. https://doi.org/10.1109/WODES.2008.4605925
49. S. A. Cook, *The complexity of theorem-proving procedures*, In: Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook, 2023, 143−152. https://doi.org/10.1145/800157.805047
50. J. J. Hopfield, D. W. Tank, "Neural" computation of decisions in optimization problems, *Biol. Cybern.*, **52** (1985), 141−152. https://doi.org/ 10.1007/BF00339943
51. J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, *P. Natl. Acad. Sci.*, **81** (1984), 3088−3092. https://doi.org/10.1073/pnas.81.10.3088

52. C. E. Shannon, A mathematical theory of communication, *Bell Syst. Tech. J.*, **27** (1948), 379−423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
53. K. Deb, Nonlinear goal programming using multi-objective genetic algorithms, *J. Oper. Res. Soc.*, **52** (2001), 291−302. https://www.jstor.org/stable/254066
54. C. H. Papadimitriou, *Computational complexity Addison-Wesley reading*, Massachusetts Google Scholar, 1994.
55. M. R. Gary, D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, WH Freman and Co.,1979.
56. H. Ma, H. Wei, Y. Tian, R. Cheng, X. Zhang, A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints, *Inform. Sci.*, **560** (2021), 68−91. https://doi.org/10.1016/j.ins.2021.01.029
57. A. Zhou, B. Y. Qu, H. Li, S. Z. Zhao, P. N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, *Swarm Evol. Comput.*, **1** (2011), 32−49. https://doi.org/10.1016/j.swevo.2011.03.001
58. V. Osuna-Enciso, E. Cuevas, B. M. Castañeda, A diversity metric for population-based metaheuristic algorithms, *Inform. Sci.*, **586** (2022), 192−208. https://doi.org/10.1016/j.ins.2021.11.073
59. C. J. Willmott, K. Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, *Clim. Res.*, **30** (2005), 79−82. https://doi.org/10.3354/CR030079
60. N. E. Zamri, A. Alway, A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Modified imperialistic competitive algorithm in Hopfield neural network for Boolean three satisfiability logic mining, *Pertanika J. Sci. Tech.*, **28** (2020).
61. E. De Santis, A. Martino, A. Rizzi, On component-wise dissimilarity measures and metric properties in pattern recognition, *PeerJ Comput. Sci.*, **8** (2022), e1106. https://doi.org/10.7717/peerj-cs.1106
62. S. A. Karim, N. E. Zamri, A. Alway, M. S. M. Kasihmuddin, A. I. M. Ismail, M. A. Mansor, et al., Random satisfiability: A higher-order logical approach in discrete Hopfield neural network, *IEEE Access*, **9** (2021), 50831−50845. https://doi.org/10.1109/ACCESS.2021.3068998
63. R. M. Karp, *Reducibility among combinatorial problems*, Springer Berlin Heidelberg, 2010, 219−241. Available from: https://www.mendeley.com/catalogue/4a2bffde-6de3-3f46-a030-a9b57f8fd14f/.
64. S. A. Cook, *The complexity of theorem proving procedures*, In: Proceedings of the third annual ACM symposium on Theory of computing, New York, 1971, 151−158. https://doi.org/10.1145/800157.805047
65. C. P. Gomes, B. Selman, H. Kautz, *Boosting combinatorial search through randomization*, AAAI/IAAI, **98** (1998), 431−437.
66. Z. C. Zhou, D. Y. Xu, L. J. Lu, Strictly regular random (3, s)-SAT model and its phase transition phenomenon, *J. Beijing Univ. Aeronaut. Astronaut.*, **42** (2016), 2563−2571. https://doi.org/10.13700/j.bh.1001-5965.2015.0845(in Chinese)
67. A. Alway, N. E. Zamri, M. S. M. Kasihmuddin, A. Mansor, S. Sathasivam, Palm oil trend analysis via logic mining with discrete hopfield neural network, *Pertanika J. Sci. Tech.*, **28** (2020).

68. X. Xie, S. Sathasivam, H. Ma, Modeling of 3 SAT discrete Hopfield neural network optimization using genetic algorithm optimized K-modes clustering, *AIMS Math.*, **9** (2024), 28100−28129. https://doi.org/10.3934/math.20241363