AIMS Mathematics, 11(5): 15074–15119.
DOI: 10.3934/math.2026621
Received: 25 August 2025
Revised: 03 January 2026
Accepted: 13 January 2026
Published: 29 May 2026

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

Research article

Multi-objective optimization via hybrid exhaustive search with intelligent mutation for major 2 satisfiability in Discrete Hopfield Neural Networks

Alyaa Alway[1], Mohd Shareduwan Mohd Kasihmuddin[2], Mohd. Asyraf Mansor[1,*], Nur Ezlin Zamri[3], Guo Yueling[4], Siti Zulaikha Mohd Jamaludin[2], Azleena Mohd Kassim[5]

[1] School of Distance Education, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia
[2] School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia
[3] Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia
[4] School of Science, Hunan Institute of Technology, Hengyang 421002, China
[5] School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia

* **Correspondence:** Email: asyrafman@usm.my; Tel: +6046533935.

Abstract: The diversification of retrieved final neuron states through non-systematic satisfiability logical representation is pivotal to ensuring the optimality and functionality of Discrete Hopfield Neural Networks (DHNN) under varying neuron complexities. However, DHNN learning frameworks are predominantly designed for single-objective optimization, which often leads to repetitive neuron state patterns, overfitting, and limited storage capacity, particularly under increasing neuron complexity. These limitations indicate the need for a learning mechanism that can simultaneously enhance solution optimality and diversity. Motivated by this gap, we proposed a multi-objective DHNN framework based on a non-systematic Major 2 Satisfiability (MAJ2SAT) logical representation integrated with a Hybrid Exhaustive Search (HES) learning algorithm enhanced by an intelligent mutation operator. The proposed framework jointly optimized neuron fitness and neuron state diversity, enabling the systematic generation of high-quality and diversified neuron states. Unlike conventional exhaustive or heuristic search methods, the intelligent mutation operator selectively modified neuron states associated with unsatisfied clauses, thereby improving exploration efficiency while preserving solution feasibility. To further enhance the learning capability of DHNN, the proposed model introduced the concept of power strings, which facilitated the construction of multiple content addressable memories and effectively expanded the storage capacity of the network. Extensive experiments conducted on simulated datasets demonstrated that the proposed approach consistently I

outperforms several state-of-the-art learning algorithms across problem sizes. The results showed reduced learning error in neuron diversity, increased total neuron variation, and a higher global minimum attainment ratio under varying clause configurations. Overall, the proposed multi-objective DHNN–MAJ2HES framework establishes a robust and scalable learning paradigm that enhances solution quality and diversity, with strong potential for extension to other logic-based neural optimization problems.

Keywords: major 2 satisfiability; Discrete Hopfield Neural Network; multi-objective optimization; HES; intelligent mutation; power strings
Mathematics Subject Classification: 68N17, 68R07, 68T27

1. Introduction

Artificial Neural Networks (ANN) are among the most powerful models used in optimization problems. The ANN model can learn data effectively due to its well-known memory capability. This includes auto associative and associative properties [1]. Among these models, Discrete Hopfield Neural Networks (DHNN) belong to the class of recurrent neural networks and have been widely applied in complex optimization problems and pattern retrieval tasks. The DHNN model introduced by Hopfield and Tank demonstrates strong memory performance even under noisy conditions [2]. This capability originates from its Content Addressable Memory (CAM), which encodes input and output patterns through synaptic weights. These synaptic weights are later applied during the testing phase to generate final neuron states as outputs. However, as the number of stored patterns increases, the network may retrieve suboptimal solutions. To address this issue, Abdullah introduced the Wan Abdullah method to compute synaptic weights. This method ensures stable convergence and supports the attainment of global minimum solutions [3]. Further enhancement was achieved by Sathasivam through the integration of Horn Satisfiability logic within the DHNN framework [4]. This approach represented input and output patterns using logical clauses. Experimental results showed that the global minima ratio decreases as the number of neurons increased. This observation suggests that improving the satisfiability representation is essential for enhancing DHNN performance.

Motivated by these findings, logic programming has become an important research direction within artificial intelligence. Boolean Satisfiability (SAT) logic is commonly expressed in conjunctive normal form and produces binary outcomes. SAT logic has been extensively embedded into DHNN models and is commonly referred to as DHNN-SAT. In this framework, each neuron corresponds to a variable or literal. Researchers have focused on Horn SAT before progressing toward systematic k Satisfiability. In systematic kSAT, each clause contains non repetitive literals. As reported in [5], systematic 2SAT was incorporated into DHNN and achieved a higher global minima ratio compared to Horn SAT. Building on this progress, Mansor *et al.* introduced systematic 3SAT to further improve solution quality [6]. Their findings showed that 3SAT outperforms 2SAT in terms of global minima ratio. These results highlight the importance of cost function minimization in DHNN optimization. Beyond systematic SAT, Kasihmuddin *et al.* introduced Maximum k Satisfiability (MAXkSAT) to incorporate non satisfiable clauses [7]. Their study demonstrated that MAX2SAT and MAX3SAT surpasses Kernel Hopfield Neural Networks in learning error and global minima ratio. Another development was the Discrete Mutation Hopfield Neural Network, which applied an estimation

distribution algorithm during the testing phase [8]. This approach expanded the solution space and increased neuron state diversity. These advancements indicate that optimization algorithms play a critical role in improving the DHNN learning phase.

In parallel with logic-based optimization studies, advances in Hopfield neural networks have entailed alternative research directions focusing on nonlinear dynamics and hardware-oriented implementations. In particular, memristive Hopfield neural networks [9] have been shown to exhibit rich dynamical behaviors, including periodic and chaotic bursting, multiscroll attractors, and practical realization on reconfigurable hardware platforms such as field-programmable gate arrays. These studies primarily emphasize continuous-time dynamics, circuit design, and application-driven demonstrations such as image encryption. While such developments highlight the versatility of Hopfield-based models, they do not address optimization-driven learning, satisfiability-based logical representation, or multi-objective performance evaluation in discrete Hopfield neural networks. Therefore, advancing logic-based DHNN frameworks remains necessary to improve learning efficiency, solution diversity, and storage capacity.

In solving complex optimization problems, heuristic and metaheuristic approaches are widely employed. Heuristic methods such as Exhaustive Search lack search operators and rely on trial-and-error mechanisms [10]. In contrast, metaheuristics include global and local search operators. These operators enhance exploration and exploitation capabilities. Within the DHNN context, metaheuristics are commonly applied to minimize the cost function during learning [11]. Early DHNN-SAT models relied heavily on Exhaustive Search. However, studies revealed that Exhaustive Search performs poorly as the number of neurons increases [5]. Genetic Algorithm was later introduced to address this limitation through mutation and crossover operators. These operators reduce learning complexity and improve performance. Election Algorithm was also implemented in non-systematic SAT based DHNN models [12]. This algorithm introduces exploration and exploitation strategies through socio political operators. Comparative studies using Random 2 Satisfiability showed that Election Algorithm outperformed Exhaustive Search and Genetic Algorithm in most learning metrics. These findings emphasize the importance of operator rich metaheuristics. Despite these efforts, the integration of non-systematic SAT with advanced metaheuristics in DHNN remains limited. This gap motivates further investigation.

In subsequent studies, researchers explored Weighted Random k Satisfiability (*r*2SAT) within the DHNN framework [13]. This model regulated the proportion of negative literals to improve navigation within the SAT space. Comparative evaluations demonstrated that metaheuristic assisted learning generates more diverse neuron states and improved global minima attainment. Karim *et al.* later introduced Random 3 Satisfiability (RAN3SAT) to increase logical complexity and further enhance performance [14]. This approach achieved higher global minima ratios compared to Random 2 Satisfiability (RAN2SAT). Election Algorithm was later combined with RAN3SAT to further optimize learning [15]. The results showed improved diversity and reduced learning error. Although these researchers achieved strong performance in single objective optimization, they did not focus on expanding the DHNN search space. This limitation restricts the model's ability to handle more complex optimization tasks.

Advances in artificial intelligence have intensified research on optimization problems across domains. These problems often involve objectives that must be optimized simultaneously. Evolutionary algorithms and metaheuristics have shown strong performance in multi objective optimization within large search spaces [16]. Studies have shown that modified metaheuristic strategies with enhanced exploration and exploitation mechanisms can significantly improve

convergence efficiency and solve diversity in complex optimization problems. For example, dynamic swarm-based metaheuristics have achieved notable improvements in convergence speed and population diversity when solving multi–objective problems with large decision spaces [17]. Similarly, nature inspired algorithms that emphasize structured and adaptive search behaviors have been shown to provide effective alternatives for solving complex and constrained optimization problems [18]. However, conventional evolutionary algorithms often suffer from high computational cost as the decision space expands. To overcome this challenge, modified operators are required to balance efficiency and performance. Karim *et al.* addressed this issue by integrating Random 3 Satisfiability into the Hybrid Election Algorithm [19]. Their approach introduced a caretaker party operator to improve search balance. This model achieved higher fitness values, greater neuron diversity, and increased storage capacity. Nevertheless, excessive algorithm modification can increase computational complexity, as highlighted by Luo *et al.* [20]. Therefore, efficient and minimal operator design remains crucial. Motivated by these insights, we extend the work of Alway *et al.* by integrating Major 2 Satisfiability (MAJ2SAT) into the DHNN framework for multi objective optimization [21].

It is important to emphasize that our objective of this study is to investigate the effectiveness of logic representation and learning optimization within the DHNN framework rather than application-specific data modeling. Major 2 Satisfiability is selected due to its majority-based logical structure, which introduces redundancy and robustness in clause representation, making it suitable for evaluating complex pattern storage and retrieval behavior in DHNN. The use of simulated datasets enables controlled and systematic assessment of learning performance, convergence behavior, and multi-objective optimization capability without the influence of external data variability. This abstraction enables fair comparison across metaheuristic strategies and provides clear insight into the contribution of MAJ2SAT and HES to DHNN optimization.

Therefore, the major contributions of this paper are summarized as follows:

1. Major 2 Satisfiability is hybridized into the discrete Hopfield neural network using second and third order conjunctive normal form logic. This formulation minimizes the cost function and produces optimal synaptic weights.
2. The mutation operator from the Genetic Algorithm is adapted and embedded into HES. This operator is referred to as intelligent mutation and enhances the DHNN learning phase.
3. We address multi objective optimization by maximizing fitness, diversity, and power string utilization. The proposed approach aims to reduce computational cost and improve storage capacity.
4. The proposed DHNN-MAJ2HES model is evaluated against five benchmark models using simulated datasets. Performance is assessed in learning and testing phases.

The remainder of this paper is organized as follows: In Section 2, we present our motivation of the study. In Sections 3 and 4, we describe Major 2 Satisfiability and the discrete Hopfield neural network. In Section 5, we explain the multi objective functions. In Section 6, we introduce the HES algorithm. In Section 7, we outline the experimental setup. In Section 8, we present the results and discussion. We conclude with a summary of findings. Figure 1 illustrates the overall framework of the proposed model.

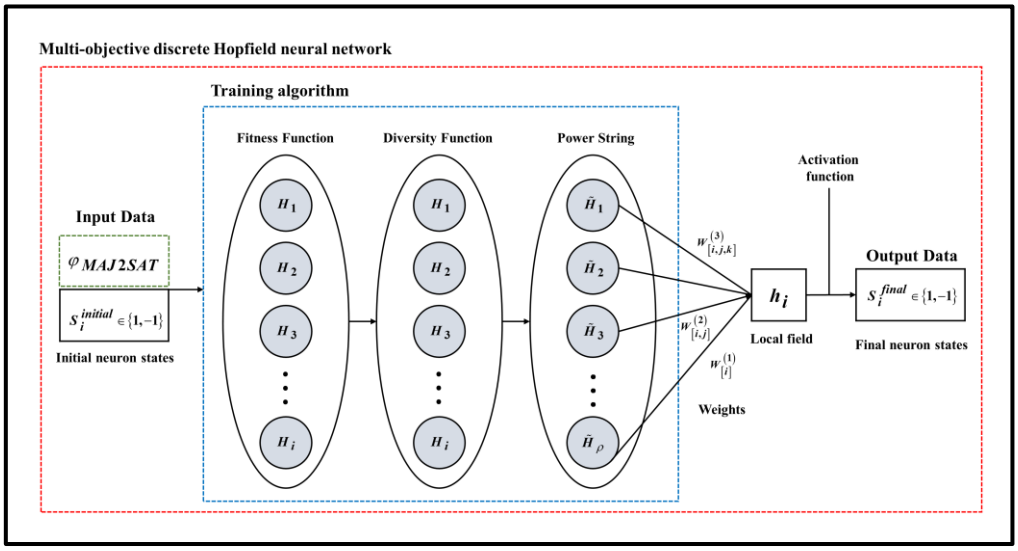

Figure 1. Diagrammatic view of the DHNN–MAJ2HES model.

2. Motivation

In this section, we highlight the key limitations of existing DHNN models and learning strategies, which motivate the development of the proposed MAJ2HES framework. Specifically, we focus on the rigidity of systematic SAT logic, excessive modification of metaheuristic algorithms, limited storage capacity in DHNN, and insufficient diversification of synaptic weights.

2.1. Inflexible systematic SAT logic

In systematic 2SAT logic, the number of literals per clause is restricted to two, constraining the logical representation to a single SAT structure. Studies have shown that DHNN performance is strongly influenced by the flexibility of the underlying SAT formulation [22–23]. Non-systematic kSAT logic provides a more adaptable structure by allowing variable clause lengths. For example, Karim *et al.* introduced higher-order RAN3SAT to improve DHNN performance [14]. In addition, Reuben *et al.* demonstrated that majority-based Boolean logic performs efficiently in modern integrated circuits [24], while Chattopadhyay *et al.* reported that majority elements enhance decision-making processes [25]. Motivated by these findings, we incorporate a majority term into a flexible higher-order logical structure to improve DHNN learning capability.

2.2. Excessive modifications in metaheuristic algorithms

Researchers have proposed numerous enhancements to existing metaheuristic algorithms to improve optimization performance. However, excessive modifications do not necessarily guarantee robust or feasible solutions. For instance, Song *et al.* proposed an adaptive artificial bee colony algorithm by introducing additional search equations to balance exploration and exploitation [26], yet their approach suffered from identical fitness values during the onlooker bee phase. In contrast, exhaustive search (ES) remains a simple yet reliable algorithm with guaranteed optimality, as demonstrated by Kammerer *et al.* [27]. Rather than extensively altering ES, we introduce a single

Genetic Algorithm-inspired mutation operator to enhance exploration while preserving the fundamental characteristics of ES.

2.3. Limited storage capacity of DHNN

The DHNN model employs Content Addressable Memory (CAM) to store neuron states, logical strings, and synaptic weights obtained during learning. The presence of unsatisfied interpretations can adversely affect the storage of optimal synaptic weights. Moreover, the storage capacity of CAM is limited to approximately 14% of the DHNN size [28]. Researchers [22,29] have focused on single-objective optimization, restricting DHNN to a single CAM and limiting solution space exploration. Researchers integrating SAT logic into DHNN has enabled multi-objective learning with multiple CAMs [19]. Building on these advances, we propose a hybrid metaheuristic framework to construct a multi-objective DHNN with multiple CAMs, thereby expanding storage capacity and solution diversity.

2.4. Promoting diversification of synaptic weights

Neuron state generation during learning directly influences the synaptic weights of DHNN. Earlier models integrating systematic SAT logic [5,10] did not explicitly emphasize neuron state diversity, increasing the likelihood of repetitive solutions and overfitting. Karim *et al.* addressed this issue by introducing a high-order logic structure with the Hybrid Election Algorithm, demonstrating improved neuron state diversity and synaptic weight optimization [19]. High diversification of neuron states is indicative of enhanced synaptic weight diversity, which is essential for producing high-quality and distinct solutions in multi-objective optimization. Motivated by this observation, the proposed MAJ2HES framework explicitly promotes diversification to improve DHNN robustness and solution quality.

3. Materials and methods

In this section, we describe the methodology employed in developing the proposed DHNN–MAJ2HES model. First, the formulation of Major 2 Satisfiability (MAJ2SAT) logic and its integration into the Discrete Hopfield Neural Network (DHNN) framework are presented. Next, the construction of the multi-objective DHNN model is explained. Finally, the HES algorithm, which serves as the learning mechanism of the proposed model, is introduced. This systematic description provides a clear understanding of how the proposed approach operates in solving complex multi-objective optimization problems.

3.1. Major 2 satisfiability

The logical structure defining the input and output neurons of the Discrete Hopfield Neural Network (DHNN) is based on a non-systematic Major 2 Satisfiability (MAJ2SAT) formulation. The term *major* refers to the dominance of second-order (2SAT) clauses within the logical structure. From a flexibility perspective, studies on non-systematic SAT have demonstrated that MAJ2SAT provides a suitable logical representation for improving the performance of DHNN models [21]. The MAJ2SAT logical structure, denoted by $\varphi_{MAJ2SAT}$, is characterized by the inclusion of multiple SAT clause orders, the ratio of second-order clauses, and the total number of clauses. The key elements of the MAJ2SAT formulation are defined as follows:

- A set of m second-order clauses
- A set of n third-order clauses
- A set of literals $a_i \in \{1, -1\}$
- The ratio of second-order clauses, α
- The total number of clauses, TC

The general formulation of MAJ2SAT is expressed as:

$$\varphi_{MAJ2SAT} = \bigwedge_{i=1}^{m} A_i^{(2)} \ \wedge \ \bigwedge_{i=1}^{n} A_i^{(3)}, \tag{1}$$

where $A_i^{(2)}$ and $A_i^{(3)}$ denote the i-th second-order and third-order clauses, respectively. Index i represents the clause index. Each second-order clause and third-order clause are defined in Eqs (2) and (3), respectively, as below.

$$\bigwedge_{i=1}^{m} A_i^{(2)} = (a_1 \vee b_1) \wedge \cdots \wedge (a_m \vee b_m), \tag{2}$$

$$\bigwedge_{i=1}^{n} A_i^{(3)} = (c_1 \vee d_1 \vee e_1) \wedge \cdots \wedge (c_n \vee d_n \vee e_n), \tag{3}$$

where a_i, b_i, c_i, d_i, e_i are literals represented by bipolar neuron states in $\{1, -1\}$. The subscripts associated with the literals serve as identifiers and do not indicate temporal or iterative indices. The proportion of second-order clauses in the MAJ2SAT structure is defined as in Eq (4)

$$\alpha = \frac{m}{TC}, \tag{4}$$

where $TC = m + n$ represents the total number of clauses. To ensure a non-systematic logical structure, the value of α is constrained such that $0.5 < \alpha < 1$. When $\alpha = 0.5$, the number of second-order clauses equals the number of third-order clauses, yielding a structure similar to RAN3SAT. Conversely, when $\alpha = 1$, the formulation reduces to a purely systematic 2SAT structure. Therefore, maintaining αwithin the specified range ensures a balanced and flexible logical representation.

An example MAJ2SAT structure with $\alpha = 0.67$ and $TC = 3$ is illustrated as follows:

$$\varphi_{MAJ2SAT} = (\neg a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge (c_1 \vee d_1 \vee \neg e_1). \tag{5}$$

In this example, the first two clauses correspond to second-order clauses, while the third clause represents a third-order clause. The integration of non-systematic SAT knowledge into the learning and testing phases of DHNN has gained increasing attention [23–29]. Accordingly, we employ a non-systematic MAJ2SAT formulation to enhance the learning capability and overall performance of the DHNN model.

3.2. Major 2 satisfiability in the Discrete Hopfield Neural Network

A pioneer work by the researchers in [30] emphasized the potential of Satisfiability (SAT) to represent neuron connections in symmetric weight networks, such as the Hopfield network. It was further explained that the set of SAT truth assignments may correspond to the set of solutions that

minimize the network cost function. Notably, the discrete Hopfield neural network (DHNN) was initially introduced by Hopfield and Tank with the associative property of Content Addressable Memory (CAM) for storing synaptic weights [2]. Other key characteristics of DHNN include a feedback architecture with no hidden layers and the use of bipolar neuron states in $\{1, -1\}$. Inspired by earlier studies, Abdullah introduced a method that embeds SAT as a logical rule in DHNN and optimizes logical consistency to retrieve synaptic weights [3], later known as the Wan Abdullah (WA) method. This work also demonstrated the applicability of the method to non-Horn and higher-order clauses.

Accordingly, in this paper, the formulated $\varphi_{MAJ2SAT}$ is embedded into the DHNN model, referred to as DHNN–MAJ2SAT. The primary motivation for embedding $\varphi_{MAJ2SAT}$ into DHNN is to represent neuron states S_i through literals in a logical structure, thereby providing a symbolic representation for an otherwise non-symbolic DHNN model. This integration reveals complex inter-neuron relationships and introduces nonlinearity in the information processing mechanism, from raw inputs to optimal final neuron states. The DHNN–MAJ2SAT model consists of two major phases: learning and testing. During the learning phase, the objective is to minimize the cost function, where a zero-cost value indicates that all clauses in $\varphi_{MAJ2SAT}$ are satisfied. This phase is essential for ensuring effective synaptic weight management [29] and directly influences the generation of global energy solutions during the testing phase via local field computations. The cost function of DHNN–MAJ2SAT is formulated in Eq (6) as

$$E_{\varphi_{MAJ2SAT}} = \frac{1}{4}\sum_{j=1}^{m}\left(\prod_{i=1}^{2} Q_i\right)_j + \frac{1}{8}\sum_{j=1}^{n}\left(\prod_{i=1}^{3} Q_i\right)_j, \tag{6}$$

The literal contribution used in Eq (6) is defined in Eq (7) as

$$Q_i = \begin{cases} (1 + S_{a_1}), & \text{if } a_1, \\ (1 - S_{a_1}), & \text{otherwise.} \end{cases} \tag{7}$$

Here, S_i denotes a bipolar neuron state in $\{1, -1\}$, while m and n represent the total number of second-order and third-order clauses, respectively. Index j is used exclusively to enumerate clauses. Based on the satisfied assignments of $\varphi_{MAJ2SAT}$, the probability of achieving a zero-cost solution is given in Eq (8) as

$$P\left(E_{\varphi_{MAJ2SAT}} = 0\right) = \left(\frac{3}{4}\right)^m \left(\frac{7}{8}\right)^n. \tag{8}$$

SAT-based DHNN studies have reported lower probabilities of satisfied assignments than those predicted by Equation (8), primarily due to the inclusion of first-order logic [29,31], which degrades the effectiveness of the learning phase. In this work, the synaptic weights are derived using the Wan Abdullah method by equating Eq (8) with the Lyapunov energy function defined in Eq (9) as

$$\begin{aligned} H_{\varphi_{MAJ2SAT}}(t) = \; & -\frac{1}{3}\sum_{i=1,i\neq j\neq k}^{N}\sum_{j=1,i\neq j\neq k}^{N}\sum_{k=1,i\neq j\neq k}^{N} W_{i,j,k}^{(3)} S_i S_j S_k \\ & -\frac{1}{2}\sum_{i=1,i\neq j}^{N}\sum_{j=1,i\neq j}^{N} W_{i,j}^{(2)} S_i S_j - \sum_{i=1}^{N} W_i^{(1)} S_i, \end{aligned} \tag{9}$$

where indices $i, j, k \in \{1, \ldots, N\}$ denote neuron indices, N is the total number of neurons, and $W_{i,j,k}^{(3)}$, $W_{i,j}^{(2)}$, and $W_i^{(1)}$ correspond to third, second, and first-order synaptic weights, respectively. To ensure the convergence of the DHNN, the synaptic weights are arranged in matrix form with symmetry and zero-diagonal constraints. In particular, the second-order weight matrix is defined as $W^{(2)} = [W_{i,j}^{(2)}]_{N\times N}$, where $W_{i,j}^{(2)} = W_{j,i}^{(2)}$ and $W_{i,i}^{(2)} = 0$, thereby eliminating self-loop connections and ensuring that all diagonal elements are zero [2].

Each string of satisfied assignments for $\varphi_{MAJ2SAT}$ corresponds to a complete set of synaptic weights across all interaction orders and is stored in the Content Addressable Memory (CAM) for subsequent retrieval during the testing phase. In this paper, multiple ρ-CAMs are introduced, where ρ represents the number of satisfied assignment strings of $\varphi_{MAJ2SAT}$ obtained during the learning phase of DHNN–MAJ2SAT. This approach was initially proposed by Karim *et al.* [19] to enhance the storage capacity of the fundamental DHNN. By incorporating CAMs, the overfitting issue associated with synaptic weights can be alleviated, while the search space for locating global solutions during the testing phase is effectively expanded. Furthermore, we propose improvements to the learning algorithm to successfully generate non-repetitive ρ-CAMs, which will be discussed in the subsequent section. Each ρ-CAM performs its own testing phase, thereby improving the diversification of the final neuron states $\left(S_i^{final}\right)$.

Accordingly, the generation of the optimal final neuron state S_i^{final} for each ρ-CAM in DHNN–MAJ2SAT is computed using the local field formulation. The local field computation is defined in Eq (10) as

$$h_i(t) = \sum_{k=1,\, j\neq k}^{N} \sum_{j=1,\, j\neq i}^{N} W_{i,j,k}^{(3)} S_j S_k - \sum_{j=1,\, j\neq i}^{N} W_{i,j}^{(2)} S_j + W_i^{(1)}, \tag{10}$$

where $h_i(t)$ denotes the local field acting on neuron i at time t. The neuron state update rule corresponding to the local field is given in Eq (11) as

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

where Eq (11) applies the Hyperbolic Tangent Activation Function (HTAF) to squash the local field value h_i. Parameter ζ_i denotes the threshold constraint of DHNN–MAJ2SAT and can be set to zero to ensure that the network energy decreases uniformly [32]. To evaluate the quality of the final neuron states in terms of energy convergence, the stopping criterion is defined in Eq (12) as

$$| H_{\varphi_{MAJ2SAT}}^{\min} - H_{\varphi_{MAJ2SAT}} | \leq \delta, \tag{12}$$

where δ is a tolerance value, typically set to 0.001, in agreement with other studies [4,19]. The minimum energy of the network is given in Eq (13) as

$$H_{\varphi_{MAJ2SAT}}^{\min} = -\frac{n+2m}{8}, \tag{13}$$

which is derived based on the total number of second-order and third-order clauses. To formally establish the convergence of the DHNN–MAJ2SAT model, Theorem 1 is introduced, which guarantees the nonincreasing property of the energy function under asynchronous updates.

Theorem 1

Suppose that the DHNN model is defined by $D = (W, \zeta)$. When D operates under asynchronous serial mode with symmetric and zero-diagonal weight matrix W, the network always converges to a stable state S_i^{final}.

To prove Theorem 1, the Lyapunov energy at time $t+1$ is expressed in Eq (14) as

$$H_{\varphi_{MAJ2SAT}}(t+1) = -\frac{1}{3}\sum_{i=1,\, i\neq j\neq k}^{N}\sum_{j=1,\, i\neq j\neq k}^{N}\sum_{k=1,\, i\neq j\neq k}^{N} W_{i,j,k}^{(3)} S_i^* S_j S_k - \frac{1}{2}\sum_{i=1,\, i\neq j}^{N}\sum_{j=1,\, i\neq j}^{N} W_{i,j}^{(2)} S_i^* S_j - \sum_{i=1}^{N} W_i^{(1)} S_i^*, \tag{14}$$

where $S_i^* = S_i(t+1)$. The corresponding change in energy is defined in Eq (15) as

$$\Delta H_{\varphi_{MAJ2SAT}} = H_{\varphi_{MAJ2SAT}}(t) - H_{\varphi_{MAJ2SAT}}(t+1). \tag{15}$$

By substituting Eqs (9) and (14) into Eq (15), the energy difference can be simplified as shown in Eq (16),

$$\Delta H_{\varphi_{MAJ2SAT}} = -(S_i - S_i^*)\left(\sum_{k=1,\, j\neq k}^{N}\sum_{j=1,\, j\neq i}^{N} W_{i,j,k}^{(3)} S_j S_k - \sum_{j=1,\, j\neq i}^{N} W_{i,j}^{(2)} S_j + W_i^{(1)}\right), \tag{16}$$

which can be further compactly expressed in Eq (17) as

$$\Delta H_{\varphi_{MAJ2SAT}} = -(S_i - S_i^*)h_i(t). \tag{17}$$

Equations (16) and (17) demonstrate that the energy of the DHNN–MAJ2SAT model is nonincreasing and converges to a point at which further energy reduction is not possible. Hence, the convergence of the DHNN energy function, as stated in Theorem 1, is formally established. In this paper, the resulting final neuron states S_i^{final} are evaluated using multiple performance metrics related to diversity and global energy solutions, thereby highlighting the effectiveness of DHNN–MAJ2SAT in producing global minimum solutions for combinatorial optimization tasks.

3.3. Multi–objective Discrete Hopfield Neural Network

In this study, a multi-objective Discrete Hopfield Neural Network (MOF-DHNN) is proposed to address limitations associated with restricted storage capacity and insufficient solution diversity in conventional DHNN models. Unlike the single-objective DHNN (SOF-DHNN), which typically converges to a single optimal neuron state, the MOF-DHNN is designed to generate multiple high-quality neuron state configurations by jointly considering several optimization objectives. Studies on

DHNN optimization have primarily entailed minimizing a single cost function derived from systematic SAT logic [33,34]. Although such approaches are effective in achieving global energy minimization, they often neglect the quality and diversity of the resulting neuron states. This limitation increases the likelihood of repetitive solutions and restricts the model to a single Content Addressable Memory (CAM), thereby constraining the exploration of the solution space.

To overcome these issues, the proposed MOF-DHNN integrates a non-systematic MAJ2SAT logical structure, denoted as $\varphi_{MAJ2SAT}$, which enables greater flexibility in logical representation and supports multi-objective optimization. The MAJ2SAT formulation plays a critical role in the MOF-DHNN, as it encodes satisfied logical assignments that are directly associated with the optimization objectives. In the proposed framework, three objective functions are introduced: The fitness function f, the diversity function d, and the production of power strings $\widetilde{H}$. These objectives are formulated within a hierarchical multi-objective optimization structure, rather than a Pareto-based framework. Specifically, objectives are optimized sequentially, where higher-priority objectives must be satisfied before subsequent objectives are evaluated. This formulation avoids ambiguity associated with Pareto dominance and is well suited to the layered architecture of DHNN. Accordingly, the MOF-DHNN is defined as a multi-objective maximization problem, expressed as in Eq (18)

$$\text{Maximize: } F(f, d, \widetilde{H}), \tag{18}$$

where each objective contributes to improving the quality, diversity, and storage capability of the DHNN model. The detailed formulations of these objective functions are described in the following subsections.

3.3.1. Fitness function

The first objective function aims to maximize the number of satisfied clauses in the MAJ2SAT formulation, as shown in Eq (19). The fitness function evaluates the degree to which a neuron state configuration satisfies the logical constraints encoded in $\varphi_{MAJ2SAT}$. Achieving maximum fitness corresponds to minimizing the DHNN energy function, i.e., $E_{\varphi_{MAJ2SAT}} = 0$, which ensures optimal synaptic weight formation.

The fitness maximization problem is defined as

$$\max f = \begin{cases} \sum_{i=1}^{TC} f_i, \\ f_i > 0, i = 1,2, \dots, TC, \end{cases} \tag{19}$$

where f_i denotes the satisfaction status of the i-th clause in $\varphi_{MAJ2SAT}$, and TC represents the total number of clauses, given by $TC = m + n$. The maximum fitness value is therefore $f_{\max} = TC$, indicating that all clauses are satisfied.

3.3.2. Diversity function

While fitness optimization ensures logical consistency, it does not guarantee diversity among neuron state configurations. To address this limitation, a diversity function is introduced as the second objective to promote variation in neuron states and reduce the risk of overfitting. The diversity function measures the difference between generated neuron states and benchmark neuron states. Let d_i denote the number of distinct neuron states in a solution. The diversity maximization problem is defined as in Eq (20)

$$\max\ d = \begin{cases} \mu \sum_{i=1}^{2m+3n} d_i = \mu(2m+3n), \\ d_i > 0, \end{cases} \tag{20}$$

where μ is the diversity threshold, and m and n represent the number of 2SAT and 3SAT clauses, respectively. In this study, $\mu = 0.1$. Optimal diversity is achieved when the number of distinct neuron states reaches $\mu(2m + 3n)$, ensuring sufficient variation among candidate solutions.

3.3.3. Power string production

The third objective focuses on the production of power strings, denoted as $\widetilde{H}$. A power string is defined as a neuron state configuration that simultaneously satisfies the maximum fitness and diversity criteria. Each power string is unique and represents a high-quality solution suitable for storage in CAM. The power string maximization problem is formulated as Eq (21)

$$\max \widetilde{H} = \begin{cases} \widetilde{H} \in \max(f, d), \\ \widetilde{H} \neq 0, \\ \widetilde{H} \in \mathbb{N}. \end{cases} \tag{21}$$

Each selected power string is stored in its own CAM, resulting in a multi-CAM DHNN architecture. If ρ power strings are generated, the DHNN model will consist of $\rho -$CAMs. This mechanism enables the MOF-DHNN to expand its storage capacity while maintaining solution quality.

Figure 2 illustrates the structural differences between the SOF-DHNN and the proposed MOF-DHNN. In contrast to the single-layer architecture of SOF-DHNN, the MOF-DHNN employs a layered structure consisting of a fitness layer, a diversity layer, and a power string layer. This layered formulation supports hierarchical multi-objective optimization and facilitates the generation of diversified and high-quality neuron states.

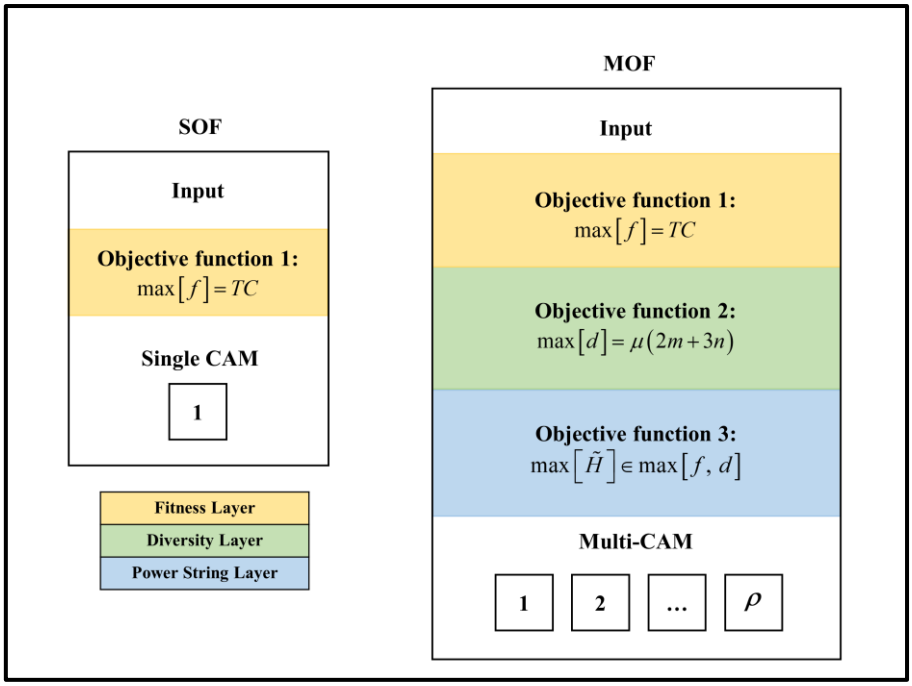

Figure 2. An illustration of SOF and MOF of the DHNN model.

3.4. The Hybrid Exhaustive Search algorithm

To achieve the proposed multi-objective functions (MOF), a learning algorithm is incorporated into the DHNN framework. In this study, a HES algorithm is implemented within the DHNN model using MAJ2SAT logic, referred to as DHNN–MAJ2HES. The HES algorithm is inspired by the classical Exhaustive Search (ES) method but is enhanced with a targeted search operator known as intelligent mutation. Conventional ES relies on a trial-and-error strategy that evaluates all possible candidate solutions to guarantee optimality. Although ES is theoretically optimal, its computational cost becomes prohibitive for large problem sizes [35]. In the proposed framework, exhaustiveness is not applied to the complete neuron state space 2^N. Instead, exhaustive evaluation is restricted to the clause satisfaction level, while solution improvement is guided by intelligent mutation within bounded generations. This design significantly reduces computational complexity while preserving the systematic nature of ES.

The intelligent mutation operator is inspired by the mutation mechanism in Genetic Algorithms (GA), but with fundamental differences. Standard GA mutation typically flips genes randomly, whereas the proposed intelligent mutation detects unsatisfied MAJ2SAT clauses and selectively modifies only the neuron states contributing to unsatisfaction. This targeted detect-and-mutate strategy enables the algorithm to focus on promising regions of the search space, thereby reducing unnecessary exploration and accelerating convergence.

Unlike classical SAT heuristics such as GSAT or WalkSAT [36–38], which rely on probabilistic variable flipping and random walks, the proposed intelligent mutation operates in a deterministic and clause-guided manner within the DHNN framework. Mutation is applied only to neuron states associated with unsatisfied clauses and is evaluated jointly using fitness and diversity objectives. This integration distinguishes the proposed approach from conventional local search heuristics and aligns it with neural optimization principles. Overall, the DHNN–MAJ2HES algorithm consists of six major stages: population initialization, intelligent mutation, fitness assessment, diversity assessment, selection of power strings, and termination.

3.4.1. The Full Implementation DHNN-MAJ2HES Algorithm

A detailed description and stages involved in multi–objective DHNN–MAJ2HES algorithm is presented below.

Stage 1: Population initialization

Each iteration of the HES algorithm utilizes a population *P* with a random distribution, denoted by $P = \left[H_1, H_2, H_3, \ldots, H_i \right]$. Every $H_i \in P$, is referred to as a solution of $\varphi_{MAJ2SAT}$. $H_i = \left[A_{i1}^{(k)}, A_{i2}^{(k)}, \ldots A_{iD}^{(k)} \right]$ shows the solution consists of a set of clauses, $A_{iD}^{(k)}$, where $k = 2,3$, denoting the 2SAT and 3SAT clause with *i* iteration and dimension space *D*. Moreover, each clause represented either a satisfied (true) or unsatisfied (false) solution. Then, the solutions pass to a fitness calculation based on satisfied clauses.

Stage 2: Intelligent mutation

This operator simulates the mutation operator in GA. It is obvious that not all solutions return maximum fitness value at the initial stage, but only a proportion of the population. Therefore, this operator introduces the detection of all unsatisfied clauses in each solution before mutating the neuron states, $S_i \in A_i^{(k)}$. Figure 3 shows the neuron states before $\left(S_i^{before}\right)$ and after $\left(S_i^{after}\right)$ applying intelligent mutation. Additionally, the mutation rate is the probability that states in given unsatisfied clauses in each solution will change from 1 to –1 or vice versa. If mutation rate is equal to 1, there is 100% possibility the solution will be mutated.

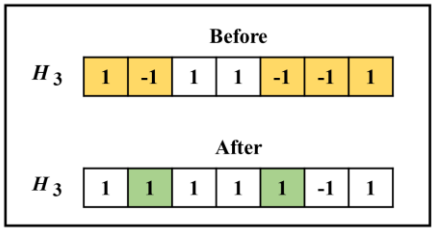

Figure 3. The neuron states before and after applying the intelligent mutation.

To mathematically model this operator, at each iteration, for each generation in intelligent mutation operator, y number of solutions are selected to mutate because of $f_i \neq TC$. Here, it is assumed that $TC = 3$ by referring to Eq (4). The operator detect solution that gives $f_i = 1$ instead of $f_i = 3$. Thus, the operator will attempt to mutate the neuron states toward the maximum fitness. As can be seen from Figure 3, the yellow box shows that the algorithm managed to detect states in unsatisfied clause. These states will be mutated according to mutation rate. Moreover, green states are the updated states which give the best solution $f_i = 3$. The process will iterate until all sets of improved solutions attain maximum fitness. The iteration will stop until it reaches the maximum number of generations (*NG*). Overall, the differences between normal mutation with intelligent mutation are the nature of mutation and the number of states to be mutated. In normal mutation [5], the states are mutated randomly with random number of states to mutate. Moreover, intelligent mutation applies a detection process on the unsatisfied clause before it mutates the states in that unsatisfied clause only. Intelligent mutation does not mutate the states that will lead to unsatisfied logic. Eq (22) shows the piecewise equation on which states need to be mutated.

$$S_i = \begin{cases} S_i^{after} \neq S_i^{before}, & A_i^{(k)} = -1 \\ S_i^{after} = S_i^{before}, & A_i^{(k)} = 1 \end{cases} \tag{22}$$

Stage 3: Fitness assessment

All sets of solutions that have passed the intelligent mutation will be evaluated based on the fitness value according to Eqs (23) and (24). The process will iterate until all sets of improved solutions attain $f_{\max}$. Furthermore, the iteration will stop until it reaches the maximum number of generations.

$$f = \sum_{i=1}^{m} A_i^{(2)} + \sum_{i=1}^{n} A_i^{(3)} \tag{23}$$

$$A_i^{(k)} = \begin{cases} 1, & satisfied \\ 0, & unsatisfied \end{cases} \tag{24}$$

Stage 4: Diversity assessment

The effectiveness of the diversity phase is computed based on the full fittest solution. The full fittest solution is defined as all solutions that achieves $f_{\max}$. The diversity neuron states for each solution will be evaluated whether equal and exceed $d_{\max}$ or not. A solution that exceeds $d_{\max}$ will return μ value. This condition needs to be fulfilled by all sets of solutions. Otherwise, the algorithm will be repeated until a maximum number of learning in the learning phase of DHNN. The total number of different neuron states that exceed μ $\left(d \geq \mu\right)$ is denoted as optimal diversity results as referred to Eqs (25) and (26).

$$d = \begin{cases} 1 & if \ \left(S\,a_1\right) = \left(\neg S\,a_1\right) \\ 0 & otherwise \end{cases} \tag{25}$$

$$\sum_{i=1}^{2m+3n} \frac{d_i}{2m+3n} \geq \mu \tag{26}$$

Stage 5: Selection of power strings

This stage is the most significant process in assuring the MOF of the proposed algorithm is achieved. The aim to increase the storage capacity of DHNN model depends on the production of power strings. Here, the solution that achieve two objective functions which are maximum fitness and maximum diversity is called as power string $\left(\tilde{H}\right)$. We select ρ number of power strings $\left(\tilde{H} = \left[\tilde{H}_1, \tilde{H}_2, \ldots, \tilde{H}_\rho\right]\right)$ to store in CAM thus carry the information into the testing phase of DHNN.

Stage 6: Termination criteria

The process of finding $\tilde{H}_\rho$ will keep iterating until reach maximum number of learning. If the simulation is unable to retrieve $\tilde{H}_\rho$, the process will be repeated from initialization phase until the selection of power strings. Figure 4 shows the overall process in HES algorithm. Moreover, Algorithm 1 outlines the proposed MAJ2HES learning framework. The HES component systematically explores feasible neuron state combinations within the MAJ2SAT solution space, while an intelligent mutation operator is applied across successive generations $(\text{gen} = 1, \ldots, NG)$ to identify and modify neuron states associated with unsatisfied MAJ2SAT clauses. Multi-objective optimization is achieved by jointly considering fitness, diversity, and power string utilization, thereby ensuring a balanced trade-off between exploration and exploitation throughout the learning process.

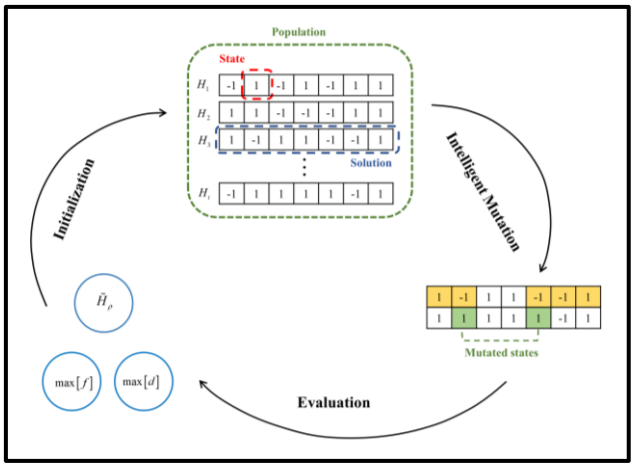

Figure 4. A schematic of the HES algorithm.

ALGORITHM 1. Pseudocode for Multi-objective DHNN with Hybrid Exhaustive Search.

Input:

Population size P
MAJ2SAT clause ratio α
Total clauses TC
Diversity threshold μ
Maximum learning iteration ω
Maximum generation NG

Output:

Number of power strings ρ

BEGIN
Repeat.
While iter $\leq \omega$ do.
Initialization:
Randomly initialize the population P in the search space.
Evaluate the fitness value $f(H_i)$ for all solutions H_i.
For gen $= 1$ to NG do.
Intelligent Mutation:
Detect unsatisfied clauses $A_i^{(k)} = -1$.
Mutate the neuron states S_i from 1to -1 or vice versa to produce an updated solution H_i.
Fitness Assessment:
Select the top x solutions with maximum fitness $f_{\max}$ to calculate the fitness error.
End for.
Diversity Assessment:
Select the top x solutions with maximum diversity $d_{\max}$ to calculate the diversity error.
Selection of Power Strings:
Select power strings ρ, which consist of solutions satisfying both $f_{\max}$ and $d_{\max}$.
End while.
Until the desired number of power strings ρ is achieved.
Return the number of power strings ρ.
END

3.4.2. Storage capacity expansion of DHNN–MAJ2HES

A large set of potential solutions may require increased storage capacity to produce optimal outputs in the DHNN model. The researchers in [28] report that the storage capacity of a conventional DHNN is approximately limited to $0.14N$, where N denotes the number of optimal solutions that can be stored. In other studies [12,20], the search space and storage capacity of DHNN were restricted by the use of a single Content Addressable Memory (CAM), resulting in a storage complexity of approximately 0.14. In contrast, the proposed DHNN–MAJ2HES framework expands storage capacity through the selective generation of power strings, rather than through naive replication of network structures. Each power string represents a high-quality neuron state configuration that simultaneously satisfies fitness and diversity constraints. Only these qualified solutions are stored in CAM, ensuring that additional memory resources are allocated conditionally and efficiently.

Following the recommendation in [39], multiple CAMs are introduced only after the generation of optimal final neuron states S_i^{final}. As a result, the storage complexity of the DHNN–MAJ2HES model becomes proportional to the number of power strings ρ, yielding an effective storage capacity of approximately 0.14ρ. This mechanism enables the proposed algorithm to increase storage capacity by up to ρ times while avoiding redundant memory allocation. Although the use of multiple CAMs introduces additional hardware requirements, this increase is not a direct scaling of network resources. Instead, it reflects an algorithmic improvement in solution selection and storage efficiency, as only diversified and optimal neuron states are retained. Consequently, the proposed approach achieves a balanced trade-off between enhanced storage capacity and hardware cost.

4. Experimental study

The effectiveness of the DHNN–MAJ2HES model is described in this section. The experiment runs using the A315–57G device with Windows 11, an Intel® Core™ i7–1065G7 processor, and 8GB RAM. The learning and testing data set employed in our study are generated synthetically by the software Dev C++ (version 5.11). It is important to use same device and programming language to avoid bias in generating results. The simulated data sets are produced in bipolar neuron states either 1 or –1 $\left(S_i \in \{1,-1\}\right)$. The utilization of bipolar representation is important in this paper to ensure the DHNN model can operate in asynchronous neuron updating and carry the optimal synaptic weight [40].

4.1. Experimental setting

We cover a wide range dimension space of a higher number of *TC* and α. The value of *TC* range from 10 to 40 considering the step size equal to 10. The range for α is dependent on the *TC* size. In this paper, the step size for α is set to 0.1 and 0.05. For $TC = 10,30$ the range is $\alpha = \{0.6, 0.7, 0.8, 0.9\}$. Moreover, as for $TC = 20,40$, the range is $\alpha = \{0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$. This means that we consider the imbalance of α in this paper to show the flexibility of $\varphi_{MAJ2SAT}$ to explore parameter settings. Additionally, we want to prove that the results of the proposed DHNN–MAJ2HES model are consistent throughout the experiment with a variation of *TC* and α. The range for *TC* and α can be set to a high or low value. However, we restrict the range according to Table 1 to have a fair comparison between two different ranges of α. Table 1 shows the possible combination

of 2SAT and 3SAT clauses generated in this paper according to TC and α. Although Table 1 reports large neuron counts for higher values of TC, the proposed DHNN–MAJ2HES algorithm does not perform exhaustive enumeration over the full neuron state space 2^{NN}. Instead, exhaustive evaluation is applied at the clause satisfaction level, while solution updates are guided by intelligent mutation within bounded generations.

Table 1. Possible combinations of 2SAT and 3SAT clauses for different values of TC and α, and the corresponding number of neurons (NN).

TC	***m***	***n***	α	***NN***
10	6	4	0.6	24
	7	3	0.7	23
	8	2	0.8	22
	9	1	0.9	21
20	11	9	0.55	49
	12	8	0.6	48
	13	7	0.65	47
	14	6	0.7	46
	15	5	0.75	45
	16	4	0.8	44
	17	3	0.85	43
	18	2	0.9	42
	19	1	0.95	41
30	18	12	0.6	72
	21	9	0.7	69
	24	6	0.8	66
	27	3	0.9	63
40	22	18	0.55	98
	24	16	0.6	96
	26	14	0.65	94
	28	12	0.7	92
	30	10	0.75	90
	32	8	0.8	88
	34	6	0.85	86
	36	4	0.9	84
	38	2	0.95	82

4.2. Parameter assignment

Following most of the guidance provided in [19], the proposed model executes the learning and testing phase 100 times independently. The difference is that we run the algorithm based on one neuron combination, unlike in [19], where a 100–neuron combination was run. This is due to the reason that we want to see the real implication of one neuron combination to the results and reduce the computational time. The population size is initialized to 100 solutions (or strings) for all proposed algorithms. A set of 100 solutions is obtained randomly in each simulated data set. The parameter

settings for all benchmark algorithms are adopted from their respective reference studies and commonly used configurations in the literature to ensure fair comparison, while the population size is fixed across all methods. In accordance with the settings suggested in the original works, additional parameters for each method are specified and listed in Table 2. The neuron states in the learning and testing phase are initialized randomly following the work by [40]. Notably, the simulation that exceeds more than 24 hours is terminated, and the results are not valid. The Hyperbolic Tangent activation function (HTAF) is used in this experiment to increase the capability of DHNN–MAJ2HES model in retrieving stable final neuron states.

Table 2. Parameters specifications of DHNN–MAJ2HES and comparison algorithms.

Parameter	**Value/Range**
Number of power strings, (ρ)	5
2SAT ratio in $\varphi_{MAJ2SAT}$, (α)	$0.55 \leq \alpha \leq 0.95$
Total clauses in $\varphi_{MAJ2SAT}$, (TC)	$10 \leq TC \leq 40$
Diversity threshold, (μ)	0.1
Population size, (P)	100
Neuron combination	1
Number of generations, (NG)	100
Number of learning, (ω)	100 [19]
Number of trials, (υ)	100 [19]
Threshold time	24 hours [19]
Tolerance value, (∂)	0.001 [4]
Relaxation rate	3 [4]

4.3. Baseline models

To investigate the performance of the DHNN–MAJ2HES model, we select six benchmark algorithms representing metaheuristic and heuristic approaches: Particle Swarm Optimization (PSO), Clonal Selection Algorithm (CSA), Election Algorithm (EA), Genetic Algorithm (GA), Artificial Bee Colony (ABC), and Exhaustive Search (ES). GA and CSA are evolutionary–based, PSO and ABC utilize swarm–intelligence principles, EA is rooted in socio–political algorithms, and ES operates as a heuristic algorithm. The key distinction among these methods lies in the presence of global and local search operators. It is crucial to note that HES algorithms are compared to these benchmarks without the integration of an intelligent mutation operator, ensuring a fair comparison. By focusing on individual merits, this approach enables us to highlight the distinctive features of DHNN–MAJ2HES, emphasizing the effectiveness of its intelligent mutation operator as a vital component. This comprehensive evaluation provides valuable insights into the unique advantages offered by our proposed approach within the landscape of optimization algorithms. Below are the detailed descriptions of chosen algorithms that will be compared with HES.

4.3.1. Particle Swarm Optimization

PSO was initially introduced in [41], which imitates the social behavior of birds flocking. There

are no recent studies from the perspective of SAT in DHNN that utilizes PSO as the learning algorithm to achieve a zero-cost function. However, the researchers in [40] concluded that PSO obtained competitive performance compared to the Grey Wolf Optimization (GWO) algorithm in terms locating local–best and global–best solutions. This creates an opportunity to see the ability of PSO operators to learn MAJ2SAT logical rules in the learning phase of DHNN. In PSO, each bird in the flock serves as a swarm particle. This particle in turn represents a potential solution (a leader who has the closest position to the food). The particle swarm searches space in each dimension for the best solution that optimizes the problem at hand [42].

4.3.2. Clonal Selection algorithm

The CSA mimics the clonal selection principle where the fundamental features of an immune response to an antigenic stimulus [43]. CSA consists of cloning, mutation, and selection operators. Cloning is used to generate new individuals with higher affinity values. Clones are perturbed by mutation, which also keeps the population diverse. Lower affinity individuals are eliminated through selection. Through an experiment carried out by Zamri *et al.* with a simulated dataset, CSA has showed optimal performance in the learning phase of DHNN due to the somatic hypermutation operator [11].

4.3.3. Election algorithm

In the last few years, the Election algorithm (EA) has been popular in DHNN because of the iterative operators and systematic partition of the solution space that can reduce the potential of DHNN trapped in a suboptimal solution. An experiment conducted by Bazuhair *et al.* displayed the potential of EA in achieving optimal synaptic weight management of RAN3SAT in DHNN [15]. Even when RAN3SAT considered high number of first–order logic, EA can locate the satisfied interpretation of the logical rule.

4.3.4. Genetic algorithm

The Genetic algorithm (GA) is based on finding a potential solution set for a given population. Once the first population is generated, the next generation evolves to produce a better near–optimal population. The crossover and mutation operator will occur to improve the subsequent population of the solution. The GA can improve the learning phase of DHNN and attains 90% of global minima solutions [5]. This study further concluded that GA works efficiently in DHNN due to the balanced optimization operators in the algorithm such as crossover (responsible for exploring solutions in other search spaces) and mutation (responsible for exploiting near–optimal solutions).

4.3.5. Artificial Bee Colony algorithm

Inspired by the foraging behavior of honeybees, Karaboga and Basturk proposed a new variant of swarm intelligence, namely Artificial Bee Colony (ABC) [44]. There are three major operators in ABC: employed bee, onlooker bee, and scout bee. It is worth mentioning that the performance of ABC depends on the updated rule equation of ABC, which controls the movement of the bees toward profitable nectar around the hive. The work by Kasihmuddin *et al.* reported the superior performance

of ABC in learning the logical rule of 2SAT in the Radial Basis Function Neural Network (RBFNN) [45]. The proposed ABC outperforms other state–of the art algorithms such as GA and PSO.

4.3.6. Exhaustive Search algorithm

The ES algorithm, also known as the brute-force search algorithm, is a simple and easy-to-implement approach [46]. However, this algorithm may incur high computational cost and can become infeasible for large problem sizes. Despite this limitation, ES is intentionally included in this study as a baseline model, particularly in the investigation of SAT-based DHNN models, where ES is widely regarded as the benchmark searching technique during the learning phase. Although ES does not employ any optimization operators, it has been reported to effectively minimize the DHNN cost function when the number of neurons is small [12]. Therefore, its inclusion enables a direct and fair assessment of how the proposed HES improves upon the fundamental ES framework.

4.4. Performance indicators

The effectiveness of the DHNN–MAJ2HES in terms of learning and testing phase is assessed using several performance metrics. In the learning phase of DHNN–MAJ2HES, the Mean Absolute Error (MAE) for respective objective functions as in Eqs (23) and (25) is formulated in Eqs (27) and (28). According to a study by the researchers in [47], MAE is widely used to measure model fitting. In this context, the value of MAE is convenient to show us how close the obtained value is with the desired value. As presented in Eq (27), the formulated MAE to evaluate the performance of respective learning algorithms in terms of fitness is represented as MAE_f. Notably, the values of MAE_f investigate the minimization of cost function of DHNN.

$$MAE_f = \frac{\sum_{i=1}^{\rho}\sum_{j=1}^{\omega}\left|f_{\max} - f_{ij}\right|}{\rho\omega}, \tag{27}$$

where f_{ij} refers to fitness value for *j*–th number of learning for *i*–th power string. Additionally, another formulated MAE to measure the quality of satisfied neuron interpretation in terms of diversity is represented as MAE_d, as shown in Eq (28). The values of MAE_d indicate the diversified neuron states of MAJ2SAT.

$$MAE_d = \frac{\sum_{i=1}^{\rho}\sum_{j=1}^{\omega}\left|d_{\max} - d_{ij}\right|}{\rho\omega}, \tag{28}$$

whereby d_{ij} is the diversity value for *j*–th number of learning for the *i*–th power string. All solutions that obtain $d_{ij} \geq \omega$ will standardized to the $d_{\max}$ value. This explains the strategy of evaluating

diversity neuron states. The total number of power strings or ρ depicts the total number of strings that possess the desired quality of strings in terms of fitness and diversity. A high number of ρ is considered optimal, as this exhibits the successful rate of the respective learning algorithm in solving MOF problems. Eq (29) highlights the criteria considered in categorizing ρ as follows:

$$\rho = n\left(f_{\max} \cap d_{\max}\right) \tag{29}$$

Conjointly, three performance metrics are assessed in the testing phase to investigate the impact of HES compared to other baselines in terms of the production of final neuron states. The first performance metrics to assess the testing phase is the ratio of global solutions or R_G. R_G are widely used in related studies such as in [48] and [23]. According to both works, when R_G approaches one $\left(R_G \to 1\right)$, this finding affirms that the logical rule is successfully embedded in DHNN. Equation (30) shows the formulation of R_G as follows:

$$R_G = \frac{\sum_{i=1}^{\rho}\sum_{j=1}^{\upsilon} G_{\varphi_{MAJ2SAT}}}{\rho\upsilon} \tag{30}$$

where $G_{\varphi_{MAJ2SAT}}$ is defined as number of global minimum solutions more than ∂. Equations (31) and (32) measure the quality of the retrieved final neuron states by respective DHNN models. In this context, the quality refers to whether the states are overfitting or not. R_τ, or ratio of variations, evaluates the variations of neuron states, where higher values of R_τ showcase dissimilarity among other retrieved states. Based on a study by the researchers in [19], high variations highlight DHNN's ability to locate non–overfitting solutions. In relation to this, the researchers in [19] also included a similarity index measurement formula to measure the co–occurrences of positive–negative states. This has inspired us to include the Jaccard Index or R_{JDSI} to see the positive states produced by the DHNN–MAJ2HES model and other baselines. The formulation of R_{JDSI} can be seen in Eq (33). Moreover, Table 3 depicts the details of all coefficients in R_{JDSI}.

$$R_\tau = \frac{\sum_{i=1}^{\rho}\sum_{j=1}^{\upsilon} J_i^{(n)}}{\rho\upsilon} \tag{31}$$

$$J_i^{(n)} = \begin{cases} 0, & if \quad \varphi_{MAJ2SAT}^{n} = \varphi_{MAJ2SAT}^{n+1} \\ 1, & if \quad \varphi_{MAJ2SAT}^{n} \neq \varphi_{MAJ2SAT}^{n+1} \end{cases} \tag{32}$$

$$R_{\text{JDSI}} = \sum_{i=1}^{\rho} \frac{p_i}{\rho\left(p_i + q_i + r_i\right)} \tag{33}$$

Table 3. Similarity coefficient in R_{JDSI}.

Coefficient	**Benchmark,** $\hat{S}_i$	**Final,** S_i^{final}
p	1	1
q	1	–1
r	–1	1

4.5. Full implementation of the DHNN–MAJ2HES algorithm

The basic idea of implementing $\varphi_{MAJ2SAT}$ and the HES algorithm in DHNN is to improve the pioneer work from having one objective function into multi–objective functions. In this way, multi–objective functions can increase the search space and storage capacity of DHNN. As the HES algorithm takes advantage of the original ES algorithm, this new algorithm is proposed to outperform other benchmark learning algorithms. As shown in Figure 5, we provide a flowchart of the proposed model, which implements $\varphi_{MAJ2SAT}$ and HES in DHNN (DHNN–MAJ2HES) in solving multi–objective functions. The proposed DHNN–MAJ2HES model is split into the learning phase and testing phase. In the early process, it is important to create $\varphi_{MAJ2SAT}$ according to α, which the number of 2SAT clauses (m) need to be more than the number of 3SAT clauses (n). The process then follows solving multi–objective functions through the learning algorithms. In this paper, the learning algorithm used is the HES to navigate the fitness and diversity stage. The output of HES is the production of power strings with the top highest positive literals. After that, the synaptic weight calculation using the Wan Abdullah method is performed, and all the information is fused and stored in Content Addressable Memory (CAM). It is important to store each power string according to its own CAM. The testing phase is focused on producing high quality and stable final neuron states. This phase starts by calculating the local field equation followed by the final neuron states filtering through the Hyperbolic Tangent activation function (HTAF). Notably, the proposed model needs to be relaxed using the Sathasivam relaxation method with a relaxation rate equaling three to generate more stable final neuron state with global minimum energy. At the end of the process, the performance of the proposed model will be evaluated using suitable metrics. The results are recorded to verify the effectiveness of the DHNN–MAJ2HES model outperforming the benchmark algorithms.

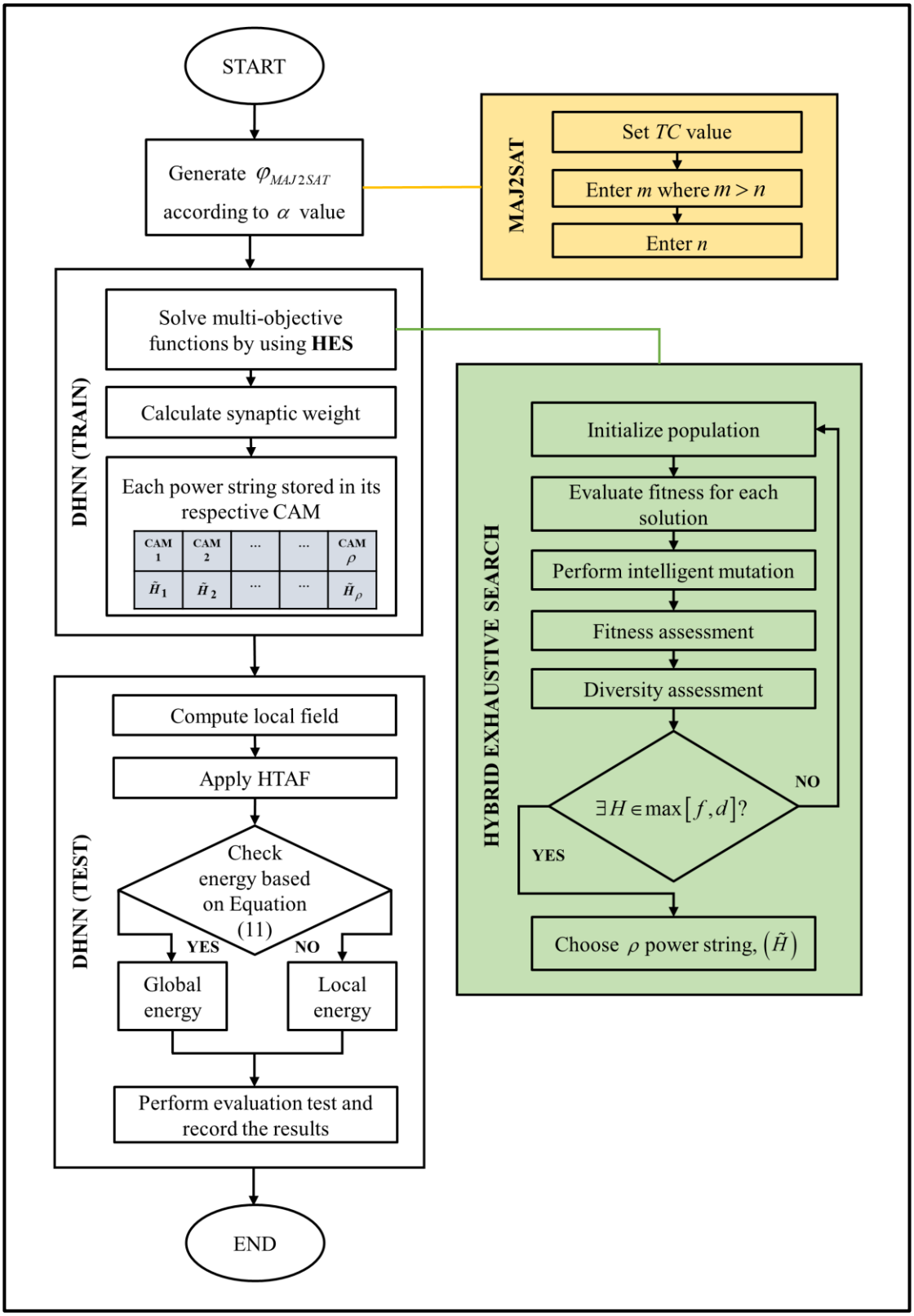

Figure 5. A flowchart of the proposed DHNN–MAJ2HES model.

5. Results and discussion

The findings of the experiments are presented and analyzed in this section. First, the performance of DHNN–MAJ2HES is analyzed and then compared with metaheuristics. The results are divided into two phases, namely the learning phase and the testing phase. They are reported using graphical representations for four cases, average results summarizing all cases, and the Friedman test for the chosen metric in each phase. Graphical plots and numerical tables are presented to provide a comprehensive evaluation. In some cases, algorithms exhibit identical or near-identical performance, causing their curves to coincide in the figures. Therefore, the following tables are included to enable precise quantitative comparisons when visual distinction is limited.

5.1. Analysis of the learning phase

For the analysis of learning phase performance, four cases with different *TC* sizes are selected to evaluate the behavior of each metaheuristic under varying network complexity. The results of the learning phase are presented in Figure 6 and Table 4, where the mean absolute error of the fitness function $\left(MAE_f\right)$ is used to assess the effectiveness of each algorithm in minimizing the DHNN cost function. The objective of this evaluation is to determine which optimization strategy performs most effectively with the MAJ2SAT fitness formulation in achieving fully satisfied interpretations. From the results, the proposed HES consistently achieves $MAE_f = 0$ across all tested TC and α values, indicating complete cost function minimization during the learning phase. This outcome does not arise from dataset simplicity or overfitting, but rather from the targeted learning mechanism embedded in HES. Specifically, the intelligent mutation operator detects unsatisfied MAJ2SAT clauses and selectively mutates only the neuron states associated with these clauses. This clause-driven mutation strategy prevents unnecessary random state flipping and ensures that learning progress is directly guided toward cost reduction. As a result, the DHNN consistently converges to satisfied interpretations regardless of problem size.

The second–best algorithm is PSO that managed to achieve optimal MAE_f at $TC = 10$ and $TC = 20$. However, PSO start showing increasing MAE_f results when *TC* gets higher. This trend is also like the performance of the EA, which attains optimal MAE_f at $TC = 10$ and the error increasing as TC increases. We strongly agree that both algorithms are unable to achieve full satisfied interpretation as *TC*. The main reason behind this result is that PSO is easily trapped in the suboptimal solution despite their fast convergence speed [49]. The particle's position is not guaranteed to always achieve optimal results due to the transfer function of PSO and the influenced of randomly generated values that decide the 1 or –1 interpretation [50]. As for EA, the positive advertisement is unable to give an assurance that the current voter will improve its fitness. This is because of the process of flipping the neuron states randomly (from 1 to –1 or from –1 to 1) on the chosen voters occurring. Therefore, there is a possibility that the current voter has less fitness compared to previous voters. According to Table 4, we can see that the performance of CSA, GA, ES, and ABC is similar. These four metaheuristics share almost the same MAE_f in each case. For example, in the first case, which is $TC = 10$, these algorithms obtain $MAE_f = 9.9$. This explains that the stated algorithms require 99 iterations to achieve $E_{\varphi_{MAJ2SAT}} = 0$.

Table 4. MAE_f values obtained per metaheuristic for all *TC*. The bracket indicates the ratio of improvement.

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
10	0.60	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)
	0.70	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)
	0.80	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)
	0.90	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)	9.9000 (1.0000)
	Average	**0.0000**	**0.0000**	**0.0000**	9.9000	9.9000	9.9000	9.9000
20	0.55	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.9200 (1.0000)	19.8000 (1.0000)
	0.60	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8800 (1.0000)
	0.65	**0.0000** (0.0000)	0.0400 (1.0000)	**0.0000** (0.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.9600 (1.0000)
	0.70	**0.0000** (0.0000)	0.0400 (1.0000)	**0.0000** (0.0000)	19.8000 (1.0000)	19.8800 (1.0000)	19.8400 (1.0000)	19.8000 (1.0000)
	0.75	**0.0000** (0.0000)	**0.0000** (0.0000)	4.0000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)
	0.80	**0.0000** (0.0000)	**0.0000** (0.0000)	4.0000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)
	0.85	**0.0000** (0.0000)	**0.0000** (0.0000)	8.0000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)	19.8000 (1.0000)
	0.90	**0.0**000 (0.0000)	0.1200 (1.0000)	8.0000 (1.0000)	19.8000 (1.0000)	19.8800 (1.0000)	19.8000 (1.0000)	19.8400 (1.0000)
	0.95	**0.0000** (0.0000)	0.0400 (1.0000)	16.0000 (1.0000)	19.8000 (1.0000)	19.8800 (1.0000)	19.8400 (1.0000)	19.8000 (1.0000)
	Average	**0.0000**	0.0267	4.4444	19.8000	19.8267	19.8222	19.8311
30	0.60	**0.0000** (0.0000)	3.0000 (1.0000)	28.0000 (1.0000)	29.7000 (1.0000)	29.7600 (1.0000)	29.7000 (1.0000)	30.0000 (1.0000)
	0.70	**0.0000** (0.0000)	1.1400 (1.0000)	24.0000 (1.0000)	29.7000 (1.0000)	29.9400 (1.0000)	29.8800 (1.0000)	29.8200 (1.0000)
	0.80	**0.0000** (0.0000)	3.8400 (1.0000)	18.0000 (1.0000)	29.7000 (1.0000)	30.0000 (1.0000)	29.9400 (1.0000)	30.0000 (1.0000)
	0.90	**0.0000** (0.0000)	3.3600 (1.0000)	27.0000 (1.0000)	29.7000 (1.0000)	30.0000 (1.0000)	29.9400 (1.0000)	30.0000 (1.0000)
	Average	**0.0000**	2.8350	24.2500	29.7000	29.9250	29.8650	29.9550
40	0.55	**0.0000** (0.0000)	1.9200 (1.0000)	36.0000 (1.0000)	39.8400 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)
	0.60	**0.0000** (0.0000)	16.2400 (1.0000)	37.3333 (1.0000)	39.6800 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	39.8400 (1.0000)
	0.65	**0.0000** (0.0000)	16.8000 (1.0000)	40.0000 (1.0000)	39.6000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)
	0.70	**0.0000** (0.0000)	16.5600 (1.0000)	36.0000 (1.0000)	39.6000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)
	0.75	**0.0000** (0.0000)	9.7600 (1.0000)	36.0000 (1.0000)	39.6000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)
	0.80	**0.0000** (0.0000)	16.0800 (1.0000)	37.3333 (1.0000)	39.6000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)
	0.85	**0.0000** (0.0000)	21.2800 (1.0000)	36.0000 (1.0000)	39.6000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)
	0.90	**0.0000** (0.0000)	27.6000 (1.0000)	37.3333 (1.0000)	39.6000 (1.0000)	40.0000 (1.0000)	39.9200 (1.0000)	40.0000 (1.0000)

Continued on next page

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
	0.95	**0.0000** (0.0000)	15.1200 (1.0000)	37.3333 (1.0000)	39.6000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)	40.0000 (1.0000)
	Average	**0.0000**	15.7067	37.0370	39.6355	40.0000	39.9911	39.9822
Overall Average		**0.0000**	5.8823	18.0897	26.6661	26.8362	26.8223	26.8362
Minimum		**0.0000**	0.0000	0.0000	9.9000	9.9000	9.9000	9.9000
Maximum		**0.0000**	27.6000	40.0000	39.8400	40.0000	40.0000	40.0000
Std		**0.0000**	8.2220	16.0287	10.9183	11.0720	11.0668	11.0651
Win/Equal/Loss		**15/11/0**	0/9/15	0/8/18	0/0/26	0/0/26	0/0/26	0/0/26

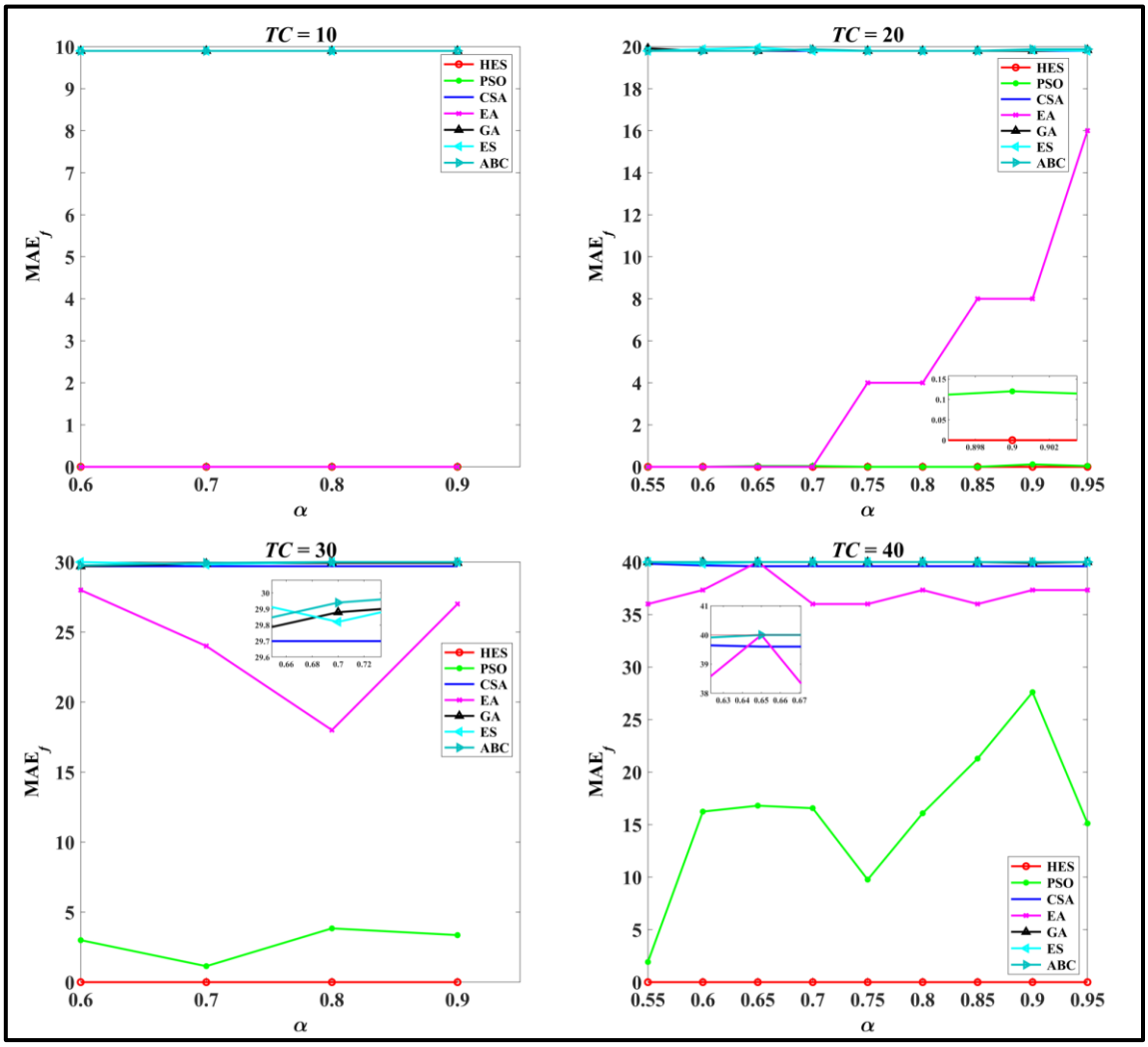

Figure 6. MAE_f values produced per metaheuristic for different α with respect to *TC*.

Importantly, the robustness of the proposed DHNN–MAJ2HES model is further supported by its stable performance across a wide range of α values and increasing *TC* sizes. The persistence of $MAE_f = 0$ under these varying conditions demonstrates that the learning behavior is not sensitive to specific parameter settings. Moreover, MAE_f reflects the inability to reach a fully satisfied interpretation and the number of iterations required to achieve a zero cost function. Therefore, the zero-error results obtained by HES indicate not only solution optimality but also learning efficiency. These

findings confirm that the proposed HES-based learning strategy provides a robust and scalable optimization mechanism for DHNN under multi-objective constraints.

Following the previous fitness phase, all sets of solutions (strings) with improved fitness are brought forward in the diversity phase. The point of evaluating the diversity function is to identify which algorithms can avoid redundancy and promote a diversified neuron state with respect to the initial combination of $\varphi_{MAJ2SAT}$ in the learning phase of the DHNN model. Less MAE_d denotes the more diverse of the learning phase of the DHNN–MAJ2SAT model. As may be seen from the results of $TC = 10$ in Figure 7, all tested metaheuristics achieve optimal $MAE_d = 0$. It should be noticed that all these metaheuristics have equal and more than 10% diversified neuron states with a lowest dimension space $TC = 10$. In other words, suppose when $\alpha = 0.6$ at $TC = 10$, the number of neurons is equal to 24. Therefore, there exists one set of newly generated solutions that contains at least two neuron states that are different with the initial set of solutions. Additionally, HES, PSO, and CSA depict competitive performances at all α and TC. These three algorithms successfully produce a new set of solutions with more diversified neuron states. The importance of having diversity in the learning phase of the DHNN model is to ensure that the quality of final neuron states is approaching global minima solutions. HES can preserve optimal diversity because the solutions focus on exploring and exploiting the solution space effectively. In [51], the researchers discussed classical PSO, when there is no effort of producing diversity particles (solutions). This surely will result in the swarm falling into suboptimal or local solutions. This justifies that having a diversification set of solutions helps the proposed model obtain high quality results. Furthermore, although CSA shows the suboptimal result for MAE_f in Table 5, and this algorithm successfully gains more than threshold diversity value in the diversity phase. This is due to the fact that CSA is known as the search algorithm with a high iterative process of generate and testing that can increase the diversity of the solutions [52]. The result of the diversity error for the ES algorithm is the worst among all metaheuristics, which can be seen in Figure 7 in each case. ES depicts high MAE_d in each case.

ES exhibits the highest peak of diversity error at $TC = 20$, $TC = 30$, and $TC = 40$. As the decision space increases, the exhaustive nature of ES becomes increasingly impractical, preventing the MAJ2SAT formulation from consistently achieving maximum fitness within the allocated computational time. Consequently, ES tends to repeatedly generate identical neuron states due to its trial-and-error search strategy and the absence of dedicated operators for promoting solution diversity [21]. Unlike metaheuristic-based approaches, ES lacks mechanisms to generate new non-redundant neuron states from previously obtained solutions. As a result, the search process is often prematurely terminated before sufficient exploration of the solution space can be achieved. Therefore, the high diversity error observed for ES in Figure 7 is primarily attributed to an incomplete search caused by computational time constraints at higher TC values, rather than limitations of the underlying logical formulation. This premature termination restricts the number of retrieved neuron states and leads to reduced apparent diversity in the learning phase. The result trend of ES is almost similar to ABC. The error difference between ABC and ES is only 0.2846. ABC suffers from limited colony diversity when tackling an optimization problem [39]. It is worth mentioning that the random feature of ABC, which uses the probability distribution, might become trapped in a suboptimal solution. The characteristics of the operator included in the metaheuristic can help in escaping suboptimal solutions and ensure the diversity of the neuron states attained.

Table 5. MAE_d values obtained per metaheuristic for all *TC*. The bracket indicates the ratio of improvement.

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
10	0.60	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)
	0.70	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)
	0.80	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)
	0.90	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)
	Average	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
20	0.55	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	1.2000 (1.0000)	**0.0000** (0.0000)
	0.60	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	1.9200 (1.0000)
	0.65	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	3.7600 (1.0000)
	0.70	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	0.4000 (1.0000)	**0.0000** (0.0000)
	0.75	**0.0000** (0.0000)	**0.0000** (0.0000)	0.9000 (1.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)
	0.80	**0.0000** (0.0000)	**0.0000** (0.0000)	0.8800 (1.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)
	0.85	**0.0000** (0.0000)	**0.0000** (0.0000)	1.7200 (1.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.0000** (0.0000)
	0.90	**0.0000** (0.0000)	**0.0000** (0.0000)	1.6800 (1.0000)	**0.0000** (0.0000)	0.8000 (1.0000)	**0.0000** (0.0000)	0.8400 (1.0000)
	0.95	**0.0000** (0.0000)	**0.0000** (0.0000)	3.2800 (1.0000)	**0.0000** (0.0000)	1.6400 (1.0000)	0.4000 (1.0000)	**0.0000** (0.0000)
	Average	**0.0000**	**0.0000**	0.9400	**0.0000**	0.2711	0.2222	0.7244
30	0.60	**0.0000** (0.0000)	**0.0000** (0.0000)	5.7600 (1.0000)	**0.0000** (0.0000)	0.6000 (1.0000)	**0.0000** (1.0000)	7.2000 (1.0000)
	0.70	**0.0000** (0.0000)	**0.0000** (0.0000)	4.1400 (1.0000)	**0.0000** (0.0000)	2.4000 (1.0000)	1.8000 (1.0000)	2.7600 (1.0000)
	0.80	**0.0000** (0.0000)	**0.0000** (0.0000)	3.9600 (1.0000)	**0.0000** (0.0000)	6.6000 (1.0000)	2.4000 (1.0000)	6.6000 (1.0000)
	0.90	**0.0000** (0.0000)	**0.0000** (0.0000)	5.0400 (1.0000)	**0.0000** (0.0000)	6.3000 (1.0000)	2.4000 (1.0000)	6.3000 (1.0000)
	Average	**0.0000**	**0.0000**	4.7250	**0.0000**	3.9750	1.6500	5.7150
40	0.55	**0.0000** (0.0000)	**0.0000** (0.0000)	7.8400 (1.0000)	**0.0000** (0.0000)	9.8000 (1.0000)	9.8000 (1.0000)	9.8000 (1.0000)
	0.60	**0.0000** (0.0000)	**0.0000** (0.0000)	7.6800 (1.0000)	**0.0000** (0.0000)	9.6000 (1.0000)	9.6000 (1.0000)	5.7600 (1.0000)
	0.65	**0.0000** (0.0000)	**0.0000** (0.0000)	9.4000 (1.0000)	**0.0000** (0.0000)	9.4000 (1.0000)	9.4000 (1.0000)	9.4000 (1.0000)
	0.70	**0.0000** (0.0000)	**0.0000** (0.0000)	7.3600 (1.0000)	**0.0000** (0.0000)	9.2000 (1.0000)	9.2000 (1.0000)	9.2000 (1.0000)
	0.75	**0.0000** (0.0000)	**0.0000** (0.0000)	7.2000 (1.0000)	**0.0000** (0.0000)	9.0000 (1.0000)	8.8000 (1.0000)	9.2000 (1.0000)
	0.80	**0.0000** (0.0000)	**0.0000** (0.0000)	7.0400 (1.0000)	**0.0000** (0.0000)	8.8000 (1.0000)	8.6000 (1.0000)	8.8000 (1.0000)
	0.85	**0.0000** (0.0000)	**0.0000** (0.0000)	6.8800 (1.0000)	**0.0000** (0.0000)	8.6000 (1.0000)	8.4000 (1.0000)	8.6000 (1.0000)
	0.90	**0.0000** (0.0000)	**0.0000** (0.0000)	6.7200 (1.0000)	**0.0000** (0.0000)	8.4000 (1.0000)	3.2000 (1.0000)	8.4000 (1.0000)

Continued on next page

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
	0.95	**0.0000** (0.0000)	**0.0000** (0.0000)	4.9200 (1.0000)	**0.0000** (0.0000)	8.2000 (1.0000)	8.2000 (1.0000)	8.2000 (1.0000)
	Average	**0.0000**	**0.0000**	7.2267	**0.0000**	9.0000	8.3556	8.5956
Overall Average		**0.0000**	**0.0000**	3.5538	**0.0000**	3.8208	3.2231	4.1054
Minimum		**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Maximum		**0.0000**	**0.0000**	9.4000	**0.0000**	9.8000	9.8000	9.8000
Std		**0.0000**	**0.0000**	3.1758	**0.0000**	4.1350	3.9568	3.9178
Win/Equal/Loss		**0/26/0**	**0/26/0**	0/8/18	**0/26/0**	0/11/15	0/10/16	0/10/16

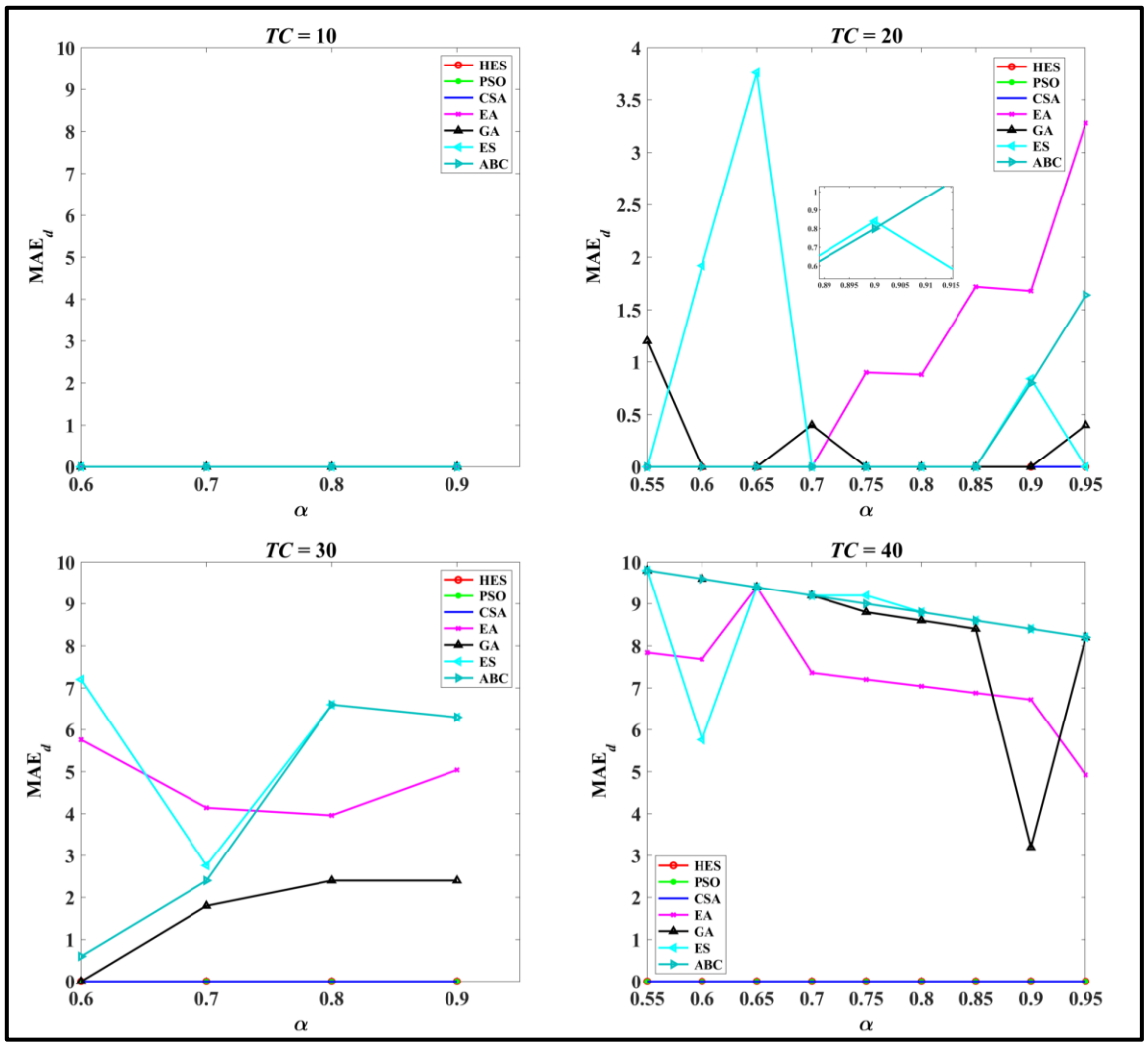

Figure 7. MAE_d values produced per metaheuristic for different α with respect to *TC*.

The number of power strings obtained in each scenario per metaheuristic is depicted in Figure 8. Table 6 shows the average number of power strings completed during the experiment. The number of power strings is similar to the number of CAM that DHNN can produce. In this simulation, a set of solutions that satisfy maximum fitness and maximum diversity with more top positive states is considered a power string and will be stored in CAM. It is revealed from the figure that HES constantly achieves a maximum number of power strings and covers the solution search space effectively. One major reason behind this is that $\varphi_{MAJ2SAT}$ has a good compatibility with the HES algorithm. The inclusion of second and third order logic in $\varphi_{MAJ2SAT}$ gives a high probability of satisfied

interpretation compared to RAN2SAT that consists of first and second order logic [12]. The full utilization of second and third order logic in the proposed model can maximize the production of power strings. The work in [19] neglects the involvement of mutating the first order logic, creating bias in the model. Other than that, PSO portrays competitive performance as HES particularly at $TC = 20$ and $TC = 30$. From $TC = 40$, based on Figure 8, we can see that the number of power strings achieved by PSO decreases as α increases. The average difference between PSO and HES is only 20 power strings. As the dimension of space increases, PSO tends to be stuck in local minima solutions. After PSO, CSA ranks as third best algorithm in producing power strings. Moreover, GA and ABC rank as the fourth best algorithms in the learning phase of DHNN, followed by ES and EA, which are the worst among all. From the observation, we can see that the performance of swarm intelligence and evolutionary algorithms give good performance compared to the heuristic and socio–political algorithm. According to the researchers in [53], who assessed the performance between swarm intelligence and evolutionary algorithms, they concluded that both types of metaheuristics perform well when dealing with the most problems. This proves that these metaheuristics rank as top metaheuristics in solving multi–objective optimization problems. It has been pointed out that the mutation operator helps avoid poor performance [54,55]. Thus, our findings of HES, which implement intelligent mutation, is evidenced to be the best metaheuristic among all benchmark metaheuristics since it is inspired from GA.

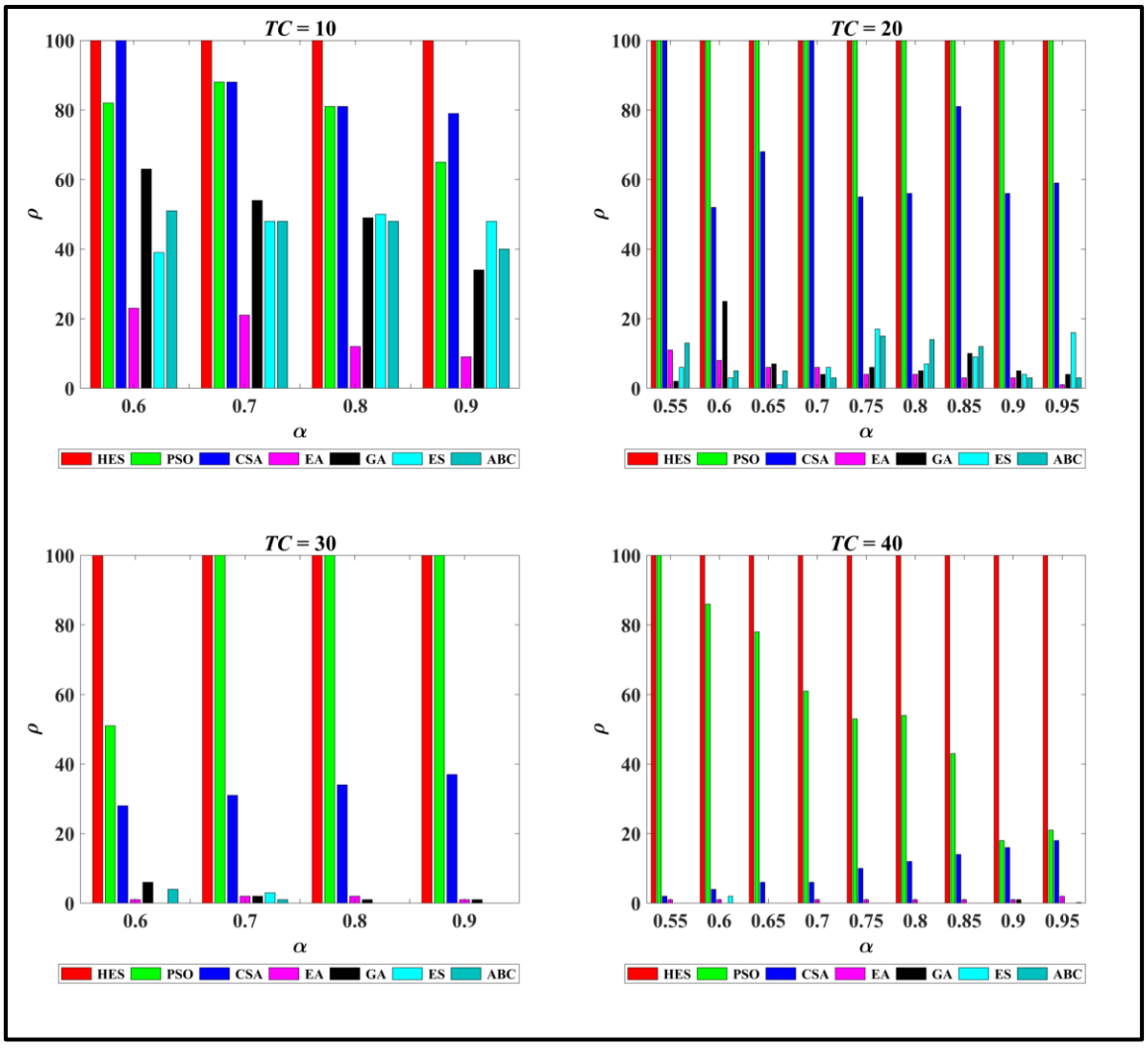

Figure 8. Number of power strings produced per metaheuristic for different α with respect to *TC*.

Table 6. Average number of power strings obtained per metaheuristic for all *TCs*. The bracket indicates the ratio of improvement. ** indicates a value is not available due to no power strings achieved.

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
10	0.60	**100.00** (0.0000)	82.00 (0.22)	23.00 (3.35)	**100.00** (0.00)	51.00 (0.96)	63.00 (0.59)	39.00 (1.56)
	0.70	**100.00** (0.00)	88.00 (0.14)	21.00 (3.76)	88.00 (0.14)	48.00 (1.08)	54.00 (0.85)	48.00 (1.08)
	0.80	**100.00** (0.00)	81.00 (0.23)	12.00 (7.33)	81.00 (0.23)	48.00 (1.08)	49.00 (1.04)	50.00 (1.00)
	0.90	**100.00** (0.00)	65.00 (0.54)	9.00 (10.11)	79.00 (0.27)	40.00 (1.50)	34.00 (1.94)	48.00 (1.08)
	Average	**100.00**	79.00	16.25	87.00	46.75	50.00	46.25
20	0.55	**100.00** (0.00)	**100.00** (0.00)	11.00 (8.09)	**100.00** (0.00)	13.00 (6.69)	2.00 (49.00)	6.00 (15.67)
	0.60	**100.00** (0.00)	**100.00** (0.00)	8.00 (11.50)	52.00 (0.92)	5.00 (19.00)	25.00 (3.00)	3.00 (32.33)
	0.65	**100.00** (0.00)	**100.00** (0.00)	6.00 (15.67)	68.00 (0.47)	5.00 (19.00)	7.00 (13.29)	1.00 (99.00)
	0.70	**100.00** (0.00)	**100.00** (0.00)	6.00 (15.67)	**100.00** (0.00)	3.00 (32.33)	4.00 (24.00)	6.00 (15.67)
	0.75	**100.00** (0.00)	**100.00** (0.00)	4.00 (24.00)	55.00 (0.82)	15.00 (5.67)	6.00 (15.67)	17.00 (4.88)
	0.80	**100.00** (0.00)	**100.00** (0.00)	4.00 (24.00)	56.00 (0.79)	14.00 (6.14)	5.00 (19.00)	7.00 (13.29)
	0.85	**100.00** (0.00)	**100.00** (0.00)	3.00 (32.33)	81.00 (0.23)	12.00 (7.33)	10.00 (9.00)	9.00 (10.11)
	0.90	**100.00** (0.00)	**100.00** (0.00)	3.00 (32.33)	56.00 (0.79)	3.00 (32.33)	5.00 (19.00)	4.00 (24.00)
	0.95	**100.00** (0.00)	**100.00** (0.00)	1.00 (99.00)	59.00 (0.69)	3.00 (32.33)	4.00 (24.00)	16.00 (5.25)
	Average	**100.00**	**100.00**	5.11	69.67	8.11	7.56	7.67
30	0.60	**100.00** (0.00)	51.00 (0.96)	1.00 (99.00)	28.00 (2.57)	4.00 (24.00)	6.00 (15.67)	0.00 (**)
	0.70	**100.00** (0.00)	**100.00** (0.00)	2.00 (49.00)	31.00 (2.23)	1.00 (99.00)	2.00 (49.00)	3.00 (32.33)
	0.80	**100.00** (0.00)	**100.00** (0.00)	2.00 (49.00)	34.00 (1.94)	0.00 (**)	1.00 (99.00)	0.00 (**)
	0.90	**100.00** (0.00)	**100.00** (0.00)	1.00 (99.00)	37.00 (1.70)	0.00 (**)	1.00 (99.00)	0.00 (**)
	Average	**100.00**	87.75	1.50	32.50	1.25	2.50	0.75
40	0.55	**100.00** (0.00)	**100.00** (0.00)	1.00 (99.00)	2.00 (49.00)	0.00 (**)	0.00 (**)	0.00 (**)
	0.60	**100.00** (0.00)	86.00 (0.16)	1.00 (99.00)	4.00 (24.00)	0.00 (**)	0.00 (**)	2.00 (49.00)
	0.65	**100.00** (0.00)	78.00 (0.28)	0.00 (**)	6.00 (15.67)	0.00 (**)	0.00 (**)	0.00 (**)
	0.70	**100.00** (0.00)	61.00 (0.64)	1.00 (99.00)	6.00 (15.67)	0.00 (**)	0.00 (**)	0.00 (**)
	0.75	**100.00** (0.00)	53.00 (0.89)	1.00 (99.00)	10.00 (9.00)	0.00 (**)	0.00 (**)	0.00 (**)
	0.80	**100.00** (0.00)	54.00 (0.85)	1.00 (99.00)	12.00 (7.33)	0.00 (**)	0.00 (**)	0.00 (**)
	0.85	**100.00** (0.00)	43.00 (1.33)	1.00 (99.00)	14.00 (6.14)	0.00 (**)	0.00 (**)	0.00 (**)

Continued on next page

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
	0.90	**100.00** (0.00)	18.00 (4.56)	1.00 (99.00)	16.00 (5.25)	0.00 (**)	1.00 (99.00)	0.00 (**)
	0.95	**100.00** (0.00)	21.00 (3.76)	2.00 (49.00)	18.00 (4.56)	0.00 (**)	0.00 (**)	0.00 (**)
	Average	**100.00**	57.11	1.00	9.78	0.00	0.11	0.22
Overall Average		**100.00**	80.04	4.85	45.88	10.19	10.73	9.96
Minimum		**100.00**	18.00	0.00	2.00	0.00	0.00	0.00
Maximum		**100.00**	**100.00**	23.00	**100.00**	51.00	63.00	50.00
Std		**0.00**	25.45	5.93	32.73	16.33	17.96	16.21
Win/Equal/Loss		**12/14/0**	0/13/13	0/0/26	0/3/23	0/0/26	0/0/26	0/0/26

5.2. Analysis of the testing phase

To test the robustness of the DHNN–MAJ2HES model in the testing phase, it is executed for three evaluation metrics with six benchmark metaheuristics. In the testing phase, the quality of final neuron states is discussed thoroughly. In producing high quality final neuron states, it is important to analyze synaptic weight management and the number of global solutions accomplished by the model. Therefore, the results for the ratio of global minima solution, ratio of total neuron variations, and total of Jaccard index are represented and debated in this subsection. Figure 9 and Table 7 show the results for the ratio of global solutions achieved by each metaheuristic. The number of power strings achieved by the proposed model is highly related to the ratio of global minimum solutions $\left(R_G\right)$. In this experiment, five optimal power strings are selected to store in respective CAM. Specifically, the information in five CAM are analyzed in the testing phase. Note that the characteristics of the optimal power string are that it must comprise maximum fitness, maximum diversity, top positive neuron states, and be unique. Every R_G achieved by the proposed model is denoted based on the number of five power strings. In Table 7, $R_G = 1$ implies that all five power strings successfully stored in CAM can produce global minima solutions. In short, all final neuron states achieve optimal solutions. As shown in Figure 9, HES and PSO dominate the search space as both produce maximum R_G. As for CSA, although the number of power strings attain more than five, the global minimum solutions are unable to give an optimal value. This is because neuron oscillations occur during the testing phase. It is likely that neuron oscillation happens because of the high number of clauses [56]. Therefore, HTAF is needed to reduce the neuron oscillations [57]. from Figure 9 and Table 7, we can analyze that EA, GA, ES, and ABC start showing suboptimal R_G when $TC = 30$ and $TC = 40$. ABC gives lowest $R_G = 0$ throughout $0.55 \le \alpha \le 0.95$ at $TC = 40$. This trend is quite similar to ES, which gets $R_G = 0.15$ on average at $TC = 30$. The justification to this trend that converge to suboptimal R_G is due to poor synaptic weight management. The concept of this multi–objective DHNN is if there exists a model that cannot achieve desired 5–CAM, then the synaptic weight will be randomized. Randomized synaptic weight is well defined as wrong a synaptic weight is assigned to the respective clause in $\varphi_{MAJ2SAT}$. For example, when $A_1^{(2)} = \left(a_1 \vee b_1\right)$, the optimal synaptic weight should be $\left\{W_{a_1}^{(2)}, W_{b_1}^{(2)}, W_{a_1 b_1}^{(2)}\right\} = \{0.25, 0.25, -0.25\}$. If synaptic weight achieved by the proposed model for the stated clause is different from the optimal synaptic weight value, then it is signified that a randomized synaptic weight is being assigned. This is applicable only to the model that cannot achieve

5–CAM in the learning phase. To clarify, this penalty concept has been applied to ensure that the model will not execute the simulation. Additionally, the calculation of the Wan Abdullah method is crucial to confirm the optimal synaptic weight management. Therefore, the producibility of maximum 5–CAM in the learning phase is crucial to guarantee that high global minimum solutions are being produced.

Table 7. Average ratio of global solutions obtained per metaheuristic for all *TCs*. The bracket indicates the ratio of improvement. ** indicates the value is not available due to no global minima solutions are achieved.

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
10	0.60	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)
	0.70	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)
	0.80	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)
	0.90	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)
	Average	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
20	0.55	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	0.4000 (1.5)	**1.0000** (0.0000)
	0.60	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	0.6000 (0.6667)
	0.65	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)
	0.70	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	0.8000 (0.2500)	**1.0000** (0.0000)
	0.75	**1.0000** (0.0000)	**1.0000** (0.0000)	0.8000 (0.2500)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)
	0.80	**1.0000** (0.0000)	**1.0000** (0.0000)	0.8000 (0.2500)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)
	0.85	**1.0000** (0.0000)	**1.0000** (0.0000)	0.6000 (0.6667)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)	**1.0000** (0.0000)
	0.90	**1.0000** (0.0000)	**1.0000** (0.0000)	0.6000 (0.6667)	**1.0000** (0.0000)	0.6000 (0.6667)	**1.0000** (0.0000)	0.8000 (0.2500)
	0.95	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.6000 (0.6667)	0.8000 (0.2500)	**1.0000** (0.0000)
	Average	**1.0000**	**1.0000**	0.7778	**1.0000**	0.9111	0.8889	0.8444
30	0.60	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.8000 (0.2500)	**1.0000** (0.0000)	0.0000 (**)
	0.70	**1.0000** (0.0000)	**1.0000** (0.0000)	0.4000 (1.5000)	**1.0000** (0.0000)	0.2000 (4.0000)	0.4000 (1.5000)	0.6000 (0.6667)
	0.80	**1.0000** (0.0000)	**1.0000** (0.0000)	0.4000 (1.5000)	**1.0000** (0.0000)	0.0000 (**)	0.2000 (4.0000)	0.0000 (**)
	0.90	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.2000 (4.0000)	0.0000 (**)
	Average	**1.0000**	**1.0000**	0.3000	**1.0000**	0.2500	0.4500	0.1500
40	0.55	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	0.4000 (1.5000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.60	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	0.8000 (0.2500)	0.0000 (**)	0.0000 (**)	0.4000 (1.5000)
	0.65	**1.0000** (0.0000)	**1.0000** (0.0000)	0.0000 (**)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.70	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)

Continued on next page

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
	0.75	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.80	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.85	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.90	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.2000 (4.0000)	0.0000 (**)
	0.95	**1.0000** (0.0000)	**1.0000** (0.0000)	0.4000 (1.5000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	Average	**1.0000**	**1.0000**	0.2000	0.9111	0.0000	0.0222	0.0444
Overall Average		**1.0000**	**1.0000**	0.5385	0.9692	0.5077	0.5385	0.4846
Minimum		**1.0000**	**1.0000**	0.0000	0.4000	0.0000	0.0000	0.0000
Maximum		**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Std		**0.0000**	**0.0000**	0.3585	0.1202	0.4682	0.4464	0.4580
Win/Equal/Loss		**0/26/0**	**0/26/0**	0/8/18	0/24/2	0/11/15	0/11/15	0/10/16

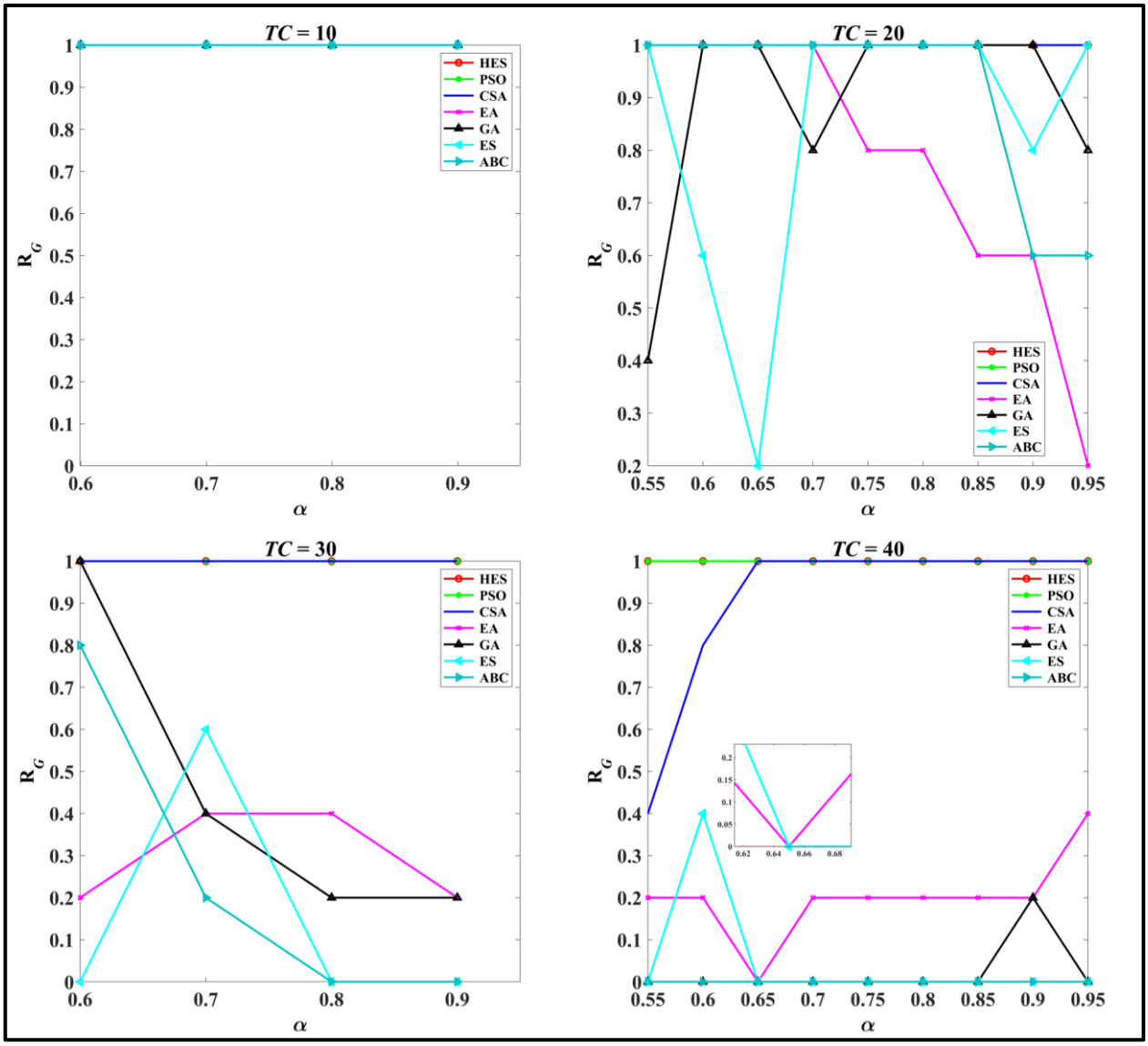

Figure 9. Ratio of global minima solutions per metaheuristic for different α with respect to *TC*.

Another fact on how we can measure the quality of final neuron states attained by the proposed model is through the total logical variations' metric $\left(R_\tau\right)$. This metric measures how many logical variations produced by the proposed model, with the structure of previous $\varphi_{MAJ2SAT}$, must not be similar to the current $R_\tau = 0$. The maximum number of logical variations is 500, where each power string generates 100 variations since there is 100 number of trials in the testing phase of DHNN. Although, the maximum number of logical variations is 500, the HES algorithm remains the best metaheuristic with 433 logical variations, which is equivalent to $R_\tau = 0.8679$, on average. As expected, Table 8 illustrates that the second–best algorithm is obtained by PSO. PSO outperform HES at $TC = 40$ by 0.018 because of no mutation calculations, which slightly affects the logical variations computation [21]. Furthermore, most fluctuations happen at $TC = 20$ as α keeps increasing. This is likely to happen since there is no optimizer at the testing phase to ensure the smoothness of retrieving final neuron states with global minimum solutions and more logical variations. Moreover, from Figure 10, we can examine that ES is the worst algorithm since it gives $R_\tau = 0$ at $\alpha = 0.8, 0.9$ when $TC = 30$. ES also accomplishes suboptimal $R_\tau = 0$ at $TC = 40$ for all α except when $\alpha \neq 0.6$ that gives $R_\tau = 0.4$. The reason for this suboptimal trend for metaheuristics, which acquire the value of $R_\tau \to 0$, is due to the absence of 5–CAM stored in the testing phase. Moreover, investigating the ratio of logical variations is important since it denotes the flexibility of the DHNN–MAJ2HES model in the testing phase.

As in [21], the range for the Jaccard index (JDSI) is within $(0.5,1)$. A lower JDSI value denotes the optimal results. However, in this paper, the goal of evaluating the Jaccard similarity index ratio $\left(R_{\text{JDSI}}\right)$ is to calculate the number of positive final neuron states in the generated solutions compared to the benchmark neuron states based on the ρ–CAM. Higher R_{JDSI} denotes a high number of positive states achieved by the model. The reason we assess positive states is to verify the flexibility of the DHNN–MAJ2HES model in achieving any condition that has been set. The flexibility of the DHNN–MAJ2HES model is proven since it gives the best value according to the average value of R_{JDSI} based on Table 9. Figure 11 portrays the R_{JDSI} value per metaheuristic per *TC* for all ranges of α. According to $TC = 10$, GA outperforms other models at $\alpha = 0.6, 0.8, 0.9$. Moreover, GA can produce 75.29% positive states compared to HES, which is only 69.73% when confronted with the lowest dimension of $21 \leq NN \leq 24$. As dimension space increases, CSA manages to get the best performance at $TC = 20$. We can observe that there is a competitive performance between CSA (blue line) and HES (red line) during $0.55 \leq \alpha \leq 0.75$ and leaving HES behind after $\alpha = 0.75$. Judging from these two cases, we can state that evolutionary algorithms can produce more positive states. This analysis explains one of the reasons HES algorithms inherit the mutation operator inspired by GA and CSA. Another viewpoint we explore is that many fluctuations happen at the lowest *TC*. Fluctuations are likely to happen due to the neuron oscillations in finding high–quality final neuron states. Furthermore, there is no optimizer or conditions set in the testing phase to reduce neuron oscillations. Moving to the highest, which is $TC = 30, 40$, HES successfully outperforms all benchmark models for all α. The most optimal R_{JDSI} result attained by HES is 72% at $\alpha = 0.6, 0.95$ when $TC = 40$. The performance of HES has proved that this proposed model can maintain a high number of positive states despite there being no modifications made to retain the characteristics of 5–CAM in the testing phase. Having more positive final neuron states in the non–simulated data sets can increase the accuracy of the proposed model in doing data mining [58].

Overall, the DHNN–MAJ2HES model has the top rank in all metrics for both phases. The second rank and third rank belong to the DHNN–MAJ2PSO and DHNN–MAJ2CSA models, respectively. The DHNN–MAJ2EA and DHNN–MAJ2GA model switch the fourth and fifth ranks in the learning and testing phase, respectively. This is followed by DHNN–MAJ2SATES in sixth place, and DHNN–MAJ2ABC is the worst model. Generally, we can validate that the DHNN–MAJ2HES model achieves the best performance for tackling multi–objective optimization problems in both phases of DHNN. We note that the Friedman test is conducted with six degrees of freedom for metrics ρ and R_G. The p–value for the ρ metric is 2.56×10^{-24} with a chi–square of $\chi^2 = 123.824$. Additionally, the p–value for the R_G metric is 5.04×10^{-8} with a chi–square of $\chi^2 = 44.8393$. From these results, the null hypothesis is rejected. Therefore, we can confirm that the DHNN–MAJ2HES model is not similar to other benchmark learning algorithms.

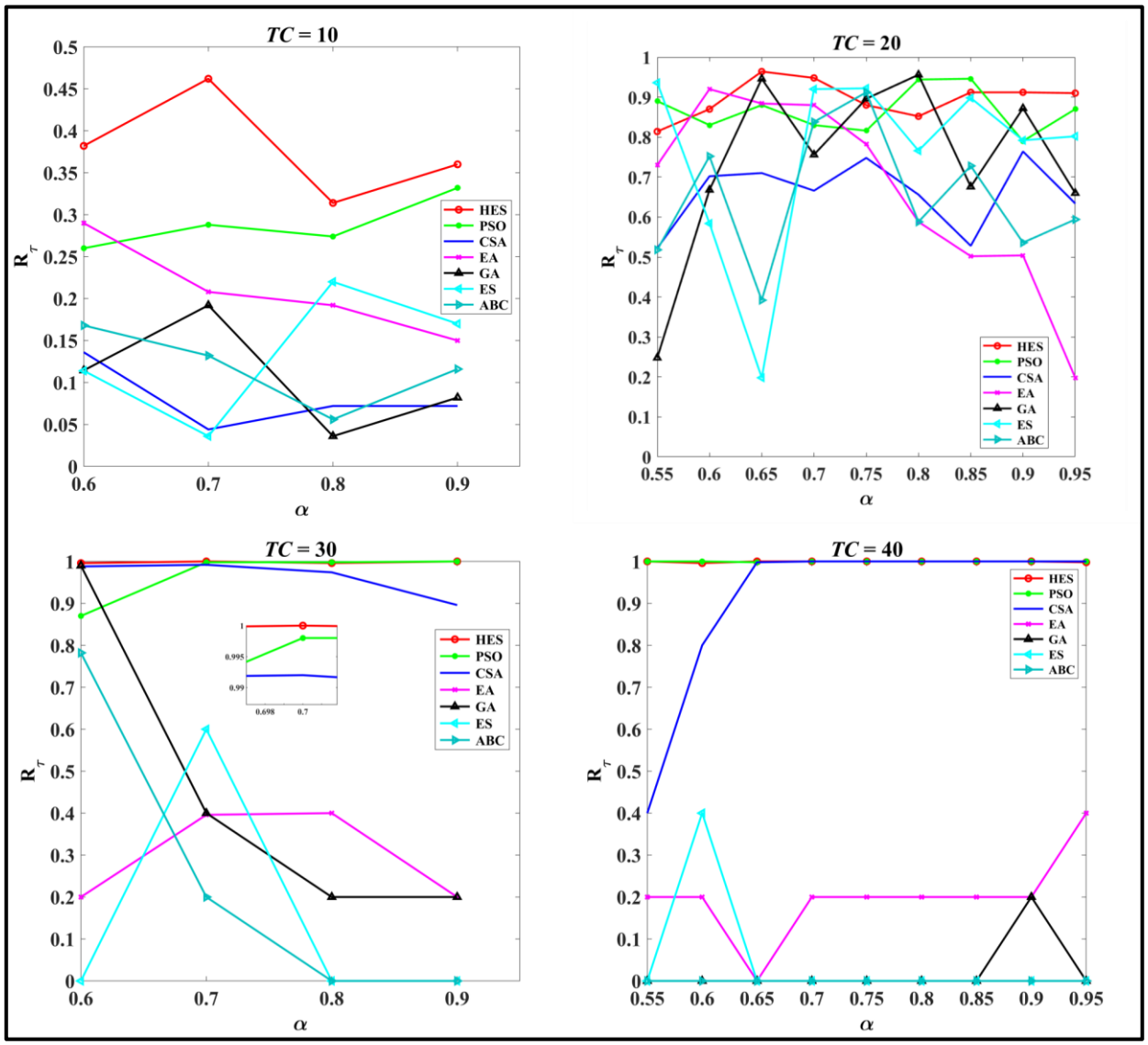

Figure 10. Ratio of total neuron variations per metaheuristic for different α with respect to *TC*.

Table 8. Average ratio of total neuron variations obtained per metaheuristic for all *TCs*. The bracket indicates the ratio of improvement. ** indicates that the value is not available due to no neuron variations being achieved.

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
10	0.60	**0.3820** (0.0000)	0.2600 (0.4692)	0.2900 (0.3172)	0.1360 (1.8088)	0.1680 (1.2738)	0.1144 (2.3392)	0.1140 (2.3509)
	0.70	**0.4620** (0.0000)	0.2880 (0.6042)	0.2080 (1.2212)	0.0440 (9.5000)	0.1320 (2.5000)	0.1920 (1.4063)	0.0360 (11.8333)
	0.80	**0.3140** (0.0000)	0.2740 (0.1460)	0.1920 (0.6354)	0.0720 (3.3611)	0.0560 (4.6071)	0.0360 (7.7222)	0.2200 (0.4273)
	0.90	**0.3600** (0.0000)	0.3320 (0.0843)	0.1500 (1.4000)	0.0720 (4.0000)	0.1160 (2.1035)	0.0820 (3.3902)	0.1700 (1.1177)
	Average	**0.3795**	0.2885	0.2100	0.0810	0.1180	0.1061	0.1350
20	0.55	0.8140 (–0.1499)	0.8900 (0.0517)	0.7300 (0.2822)	0.5220 (0.7931)	0.5180 (0.8070)	0.2480 (2.7742)	**0.9360** (0.0000)
	0.60	0.8700 (–0.0575)	0.8300 (0.1084)	**0.9200** (0.0000)	0.7020 (0.3105)	0.7520 (0.2234)	0.6680 (0.3773)	0.5840 (0.5753)
	0.65	**0.9640** (0.0000)	0.8800 (0.0955)	0.8840 (0.0905)	0.7100 (0.3578)	0.3920 (1.4592)	0.9460 (0.0190)	0.1980 (3.8687)
	0.70	**0.9480** (0.0000)	0.8300 (0.1422)	0.8800 (0.0773)	0.6660 (0.4234)	0.8380 (0.1313)	0.7560 (0.2540)	0.9200 (0.0304)
	0.75	0.8800 (–0.0477)	0.8160 (0.1299)	0.7820 (0.1790)	0.7480 (0.2326)	0.9120 (0.0110)	0.8960 (0.0290)	**0.9220** (0.0000)
	0.80	0.8520 (–0.1221)	0.9440 (0.0127)	0.5880 (0.6259)	0.6560 (0.4573)	0.5880 (0.6259)	**0.9560** (0.0000)	0.7660 (0.2480)
	0.85	0.9120 (–0.0373)	**0.9460** (0.0000)	0.5020 (0.8845)	0.5280 (0.7917)	0.7280 (0.2995)	0.6760 (0.3994)	0.8980 (0.0535)
	0.90	**0.9120** (0.0000)	0.7900 (0.1544)	0.5040 (0.8095)	0.7640 (0.1937)	0.5360 (0.7015)	0.8720 (0.0459)	0.7920 (0.1515)
	0.95	**0.9100** (0.0000)	0.8700 (0.0460)	0.1980 (3.5960)	0.6340 (0.4353)	0.5940 (0.5320)	0.6600 (0.3788)	0.8020 (0.1347)
	Average	**0.8958**	0.8662	0.6653	0.6589	0.6509	0.7420	0.7576
30	0.60	**0.9960** (0.0000)	0.8700 (0.1448)	0.2000 (3.9800)	0.9880 (0.0081)	0.7820 (0.2737)	0.9900 (0.0061)	0.0000 (**)
	0.70	**1.0000** (0.0000)	0.9980 (0.002)	0.3960 (1.5253)	0.9920 (0.0081)	0.2000 (4.0000)	0.4000 (1.5000)	0.6000 (0.6667)
	0.80	0.9960 (–0.0020)	**0.9980** (0.0000)	0.4000 (1.4950)	0.9740 (0.0246)	0.0000 (**)	0.2000 (3.9900)	0.0000 (**)
	0.90	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	0.8960 (0.1161)	0.0000 (**)	0.2000 (4.0000)	0.0000 (**)
	Average	**0.9980**	0.9665	0.2990	0.9625	0.2455	0.4475	0.1500
40	0.55	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	0.4000 (1.5000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.60	0.9960 (–0.0040)	**1.0000** (0.0000)	0.2000 (4.0000)	0.8000 (0.2500)	0.0000 (**)	0.0000 (**)	0.4000 (1.5000)
	0.65	**1.0000** (0.0000)	0.9980 (0.0020)	0.0000 (**)	0.9980 (0.0020)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.70	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.75	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.80	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.85	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)

Continued on next page

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
	0.90	**1.0000** (0.0000)	**1.0000** (0.0000)	0.2000 (4.0000)	**1.0000** (0.0000)	0.0000 (**)	0.2000 (4.0000)	0.0000 (**)
	0.95	0.9980 (–0.0020)	**1.0000** (0.0000)	0.4000 (1.5000)	**1.0000** (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	Average	0.9993	**0.9998**	0.2000	0.9109	0.0000	0.0222	0.0444
Overall Average		**0.8679**	0.8390	0.3778	0.7039	0.2812	0.3497	0.3215
Minimum		**0.3140**	0.2600	0.0000	0.0440	0.0000	0.0000	0.0000
Maximum		**1.0000**	**1.0000**	0.9200	**1.0000**	0.9120	0.9900	0.9360
Std		**0.2160**	0.2446	0.2594	0.3176	0.3219	0.3660	0.3691
Win/Equal/Loss		**11/7/8**	3/8/15	1/0/25	0/6/20	0/0/26	1/0/25	2/0/24

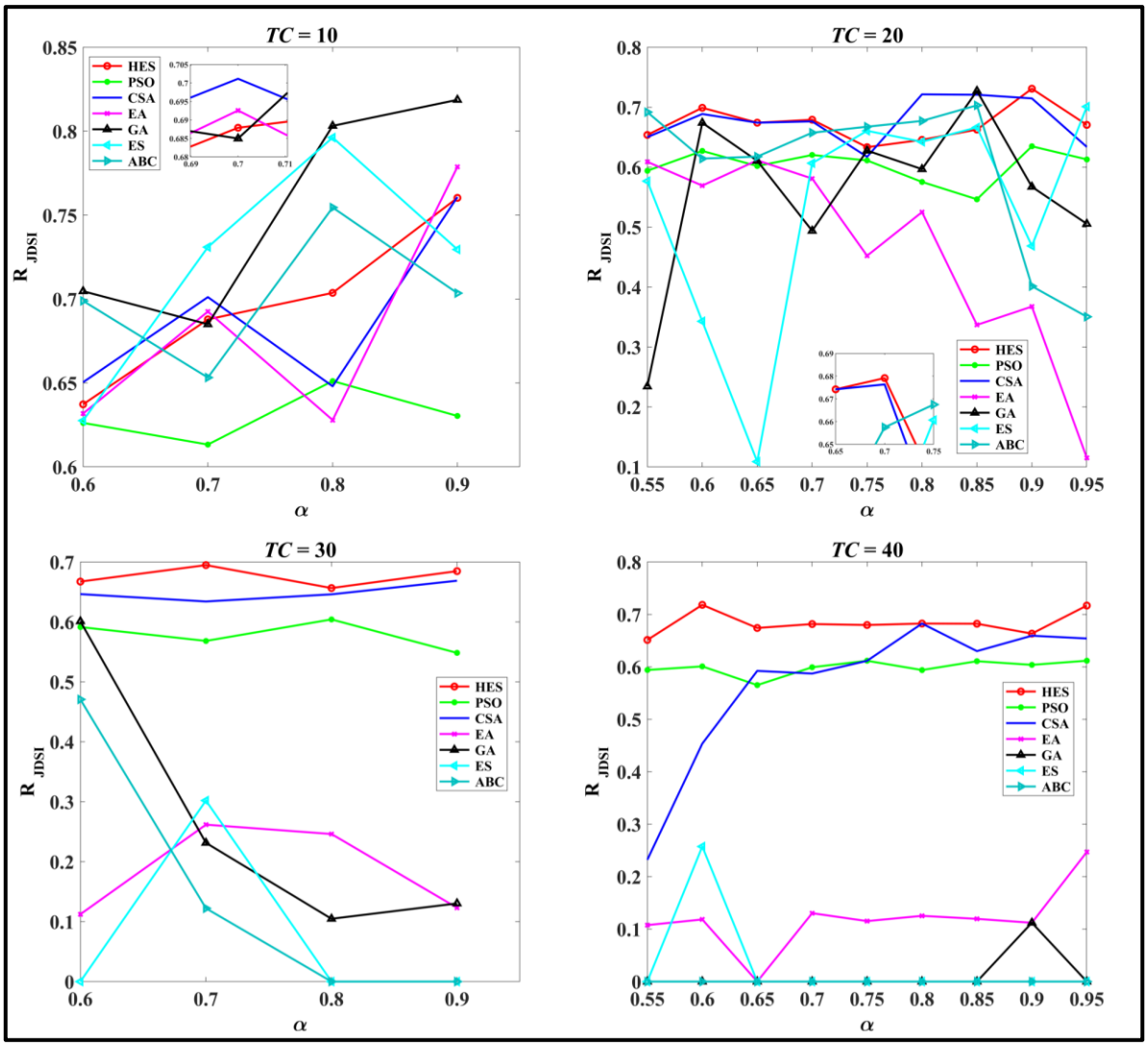

Figure 11. Ratio of Jaccard index per metaheuristic for different α with respect to *TC*.

Table 9. Average ratio of Jaccard index obtained per metaheuristic for all *TCs*. The bracket indicates the ratio of improvement. ** indicates that the value is not available due to no Jaccard index being obtained.

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
10	0.60	0.6373 (–0.1058)	0.6262 (0.1253)	0.6318 (0.1154)	0.6504 (0.0834)	0.6989 (0.0083)	**0.7047** (0.0000)	0.6276 (0.1228)
	0.70	0.6879 (–0.0626)	0.6133 (0.1918)	0.6926 (0.0554)	0.7012 (0.0425)	0.6532 (0.1190)	0.6850 (0.0671)	**0.7309** (0.0000)
	0.80	0.7037 (–0.1414)	0.6511 (0.2335)	0.6279 (0.2792)	0.6480 (0.2394)	0.7545 (0.0645)	**0.8032** (0.0000)	0.7962 (0.0087)
	0.90	0.7603 (–0.0768)	0.6304 (0.2988)	0.7788 (0.0513)	0.7608 (0.0761)	0.7035 (0.1637)	**0.8187** (0.0000)	0.7295 (0.1222)
	Average	0.6973	0.6303	0.6828	0.6901	0.7025	**0.7529**	0.7211
20	0.55	0.6538 (–0.0581)	0.5940 (0.1645)	0.6089 (0.1361)	0.6501 (0.0641)	**0.6918** (0.0000)	0.2342 (1.9531)	0.5767 (0.1995)
	0.60	**0.6991** (0.0000)	0.6271 (0.1148)	0.5692 (0.2282)	0.6887 (0.0151)	0.6143 (0.1380)	0.6740 (0.0372)	0.3429 (1.0388)
	0.65	**0.6742** (0.0000)	0.6019 (0.1201)	0.6115 (0.1026)	0.6742 (0.0000)	0.6176 (0.0916)	0.6106 (0.1042)	0.1087 (5.1996)
	0.70	**0.6792** (0.0000)	0.6203 (0.0949)	0.5810 (0.1689)	0.6764 (0.0041)	0.6575 (0.0330)	0.4938 (0.3753)	0.6066 (0.1196)
	0.75	0.6334 (–0.0539)	0.6112 (0.0921)	0.4522 (0.4761)	0.6172 (0.0814)	**0.6675** (0.0000)	0.6277 (0.0634)	0.6607 (0.0103)
	0.80	0.6456 (–0.1176)	0.5754 (0.2540)	0.5251 (0.3741)	**0.7215** (0.0000)	0.6772 (0.0654)	0.5967 (0.2092)	0.6424 (0.1232)
	0.85	0.6627 (–0.0972)	0.5464 (0.3307)	0.3369 (1.1584)	0.7208 (0.0088)	0.7032 (0.0341)	**0.7271** (0.0000)	0.6664 (0.0912)
	0.90	**0.7309** (0.0000)	0.6348 (0.1513)	0.3674 (0.9893)	0.7147 (0.0226)	0.4014 (0.8207)	0.5673 (0.2885)	0.4686 (0.5598)
	0.95	0.6706 (–0.0455)	0.6129 (0.1437)	0.1150 (5.0951)	0.6338 (0.1061)	0.3505 (1.0003)	0.5055 (0.3868)	**0.7010** (0.0000)
	Average	0.6722	0.6027	0.4630	**0.6775**	0.5979	0.5597	0.5304
30	0.60	**0.6672** (0.0000)	0.5914 (0.1283)	0.1123 (4.9435)	0.6464 (0.0323)	0.4707 (0.4177)	0.6009 (0.1104)	0.0000 (**)
	0.70	**0.6948** (0.0000)	0.5683 (0.2226)	0.2617 (1.6547)	0.6340 (0.0958)	0.1222 (4.6841)	0.2314 (2.0022)	0.3020 (1.3009)
	0.80	**0.6565** (0.0000)	0.6041 (0.0866)	0.2462 (1.6669)	0.6460 (0.0162)	0.0000 (**)	0.1047 (5.2693)	0.0000 (**)
	0.90	**0.6847** (0.0000)	0.5483 (0.2489)	0.1231 (4.5643)	0.6687 (0.0240)	0.0000 (**)	0.1304 (4.2526)	0.0000 (**)
	Average	**0.6758**	0.5780	0.1858	0.6488	0.1482	0.2669	0.0755
40	0.55	**0.6513** (0.0000)	0.5940 (0.0965)	0.1076 (5.0516)	0.2322 (1.8047)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.60	**0.7182** (0.0000)	0.6009 (0.1952)	0.1184 (5.0675)	0.4536 (0.5833)	0.0000 (**)	0.0000 (**)	0.2575 (1.7896)
	0.65	**0.6741** (0.0000)	0.5652 (0.1927)	0.0000 (**)	0.5923 (0.1381)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.70	**0.6816** (0.0000)	0.5993 (0.1374)	0.1303 (4.2293)	0.5872 (0.1607)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.75	**0.6798** (0.0000)	0.6116 (0.1114)	0.1153 (4.8982)	0.6115 (0.1116)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.80	**0.6825** (0.0000)	0.5939 (0.1493)	0.1253 (4.4469)	0.6825 (0.0000)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	0.85	**0.6822** (0.0000)	0.6108 (0.1168)	0.1195 (4.7078)	0.6300 (0.0829)	0.0000 (**)	0.0000 (**)	0.0000 (**)

Continued on next page

TC	α	HES	PSO	EA	CSA	ABC	GA	ES
	0.90	**0.6634** (0.0000)	0.6038 (0.0986)	0.1120 (4.9216)	0.6594 (0.0060)	0.0000 (**)	0.1122 (4.9143)	0.0000 (**)
	0.95	**0.7168** (0.0000)	0.6117 (0.1718)	0.2470 (1.9018)	0.6541 (0.0959)	0.0000 (**)	0.0000 (**)	0.0000 (**)
	Average	**0.6833**	0.5990	0.1195	0.5670	0.0000	0.0125	0.0286
Overall Average		**0.6804**	0.6019	0.3353	0.6368	0.3378	0.3549	0.3161
Minimum		**0.6334**	0.5464	0.0000	0.2322	0.0000	0.0000	0.0000
Maximum		0.7603	0.6511	0.7788	0.7608	0.7545	**0.8187**	0.7962
Std		0.0286	**0.0248**	0.2338	0.0986	0.3163	0.3065	0.3094
Win/Equal/Loss		**15/2/9**	0/0/26	0/0/26	1/2/23	2/0/24	4/0/22	2/0/24

The superior performance of the proposed DHNN–MAJ2HES model can be attributed to the synergistic interaction between the MAJ2SAT logical formulation and the intelligent mutation mechanism embedded in the HES. The majority-based structure of MAJ2SAT introduces redundancy in clause representation, which increases the probability of getting satisfied interpretations and stabilizes convergence toward global minimum. This effect is particularly evident in the improved global minima ratio and enhanced neuron state diversity, as majority voting reduces sensitivity to individual literal fluctuations. In addition, the intelligent mutation operator plays a critical role by selectively modifying neuron states associated with unsatisfied clauses, rather than performing random or population-wide updates. This targeted mutation strategy preserves high-quality partial solutions while efficiently correcting local inconsistencies, thereby preventing premature convergence and repetitive neuron state patterns commonly observed in conventional metaheuristics. Nevertheless, the proposed model may exhibit reduced adaptability in scenarios involving extremely large neuron counts or higher-order logical dependencies, where the current MAJ2SAT formulation and limited mutation operators may constrain exploration. These observations suggest that while DHNN–MAJ2HES is highly effective for structured multi-objective optimization, further extensions incorporating adaptive operators or higher-order satisfiability rules could enhance its scalability and generality.

6. Conclusions

In this study, we addressed several limitations observed in conventional discrete Hopfield neural networks, particularly their reliance on single-objective optimization, limited search space exploration, and reduced storage capacity as network complexity increases. Previous DHNN learning approaches often suffer from repetitive neuron states, suboptimal convergence, and insufficient diversity, which restrict their effectiveness in solving complex optimization problems.

To overcome these challenges, we proposed a novel DHNN–MAJ2HES model that integrates Major 2 Satisfiability logic with a HES algorithm enhanced by an intelligent mutation operator. The proposed MAJ2SAT formulation, incorporating 2SAT and 3SAT structures, was successfully embedded into the DHNN framework to improve cost function minimization. In addition, the intelligent mutation mechanism significantly enhanced the learning phase by expanding the search space and improving exploration–exploitation balance. Experimental results demonstrated that the proposed DHNN–MAJ2HES model consistently outperformed six state-of-the-art learning algorithms across all evaluation metrics in learning and testing phases. Notably, the model achieved zero cost function with higher neuron state diversity and increased storage capacity, expanding from a single CAM to multiple CAM configurations. These findings confirm the effectiveness of hybrid optimization strategies in improving DHNN learning performance under multi-objective settings.

Despite the encouraging results, the proposed model presents several limitations. The HES framework employs a limited set of search operators, which may restrict adaptability in highly complex or large-scale problem instances. In addition, the logical formulation is confined to Major 2 Satisfiability, which may limit representational expressiveness for higher-order logical dependencies. In future work, we will focus on incorporating additional adaptive search operators to further balance exploration and exploitation, such as advanced initialization strategies inspired by particle swarm optimization [59]. Moreover, extending the proposed framework to higher-order logical rules, such as Major 3 Satisfiability, is expected to further enhance solution diversity and optimization capability in multi-objective DHNN models [19].

Author contributions

Alyaa Alway: Conceptualization, Methodology, Software, Project Administration; Mohd Shareduwan Mohd Kasihmuddin: Supervision, Writing– Original draft preparation; Mohd. Asyraf Mansor: Visualization, Investigation; Nur Ezlin Zamri: Formal analysis; Guo Yueling: Data curation; Siti Zulaikha Mohd Jamaludin: Writing– Reviewing and Editing; Azleena Mohd Kassim: Validation.

Use of Generative-AI tools declaration

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

Acknowledgments

We would like to thank all researchers from Artificial Intelligence Research Development Group (AIRDG) for their continuous support. This work was supported by Universiti Sains Malaysia, Bridging Grant with Project No: R501-LR-RND003-0000002087-0000. All authors gratefully acknowledge the support from Universiti Sains Malaysia.

Conflict of interest

The authors declare that they have no competing interests.

References

1. C. Zhou, X. Zeng, H. Jiang, L. Han, A generalized bipolar auto-associative memory model based on discrete recurrent neural networks, Neurocomputing, **162** (2015), 201–208. https://doi.org/10.1016/j.neucom.2015.03.052
2. J. J. Hopfield, D. W. Tank, "Neural" computation of decisions in optimization problems, Biol. Cybern., **52** (1985), 141–152. https://doi.org/10.1007/BF00339943
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. S. Sathasivam, Upgrading logic programming in Hopfield network, Sains Malays., **39** (2010), 115–118.

5. M. S. M. Kasihmuddin, M. A. Mansor, S. Sathasivam, Hybrid genetic algorithm in the Hopfield network for logic satisfiability problem, Pertanika J. Sci. Technol., **25** (2017), 139–152.
6. M. A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Artificial immune system paradigm in the Hopfield network for 3-satisfiability problem, Pertanika J. Sci. Technol., **25** (2017), 1173–1188.
7. M. S. M. Kasihmuddin, M. A. Mansor, S. Sathasivam, Discrete Hopfield neural network in restricted maximum k-satisfiability logic programming, Sains Malays., **47** (2018), 1327–1335. http://dx.doi.org/10.17576/jsm-2018-4706-30
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). https://doi.org/10.3390/math7111133
9. F. Yu, S. He, W. Yao, S. Cai, Q. Xu, Bursting firings in memristive Hopfield neural network with image encryption and hardware implementation, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., **44** (2025), 4564–4576. https://doi.org/10.1109/TCAD.2025.3567878
10. M. Mendler, M. Döhler, C. E. Ventura, Sensor placement with optimal damage detectability for statistical damage detection, Mech. Syst. Signal Process., **170** (2022), 108767. https://doi.org/10.1016/j.ymssp.2021.108767
11. N. E. Zamri, M. A. Mansor, M. S. M. Kasihmuddin, A. Alway, S. Z. M. Jamaludin, S. A. Alzaeemi, Amazon employees resources access data extraction via clonal selection algorithm and logic mining approach, Entropy, **22** (2020), 596. https://doi.org/10.3390/e22060596
12. 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
13. N. E. Zamri, S. A. Azhar, M. A. Mansor, A. Alway, M. S. M. Kasihmuddin, Weighted random k satisfiability for k = 1, 2 (r2sat) in discrete Hopfield neural network, Appl. Soft Comput., **126** (2022), 109312. https://doi.org/10.1016/j.asoc.2022.109312
14. S. A. Karim, N. E. Zamri, A. Alway, M. S. M. Kasihmuddin, A. I. M. Ismail, M. A. Mansor, et al., Random satisfiability: A higher-order logical approach in discrete Hopfield Neural Network, IEEE Access, **9** (2021), 50831–50845. https://doi.org/10.1109/ACCESS.2021.3068998
15. M. M. Bazuhair, S. Z. M. Jamaludin, N. E. Zamri, M. S. M. Kasihmuddin, M. A. Mansor, A. Alway, et al., Novel Hopfield neural network model with election algorithm for random 3 satisfiability, Processes, **9** (2021), 1292. https://doi.org/10.3390/pr9081292
16. N. Noilublao, S. Bureerat, Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms, Comput. Struct., **89** (2011), 2531–2538. https://doi.org/10.1016/j.compstruc.2011.08.010
17. D. Zhan, A. Q. Tian, S. Q. Ni, Optimizing PID control for multi-model adaptive high-speed rail platform door systems with an improved metaheuristic approach, Int. J. Electr. Power Energy Syst., **169** (2025), 110738. https://doi.org/10.1016/j.ijepes.2025.110738
18. A. Q. Tian, F. F. Liu, H. X. Lv, Snow Geese Algorithm: A novel migration-inspired meta-heuristic algorithm for constrained engineering optimization problems, Appl. Math. Model., **126** (2024), 327–347. https://doi.org/10.1016/j.apm.2023.10.045
19. 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

20. J. Luo, J. Zhou, X. Jiang, H. Lv, A modification of the imperialist competitive algorithm with hybrid methods for multi-objective optimization problems, Symmetry, **14** (2022), 173. https://doi.org/10.3390/sym14010173
21. A. Alway, N. E. Zamri, S. A. Karim, M. A. Mansor, M. S. Mohd Kasihmuddin, M. Mohammed Bazuhair, Major 2 satisfiability logic in discrete Hopfield neural network, Int. J. Comput. Math., **99** (2022), 924–948. https://doi.org/10.1080/00207160.2021.1939870
22. J. Chen, M. S. M. Kasihmuddin, Y. Gao, Y. Guo, M. A. Mansor, N. A. Romli, et al., PRO2SAT: Systematic probabilistic satisfiability logic in discrete Hopfield neural network, Adv. Eng. Softw., **175** (2023), 103355. https://doi.org/10.1016/j.advengsoft.2022.103355
23. Y. Gao, Y. Guo, N. A. Romli, M. S. M. Kasihmuddin, W. Chen, M. A. Mansor, et al., GRAN3SAT: Creating flexible higher-order logic satisfiability in the discrete Hopfield neural network, Mathematics, **10** (2022), 1899. https://doi.org/10.3390/math10111899
24. J. Reuben, Rediscovering majority logic in the post-CMOS era: A perspective from in-memory computing, J. Low Power Electron. Appl., **10** (2020), 28. https://doi.org/10.3390/jlpea10030028
25. A. Chattopadhyay, L. Amarú, M. Soeken, P. E. Gaillardon, G. De Micheli, Notes on majority Boolean algebra, Proc. IEEE Int. Symp. Multiple-Valued Logic (ISMVL), (2016), 50–55. https://doi.org/10.1109/ISMVL.2016.21
26. X. Song, Q. Yan, M. Zhao, An adaptive artificial bee colony algorithm based on objective function value information, Appl. Soft Comput., **55** (2017), 384–401. https://doi.org/10.1016/j.asoc.2017.01.031
27. L. Kammerer, G. Kronberger, B. Burlacu, S. M. Winkler, M. Kommenda, M. Affenzeller, Symbolic regression by exhaustive search: reducing the search space using syntactical constraints and efficient semantic structure deduplication, in Genetic Programming Theory and Practice XVII, Springer, Cham, (2020), 79–99. https://doi.org/10.1007/978-3-030-39958-0_5
28. G. Gosti, V. Folli, M. Leonetti, G. Ruocco, Beyond the maximum storage capacity limit in Hopfield recurrent neural networks, Entropy, **21** (2019), 726. https://doi.org/10.3390/e21080726
29. Y. Guo, M. S. M. Kasihmuddin, Y. Gao, M. A. Mansor, H. A. Wahab, N. E. Zamri, et al., YRAN2SAT: A novel flexible random satisfiability logical rule in discrete Hopfield neural network, Adv. Eng. Softw., **171** (2022), 103169.
30. G. Pinkas, Symmetric neural networks and propositional logic satisfiability, Neural Comput., **3** (1991), 282–291. https://doi.org/10.1162/neco.1991.3.2.282
31. S. S. M. Sidik, N. E. Zamri, M. S. M. Kasihmuddin, H. A. Wahab, Y. 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
32. J. Bruck, J. W. Goodman, A generalized convergence theorem for neural networks, IEEE Trans. Inf. Theory, **34** (1988), 1089–1092. https://doi.org/10.1109/18.21239
33. L. C. Kho, M. S. M. Kasihmuddin, M. Mansor, S. Sathasivam, Logic mining in league of legends, Pertanika J. Sci. Technol., **28** (2020), 211–225.
34. M. A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Grey wolf optimization algorithm with discrete hopfield neural network for 3 satisfiability analysis, J. Phys.: Conf. Ser., **1821** (2021), 012038. https://doi.org/10.1088/1742-6596/1821/1/012038
35. M. S. Raza, U. Qamar, A hybrid feature selection approach based on heuristic and exhaustive algorithms using Rough set theory, Proc. Int. Conf. Internet Things Cloud Comput., (2016), 1–7. https://doi.org/10.1145/2896387.2896432

36. H. H. Hoos, T. Stützle, Local search algorithms for SAT: An empirical evaluation, J. Autom. Reason., **24** (2000), 421–481. https://doi.org/10.1023/A:1006350622830
37. P. Berenbrink, A. Coja-Oghlan, C. Cooper, T. Götte, L. Hintze, P. Zakharov, WalkSAT is linear on random 2-SAT, SIAM J. Discrete Math., **39** (2025), 1939–1952. https://doi.org/10.1137/24M1718639
38. G. Pedretti, F. Böhm, T. Bhattacharya, A. Heittmann, X. Zhang, M. Hizzani, T. Van Vaerenbergh, Solving Boolean satisfiability problems with resistive content addressable memories, npj Unconv. Comput., **2** (2025), 1–8. https://doi.org/10.1038/s44335-025-00020-w
39. M. E. Alshammari, M. A. Ramli, I. M. Mehedi, Hybrid chaotic maps-based artificial bee colony for solving wind energy-integrated power dispatch problem, Energies, **15** (2022), 4578. https://doi.org/10.3390/en15134578
40. Q. M. Alzubi, M. Anbar, Y. Sanjalawe, M. A. Al-Betar, R. Abdullah, Intrusion detection system based on hybridizing a modified binary grey wolf optimization and particle swarm optimization, Expert Syst. Appl., **204** (2022), 117597. https://doi.org/10.1016/j.eswa.2022.117597
41. R. Eberhart, J. Kennedy, Particle swarm optimization, Proc. IEEE Int. Conf. Neural Netw., **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
42. J. Too, A. R. Abdullah, N. Mohd Saad, W. Tee, EMG feature selection and classification using a Pbest-guide binary particle swarm optimization, Computation, **7** (2019), 12. https://doi.org/10.3390/computation7010012
43. M. Pantourakis, S. Tsafarakis, K. Zervoudakis, E. Altsitsiadis, A. Andronikidis, V. Ntamadaki, Clonal selection algorithms for optimal product line design: a comparative study, Eur. J. Oper. Res., **298** (2022), 585–595. https://doi.org/10.1016/j.ejor.2021.07.006
44. D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, IFSA World Congr., 2007, 789–798. https://doi.org/10.1007/978-3-540-72950-1_77
45. M. S. M. Kasihmuddin, M. A. Mansor, S. A. Alzaeemi, S. Sathasivam, Satisfiability logic analysis via radial basis function neural network with artificial bee colony algorithm, Int. J. Interact. Multimed. Artif. Intell., **6** (2021). https://doi.org/10.9781/ijimai.2020.06.002
46. S. Barbero, E. Bellini, C. Sanna, J. Verbel, Practical complexities of probabilistic algorithms for solving Boolean polynomial systems, Discret. Appl. Math., **309** (2022), 13–31. https://doi.org/10.1016/j.dam.2021.11.014
47. D. S. K. Karunasingha, Root mean square error or mean absolute error? Use their ratio as well, Inf. Sci., **585** (2022), 609–629. https://doi.org/10.1016/j.ins.2021.11.036
48. 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
49. G. A. Senthil, A. Raaza, N. Kumar, Internet of things energy efficient cluster-based routing using hybrid particle swarm optimization for wireless sensor network, Wirel. Pers. Commun., **122** (2022), 2603–2619. https://doi.org/10.1007/s11277-021-09015-9
50. S. Mirjalili, A. Lewis, S-shaped versus V-shaped transfer functions for binary particle swarm optimization, Swarm Evol. Comput., **9** (2013), 1–14. https://doi.org/10.1016/j.swevo.2012.09.002

51. Q. Yang, Y. W. Bian, X. D. Gao, D. D. Xu, Z. Y. Lu, S.W. Jeon, et al., Stochastic triad topology based particle swarm optimization for global numerical optimization, Mathematics, **10** (2022), 1032. https://doi.org/10.3390/math10071032
52. M. Shi, Y. Zhang, H. Wang, J. Hu, X. Wu, A clonal selection optimization system for multiparty secure computing, Complex., **2021** (2021), 7638394. https://doi.org/10.1155/2021/7638394
53. A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, P. M. Rowinski, Swarm intelligence and evolutionary algorithms: performance versus speed, Inf. Sci., **384** (2017), 34–85. https://doi.org/10.1016/j.ins.2016.12.028
54. J. Yuan, Z. Liu, Y. Lian, L. Chen, Q. An, L. Wang, et al., Global optimization of UAV area coverage path planning based on good point set and genetic algorithm, Aerospace, **9** (2022), 86. https://doi.org/10.3390/aerospace9020086
55. M. A. Al-qaness, A. A. Ewees, H. Fan, L. Abualigah, M. A. Elaziz, Boosted ANFIS model using augmented marine predator algorithm with mutation operators for wind power forecasting, Appl. Energy, **314** (2022), 118851. https://doi.org/10.1016/j.apenergy.2022.118851
56. M. S. M. Kasihmuddin, S. Sathasivam, Accelerating activation function in higher order logic programming, AIP Conf. Proc., **1750** (2016), 030006. https://doi.org/10.1063/1.4954542
57. D. P. Rini, S. M. Shamsuddin, S. S. Yuhaniz, Particle swarm optimization: technique, system and challenges, Int. J. Comput. Appl., **14** (2011), 19–26. https://doi.org/10.5120/ijais-3651
58. K. M. Smolka, I. Verheul, K. Burmeister-Lamp, P. P. Heugens, Get it together! Synergistic effects of causal and effectual decision-making logics on venture performance, Entrep. Theory Pract., **42** (2018), 571–604. https://doi.org/10.1177/10422587187834
59. S. Yousif, M. P. Saka, Optimum design of post-tensioned flat slabs with its columns to ACI 318-11 using population based beetle antenna search algorithm, Comput. Struct., **256** (2021), 106520. https://doi.org/10.1016/j.compstruc.2021.106520

AIMS Press