AIMS Mathematics, 11(7): 22495–22542.
DOI: 10.3934/math.2026910
Received: 20 March 2026
Revised: 07 May 2026
Accepted: 12 May 2026
Published: 27 July 2026

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

Research article

Two-stage asymmetric hybrid binary particle swarm optimization with adaptive-parameter learning in discrete Hopfield neural networks

Yunjie Chang[1,2], Mohd Shareduwan Mohd Kasihmuddin[1,*], Yuan Gao[3], Yueling Guo[4], Nur Ezlin Zamri[5] and Xiaofeng Jiang[1]

[1] School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia

[2] School of Computer Science and Engineering, Hunan Institute of Technology, Hengyang 421002, Hunan, China

[3] School of Intelligent Medicine, Chengdu University of Traditional Chinese Medicine, Chengdu 611137, Sichuan, China

[4] School of Science, Hunan Institute of Technology, Hengyang 421002, Hunan, China

[5] Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

* **Correspondence:** Email: shareduwan@usm.my.

Abstract: Propositional satisfiability in discrete Hopfield neural network (DHNN) is widely studied, but its practical use is limited by low storage capacity, slow convergence, and repetitive neuron states. To address these limitations, we proposed a two-stage asymmetric hybrid binary particle swarm optimization (HBPSO) framework for enhancing the weighted C-type random 2-satisfiability logic in DHNN. In the first stage, a multi-objective-guided strategy combining fixed-parameter and adaptive-parameter HBPSO was introduced to improve storage capacity and guide the network toward optimal synaptic weight configurations. In the second stage, the fixed-parameter HBPSO was employed to further reduce repetitive neuron states and enhance solution diversity. Experimental results showed that the method achieves 100% fitness, identifies over 8.4 distinct optimal weight configurations with a 100% uniqueness ratio, and reduces the average iterations to 25.81. It also remained robust under noise levels from 0.1 to 1.0, with only a 41.2% increase in iterations. Overall, the proposed asymmetric HBPSO framework provides an effective approach for solving complex combinatorial optimization problems in computational intelligence.

Keywords: discrete Hopfield neural network; propositional satisfiability; hybrid binary particle swarm optimization; computational intelligence; combinatorial optimization
Mathematics Subject Classification: 68T05, 90C27, 90C59

1. Introduction

The implementation of propositional satisfiability (SAT) logic in discrete Hopfield neural networks (DHNN) has attracted increasing attention in neural-symbolic computing. This interest is largely motivated by the capability of such models to construct interpretable computational frameworks that can extract explicit and human-interpretable knowledge from real-world datasets [1]. By embedding SAT rules as symbolic constraints, DHNN can alleviate the "black-box" limitation commonly associated with conventional neural architectures and improve the mathematical interpretability of learning processes [2]. Since Abdullah first demonstrated that SAT can be implemented in DHNN as a unified computational framework [3], numerous SAT-based logic formulations have been proposed to enhance the representational capability and computational performance of DHNN models. Despite these developments, several limitations in the learning and retrieval phases, including restricted storage capacity, slow convergence, and repetitive neuron states, continue to hinder the practical applicability of DHNN-based neural-symbolic systems.

1.1. Motivations

1.1.1. The convergence of suboptimal synaptic weights

During the learning phase of the DHNN, the convergence capability in obtaining optimal synaptic weights decrease sharply as the number of neurons becoming large. This is because the search space for optimal synaptic weights increases exponentially, reaching 2^n, where n denotes the number of neurons [4]. Because synaptic weights determine connection strength and govern system dynamics, suboptimal weight estimation undermines the effective minimization of the Lyapunov energy function. Consequently, the network becomes highly prone to entrapment in local minima, especially in high-dimensional search spaces. This limitation substantially reduces the proportion of globally optimal solutions, thereby compromising the reliability of the model in solving complex combinatorial optimization problems. Despite this well-established challenge, most optimization approaches in the learning phase primarily emphasize achieving locally optimal synaptic weights while overlooking the inherent storage capacity limitation of the DHNN. As a result, the issue of efficient and scalable convergence in large-scale DHNNs remains insufficiently addressed. This gap highlights the urgent need for effective strategies that enhance convergence toward globally optimal synaptic weights while accounting for the structural limitations of large-scale networks.

1.1.2. The repeated convergence to identical neuron states

During the retrieval phase, the DHNN repeatedly converges to identical neuron states rather than exploring diverse state patterns, leading to overfitting [4]. This behavior produces highly redundant final neuron states with limited diversity, thereby restricting the capacity of the model to represent complex and heterogeneous knowledge structures. As a consequence, repetitive neuron states significantly reduce the adaptability of the DHNN and constrain its applicability across problem domains. Although researchers have attempted to improve the retrieval performance of DHNN, most approaches rely on fixed or single metaheuristic strategies, which inherently limit exploration capability. More importantly, researchers seldom consider integrated or two-stage optimization frameworks that simultaneously enhance the learning and retrieval phases to improve the diversity

of the final neuron states. Consequently, the issue of insufficient state diversity remains inadequately addressed. To overcome this limitation, it is essential to develop an effective mechanism that enables the model to explore a broader range of configurations and better capture complex patterns.

1.2. Research objective

To address limitations and enhance the applicability of logic-rule-based models across real-world problem domains, we propose a novel two-stage asymmetric hybrid binary particle swarm optimization framework with adaptive parameter learning for weighted C-type random 2-satisfiability (WCRAN2SAT) in DHNN. Our primary objective is to enhance solution diversity and storage capacity by generating multiple non-duplicated optimal configurations, rather than concentrating solely on conventional single-solution SAT solving. Our research objectives are as follows:

(a) To develop a fixed-parameter hybrid binary particle swarm optimization (HBPSO) by enhancing the conventional binary particle swarm optimization through the integration of a sigmoid transfer function with velocity clamping, a mutation operator, and a greedy search mechanism. These improvements are designed to strengthen the balance between exploration and exploitation capabilities.

(b) To formulate a multi-objective function that incorporates fitness value, the number of non-duplicated optimal synaptic weights, and the diversity of these weight configurations. This multi-objective framework guides the HBPSO to achieve superior learning performance in terms of weight convergence and storage enhancement.

(c) To propose an adaptive-parameter HBPSO built upon the fixed-parameter version. A comprehensive robustness analysis is conducted, including dynamic parameter adjustment, and resistance to noise. The adaptive mechanism is intended to enhance the model's stability under dynamic and noisy environments.

(d) To design a cascaded two-stage optimization strategy for improving the performance of DHNN models. In the first-stage optimization, fixed-parameter HBPSO and adaptive-parameter HBPSO are employed in alignment with a multi-objective function to generate multiple unduplicated optimal synaptic weight configurations. In the second-stage optimization, the DHNN is further refined to produce a higher number of unduplicated global minimum energy solutions. Through this cascaded optimization process, the effective storage capacity of the network is significantly enhanced.

The remainder of this paper is organized as follows: In Section 2, we review related works. In Section 3, we detail WCRAN2SAT in DHNN. In Section 4, we present the proposed work, including the multi-objective function, the proposed fixed and adaptive-parameter HBPSO algorithm, and two-stage optimization for WCRAN2SAT. In Section 5, we outline the experimental setup. In Section 6, we report results and discussions, covering two-stage optimization performance, sensitivity analysis for the fixed-parameter HBPSO, and robustness analysis for the adaptive-parameter HBPSO. Finally, in Section 7, we conclude the paper and discuss future research directions.

2. Literature review

2.1. Optimization in the learning phase

During the learning phase, the ability of the DHNN to obtain optimal synaptic weights declines significantly as the number of neurons increases. To address this limitation, various metaheuristic algorithms have been proposed. For example, the election algorithm (EA) proposed by Sathasivam et al. [5] has demonstrated a substantial reduction in training errors compared with exhaustive search methods. Later, Zamri et al. [4] developed the genetic algorithm (GA) to strengthen global exploration capability and accelerate convergence toward optimal synaptic weights. Recognizing the trade-off between accuracy and population diversity, Karim et al. [6] proposed the hybrid election algorithm (HEA), which further expanded the effective storage capacity of the network while mitigating repetitive neuron state generation. Moreover, research attention has shifted toward memory-guided and pheromone-driven optimization paradigms to better navigate high-dimensional and combinatorial search spaces. Gao et al. [1] proposed the binary optimal solution ant colony optimization (BOSACO), which enhances convergence accuracy and exploration-exploitation balance.

Collectively, these metaheuristic frameworks have substantially improved training performance in terms of optimal synaptic weights with competitive computational efficiency and solution quality. Nevertheless, most researchers in this area focus primarily on single-objective optimization and fail to address the storage capacity limitations of DHNN. Although metaheuristic models improve optimization performance, they fail to simultaneously expand storage capacity while maintaining stable convergence to optimal synaptic weights. Overcoming this limitation is essential for developing scalable, high-capacity, and robust DHNN systems.

2.2. Optimization in the retrieval phase

An inherent limitation of DHNN is that its retrieval mechanism recalls only patterns stored in content-addressable memory (CAM) without ensuring convergence to unique final neuron states. To overcome this shortcoming, Sathasivam et al. [7] introduced the Boltzmann Hopfield neural network (BHNN), integrating probabilistic state transitions through a Boltzmann machine to enhance global search capability and avoid local minima. Subsequently, Kasihmuddin et al. [8] incorporated the estimation of distribution algorithm (EDA) into DHNN to improve retrieval diversity and exploration capability. Later, Sathasivam et al. [9] proposed the mean field theory-based Hopfield neural network (MFTHNN), employing mean field approximation to enhance neuronal state diversity. However, this approach employs a single-stage optimization strategy that primarily strengthens the learning phase while leaving the retrieval phase relatively underexplored. Building upon this idea, Guo et al. [2] proposed a dual-optimization framework integrating a hybrid differential evolution algorithm (HDEA) with a mutation-based activation function and a swarm mutation mechanism to further improve state diversity.

Although this framework demonstrates strong performance in improving retrieval diversity, it does not incorporate metaheuristic strategies directly into the retrieval phase. Additionally, according to the No-Free-Lunch (NFL) theorem [10], no single optimization algorithm can achieve superior performance across all classes of problems. Nevertheless, despite this well-established principle,

most studies remain restricted to fixed or single metaheuristic strategies applied solely during the learning phase, thereby overlooking optimization in the retrieval phase and limiting overall network performance.

2.3. Multi-stage optimization methods

Beyond DHNN-specific optimization, studies have emphasized the importance of adaptive mechanisms and structural learning. For instance, Yang et al. [11] proposed an adaptive data structure regularization framework that jointly learns feature representations and underlying data structures, where structure learning and feature selection mutually reinforce each other to improve discriminative capability. This unified optimization perspective highlights the benefit of coupling structural information with the search process. Zhou et al. [12] employed an attention-based spatio-temporal graph neural network to model dynamic interactions, demonstrating that adaptive weighting mechanisms significantly improve system behavior modeling. In addition, Chen et al. [13] proposed a tri-population evolutionary framework with an adaptive stage-switching mechanism, showing that balancing convergence and diversity through dynamic control can effectively enhance search performance. These findings suggest that coordinated multi-stage optimization and adaptive scheduling are critical for improving convergence and diversity.

Researchers have investigated alternative paradigms for solving constraint satisfaction and combinatorial optimization problems, particularly learned solvers and hybrid methods. For instance, Heydaribeni et al. [14] proposed HypOp, a distributed hypergraph neural network-based framework for constrained combinatorial optimization. Additionally, scalable discrete diffusion samplers have been proposed for combinatorial optimization and statistical physics, demonstrating strong performance in generating solutions over discrete energy landscapes.

Despite these advantages, learning-based approaches rely on training procedures and may exhibit limited generalization performance under distribution shift or on out-of-distribution instances. In contrast, optimization-driven methods, such as the DHNN framework considered in this work, do not require data-driven training and instead operate based on an energy minimization principle, which can provide stable convergence behavior for SAT-based combinatorial optimization problems.

2.4. Binary particle swarm optimization

Binary particle swarm optimization (BPSO) [15] is a discrete variant of the standard particle swarm optimization (PSO). PSO is popular for its simplicity, fast convergence, and minimal parameter tuning. BPSO has shown effectiveness in high-dimensional optimization problems, providing stable convergence and accurate solutions [16]. However, it is prone to getting trapped in local optima [17]. To address this limitation, several strategies have been proposed. One common approach is the introduction of mutation operators, which increase population diversity and enhance the ability to escape local optima [18]. In addition, greedy search strategies can efficiently generate high-quality initial solutions, thereby reducing computational cost while improving overall performance. Moreover, hybrid methods combining PSO with greedy forward selection have been developed to integrate global exploration with local refinement, achieving a better balance between search efficiency and solution quality. This approach improves both classification accuracy and feature subset selection [19]. Despite these advances, BPSO variants have not been applied to optimize SAT logic rules in DHNN. This

reveals an important research gap. Developing an efficient and effective BPSO framework to enhance learning and retrieval in DHNN remains an open challenge.

3. Weighted C-type random 2 satisfiability in DHNN

In this section, we present the formulation of WCRAN2SAT, followed by a detailed description of its implementation within DHNN. The overall process consists of three phases: the logic phase, the learning phase, and the retrieval phase.

3.1. Formulation of the WCRAN2SAT

The general formulation of WCRAN2SAT is described in

$$P_{wcran2sat} = \left(\bigwedge_{i=1}^{m_1} \lambda_i^{(1)} \right) \wedge \left(\bigwedge_{j=1}^{m_2} \left(\lambda_{2j-1}^{(2)} \vee \lambda_{2j}^{(2)} \right) \right), \tag{3.1}$$

where m_1 and m_2 denote the numbers of first-order and second-order clauses, respectively. The literal $\lambda_i^{(1)}$ represents the literal in the i-th first-order clause, whereas $\lambda_{2j-1}^{(2)}$ and $\lambda_{2j}^{(2)}$ denote the literals in the j-th second-order clause. The first-order literal satisfies $\lambda_i^{(1)} \in \{p_i, \neg p_i\}$, and the second-order literal satisfies $\lambda_k^{(2)} \in \{q_k, \neg q_k\}$. Here, p_i and q_k represent positive literals, while $\neg p_i$ and $\neg q_k$ denote the corresponding negative literals.

In $P_{\text{wcran2sat}}$, the total number of negative literals, denoted by N_e, is determined according to the predefined weight parameter w:

$$N_e = w\, n. \tag{3.2}$$

In Eq (3.2), w represents the distribution of the negative literals among total neurons, and n represents the total number of literals, which can be calculated using the following equation:

$$n = m_1 + 2m_2. \tag{3.3}$$

Note that the positions of the negative literals in $P_{wcran2sat}$ are generated randomly. Even when the total number of negative literals (N_e) is the same for a given logical structure of $P_{wcran2sat}$, the positions of the negative literals may vary.

3.2. The logic phase

During the logic phase, all possible structures of $P_{wcran2sat}$, corresponding to negative literal distributions ranging from 0.1 to 0.9, are generated using the BABC algorithm [20]. Table 1 presents an example of a possible structure of $P_{wcran2sat}$ for $n = 10$ and $w = \{0.1, 0.5, 0.9\}$.

Table 1. Possible structures of $P_{wcran2sat}$ for $n = 10$ and $w = \{0.1, 0.5, 0.9\}$.

w	N_e	m_1	m_2	$P_{wcran2sat}$
0.1	1	0	5	$(b_1 \vee b_2) \wedge (\neg b_3 \vee b_4) \wedge (b_5 \vee b_6) \wedge (b_7 \vee b_8) \wedge (b_9 \vee b_{10})$
		2	4	$a_1 \wedge \neg a_2 \wedge (b_1 \vee b_2) \wedge (b_3 \vee b_4) \wedge (b_5 \vee b_6) \wedge (b_7 \vee b_8)$
		4	3	$a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge (b_1 \vee b_2) \wedge (b_3 \vee b_4) \wedge (b_5 \vee \neg b_6)$
		6	2	$a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge \neg a_5 \wedge a_6 \wedge (b_1 \vee b_2) \wedge (b_3 \vee b_4)$
		8	1	$a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge a_6 \wedge a_7 \vee \neg a_8 \wedge (b_1 \vee b_2)$
		10	0	$a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge a_6 \wedge a_7 \wedge a_8 \wedge \neg a_9 \wedge a_{10}$
0.5	5	0	5	$(\neg b_1 \vee b_2) \wedge (\neg b_3 \vee \neg b_4) \wedge (b_5 \vee \neg b_6) \wedge (\neg b_7 \vee b_8) \wedge (b_9 \vee b_{10})$
		2	4	$\neg a_1 \wedge \neg a_2 \wedge (b_1 \vee \neg b_2) \wedge (b_3 \vee \neg b_4) \wedge (b_5 \vee b_6) \wedge (b_7 \vee \neg b_8)$
		4	3	$\neg a_1 \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge (\neg b_1 \vee b_2) \wedge (\neg b_3 \vee b_4) \wedge (b_5 \vee \neg b_6)$
		6	2	$a_1 \wedge a_2 \wedge \neg a_3 \wedge a_4 \wedge \neg a_5 \wedge \neg a_6 \wedge (b_1 \vee \neg b_2) \wedge (\neg b_3 \vee b_4)$
		8	1	$a_1 \wedge \neg a_2 \wedge a_3 \wedge \neg a_4 \wedge \neg a_5 \wedge a_6 \wedge a_7 \vee \neg a_8 \wedge (b_1 \vee \neg b_2)$
		10	0	$a_1 \wedge \neg a_2 \wedge \neg a_3 \wedge a_4 \wedge a_5 \wedge \neg a_6 \wedge \neg a_7 \wedge a_8 \wedge \neg a_9 \wedge a_{10}$
0.9	9	0	5	$(\neg b_1 \vee \neg b_2) \wedge (\neg b_3 \vee \neg b_4) \wedge (\neg b_5 \vee \neg b_6) \wedge (\neg b_7 \vee b_8) \wedge (\neg b_9 \vee \neg b_{10})$
		2	4	$\neg a_1 \wedge \neg a_2 \wedge (b_1 \vee \neg b_2) \wedge (\neg b_3 \vee \neg b_4) \wedge (\neg b_5 \vee \neg b_6) \wedge (\neg b_7 \vee \neg b_8)$
		4	3	$\neg a_1 \wedge \neg a_2 \wedge \neg a_3 \wedge a_4 \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg b_3 \vee \neg b_4) \wedge (\neg b_5 \vee \neg b_6)$
		6	2	$a_1 \wedge \neg a_2 \wedge \neg a_3 \wedge \neg a_4 \wedge \neg a_5 \wedge \neg a_6 \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg b_3 \vee \neg b_4)$
		8	1	$\neg a_1 \wedge \neg a_2 \wedge \neg a_3 \wedge \neg a_4 \wedge \neg a_5 \wedge \neg a_6 \wedge \neg a_7 \vee \neg a_8 \wedge (b_1 \vee \neg b_2)$
		10	0	$\neg a_1 \wedge \neg a_2 \wedge \neg a_3 \wedge \neg a_4 \wedge \neg a_5 \wedge a_6 \wedge \neg a_7 \wedge \neg a_8 \wedge \neg a_9 \wedge \neg a_{10}$

3.3. The learning phase

The main objective of the learning phase is to obtain the synaptic weights of the DHNN. It was shown by Abdullah [3] that the synaptic weights are deterministically derived by matching the cost function with the Lyapunov energy function. The cost function $C_{P_{wcran2sat}}$ is defined according to the following equation:

$$C_{P_{\text{wcran2sat}}} = \sum_{i=1}^{m_1} \left(\prod_{j=1}^{1} Q_{ij} \right) + \sum_{i=1}^{m_2} \left(\prod_{j=1}^{2} Q_{ij} \right). \tag{3.4}$$

This unified formulation provides a consistent mathematical framework for modeling first-order and second-order terms. Specifically, the first summation aggregates the contributions of first-order clauses, where each clause is represented by a single inconsistency measure. The second summation models second-order interactions, where the joint effect of two literals within a clause is captured through their multiplicative combination, reflecting their coupled inconsistency. In Eq (3.4), Q_{ij} denotes the inconsistency measure of the j-th literal in the i-th clause of $P_{\text{wcran2sat}}$, which is defined based on the polarity of the literal as follows:

$$Q_{ij} = \begin{cases} \frac{1}{2}(1 - s_x), & \text{if } \neg x, \\ \frac{1}{2}(1 + s_x), & \text{otherwise,} \end{cases} \tag{3.5}$$

where s_x denotes the state variable associated with literal x in $P_{\text{wcran2sat}}$. This formulation defines the inconsistency measure based on the polarity of each literal. For a negated literal ($\neg x$), the inconsistency

is given by $\frac{1}{2}(1 - s_x)$; otherwise, it is given by $\frac{1}{2}(1 + s_x)$.

Similarly, the Lyapunov energy function $E_{P_{wcran2sat}}$ for DHNN is expressed in

$$E_{P_{wcran2sat}} = -\sum_{i=1}^{n} w_i^{(1)} s_i - \frac{1}{2} \sum_{\substack{i=1 \\ i\neq j}}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij}^{(2)} s_i s_j, \tag{3.6}$$

where $w_i^{(1)}$ represent the weights of neuron i in first-order clause, and $w_{ij}^{(2)}$ represents the synaptic weight between neuron i and neuron j in second-order clauses.

By matching the cost function in Eq (3.4) with the Lyapunov energy function in Eq (3.6), the synaptic weights are systematically derived through a consistent mapping. Once the synaptic weights are obtained, they are stored in the CAM to retrieve the minimum energy states of the DHNN in the retrieval phase.

3.4. The retrieval phase

During the retrieval phase, the global or local minimum energies will be obtained based on the synaptic weights stored in the CAM. The local field of each neuron is then computed using the following equation:

$$h_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} W_{ij}^{(2)} S_j + W_i^{(1)}. \tag{3.7}$$

In Eq (3.7), h_i denotes the local field of neuron i in the DHNN. In DHNN, the local field of neuron i represents the total input signal it receives, which is obtained by summing the contributions from all other neurons weighted by the pairwise connections $W_{ij}^{(2)}$, together with a bias term $W_i^{(1)}$. In addition, $S_j \in \{-1, 1\}$ denotes the state of neuron j. In this paper, the subsequent state of the neuron j is determined by

$$S_j(t+1) = \begin{cases} 1, & \text{if } \tanh(h_j) \geq 0, \\ -1, & \text{otherwise.} \end{cases} \tag{3.8}$$

In Eq (3.8), $\tanh(h_j)$ is the hyperbolic tangent activation function (HTAF), which is used to stabilize neuron states and prevent the network from being trapped in local minima. HTAF is defined as

$$\tanh(h_j) = \frac{e^{h_j} - e^{-h_j}}{e^{h_j} + e^{-h_j}}, \tag{3.9}$$

where h_j is the local field of neuron j in the DHNN, as defined in Eq (3.7). This function maps the local field to a value in the range [-1,1], providing a smooth, sigmoidal-like activation. As DHNN updates neuron states, the network energy consistently decreases until it reaches a stable state. The primary challenge in $P_{wcran2sat}$ is identifying the global minimum energy of the network, which is determined using the following equation:

$$\left| E_{P_{wcran2sat}} - E_{P_{wcran2sat}}^{\min} \right| \leq Tol. \tag{3.10}$$

This condition ensures that the final energy $E_{P_{wcran2sat}}$ of the network is sufficiently close to the minimum Lyapunov energy $E_{P_{wcran2sat}}^{\min}$, which is analytically given by

$$E_{P_{wcran2sat}}^{\min} = -\frac{m_1 + 4m_2}{4}. \tag{3.11}$$

Equation (3.10) defines the theoretical lower bound of the energy function. In this paper, the tolerance value Tol is set to 0.001. When the computed final energy $E_{P_{wcran2sat}}$ is sufficiently close to the theoretical minimum $E^{\min}_{P_{wcran2sat}}$, it implies that the global minimum energy has been reached. This condition indicates that the DHNN has converged to a global stable state, where all neurons maintain consistent outputs.

4. The proposed work

In this section, we present the proposed HBPSO framework. First, the multi-objective function guiding the first-stage optimization is formulated. Next, the fixed-parameter HBPSO is detailed, followed by its adaptive-parameter extension to further enhance stability and adaptability. Finally, a two-stage optimization strategy for WCRAN2SAT in DHNN is established, integrating global exploration and local refinement within a unified framework.

4.1. The proposed multi-objective function

In this paper, the multi-objective function is defined as follows:

$$\begin{aligned} &\max\ F(X) = (f_1(X), f_2(X), f_3(X)) \quad \text{(lexicographic order)} \\ &\quad \text{s.t.} \quad X \in \mathcal{X}, \\ &f_1(X) = \sum_{i=1}^{k} \mathbb{I}(f_{\text{fit}}(X_i) = f_{\text{fit}}^{\max}), \\ &f_2(X) = |\{X_i \mid 1 \le i \le k\}|_{\text{unique}}, \\ &f_3(X) = \frac{f_2(X)}{f_1(X)}. \end{aligned} \tag{4.1}$$

The objective function $F(X)$ is formulated as a vector-valued optimization problem consisting of three components, namely $f_1(X)$, $f_2(X)$, and $f_3(X)$, which are optimized under a lexicographic ordering scheme. Specifically, $f_1(X)$ measures the number of solutions achieving the maximum fitness value $f_{\text{fit}}^{\max}$, $f_2(X)$ quantifies the diversity of solutions in X, and $f_3(X)$ represents the ratio between diversity and optimal solutions. The lexicographic order implies a strict priority structure: $f_1(X)$ is considered first, and only when two solutions are equivalent in $f_1(X)$, the comparison proceeds to $f_2(X)$; similarly, $f_3(X)$ is evaluated only when $f_1(X)$ and $f_2(X)$ are identical.

In Eq (4.1), X denotes the set of k candidate assignment of $P_{wcran2sat}$, represented in

$$X = \{X_1, X_2, \ldots, X_k\}, \tag{4.2}$$

where each $X_i = [S_1, S_2, \ldots, S_n]$ is a vector of length n with binary elements $S_j \in \{-1, 1\}$. $f_1(X)$ represents the maximum number of satisfying interpretations that achieve the maximum fitness value.

f_{fit} is the fitness function of $P_{wcran2sat}$. It represents the total number of satisfied clauses, which is computed as follows:

$$f_{fit}(X_i) = \sum_{j=1}^{m_1+m_2} C_j. \tag{4.3}$$

The fitness function defined in Eq (4.3) evaluates a candidate assignment (X_i) by counting the number of logical clauses it satisfies. Specifically, the fitness value $f_{fit}(X_i)$ is the summation of the satisfaction indicators C_j over the first-order clause m_1 and the second-order clause m_2. C_j is calculated according to the following equation:

$$C_j = \begin{cases} 1, & \text{True}, \\ 0, & \text{False}, \end{cases} \tag{4.4}$$

where C_j is a binary variable that reflects whether the j-th clause is satisfied by the assignment (X_i). Thus, the fitness function measures the quality of a candidate assignment by the total number of clauses it satisfies, where each satisfied clause contributes one unit to the fitness score.

Based on Eq (4.3), it can be observed that the maximum fitness value f_{fit} is $m_1 + m_2$, which corresponds to the total number of clauses in $P_{wcran2sat}$. When a candidate assignment achieves this maximum value, it indicates that the optimal synaptic weight of $P_{wcran2sat}$ has been obtained. Function $f_2(X)$ measures the diversity of the optimal synaptic weights. For $P_{wcran2sat}$, a higher diversity of synaptic weights stored in the CAM implies a greater storage capacity, since more distinct weight configurations reduce redundancy among stored patterns and thus enable the network to encode a larger number of separable solutions. $f_3(X)$ focuses on the efficiency of generating unduplicated optimal synaptic weight. It is computed as the ratio between the number of unduplicated optimal synaptic weight and the total number of optimal synaptic weight. Overall, function $f_3(X)$ encourages candidate interpretations of $P_{wcran2sat}$ that are not only highly fitness value but also diverse.

4.2. The proposed hybrid binary particle swarm optimization (HBPSO)

4.2.1. The feature of HBPSO

In this paper, the novelty of HBPSO over standard BPSO is reflected in three key components.

First, we have a refined velocity update scheme incorporating sigmoid-based probabilistic mapping with a clamping mechanism to improve search stability. This design effectively controls the magnitude of velocity, preventing excessive oscillation while maintaining sufficient diversity. Theoretically, this can be viewed as constraining the search step within a bounded region, which improves convergence stability and avoids erratic updates. The sigmoid function ($\sigma(v_{ij})$) employed in this thesis is defined in

$$\sigma(v_{ij}) = \frac{1}{1 + e^{-v_{ij}}}, \tag{4.5}$$

where v_{ij} represents the velocity of the particle. In the context of DHNN, the position of the particle represents the neuron states with the bipolar values of -1 or +1. Figure 1 shows the sigmoid function of particle velocity.

As shown in Figure 1, the value of $\sigma(v_{id})$ will stabilize at 1 when $v_{id} \geq 5$ and at 0 when $v_{id} \leq -5$. In contrast, the curve is steeper within the interval $[-5, +5]$. To prevent $\sigma(v_{id})$ from stabilizing, v_{id} is constrained within the range of $[-5, +5]$. This constraint significantly increases the probability of position changes for all particles, resulting in more diverse $\sigma(v_{ij})$ values. Essentially, this approach enhances the exploration capability of the algorithm by enabling more rapid changes for all neuron states in DHNN.

Velocity clamping is based on

$$v_{ij} = \max\left(v_{min}, \min(v_{ij}, v_{max})\right), \tag{4.6}$$

where v_{max} represents the upper bound of velocity, and $v_{min} = -v_{max}$ is the velocity lower bound. To maintain a balance between exploration and convergence, an appropriate value of v_{max} is required. This is due to excessively large v_{max} may cause unstable jumps, whereas overly small values can slow convergence. Based on the finding of [21], v_{max} is set to 5 in this study.

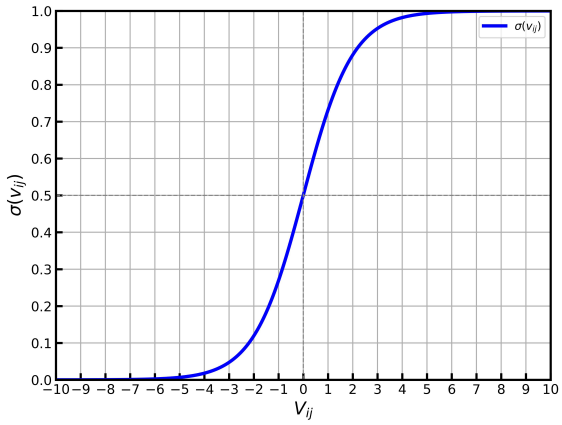


Figure 1. Sigmoid function of particle velocity.

Second, we have a mutation operator to enhance population diversity and mitigate premature convergence. To address the premature convergence issue commonly observed in BPSO, a mutation operator is incorporated into HBPSO. This operator introduces controlled randomness into particle positions, enabling the algorithm to escape local optima. From a theoretical standpoint, mutation increases the ergodicity of the search process, ensuring a non-zero probability of exploring unexplored regions of the solution space. In this paper, each position element x_{ij} undergoes sign flipping with mutation probability μ according to the following equation:

$$x'_{ij} = \begin{cases} x_{ij} \cdot -1, & \text{if } r < \mu, \\ x_{ij}, & \text{otherwise.} \end{cases} \tag{4.7}$$

Based on Eq (4.7), if $r \sim U[0, 1]$ satisfies $r < \mu$, then $x_{ij} \leftarrow -x_{ij}$; otherwise, it remains unchanged. This probabilistic mechanism enhances exploration and helps escape local optima. The mutation operator modifies the state of a randomly selected neuron by flipping its value between -1 and 1. This process enhances the local search capabilities of the algorithm by enabling the exploration of nearby solutions.

Third, we have a greedy search strategy to strengthen local exploitation capability. HBPSO further integrates a greedy search mechanism to enhance local exploitation. By selectively accepting improved solutions, this strategy accelerates convergence toward high-quality solutions. This can be interpreted as a local refinement process that reduces solution variance and improves accuracy in the later stages of optimization. The greedy strategy described as Eq (4.8) is adopted to select the better solution between the updated and current solution.

$$x_i^{t+1} = \begin{cases} x_i^{new}, & \text{if } f(x_i^{new}) > f(x_i^t), \\ x_i^t, & \text{otherwise.} \end{cases} \tag{4.8}$$

For particle i, the candidate solution x_i^{new} generated at iteration $t + 1$ is compared to the current solution x_i^t based on their fitness values, computed using Eq (4.3). If the fitness of the new solution exceeds

that of the current solution, the particle updates its state accordingly; otherwise, the current solution is preserved to prevent performance degradation.

It is noted that HBPSO and Hopfield dynamics are organized in a strictly sequential two-phase architecture rather than a hybrid scheme. In the first phase, HBPSO performs global exploration to generate high-quality initial neuron states. The velocity control, mutation, and greedy mechanisms enhance diversity and ergodicity, enabling effective exploration without being constrained by energy dynamics. In the second phase, Hopfield neural dynamics with HTAF are applied independently for local refinement. The neuron updates are governed by the Lyapunov energy function, which guarantees a monotonic decrease and ensures stable convergence to a local minimum. Therefore, the proposed framework is theoretically grounded as a combination of stochastic global search and deterministic energy-based optimization, achieving an effective balance between exploration and convergence stability.

4.2.2. The fixed-parameter HBPSO

In fixed-parameter HBPSO, each particle represents a candidate solution and moves through the search space by updating its velocity and position to approach the optimum over successive iterations. The solution corresponds to the neuron assignments of the DHNN. In this paper, particle positions and velocities are defined as

$$X^0 = \{x^1, x^2, \ldots, x^{SN}\}, \quad x^i = (x_{i1}^0, x_{i2}^0, \ldots, x_{iD}^0), \quad i = 1, 2, \ldots, SN, \tag{4.9}$$

where each particle x^i represents a candidate solution in the D-dimensional search space. The search space dimension D corresponds to the total number of neurons in the DHNN. Each dimension is initialized stochastically:

$$x_{ij}^0 = \begin{cases} +1, & \text{with probability 0.5,} \\ -1, & \text{with probability 0.5,} \end{cases} \quad j = 1, 2, \ldots, D. \tag{4.10}$$

Velocities are initialized randomly from a uniform distribution based on Eq (4.11). This stochastic initialization ensures diversity in the swarm and enables effective exploration of the solution space.

$$\mathbf{v}^i = (v_{i1}^0, v_{i2}^0, \ldots, v_{iD}^0), \quad v_{ij}^0 \sim U(v_{\min}, v_{\max}), \quad j = 1, 2, \ldots, D. \tag{4.11}$$

The velocity of each particle v_{ij} is updated as

$$\begin{aligned} v_{ij}(t+1) = w_{iner}\, v_{ij}(t) &+ c_1\, r_1 \left(p_{ij} - x_{ij}(t)\right) \\ &+ c_2\, r_2 \left(g_j - x_{ij}(t)\right), \quad i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, D, \end{aligned} \tag{4.12}$$

where w_{iner} is the inertia weight, c_1 and c_2 are cognitive and social acceleration coefficients, and $r_1, r_2 \sim U[0, 1]$ are random factors. The term p_{ij} refers to the personal best position previously attained by the i-th particle in the j-th dimension, and g_j denotes the global best position identified by the swarm in that dimension. In PSO, excessively large velocity values can cause the sigmoid output to approach 0 or 1, which in turn leads to poor exploration [21]. This issue becomes more severe in complex and large-scale optimization problems, where a more diverse search is required to avoid local optima.

To mitigate this issue, velocity clamping is applied to constrain velocity within a predefined range, resulting in a more effective search behavior.

The position of particle x_{ij} is updated according to the following equation:

$$x_{ij} = \begin{cases} 1, & \text{if } r < \sigma(v_{ij}), \\ -1, & \text{otherwise.} \end{cases} \tag{4.13}$$

In Eq (4.13), r is a random number in $(0, 1)$.

Overall, the corresponding pseudocode of the fixed-parameter HBPSO is presented in Algorithm 1.

Algorithm 1 The fixed-parameter HBPSO

Require: Number of particles SN, maximum iterations $I_{\max}$, dimension D, mutation rate μ
Ensure: a set of optimized neuron assignments X^*

```
Initialize global best solution g
for i = 1 to SN do
    Initialize particle position x_i and velocity v_i using Eqs (4.9)–(4.11)
end for
for t = 1 to I_max do
    for i = 1 to SN do
        for j = 1 to D do
            Update velocity v_ij using Eqs (4.5) and (4.6)
            Update position x_ij using Eq (4.13)
        end for
        Evaluate fitness f_i using Eq (4.3)
        Update personal best p_i and global best g
        for j = 1 to D do
            Apply mutation with rate μ using Eq (4.7)
        end for
    end for
end for
return set of optimized neuron assignments X*
```

4.3. The adaptive-parameter HBPSO

In addition to the aforementioned components, namely a refined velocity update scheme, a mutation operator, and a greedy search strategy, the adaptive-parameter HBPSO introduces a dynamic adjustment mechanism. Specifically, this mechanism adaptively updates three key parameters: (a) the inertia weight, (b) the acceleration coefficients, and (c) the mutation rate. This dynamic adjustment enables a more effective balance between global exploration and local exploitation, thereby improving convergence performance and solution quality. The overall execution procedure of the adaptive-parameter HBPSO remains identical to that of the fixed-parameter version, with the key difference being that the parameter values are dynamically updated during the optimization process.

4.3.1. Adaptive inertia weight

In PSO, the search process initially emphasizes exploration and gradually shifts toward exploitation as iterations progress [22]. Moreover, a smaller inertia weight promotes exploitation, whereas a larger inertia weight enhances exploration. In this paper, to align the dynamic inertia weight with the search mechanism of PSO, we propose an oscillation-based strategy for adjusting inertia weight by introducing periodic variations. The proposed inertia weight adaptation formula is

$$w_{\text{iner}} = w_{\min} + (w_{\max} - w_{\min}) \cdot \left(1 - e^{-\frac{\alpha}{T}}\right) + A \cdot \sin\left(B \cdot \pi \cdot \frac{t}{T}\right). \tag{4.14}$$

Parameter w_{iner} is the inertia weight, where w_{max} and w_{min} denote its maximum and minimum values, respectively. The decay rate is controlled by α, while parameters A and B determine the amplitude and frequency of the oscillation. Additionally, t represents the current iteration, and T denotes the total number of iterations.

The inertia weight in Eq (4.14) is designed as a hybrid adaptive-oscillatory scheme, combining an exponential decay term with a bounded sinusoidal perturbation. The exponential component provides a smooth transition from exploration to exploitation, while the oscillatory term enhances population diversity and helps escape local optima. To ensure stability, the inertia weight is strictly constrained within $[w_{\min}, w_{\max}]$, preventing divergence or destabilization. Additionally, ablation studies confirm that the oscillatory component improves convergence robustness and solution quality.

4.3.2. Adaptive acceleration coefficients

The update strategy for the parameters' acceleration coefficients c_1 and c_2 is designed to enhance global exploration in the early iterations and promote local exploitation in the later stages. Specifically, c_1 gradually decreases from its maximum value to the optimal range, while c_2 slowly increases from its minimum value to the optimal range. Inspired by [23], the dynamic update of the acceleration coefficients is according to the following equation:

$$\begin{aligned} c_1 &= c_1^{max} - \frac{(c_1^{max} - c_1^{min})}{T} \cdot t, \\ c_2 &= c_2^{max} + \frac{(c_2^{max} - c_2^{min})}{T} \cdot t. \end{aligned} \tag{4.15}$$

Here, c_1^{max} and c_2^{max} represent the maximum values of c_1 and c_2, respectively, while c_1^{min} and c_2^{min} denote their minimum values.

4.3.3. Adaptive mutation rate

To determine the most effective mutation rate and enhance adaptability across search space scales, it is crucial to explore a dynamic mutation rate for HBPSO. Inspired by the work of Song et al. [24], we propose a dynamic mutation strategy that decreases based on the total number of neurons in DHNN (n). Furthermore, two dynamic mutation rate strategies are introduced, namely linear decay and exponential decay. The linear decay of the mutation rate is described in

$$\begin{aligned} mr &= mr_{inid} + \left(\frac{mr_{end} - mr_{init}}{T_{\max} - T_{min}}\right) \cdot (t - T_{min}), \\ mr_{end} &= 0.2 + r \cdot 0.1, \quad r \in [0, 1]. \end{aligned} \tag{4.16}$$

Here, m_r represents the mutation rate, while mr_{inid} and mr_{end} denote its initial and final values of the linear decay, respectively. Additionally, T_{max} and T_{min} represent the maximum and minimum values of n. Letter t is the current value of n. The exponential decay is presented in

$$mr(n) = mr_{\text{inie}} \cdot e^{dr \cdot n}. \tag{4.17}$$

In this equation, $mr(n)$ denotes the mutation rate at iteration n, mr_{inie} represents the initial mutation rate at $n = 0$, dr is the exponential rate parameter that controls the rate of change of the mutation process over iterations, and n is the iteration index indicating the current generation number in the optimization process. Additionally, e is the natural exponential base.

4.4. Tow-stage optimization for WCRAN2SAT in DHNN

4.4.1. The first-stage optimization

In the first stage, optimization is performed over neuron assignments rather than synaptic weights. HBPSO is used to search for a consistent interpretation that satisfies all logical clauses and minimizes the cost function. The synaptic weights are deterministically derived via the Wan Abdullah method by matching the cost function with the Lyapunov energy function and are therefore not treated as decision variables. Consequently, the search process operates over the combinatorial space of neuron states under fixed weights.

To address the presence of multiple local minima, HBPSO is employed to enhance exploration and improve the likelihood of reaching a globally consistent solution. Specifically, HBPSO is employed to enhance the learning capability by optimizing the multi-objective function defined in Eq (4.1). In this stage, each particle encodes the neuron assignments of the DHNN. In this process, the duplicate-checking mechanism plays a crucial role, as it ensures the maximization of $f_1(X)$, $f_2(X)$, and $f_3(X)$ defined in Eq (4.1).

The proposed method adopts a lexicographic optimization strategy, where objectives are evaluated in a strict priority order [25], ensuring that higher-priority objectives dominate the decision process. Specifically, candidate solutions are evaluated sequentially: $f_1(X)$ is first assessed, and only if it is not worse than the current solution does the evaluation proceed to $f_2(X)$; $f_3(X)$ is considered only when both $f_1(X)$ and $f_2(X)$ are satisfied. Accordingly, the multi-objective problem is reformulated as a hierarchical decision-making process with strict priority ordering. A particle adopts the new solution if it is superior under this lexicographic comparison; otherwise, the solution is retained, ensuring that lower-priority objectives are considered only when higher-priority criteria are equivalent. The corresponding pseudocode is presented in Algorithm 2.

It should be noted that the computational complexity of the duplicate-checking mechanism is $O(k^2n)$, where k denotes the number of candidate solutions and n represents the dimensionality of each solution vector.

The pseudocode of the first-stage optimization phase is provided in Algorithm 3. It should be noted that, when applying the duplicate-checking mechanism in Algorithm 2, the multi-objective optimization defined by Eq (4.1) is implicitly implemented via a lexicographic decision rule. In this framework, the multi-objective evaluation is decomposed into ordered comparisons of $f_1(X)$, $f_2(X)$, and $f_3(X)$.

Algorithm 2 Duplicate-checking mechanism with multi-objective control

Require: A set of candidate solutions $\mathcal{X} = \{X_1, X_2, \ldots, X_k\}$, where $X_i = [S_1, S_2, \ldots, S_n]$
Ensure: A set of unduplicated solutions $\mathcal{X}^*$

Initialize $\mathcal{X}^* \leftarrow \emptyset$
for each $X_i \in \mathcal{X}$ **do**
  if X_i satisfies $f_1(X)$ defined in Eq (4.1) **then**
    isDuplicate $\leftarrow$ **false**
    for each $X_j \in \mathcal{X}^*$ **do**
      if $X_i = X_j$ (element-wise equality) **then**
        isDuplicate $\leftarrow$ **true**
        break
      end if
    end for
    if isDuplicate = **false then**
      $\mathcal{X}^* \leftarrow \mathcal{X}^* \cup \{X_i\}$
    end if
  end if
end for
Apply multi-objective selection:
if $|\mathcal{X}^*|$ satisfies the maximum value of $f_2(X)$ defined in Eq (4.1) **then**
  if $\frac{|\mathcal{X}^*|}{|\mathcal{X}|}$ satisfies the maximum value of $f_3(X)$ defined in Eq (4.1) **then**
    return $\mathcal{X}^*$
  end if
end if

Algorithm 3 First-stage optimization procedure based on HBPSO

Require: Number of particles SN, maximum iterations $I_{\max}$, dimension D, mutation rate μ
Ensure: Set of optimized neuron assignments X^*

Initialize global best solution g
Call HBPSO optimization module (Algorithm 1)
Obtain optimized particle set X^* from HBPSO
Apply duplicate-checking mechanism using Algorithm 2
return unduplicated optimized neuron assignments X^*

4.4.2. The second-stage optimization

In the second-stage, the optimization procedure is designed as a two-phase sequential pipeline consisting of HBPSO-based initialization and Hopfield-based refinement. In the first phase, HBPSO is employed as a metaheuristic search strategy to generate high-quality initial neuron state configurations. In the second phase, Hopfield neural dynamics with the HTAF activation function are applied to each selected initial state.

In this formulation, HBPSO and Hopfield dynamics operate in a strictly sequential manner. HBPSO

is responsible for global exploration and initialization, while Hopfield dynamics performs local refinement and energy minimization. The synaptic weights remain fixed throughout the process and are not involved in the optimization procedure. Instead, they are predetermined by the Wan Abdullah method. Next, a duplicate-checking mechanism is applied to ensure that the obtained solutions correspond to distinct unduplicated stable states. Finally, duplicate removal is applied to ensure that the obtained solutions correspond to distinct unduplicated energy states.

Notably, the fitness function $f_{ret}(X_i)$ in this stage is defined as

$$f_{ret}(X_i) = \left|E_{P_{wcran2sat}} - E^{\min}_{P_{wcran2sat}}\right|, \qquad f_{ret}(X_i) \leq Tol. \tag{4.18}$$

The fitness function quantifies the proximity of a candidate state to the global minimum of the energy landscape, ensuring that only neuron states sufficiently close to the optimal configuration are retained. The pseudocode for this phase is presented in Algorithm 4.

Algorithm 4 Second-stage optimization procedure

Require: Number of particles SN, maximum iterations $I_{\max}$, dimension D, mutation rate μ
Ensure: Final consistent neuron states

Phase 1: HBPSO-based Initialization
Apply Initialization using Algorithm 1
Phase 2: Hopfield Refinement with HTAF
for each initial state $x \in X^{(0)}$ **do**
    for $t = 1$ to $I_{\max}$ **do**
        Compute local field h_i using Eq (3.7)
        Update neuron state using HTAF Eq (3.9)
        Compute energy using Eq (3.6)
        if energy converges **then**
            break
        end if
    end for
    Store final state s
end for
Apply duplicate removal using Algorithm 2

5. Experimental setup

In this section, we present the experimental design and evaluation framework for validating the proposed method. The experimental environment is first introduced. Benchmark approaches and a parameter design strategy are then described. Performance metrics are formulated for objective assessment, and the simulation dataset is presented.

5.1. Experimental environment and configuration

5.1.1. Experimental environment and constant parameters

In this study, all experiments are conducted as simulations using Visual Studio Code (version 1.87.0), with all programs written in C. To ensure consistency and reduce potential bias, all experiments are performed in the same simulation environment and on the same device: a personal computer running Windows 11, equipped with a 12th Gen Intel(R) Core(TM) i7-12700H processor and 32 GB of RAM.

All datasets used in the experiments are synthetically generated by the program. This choice is motivated by the major goals of enhancing the diversity of logical representations and improving the storage capacity of the network through unduplicated synaptic weight configurations and corresponding neuron states. Simulated data provide a fully controlled and scalable evaluation environment, enabling precise regulation of key factors such as variability and the distribution of minimum energy states. This ensures systematic, consistent, and reproducible evaluation under experimental conditions. Moreover, simulated datasets offer a flexible and easily generated testbed for exploring a wide range of parameter configurations, making them particularly suitable for assessing the robustness and behavioral characteristics of the proposed framework.

The constant parameters are listed in Table 2. These parameters are selected based on a combination of widely accepted settings in the literature and empirical tuning. The population size and maximum iteration number are chosen to balance computational cost and convergence performance, as commonly adopted in swarm-based optimization methods. Additionally, there are five clause structural configurations in WCRAN2SAT labeled Case 1 through Case 5. Case 1 corresponds to the lowest proportion of first-order clauses, whereas Case 5 corresponds to the highest proportion. For brevity, only the results for Cases 1, 3, and 5 are presented in this paper, as these cases adequately capture the most significant differences among the configurations of $P_{wcran2sat}$ for all five cases. This is summarized in Table 3.

Table 2. Constant parameters.

Parameter	Value
Number of combinations (c)	100 [8]
Number of trials	100 [8]
Weight value (w)	0.1, 0.5, 0.9
Clause structural configurations	Case 1, Case 3, Case 5
Number of neurons (n)	[10, 100] [4]
Number of learning (N_{learn})	100
Learning iterations (N_{lite})	$N_{\text{lite}} \leq N_{\text{learn}}$ [4]
Synaptic weight method	Wan Abdullah method [3]
Population size (N_{pop})	200
Maximum iterations ($Iter_{\max}$)	100
Tolerance value (Tol)	0.001 [8]

Table 3. First-order and second-order clause numbers for representative cases.

n	Case 1	Case 3	Case 5
10	{0, 5}	{4, 3}	{8, 1}
20	{0,10}	{10,5}	{18,1}
30	{0,15}	{14,8}	{26,2}
40	{0,20}	{20,10}	{36,2}
50	{0,25}	{24,13}	{44,3}
60	{0,30}	{30,15}	{54,3}
70	{0,35}	{34,18}	{62,4}
80	{0,40}	{40,20}	{72,4}
90	{0,45}	{44,23}	{80,5}
100	{0,50}	{50,25}	{90,5}

5.2. Comparison of models and parameters

5.2.1. Comparison of models

The comparison models in the the fist-stage optimization are summarized as follows:

(a) Genetic algorithm (GA) [4]: The GA is an evolutionary optimization method that applies selection, crossover, and mutation operators to effectively balance exploration and exploitation in the search space.

(b) Binary artificial bee colony (BABC) [26]: The BABC algorithm coordinates employed, onlooker, and scout bees to explore the search space, further enhanced by a NAND operator for generating candidate solutions.

(c) Binary differential evolution algorithm (BDEA) [27]: The BDEA evolves candidate solutions through mutation, crossover, and selection, while a sigmoid mapping ensures compatibility with a binary search space.

(d) Election algorithm (EA) [5]: The EA iteratively selects the optimal solution by combining positive and negative advertisement, coalition formation, and final evaluation steps.

(e) Binary gravitational search algorithm (BGSA) [28]: The BGSA models candidate solutions as masses that move under fitness-weighted gravitational forces, with a V-shaped probability function facilitating binary representation.

(f) Gray wolf optimization (GWO) [4]: The GWO algorithm leverages a hierarchical social structure, comprising Alpha, Beta, Delta, and Omega wolves, to guide the exploration and exploitation of candidate solutions.

(g) Binary harmony search (BHS) [29]: The BHS generates new candidate solutions by considering harmony memory and performing pitch adjustments, achieving a balance between exploitation and exploration.

(h) Binary moth-flame optimization (BMFO) [30]: The BMFO algorithm simulates moths following logarithmic spirals around flames, with a transfer function adapting continuous movements into a binary search space.

(i) Binary particle swarm optimization (BPSO) [21]: The BPSO maps particle velocities through a sigmoid function to determine their positions in a binary search space.

(j) Binary teaching-learning-based optimization (BTLBO) [31]: This algorithm consists of teacher and learner phases, using a mapping function to handle binary solutions while promoting diversity among candidate solutions.

(k) Exhaustive search (ES) [32]: The ES method systematically enumerates all possible solutions to guarantee optimality, although it incurs a high computational cost for large-scale problems.

The comparison models in the the second-stage optimization are summarized as follows:

(a) Binary particle swarm optimization (BPSO) [21]: The BPSO algorithm employs a sigmoid function to map particle velocities into binary positions, enabling effective exploration of the search space.

(b) Hybrid election algorithm (HEA) [6]: The HEA extends the standard election algorithm by introducing a caretaker party stage and a shift mutation operator, which increase population diversity and enhance exploration of the search space.

(c) Estimation of distribution algorithm (EDA) [8]: The EDA generates new solutions using a probabilistic model, specifically a Gaussian distribution, by updating model parameters based on the current population to focus the search on promising regions.

(d) Hybrid Boltzmann Hopfield neural network (BHNN) [7]: The BHNN incorporates a Boltzmann machine within a Hopfield network, performing stochastic optimization through simulated annealing until thermal equilibrium is reached.

(e) Hopfield neural network with mean field theory (MFTHNN) [9]: The MFTHNN integrates mean-field theory into the Hopfield network, approximating the influence of all other neurons as an average to improve retrieval performance.

(f) Hyperbolic tangent activation function (HTAF): The HTAF, defined in Eq (3.9), stabilizes neuron states and prevents the network from being trapped in local minima.

5.2.2. Private parameters of the comparison models

The parameter settings used in the first-stage optimization for HBPSO, BPSO, GA, BABC, BDEA, BGSA, GWO, BHS, BMFO, BTLBO, ES, and EA are summarized in Table 4. It was noted that a mutation strategy was adopted in HBPSO/BPSO to enhance exploitation and prevent the algorithm from becoming trapped in local optima [33]. Therefore, a relatively small mutation rate is preferred, as larger values may introduce excessive randomness and weaken exploitation. Based on [34], the mutation rate is set to 0.01. The inertia weight is set to 1.0 in accordance with the characteristics of BPSO, in which the velocity denotes the probability of bit flipping rather than a physical displacement, being mapped through a sigmoid function [15]. This setting preserves velocity information, maintaining variability in flipping probabilities and preventing premature stagnation. In combination with the small mutation rate, which provides only mild perturbations, the algorithm can effectively escape local optima while maintaining a balanced trade-off between exploration and exploitation.

Table 4. Private parameters of the comparison models for the first-stage optimization.

Algorithm	Parameter	Value
HBPSO/BPSO	c_1	2.0 [35]
	c_2	2.0 [35]
	Inertia weight (w_{iner})	1.0
	Mutation rate (R_m)	0.01 [34]
GA [4]	Selection rate	0.1
	Crossover rate	1
	Mutation rate (R_m)	0.01
BABC [4]	Number of employed bees	50
	Number of onlooker bees	50
	Number of scout bees	10
	Number of trail	5
BDEA [27]	Scaling factor (F)	0.8
	Crossover rate (CR)	0.5
BGSA [28]	Initial gravitational constant (α)	0.8
	Gravitational constant decay ratio (β)	0.2
GWO [4]	–no private parameters–	
BHS [29]	Harmony memory consideration rate (HMCR)	0.8
	Pitch adjusting rate (PAR)	0.2
	Number of new candidates	72
BMFO [30]	Threshold value of standard deviation (T)	100
BTLBO [31]	Teaching factor	1.0
	Threshold value of standard deviation (T)	$1e^{-0.5}$
ES	–no private parameters–	
EA/HEA [5, 6]	Number of parties	4
	Positive advertisement ratio	0.5
	Negative advertisement ratio	0.5
	Mutation rate	0.1

The second-stage optimization adopts the same parameter configurations for HBPSO and PSO as those specified in Table 4. In contrast, the parameter settings for the remaining approaches, including EDA, BHNN, MFTHNN, and THAF, are provided in Table 5. To ensure a fair comparison of algorithm performance, all parameter settings reported in the table are directly adopted from the corresponding reference studies. This avoids introducing bias caused by inconsistent or manually tuned parameters, and guarantees that each algorithm is evaluated under its originally proposed configuration. Therefore, the comparison reflects the intrinsic performance of the algorithms rather than differences in parameter selection.

Table 5. Private parameters of the comparison models for the second-stage optimization.

Algorithm	Parameter	Value
EDA [8]	Mutation rate	0.01
	Selection rate	0.1
BHNN/MFTHNN [9]	Tolerance value	0.001
	Selection rate	0.1
	Temperature (T)	70

5.3. Performance metrics

5.3.1. Storage capacity in DHNN

In classical Hopfield networks, storage capacity is defined as the maximum number of patterns that can be stored as stable attractors and reliably retrieved from corrupted or incomplete inputs, typically normalized by the network size n as the ratio p/n. A well-known theoretical result shows that for large networks with random, uncorrelated bipolar patterns, this capacity is approximately $p/n \approx 0.138$, representing the critical threshold beyond which retrieval performance rapidly degrades due to interference and the emergence of spurious attractors.

However, in this DHNN-based optimization framework, this classical definition is difficult to directly apply or measure. This is primarily because the notion of a fixed, predefined set of patterns p is not explicitly given in our model. Instead, patterns emerge implicitly through the optimization dynamics and are represented by synaptic weight configurations associated with energy minima. As a result, there is no straightforward way to enumerate or predefine all candidate patterns for evaluating successful storage and retrieval in the classical sense. Therefore, the traditional metric p/n cannot be directly computed, and storage behavior must instead be assessed through empirical proxies such as convergence to stable states and diversity of generated optimal solutions.

To address this issue, a well-defined alternative evaluation strategy grounded in observable system behavior is adopted in this paper. Specifically, the effective storage capability of the DHNN is assessed from two complementary perspectives: (a) the number of unduplicated optimal synaptic weight configurations generated during the first-stage optimization process, and (b) the retrieval performance, quantified by the number of unduplicated global minimum energy solutions obtained during the second-stage optimization process. This evaluation strategy is motivated by the following considerations.

First, a larger number of unduplicated optimal synaptic weight configurations reflects greater diversity in the underlying energy landscapes. Such diversity implies a richer set of potential stable states that the network can represent, thereby indicating an increased capacity to encode distinct memory structures. Hence, it serves as an empirical indicator of enhanced effective storage capability.

Second, a higher number of unduplicated global minimum energy solutions indicates greater diversity in the final states of the DHNN, suggesting the presence of more distinct stable attractors. This, in turn, reflects a stronger ability of the network to converge to stable states from perturbed initial conditions. Therefore, it provides an additional empirical indicator of improved effective storage capability, rather than a direct measure of classical storage capacity.

5.3.2. Metrics for the first-stage optimization

In the first-stage optimization, the performance metrics are designed to evaluate the capability of maximizing the multi-objective function (4.1). To better assess the proposed method from a diversity-oriented perspective, the evaluation particularly focuses on solution quality and diversity characteristics. Specifically, three metrics are employed: (a) the ratio of satisfied clauses R_l^s, (b) the number of unduplicated optimal synaptic weights N_l^u, which quantifies the diversity of generated configurations, and (c) the diversity rate R_l^u.

The ratio of satisfied clauses R_l^s evaluates the effectiveness of maximizing $f_1(X)$ by measuring the proportion of solutions that achieve the maximum fitness during the learning phase. A higher R_l^s

indicates stronger capability in obtaining optimal synaptic weights of the DHNN. Specifically, $R_l^s = 1$ means all solutions reach the optimal fitness, whereas $R_l^s = 0$ indicates that no optimal synaptic weight is obtained. The metric is computed according to the following equation:

$$R_l^s = \frac{1}{n} \sum_{i=1}^{c} N_l^s, \tag{5.1}$$

where c denotes the number of combinations in the learning phase, and N_l^s is the number of satisfied clauses generated in each iteration.

In this paper, the unduplicated optimal synaptic weights N_l^u are those achieving the maximum fitness without duplication. For instance, $S_1 = -1, -1, -1, -1, -1, -1$ and $S_2 = -1, -1, -1, -1, -1, 1$ are unduplicated since they differ in the last bit. N_l^u quantifies the ability of the model to satisfy the objective function $f_2(X)$ and is computed using the following equation:

$$N_l^u = \sum_{i=1}^{c} n_i^u, \tag{5.2}$$

where c denotes the number of combinations, and n_i^u represents the number of unduplicated optimal synaptic weight configurations obtained in the i-th combination. Based on Eq (5.2), a higher value of N_l^u indicates a greater capacity to generate unduplicated optimal synaptic weights. The rato of diversity R_l^u is to evaluate the efficiency of the model in maximizing the objective function $f_3(X)$. R_l^u is calculated according to the following equation:

$$R_l^u = \frac{N_l^u}{N_l^s}, \tag{5.3}$$

where N_l^u represents the number of unduplicated optimal synaptic weights, while N_l^s denotes the total number of optimal synaptic weights. A higher R_l^u indicates a greater proportion of unduplicated optimal synaptic weights within the total solution set, reflecting increased efficiency in generating solutions.

5.3.3. Metrics for the second-stage optimization

In the second-stage optimization, metrics are employed to accurately evaluate the performance of the proposed method in enhancing the generation of final neuron states. These states are assessed based on the unduplicated global minimum energies. Two performance metrics are used in this stage: (a) the total number of unduplicated global minimum energies N_r^u, and (b) the ratio of unduplicated global minimum energies R_r^u. N_r^u measures the method's capability to produce unduplicated global minimum energies and is calculated according to the following equation:

$$N_r^u = \sum_{i=1}^{n} n_i. \tag{5.4}$$

In Eq (5.4), n denotes the number of iterations, and n_i represents the number of unduplicated final global minimum energies. R_r^u evaluates the efficiency of the proposed method in producing an unduplicated global minimum, as calculated based on

$$R_r^u = \frac{N_r^u}{N_r^e}, \tag{5.5}$$

where N_r^u is the total number of unduplicated final global minimum energies. N_r^e is the number of global minimum energies. Based on Eq (5.5), higher R_r^u values indicate greater efficiency in generating unduplicated global minimum energies.

6. Results and discussions

In this section, we evaluate the performance of the proposed HBPSO under fixed and adaptive parameter settings. First, the first-stage optimization performance of the fixed-parameter HBPSO is investigated. Then, the second-stage optimization performance is analyzed. A sensitivity analysis is subsequently conducted. Finally, the robustness of the adaptive-parameter HBPSO is evaluated under dynamic and noisy environments.

6.1. First-stage optimization performance for the fixed-parameter HBPSO

6.1.1. The ratio of satisfied clause

The ratio of satisfied clause (R_l^s) is used to evaluate whether the comparison methods meet the first objective of the multi-objective in Eq (4.1). The comparison of R_l^s values in Cases 1, 3, and 5 for w = 0.1, 0.5, and 0.9 are shown in Figure 2. The results indicate that in Case 1, HBPSO, BABC, BPSO, BDEA, HS, and GA consistently achieve the maximum R_l^s value, whereas EA, BGSA, GWO, BMFO, and ES fail to do so. In particular, ES performs the worst when $n \geq 40$. The performance of TLBO, GWO, and ES is influenced by the weight value w. This outcome can be attributed to the low ratio of first-order clauses in Case 1, which enables metaheuristic algorithms with balanced global and local search capabilities to obtain satisfactory assignments efficiently. For Case 3, only HBPSO, BABC, and BPSO maintain the maximum R_l^s value. The increased ratio of first-order clauses reduces the probability of achieving satisfactory assignments for other approaches. BDEA becomes less effective when n is large, while BHS suffers from premature convergence, particularly when $n \geq 90$. For Case 5, only BPSO and HBPSO consistently achieve the maximum R_l^s value across all combinations of w and n. The higher proportion of first-order clauses makes these instances more challenging.

Based on Figure 2, Case 5 presents the greatest difficulty in achieving the maximum value of R_l^s. In other words, it constitutes the most challenging scenario and is therefore the most suitable for assessing the performance of the baseline models. Accordingly, Table 6 reports the results of the Friedman test for the ratio of satisfied clauses (R_l^s) at a 0.95 confidence level. The test statistic χ^2 follows a Chi-square distribution with 11 degrees of freedom. The corresponding p-value indicates that the performance differences among the models are statistically significant. Therefore, a post-hoc analysis is conducted to further identify pairwise differences among the models. A critical difference (CD) diagram is employed to visualize the results of the Nemenyi post-hoc test, illustrating the average ranks of the algorithms and the statistically significant differences among them. In the CD diagram, if the difference in average ranks between models does not exceed the critical difference, they are considered not significantly different at the chosen significance level. Such models are grouped together and connected by red lines.

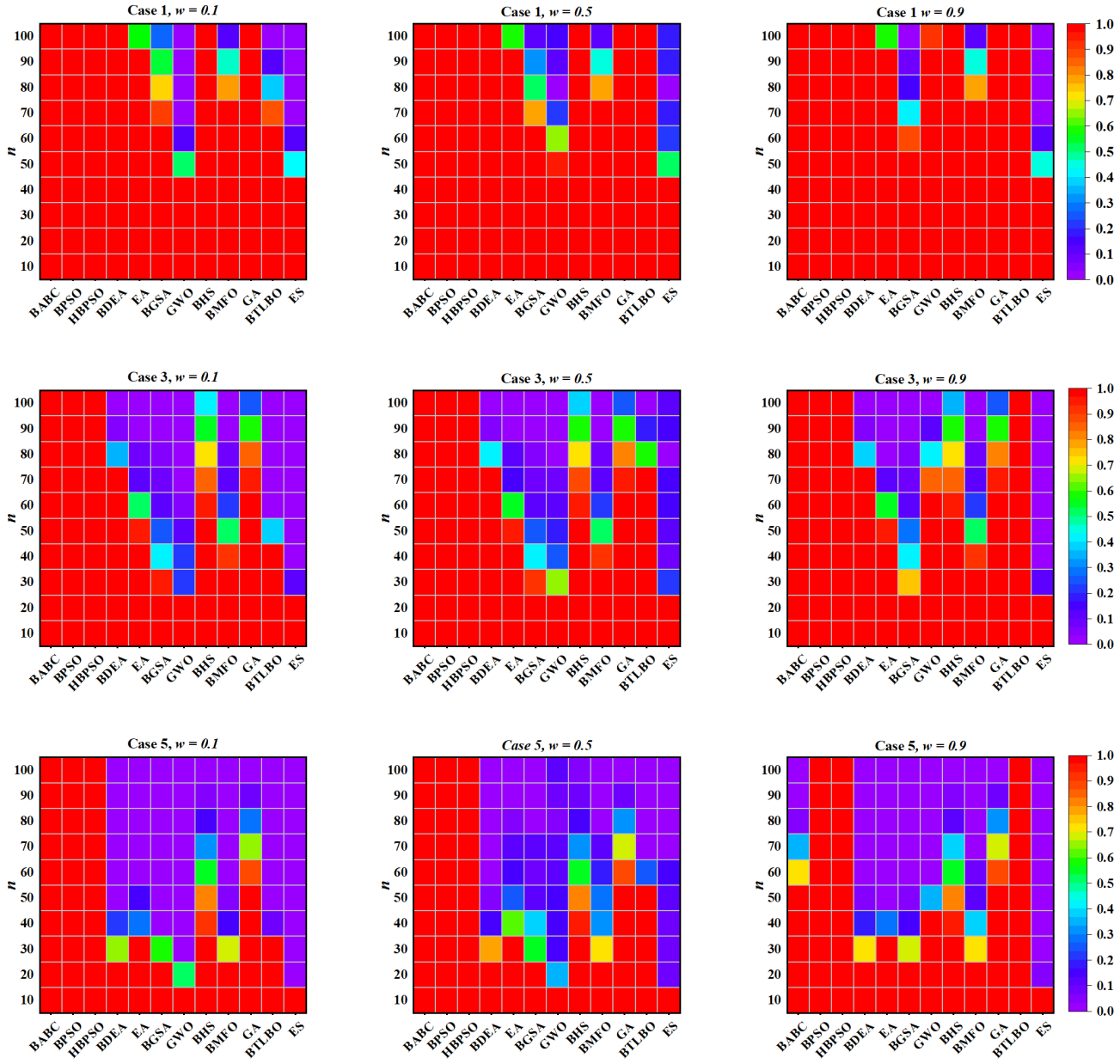


Figure 2. Heatmap of comparison of ratio of satisfied clause (R_l^s) values in Cases 1, 3, and 5 for $w = 0.1$, 0.5, and 0.9. The horizontal axis represents different algorithms, and a shared legend is placed on the rightmost side of each row to improve readability.

Table 6. Friedman test results for ratio of satisfied clause (R_l^s) in Case 5.

r	χ^2	df	p-value	Final hypothesis
0.1	78.653	11	2.68×10^{-12}	Reject H_0
0.5	69.738	11	1.37×10^{-10}	Reject H_0
0.9	77.396	11	4.69×10^{-12}	Reject H_0

Figure 3 shows the CD diagram of the ratio of satisfied clauses (R_l^s) for Case 5 with w = 0.1, 0.5, and 0.9. This figure reveals a performance stratification. Specifically, BABC, BPSO, HBPSO, and BTLBO form a stable top-performing group with closely clustered average ranks and no statistically significant differences among them. In contrast, ES, BDEA, BGSA, BMFO, and GWO consistently exhibit significantly lower performance, showing rank differences exceeding the critical difference when compared with the top-tier methods. Algorithms such as GA, BHS, and EA lie in an intermediate region, where their performance is not consistently distinguishable from either the best or worst groups depending on the value of w. Overall, the results demonstrate a robust and statistically validated separation between high-performing swarm-based optimizers and weaker baseline methods across all scenarios.

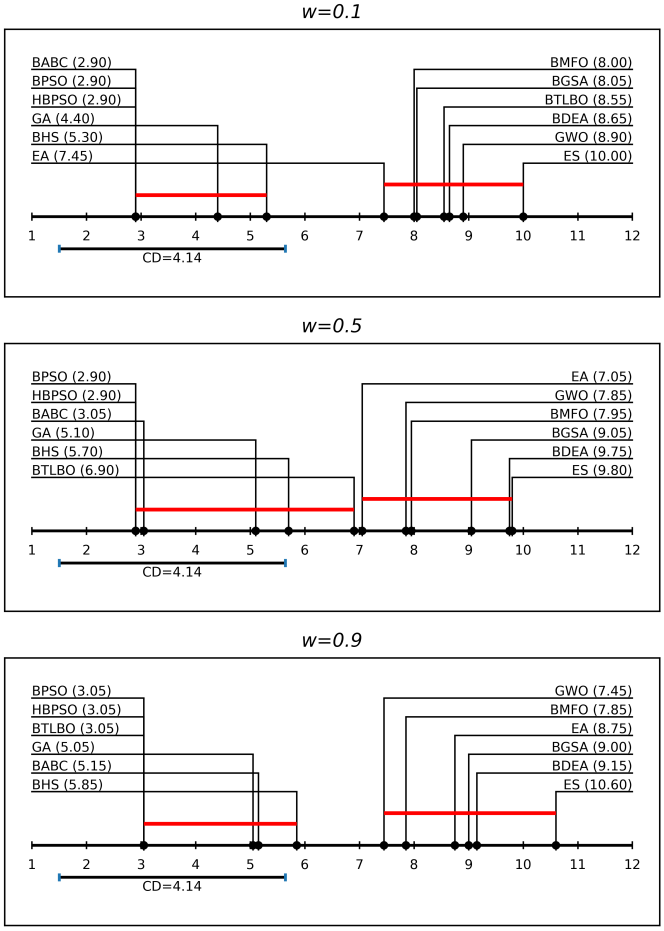


Figure 3. The CD diagram of the ratio of satisfied clauses (R_l^s) for Case 5 with w = 0.1, 0.5, and 0.9. The red lines indicate groups of algorithms that are not statistically significantly different according to the Nemenyi post-hoc test.

The ratio of satisfied clauses (R_l^s) is evaluated using the fitness function $f_1(x)$, as defined in the multi-objective formulation in Eq (4.1). All baseline methods in the experiments adopt this fitness function for consistency. Furthermore, a lexicographic optimization strategy is employed in this study. Among all test cases, Case 5 is the most challenging in terms of achieving the maximum value of R_l^s, making it particularly suitable for evaluating the convergence capability of the baseline models. Therefore,

the number of iterations required to reach the maximum R_l^s in Case 5 is used as the primary metric for assessing convergence performance. Figure 4 shows the convergence performance comparison of HBPSO and baseline algorithms under different problem scales in Case 5.

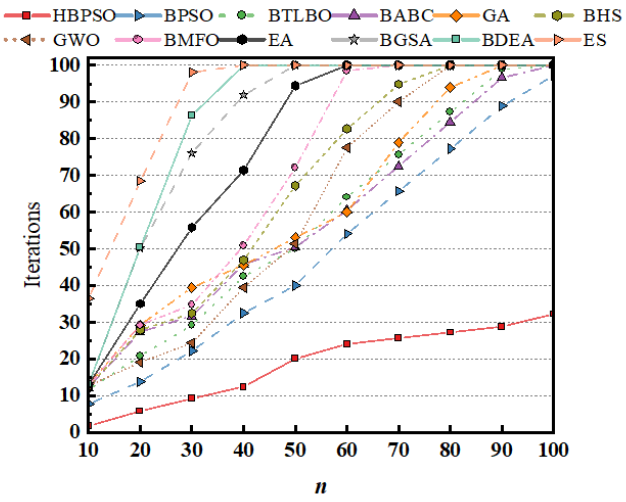


Figure 4. Convergence performance comparison of HBPSO and baseline algorithms under different problem scales.

Based on Figure 4, HBPSO consistently demonstrates faster and more stable convergence behavior compared to all baseline algorithms. Specifically, HBPSO exhibits a gradual and smooth increase in convergence values, reaching only 32.30 at n=100, whereas most competing algorithms rapidly approach or reach the maximum value (100), indicating premature convergence or stagnation.

HBPSO demonstrates strong convergence performance due to the synergistic effect of multiple mechanisms. First, the refined velocity update scheme enhances search stability and mitigates excessive oscillations. Second, the mutation operator improves population diversity, enabling the algorithm to escape local optima and reduce the risk of premature convergence. Finally, the greedy search strategy strengthens local exploitation, enabling more effective refinement of promising solutions. Together, these mechanisms maintain a good balance between exploration and exploitation, resulting in improved convergence efficiency and robustness.

In contrast, the acceleration mechanism employed in BGSA does not perform effectively [36], resulting in a lower probability of obtaining optimal synaptic weights compared to other metaheuristic approaches. The BMFO algorithm [30] suffers from an imbalance between exploration and exploitation, which limits its capability to identify optimal synaptic weights for $P_{wcran2sat}$. Furthermore, GWO tends to generate a larger proportion of bits with value -1 than $+1$, thereby increasing the likelihood of obtaining optimal synaptic weights for larger weight values w in $P_{wcran2sat}$. This characteristic explains its improved performance under higher weight settings. Similarly, TLBO also produces more bits with value -1 within the solution population, leading to better R_l^s values as w increases. In contrast, ES lacks effective exploration and exploitation mechanisms due to its reliance on a trial-and-error search strategy for locating optimal solutions, making it the least effective approach for obtaining optimal synaptic weights among all the methods considered.

6.1.2. The number of unduplicated optimal synaptic weights

In this section, the number of unduplicated optimal synaptic weights (N_l^u) is used to compare the ability of methods to satisfy the objective function $f_2(X)$ in Eq (4.1). A higher N_l^u indicates superior performance in generating unduplicated optimal synaptic weights. As the value of n increases, locating unduplicated optimal synaptic weights becomes increasingly challenging. Therefore, in this paper, the maximum value of the N_l^u value is set to 10, corresponding to the range of n ($10 \leq n \leq 100$). Figure 5 presents the comparison of N_l^u in Cases 1, 3, and 5 for $w = 0.1$, 0.5, and 0.9.

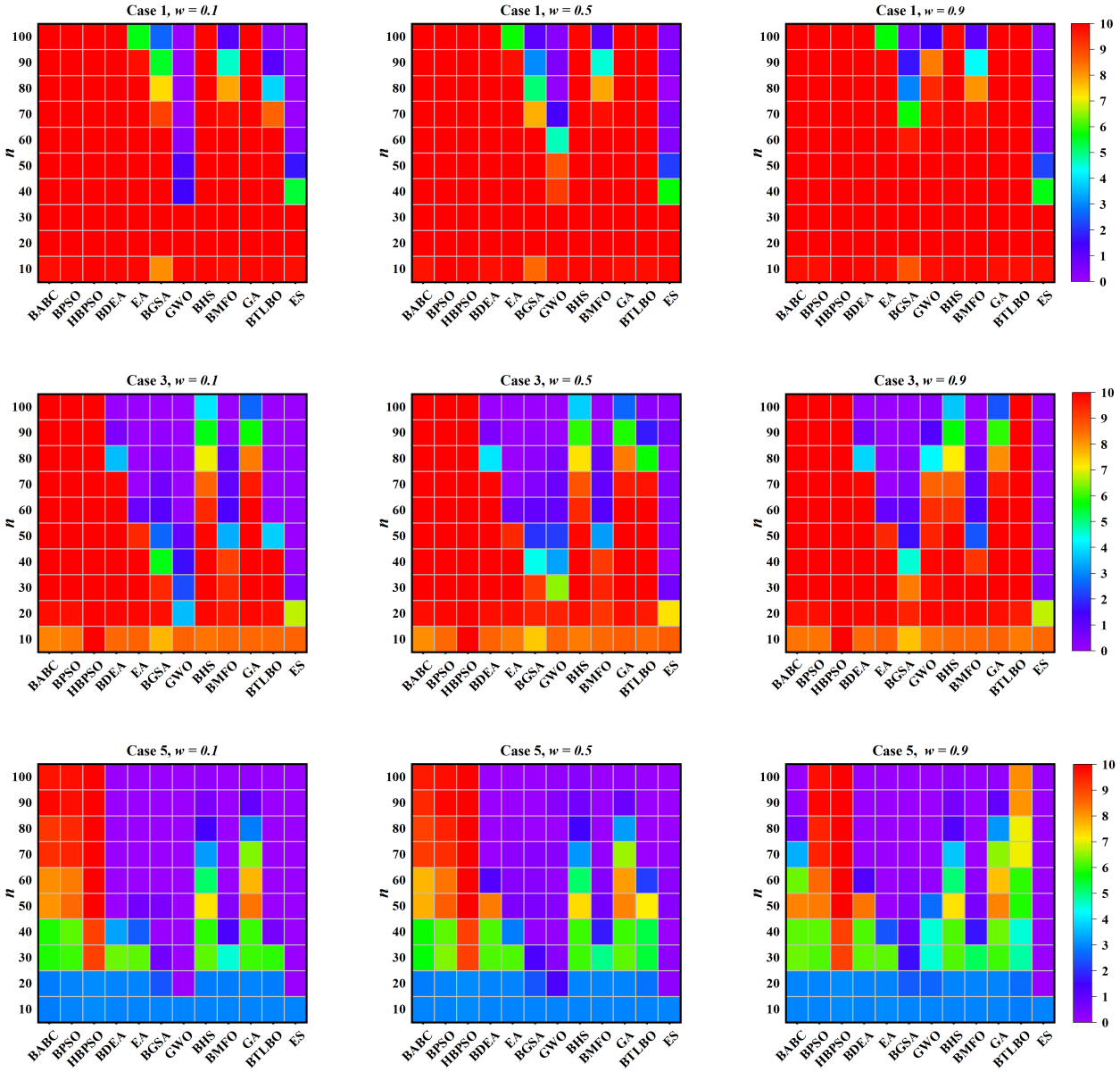


Figure 5. Heatmap of the comparisons for the number of unduplicated optimal synaptic weights (N_l^u) in Cases 1, 3, and 5 for $w = 0.1$, 0.5, and 0.9. The horizontal axis represents different algorithms, and a shared legend is placed on the rightmost side of each row to improve readability.

In Case 1, BABC, BPSO, and HBPSO consistently achieve N_l^u values above 9.67. Specifically, for $w = 0.1$, BPSO, HBPSO, BDEA, BHS, and GA generate an average of more than 9.8 unduplicated optimal synaptic weights. For $w = 0.3$, BABC, BPSO, HBPSO, and BDEA exceed 9.8, while for $w = 0.5$, only BPSO, HBPSO, and GA surpass this threshold. At $w = 0.9$, BABC, BPSO, HBPSO, and GA again produce more than 9.8 unduplicated weights. In Case 3, BABC, BPSO, and HBPSO consistently yield N_l^u values above 8.14 across all w and n. For $w = 0.1$, BABC, BPSO, HBPSO, BDEA, BHS, BMFO, and GA maintain $N_l^u > 0$ for all n, whereas EA, GWO, BTLBO, and ES drop to zero. At $w = 0.5$, EA and GWO reach zero for $n = 100$, and at $w = 0.9$, EA and ES produce zero values. In Case 5, HBPSO demonstrates the best performance with an average N_l^u of 8.4. However, the minimum N_l^u for BPSO and BABC falls to 2.86 when $n = 10$ and $n = 20$. For $w = 0.1$, BABC, BPSO, HBPSO, BHS, and GA consistently produce $N_l^u > 0$. At $w = 0.5$, GWO and BHS also yield nonzero values in addition to BABC, BPSO, HBPSO, BHS, and GA. For $w = 0.9$, BABC, BPSO, HBPSO, BHS, GA, and BTLBO maintain nonzero N_l^u values.

Since Case 5 is the most challenging in achieving the mazimum value of N_l^u, Table 7 reports the results of the Friedman test for these metrics. The corresponding p-value indicates that the performance differences among the models are statistically significant. A CD diagram is used to visualize the results of the Nemenyi post-hoc test. Figure 6 shows the CD diagram result of the number of unduplicated optimal synaptic weights (N_l^u) for Case 5 with $w = 0.1$, 0.5, and 0.9. The Nemenyi post-hoc comparison further reveals that HBPSO exhibits substantial performance gaps compared with most competing algorithms, as many rank differences exceed the critical difference. This indicates that the superiority of HBPSO is statistically significant rather than marginal. In particular, the performance gap between HBPSO and algorithms such as ES, BGSA, and GWO is especially pronounced across all weight configurations. Consequently, it can be concluded that the proposed HBPSO effectively maximizes the objective function $f_2(X)$ for all tested values of w and n.

HBPSO consistently achieves the highest N_l^u values across all considered values of w and n, with BPSO and BABC ranking second and third, respectively. Performance varies notably across cases. Case 1, characterized by a low proportion of first-order clauses, enables most population-based metaheuristics to attain relatively high N_l^u values due to their inherent exploration capabilities and population diversity maintenance [37]. Conversely, Case 5, exhibiting the highest proportion of first-order clauses, poses the greatest challenge. In this scenario, only BABC, BPSO, and HBPSO demonstrate superior performance. The advantage of BABC arises from the use of the "NAND" operator, which promotes population diversity and facilitates higher N_l^u values [26], whereas PSO benefits from a sigmoid function that normalizes particle velocity, enhancing convergence and solution quality [21].

Table 7. Friedman test results for number of unduplicated optimal synaptic weights (N_l^u) in Case 5.

r	χ^2	df	p-value	Final hypothesis
0.1	68.125	11	2.77×10^{-10}	Reject H_0
0.5	78.791	11	2.52×10^{-12}	Reject H_0
0.9	72.239	11	4.57×10^{-11}	Reject H_0

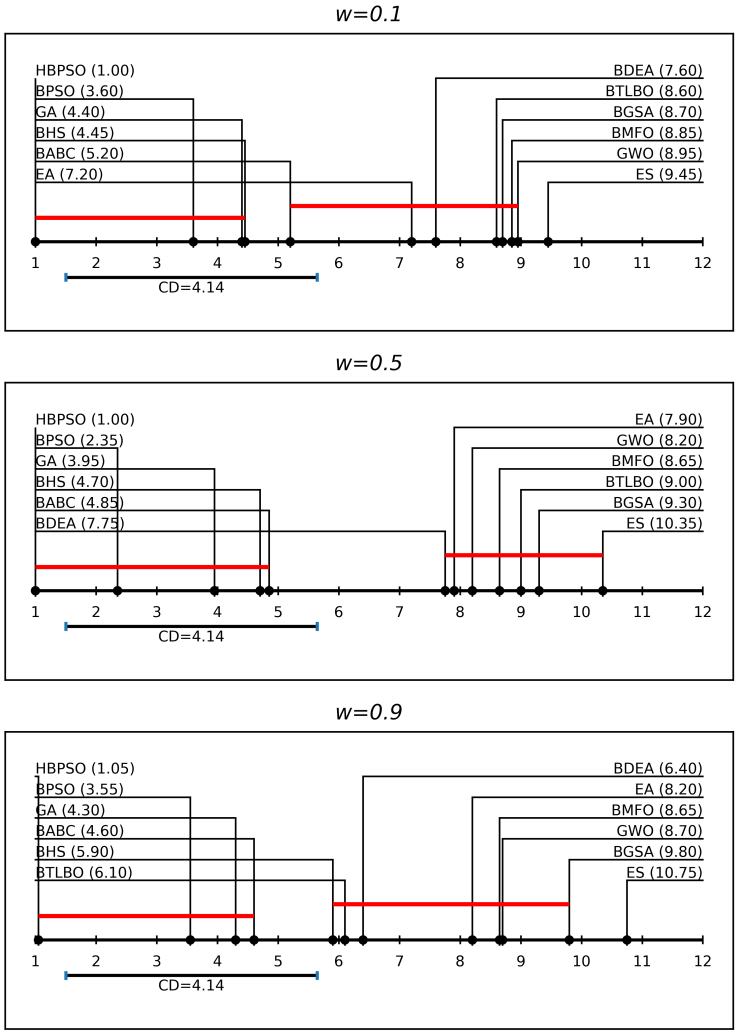


Figure 6. The CD diagram number of unduplicated optimal synaptic weights (N_l^u) for Case 5 with $w = 0.1$, 0.5, and 0.9. The red lines indicate groups of algorithms that are not statistically significantly different according to the Nemenyi post-hoc test.

6.1.3. The ratio of diversity

To evaluate the capability of different methods in maximizing the objective function $f_3(X)$ in Eq (4.1), we employ the diversity ratio metric (R_l^u). The R_l^u value is determined by three criteria: (a) achieving the maximum fitness value, (b) generating multiple optimal synaptic weights, and (c) ensuring that all optimal synaptic weights are unduplicated. Therefore, a higher R_l^s value reflects greater efficiency in producing unduplicated optimal synaptic weights. If the maximum fitness value cannot be achieved, multiple optimal synaptic weights cannot be obtained either, resulting in $N_l^u = 0$ and $R_l^u = 0$. Conversely, $R_l^u = 1$ indicates that all solutions produced by the algorithm are distinct. Accordingly, the maximum achievable value of R_l^u in this study is 1.

Figure 7 illustrates a comparison of R_l^u in Cases 1, 3, and 5 for $w = 0.1$, 0.5, and 0.9. In Case 1, BPSO, HBPSO, and EA consistently achieve R_l^u values above 0.98. Additional algorithms exceed this threshold only at specific w values: BDEA, GA, and BTLBO at $w = 0.1$; BABC, BDEA, and BMFO

at $w = 0.3$; BHS, BMFO, and GA at $w = 0.5$; and BABC, GWO, BHS, and GA at $w = 0.9$. In Case 3, characterized by moderate complexity, BPSO, HBPSO, BDEA, BMFO, and GA consistently achieve R_l^u above 0.84, while BTLBO reaches this threshold only at $w = 0.5$. Notably, BTLBO and ES fail to maintain R_l^u above 0.84 at $w = 0.7$, highlighting the sensitivity of less robust algorithms to instance parameters. In the most challenging Case 5, only BABC, BPSO, HBPSO, BHS, and GA consistently exceed $R_l^u = 0.29$, with GWO additionally surpassing this threshold at $w = 0.5$. For $w = 0.9$, BABC, BPSO, HBPSO, BHS, GA, and BTLBO exceed 0.29 of R_l^u values.

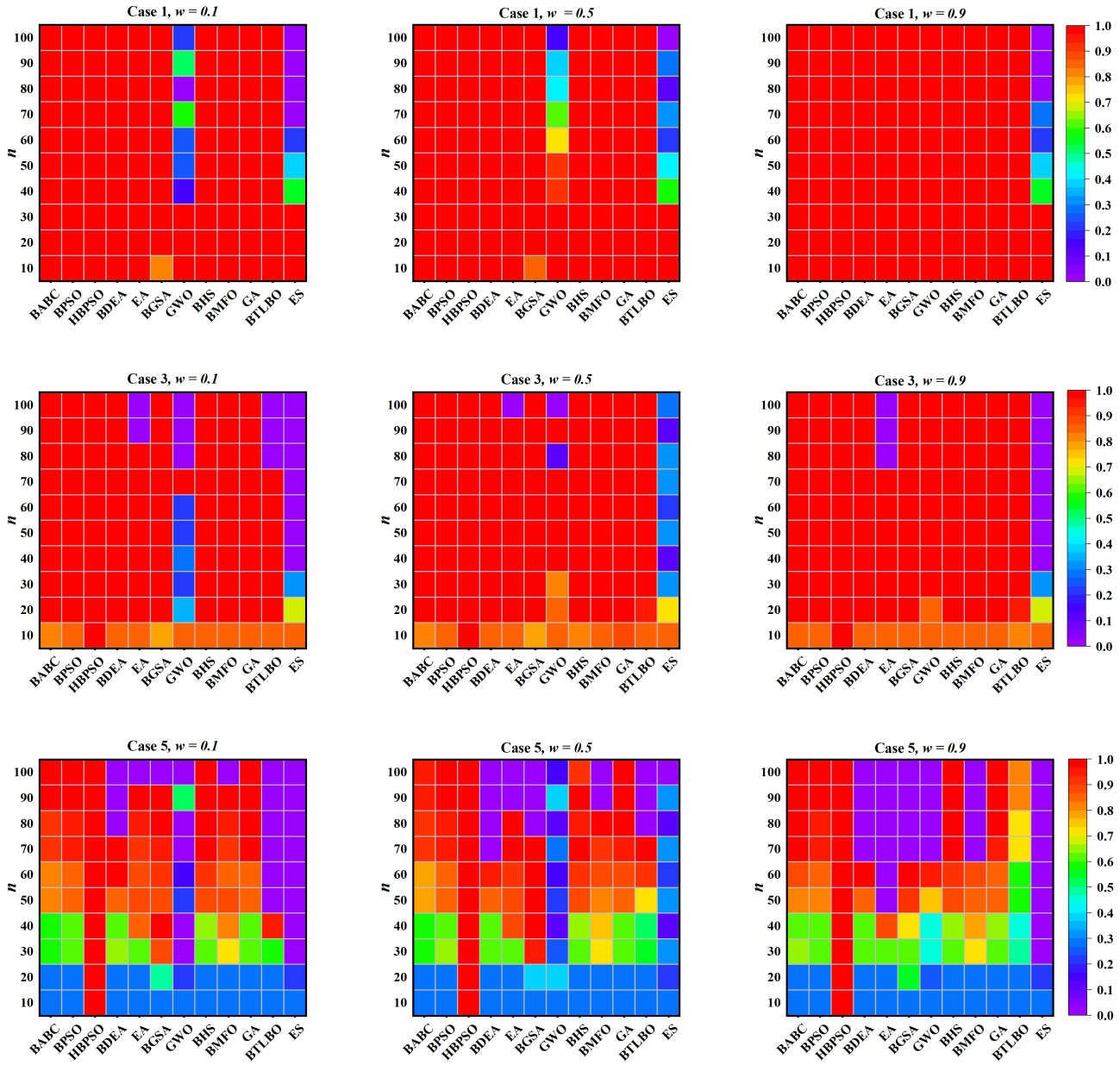


Figure 7. The ratio of diversity (R_l^u) in Cases 1, 3, and 5 for $w = 0.1$, 0.5, and 0.9. The horizontal axis represents different algorithms, and a shared legend is placed on the rightmost side of each row to improve readability.

Table 8 shows the Friedman test results for the metrics of R_l^u. Based on the obtained p-values, the performance of these models in maximizing R_l^u are statistically significant. Figure 8 shows the CD diagram result of the ratio of diversity (R_l^u) for Case 5 with $w = 0.1$, 0.5, and 0.9.

Table 8. Friedman test results for the ratio of diversity (R_l^u) in Case 5.

r	χ^2	df	p-value	Final hypothesis
0.1	56.123	11	4.82×10^{-8}	Reject H_0
0.5	55.483	11	6.32×10^{-8}	Reject H_0
0.9	54.848	11	8.26×10^{-8}	Reject H_0

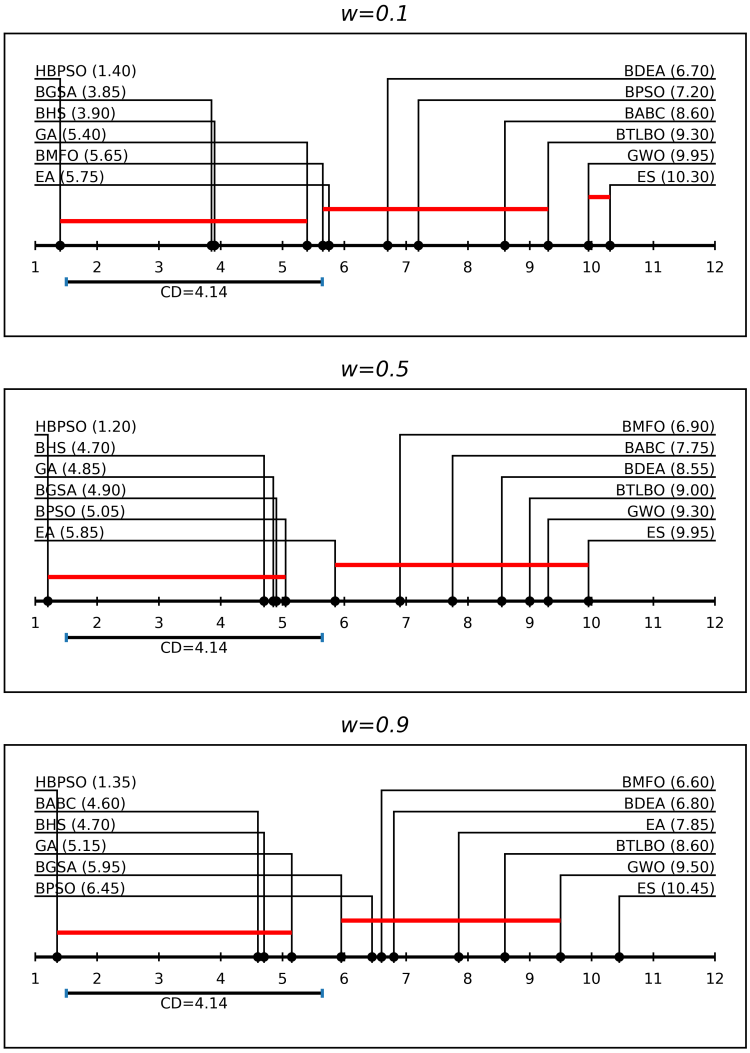


Figure 8. The CD diagram result of the ratio of diversity (R_l^u) for Case 5 with $w = 0.1$, 0.5, and 0.9. The red lines indicate groups of algorithms that are not statistically significantly different according to the Nemenyi post-hoc test.

Based on Figure 8, HBPSO consistently achieves the best performance across all cases. This demonstrates its strong stability and robustness under different parameter settings. In contrast,

traditional and heuristic algorithms such as ES, GWO, and BTLBO consistently rank at the bottom, indicating inferior performance. Other methods, including BPSO, GA, BABC, BGSA, and EA, show moderate performance but remain significantly worse than HBPSO. Overall, as n increases from 10 to 100, HBPSO consistently achieves 100% unduplicated optimal synaptic weights. This highlights its robustness and stability in consistently attaining the maximum R_l^u values for the objective function $f_3(X)$.

6.2. Second-stage optimization performance for fixed-parameter HBPSO

6.2.1. Total number of unduplicated final global minimum energies

The ability to generate unduplicated global minimum energies in the second-stage optimization is influenced by the total number of literals n and different models. For a better performance comparison, the maximum number of unduplicated global minimum energies is set to 20 in this experiment. Additionally, the program is executed 100 times for each compared method, which will result in a total of 2000 unduplicated global minimum energies. Figure 9 illustrates the number of unduplicated final global minimum energies in the the second-stage optimization for Cases 1, 3, and 5.

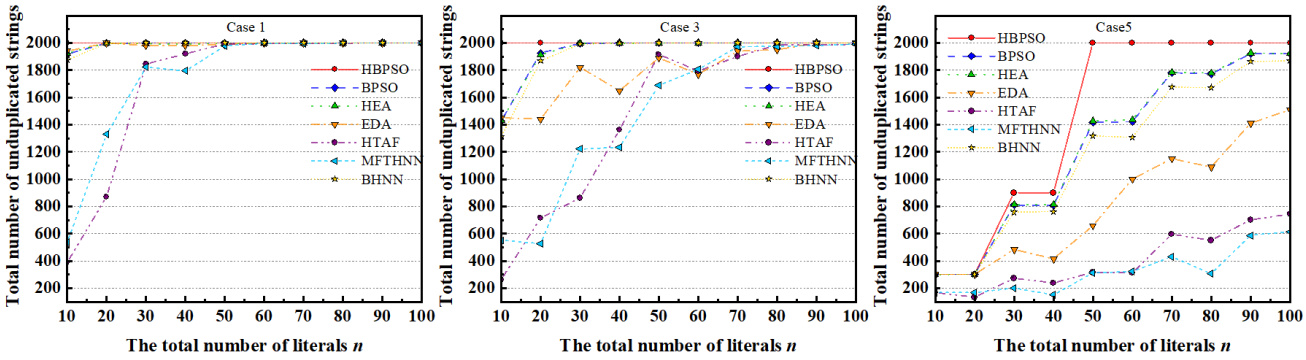


Figure 9. The total number of unduplicated final global minimum energies (N_r^u) values for Cases 1, 3, and 5 in the second-stage optimization.

Based on Figure 9, Case 1 is the easiest scenario for obtaining the global minimum energies, owing to its highest proportion of second-order clauses. Within this case, the metaheuristics HBPSO, BPSO, HEA, and EDA achieve average N_r^u values of 2000.00, 1991.74, 1990.52, and 1989.00, respectively. In comparison, HTAF, MFTHNN, and BHNN produce lower average N_r^u values of 1700.89, 1745.18, and 1987.13, respectively. The N_r^u values generally decrease as the proportion of second-order clauses decrease, making Case 5 the most challenging scenario for achieving high N_r^u values. In Case 5, HBPSO, BPSO, HEA, and EDA attain average N_r^u values of 1440.00, 1244.61, 1249.79, and 831.37, respectively, while HTAF, MFTHNN, and BHNN yield values of 402.45, 325.18, and 1182.23, respectively.

Table 9 presents the Friedman test results for the N_r^u metrics at a 95% confidence level. Based on the p-values reported in Table 9, the ability of HBPSO, BPSO, HEA, EDA, BHNN, MFTHNN, and HTAF to generate unduplicated minimum solutions is statistically significant. Figure 10 shows the CD diagram result of the total number of unduplicated final global minimum energies (N_r^u) for Cases 1, 3, and 5.

Table 9. Friedman test results of total number of unduplicated final global minimum energies (N_r^u) for Cases 1, 3, and 5.

Case	χ^2	df	p-value	Final Hypothesis
Case1	32.636	6	1.20×10^{-5}	Reject H_0
Case3	43.996	6	7.40×10^{-8}	Reject H_0
Case5	54.639	6	5.48×10^{-10}	Reject H_0

Based on Figure 10, the critical difference is set to CD=2.26 for the Nemenyi post-hoc test. From the average ranks, HBPSO consistently achieves the best performance under all configurations. In contrast, ES, GWO, BMFO, and BTLBO consistently rank at the bottom, indicating significantly weaker performance. Other methods, including BPSO, GA, BABC, BGSA, EA, and BDEA, show intermediate performance levels but remain inferior to HBPSO.

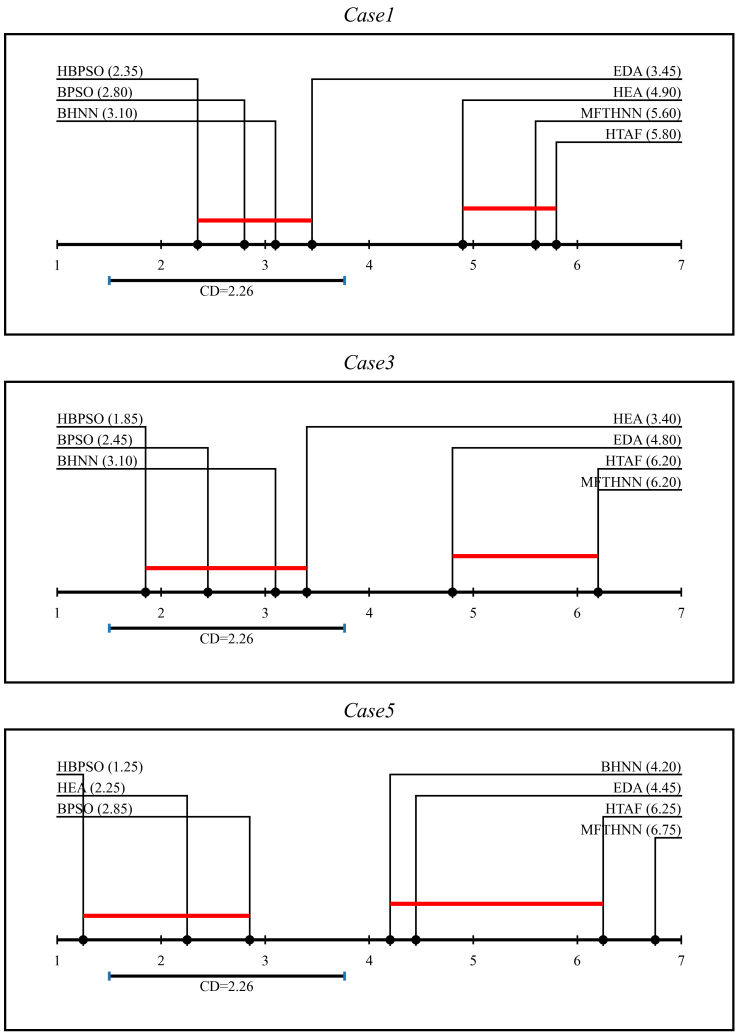


Figure 10. The CD diagram of the total number of unduplicated final global minimum energies (N_r^u) for Cases 1, 3, and 5. The red lines indicate groups of algorithms that are not statistically significantly different according to the Nemenyi post-hoc test.

The results show that the HBPSO consistently achieves the highest N_r^u values, owing to its enhanced exploration and exploitation via the sigmoid function, mutation strategy, greedy search, and duplication-checking mechanism. BPSO ranks second, benefiting from its balance of exploration and exploitation and the diversity induced by its sigmoid function [17]. Other methods struggle to attain high N_r^u values due to insufficient mechanisms for generating global minimum energies. Although EDA's perturbation strategy reduces neuron oscillation [8], its sampling focuses on similar individuals, limiting solution diversity [11]. HEA improves solution diversity through a "Caretaker" operator [6], but is less effective for WCRAN2SAT where higher-order clauses simplify fitness maximization. BHNN and MFTHNN fail to reach the optimal energy levels required [8], while HTAF performs worst due to repetitive solutions from the hyperbolic tangent activation [4]. Overall, the results show that HBPSO attains an average of over 1440 unduplicated final global minimum energies, demonstrating its robustness and stability through an effective balance between exploration and exploitation.

6.2.2. The ratio of unduplicated final global minimum energies

In this section, the ratio of unduplicated final global minimum energies R_r^u is used to evaluate the efficiency of the comparison methods. An R_r^u value of 1 indicates that all final global minimum energies from $P_{wcran2sat}$ are unique, so higher values reflect greater effectiveness in generating solutions. Figure 11 illustrates the R_r^u values for Cases 1, 3, and 5.

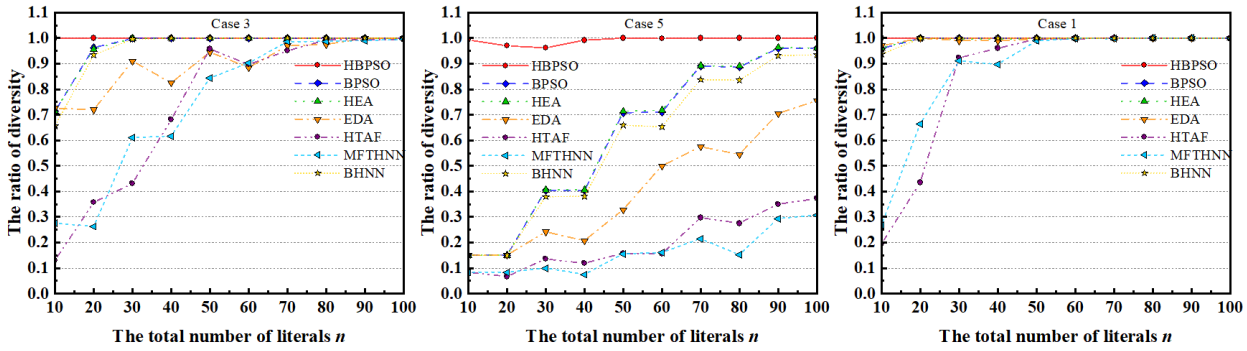


Figure 11. The ratio of unduplicated final global minimum energies (R_r^u) values for Cases 1, 3, and 5 in the second-stage optimization.

Based on Figure 11, Case 1 is the easiest scenario for obtaining unduplicated global minimum energies due to its high proportion of second-order clauses. Accordingly, HBPSO, BPSO, HEA, and EDA achieve average R_r^u values of 1.00, 1.00, 1.00, and 0.99, respectively. In contrast, HTAF, MFTHNN, and BHNN obtain 0.85, 0.87, and 0.99, respectively. Case 5 is therefore the most challenging, where BPSO, HEA, and EDA achieve values slightly above 0.15, while HTAF records only 0.08.

The Friedman test results for the ratio of the unduplicated final global minimum energies (R_r^u) for Cases 1, 3, and 5 are listed in Table 10. Based on the p-value obtained from this table, the capability to generate the maximum value of R_r^u is statistically significant for all the models. The CD diagram result of the ratio of unduplicated final global minimum energies (R_r^u) values for Cases 1, 3, and 5 are illustrated in Figure 12.

Table 10. Friedman test results of the ratio of unduplicated final global minimum energies (R_r^u) for Cases 1, 3, and 5.

Case	χ^2	df	p-value	Final hypothesis
1	22.353	6	1.04×10^{-3}	Reject H_0
3	44.523	6	5.82×10^{-8}	Reject H_0
5	56.231	6	2.61×10^{-10}	Reject H_0

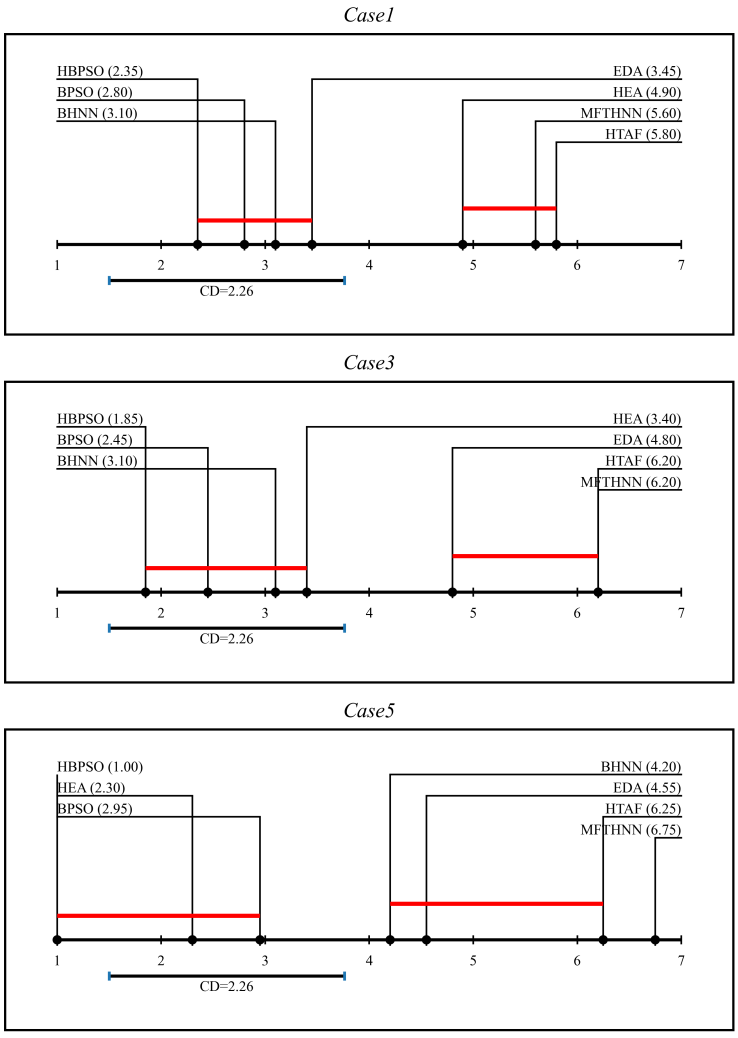


Figure 12. The CD diagram of the ratio of unduplicated final global minimum energies (R_r^u) values for Cases 1, 3, and 5. The red lines indicate groups of algorithms that are not statistically significantly different according to the Nemenyi post-hoc test.

Based on Figure 12, in terms of average ranks, HBPSO consistently achieves the best or near-best performance across all cases. In Case 5, the superiority of HBPSO becomes more pronounced. The rank differences between HBPSO and all other algorithms, including HEA, BPSO, BHNN, EDA, HTAF, and MFTHNN, are consistently larger than $CD = 2.26$. Overall, the results demonstrate that HBPSO consistently outperforms all competing algorithms across cases, with its advantage becoming

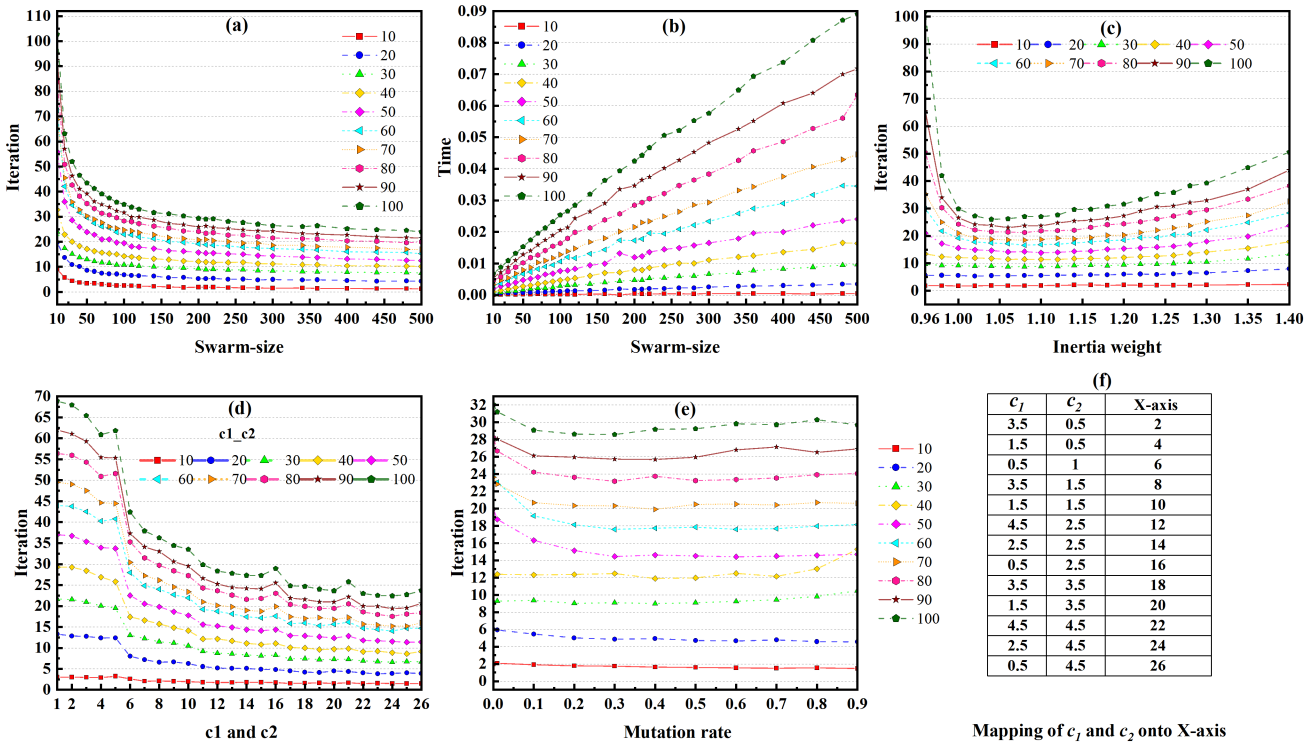
increasingly significant as problem complexity increases. This highlights the robustness, stability, and statistical superiority of the proposed method.

6.3. Sensitivity analysis for fixed-parameter HBPSO

Several key parameters critically influence the performance of fixed-parameter HBPSO, where even minor adjustments can significantly affect the effectiveness of the model. Therefore, it is essential to conduct a sensitivity analysis to systematically evaluate their impact. In this study, such an analysis is performed to examine how variations in these parameters affect the model's ability to optimize the multi-objective function defined in Eq (4.1).

Our primary objective is to identify the most influential parameter and determine its optimal configuration. Specifically, performance is evaluated in terms of convergence capability, measured by the number of iterations required to reach the maximum value of the multi-objective function defined in Eq (4.1). Notably, Case 5 presents the greatest challenge in attaining this maximum because it contains the highest proportion of first-order clauses, thereby increasing the complexity of the search landscape. To comprehensively examine the effects of parameter variations under extreme conditions and across problem scales, experiments are conducted for Case 5 within the range $10 \leq n \leq 100$. The experimental design follows the methodology proposed by Hamby [38], in which a single parameter is systematically varied, while all other parameters remain fixed. The resulting algorithmic performance is then analyzed to isolate and quantify the influence of each parameter.

Figure 13 illustrates the impact of parameter variations on the convergence capability of the fixed-parameter HBPSO in Case 5, providing a comparison of how individual parameter adjustments affect optimization efficiency across problem sizes.



c_1	c_2	X-axis
3.5	0.5	2
1.5	0.5	4
0.5	1	6
3.5	1.5	8
1.5	1.5	10
4.5	2.5	12
2.5	2.5	14
0.5	2.5	16
3.5	3.5	18
1.5	3.5	20
4.5	4.5	22
2.5	4.5	24
0.5	4.5	26

Figure 13. The impact of parameters on the convergence capability of Case 5.

Figure 13(a) shows the effect of swarm size (N_{swarm}) on convergence capability, and Figure 13(b) shows its effect on computational cost. Increasing the swarm size slows convergence but improves diversity by providing more candidate solutions, raising the likelihood of reaching the maximum of the multi-objective function (4.1). However, this comes at the cost of longer execution times. Beyond $N_{swarm} = 200$, iterations decrease slowly while execution time grows linearly. Therefore, N_{swarm} is set to 200 in all experiments to balance convergence speed and computational cost.

Figure 13(c) illustrates the influence of inertia weight (w_{iner}) on convergence. The optimal range is $[1.04, 1.06]$, which appears to be particularly suitable for guiding the search toward the maximum value of the multi-objective function. This range likely maintains sufficient exploration capability while preserving convergence stability, thereby achieving an effective balance between global search and local refinement. As observed by Gad [39], an excessively high inertia weight can cause velocity to be overly influenced by previous steps, leading to large oscillations. However, this fluctuation enables HBPSO to explore a broader region of the search space more efficiently. Consequently, it mitigates the risk of premature convergence and improves the probability of reaching the maximum value of the multi-objective function.

Figure 13(d) examines the acceleration coefficients c_1 and c_2. To ensure positivity, c_1 decreases from 4.5 to 0.5 as c_2 increases from 0.5 to 4.5. The corresponding mappings of c_1 and c_2 onto the X-axis are presented in Figure 13(f). In PSO, c_1 is primarily associated with exploration, whereas c_2 governs exploitation [23]. As shown in Figure 13(d), the optimal configuration occurs when $c_1 < c_2$, particularly with $c_2 = 4.5$. This setting emphasizes exploitation over exploration, enabling HBPSO to refine candidate solutions more effectively and converge more rapidly toward the maximum value of the multi-objective function.

Figure 13(e) presents the effect of mutation rate, with an optimal range of $[0.2, 0.4]$. This is because the mutation and greedy search mechanisms in HBPSO ensure a balanced trade-off between exploration and exploitation. Specifically, this mutation rate range helps prevent stagnation while preserving the effectiveness of the greedy search process. For large-scale problems, it enhances performance by introducing sufficient solution diversity without making the search excessively random.

Sensitivity analysis demonstrates that the fixed-parameter HBPSO maintains strong convergence performance across a wide range of parameter settings. This robustness can be primarily attributed to the synergistic effects of the mutation operator and the greedy search mechanism. Specifically, the mutation operator enhances global exploration, enabling HBPSO to effectively traverse the search space even when parameters such as inertia weight or swarm size are not ideally configured. In parallel, the greedy search strategy strengthens local exploitation by refining candidate solutions, thereby compensating for uneven initial particle distributions and accelerating convergence. Nevertheless, convergence remains sensitive to swarm size, c_1, c_2, w_{iner}, and mutation rate. Consequently, selecting appropriate parameter values is essential for improving algorithmic performance and for guiding the development of adaptive-parameter strategies.

6.4. Robust analysis in dynamic and noisy environments for adaptive-parameter HBPSO

6.4.1. The parameters of adaptive-parameter HBPSO

Based on the sensitivity analysis, the optimal parameter ranges are identified as follows: $c_1 < c_2$ with $c_2 = 4.5$, an inertia weight within $[1.04, 1.06]$, and a mutation rate typically in the range $[0.2, 0.4]$. The swarm size is set to 200 to achieve a balanced trade-off between computational cost and convergence speed. Additionally, to facilitate rapid bit flipping and enhance exploration, velocity v_{id} is constrained within [-5, +5] by setting $v_{\max} = 5$. These findings provide practical guidelines for dynamic parameters turing and establish a solid foundation for the adaptive-parameter mechanisms developed in the remainder of this paper. The parameter settings for the adaptive HBPSO are summarized in Table 11.

Table 11. Parameter settings of adaptive-parameter HBPSO.

Category	**Parameter**	**Value**
Adaptive inertia weight	w_{max}	1.40
	w_{min}	1.04
	α	5
	A	0.01
	B	5
Adaptive acceleration coefficients	$c_1^{\max}$	4.5
	$c_1^{\min}$	0.5
	$c_2^{\max}$	4.5
	$c_2^{\min}$	0.5
Adaptive mutation rate	mr_{inid}	0.4
	mr_{end}	0.2
	T_{max}	100
	T_{min}	10
	mr_{inie}	0.4199
	dr	−0.00485
	e	2.718

The adaptive inertia weight is oscillation-based strategy based on Eq (4.14). Specifically, $w_{\max} = 1.40$ and $w_{\min} = 1.04$ define a relatively narrow interval, which ensures that the inertia weight remains within a stable range while enabling adaptive variation during the search process. The decay rate parameter $\alpha = 5$ controls the exponential convergence speed of the inertia weight, enabling a gradual transition from exploration to exploitation. In addition, the oscillation components are governed by $A = 0.01$ and $B = 5$. The small amplitude A introduces only mild perturbations, preventing excessive fluctuations in the inertia weight and maintaining search stability. Moreover, frequency parameter B regulates the oscillatory behavior across iterations, enhancing diversity in the search dynamics without disrupting convergence. Overall, these parameter settings are designed to achieve a balanced trade-off between global exploration and local exploitation.

Furthermore, the update strategy of the acceleration coefficients c_1 and c_2 is designed to enhance global exploration in the early iterations and promote local exploitation in the later stages. Based

on Eq (4.15), the maximum values are set as $c_1^{max} = 4.5$ and $c_2^{max} = 4.5$, which are determined through sensitivity analysis. The minimum values are $c_1^{min} = 0.5$ and $c_2^{min} = 0.5$, defining the lower bounds of the parameter ranges. In particular, the gradual decrease of c_1 reduces the cognitive (self-learning) influence over iterations, while the corresponding adjustment of c_2 strengthens the social learning component. This complementary mechanism enables a smooth transition from exploration to exploitation during the optimization process.

There are two dynamic mutation rate strategies are introduced: One follows a linear decay, as described in Eq (4.16), while the other follows an exponential decay, as presented in Eq (4.17). For linear decay, $mr_{inid} = 0.4$ and $mr_{end} = 0.2$ are determined through sensitivity analysis. Additionally, $T_{max} = 100$ because the value of maximum iterations is 100 in Table 2. $T_{min} = 10$ is introduced to avoid an excessively small scaling factor in the early stage, thereby ensuring sufficient mutation activity and maintaining population diversity at the beginning of the search process.

For exponential decay, the parameter settings are not arbitrarily chosen but are determined based on the characteristics of the adaptive mutation mechanism and empirical analysis. Specifically, $mr_{\text{init}} = 0.4199$ defines the initial level of randomness at iteration $n = 0$, namely $mr(0) = 0.4199$. This relatively large value is chosen to ensure sufficient population diversity and strong global exploration ability at the early stage. If a much larger value were used, the search process would become overly stochastic and unstable, while a smaller value would reduce exploration capability. Moreover, parameter $dr = -0.00485$ controls the exponential decay speed. Since $dr < 0$, the mutation rate decreases monotonically with iterations. The magnitude of dr is carefully selected: A larger absolute value would cause rapid decay and lead to premature convergence, whereas a smaller absolute value would maintain excessive randomness and slow convergence. Furthermore, constant $e = 2.718$ is the base of the natural exponential function, which guarantees a smooth and continuous nonlinear decay profile. This ensures that the mutation rate transitions gradually from exploration-oriented behavior to exploitation-oriented behavior.

6.4.2. Ablation analysis for adaptive-parameter HBPSO

To investigate the influence of parameter scheduling strategies, we conduct a comprehensive ablation study over all combinations of dynamic control settings (D) and mutation schedules (M). Each configuration (D,M) represents a specific parameterization of the proposed HPSO-based DHNN framework, where D controls the adaptation strategy of PSO parameters c_1, c_2, and inertia weight, and M defines the mutation rate schedule. The definitions and corresponding labels of these control variables are provided in Table 12.

Table 12. Definition of control variables and corresponding labels.

Control variables	Label	Description
D (Dynamic)	0	Fixed c_1 and c_2, and inertia weight simultaneously
	1	Fixed inertia weight, dynamically adjust c_1 and c_2
	2	Fixed c_1 and c_2, dynamically adjust inertia weight
	3	Dynamically adjust c_1 and c_2, and inertia weight simultaneously
M (Method)	0	Fixed mutation rate
	1	Linear delayed mutation rate
	2	Exponential delayed mutation rate

Table 13 presents the ablation study results of different dynamic control (D) and mutation schedule (M) configurations across problem sizes n. All results are reported as mean ± standard deviation over 50 independent runs. Based on the experimental results, the influence of w on runtime and convergence behavior is observed to be negligible within the tested range. Therefore, w is treated as a non-sensitive parameter in the reported experiments and is omitted as a separate variable in Table 13.

Table 13. Ablation study of dynamic control (D) and mutation schedule (M) configurations.

D	M	**10**	**20**	**30**	**40**	**50**
0	0	1.79±0.12	5.86±0.31	9.26±0.42	12.52±0.55	**20.10**±0.88-
0	1	1.66±0.10	**4.82**±0.22*	9.15±0.40	12.50±0.53	14.65±0.61
0	2	1.65±0.09	4.93±0.21	**9.07**±0.38*	**12.21**±0.50*	**14.63**±0.60*
1	0	2.65±0.15	8.53±0.36	13.08±0.58	15.83±0.66	18.21±0.77
1	1	2.19±0.13	8.10±0.34	12.85±0.55	16.09±0.68	17.49±0.73
1	2	2.49±0.14	8.11±0.35	12.59±0.54	15.68±0.67	17.59±0.74
2	0	1.81±0.11	5.44±0.25	9.22±0.41	12.58±0.52	15.74±0.64
2	1	**1.53**±0.08*	4.87±0.20	10.09±0.43	14.27±0.60	17.46±0.73
2	2	1.57±0.09	5.10±0.23	9.95±0.42	14.41±0.61	17.50±0.74
3	0	**2.73**±0.16-	**8.72**±0.37-	13.01±0.56	15.85±0.66	18.15±0.76
3	1	2.21±0.13	8.32±0.35	13.98±0.59	17.16±0.72	19.15±0.80
3	2	2.25±0.13	8.42±0.36	**14.17**±0.60-	**17.25**±0.73-	19.24±0.81
Av.		2.04	6.77	11.37	14.70	17.49
D	M	**60**	**70**	**80**	**90**	**100**
0	0	**24.14**±1.02-	**25.75**±1.11-	**27.29**±1.18-	**28.90**±1.25-	**32.30**±1.41-
0	1	17.88±0.74	20.27±0.85	23.56±0.97	25.87±1.05	29.31±1.20
0	2	**17.68**±0.72*	20.36±0.84	23.56±0.98	26.30±1.08	29.13±1.21
1	0	21.17±0.89	22.10±0.95	24.22±1.03	26.03±1.10	28.07±1.18
1	1	20.01±0.86	21.79±0.92	23.88±1.01	25.67±1.08	27.60±1.15
1	2	20.12±0.85	21.41±0.91	23.82±1.00	25.62±1.07	27.72±1.16
2	0	18.66±0.76	**19.73**±0.82*	**22.30**±0.93*	**23.70**±0.99*	26.36±1.09
2	1	20.44±0.87	22.32±0.94	24.59±1.02	26.42±1.11	28.50±1.19
2	2	20.63±0.88	22.45±0.95	24.77±1.03	26.60±1.12	28.60±1.20
3	0	20.33±0.86	20.98±0.90	22.69±0.96	23.92±1.01	**25.81**±1.08*
3	1	21.31±0.90	22.62±0.95	24.52±1.03	25.76±1.10	27.59±1.17
3	2	21.32±0.90	22.92±0.97	24.41±1.02	25.74±1.09	27.33±1.16
Av.		20.31	21.89	24.13	25.88	28.20

Note: Values are reported as mean ± standard deviation over multiple independent runs. The bolded value with "*" indicates the best performance, and the bolded value with "-" indicates the worst performance. D represents "dynamic", M stands for "method", and Av. denotes the average number of iterations.

As shown in Table 13, the average number of iterations required for convergence increases from 2.04 at n=10 to 28.20 at n=100, reflecting the expected growth in computational complexity as the problem size increases. The results show problem size (n) and iteration budget effects indicate how convergence behavior evolves with increasing search space size, and clarify that performance improvements are

not driven by iteration budget scaling. Instead, the observed convergence trends primarily reflect the intrinsic scalability of the proposed optimization mechanism.

Under this unified encoding scheme, the combination (D, M) fully specifies a distinct optimization configuration. For instance, $(D, M) = (0, 0)$ represents a fully static baseline without parameter adaptation, whereas $(D, M) = (2, 1)$ corresponds to a configuration with adaptive inertia weight and linear delayed mutation. From the experimental results, it can be observed that configurations with adaptive parameter control (higher values of D) generally achieve better convergence efficiency compared to fully static settings. This indicates that dynamic adjustment of PSO parameters enhances the balance between exploration and exploitation, thereby accelerating convergence. In addition, time-varying mutation strategies ($M = 1$ and $M = 2$) consistently improve convergence stability over the fixed mutation strategy ($M = 0$). This suggests that introducing controlled stochastic perturbation helps maintain population diversity during the search process, thereby enhancing exploration capability while avoiding premature convergence.

Furthermore, the interaction between D and M suggests that the best performance is achieved when adaptive parameter control is combined with time-varying mutation strategies. This demonstrates that the proposed performance improvements are not attributable to a single mechanism but arise from the coordinated effect of dynamic control and mutation scheduling within the DHNN optimization framework.

In summary, the average number of iterations required for convergence ranges from 2.04 for $n = 10$ to 28.20 for $n = 100$, reflecting the expected growth in computational complexity as the problem size expands. Even in the largest search space, the worst-case and best-case convergence behaviors remain within a relatively narrow range, with 32.3 iterations observed under fixed parameter settings and 25.81 iterations achieved under dynamically adjusted parameters, respectively.

These results demonstrate that fixed and adaptive configurations of HBPSO maintain robust convergence behavior across problem scales. Nevertheless, adaptive parameter control consistently yields improved convergence efficiency compared to fixed strategies, highlighting its effectiveness in guiding the search process.

This observation is also consistent with the No Free Lunch (NFL) theorem, which states that no single optimization strategy or parameter configuration can be universally optimal across all problem instances. Therefore, adaptive parameter tuning becomes particularly important in enhancing the performance of HBPSO for solving the WCRAN2SAT problem, as it enables the algorithm to better adjust to problem-specific characteristics and improve convergence efficiency.

6.4.3. Robustness analysis for noise

Many metaheuristic algorithms struggle or fail when solving problems significantly affected by noise, a common challenge in real-world scenarios [40]. In this paper, experiments are conducted in a simulated environment where noise intensity ranges from 0.1 to 1, following a Gaussian distribution. The same control variables listed in Table 12 are used to ensure consistency. The impact of different noise intensities on convergence performance for dynamic parameter settings is illustrated in Figure 14.

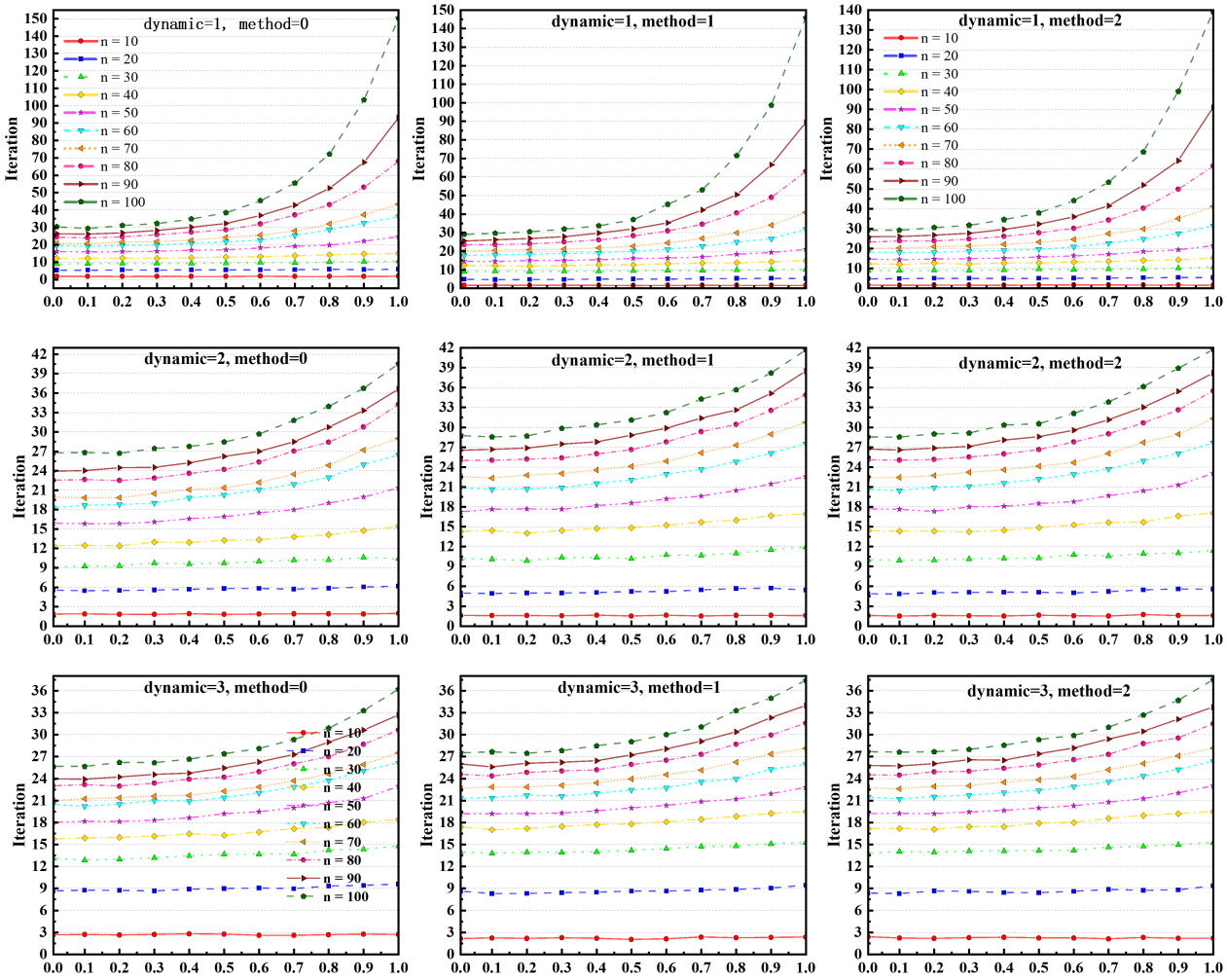


Figure 14. Convergence capability under different noise intensities for dynamic parameters.

Experimental results show that the dynamic parameter combinations (D, M) = (0, 0), (0, 1), and (0, 2) fail to reach the maximum of the multi-objective function under high noise, so they are excluded from Figure 14. For $n \leq 40$, noise has limited impact. However, as n increases, the influence of noise becomes increasingly significant. The injected noise amplifies exploratory behavior, causing the algorithm to search the space more randomly and making convergence more difficult. In small search spaces, the mutation operator and velocity transfer mitigate this effect, whereas in large spaces, adaptive-parameter HBPSO moves more erratically, reducing the likelihood of finding the function maximum.

The most effective parameter combinations for handling noise are (D, M) = (3, 0), (3, 1), and (3, 2), with (3, 0) being optimal. The combination of (3, 0) means dynamically adjusted acceleration coefficients along with inertia weight while maintaining a fixed mutation rate. For $n = 100$, increasing noise from 0.1 to 1.0 raises iterations by only 41.12% (25.65 → 36.2), whereas (1, 0) causes a 394.08% increase (30.42 → 150.31). It is worth noting that this is not the worst performance, as the parameter combinations (D, M) = (0, 0), (0, 1), and (0, 2) perform worse and are therefore omitted from Figure 14.

The results indicate the most effective strategy for mitigating noise is dynamically adjusting c_1, c_2, and the inertia weight, which balances exploration and exploitation and improves convergence. In

contrast, fixed parameters hinder the ability of the HBPSO to locate the multi-objective maximum, as noise-induced randomness cannot be properly managed. Since the search process in HBPSO is inherently dynamic, requiring strong exploration in the early stages and effective exploitation in the later stages, fixing these parameters disrupts this natural transition and leads to inefficient search behavior. Consequently, parameter settings with $D = 0$, 1, and 2 exhibit weaker noise-handling capability compared to $D = 3$.

Based on the experimental results, an additional observation is that mutation rate has a minor impact on noise resistance, with a fixed rate performing better than a dynamic one. While the mutation operator increases solution diversity and prevents premature convergence, a dynamic mutation rate adds extra randomness, making the search unstable and more sensitive to noise. A dynamic mutation rate further amplifies this randomness. When the mutation rate is high, solutions change too frequently, making the search process unstable and more sensitive to noise. Conversely, when the mutation rate is low, its influence becomes negligible, enabling noise to continue disrupting the search. Overall, dynamic mutation does not effectively help HBPSO overcome noise when locating the maximum value of the multi-objective function, as it introduces additional randomness that amplifies, rather than counteracts, the effects of noise.

7. Conclusions

In this paper,we propose a two-stage asymmetric HBPSO framework to optimize the weighted C-type random 2-satisfiability in DHNN. In the first-stage optimization, a fixed-parameter and adaptive-parameter multi-objective HBPSO improves the convergence to diversity of the unduplicated optimal synaptic weights. In the second stage, fixed-parameter HBPSO mitigates issues from repetitive neuron states. The results from the first-stage optimization demonstrate that fixed-parameter HBPSO maximizes the fitness value, achieving 100% satisfied clauses, and identifies over 8.4 unduplicated optimal synaptic weights with a 100% unduplicated ratio. For adaptive-parameter HBPSO, the average number of iterations required for convergence ranges from 2.04 when $n = 10$ to 28.20 when $n = 100$, with the best-performing configuration in the largest search space requiring only 25.81 iterations. Noise robustness analysis further shows that when the noise intensity increases from 0.1 to 1.0, the number of iterations rises moderately from 25.65 to 36.2, corresponding to a 41.2% increase despite a tenfold amplification of noise. In the second-stage optimization, the model achieves an average of more than 1,440 unduplicated global minimum energies with an unduplicated ratio exceeding 0.99.

The main contribution of this study is the development of an asymmetric two-stage optimization framework based on HPSO to improve the performance of SAT-based discrete Hopfield neural network models. In the proposed framework, the first-stage optimization employs a transition from fixed-parameter to adaptive-parameter HPSO to facilitate global convergence of synaptic weights, while the second-stage optimization adopts fixed-parameter HPSO to maintain computational efficiency and reduce repetitive neuron states. Through this two-stage optimization mechanism, the proposed framework effectively enhances the diversity of stable states and improves the effective storage capacity of the network. The proposed approach therefore provides an efficient computational strategy for improving the learning and retrieval processes in DHNN-based neural-symbolic systems.

This study exhibits strong methodological contributions and practical significance. From a methodological perspective, it demonstrates that integrating adaptive and fixed-parameter optimization

mechanisms within a unified two-stage framework can effectively enhance convergence behavior while increasing solution diversity in SAT-driven neural models. From an application perspective, the proposed framework provides a robust and scalable approach for solving complex combinatorial optimization problems while facilitating interpretable knowledge extraction in data-driven engineering and intelligent decision-making systems. Through this two-stage optimization mechanism, the proposed framework further improves the effective storage capacity of the DHNN by enriching the diversity of stable states and enhancing the ability of the network to represent distinct memory structures.

As can be observed from the experimental results, there are two limitations in this work. First, the datasets used in the experiment are derived from simulations. Hence, the ability of the proposed model to process real-world datasets has not been examined. Second, while the proposed HBPSO is effective for WCRAN2SAT, its effectiveness for other models remains untested. To address these limitations, in future research, we will focus on applying the proposed model to solve real-world decision-making problems, such as classification or prediction tasks. Additionally, a comparative study will be conducted between the proposed method and other logic mining approaches.

Author contributions

Yunjie Chang: Conceptualization, Methodology, Writing–original draft; Mohd Shareduwan Mohd Kasihmuddin: Validation; Yuan Gao: Writing–review & editing; Yueling Guo: Visualization; Nur Ezlin Zamri: Supervision; Xiaofeng Jiang: Software. 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 paper.

Acknowledgments

This research was supported by Universiti Sains Malaysia for Short Term Grant with Grant No. 304/PMATHS/6315655.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Y. Gao, M. S. M. Kasihmuddin, J. Chen, C. F. Zheng, N. A. Romli, M. A. Mansor, et al., Binary ant colony optimization algorithm in learning random satisfiability logic for discrete Hopfield neural network, *Appl. Soft Comput.*, **166** (2024), 112192. https://doi.org/10.1016/j.asoc.2024.112192

2. Y. L. 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
3. W. A. T. W. Abdullah, Logic programming on a neural network, *Int. J. Intell. Syst.*, **7** (1992), 513–519. https://doi.org/10.1002/int.4550070604
4. N. E. Zamri, S. A. Azhar, S. S. M. Sidik, M. A. Mansor, M. S. M. Kasihmuddin, S. P. A. Pakruddin, et al., Multi-discrete genetic algorithm in Hopfield neural network with weighted random k satisfiability, *Neural Comput. Appl.*, **34** (2022), 19283–19311. https://doi.org/10.1007/s00521-022-07541-6
5. 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
6. 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
7. S. Sathasivam, Boltzmann machine and new activation function comparison, *Appl. Math. Sci.*, **5** (2011), 3853–3860.
8. M. S. M. Kasihmuddin, M. A. Mansor, M. F. M. Basir, S. Sathasivam, Discrete mutation Hopfield neural network in propositional satisfiability, *Mathematics*, **7** (2019), 1133. https://doi.org/10.3390/MATH7111133
9. S. Sathasivam, S. A. Alzaeemi, M. Velavan, Mean-field theory in Hopfield neural network for doing 2 satisfiability logic programming, *Int. J. Modern Ed. Comput. Sci.*, **12** (2020), 27–39. https://doi.org/10.5815/ijmecs.2020.04.03
10. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893
11. Q. Yang, Y. Li, X. D. Gao, Y. Y. Ma, Z. Y. Lu, S. W. Jeon, et al., An adaptive covariance scaling estimation of distribution algorithm, *Mathematics*, **9** (2021), 3207. https://doi.org/10.3390/math9243207
12. H. Zhou, D. C. Ren, H. X. Xia, M. Y. Fan, X. Yang, H. Huang, AST-GNN: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction, *Neurocomputing*, **445** (2021), 298–308. https://doi.org/10.1016/j.neucom.2021.03.024
13. J. Chen, X. B. Li, Y. X. Zhao, Z. D. Ma, Z. M. Han, Y. X. Bao, TPCMEA: A tri-population evolutionary algorithm with adaptive stage-switching for complex CMOPs, *Appl. Soft Comput.*, **184** (2025), 113792. https://doi.org/10.1016/j.asoc.2025.113792
14. N. Heydaribeni, X. R. Zhan, R. S. Zhang, T. Eliassi-Rad, F. Koushanfar, Distributed constrained combinatorial optimization leveraging hypergraph neural networks, *Nat. Machine Intell.*, **6** (2024), 664–672. https://doi.org/10.1038/s42256-024-00833-7
15. J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, In: *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, **5** (1997), 4104–4108. https://doi.org/10.1109/ICSMC.1997.637339

16. S. Gupta, S. Gupta, Fitness and historical success information-assisted binary particle swarm optimization for feature selection, *Knowl. Based Syst.*, **306** (2024), 112699. https://doi.org/10.1016/j.knosys.2024.112699
17. T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, S. Mirjalili, Particle swarm optimization: a comprehensive survey, *IEEE Access*, **10** (2022), 10031–10061. https://doi.org/10.1109/ACCESS.2022.3142859
18. H. Moazen, S. Molaei, L. Farzinvash, M. Sabaei, PSO-ELPM: PSO with elite learning, enhanced parameter updating, and exponential mutation operator, *Inform. Sci.*, **628** (2023), 70–91. https://doi.org/10.1016/j.ins.2023.01.103
19. K. G. Reddy, D. Mishra, An effective initialization for fuzzy PSO with greedy forward selection in feature selection, *Int. J. Data Sci. Anal.*, **20** (2025), 4103–4126. https://doi.org/10.1007/s41060-024-00712-9
20. Y. J. Chang, M. S. M. Kasihmuddin, W. N. A. Ruzai, Y. L. Guo, J. Chen, Weighted C-type random 2 satisfiability in discrete Hopfield neural network, *Eng. Appl. Artif. Intell.*, **160** (2025), 111760. https://doi.org/10.1016/j.engappai.2025.111760
21. J. C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, *Appl. Math. Comput.*, **218** (2012), 11042–11061. https://doi.org/10.1016/j.amc.2012.05.001
22. J. H. Liu, Y. Mei, X. D. Li, An analysis of the inertia weight parameter for binary particle swarm optimization, *IEEE Trans. Evol. Comput.*, **20** (2016), 666–681. https://doi.org/10.1109/TEVC.2015.2503422
23. M. Isiet, M. Gadala, Sensitivity analysis of control parameters in particle swarm optimization, *J. Comput. Sci.*, **41** (2020), 101086. https://doi.org/10.1016/j.jocs.2020.101086
24. Y. J. Song, X. Cai, X. B. Zhou, B. Zhang, H. L. Chen, Y. G. Li, et al., Dynamic hybrid mechanism-based differential evolution algorithm and its application, *Expert Syst. Appl.*, **213** (2023), 118834. https://doi.org/10.1016/j.eswa.2022.118834
25. J. B. Liu, Y. Q. Jv, Z. W. Wang, Y. Zhang, H. J. Sun, H. Y. Zheng, Lexicographic optimization-based priority ascending strategy for feasibility judgment and soft constraint adjustment, *IEEE Trans. Autom. Sci. Eng.*, **22** (2025), 8828–8843. https://doi.org/10.1109/TASE.2024.3490620
26. S. S. M. Sidik, N. E. Zamri, M. S. M. Kasihmuddin, H. A. Wahab, Y. L. Guo, M. A. Mansor, Non-systematic weighted satisfiability in discrete Hopfield neural network using binary artificial bee colony optimization, *Mathematics*, **10** (2022), 1129. https://doi.org/10.3390/math10071129
27. A. P. Engelbrecht, G. Pampara, Binary differential evolution strategies, In: *2007 IEEE Congress on Evolutionary Computation*, Singapore, 2007, 1942–1947. https://doi.org/10.1109/CEC.2007.4424711
28. E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, BGSA: binary gravitational search algorithm, *Nat. Comput.*, **9** (2010), 727–745. https://doi.org/10.1007/s11047-009-9175-3
29. L. Wang, R. X. Yang, Y. Xu, Q. Niu, P. M. Pardalos, M. Fei, An improved adaptive binary harmony search algorithm, *Inform. Sci.*, **232** (2013), 58–87. https://doi.org/10.1016/J.INS.2012.12.043
30. I. Tumar, Y. Hassouneh, H. Turabieh, T. Thaher, Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction, *IEEE Access*, **8** (2020), 8041–8055. https://doi.org/10.1109/ACCESS.2020.2964321

31. T. T. Khuat, M. H. Le, Binary teaching-learning-based optimization algorithm with a new update mechanism for sample subset optimization in software defect prediction, *Soft Comput.*, **23** (2019), 9919–9935. https://doi.org/10.1007/s00500-018-3546-6

32. C. W. Huang, Y. X. Li, X. Yao, A survey of automatic parameter tuning methods for metaheuristics, *IEEE Trans. Evol. Comput.*, **24** (2020), 201–216. https://doi.org/10.1109/TEVC.2019.2921598

33. N. H. A. Rahman, A. F. Zobaa, Integrated mutation strategy with modified binary PSO algorithm for optimal PMUs placement, *IEEE Trans. Ind. Inform.*, **13** (2017), 3124–3133. https://doi.org/10.1109/TII.2017.2708724

34. X. Wang, D. S. K. Ting, P. Henshaw, Mutation particle swarm optimization (M-PSO) of a thermoelectric generator in a multi-variable space, *Energ. Convers. Manage.*, **224** (2020), 113387. https://doi.org/10.1016/j.enconman.2020.113387

35. P. S. Chen, Z. Y. Zeng, Developing two heuristic algorithms with metaheuristic algorithms to improve solutions of optimization problems with soft and hard constraints: an application to nurse rostering problems, *Appl. Soft Comput.*, **93** (2020), 106336. https://doi.org/10.1016/j.asoc.2020.106336

36. M. A. Khanesar, R. Bansal, G. Martínez-Arellano, D. T. Branson, XOR binary gravitational search algorithm with repository: Industry 4.0 applications, *Appl. Sci.*, **10** (2020), 6451. https://doi.org/10.3390/APP10186451

37. M. N. Omidvar, X. D. Li, X. Yao, A review of population-based metaheuristics for large-scale black-box global optimization–Part I, *IEEE Trans. Evol. Comput.*, **26** (2022), 802–822. https://doi.org/10.1109/TEVC.2021.3130838

38. D. M. Hamby, A review of techniques for parameter sensitivity analysis of environmental models, *Environ. Monit. Assess.*, **32** (1994), 135–154. https://doi.org/10.1007/BF00547132

39. A. G. Gad, Particle swarm optimization algorithm and its applications: a systematic review, *Arch. Comput. Methods Eng.*, **29** (2022), 2531–2561. https://doi.org/10.1007/s11831-021-09694-4

40. Z. H. Li, S. Zhang, X. Y. Cai, Q. F. Zhang, X. M. Zhu, Z. Fan, et al., Noisy optimization by evolution strategies with online population size learning, *IEEE Trans. Syst. Man Cybernet. Syst.*, **52** (2022), 5816–5828. https://doi.org/10.1109/TSMC.2021.3131482

AIMS Press