AIMS Mathematics, 9(10): 28100–28129.
DOI: 10.3934/math.20241363
Received: 07 August 2024
Revised: 05 September 2024
Accepted: 18 September 2024
Published: 27 September 2024

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

Research article

Modeling of 3 SAT discrete Hopfield neural network optimization using genetic algorithm optimized K-modes clustering

Xiaojun Xie[1,2], Saratha Sathasivam[2,*] and Hong Ma[2,3]

[1] School of General Education, Guangzhou College of Technology and Business, 510850, Guangzhou, China
[2] School of Mathematical Sciences, Universiti Sains Malaysia (USM), Penang 11800, Malaysia
[3] School of Financial Mathematics and Statistics, Guangdong University of Finance, 510521, Guangzhou, China

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

Abstract: The discrete Hopfield neural network 3-satisfiability (DHNN-3SAT) model represents an innovative application of deep learning techniques to the Boolean SAT problem. Existing research indicated that the DHNN-3SAT model demonstrated significant advantages in handling 3SAT problem instances of varying scales and complexities. Compared to traditional heuristic algorithms, this model converged to local minima more rapidly and exhibited enhanced exploration capabilities within the global search space. However, the model faced several challenges and limitations. As constraints in SAT problems dynamically increased, decreased, or changed, and as problem scales expanded, the model's computational complexity and storage requirements may increase dramatically, leading to reduced performance in handling large-scale SAT problems. To address these challenges, this paper first introduced a method for designing network synaptic weights based on fundamental logical clauses. This method effectively utilized the synaptic weight information from the original SAT problem within the DHNN network, thereby significantly reducing redundant computations. Concrete examples illustrated the design process of network synaptic weights when constraints were added, removed, or updated, offering new approaches for managing the evolving constraints in SAT problems. Subsequently, the paper presented a DHNN-3SAT model optimized by genetic algorithms combined with K-modes clustering. This model employed genetic algorithm-optimized K-modes clustering to effectively cluster the initial space, significantly reducing the search space. This approach minimized the likelihood of redundant searches and reduced the risk of getting trapped in local minima, thus improving search efficiency. Experimental tests on benchmark datasets showed that the proposed model outperformed traditional DHNN-3SAT models, DHNN-3SAT models

combined with genetic algorithms, and DHNN-3SAT models combined with imperialist competitive algorithms across four evaluation metrics. This study not only broadened the application of DHNN in solving 3SAT problems but also provided valuable insights and guidance for future research.

Keywords: discrete Hopfield neural network; 3SAT; genetic algorithm; K-modes clustering
Mathematics Subject Classification: 03B52, 68T27, 68N17, 68W99

1. Introduction

The Boolean satisfiability (SAT) problem is a classical issue in computational complexity theory and has been a significant research subject in computer science and artificial intelligence since the 1970s [1]. In 1971, S. A. Cook [2] proved that the SAT problem is the world's first NP-complete problem, meaning any NP problem can be reduced to the SAT problem for a polynomial-time solution. The SAT problem serves as a benchmark for the difficulty of a category of problems known as the core of NP-complete problems. It plays a crucial role in various areas of computer science, including theoretical computer science, complexity theory, cryptosystems, and artificial intelligence [3–6]. With the advancements in computer hardware performance and algorithm design, traditional SAT solvers have become effective in many practical applications [7–10]. However, as problems grow in size and complexity, traditional methods often face challenges such as inefficiency and high consumption of computational resources. This has prompted researchers to explore new solution methods and techniques. Among these, the discrete Hopfield neural network (DHNN) [11], a classical neural network model, has shown significant potential and effectiveness in solving combinatorial optimization problems since its inception. Hopfield [11] demonstrated the stability of network dynamics, highlighting that the evolution of network states is essentially a process of energy minimization. When the association weights are symmetric, the system reaches a stable state. This stable equilibrium point aligns with the correct storage state, providing a clear physical explanation for associative memory. The network, by emphasizing the collective function of neurons from a systems perspective, offers preliminary insights into the nature of associative memory. Due to its robust memory capabilities and parallel processing power, the DHNN is particularly effective in addressing combinatorial optimization problems such as SAT [12–14].

In the study of SAT problems, the 3-satisfiability (3SAT) logic has received significant attention from researchers because higher-order Boolean SAT can be converted or reduced to the 3SAT form [15]. In the 3SAT problem, each clause contains three literals, making it more complex and closer to practical logic constraint problems. To address the 3SAT problem, researchers have mapped the variables and clauses of a Boolean formula into the neurons and energy functions of a discrete Hopfield network. In this network, each variable and clause is encoded as a neuron's state and connection weights [16–18]. A satisfying solution to the Boolean formula is then found by adjusting the neuron states to minimize the energy function. This method of solving the 3-SAT problem implemented in a DHNN is referred to as the DHNN-3SAT model. The DHNN-3SAT model has garnered extensive attention and research interest due to its significant improvement in solving ability and effectiveness on 3SAT problems [19,20]. Early research efforts focused on basic discrete Hopfield network structures, utilizing simple connection weights and update rules. As research progressed, scholars proposed various improvement and optimization strategies to enhance the network's performance and efficiency. In 1992, Wan Abdullah successfully integrated special logic

programming as symbolic rules into a DHNN [21], and in 2011, Sathasivam and Abdullah extended this approach and formally named it the Wan Abdullah method (WA method) [22]. In 2014, Sathasivam et al. embedded higher-order SAT into DHNN [23]. Kasihmuddin et al. [24] applied k-satisfiability planning in DHNN. In 2017, Mansor et al. [25] demonstrated the hybrid use of the DHNN artificial immune system for the 3-SAT problem. Subsequently, Kasihmuddin et al. [26] proposed a genetic algorithm for k-satisfiability logic programming based on DHNN. In 2021, Mansor and Sathasivam [12] proposed a DHNN-3SAT optimal performance index. In 2023, Azizan and Sathasivam [27] proposed a DHNN model with a 3SAT fuzzy logic model of DHNN. However, as researchers delved deeper into the DHNN-SAT model, they found that its computational efficiency is not optimal for large-scale problems due to the inherent limitations of the DHNN, with a tendency to fall into local minima. To address these issues, researchers have been working to integrate heuristic algorithms into the optimization process [28–31] to enhance the accuracy of the DHNN-SAT model. Currently, these research methods are achieving high global minimum ratios in DHNN-SAT models with fewer neurons. By adjusting the structure and parameters of the neural network, researchers [32] have been exploring various model variations and optimization strategies to further enhance the performance and generalizability of the model. These efforts not only offer a new perspective and approach to understanding and solving SAT problems but also make significant contributions and provide inspiration for the application of DHNNs in combinatorial optimization and discrete problem-solving.

Although the DHNN-3SAT model has been successful in addressing certain problems, it still has some challenges and limitations. First, the model's computational complexity and storage requirements may increase significantly with the problem size, leading to performance issues when dealing with large-scale SAT problems. Second, the model's training and optimization process may be sensitive to parameter tuning and initialization, necessitating more experimental validation and tuning. Third, it may take a longer time to reach a stable solution when dealing with complex problems, which can impact its practical application in engineering and other fields. Lastly, in real-life scenarios, the constraints of SAT problems often change over time, leading to the need for network redesign and the generation of a large number of redundant computations with the increase, decrease, and update of large-scale constraints, ultimately limiting the traditional DHNN-3SAT model's performance.

To address the changing constraints of the SAT problem and the increasing size and complexity of the network, this paper proposes a WA method based on basic logical clauses. This method utilizes information about the synaptic weights of the original SAT problem in the DHNN, leading to significant savings in repetitive calculations. In addition, to tackle the issue of increasing Boolean variables and logical clauses leading to a rapidly expanding solution space and the traditional DHNN-WA model being prone to oscillations and local minima, this paper introduces a DHNN 3SAT model, based on a genetic algorithm-optimized K-modes clustering. This approach uses the genetic optimization K-modes clustering algorithm to cluster the initial space, reducing the retrieval space and avoiding repeated searches, thus improving retrieval efficiency.

The paper is organized as follows: Section 2 introduces the knowledge related to the research, including 3SAT and DHNN. Section 3 details the implementation and workflow for determining the synaptic weights of the DHNN 3SAT model using the WA method. To address the issue of a large number of redundant computations caused by the changing constraints of the 3SAT problem, the basic logic clause-based WA (BLC-WA) method is proposed. Section 4 introduces the K-modes clustering algorithm optimized by a genetic algorithm. Section 5 details the implementation steps and development process of the DHNN 3SAT model based on genetic algorithm-optimized K-modes

clustering. Section 6 presents an experimental comparative analysis of the DHNN-3SAT model based on the genetic optimization K-modes clustering algorithm (DHNN-3SAT-GAKM) model proposed in this paper, and the three models DHNN-3SAT-WA, the DHNN-3SAT-WA model by using the Genetic Algorithm (DHNN-3SAT-GA), and the DHNN-3SAT-WA model by using Competition Algorithm (DHNN-3SAT-ICA), which are comprehensively evaluated using four evaluation metrics. Finally, Section 7 summarizes the work presented in this paper.

2. Theoretical background

2.1. Boolean 3SAT Logic

Definition 2.1. 3SAT is a satisfiability problem for a set of logical clauses consisting strictly of 3 literal variables. 3SAT problems can be expressed in 3 conjunctive normal forms (CNFs). Let the set of Boolean variables be $\{S_1, S_2, \cdots, S_n\}$ and the set of logical clauses be $\{C_1, C_2, \cdots, C_m\}$, then the general form of a CNF 3SAT formula P containing n Boolean variables and m logical clauses is defined as:

$$P = \wedge_{k=1}^{m} C_k, \tag{1}$$

where the clause C_k consists of 3 literals connected by the classical operator or ($\vee$): $C_k = Z_{(k,1)} \vee Z_{(k,2)} \vee Z_{(k,3)}$, and the state of the literals can be either a positive variable or the negation of a positive variable, i.e., $Z_{(k,i)} = S_j$ or $Z_{(k,i)} = \neg S_j, 1 \leq k \leq m, 1 \leq i \leq 3, 1 \leq j \leq n$. Each literal variable takes on the binary discrete value $\{1, -1\}$, where 1 denotes true and -1 denotes false. Each clause in 3SAT contains unique variables, meaning there is no repetition of the same variable (variable or negation of a variable) in clause C_k. Additionally, there are no repeated logical clauses within logical rules.

The problem denoted by 3SAT can be formally described as follows: Given a 3SAT formula, the task is to determine if there is an assignment of Boolean variables that makes the entire formula true. In particular, each clause in the formula must have at least one true literal for the whole formula to be true.

Instance. Suppose that for given a 3SAT problem, the conversion to the CNF 3SAT formula is:

$$P = (S_1 \vee S_2 \vee S_3) \wedge (\neg S_1 \vee S_2 \vee S_3) \wedge (S_1 \vee \neg S_2 \vee S_3) \wedge (S_1 \vee S_2 \vee \neg S_3) \wedge (\neg S_1 \vee \neg S_2 \vee S_3) \wedge (\neg S_1 \vee S_2 \vee \neg S_3) \wedge (S_1 \vee \neg S_2 \vee \neg S_3) \wedge (S_1 \vee S_2 \vee S_4). \tag{2}$$

In Eq (2), P is satisfiable if there exists a set of values for the variable S_1, S_2, S_3, S_4 such that $P = 1$; otherwise, P is unsatisfiable.

The problem regarding 3SAT is a fundamental issue in computational complexity theory. Its NP-completeness and wide range of applications make it a crucial subject of research in both theoretical and practical contexts. Through a thorough examination of the 3SAT problem, a better understanding of computational complexity theory can be achieved, and effective tools and methods for solving practical problems can be provided. This study contributes to the advancement of computer science by exploring solution methods for 3SAT problems.

2.2. DHNN

Neural networks can be divided into two types based on the flow of information: Feed-forward and feedback neural networks. The output of a feedforward neural network depends only on the current input vector and weight matrix, independent of the network's previous input state. An

example of this is the commonly used back propagation (BP) neural network. In 1982, physicist professor J. J. Hopfield proposed [11] a single-layer feedback neural network, later called the Hopfield neural network. This network is of two types: Continuous Hopfield neural network (CHNN) and discrete Hopfield neural network (DHNN) [33,34]. DHNN has garnered significant attention due to its concise network structure and powerful memory function. It holds potential practical value in image recovery and optimization problems [35–37]. Figure 1 depicts the topology of a DHNN network with n neurons. Each neuron is functionally identical and interconnected in pairs. The neurons are represented by the set $O = \{o_1, o_2, \cdots, o_n\}$, and their corresponding states are denoted by the vector $X = (x_1, x_2, \cdots, x_n)$, and the value of x_i takes binary discrete values, typically $\{-1,1\}$ or $\{0,1\}$. The state of the network is described as $X(t) = \big(x_1(t), x_1(t), \cdots, x_n(t)\big)$ at time t, and the DHNN is stimulated by an external input to start its evolution. The outputs of localized lots are generated before the final state. The output of the local lot of the double link is:

$$h_i(t) = \sum_j w_{ij} x_i(t) - w_i, \tag{3}$$

where w_i denotes a predefined threshold. The output of higher-order linked local lots is represented by Eq (4) as proposed by Mansor et al [22].

$$h_i(t) = \cdots + \sum_j \sum_k w_{ijk} x_i(t) x_j(t) + \sum_{j=1}^{n} w_{ij} x_i(t) - w_i. \tag{4}$$

The output state of the neuron o_i at the time $t+1$ is denoted as:

$$x_i(t+1) = sgn\big(h_i(t)\big) = \begin{cases} 1, h_i(t) \geq 0, \\ -1, h_i(t) < 0, \end{cases} \tag{5}$$

where "sgn" denotes the sign function and w_{ij} denotes the connection weights of neuron o_i and neuron o_j, with the weights specified as follows.

$$w_{ij} = \begin{cases} w_{ji}, i \neq j, \\ 0, i = j. \end{cases} \tag{6}$$

In the network training phase, the Hebbian rule is usually used to calculate the weights w_{ij} as:

$$w_{ij} = \sum_{s=1}^{m} (2x_i^s - 1)\big(2x_j^s - 1\big), \tag{7}$$

where m denotes the number of samples to be memorized.

The DHNN is essentially a nonlinear dynamical system. The network starts evolving from an initial state, and the DHNN is considered stable when its state no longer changes after a finite number of iterations. In DHNN, stability is determined by introducing the Lyapunov function as the energy function, which serves as an indicator of stability [38]. The system reaches stability when the energy function reaches a minimum point of invariance. The energy function in DHNN is defined as:

$$E(X) = \cdots - \frac{1}{3} \sum_i \sum_j \sum_k w_{ijk}\, x_i x_j x_k - \frac{1}{2} \sum_i \sum_j w_{ij}\, x_i x_j - \sum_i w_i x_i. \tag{8}$$

In 1983, Cohen and S. Grossberg showed that DHNNs evolve with a decreasing energy function and that a stable state of the network corresponds to a minimal value of the energy function. Consequently, for each stable state, we can check whether this state represents a global minimum by determining whether the energy function has reached a minimum [28]. If Eq (9) is satisfied, the stable state is considered a global minimum; otherwise, it is a local minimum.

$$|E(X) - E_{min}| < \delta, \tag{9}$$

where E_{min} denotes the minimum value of the energy function and δ is the user-defined tolerance value.

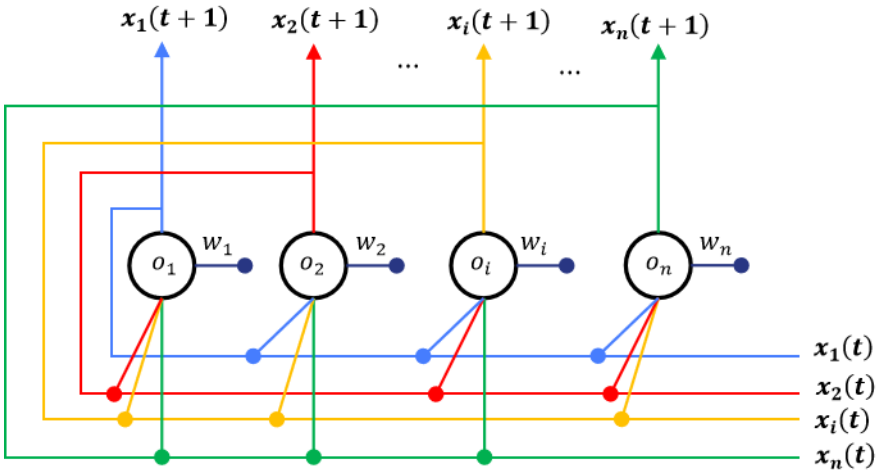


Figure 1. DHNN topology.

3. Design of DHNN-3SAT model weights

In this section, we will start by determining the synaptic weights of the DHNN 3SAT model using the WA method [21]. This method is a computational approach for deriving the synaptic weights of a network by aligning the cost function with the DHNN energy function. Our study acknowledges some challenges in this comparative method of deriving network synaptic weights, particularly as the number of variables and logical clauses increase. Additionally, the addition, deletion, and updating of logical clauses result in a large number of redundant computations. To tackle these issues, this section will outline the cost function of the basic logical clauses and compute the network synaptic weights by establishing the basic logical clauses of the CNF 3SAT formulae. This approach will allow for a more adaptable implementation in computing the network synaptic weights of the 3SAT when incorporated in a DHNN. The method is termed the BLC-WA method. Furthermore, the detailed calculation process using the BLC-WA method will be demonstrated with specific examples as logical clauses are added, deleted, and updated.

3.1. WA method

The WA method introduces a cost function based on propositional logic rules for the first time. It derives the synaptic weights of the network by comparing the cost function with the DHNN energy function, presenting a novel approach to using DHNN for solving the SAT problem. In this study, the WA method is used to incorporate the 3SAT problem into the DHNN for computing the network synaptic weights. The flowchart illustrating the implementation of the WA method is shown in Figure 2. The specific steps are as follows:

Step 1. Given any 3SAT problem, transform it into a CNF 3SAT formula P. Suppose the formula P contains n Boolean variables and m logical clauses.

Step 2. The 3SAT formula P is embedded into the DHNN, and for each Boolean variable, a unique neuron is specified. At moment t, the state of these neurons is denoted by $\{S_1^t, S_2^t, \cdots, S_n^t\}$.

Step 3. Applying De. Morgan's law to obtain $\neg P$. When $\neg P = 0$, correspond to the consistency interpretation of P; when $\neg P = 1$, correspond to the fact that at least one clause of P is not satisfied.

Step 4. Deriving the cost function E_P. When the literal variable in $\neg P$ is represented by $\frac{1}{2}(1 - S_i)$ when it is $\neg S_i$ and $\frac{1}{2}(1 + S_i)$ when it is S_i, the logical clauses are internally connected by the multiplication operation and between logical clauses by addition. This creates the cost function E_P. The magnitude of E_P corresponds to the degree to which all logical clauses are satisfied. When $E_P = 0$ it represents a consistent interpretation of P. A larger value of E_P represents a larger number of unsatisfied logical clauses.

Step 5. Comparing the cost function E_P with the energy function $E(X)$, the DHNN synaptic weight matrix W corresponding to the 3SAT formula P is obtained.

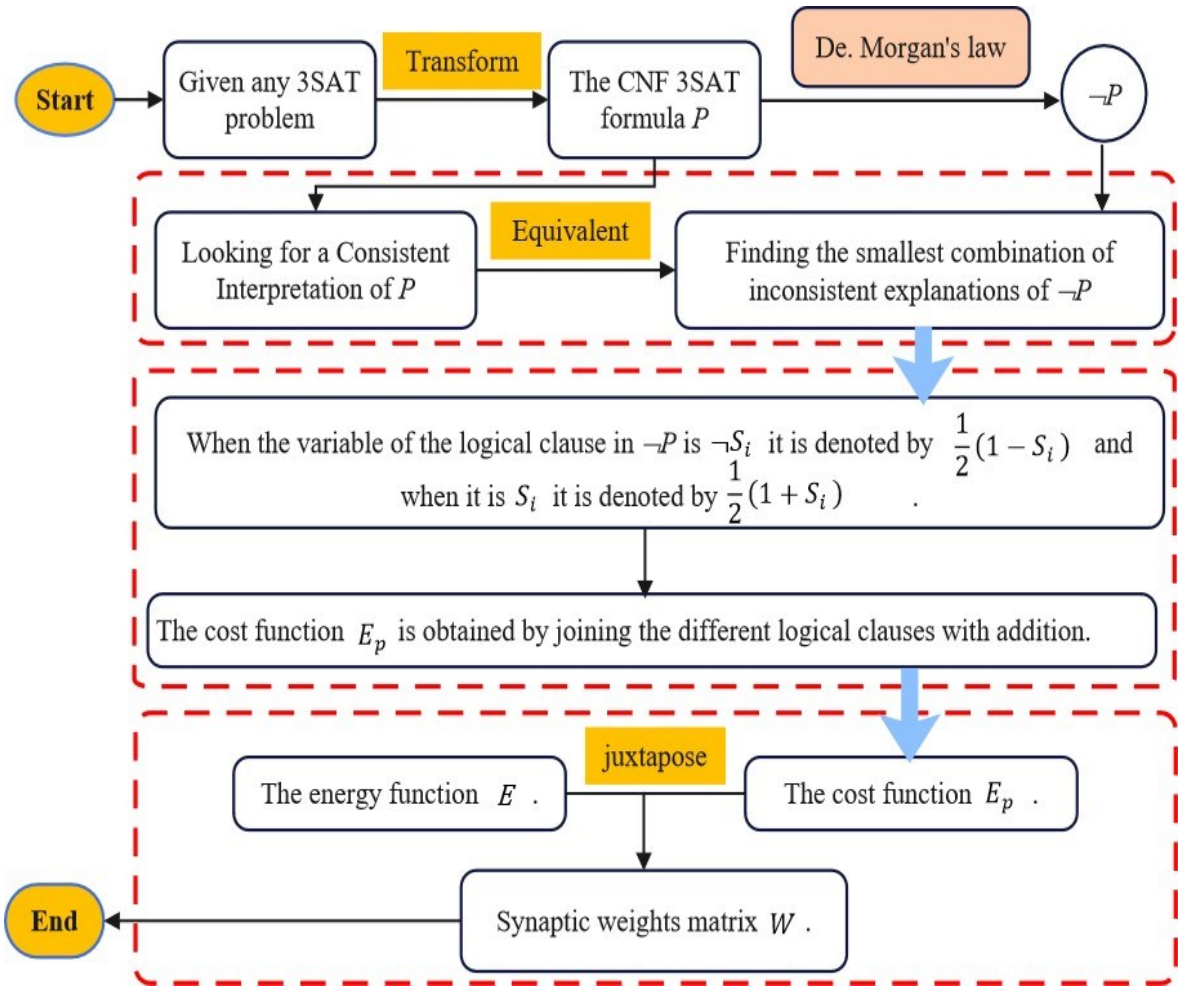


Figure 2. Design a flowchart of the real WA method.

3.2. WA method applied to 3SAT instance

In this section, we use the problem of Eq (2) in Section 2.1 as an example to illustrate the process of computing synaptic weights in the DHNN using the WA method of embedding logical clauses into the DHNN.

To determine whether Eq (2) is satisfiable, the negation of Eq (2) is applied to De Morgan's law, which results in:

$$\neg P = (\neg S_1 \wedge \neg S_2 \wedge \neg S_3) \vee (S_1 \wedge \neg S_2 \wedge \neg S_3) \vee (\neg S_1 \wedge S_2 \wedge \neg S_3) \vee (\neg S_1 \wedge \neg S_2 \wedge S_3) \vee (S_1 \wedge S_2 \wedge \neg S_3) \vee (S_1 \wedge \neg S_2 \wedge S_3) \vee (\neg S_1 \wedge S_2 \wedge S_3) \vee (\neg S_1 \wedge \neg S_2 \wedge \neg S_4). \quad (10)$$

Since seeking a consistent interpretation of the terms of Eq (2) is the same as finding the smallest combination of inconsistent interpretations of Eq (10), the cost function can be defined as follows:

$$E_P = \frac{1}{2}(1 - S_1)\frac{1}{2}(1 - S_2)\frac{1}{2}(1 - S_3) + \frac{1}{2}(1 + S_1)\frac{1}{2}(1 - S_2)\frac{1}{2}(1 - S_3) + \frac{1}{2}(1 - S_1)\frac{1}{2}(1 + S_2)\frac{1}{2}(1 - S_3) + \frac{1}{2}(1 - S_1)\frac{1}{2}(1 - S_2)\frac{1}{2}(1 + S_3) + \frac{1}{2}(1 + S_1)\frac{1}{2}(1 + S_2)\frac{1}{2}(1 - S_3) + \frac{1}{2}(1 + S_1)\frac{1}{2}(1 - S_2)\frac{1}{2}(1 + S_3) + \frac{1}{2}(1 - S_1)\frac{1}{2}(1 + S_2)\frac{1}{2}(1 + S_3) + \frac{1}{2}(1 - S_1)\frac{1}{2}(1 - S_2)\frac{1}{2}(1 - S_4)$$

$$= -\frac{1}{8}S_1S_2S_3 - \frac{1}{8}S_1S_2S_4 - \frac{1}{8}S_1S_3 + \frac{1}{8}S_1S_4 - \frac{1}{8}S_2S_3 + \frac{1}{8}S_2S_4 - \frac{1}{4}S_1 - \frac{1}{4}S_2 - \frac{1}{8}S_3 - \frac{1}{8}S_4 + 1 \quad (11)$$

When the formula of Eq (2) provided The 3SAT formula is satisfied, the cost function E_P reaches the minimum value of 0. At this point, the energy function for the corresponding DHNN converges to the global minimum, causing both the cost function and energy function to reach their minimum values. The network's synaptic weight matrix W_P, embedded in the DHNN by Eq (2), is derived by comparing the cost function (11) with the energy function (8) using the WA method, and the results are shown in Table 1.

Table 1. WA method for 3SAT.

Weights	w_{123}	w_{124}	w_{134}	w_{234}	w_{12}	w_{13}	w_{14}	w_{23}	w_{24}	w_{34}	w_1	w_2	w_3	w_4
P	$\frac{1}{16}$	$\frac{1}{16}$	0	0	0	$\frac{1}{8}$	$-\frac{1}{8}$	$\frac{1}{8}$	$-\frac{1}{8}$	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$

3.3. BLC-WA method

Definition 3.1. For any CNF formula containing n Boolean variables and m logical clauses, it can be viewed as consisting of the following eight basic logical clauses:

$$C_1^l = (S_i \vee S_j \vee S_k), \quad C_2^l = (\neg S_i \vee S_j \vee S_k),$$
$$C_3^l = (S_i \vee \neg S_j \vee S_k), C_4^l = (S_i \vee S_j \vee \neg S_k),$$
$$C_5^l = (\neg S_i \vee \neg S_j \vee S_k), C_6^l = (\neg S_i \vee S_j \vee \neg S_k),$$
$$C_7^l = (S_i \vee \neg S_j \vee \neg S_k), C_8^l = (\neg S_i \vee \neg S_j \vee \neg S_k), \quad (12)$$

where l denotes the index at which the basic logical clause is looked up.

Applying De Morgan's law to the eight basic logical clauses in Eq (12) yields the corresponding negative basic logical clauses:

$$\neg C_1^l = (\neg S_i \wedge \neg S_j \wedge \neg S_k), \quad \neg C_2^l = (S_i \wedge \neg S_j \wedge \neg S_k),$$
$$\neg C_3^l = (\neg S_i \wedge S_j \wedge \neg S_k), C_4^l = (\neg S_i \wedge \neg S_j \wedge S_k),$$
$$\neg C_5^l = (S_i \wedge S_j \wedge \neg S_k), \neg C_6^l = (S_i \wedge \neg S_j \wedge S_k),$$

$$\neg C_7^l = \left(\neg S_i \wedge S_j \wedge S_k\right), \neg C_8^l = \left(S_i \wedge S_j \wedge S_k\right). \tag{13}$$

The consistency clause of each basic logic clause in seeking pair Eq (12) is equal to the minimum of the inconsistency clause of the negated basic logic clause in seeking pair Eq (13). The corresponding cost function for each basic logic clause is defined as follows:

$$\begin{aligned}
&E_{C_1^l} = \tfrac{1}{2}(1-S_i)\tfrac{1}{2}\left(1-S_j\right)\tfrac{1}{2}(1-S_k),\ E_{C_2^l} = \tfrac{1}{2}(1+S_i)\tfrac{1}{2}\left(1-S_j\right)\tfrac{1}{2}(1-S_k),\\
&E_{C_3^l} = \tfrac{1}{2}(1-S_i)\tfrac{1}{2}\left(1+S_j\right)\tfrac{1}{2}(1-S_k),\ E_{C_4^l} = \tfrac{1}{2}(1-S_i)\tfrac{1}{2}\left(1-S_j\right)\tfrac{1}{2}(1+S_k),\\
&E_{C_5^l} = \tfrac{1}{2}(1+S_i)\tfrac{1}{2}\left(1+S_j\right)\tfrac{1}{2}(1-S_k),\ E_{C_6^l} = \tfrac{1}{2}(1+S_i)\tfrac{1}{2}\left(1-S_j\right)\tfrac{1}{2}(1+S_k),\\
&E_{C_7^l} = \tfrac{1}{2}(1-S_i)\tfrac{1}{2}\left(1+S_j\right)\tfrac{1}{2}(1+S_k),\ E_{C_8^l} = \tfrac{1}{2}(1+S_i)\tfrac{1}{2}\left(1+S_j\right)\tfrac{1}{2}(1+S_k).
\end{aligned} \tag{14}$$

Each basic logic clause is embedded into a DHNN separately. When each basic logic clause is satisfiable, the corresponding DHNN converges to the global minimum. At this point, the cost function and the corresponding energy function of the basic logic clause reach their minimum values. By comparing the cost function (14) of the basic logic clauses with the energy function (8), the basic logic clause weight matrix of the 3SAT formula can be derived. This weight matrix is abbreviated as 3SAT-BLCWM, and the results are shown in Table 2.

Table 2. 3SAT-BLCWM.

Weights	C_1^l	C_2^l	C_3^l	C_4^l	C_5^l	C_6^l	C_7^l	C_8^l
w_i	$1/8$	$-1/8$	$1/8$	$1/8$	$-1/8$	$-1/8$	$1/8$	$-1/8$
w_j	$1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$	$-1/8$
w_k	$1/8$	$1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$	$-1/8$	$-1/8$
w_{ij}	$-1/8$	$1/8$	$1/8$	$-1/8$	$-1/8$	$1/8$	$1/8$	$-1/8$
w_{ik}	$-1/8$	$-1/8$	$1/8$	$1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$
w_{jk}	$-1/8$	$1/8$	$-1/8$	$1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$
w_{ijk}	$1/16$	$-1/16$	$-1/16$	$-1/16$	$1/16$	$1/16$	$1/6$	$-1/16$

Any 3SAT formula can be seen as made up of the basic logical clauses in Eq (12). Each logical clause in the 3SAT formula corresponds to a basic logical clause. So, when the DHNN learns a new logical clause, it only needs to identify the corresponding basic logical clauses, then refer to Table 2, and combine and calculate the network weight values to derive the network synaptic weights after adding a new logical clause. Figure 3 illustrates the flowchart of calculating network synaptic weights using the BLC-WA method. The specific steps for calculating network synaptic weights using the BLC-WA method are as follows:

Step 1. Given any 3SAT problem, transform it into CNF 3SAT formula P, which is assumed to contain N Boolean variables and M logical clauses;

Step 2. The 3SAT-BLCWM was established using the WA method (Table 2);

Step 3. Analyze CNF 3SAT formula P to map each logical clause to the basic logical clause;

Step 4. Based on Table 2 (3SAT-BLCWM), the weights corresponding to each logical clause of the 3SAT formula P are found. These weights are then spelled out by columns into the indexed result weight matrix W';

Step 5 The weight matrix W of formula P for 3SAT is obtained by combining and summing the

result weight matrices W' by columns.

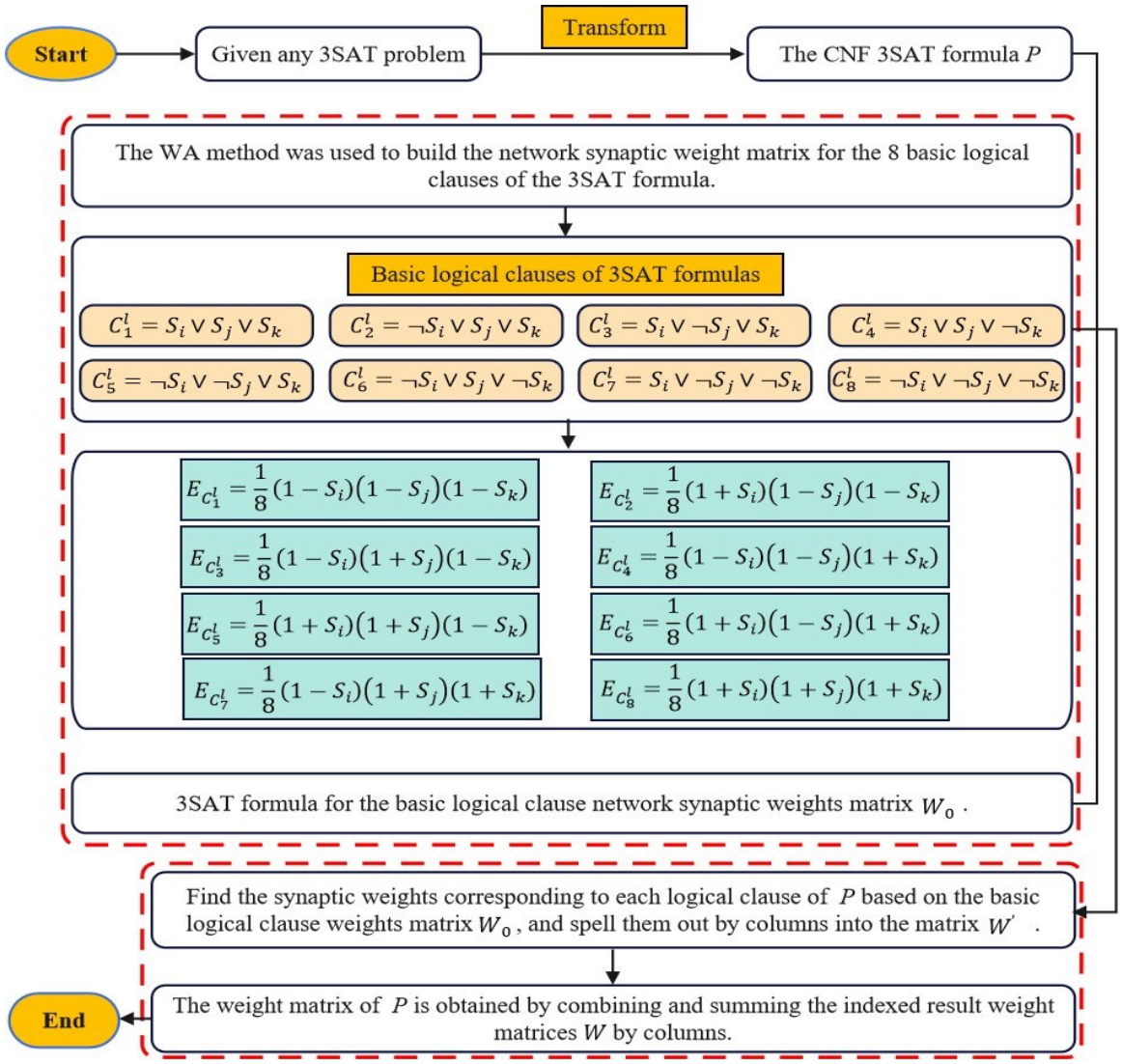


Figure 3. Flowchart of calculating connection weights based on the BLC-WA method.

Next, we compute the 3SAT instances of Section 2.1 using the BLC-WA method based on Eq (2) which can be written in correspondence with the basic logical clause as:

$$P = C_1^1 \wedge C_2^1 \wedge C_3^1 \wedge C_4^1 \wedge C_5^1 \wedge C_6^1 \wedge C_7^1 \wedge C_1^2. \tag{15}$$

According to Table 2 (3SAT-BLCWM), the network synaptic weights corresponding to each logical clause of formula P can be found by indexing the results of the weight matrix W' using the columns. The results are shown in Table 3. The network synaptic weight matrix W_P for the 3SAT formula P can be obtained by merging and adding the indexing results according to the columns, which are also shown in Table 3.

Table 3. Calculation of synaptic weights of CNF 3SAT network based on BLC-WA method.

Weights	C_1^1	C_2^1	C_3^1	C_4^1	C_5^1	C_6^1	C_7^1	C_1^2	P
w_1	$1/8$	$-1/8$	$1/8$	$1/8$	$-1/8$	$-1/8$	$1/8$	$1/8$	$1/4$
w_2	$1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$	$1/8$	$1/4$
w_3	$1/8$	$1/8$	$1/8$	$-1/8$	$1/8$	$-1/8$	$-1/8$	0	$1/8$
w_4	0	0	0	0	0	0	0	$1/8$	$1/8$
w_{12}	$-1/8$	$1/8$	$1/8$	$-1/8$	$-1/8$	$1/8$	$1/8$	$-1/8$	0
w_{13}	$-1/8$	$-1/8$	$1/8$	$1/8$	$1/8$	$-1/8$	$1/8$	0	$1/8$
w_{14}	0	0	0	0	0	0	0	$-1/8$	$-1/8$
w_{23}	$-1/8$	$1/8$	$-1/8$	$1/8$	$1/8$	$-1/8$	$1/8$	0	$1/8$
w_{24}	0	0	0	0	0	0	0	$-1/8$	$-1/8$
w_{34}	0	0	0	0	0	0	0	0	0
w_{123}	$1/16$	$-1/16$	$-1/16$	$-1/16$	$1/16$	$1/16$	$1/16$	0	$1/16$
w_{124}	0	0	0	0	0	0	0	$1/16$	$1/16$
w_{234}	0	0	0	0	0	0	0	0	0

3.4. Design of weights for dynamic constraints for the 3SAT problem

In SAT problems, the constraints often change, with constraints increasing, decreasing, and updating. The traditional DHNN-SAT method requires redesigning the weights of the SAT problem and constructing a new DHNN when facing this type of problem, which does not utilize the original SAT problem's information. As the original SAT problem's constraints increase, the corresponding original CNF formulation also increases the logical clauses, leading to a large number of redundant computations when dealing with large-scale logical clauses. This severely limits the effectiveness of the traditional DHNN-SAT method in solving problems with large-scale increasing constraints. To address this issue, this study proposes the BLC-WA method, a new design method for the SAT problem with changing constraints. This method utilizes the synaptic weight information of the original SAT problem in DHNN, saving a significant amount of repeated calculations. In the following section, the network synaptic weight design method for SAT problems with increasing, decreasing, and updating constraints will be introduced, providing a new approach for solving SAT problems with constantly changing constraints.

3.4.1. Adding constraints

The addition of constraints to the original SAT problem is equivalent to adding logical clauses to the CNF SAT formula.

There is a 3SAT problem that translates into the CNF 3SAT formula:

$$P = C_1^{l_1} \wedge C_2^{l_2} \wedge \cdots \wedge C_m^{l_m}. \tag{16}$$

When r logical clauses are added, the original CNF 3SAT formula becomes:

$$P_{add} = C_1^{l_1} \wedge C_2^{l_2} \cdots \wedge C_m^{l_m} \wedge C_{m+1}^{l_{m+1}} \wedge C_{m+2}^{l_{m+2}} \wedge \cdots \wedge C_{m+r}^{l_{m+2}}. \tag{17}$$

Figure 4 depicts the flowchart of the BLC-WA method for solving the original 3SAT problem with additional constraints. This method is implemented as follows:

Step 1. Let the original CNF 3SAT formula P become P_{add} by adding r logical clauses (Eq 14);

Step 2. The basic logical clauses were mapped to the additional logical clauses, and the synaptic weights of the additional logical clauses were determined based on Table 2 (3SAT-BLCWM);
Step 3. The synaptic weights of the CNF 3SAT formula P_{add} after adding the r logical clauses were calculated using the following Eq (18).

$$W_{add} = W_P + W_{add(1)} + W_{add(2)} + \cdots + W_{add(r)}. \tag{18}$$

The following is a concrete demonstration of the implementation process using the 3SAT instance from Section 2.1.

Assuming that Eq (2) combines the logical clauses $C_2^2 = \neg S_1 \vee S_2 \vee S_4$ and $C_3^2 = S_1 \vee \neg S_2 \vee S_4$, the new CNF formula at this point is notated as P_{add}, specifically, as follows:

$$P_{add} = C_1^1 \wedge C_2^1 \wedge C_3^1 \wedge C_4^1 \wedge C_5^1 \wedge C_6^1 \wedge C_7^1 \wedge C_1^2 \wedge C_2^2 \wedge C_3^2. \tag{19}$$

In Section 3.3, the synaptic weights (W_P) of the formula P in the network have been obtained using the BLC-WA method. Then, the synaptic weights of the newly added logical clauses (C_2^2 and C_3^2) are obtained by searching for Table 2 (3SAT-BLCWM) and then combined and summed with the synaptic weights (W_P) of the formula P to obtain the new synaptic weights (W_{add}) of the CNF formula 3SAT P_{add}. The calculation results are shown in Table 4.

Table 4. Synaptic weights after adding, subtracting, and updating logical clauses.

Weights	P	C_2^2	C_3^2	P_{add}	C_2^1	C_3^1	P_{dec}	C_7^1	C_1^2	C_8^1	C_1^3	P_{upd}
w_1	$1/4$	$-1/8$	$1/8$	$1/4$	$-1/8$	$1/8$	$1/4$	$1/8$	$1/8$	$-1/8$	0	$-1/8$
w_2	$1/4$	$1/8$	$-1/8$	$1/4$	$1/8$	$-1/8$	$1/4$	$-1/8$	$1/8$	$-1/8$	$1/8$	$1/4$
w_3	$1/8$	0	0	$1/8$	$1/8$	$1/8$	$-1/8$	$-1/8$	0	$-1/8$	$1/8$	$1/4$
w_4	$1/8$	$1/8$	$1/8$	$3/8$	0	0	$1/8$	0	$1/8$	0	$1/8$	$1/8$
w_{12}	0	$1/8$	$1/8$	$1/4$	$1/8$	$1/8$	$-1/4$	$1/8$	$-1/8$	$-1/8$	0	$-1/8$
w_{13}	$1/8$	0	0	$1/8$	$-1/8$	$1/8$	$1/8$	$1/8$	0	$-1/8$	0	$-1/8$
w_{14}	$-1/8$	$-1/8$	$1/8$	$-1/8$	0	0	$-1/8$	0	$-1/8$	0	0	0
w_{23}	$1/8$	0	0	$1/8$	$1/8$	$-1/8$	$1/8$	$1/8$	0	$-1/8$	$-1/8$	$-1/4$
w_{24}	$-1/8$	$1/8$	$-1/8$	$-1/8$	0	0	$-1/8$	0	$-1/8$	0	$-1/8$	$-1/8$
w_{34}	0	0	0	0	0	0	0	0	0	0	$-1/8$	$-1/8$
w_{123}	$1/16$	0	0	$1/16$	$-1/16$	$-1/16$	$3/16$	$1/16$	0	$-1/16$	0	$-1/16$
w_{124}	$1/16$	$-1/16$	$-1/16$	$-1/16$	0	0	$1/16$	0	$1/16$	0	0	0
w_{234}	0	0	0	0	0	0	0	0	0	0	$1/16$	$1/16$

3.4.2. Declining constraints

Setting the original CNF 3SAT formula (16) reduces d logical clauses, and the original CNF 3SAT formula becomes:

$$P_{dec} = C_1^{l_1} \wedge C_2^{l_2} \wedge \cdots \wedge C_{m-d}^{l_{m-d}}. \tag{20}$$

The flowchart for solving the original 3SAT problem with reduced constraints based on the BLC-WA method is also shown in Figure 4. It is implemented as follows:
Step 1. The original CNF 3SAT formula P is reduced by d logical clauses to P_{dec};
Step 2. The basic logical clauses were mapped to the reduced logical clauses, and the synaptic weights of the reduced logical clauses were determined based on Table 2 (3SAT-BLCWM);
Step 3. The synaptic weights of the CNF 3SAT formula P_{dec} after declining the d logical clauses

were calculated using the following Eq (21).

$$W_{dec} = W_P - W_{dec(1)} - W_{dec(2)} - \cdots - W_{dec(d)}. \tag{21}$$

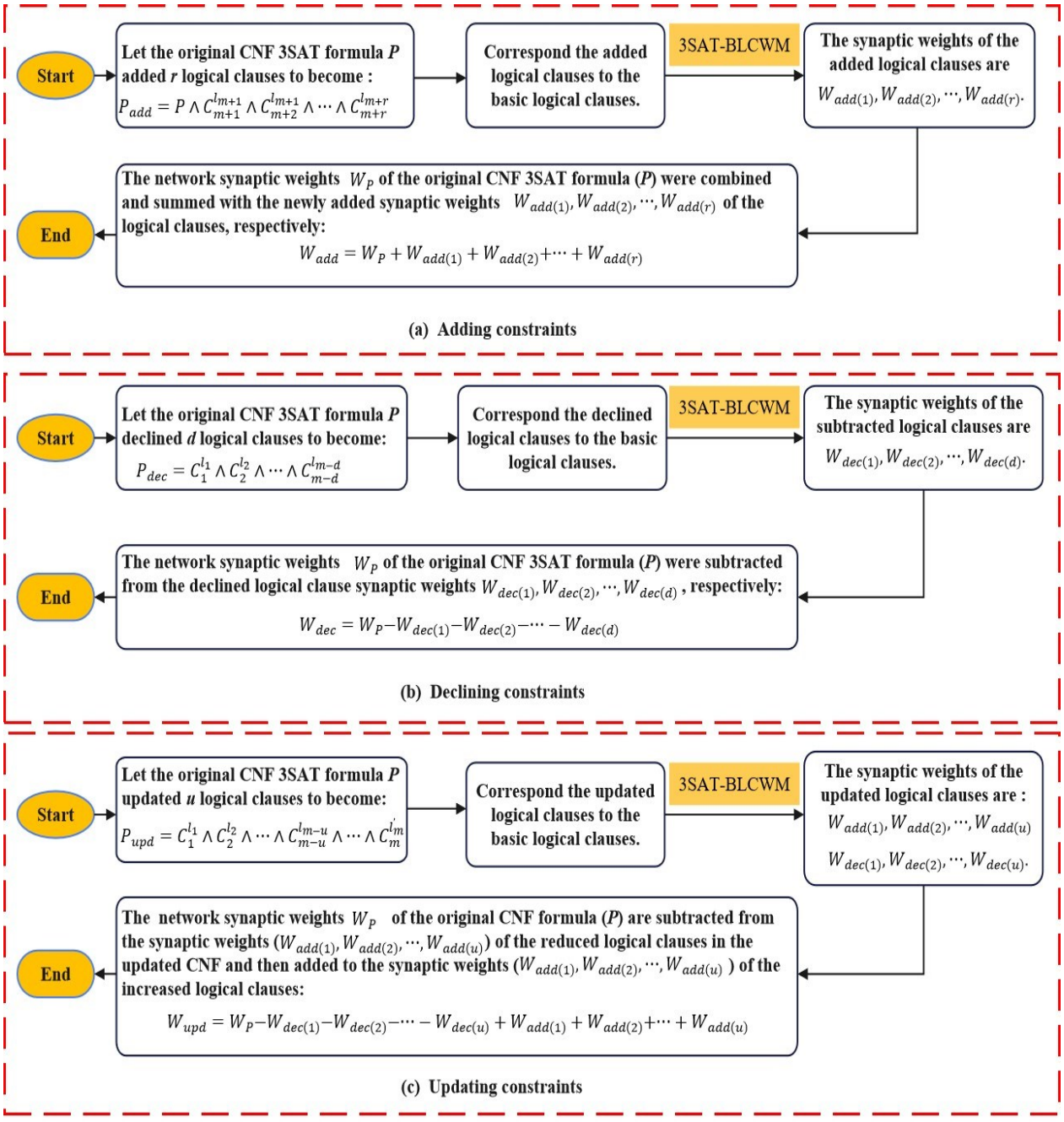


Figure 4. Flowchart of synaptic weights design based on state constraints of BLC-WA method.

The following is a concrete demonstration of the implementation process using the 3SAT instance from Section 2.1.

Assuming that Eq (2) reduces the logical clauses $C_2^1 = \neg S_1 \vee S_2 \vee S_3$ and $C_3^1 = S_1 \vee \neg S_2 \vee S_3$, the new CNF formula at this point is notated as P_{dec}, specifically, as follows:

$$P_{dec} = C_1^1 \wedge C_4^1 \wedge C_5^1 \wedge C_6^1 \wedge C_7^1 \wedge C_1^2. \tag{22}$$

To start, the synaptic weights for the reduced logical clauses C_2^1 and C_3^1 are determined by searching for Table 2 (3SAT-BLCWM). Next, the synaptic weights W_P of the original 3SAT formula P are subtracted from the synaptic weights of the reduced logical clauses. This process yields the synaptic weights W_{dec} for the new CNF 3SAT formula P_{dec}. The computational results are also displayed in Table 4.

3.4.3. Updating constraints

When the original CNF formula (16) is updated with u logical clauses, it can be regarded as a reduction of u logical clauses from the original formula and the addition of u new logical clauses. The updated CNF formula is:

$$P_{upd} = C_1^{l_1} \wedge C_2^{l_2} \wedge \cdots \wedge C_{m-u}^{l_{m-u}} \wedge C_{m-u+1}^{l'_{m-u+1}} \wedge C_{m+2}^{l'_{m-u+2}} \wedge \cdots \wedge C_m^{l'_m} \tag{23}$$

The flowchart when updating the constraints based on the BLC-WA method is also shown in Figure 4, which is implemented as follows:

Step 1. The original CNF 3SAT formula P is updated by u logical clauses to P_{upd};

Step 2. The basic logical clauses were mapped to the updated logical clauses, and the synaptic weights of the updated logical clauses were determined based on Table 2 (3SAT-BLCWM);

Step 3. The synaptic weights of the CNF 3SAT formula P_{upd} after updating the u logical clauses were calculated using the following Eq (24).

$$W_{upd} = W_P - W_{dec(1)} - W_{dec(2)} - \cdots - W_{dec(u)} + W_{add(1)} + W_{add(2)} + \cdots + W_{add(u)}. \tag{24}$$

The following is a concrete demonstration of the implementation process using the 3SAT instance from Section 2.1.

Suppose the logical clauses $C_7^1 = S_1 \vee \neg S_2 \vee \neg S_3$ and $C_1^2 = S_1 \vee S_2 \vee S_4$ in the original CNF 3SAT formula P are updated to $C_8^1 = \neg S_1 \vee \neg S_2 \vee \neg S_3$ and $C_1^3 = S_2 \vee S_3 \vee S_4$, and the updated CNF 3SAT formula is now denoted as P_{upd}, specifically for:

$$P_{upd} = C_1^1 \wedge C_2^1 \wedge C_3^1 \wedge C_4^1 \wedge C_5^1 \wedge C_6^1 \wedge C_8^1 \wedge C_1^3. \tag{25}$$

To begin, find the synaptic weights of logical clauses C_7^1, C_1^2, C_8^1 and C_1^3 by searching for Table 2 (3SAT-BLCWM). Then, subtract the synaptic weights of logical clause C_7^1, C_1^2 from the original SAT formula P. Finally, the synaptic weights of logical clause C_8^1,C_1^3 are added to obtain the network synaptic weights W_{upd} of the updated 3SAT formula P_{upd}. The results of the computation are also displayed in Table 4.

4. Optimized K-modes clustering algorithm

4.1. K-modes clustering algorithm

The K-modes clustering algorithm is a method specifically designed for handling discrete data [39–42]. It extends the traditional K-means algorithm, which is mainly used for datasets with continuous attributes. The K-modes algorithm uses the Hamming distance as a metric [43], where this distance measures the number of differing attribute values between two sample points. In this algorithm, the Hamming distance is computed by adding the number of different attribute values

between two samples, representing the degree of difference for a given sample compared to a clustering center. Finally, the samples are classified into the category that belongs to the clustering center with the smallest degree of difference. We can see the clustering process of the K-modes algorithm in Figure 5.

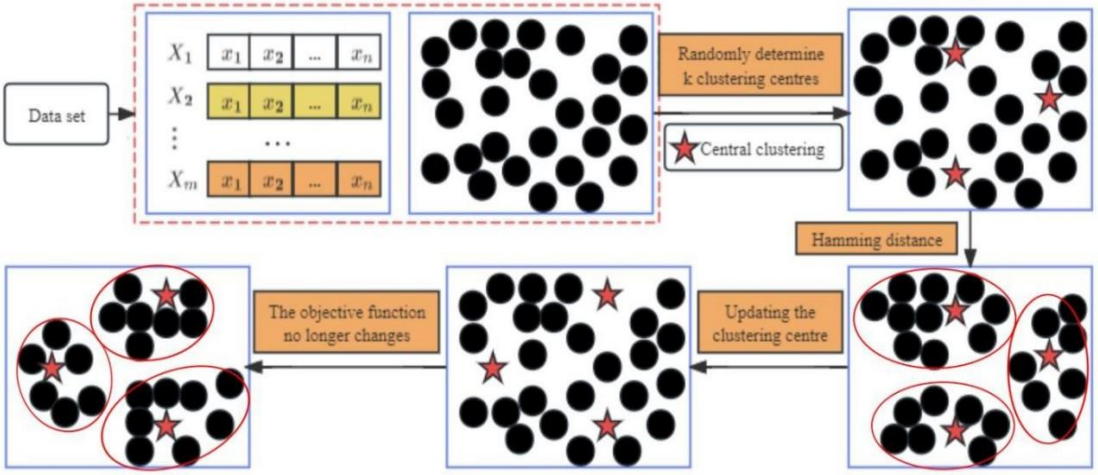


Figure 5. Clustering process of K-modes clustering algorithm.

Let $X = \{X_1, X_2, \cdots, X_m\}$ represent the set of samples to be clustered, and $X_i = (x_1, x_2, \cdots, x_n)$ represent the n-dimensional vector with each component taking discrete values. $Z = \{Z_1, Z_2, \cdots, Z_k\}$ represents the clustering center and $Z_j = (z_1, z_2, \cdots, z_n), j = 1,2, \cdots, k$. The objective function of the K-modes clustering algorithm is defined as:

$$F(\Phi, Z) \sum_{j=1}^{k} \sum_{i=1}^{m} \phi_{ij} D\left(X_i, Z_j\right), \tag{26}$$

where $\phi_{ij} \in \{0,1\}$, $\sum_{j=1}^{k} \phi_{ij} = 1, 1 \leq i \leq n, \Phi$ is the matrix of one, k denotes the number of clusters, $\phi_{ij} = 1$ if the ith object is classified in the j-th class, otherwise $\phi_{ij} = 0$. Z_j is the center of the j-th class. $D\left(X_i, Z_j\right)$ denotes the computation of the Hamming distance between X_i and Z_j:

$$D\left(X_i, Z_j\right) = \sum_{i=1}^{n} d(x_i, z_i), \tag{27}$$

where $d(x_i, z_i) = \begin{cases} 0, x_i = z_i \\ 1, x_i \neq z_i \end{cases}$.

The classification process must meet the following conditions: (1) every family must contain at least one sample; (2) each sample must belong to one and only one class. The fundamental steps of the K-modes clustering algorithm are as follows:

Step 1. Randomly identifying k clustering centers $Z_1, Z_2, \cdots, Z_k$.

Step 2. For each sample $X_i (i = 1,2, \cdots, m)$ in the dataset, its Hamming distance from the k clustering centers is calculated separately using Eq (27), and the sample X_i is classified into the category closest to the centroid.

Step 3. After dividing all the samples into clusters, the cluster center "Z_j" is recalculated, and each center component is updated to its plural.

Step 4. Repeat the process of Steps 2 and 3 above until the objective function F no longer changes.

4.2. K-modes clustering algorithm optimized by genetic algorithm

To address the limitations of the K-modes clustering algorithm, which make it difficult to determine the optimal number of clusters and easy to get stuck at a local optimum, researchers have incorporated a genetic algorithm with adaptive global optimization search capabilities into the K-modes clustering algorithm [44,45]. This involves using a fitness function to carry out genetic operations, primarily mutation, to automatically learn the cluster centroids for the K-modes algorithm. Figure 6. shows the workflow diagram of the K-modes clustering algorithm for genetic optimization, which was developed in the following steps:

Step 1. Parameter initialization. Set relevant parameters: Initial cluster number k, population size m, crossover probability p_c, variation probability p_m, maximum number of iterations t.

Step 2. Randomly generate the initial population. Randomly generate k initial clustering centers $Z_1, Z_2, \cdots, Z_k$ as initial population individuals.

Step 3. Take the population individual $Z_1, Z_2, \cdots, Z_k$ as the clustering center and use K-modes clustering algorithm for clustering.

Step 4. Calculate the fitness value of individuals in the population. Here the fitness function is defined as follows:

$$f = \frac{D_{min}}{\overline{D}(X)}, \tag{28}$$

where D_{min} is the minimum class spacing and $\overline{D}(X)$ is the average class spacing which is defined as follows:

$$D_{min} = \min_{i,j=1} D(Z_i, Z_j). \tag{29}$$

$$\overline{D}(X) = \frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{m_j}\frac{D(X_i, Z_j)}{m_j}. \tag{30}$$

This fitness function is based on the idea that class separation should be maximized while intra-class spacing should be minimized. In other words, the goal is to maximize the distance between classes (D_{min}) and minimize the variability within classes ($\overline{D}(X)$). Throughout the evolutionary process, the individual population size is represented by the k value. If the k value is less than the optimal number of clusters, increasing k leads to a decrease in D_{min} and $\overline{D}(X)$, but the clustering division is not optimal. The decrease in $\overline{D}(X)$ is more significant than D_{min}, resulting in an increase in the fitness function value. Conversely, if the k value exceeds the optimal number of clusters, the change in $\overline{D}(X)$ is not significant, and the intra-class spacing becomes very small due to secondary clustering. As a result, D_{min} becomes very small, leading to a decrease in the overall fitness function value. Therefore, this fitness function can guide the k value toward the optimal number of clusters when the initial clustering center is optimized.

Step 5. Perform selection, crossover, and mutation operations to generate a new generation population.

Step 6. Repeat Step 3 to Step 5 until the maximum number of iterations is reached.

Step 7. Calculate the fitness value for each individual in the population and select the output with the highest fitness value.

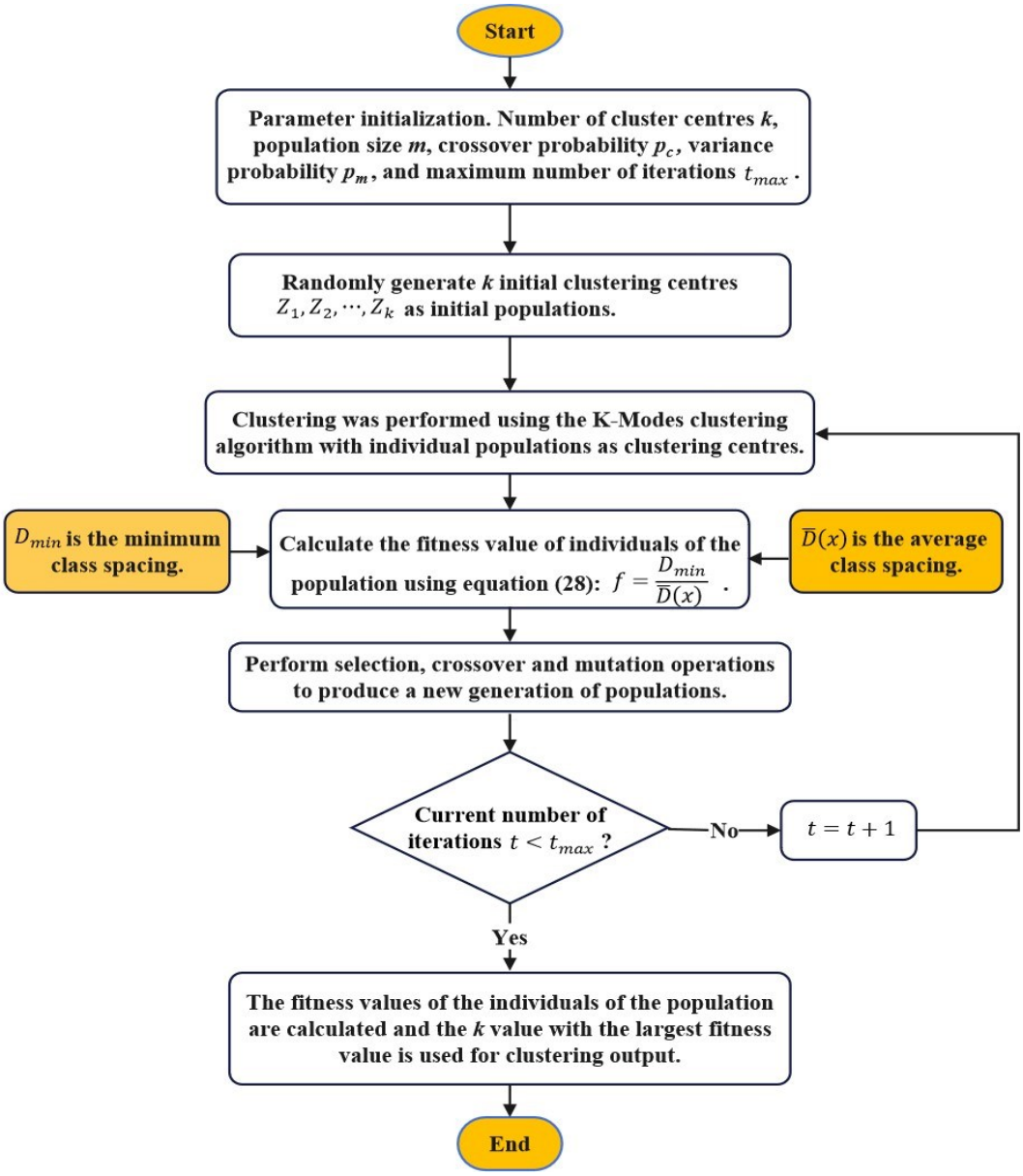


Figure 6. Workflow diagram of K-modes clustering optimized by genetic algorithm.

5. Development of DHNN-3SAT model based on genetic optimization K-modes clustering algorithm

The conventional DHNN-3SAT-WA model uses an exhaustive search (ES) during the retrieval phase [46], aiming to conduct a random search among individual candidate solutions to find a consistent interpretation that satisfies the 3SAT terms. Some researchers have proposed optimizing the traditional DHNN-3SAT-WA model by using heuristic algorithms such as the GA and ICA, denoted as DHNN-3SAT-GA [26] and DHNN-3SAT-ICA [38]. These methods can expedite the search for global or feasible solutions. However, unguided random initial assignment of candidate solutions leads to numerous repeated invalid solutions and fails to converge, often falling into a local optimum after DHNN evolution. Furthermore, as the number of Boolean variables and logical clauses increases, the size and logical complexity of the network expands, resulting in a rapid growth of the solution space. This makes the model susceptible to oscillations and more likely to land in

local minima. Therefore, reducing the DHNN-3SAT-WA model's search time and preventing it from falling into local minima is a significant challenge in this field. The implementation process of the traditional DHNN-3SAT-WA, DHNN-3SAT-GA, and DHNN-3SAT-ICA models is depicted in Figure 7.

This study proposes a new solution to address these challenges: the DHNN-3SAT model based on the genetic optimization K-modes clustering algorithm referred to as DHNN-3SAT-GAKM. In this model, candidate solutions in the allocation space are clustered using the K-modes clustering algorithm, leading to initial allocation through a random search from each class. By reducing repeated initial candidate solutions and avoiding local optima to some extent, this process accelerates the search for the global minimum, improving the efficiency of global minimum retrieval. To determine the optimal number of clusters for the K-modes clustering algorithm, the genetic algorithm with adaptive global optimization search capability is introduced. The number of clusters is determined by calculating the value of the constructed fitness function, further enhancing global search capability.

The DHNN-3SAT-GAKM model aims to find a consistent set of Boolean variable values for the 3SAT problem. During the model's initialization phase, each neuron in the DHNN is connected to a specific Boolean variable in the CNF, and the connection weights represent the relationship between the variable and the clause. A WA method using basic logical clauses will be employed to determine the cost during the learning phase. In the retrieval phase, the DHNN is utilized to evolve, update, and iterate until the network reaches a stable equilibrium state, signified by a minimal energy function value. The energy function's primary purpose is to indicate whether this stable state corresponds to a global minimum of the 3SAT problem, which in turn represents a consistent interpretation of the CNF. Please see Figure 8 for the flowchart of the DHNN-3SAT-GAKM model development, and the implementation steps are summarized as follows:

Step 1. Model Preparation. For a given 3SAT problem, transform it into the corresponding CNF formulation, denoted as P. Assume it contains n Boolean variables and m logical clauses. Initialize the optimization algorithm parameters.

Step 2. Each Boolean variable of the 3SAT formula is uniquely assigned a Hopfield neuron in the DHNN design, which consists of n neurons $O = \{o_1, o_2, \cdots, o_n\}$, with the state at moment t denoted $X(t) = \big(x_1(t), x_1(t), \cdots, x_n(t)\big)$.

Step 3. The BLC-WA method was used to calculate the 3SAT formula P synaptic weights and derive its cost function E_p. When $P = 1$, $E_p = 0$, at which time the energy function reaches its minimum value, giving $E_{min} = -\frac{m}{8}$.

Step 4. Generate an initial candidate solution space by randomly creating m initial candidate solutions $\{X_1(t), X_2(t), \cdots, X_m(t)\}$.

Step 5. The initial candidate solution $\{X_1(t), X_2(t), \cdots, X_m(t)\}$ is clustered using the K-modes clustering algorithm based on genetic optimization to obtain the optimal number of clusters k.

Step 6. Determine the candidate subset for retrieval. Candidate subset denoted as $\{Y_1(t), Y_2(t), \cdots, Y_c(t)\}$, where $c = m/k$, $Y_l(t) = \big(y_1(t), x、\ y_2(t), \cdots, y_n(t)\big), l = 1,2, \cdots, c$. $y_i(t)$ corresponds to the state of t at the moment of the neuron o_i.

Step 7. DHNN Evolution. For $Y_l(t) = \big(y_1(t), y_1(t), \cdots, y_n(t)\big)$, $l = 0, t = 0$, state updates are performed using Eq (5) until a stable state is reached. If $Y_l(t+1) \neq Y_l(t)$, then $t = t + 1$, and if $Y_l(t+1) = Y_l(t)$, the network reaches a steady state. Proceed to the next step.

Step 8. Retrieval Phase. Check if the energy of the steady state satisfies $|E - E_{min}| < \delta$. If it does, store the steady-state $Y_l(t)$ as a global minimum. If it doesn't, $l = l + 1$, consider it as a local

minimum and go back to Step 6.

Step 9. Model Evaluation. The model is assessed using the metrics of global minimum ratio, Hamming distance, CPU time, steady-state retrieval rate, and global minimum retrieval rate.

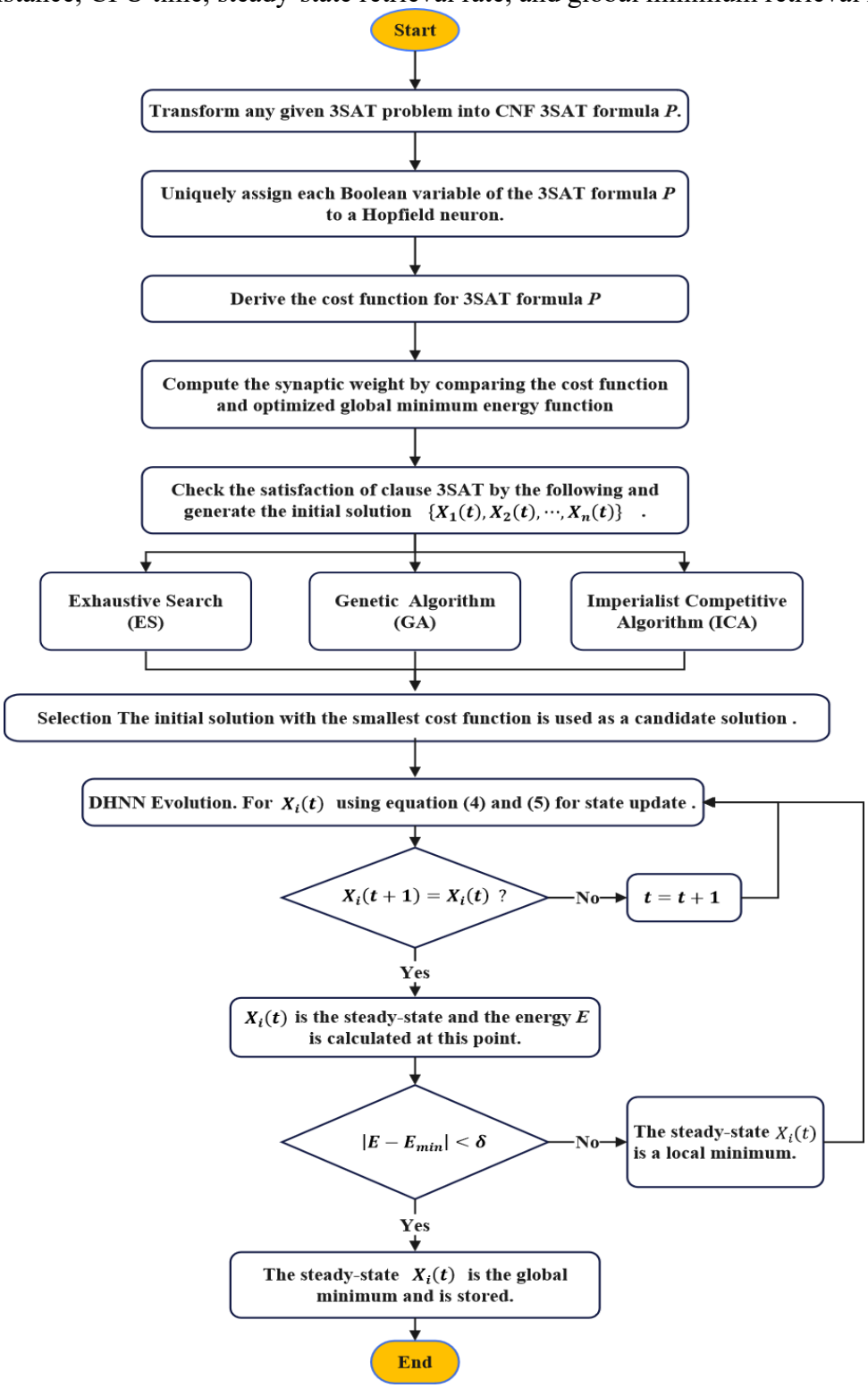


Figure 7. Implementation process of conventional DHNN-3SAT-WA, DHNN-3SAT-GA and DHNN-3SAT-ICA models.

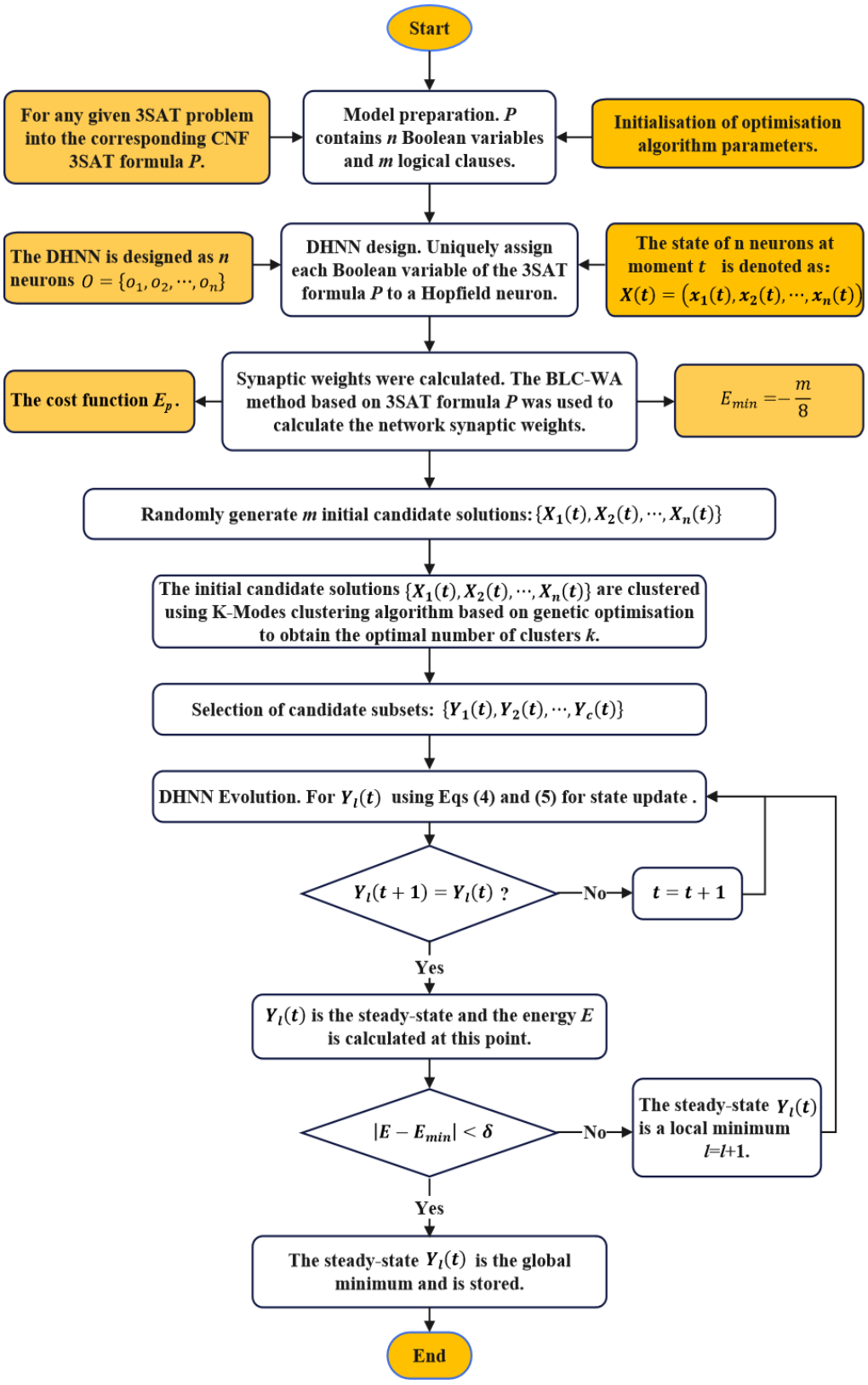


Figure 8. Flowchart for the development of DHNN-3SAT model based on K-modes clustering algorithm for genetic optimization.

6. Data experiments

To thoroughly evaluate the performance of the DHNN-3SAT-GAKM model and its ability to solve real-world application problems, this section examines its performance alongside the conventional DHNN-3SAT-WA, DHNN-3SAT-GA, and DHNN-3SAT-ICA models on a benchmark dataset. Experimental analyses were conducted to compare their performance and demonstrate the superiority of the DHNN-3SAT-GAKM model proposed in this study. The experiments were carried out using MATLAB R2023b on a laptop computer running the Windows 10 operating system, equipped with an AMD Razor R5-3500U processor and 8 GB of RAM.

6.1. Description of the dataset

This study utilizes the DIMACS Benchmark Instances AIM dataset from SATLIB, provided by Kazuo Iwama et al. (https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html). The AIM dataset comprises of 48 instances, with 24 being satisfiable and 24 unsatisfiable. To create a representative set of instances, 12 of these satisfiable instances are chosen for this study. Each instance contains three clauses, and you can find specific descriptions of the instances in Table 5.

Table 5. Description of example data.

No.	Instance	variables	Clauses	No.	Instance	variables	Clauses
1	aim-50-1_6-yes1-1	50	80	7	aim-100-3_4-yes1-1	100	340
2	aim-50-2_0-yes1-1	50	100	8	aim-100-6_0-yes1-1	100	600
3	aim-50-3_4-yes1-1	50	170	9	aim-200-1_6-yes1-1	200	320
4	aim-50-6_0-yes1-1	50	300	10	aim-200-2_0-yes1-1	200	400
5	aim-100-1_6-yes1-1	100	160	11	aim-200-3_4-yes1-1	200	680
6	aim-100-2_0-yes1-1	100	200	12	aim-200-6_0-yes1-1	200	1200

6.2. Parameter setting

In the search phase, the traditional DHNN-3SAT-WA model directly examines 10,000 different combinations of initial neuron assignments [46]. DHNN-3SAT-GA and DHNN-3SAT-ICA guide the search among these 10,000 combinations using a genetic algorithm and an imperialistic competition algorithm, respectively. This paper introduces the DHNN-3SAT-GAKM model, which utilizes genetic optimization K-modes clustering to preprocess these 10000 neuron initial allocation combinations. It then selects a candidate subset for search. This approach reduces the actual search space and minimizes repeated local searches to avoid getting stuck in local minima, thereby improving the efficiency of retrieving the global minimum. The tolerance values for the conventional DHNN-3SAT-WA model align with Sathasivam's work [16]. The CPU time thresholds are based on Zamri's settings [47]. The parameter settings can be found in Table 6. The parameter settings for the DHNN-3SAT-GA model are in line with Kasihmuddin's work [26] and are listed in Table 7. The parameter settings for the DHNN-3SAT-ICA model remain consistent with Shazli's work [38], as shown in Table 8. Table 9 details the parameter settings of the model in this paper, with optimization of the relevant parameters through iterative tuning.

Table 6. DHNN-3SAT-WA model parameter settings.

parametric	parameter value	parametric	parameter value
Initial assigned amount	10000	tolerance value δ	0.001
CPU time threshold	24 hours	-	-

Table 7. DHNN-3SAT-GA model parameter settings.

parametric	parameter value	parametric	parameter value
Initial assigned amount	10000	probability of mutation p_m	0.05
population size	50	Maximum Iterations t	100
crossover probability p_c	0.6	-	-

Table 8. DHNN-3SAT-ICA model parameter settings.

parametric	parameter value	parametric	parameter value
Initial assigned amount	10000	revolutionary rate α	0.3
population size	50	Maximum Iterations t	100

Table 9. DHNN-3SAT-GAKM model parameter settings.

parametric	parameter value	parametric	parameter value
Initial assigned amount	10000	crossover probability p_c	0.6
population size	50	probability of mutation p_m	0.05
Initial number of clusters	3	Maximum Iterations t	100

6.3. Experimental results and discussion

We use the global minimum ratio (GMR) [16] and the mean CPU time (MCT) to assess the model's performance in this paper. To provide a more comprehensive evaluation of the model's ability to find the global minimum, we introduce 2 new evaluation metrics: the mean minimum Hamming distance (MMHD) and the mean logical satisfiability ratio (MLSR). This study will utilize a total of 4 evaluation metrics, as detailed in Table 10. The calculations are based on the average of 100 repeated experiment runs for each instance, and the results are displayed in Tables 11 and 12. Figures 9 to 12 compare our model, DHNN-3SAT-GAKM, with the models DHNN-3SAT-WA, DHNN-3SAT-GA, and DHNN-3SAT-ICA across the 4 evaluation metrics.

Table 10. Assessment indicators.

Indicators	calculation formula	instructions
GMR	$GMR = \frac{N_{GM}}{T}$	*GMR* represents the ratio of the global minimum solution to the total number of runs [16]. ***GMR*** is an effective metric for assessing the efficiency of an algorithm. A model is considered robust when its GMR value is close to 1 [23]. Here, N_{GM} represents the number of times the global minimum is converged, and T represents the total number of runs.
MCT	$MCT = \frac{1}{N_{GM}} \sum_{i}^{T_{GM}} NT_i$	The MCT refers to the average time needed for each model to reach the global minimum. A smaller MCT indicates that the model is more efficient in finding the global minimum. NT_i represents the CPU time needed to find the global minimum at the ith retrieval result, and N_{GM} represents the number of times the global minimum converged.
MMHD	$MMHD = \frac{1}{T} \sum_{i}^{T} \min_{j} D(X_i, Z_j)$	The MMHD value represents the mean minimum Hamming distance, which is the average of the smallest bit difference between the retrieval result of each run and the global minimum. When the MMHD value is closer to 0, it indicates that the model retrieves a result closer to the global minimum. In this context, $D(X_i, Z_j)$ represents the Hamming distance between the retrieval result of the ith run and the global minimum, and T represents the total number of runs.
MLSR	$MLSR = \frac{1}{T} \sum_{i}^{T} \frac{N_{sat(i)}}{m}$	The MLSR value indicates the average proportion of the total number of clauses that can be satisfied by the retrieval results. The closer the *MLSR* value is to 1, the closer the model retrieval results are to the global minimum. Here, $N_{sat(i)}$ denotes the number of satisfying clauses for the ith retrieval result, and m denotes the total number of clauses.

Table 11. Comparison of experimental results.

No.	GMR		MCT		MMHD		MLSR	
	DHNN-3SAT-WA	DHNN-3SAT-GA	DHNN-3SAT-WA	DHNN-3SAT-GA	DHNN-3SAT-WA	DHNN-3SAT-GA	DHNN-3SAT-WA	DHNN-3SAT-GA
1	1.0000	1.0000	4.83	2.51	0.0000	0.0000	1	1
2	0.9123	0.9323	16.67	11.70	1.1000	1.0900	0.9744	0.9745
3	0.8234	0.8456	26.67	22.83	3.6000	3.4900	0.9355	0.9578
4	0.5802	0.6204	48.82	45.80	3.7200	3.7100	0.8923	0.8966
5	1.0000	1.0000	11.72	6.31	0.0000	0.0000	1	1
6	0.8812	0.9011	46.51	16.01	2.5000	2.4000	0.9433	0.9533
7	0.5467	0.6041	113.28	35.50	3.6200	3.6100	0.9288	0.9363
8	0.2018	0.2188	440.39	171.23	4.7400	4.2300	0.9139	0.9231
9	0.4114	0.4222	537.92	431.04	4.8600	4.4500	0.9835	0.9844
10	0.2261	0.2352	978.78	773.75	5.9800	5.6700	0.9312	0.9474
11	0.1616	0.1653	1369.44	1100.94	7.1000	6.8900	0.8537	0.8775
12	0.1413	0.1518	1566.18	1198.85	8.2200	8.1100	0.8234	0.8641

Table 12. Comparison of experimental results.

No.	GMR		MCT		MMHD		MLSR	
	DHNN-3SAT-ICA	DHNN-3SAT-GAKM	DHNN-3SAT-ICA	DHNN-3SAT-GAKM	DHNN-3SAT-ICA	DHNN-3SAT-GAKM	DHNN-3SAT-ICA	DHNN-3SAT-GAKM
1	1.0000	1.0000	2.49	2.21	0.0000	0.0000	1	1
2	0.9441	0.9658	10.64	8.29	1.0700	1.0400	0.9761	0.9783
3	0.8542	0.9126	20.45	15.08	3.2700	1.1400	0.9662	0.9751
4	0.6356	0.7229	40.18	26.87	3.3700	2.9700	0.8978	0.9354
5	1.0000	1.0000	5.49	4.68	0.0000	0.0000	1	1
6	0.9124	0.9256	14.02	11.06	2.2000	2.1000	0.9644	0.9881
7	0.6205	0.6898	30.59	22.03	3.2000	2.9300	0.9375	0.9523
8	0.2291	0.3211	141.54	74.60	3.9800	3.7600	0.9251	0.9336
9	0.4301	0.4503	435.12	396.82	4.0800	3.8900	0.9851	0.9928
10	0.2488	0.2632	752.20	678.91	5.0800	4.7200	0.9488	0.9557
11	0.1689	0.2203	1108.03	935.02	6.0800	5.5500	0.8809	0.9028
12	0.1612	0.2097	1160.96	982.28	7.0800	6.3800	0.8732	0.8822

Tables 11 and 12 show the computational results of the DHNN-3SAT-GAKM model in this paper, as well as the DHNN-3SAT-WA, DHNN-3SAT-GA, and DHNN-3SAT-ICA models. The results are presented in terms of GMR, MCT, MMHD, and MLSR. These calculations are based on the metrics formulas provided in Table 10. Figures 9 to 12 illustrate the performance differences between this paper's DHNN-3SAT-GAKM model and the DHNN-3SAT-WA, DHNN-3SAT-GA, and DHNN-3SAT-ICA models. These differences are shown in the four evaluation metrics through radargram-based visualization. The DHNN-3SAT-GAKM model outperforms the DHNN-3SAT-WA,

DHNN-3SAT-GA, and DHNN-3SAT-ICA models.

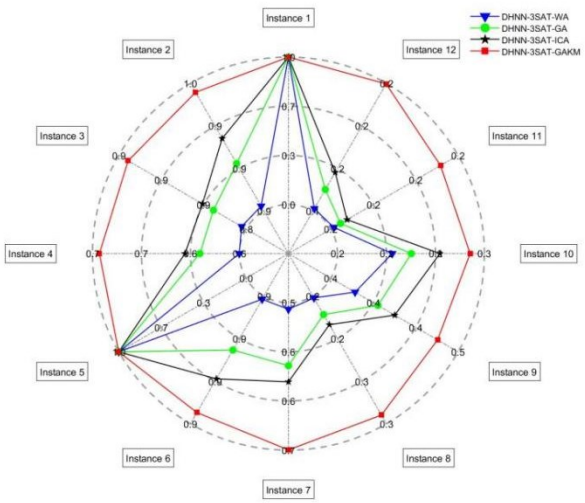
Figure 9 shows the GMR of each model in solving 3-SAT instances with varying levels of complexity. In this paper, the DHNN-3SAT-GAKM model achieved the highest GMR value, indicating its superior global retrieval ability and ability to avoid falling into local minima to some extent. Additionally, the DHNN-3SAT-GA and DHNN-3SAT-ICA models also demonstrated an improved ability over the traditional DHNN-3SAT-WA model to retrieve the global minimum to some extent. As the complexity of SAT problems, Boolean variables, and logical clauses increases, the GMRs of each model decrease rapidly. Hence, further optimization and improvement of the algorithms and architectures of the DHNN-3SAT models are needed to enhance their performance and efficiency when dealing with large-scale and complex SAT problems in the future.

In Figure 10, we can see the average time taken by each model to reach the global minimum. The MTC value of the DHNN-3SAT-GAKM model in this paper is the smallest, indicating that this model is more efficient in finding the global minimum. This is because the model initially clusters the allocation space using the K-modes clustering algorithm, enabling it to escape local minima more quickly and avoid repetitive retrieval of local minima. As a result, a large number of redundant calculations are reduced, leading to improved efficiency in converging to the global minimum. On the other hand, the traditional DHNN-3SAT-WA is more prone to getting stuck in local minima, especially as the number of local minimum solutions increases, resulting in a substantial number of repetitive evolutions and computations, ultimately affecting the efficiency of converging to the global minimum. While the DHNN-3SAT-GA and DHNN-3SAT-ICA models also use heuristic algorithms for guided search to some extent, helping to reduce the search space and speed up retrieval of the global minimum, the rapidly expanding search space due to the increasing complexity of the SAT problem can lead to longer retrieval times or even search failure. Consequently, for large-scale SAT problems, further improving the efficiency of searching for the global minimum is a future priority.

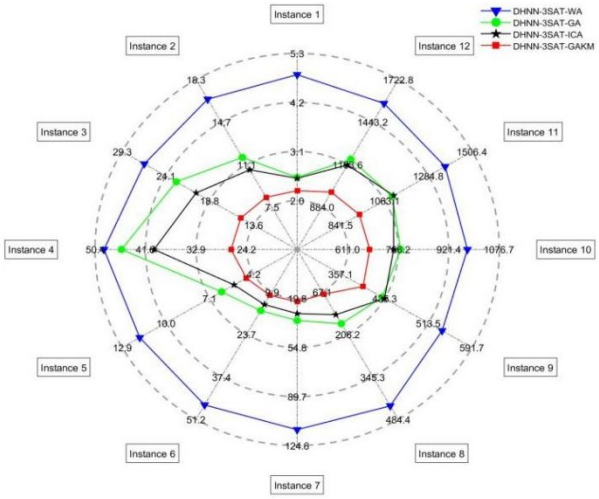
In dealing with large-scale SAT problems, the increasing logic complexity results in a progressively smaller solution space, making it very challenging to find a global minimum solution within a limited timeframe. Most attempts only end up finding the local minimum solution. The goal of model optimization at this stage is to make each retrieval result as close as possible to the global minimum solution. To evaluate the proximity of the model retrieval results to the global minimum solution, two new evaluation criteria are introduced in this study: the MMHD and the MLSR. These criteria reflect the proximity of the model retrieval results to the global optimum. A lower MMHD value and a higher MLSR value indicate that the model retrieval results are closer to the global minimum solution. Figures 11 and 12 illustrate the relationship between the MMHD and MLSR values of each model. These figures show that the DHNN-3SAT-GAKM model in this paper has the smallest MMHD value and the largest MLSR value, indicating that its retrieval results are closer to the global minimum than those of the DHNN-3SAT-WA, DHNN-3SAT-GA, and DHNN-3SAT-ICA models. This suggests that the retrieval results of the DHNN-3SAT-GAKM model are closer to the global minimum solution overall. Conversely, the DHNN-3SAT-GA and DHNN-3SAT-ICA models are closer to the global minimum than the overall retrieval results of the conventional DHNN-3SAT-WA.

Based on the combined analyses above, it can be observed that the DHNN-3SAT-GAKM model, introduced in this paper, exhibits significant improvements when compared to the traditional DHNN-3SAT-WA model, as well as the DHNN-3SAT-GA and DHNN-3SAT-ICA models that

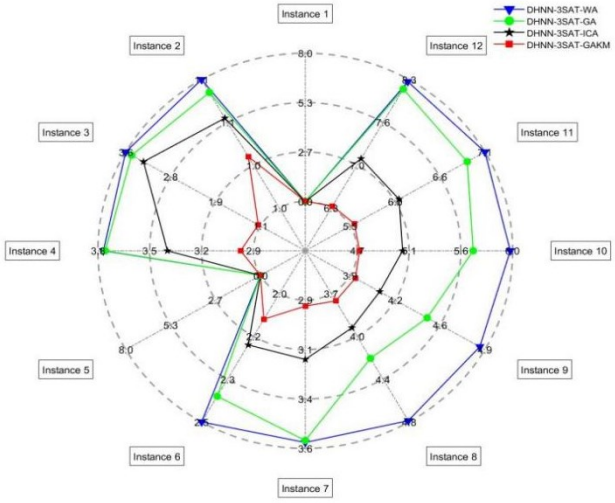
directly utilize heuristic algorithms for bootstrap retrieval. This demonstrates the superior performance of the DHNN-3SAT-GAKM model in retrieving the global minima in the SAT problem, while also highlighting its potential for practical applications.



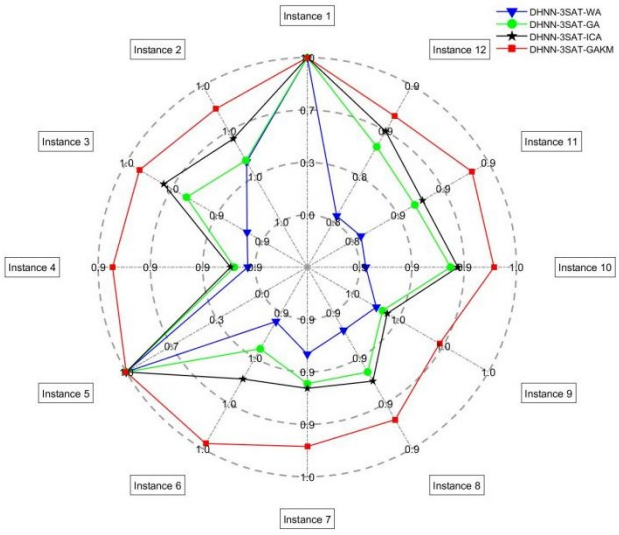
Figures 9. GMR.



Figures 10. MCT.



Figures 11. MMHD.



Figures 12. MLSR.

7. Conclusions

This paper introduces a method for designing network synaptic weights based on basic logical clauses to handle dynamic changes in constraints in the SAT problem. This method aims to utilize synaptic weight information efficiently, reducing the need for repetitive calculations in the DHNN network. Additionally, it proposes a DHNN-3SAT model based on genetic algorithm optimized K-modes clustering to address the limitations of the traditional DHNN-3SAT-WA, which tends to get stuck in local minima. The new model uses genetic algorithms to cluster the initial space, effectively reducing the retrieval space and improving retrieval efficiency. Experimental results show that the

DHNN-3SAT-GAKM model outperforms DHNN-3SAT-WA, DHNN-3SAT-GA, and DHNN-3SAT-ICA in terms of various evaluation metrics, including GMR, MCT, MMHD, and MLSR. This study not only expands the application of DHNN in solving the SAT problem but also offers valuable insights for future research.

The DHNN-3SAT model is an innovative approach to using deep learning technology to solve SAT problems, offering insights and potential for future research and applications. There are several areas for future work: first, optimizing and enhancing the algorithm and architecture of the DHNN-3SAT model to improve its performance on large-scale and complex SAT problems; second, exploring the model's extension to other NP-complete problems to demonstrate its versatility and applicability; and finally, conducting thorough research on the model in specific domains and practical applications to further promote the use of deep learning techniques in combinatorial optimization and decision-making problems. In summary, the proposal and study of the DHNN-3SAT model not only enhances methods in the field of SAT problem-solving but also provides new ideas and tools for solving complex problems using deep learning techniques. With ongoing technological and theoretical progress, the application of deep learning in combinatorial optimization problem-solving is expected to bring about broader development and deliver effective solutions for real-life complex problems.

Author contributions

Xiaojun Xie: Writing-review & editing, Writing-original draft, Methodology, Formal analysis, Conceptualization. Saratha Sathasivam: Writing-review & editing, Methodology, Funding acquisition, Conceptualization, Validation, Supervision. Hong Ma: Writing-review & editing, Writing-original draft, Methodology, Investigation, Formal analysis, Conceptualization. All authors have read and approved the final version of the manuscript for publication.

Acknowledgments

This research was supported by the Ministry of Higher Education Malaysia (MOHE) through the Fundamental Research Grant Scheme (FRGS), FRGS/1/2022/STG06/USM/02/11, and University Sains Malaysia.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. M. Järvisalo, B. D. Le, O. Roussel, L. Simon, The international SAT solver competitions, *Ai Mag.*, **33** (2012), 89–92. https://doi.org/10.1609/aimag.v33i1.2395
2. S. A Cook, The complexity of theorem-proving procedures, *Logic automata computat. Complex.*, 2023, 143152.
3. J. Rintanen, Planning as satisfiability: Heuristics, *Artif. Intell.*, **193** (2012), 45–86. https://doi.org/10.1016/j.artint.2012.08.001

4. V. Popov, An approach to the design of DNA smart programmable membranes, *Adv. Mater. Res.*, **934** (2014), 173–176. https://doi.org/10.4028/www.scientific.net/AMR.934.173
5. X. Zhang, J. Bussche, F. Picalausa, On the satisfiability problem for SPARQL patterns, *J. Artif. Intell. Res.*, **56** (2016), 403–428. https://doi.org/10.1613/jair.5028
6. A. Armando, L. Compagna, SAT-based model-checking for security protocols analysis, *Int. J. Inf. Secur.*, **7** (2008), 3–32. https://doi.org/10.1007/s10207-007-0041-y
7. C. Luo, S. Cai, W. Wu, K. Su, Double configuration checking in stochastic local search for satisfiability, *Proc. AAAI Conf. Artif. Intell.*, **28** (2014), 2703–2709. https://doi.org/10.1609/aaai.v28i1.9110
8. X. Wang, A novel approach of solving the CNF-SAT problem, *Arxiv. Prepr. Arxiv.*, **1307** (2013), 6291. https://doi.org/10.48550/arXiv.1307.6291
9. D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: Artificial bee colony (ABC) algorithm and applications, *Artif. Intell. Rev.*, **42** (2014), 21–57. https://doi.org/10.1007/s10462-012-9328-0
10. E. A. Hirsch, A. Kojevnikov, UnitWalk: A new SAT solver that uses local search guided by unit clause elimination, *Ann. Math. Artif. Intell.*, **43** (2005), 91–111. https://doi.org/10.1007/s10472-005-0421-9
11. J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci.*, **79** (1982), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554
12. M. A. Mansor, S. Sathasivam, Optimal performance evaluation metrics for satisfiability logic representation in discrete Hopfield neural network, *Int. J. Math. Comput. Sci.*, **16** (2021), 963–976.
13. C. C. Feng, S. Sathasivam, A novel processor for dynamic evolution of constrained SAT problems: The dynamic evolution variant of the discrete Hopfield neural network satisfiability model, *J. King Saud Univ., Comput. Inf. Sci.*, **36** (2024), 101927. https://doi.org/10.1016/j.jksuci.2024.101927
14. 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
15. M. A. Mansor, S. Sathasivam, Accelerating activation function for 3-satisfiability logic programming, *Int.J. Intell. Syst. Appl.*, **8** (2016), 44. https://doi.org/10.5815/ijisa.2016.10.05.
16. S. Sathasivam, Upgrading logic programming in Hopfield network, *Sains Malays.*, **39** (2010), 115–118.
17. M. A. Mansor, M. S. M Kasihmuddin, S. Sathasivam, Artificial immune system paradigm in the Hopfield network for 3-Satisfiability problem, *Pertanika J. Sci. Technol.*, **25** (2017), 1173–1188.
18. B. Bünz, M. Lamm. Graph neural networks and boolean satisfiability, *Arxiv. Prepr. Arxiv.*, **1702** (2017), 03592. https://doi.org/10.48550/arXiv.1702.03592
19. H. Xu, S. Koenig, T. K. S. Kumar, Towards effective deep learning for constraint satisfaction problems, *Int. Conf. Princ. Pract. Constraint Program.*, 2018, 588–597. https://doi.org/10.1007/978-3-319-98334-9_38
20. H. E. Dixon, M. L. Ginsberg, Combining satisfiability techniques from AI and OR, *Knowl. Eng. Rev.*, **15** (2000), 31–45. https://doi.org/10.1017/S0269888900001041
21. W. A. Abdullah, Logic programming on a neural network, *Int. J. Intell. Syst.*, **7** (1992), 513–519. https://doi.org/10.1002/int.4550070604

22. S. Sathasivam, W. A. Abdullah, Logic mining in neural network: Reverse analysis method, *Computing*, **91** (2011), 119–133. https://doi.org/10.1007/s00607-010-0117-9
23. S. Sathasivam, N. P. Fen, M. Velavan, Reverse analysis in higher order Hopfield network for higher order horn clauses, *Appl. Math. Sci.*, **8** (2014), 601–612. http://dx.doi.org/10.12988/ams.2014.310565
24. M. S. M. Kasihmuddin, M. A. Mansor, S. Sathasivam, Discrete Hopfield neural network in restricted maximum k-satisfiability logic programming, *Sains Malays.*, **47** (2018), 1327–1335. http:/ldx.doi.org110.17576/jsm-2018-4706-30
25. M. A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Robust artificial immune system in the hopfield network for maximum k-satisfiability. *Int. J. Inter. Mult. Artif. Intell.*, **4** (2017), 63–71.
26. M. S. M. Kasihmuddin, M. A. Mansor, S. Sathasivam, Hybrid genetic algorithm in the hopfield network for logic satisfiability problem. *Pertanika J. Sci. Technol.*, **25** (2017), 139–152.
27. F. L. Azizan, S. Sathasivam M. K. M. Ali, N. Roslan, C. Feng, Hybridised network of fuzzy logic and a genetic algorithm in solving 3-Satisfiability Hopfield neural networks, *Axioms*, **12** (2023), 250. https://doi.org/10.3390/axioms12030250
28. M. A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Grey wolf optimization algorithm with discrete hopfield neural network for 3 Satisfiability analysis, *J. Phys. Conf. Ser.*, **1821** (2021), 012038. https://doi.org/10.1088/1742-6596/1821/1/012038
29. M. A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Modified lion optimization algorithm with discrete Hopfield neural network for higher order Boolean satisfiability programming, *Malays. J. Math. Sci.*, **14** (2020), 47–61.
30. N. Cao, X. J. Yin, S. T. Bai, Breather wave, lump type and interaction solutions for a high dimensional evolution model, *Chaos, Solitons Fract.*, **172** (2023), 113505. https://doi.org/10.1016/j.chaos.2023.113505
31. C. 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/10.1016/10.3934/math.2024450
32. S. Bai, X. Yin, N. Cao, L. Xu, A high dimensional evolution model and its rogue wave solution, breather solution and mixed solutions *Nonlinear Dyn.*, **111** (2023), 12479–12494. https://doi.org/10.1007/s11071-023-08467-x
33. G. He, P. Tang, X. Pang, Neural network approaches to implementation of optimum multiuser detectors in code-division multiple-access channels, *Int. J. Electron.*, **80** (1996), 425-431. https://doi.org/10.1080/002072196137264
34. H. Yang, Z. Li, Z. Liu, A method of routing optimization using CHNN in MANET, *J. Ambient Intell. Hum. Comput.*, **10** (2019), 1759–1768. https://doi.org/10.1007/s12652-017-0614-1
35. B. Bao, H. Tang, Y. Su, H. Bao, M. Chen, Q. Xu, Two-Dimensional discrete Bi-Neuron Hopfield neural network with polyhedral Hyperchaos, *IEEE Trans. Circuits Syst.*, (2024), 1–12. https://doi.org/10.1109/TCSI.2024.3382259
36. W. Ma, X. Li, T. Yu, Z. Wang, A 4D discrete Hopfield neural network-based image encryption scheme with multiple diffusion modes, *Optik*, **291** (2023), 171387. https://doi.org/10.1016/j.ijleo.2023.171387
37. Q. Deng, C. Wang, H. Lin, Memristive Hopfield neural network dynamics with heterogeneous activation functions and its application, *Chaos Solitons Fract.*, **178** (2024), 114387. https://doi.org/10.1016/j.chaos.2023.114387

38. Z. S. Shazli, M. B. Tahoori, Using boolean satisfiability for computing soft error rates in early design stages, *Microelectr. Reliab.*, **50** (2010), 149–159. https://doi.org/10.1016/j.microrel.2009.08.006
39. N. Sharma, N. Gaud, K-modes clustering algorithm for categorical data, *Int. J. Comput. Appl.*, **127** (2015), 1–6. https://doi.org/10.1016/10.5120/ijca2015906708
40. F. Cao J. Liang, D. Li, L. Bai, C. Dang, A dissimilarity measure for the K-Modes clustering algorithm, *Knowl. Based Syst.*, **26** (2012), 120–127. https://doi.org/10.1016/j.knosys.2011.07.011
41. M. K. Ng, M. J. Li, J. Z. Huang, Z. He, On the impact of dissimilarity measure in k-modes clustering algorithm, *IEEE Trans. Pattern Anal. Mach. intell.,* **29** (2007), 503–507. https://doi.org/10.1109/TPAMI.2007.53
42. S. S. Khan, A. Ahmad, Cluster center initialization algorithm for K-modes clustering, *Pattern Recognit. Lett.*, **25** (2004), 1293–1302. https://doi.org/10.1016/j.patrec.2004.04.007
43. R. J. Kuo, Y. R. Zheng, T. P. Q. Nguyen, Metaheuristic-based possibilistic fuzzy k-modes algorithms for categorical data clustering, *Inf. Sci.*, **557** (2021), 1–15. https://doi.org/10.1016/j.ins.2020.12.051
44. G. Gan, J. Wu, Z. Yang, A genetic fuzzy k-Modes algorithm for clustering categorical data, *Expert Syst. Appl.*, **36** (2009), 1615–1620. https://doi.org/10.1007/11527503_23
45. F. S. Gharehchopogh, S. Haggi, An Optimization K-modes clustering algorithm with elephant herding optimization algorithm for crime clustering, *J. Adv. Comput. Eng. Technol.*, **6** (2020), 79–90.
46. J. Gu, Local search for satisfiability (SAT) problem, *IEEE Trans. Syst.*, **23** (1993), 1108–1129. https://doi.org/10.1109/21.247892
47. N. E. Zamri, M. A, Mansor, M. S. M. Kasihmuddin, A. Always, S. A. Alzaeemi, Amazon employees resources access data extraction via clonal selection algorithm and logic mining approach, *Entropy*, **22** (2020), 596. https://doi.org/10.3390/e22060596

AIMS Press