Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Optimising 2SAT Problem Resolution Through Metaheuristic-Driven Hybrid Models of Fuzzy Logic and Hopfield Neural Networks

Salaudeen Abdulwaheed Adebayo, Saratha Sathasivam*

School of Mathematical Sciences, Universiti Sains Malaysia, Penang, 11800, USM, Malaysia
E-mail: saratha@usm.my

Received: 17 December 2024; **Revised:** 4 February 2025; **Accepted:** 5 February 2025

Abstract: This study addresses the limitations of traditional Boolean-based methods for solving 2SAT problems, which mainly classify variables as true or false and struggle with imprecise information and local minima in Hopfield Neural Networks (HNN). To overcome this challenge, we developed a novel approach that integrates fuzzy logic with HNN, introducing flexibility by permitting the network to handle partial truths during the learning phase. To further enhance this approach, we incorporated Simulated Annealing (SA) and Modified Grey Wolf Optimization (MGWO) to improve the network's ability to escape local minima and find optimal solutions. We developed three hybrid models: HNN2SATFuzzy, HNN2SATFuzzySA, and HNN2SATFuzzyMGWO, and evaluated their performance using metrics such as RMSE, MAE, SSE, SMAPE, global minimum ratio, and computation time. The results show that HNN2SATFuzzyMGWO outperformed the other hybrids, offering a more robust, accurate, and efficient solution to 2SAT problems. This work extends the applicability of HNN in solving SAT problems and provides a sophisticated alternative to classical Boolean approaches, paving the way for more adaptable problem-solving techniques in this field.

***Keywords*:** hybrids, simulated annealing, performance metrics, satisfiability, recurrent network

MSC: 68T07, 90C59, 03B70

Abbreviation

HNN	Hopfield Neural Networks
RNN	Recurrent Neural Network
SA	Simulated Annealing
MGWO	Modified Grey Wolf Optimization
GWO	Grey Wolf Optimization
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
SSE	Sum of Square Error
SMAPE	Symmetric Mean Absolute Percentage Error

2-SAT	Two Satisfiability
GA	Genetic Algorithm
ABC	Artificial Bee Colony
AIS	Artificial Immune System
CAM	Content Addressable Memory
UNSAT	Unsatisfiable
INF	Implicative Normal Form
FL	Fuzzy Logic
MF	Membership Function
T	Temperature Parameters
CPU	Central Processing Unit
GMR	Global Minimum Ratio
ACO	Ant Colony Optimisation
k-SAT	K Satisfiability

1. Introduction

Hopfield neural network, an iterative auto-associative network named after John Hopfield, is a subclass of Recurrent Neural Network (RNN) in which the synaptic weight between the neurons is the same in either direction. It is mostly utilized for auto-association memory, pattern recognition, and optimization tasks [1, 2]. HNN can be combined with other algorithms to reinforce its strengths and mitigate its known drawbacks. Such integrations foster the development of more robust and powerful fusions. The integration of HNN and logic programming is a prominent attempt that has attracted significant study. Its flexibility and simplicity make it a recipe for such integration [3]. Neural networks excel at learning from datasets, enabling them to enhance automatic reasoning by identifying patterns and relationships within datasets. On the other hand, logic programming provides an avenue for symbolic reasoning, enabling systems to handle rule-based, knowledge-driven decision-making, and when combined with data-driven learning, these approaches can improve the strengths of both in developing more robust and intelligent systems [4]. Both technologies encompass computational methodologies with diverse applications in artificial intelligence [5, 6]. Wan Abdullah presented a novel technique that combines logic programming with HNN, where neurons were used to store the truth values of atoms in the Boolean formula, allowing the control of data within the CAM of [7]. However, several studies have reported that local minima problems are inherent to both discrete and continuous versions of HNN, leading scholars from different domains to engage in various means of addressing this infamous challenge, and employment of metaheuristic algorithms is one of the popular strategies.

The deployment of metaheuristic algorithms in the training phase of HNN is one of optimisation techniques that has recorded massive acceptability. A Genetic Algorithm (GA) was incorporated into the training phase of HNN to mitigate local minima problems and improve the system's efficacy [8]. Kasihmuddin et al. [9] asserted the superiority of the Artificial Bee Colony (ABC) algorithm in solving more regulated 2-SAT problems compared to a hybrid enhanced with a genetic algorithm. The study conducted by [10] stated that the Artificial Immune System (AIS) algorithm-enhanced HNN demonstrates effectiveness in solving 3-SAT problems and showcases the applicability of metaheuristic algorithms as optimisation tools in the training phase of HNN. Previous research, Blocho [11] and Gao et al. [12] reported effectiveness of deploying metaheuristic algorithms in optimization tasks. Spear [13] demonstrates the applicability of Simulated Annealing (SA) specifically in solving hard satisfiability problems and commends algorithm's performance and effectiveness. A simulated annealing artificial neural network built on harmony theory was deployed to solve propositional calculus satisfiability (SAT) problems [14]. Selman and Kautz [15] introduced a greedy search algorithm called GSAT, a sound and incomplete algorithm for the class of hard problems in the satisfiability problem; its outcome alleviates the problem of local minima, promotes asymptotic convergence, and permits random movements within the search space [11, 16, 17]. Most heuristic search techniques share many similarities with hill climbing or gradient search techniques, albeit with a few deviations; they are indifferent to a function's gradient, and neither smoothness nor continuity of the functions is a yardstick for their success [18]. Sathasivam and Wan Abdullah [19] implemented HornSAT in HNN and reported the hybrid's effectiveness in obtaining the satisfying assignment for the logic program when the energy function of HNN reaches its global minimum.

Furthermore, the conventional Boolean variable of SAT problem-solving is restricted to the binary classification of variables, which limits flexibility and makes it difficult to suit the complexity of real-world scenarios and often hinders their ability to find optimal solutions. There is a pressing need for a more adaptable approach that can handle partial truths and imprecise information, thereby enhancing HNN's performance in escaping local minima and effectively solving SAT problems. This research deploys a fuzzy logic approach to improve the training phase of the HNN for solving satisfiability problems. Previous studies have not sufficiently explored the combination of HNN with fuzzy logic in resolving 2-SAT problems; instead, they focused on fuzzifying the input neurons of HNN, neglecting the impact on the structure and quality of the input data, which are knowledge-based. Such conversions might alter or reduce the quality of knowledge in the database [20]. However, this study explores other means of fuzzification to reduce the implications of earlier attempts. To broaden the possibilities of satisfiability, each potential solution is fuzzified, accepting solutions with full or near-full membership values based on a predefined alpha value. This approach facilitates fast decision-making by allowing the network to consider solutions that satisfied the alpha threshold, thus promoting a smooth transition in reasoning and increasing the system's resilience to noise. Fuzzification aligns with human reasoning features, where solutions are not strictly binary but can fall within a spectrum of acceptability. The alpha-cut defuzzification method is employed to classify the solutions based on their performance in the fitness function. This process segregates solutions according to their membership values in the network's state, accounting for inherent uncertainty and the partial assignment of clauses. By doing so, the defuzzification process helps in managing the complexity and ambiguity present in satisfiability problems, leading to more robust outcomes.

In this study, the performance of the metaheuristic-enhanced Hopfield model will be evaluated using simulated data for the 2-SAT satisfiability problem. This approach combines the strengths of fuzzy logic with the optimisation capabilities of metaheuristic methods to enhance the network's ability to find optimal or near-optimal solutions in complex problem spaces. The 2-SAT problem is a specific case of the broader k-SAT problem, where each clause contains exactly two literals connected by a logical OR operator. In computational complexity theory, it belongs to the P complexity class, indicating its solvability in polynomial time with a computational complexity of O (n), where n denotes the number of Boolean variables. The polynomial time characteristic makes it computationally efficient and often preferred over higher-order k-SAT $(k > 2)$ and other SAT variants. Its linear time complexity accounts for rapid resolution even in large-scale instances. The preference for 2-SAT algorithms lies in their simplicity and directness compared to the more intricate logical structures encountered in higher-order k-SAT problems. Moreover, its direct correspondence to real-world situations boosts its applicability across various scientific and technological domains.

To investigate the effectiveness of integrating fuzzy logic with Hopfield Neural Networks (HNN), we proposed three hybrid models: HNN2SATFuzzy, HNN2SATFuzzySA, and HNN2SATFuzzyMGWO. These models were evaluated using simulated 2-SAT datasets. The HNN2SATFuzzy integrates fuzzy logic to screen solutions, while the other two models incorporate additional optimization techniques, Simulated Annealing (SA) and Modified Grey Wolf Optimisation (MGWO) to further improve performance. A detailed comparative analysis was conducted, focusing on energy analysis, performance metrics and computational time. The results show that the hybrid HNN2SATFuzzyMGWO demonstrates superior performance compared to both HNN2SATFuzzy and HNN2SATFuzzySA across most performance metrics. Therefore, we conclude that the HNN2SATFuzzyMGWO model offers significant improvements in both efficiency and robustness for solving 2-SAT problems.

2. Satisfiability terminologies and description

SAT is abbreviation for Satisfiability, a fundamental problem in computational complexity theory and computer science which is a task in mathematical logic that investigate the existence of ways in which variables in a propositional formula in Conjunctive Normal Form (CNF) can be assigned a Boolean value for which that formula evaluates to true

(1). A Boolean expression is a formula in propositional logic, where logical statements are variables that can be true or false. Logical formulas are formed using combination of AND, OR, NOT, IMPLIES operators. One of the most common representations of Boolean expression is Conjunctive Normal Form (CNF) which is a special form of proportional logic representation where propositional expression is written as a conjunction (AND) of clauses, and each clause is a disjunction (OR) of literals. A literal is a propositional variable or its the negation. Suppose $C = (C_1, C_2, \cdots C_m)$ denotes the set of m clauses in a particular propositional formula, each C_i is a disjunction of Boolean variables from the set $\{x_1, x_2, \dots, x_N\}$, x_i is false or true, and N denotes the number of variables in the propositional formula.

A clause in 2-SAT is defined as:

$$C_i = \bigvee_{j=1}^{2} l_{ij}, \tag{1}$$

where i is the clause index in the Boolean expression and j represents the variable index in the clause. For instance, l_{31} is interpreted as the first variable in the third clause. Within each of the clauses, the occurrence of x_i and $\neg x_i$ is limited to at most one. A clause C_i is said to be satisfied if at least one of its literals is true or its negation is false [21]. The Boolean expression is defined as:

$$B_{\text{exp}} = \bigwedge_{i=1}^{M} C_i. \tag{2}$$

Where, B_{exp} is the Boolean formula and can be equivalently rewritten by substituting C_i from Equation (1) into Equation (2) to produce Equation (3):

$$B_{\text{exp}} = \bigwedge_{i=1}^{M} \left(\bigvee_{j=1}^{2} l_{ij} \right). \tag{3}$$

Satisfiability is centered on Equation (3), which investigates the likelihood of a satisfying assignment for which:

$$B_{\text{exp}}(\text{assignment}) = \bigwedge_{i=1}^{M} \left(\bigvee_{j=1}^{2} l_{ij} \right) = 1. \tag{4}$$

The existence of an assignment that satisfied equation (4) is termed Satisfiability (SAT), while its absence is unsatisfiability, or simply UNSAT [22]. The SAT problem belongs to classical computing problems known as NP-complete problems, showcasing computational complexity heightens with problem size. As such, developing efficient algorithms capable of solving SAT instances has been a focal point of many research due to the various applications of SAT in the real world. The fact that many real-life problems across diverse domains can be effectively modelled using satisfiability due to its ability to encode logical constraints and relationships among variables in a concise and simple manner makes it appropriate for modelling of various combinatorial and constraint satisfaction problems while extending its applicability beyond electronic design, automation, software and hardware verification, and many AI applications. Its applications in many fields have called for the development of effective SAT solvers to either resolve or indicate the non-existence of any solution [23]. The solution space of satisfiability problems can be classified into deterministic and approximate solutions, each with its own strengths and drawbacks. Most prominent satisfiability solver algorithms have worst-case exponential runtimes [24], these solvers remain relevant tools in the field and inspire further research

[25]. However, one form of satisfiability problem that has gained massive popularity and wide acceptability in several domains is 2-SAT, or simply binary satisfiability [26, 27]. 2-SAT is a restricted form of the traditional Boolean satisfiability problem, where the length of the clauses is 2, that is, each clause consists of two literals which are connected by a logical OR operator.

A 2-CNF logical formula with m finite clauses can be written as shown in Equation (5):

$$B_{2\text{SAT}} = C_1 \wedge C_2 \wedge \ \dots \ \wedge C_m, \tag{5}$$

where $C_1 = (v_1 \vee v_2)$, $C_2 = (\neg v_3 \vee v_4)$, and $C_3 = (v_5 \vee \neg v_6)$. Equation (5) can be rewritten as Equation (6):

$$B_{2\text{SAT}} = (v_1 \vee v_2) \wedge (\neg v_3 \vee v_4) \wedge (v_5 \vee \neg v_6). \tag{6}$$

Similarly, this can also be rewritten in a more compact form as shown in Equation (7):

$$B_{2\text{SAT}} = \bigwedge_{i=1}^{m} C_i, \tag{7}$$

where, m denotes the number of clauses and C_i denotes the i-th clause. A propositional variable (v_i) is either true (1) or false (0) [28]. Thus, v_i represents a unique piece of information or knowledge different from other variables v_j [29]. The nature of the underlying task determines the number of variables and clauses in a Boolean formula. The higher the number of variables and clauses, the more complex the problem becomes.

3. Embedding 2-SAT logical problem in HNN

Solving 2-SAT logical problems using HNN has attracted a considerable amount of interest among scholars and researchers across different domains. The possibilities of implementing logical problems within the neural paradigm revolve around two main features: the activation dynamics and the corresponding energy function that decreases as the network evolves spontaneously toward a stable state [30]. In logic programming, it involves encoding the logical variables into the architecture of the HNN. The variables are encoded using neurons and clauses as connection weights [31]. This enables the network to effectively capture the complex relationships between variables and clauses in terms of neurons and connections. Thus, the interpretability of logic programs is translated into HNN, while utilising the dynamics of Hopfield networks to interpret the features of the embedded SAT problem through the principle of energy minimisation [32]. The configuration of the energy function is structured such that the attainment of minimum energy aligns with the satisfiability of the embedded Boolean formula. In other words, the presence of a configuration with a minimum energy value on the network's surface exhibits a successful assignment of truth values that satisfies the underlying Boolean formula [33].

In addition, iterative state update plays a crucial role in HNN in such a way that regardless of the initial state of Hopfield networks, HNN will dynamically and continuously update the states of neurons until the energy of the noisy state matches the state of one of the learnt patterns in the network [34]. Once the network has reached a stable state, the resulting network state is regarded as satisfying assignments or a model [35, 36]. The n neurons that make up the Hopfield Neural Network (HNN) have threshold values T_i. The weighted sum of the neuron outputs, where $j = 1, 2, 3, \dots, n$, represents the feedback input to the i-th neuron. Using W_{ij} as the synaptic weight that connects the j-th neuron's output to the i-th neuron's input, the activation of neurons in discrete-time HNN is governed by the relation described in Equation (8):

$$S_i = \begin{cases} 1, & \sum_{j \neq i} W_{ij} S_j > T_i \\ \\ -1, & \text{otherwise.} \end{cases} \tag{8}$$

In the Hopfield Neural Network (HNN), the activation of each neuron takes a digital value of either -1 or 1. Our primary focus here is to implement a logic program in the HNN. A logic program is a declarative computational paradigm constructed using formal logic, primarily composed of rules and facts [37]. Such programs can be converted to Implication Normal Form (INF), as shown in Equation (9), and can be readily converted into a 2-SAT Boolean formula, as illustrated in Equation (10).

$$B_{2\text{SAT}} = v_1,\ v_2 \leftarrow,\ v_3 \leftarrow v_4,\ v_5 \leftarrow v_6. \tag{9}$$

The implication representation of the logic program in Equation (9) can be converted to an equivalent 2-CNF. The equivalent clausal representation in 2-SAT is shown in Equation (10):

$$B_{2\text{SAT}} = (v_1 \vee v_2) \wedge (v_3 \vee \neg v_4) \wedge (v_5 \vee \neg v_6). \tag{10}$$

Each of the variables in Equation (10) takes a binary value of either 0 or 1 and represents a neuron in the Hopfield Neural Network (HNN). The cost function whose logical inconsistency is to be minimized can be obtained from Equation (10), which can be converted into a minimization problem that maps onto the HNN energy function.

Negating a maximization problem essentially transforms it into a minimization problem. This can be achieved by negating Equation (10) and transforming the logical operators: the conjunction $(\wedge)$ becomes multiplication, and the disjunction $(\vee)$ becomes addition.

The propositional variables can then be written in terms of their bipolar representations to suit the HNN data structure: $v_i = \frac{1}{2}(1 + S_{v_i})$, $\neg v_i = \frac{1}{2}(1 - S_{v_i})$. By negating the objective function, the problem now seeks to minimize the negative of what was originally being maximized. Equation (10) is transformed into Equation (11):

$$\neg B_{2\text{SAT}} = \frac{1}{2}(1 - S_{v_1}) \cdot \frac{1}{2}(1 - S_{v_2}) + \frac{1}{2}(1 - S_{v_3}) \cdot \frac{1}{2}(1 + S_{v_4}) + \frac{1}{2}(1 + S_{v_5}) \cdot \frac{1}{2}(1 - S_{v_6}). \tag{11}$$

Replacing $\neg B_{2\text{SAT}}$ with the energy function $E_{B_{2\text{SAT}}}$, we have:

$$E_{B_{2\text{SAT}}} = \frac{1}{2}(1 - S_{v_1}) \cdot \frac{1}{2}(1 - S_{v_2}) + \frac{1}{2}(1 - S_{v_3}) \cdot \frac{1}{2}(1 + S_{v_4}) + \frac{1}{2}(1 + S_{v_5}) \cdot \frac{1}{2}(1 - S_{v_6}). \tag{12}$$

By inspection, the assignment $(1,\ 1,\ 1,\ 1,\ 1,\ 1)$ is one of the satisfying assignments to the problem in Equation (12). Substituting this assignment such that: $S_{v_1} = S_{v_2} = S_{v_3} = S_{v_4} = S_{v_5} = S_{v_6} = 1$.

We obtain Equation (13). It can be observed that the same solution also satisfies the problem in Equation (10).

$$E_{B_{2\text{SAT}}}(1,\ 1,\ 1,\ 1,\ 1,\ 1) = 0. \tag{13}$$

This shows that satisfiability problems can be implemented in the Hopfield Neural Network (HNN), since they can be formulated as a minimization problem. Equation (12) can be written compactly as shown in Equation (14), as crafted by [38]:

$$E_{B_{2\mathrm{SAT}}} = \frac{1}{2^k} \sum_{i=1}^{N_C} \left(\prod_{j=1}^{2} C_{ij} \right), \tag{14}$$

where k is the number of literals per clause, N_C denotes the number of clauses in the Boolean formula, and C_{ij} can be defined as shown in Equation (15):

$$C_{ij} = \begin{cases} 1 + S_A, & \text{if } A \text{ is a positive literal,} \\ \\ 1 - S_A, & \text{if } \neg A \text{ is a negative literal.} \end{cases} \tag{15}$$

where, S_A denotes the state of the neuron and takes values from the set $\{-1,\ 1\}$. However, all the clauses in Equation (10) must be satisfied to have $E_{B_{2\mathrm{SAT}}} = 0$. The inability to obtain that value demonstrates the presence of clause(s) that are not satisfied [7]. The function $E_{B_{2\mathrm{SAT}}}$ outputs the number of unsatisfied clauses.

Therefore, we can also define the fitness function based on the number of clauses that evaluate to 1, as shown in Equation (16):

$$f^i = \sum_{i=1}^{N_C} C_i, \quad \text{where} \quad C_i = \begin{cases} 1, & \text{if clause is true,} \\ \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

C_i takes values from the set $\{0,\ 1\}$; it is 1 when the clause evaluates to true, and 0 otherwise.

The Hopfield Neural Network is made up of a finite number of neurons, say N, which are described by an Ising variable $S_i(t)$ where $i = 1,\ 2,\ 3,\ \ldots$. $S_i(t)$ is the state of neuron i at time t and $S_i(t) \in \{-1,\ 1\}$. HNN updates are mostly done asynchronously, and the state update is done deterministically using the relation:

$$h_i = \sum_{j=1}^{N} W_{ij}^{(2)} S_j(t) + \theta_i^{(1)}, \tag{17}$$

where i, j run over all the neurons in the network, W_{ij} is the connection weight from the j^{th} neuron to the i^{th} neuron, and θ_i is the bias associated with the i^{th} neuron. The Hopfield Neural Network (HNN) is characterized by the absence of self-connections, implying $W_{ii} = 0,\ \ \forall i \leq N$, which results in a zero-diagonal symmetric coupling matrix, i.e., $W_{ij} = W_{ji},\ \ \forall i,\ j \leq N$. The energy function of the HNN is defined in Equation (18):

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=i}^{N} W_{ij}^{(2)} S_i S_j - \sum_{i=1}^{N} \theta_i^{(1)} S_i. \tag{18}$$

The energy function plays a significant role in the dynamics of the HNN. It is a scalar value associated with each stage during state updating and it quantifies the current state of the network. The energy function forms the core of the optimization process in HNN.

Incorporating a logic program into the HNN involves encoding the objective function defined in Equation (10) into the network, such that the satisfying assignment of the formula in Equation (10) corresponds to a minimum energy state in the Hopfield Neural Network. The weights of the HNN are defined based on the atoms of the logic program, using procedures explicitly described by Abdullah [7]. Consequently, $E_{B_{2\text{SAT}}} = 0$ implies that the optimal training phase has been achieved.

The field for the higher connection model is given as shown in Equation (19):

$$h_i = \sum_{j=1}^{N}\left(\sum_{k=1}^{N} W_{ijk}^{(3)} S_j(t) S_k(t)\right) + \sum_{j=1}^{N} W_{ij}^{(2)} S_j(t) + \theta_i^{(1)}. \tag{19}$$

The corresponding energy function for a higher-order connection is calculated using Equation (20):

$$E = -\frac{1}{3}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} W_{ijk}^{(3)} S_i(t) S_j(t) S_k(t) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}^{(2)} S_i(t) S_j(t) - \sum_{i=1}^{N} \theta_i^{(1)} S_i(t), \tag{20}$$

where $W_{ijk} = W_{kji}$ for distinct $i,\ j,\ k$, and $[\ \cdot\ ]$ denotes permutations in cyclic order. Also, $W_{iii} = W_{jjj} = W_{kkk} = 0$. The updating rule for the neuron at iteration $(t+1)$ is written as:

$$S_i(t+1) = \operatorname{sgn}[h_i(t)]. \tag{21}$$

The local field updates the neuron state to a new state or maintains the previous state, which can either minimize or retain the previous network's energy value.

4. Fuzzy set, fuzzy logic, operations and transformation

The introduction of fuzzy logic and fuzzy sets by Zadeh in 1965 [39] gave birth to a different and powerful computation paradigm in mathematics and computer science. Its start-up was primarily to complement the shortfall in precise mathematical models for capturing nonlinearities, vagueness, and uncertainties [40]. Fuzzy Logic (FL) steps in to provide human-like reasoning architecture; it has since then turned into an important tool in automata theory, game theory, topology, integral linguistics, decision-making theory, and control designs, among others [41, 42]. Its ability to handle vagueness in data has paved the way for the improvement of control designs in appliances, and it has earned widespread adoption in various domains as an innovative inference machine. Fuzzy logic belongs to a class of many-valued logic or infinite truth degrees that can handle reasoning and decision-making problems marred by uncertainty or ambiguity. It is an extension of classical (Boolean) logic that allows a gradual switch between completely false and complexly true using degree of membership to determine the truth value of atoms. In fuzzy logic, exact reasoning is seen as the restrictive case of proper reasoning [43]. Jozi et al. [44] asserted that linear combinations of various fuzzy basis functions can accurately approximate continuous nonlinear functions [45]. A fuzzy set can be defined as follows: given Y as the universe of discourse, and $y \in Y$, a fuzzy set $\bar{A}$ on Y is a collection of ordered pairs:

$$\bar{A} = \{(y,\ \mu_{\bar{A}}(y)) \mid y \in Y\}, \tag{22}$$

where y is the crisp value and $\mu_{\bar{A}}(y)$ is the membership value of y. A membership function, or grade, is a function that quantifies the degree of membership for each y in the universe of discourse, and it is defined as:

$$\mu_{\bar{A}} : Y \to [0,\ 1]. \tag{23}$$

From equation (23), $\mu_{\bar{A}}$ can take any real value between 0 and 1, where 0 and 1 indicate no membership and full membership, respectively. Various forms of membership functions have been proposed, ranging from simple ones like linear, threshold, and triangular, to advanced types like trapezoidal, Gaussian, sigmoidal, and more. However, the choice of membership function depends largely on the nature of the underlying task and applications.

Most fuzzy logic systems share common basic components: fuzzification, rule evaluation, aggregation, and defuzzification. Fuzzy logic can be represented as shown in equation (24):

$$\bar{A} = \{(y_1,\ \mu_{\bar{A}}(y_1)),\ (y_2,\ \mu_{\bar{A}}(y_2)),\ (y_3,\ \mu_{\bar{A}}(y_3)),\ \ldots \mid y_1,\ y_2,\ y_3,\ \ldots \in Y\}, \tag{24}$$

or equivalently, as shown in equation (25) [46]:

$$\bar{A} = \left\{ \frac{\mu_{\bar{A}}(y_1)}{y_1} + \frac{\mu_{\bar{A}}(y_2)}{y_2} + \frac{\mu_{\bar{A}}(y_3)}{y_3} + \ldots + \frac{\mu_{\bar{A}}(y_n)}{y_n} \right\} = \left\{ \sum_{i=1}^{n} \frac{\mu_{\bar{A}}(y_i)}{y_i} \right\}. \tag{25}$$

It is worth noting that the sign '+' is not an algebraic operation but a notation. However, both fuzzy logic and fuzzy sets share similar operators with classical Boolean logic and classical set theory, respectively, albeit with some additions and modifications to the former. Fuzzy sets utilise set operators such as max, min, and complementation in place of set operators, and all other properties and identities of classical set theory are applicable. Additionally, fuzzy sets uses identities like algebraic sum, algebraic product, bounded sum, bounded difference, and bounded product [42]. We can view fuzzy logic as the arithmetization of set theory, adapting the operators to suit arithmetic operations. Commonly used operations in fuzzy logic include product, sum, and complement [47].

Product, Intersection (AND) Operation:

$$\mu_{S_1 \cap S_2}(A) = \min(\mu_{S_1}(A),\ \mu_{S_2}(A)). \tag{26}$$

Sum, Union (OR) Operation:

$$\mu_{S_1 \cup S_2}(A) = \max(\mu_{S_1}(A),\ \mu_{S_2}(A)). \tag{27}$$

Complementary Operation:

$$\overline{\mu_S(A)} = 1 - \mu_S(A). \tag{28}$$

Nevertheless, equations (29) and (30) primarily serve as extensions of the classical set operators, tailored to handle the vagueness and uncertainty inherent in information and data.

Algebraic Product:

$$S_1 \times S_2 \Leftrightarrow \mu_{S_1 \times S_2} = \mu_{S_1} \cdot \mu_{S_2}. \tag{29}$$

Algebraic Sum:

$$S_1 + S_2 \Leftrightarrow \mu_{S_1 + S_2} = \mu_{S_1} + \mu_{S_2} - \mu_{S_1} \cdot \mu_{S_2} = 1 - (1 - \mu_{S_1})(1 - \mu_{S_2}). \tag{30}$$

Equations (31)-(32) are named bounded sum, bounded difference, and bounded product, respectively, and were also proposed by Zadeh [40].

Bounded Sum:

$$S_1 \oplus S_2 \Leftrightarrow \mu_{S_1 \oplus S_2} = \min\left(1,\ \mu_{S_1} + \mu_{S_2}\right). \tag{31}$$

Bounded Difference:

$$S_1 \Theta S_2 \Leftrightarrow \mu_{S_1 \Theta S_2} = \max\left(0,\ \mu_{S_1} - \mu_{S_2}\right). \tag{32}$$

Bounded Product:

$$S_1 \odot S_2 \Leftrightarrow \mu_{S_1 \odot S_2} = \max\left(0,\ \mu_{S_1} + \mu_{S_2} - 1\right), \tag{33}$$

where the logical operators $\wedge$, $\vee$, $+$, and $-$ represent *min*, *max*, arithmetic sum, and arithmetic difference, respectively, in fuzzy logic. Using appropriate representations and valid notations, equations (26)-(28) can be deployed to verify the properties of fuzzy logic in relation to classical Boolean logic and set theory.

Furthermore, fuzzification and deffuzification are the crucial phases in a fuzzy system. Through membership function, the fuzzification phase transforms crisp or precise input values into fuzzy sets and uses identities to modify and capture the inherent vagueness in the input data. The defuzzification phase involves mapping of fuzzy output or fuzzy variables, to crisp, precise information that can be easily interpreted or understood, the stage can also involves aggregation of fuzzy output to produce a precise value based on the inference rule or results. Various methods of defuzzification have been reported in the literature, and each of which has been successful with negligible drawbacks. Some of the defuzzification methods found in the literature include alpha-cut, centroid, maxima, and so on. The choice of defuzzification method depends on the specific applications and task. Alpha-cut defuzzification is one of the defuzzification techniques used in fuzzy logic to convert a fuzzy set into a crisp output. It selects a subset of the fuzzy set based on a threshold alpha level (α). The alpha-cut method helps in situations where the most relevant elements of a fuzzy set is vital by focusing on a certain degree of membership. An α-cut defuzzification, often denoted by ACD. If μ is the Membership Function (MF), the α-cuts are points in the universe of discourse denoted by $[\mu]_\alpha$. The Average Cumulative Degree (ACD) for discrete data is computed using:

$$\text{ACD} = \frac{\sum_{i=1}^{N} \alpha_i \overline{[\mu]}_{\alpha_i}}{\sum_{i=1}^{N} \alpha_i}, \tag{34}$$

where $\overline{[\mu]}_{\alpha_i}$ is the average of the α-cut along the horizontal axis, and N is the number of discretization levels for different α values along the vertical axis. However, for a single α, the set of values on the x-axis whose membership values are greater than or equal to α are taken as the α-cut. This makes it more applicable to optimisation tasks featuring many false minima, like HNN, by dynamically navigating through the energy landscape. Additionally, various researchers have asserted the effectiveness of using simulated annealing in resolving discrete optimisation tasks, including.

5. Embedding simulated annealing within HNN

Simulated annealing is one of the oldest and most popular single-solution-based probabilistic optimisation techniques developed by [48]. It was built to mimic metallurgical annealing of materials, where the temperature of the material is raised quickly to a high temperature and cooled gradually to produce a well-aligned and low-energy crystalline structure. It is an iterative, improvement-based process that explores the search space in a more intelligent and systematic manner. It is well-regarded for its suitability for optimization tasks due to its resistance to local minima, achievable through probabilistically accepting worse results, which guides it towards an optimal or near-optimal solution in the search space [49]. This makes it more applicable to optimisation tasks featuring many false minima, like HNN, by dynamically navigating through the energy landscape. Additionally, various researchers have asserted the effectiveness of using simulated annealing in resolving discrete optimisation tasks, including satisfiability [12, 50]. Implementation of simulated annealing in HNN can be carried out using the following steps:

The algorithm commences with a randomly generated solution S, which is a vector of -1s and 1s, with its length determined by the number of variables in the SAT problem. The fitness of the initial solution is then evaluated using the fitness function $E_{\mathrm{B_{2SAT}}}(S)$, as defined in equation (15). A new solution S' is then generated randomly in the neighbourhood of the current solution S by making a random change to the current solution. Typically, a new solution is generated by flipping the value of a single binary variable. The fitness $E_{\mathrm{B_{2SAT}}}(S')$ is then computed based on a cost function, which is a minimization problem. If the difference $\Delta = E_{\mathrm{B_{2SAT}}}(S') - E_{\mathrm{B_{2SAT}}}(S)$ is less than 0, i.e., $\Delta < 0$, the transition to the new solution is accepted. If $\Delta > 0$, the transition to the new solution can be accepted probabilistically, often determined using the acceptance probability function $\exp(-\Delta/T)$, which is compared with a randomly generated number in the interval $(0,\ 1)$. This technique facilitates local minima evasion and guides the network towards a better solution. The parameter T is gradually decreased using a cooling function given by $T_{n+1} = \alpha T_n$ [51]. As the algorithm progresses, the parameter T facilitates the possibility of accepting worse solutions early in the process, allowing the algorithm to explore a larger portion of the search space, which decreases as the T decreases, allowing fewer worse solutions to be accepted over time. The procedure is repeated for another neighbour generation, fitness evaluation, and probabilistic acceptance steps until a stopping criterion of the maximum number of iterations is reached. The final solution should ideally satisfy the maximum number of clauses. The potential solution from SA is then fed into the HNN initialisation with connection weights from the Wan Abdullah method, and the network will evolve and converge to a stable state that is likely closer to the optimal solution due to the energy minimisation process inherent in HNNs. The network states whose energies deviate slightly from optimal configuration are classified as the global minimum solution; otherwise, they are regarded as the local minima solution. The flowchart in Figure 1 illustrates the step-by-step procedure of the simulated annealing algorithm, outlining its key components and decision-making points. Accompanying the flowchart is the pseudocode used in constructing the algorithm, which provides a concise representation of the logic and iterative procedures involved in the simulated annealing approach.

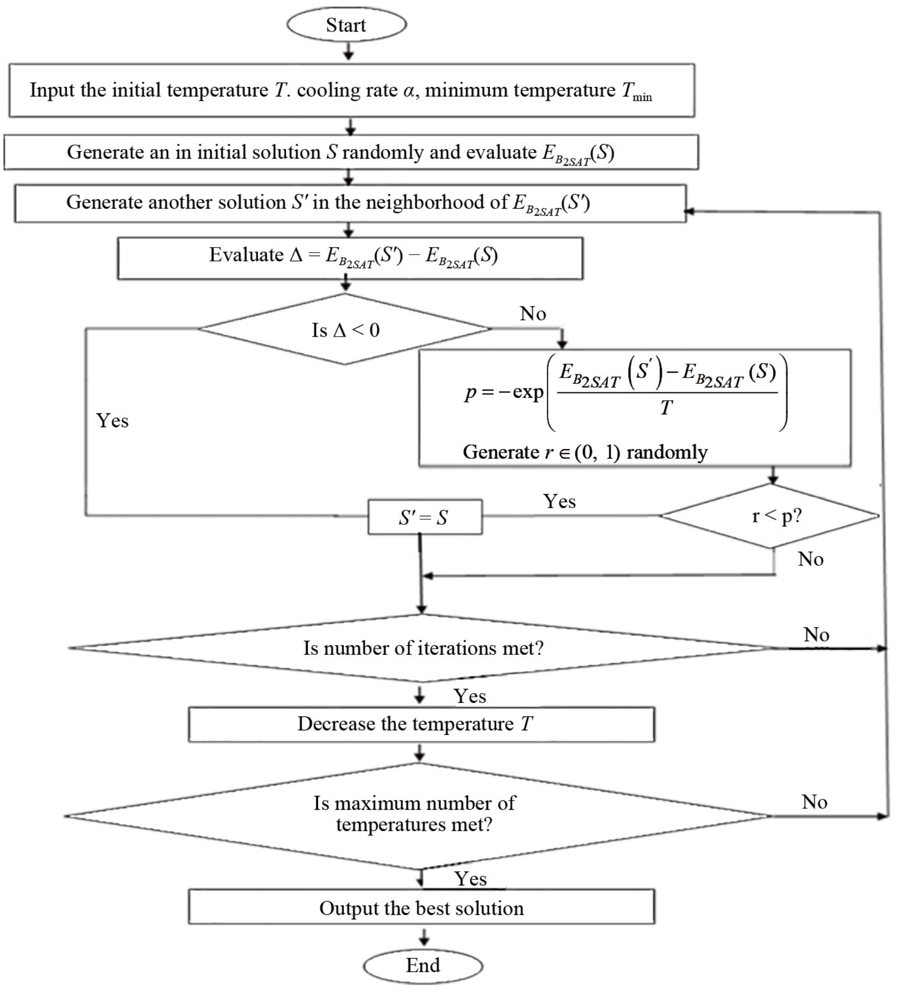

Start
Input the initial temperature T. cooling rate α, minimum temperature $T_{\min}$
Generate an in initial solution S randomly and evaluate $E_{B_{2SAT}}(S)$
Generate another solution S' in the neighborhood of $E_{B_{2SAT}}(S')$
Evaluate $\Delta = E_{B_{2SAT}}(S') - E_{B_{2SAT}}(S)$
Is $\Delta < 0$
No
Yes
$p = -\exp\left(\frac{E_{B2SAT}(S') - E_{B2SAT}(S)}{T}\right)$
Generate $r \in (0, 1)$ randomly
$S' = S$
Yes
r < p?
No
Is number of iterations met?
No
Yes
Decrease the temperature T
Is maximum number of temperatures met?
No
Yes
Output the best solution
End

Figure 1. Flowchart of the simulated annealing

Simulated Annealing Procedure

Start

Step 1: Input cooling rate (α), cost function, initial solution, cost function, temperature T, minimum temperature.

Step 2: Initialise a random potential solution (solutions with $S_\alpha \geq \alpha$), evaluate the cost function **while** temperature $T >$ minimum temperature **do:**

• Generate a new solution in the neighbourhood of initial solution by flipping one of the neuron states.

• Calculate the cost function using $E_{B_{2SAT}}(S)$ and check the changeover.

Step 3: Calculate the change in cost using $\Delta = E_{B_{2SAT}}(S') - E_{B_{2SAT}}(S)$.

1. Accept the new solution if $\Delta < 0$.

2. Based on; if $\Delta < 0$ or rand $(0,\ 1) < 0$ or $\exp\left(-\frac{\Delta}{T}\right)$.

Accept the solution if $E_{B_{2SAT}}(S') < E_{B_{2SAT}}(S)$ or rand $(0,\ 1) < \exp\left(-\frac{\Delta}{T}\right)$.

Step 4: Update the best solution when the condition is satisfied.

```
If current cost = best cost then
best solution = current solution
best cost = current cost
```

```
end if
    Decrease the temperature
temperature = temperature * cooling rate
end while
    Step 5: Return the best solution found.
return best solution
    End
```

6. GWO algorithm

Grey Wolf Optimizer (GWO) belong to population-based metaheuristic optimisation algorithms that portrays the social and hurting behaviour among grey wolves. GWO was proposed by Seyedali Mirjalili in 2014 [52]. The Canidae is the biological family name linked to dog-like carnivorous creatures in the animal kingdom, no doubt the grey wolf belongs to this family and occupies the apex predator's position in the biological food chain. Grey wolves live in groups called packs; a pack is made up of 5-12 wolves on average. Behaviour in the pack showcases dominance, separation of power and duties among the various components of the pack. The pack is headed by a male or female wolf called Alpha (σ), who is saddled with major decision-making in the pack. Alpha wolves command much respect, which is obviously seen when in the company of other members of the pack [53]. Beta (β) wolf is next to alpha wolf. They assist alpha wolves in making decisions and participate in other pack activities. The beta wolf, which can be either male or female, is most likely the best choice to replace an alpha in the event that the alpha wolf becomes incapacitated or dies. The beta wolf not only commands other subordinate provide feedback to the alpha wolves while also reinforcing the alpha's orders within the pack. Omega (ω) Grey wolf sect occupies lowest rank and often serve as scapegoats. Omegas are the lowest-ranking wolves, displaying complete submission to all dominant wolves. They are the last to eat and, despite appearing less significant, their absence can cause unrest in the pack. Omegas help care for wounded, young, and incapacitated pack members. Wolves that don't belong to the alpha, beta, or omega ranks are classified as subordinate deltas (δ), submissive to both alphas and betas but dominant over omegas. The delta group includes guards such as moles, sentinels, elders, hunters, and carers. Moles monitor the territory's borders and alert the pack to danger, while sentinels ensure the pack's security. Elders are experienced former alphas or betas, and hunters assist in capturing prey for the pack as food. The weak, sick pups, and injured wolves in the pack are taken care of by the caretakers.

We implement GWO algorithms to improve HNN's ability to solve 2-SAT problems. Similar to most population-based metaheuristic algorithms, the process begins with a set of random candidate solutions (wolves), and the population size determines the number of wolves in the population at each iteration. A large population size can lead to better exploration, but at the expenses of computational complexity. When using GWO as the optimization tool, the fittest solution is regarded as alpha (σ), beta (β) and delta (δ) which represent the second and third best solutions, respectively, while the rest are considered omega (ω). To model GWO mathematically, the encircling behaviour of a pack hunting a prey, equation (35)-(38) are employed:

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}\vec{D}, \tag{35}$$

where t denotes iteration number, $\vec{A}$ and $\vec{C}$ are position vectors for exploration and exploitation, respectively. The coefficient vectors, $\vec{X}_p(t)$ and $\vec{X}(t)$ are position vectors of grey wolf and prey respectively;

$$\vec{D} = \left|\vec{C}\vec{X}_p(t) - \vec{X}(t)\right|. \tag{36}$$

$\vec{A}$ and $\vec{C}$ are shown in equation (38) and (39)

$$\vec{A} = 2ar_1 - a, \tag{37}$$

$$\vec{C} = 2\vec{r}_2, \tag{38}$$

r_1 and r_2 denote random variables in the interval $[0,\ 1]$ that added randomness to the search, a is a lineally decreasing variable from 2 to 0, a tuning parameter for better exploration and exploitation. It is calculated for each iteration using the relation in equation (39):

$$a = 2\left[1 - \frac{t}{\text{maxiter}}\right], \tag{39}$$

The modified version used in this research takes the form:

$$a = 2\left[1 - \left(\frac{t}{\text{maxiter}}\right)^2\right]. \tag{40}$$

7. Optimizing HNN using GWO algorithm

Grey Wolf Optimization (GWO) algorithm for optimisation of network state begins with random generation of an initial population of wolves (solutions). Each wolf represents a possible solution to the SAT problem, consisting of binary variables (-1 or 1). Suppose each of the solutions, $\vec{X}_n$ has the form shown in equation (41):

$$\vec{X}_i = S_1^i,\ S_2^i,\ S_3^i,\ \ldots,\ S_N^i, \tag{41}$$

where N is the length of strings corresponding to the number of variables in the underlying logic program, and $S_i \in -1,\ 1$ represents the truth value. Each wolf $\vec{X}n$ (potential solution) in the search space is evaluated using the fitness function in equation (41):

$$E_{B2SAT}(\vec{X}i) = \frac{1}{2^2}\sum_{i=1}^{nC}\left(\prod_{j=1}^{2} C_{ij}\right)_j, \tag{42}$$

where nC denotes the number of clauses that are in the SAT problem, i denotes the clause number, j represents the index of a variable in clause C_{ij} represents the value in clause i with index j, and the superscript "2" denotes the size of the clause. $C_{ij} = (1-S_i)(1-S_j),\ (1-S_i)(1+S_j),\ (1+S_i)(1-S_j),\ (1+S_i)(1+S_j)$ and it depends on the sign of the literals in the clauses. The fitness of each potential solution is evaluated, and the solutions are sorted in descending order since our cost function is a minimization function. The best three solutions corresponding to the least three values are assigned alpha

(α), beta (β), and delta (δ), and are designated as $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$. The rest are omega (ω) wolves, denoted as $\vec{X}\omega$. Their respective positions are updated and the distance function is modified based on the class of the wolf in the pack.

The respective distance functions are evaluated using equations (42)-(44):

$$\vec{D}_\alpha = |C_1\vec{X}_\alpha(t) - \vec{X}(t)| \tag{43}$$

$$\vec{D}_\beta = |C_2\vec{X}_\beta(t) - \vec{X}(t)| \tag{44}$$

$$\vec{D}_\delta = |C_3\vec{X}_\delta(t) - \vec{X}(t)|. \tag{45}$$

The corresponding positions are updated using equations (45)-(47):

$$\vec{X}_1 = \vec{X}_\alpha(t) - \vec{A}1|C_1\vec{X}_\alpha(t) - \vec{X}(t)| \tag{46}$$

$$\vec{X}_2 = \vec{X}_\beta(t) - \vec{A}2|C_2\vec{X}_\beta(t) - \vec{X}(t)| \tag{47}$$

$$\vec{X}_3 = \vec{X}_\delta(t) - \vec{A}3|C_3\vec{X}_\delta(t) - \vec{X}(t)|. \tag{48}$$

The best solution is obtained by taking the average of:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}. \tag{49}$$

Although GWO is designed for continuous search spaces, there are various methods of converting continuous spaces to binary ones, such as using transfer functions, allowing continuous metaheuristic algorithms to solve discrete optimisation problems [54]. This study deploys the use of a sigmoid transfer function, which is defined in equation (50):

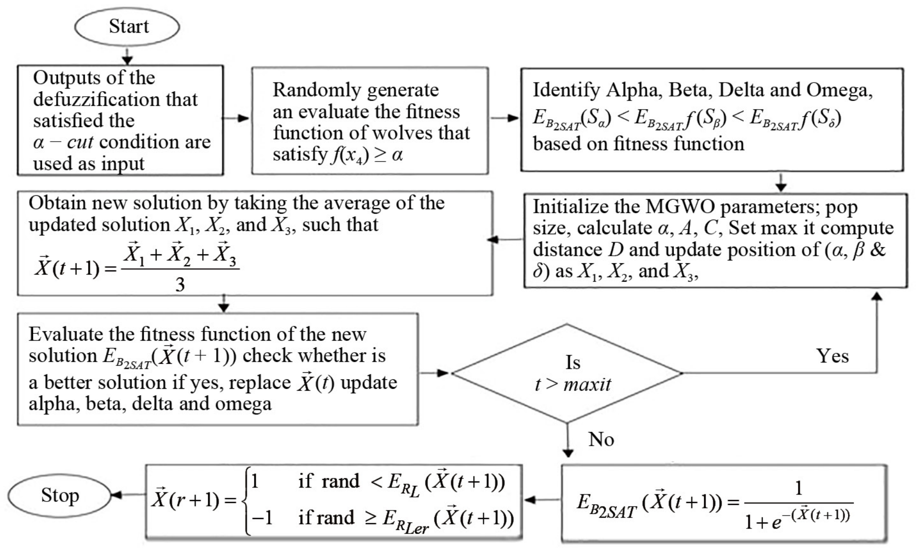
$$E_{B_{2SAT}}(\vec{X}(t+1)) = \frac{1}{1+e^{-\vec{X}(t+1)}}, \tag{50}$$

such that:

$$\vec{X}(t+1) = \begin{cases} 1 & \text{if rand} < E_{B_{2SAT}}(\vec{X}(t+1)) \\ \\ -1 & \text{if rand} \geq E_{B_{2SAT}}(\vec{X}(t+1)). \end{cases} \tag{51}$$

where rand $\in (0,\ 1)$ and $\vec{X}(t+1)$ is the updated position. The mechanism described in equations (50) and (51) ensures that the algorithm can handle binary representation in satisfiability problems effectively. We then evaluate $\vec{X}(t+1)$ to see how well it performs. The solution is compared with the previous solution $\vec{X}(t)$ and if there is an improvement, we update $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$, decrease the tuning variable a, and repeat the whole process again for the next iteration to obtain an improved $\vec{X}(t+1)$. Thus, $\vec{X}(t+1)$ will be taken as the final solution if the algorithm satisfies the stopping criteria.

The solution from GWO is then fed into the HNN, initialised with connection weights computed using the Wan Abdullah method. The network then evolves and converges to a stable state that is closer to the optimal solution due to the energy minimisation process inherent in HNNs. The network states whose energies are within the tolerance value are branded as global minima; otherwise, they are local minimum.



Start
Outputs of the defuzzification that satisfied the $\alpha - cut$ condition are used as input
Randomly generate an evaluate the fitness function of wolves that satisfy $f(x_4) \geq \alpha$
Identify Alpha, Beta, Delta and Omega, $E_{B_{2SAT}}(S_\alpha) < E_{B_{2SAT}}f(S_\beta) < E_{B_{2SAT}}f(S_\delta)$ based on fitness function
Initialize the MGWO parameters; pop size, calculate α, A, C, Set max it compute distance D and update position of (α, β & δ) as X_1, X_2, and X_3,
Obtain new solution by taking the average of the updated solution X_1, X_2, and X_3, such that $\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$
Evaluate the fitness function of the new solution $E_{B_{2SAT}}(\vec{X}(t+1))$ check whether is a better solution if yes, replace $\vec{X}(t)$ update alpha, beta, delta and omega
Is $t > maxit$
Yes
No
$E_{B2SAT}(\vec{X}(t+1)) = \frac{1}{1+e^{-(\vec{X}(t+1))}}$
$\vec{X}(r+1) = \begin{cases} 1 & \text{if rand } < E_{R_L}(\vec{X}(t+1)) \\ -1 & \text{if rand } \geq E_{R_{Ler}}(\vec{X}(t+1)) \end{cases}$
Stop

Figure 2. Flowchart of the modified grey wolf optimization algorithm

The pseudocode used in constructing the algorithm is as follows.

```
Start
Step 1: Set all the parameters.
Step 2: Initialize grey wolves' positions population randomly from the output of the fuzzy logic.
Step 3: Calculate fitness for each wolf based on the number of satisfied clauses Identify and set Alpha, Beta, and Delta positions based on fitness (3 best results).
Step 4: Repeat until convergence criteria met:
• Update position of each wolf based on alpha, beta, and delta positions.
• Calculate fitness for each updated position.
• Update alpha, beta, and delta positions based on new fitness values.
• Update exploration and exploitation rates.
Return the best position found.
End
```

8. Synergizing fuzzy logic, simulated annealing, and grey wolf optimization for 2-SAT logic programming in hopfield neural network

Optimal synaptic weights enhance the Hopfield Neural Network (HNN) with numerous benefits, including improved convergence, effective pattern completion and retrieval, stability, dynamic behaviour, and an efficient energy minimisation process, among others [55]. These attributes have contributed immensely to the success of HNN in solving complex combinatorial optimisation problems in various domains. However, there hasn't been much research on the combination of fuzzy logic and HNN in solving satisfiability problems. We therefore explore the possibilities of developing a model that will incorporate fuzzy logic with HNN to optimise satisfiability problem solutions. Our approach in this research begins with the random generation of the set of solutions $\{S_i\}_{i=1}^{n}$ to the SAT problem, such that each $S_i \in \{S_i\}_{i=1}^{n}$ is the potential solution to the SAT problem, S_i is a vector consisting of Boolean values, whose length corresponds to the number of variables in the SAT problem. Each S_i can be written as: $S_i = (S_1^i,\ S_2^i,\ \ldots,\ S_k^i)$ where S_j^i are Boolean values and k corresponds to the number of variables in the SAT problem. Each potential solution is evaluated using the fitness function defined in equation (52).

$$f(S_i) = \sum_{j=1}^{nC} (C_{ij},\ S_i), \tag{52}$$

where $S_i = (S_1^i,\ S_2^i,\ \ldots,\ S_k^i)$, $S_j^i \in \{0,\ 1\}$, nC is the number of the clauses, C_{ij} represents the ith clause in the 2-SAT problem and $(C_{ij},\ S_i)$ represents a function that evaluates to 1 if a clause C is satisfied and 0 otherwise. For each $S_i \in \{S_i\}_{i=1}^{n}$, $f(S_i)$ is calculated to know the performance of each of the assignments. However, brute force offers a straightforward approach of solving 2-SAT problem but could be problematic due to its exponential growth and order of $\mathscr{O}(2^k)$ [26].

This called for random generation of set of solution whose search space is less than $\mathscr{O}(2^k)$. The result of fitness function evaluation serves as performance indicator of each of the assignments. Therefore, equation (52) reduces to:

$$f(S_i) = u_i. \tag{53}$$

where u_i is number of clauses satisfied by the corresponding vector S_i, we fuzzified the performance of each assignment S_i using a linear function defined in equation (53), it maps the outputs to a range of values between 0 and 1, representing the degree of membership. Low-performing solutions, which satisfy fewer clauses, are mapped closer to 0, while high-performing solutions, satisfying more clauses, are mapped closer to 1.

$$\mu_A(S_i) = \begin{cases} 0 & \text{if } f(S_i) \leq a \\ \dfrac{f(S_i) - a}{b - a} & \text{if } a \leq f(S_i) \leq b \\ 1 & \text{if } f(S_i) \geq b, \end{cases} \tag{54}$$

where $a,\ b$ represent the lower and upper bound and $_A(S_i)$ is the membership value of each of the potential solutions. This function transforms count into fuzzy values. For instance, a solution that satisfying few numbers of clauses might have a value of 0.1, indicating poor assignment, while a solution satisfying nearly all clauses might have a value of 0.9 or higher, indicating high performance. This fuzzified evaluation helps in guiding the algorithm towards better solutions by emphasising the quality of the solutions rather than a binary satisfied/unsatisfied criterion.

We then defuzzified using α-Cut, a technique where the fuzzy sets are converted into crisp values based on a certain degree of membership $(\alpha\ levels)$. α-Cut defuzzification technique divides the fuzzy set into two disjoint sub-fuzzy sets based on a specified threshold α. This will partition the fuzzy set into "lower" and "upper" sub-fuzzy sets. This process helps in analysing the distribution solution in the search space. The set of solutions that satisfy α-Cut condition defined as:

$$S_\alpha = \{S_i \in \{S_i\}_{i=1}^{n} \mid \mu(S_i) \geq \alpha\}, \tag{55}$$

are considered to be solutions with higher quality and are mapped to 1. Those that failed the condition will be mapped to 0. The value of α determines the size of the solutions that are defined to be desirable ones. The higher the value of α within the interval $0 \leq \alpha \leq 1$, the lower the number of the desirable solutions. The use of alpha-cut defuzzification facilitate the usage of only high-quality solutions during the training phase of HNN, effectively verifying and identifies of optimal solutions while excluding poorly performing ones during learning phase. Performance metrics, energy analysis, and computational time are used to compare the outcome of the fuzzy implementation. The optimal solutions from the defuzzification process are further optimised using SA and MGWO, as discussed in previous sections. The figure below illustrates the flowchart for the procedure.

The flowchart in Figure 3 illustrates the step-by-step procedure of the simulated annealing algorithm, outlining its key components and decision-making points. Accompanying the flowchart is the pseudocode used in constructing the algorithm, which provides a concise representation of the logic and iterative procedures involved in the simulated annealing approach

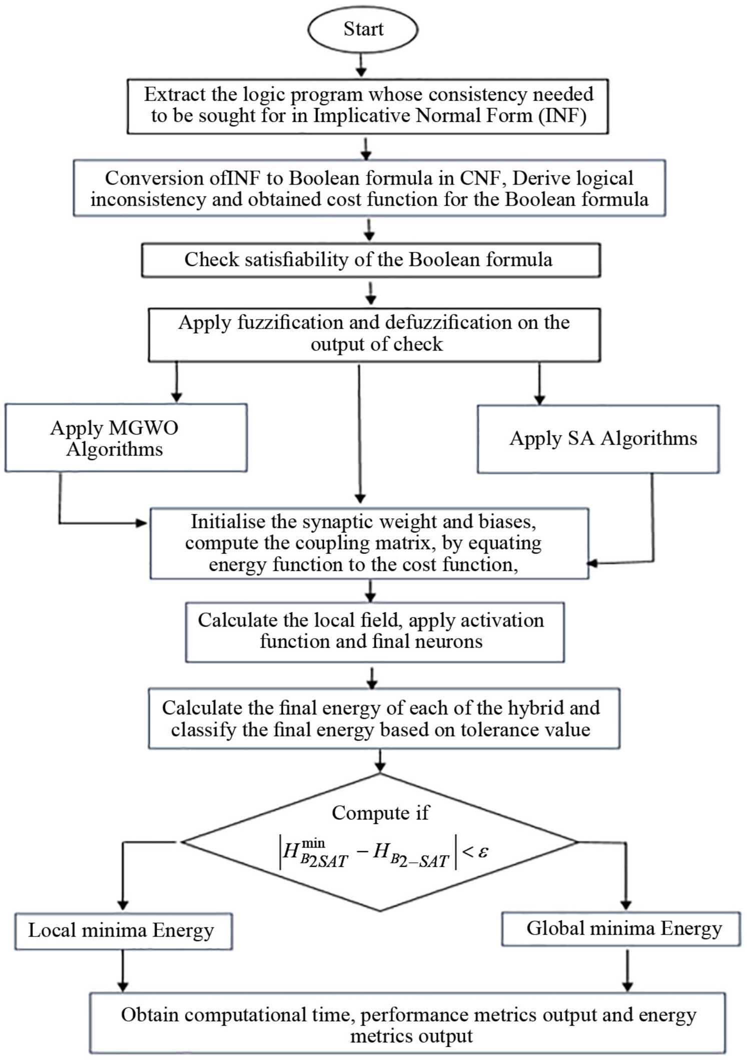

Start
Extract the logic program whose consistency needed to be sought for in Implicative Normal Form (INF)
Conversion ofINF to Boolean formula in CNF, Derive logical inconsistency and obtained cost function for the Boolean formula
Check satisfiability of the Boolean formula
Apply fuzzification and defuzzification on the output of check
Apply MGWO Algorithms
Apply SA Algorithms
Initialise the synaptic weight and biases, compute the coupling matrix, by equating energy function to the cost function,
Calculate the local field, apply activation function and final neurons
Calculate the final energy of each of the hybrid and classify the final energy based on tolerance value
Compute if $\left|H_{B2SAT}^{\min} - H_{B2-SAT}\right| < \varepsilon$
Local minima Energy
Global minima Energy
Obtain computational time, performance metrics output and energy metrics output

Figure 3. Flowchart showing the implementation of the three hybrids

9. Parameterization and experimental setup

We implemented all the hybrids HNN-2SATFuzzy, HNN-2SATFuzzySA, and HNN-2SATFuzzyMGWO using MATLAB (2023a) on a system running Windows 10, with an Intel Core i3 processor at 2.90 GHz and 8 GB of RAM. The number of variables (NN), represented by neurons, ranged from 6 to 240. For each model, the experiment was conducted by varying the number of neurons at each iteration within the specified range. The datasets used were simulated and generated to produce clauses with exactly 2 literals each, aligning with the logical structure of the hybrid models. The choice of parameters for each hybrid model was based on trial and error, selecting combinations that yielded the best results while minimising errors.

Tables 1-3 summarise the optimal parameter configurations for HNN, Simulated Annealing (HNN-2SATFuzzySA), and the modified Grey Wolf Optimisation (HNN-2SATFuzzyMGWO) algorithms, respectively

Table 1. Experimental configuration and parameters for HNN-2SAT model

Description of parameters	Numerical value
Length of clauses	2
Number of variable (neurons)	$6 \leq NN \leq 240$
Activation function	HTAF
Dataset generation method	Simulated
Maximum Combination (MaxCOMB)	100
Tolerance limit	0.001
Number of trials	100
CPU threshold time	2 hours
Synaptic weights computation method	Wan Abdullah method

Table 2. Parameters for simulated annealing used in the research

Description of parameters	Numerical value
Maximum iteration	1,000
Initial temperature	2
Final temperature	0
Cooling factor	0.99
Cooling schedule	temp = Cooling factor $\times$ temp

Table 3. Parameters for grey optimization algorithms in the research

Description of parameters	Numerical value
Maximum iteration	1,000
Number of variables	$6 \leq NN \leq 240$
Number of solutions	40
r_1, r_2	Random (0, 1)
Initialization method	Random
Coefficient vector (A, C)	Computed from a, r_1, r_2
Search range	{-1, 1}

10. Performance evaluation metrics

Performance metrics are indicators deployed to evaluate and assess the efficiency of a model. These metrics offer valuable information about the quality of output or outcome in an iterative exploration or search procedure. To evaluate the effectiveness of our models, we utilise the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Sum of Square Error (SSE), and Symmetric Mean Absolute Percentage Error (SMAPE).

10.1 *RMS*

The Root Mean Square Error (RMSE) is particularly prominent in regression analysis and machine learning, which aim to predict continuous numeric value [56]. It is the average magnitude of the difference between actual and predicted value. The range of output of RMSE is $[0, +\infty]$. A lower RMSE signifies better model performance, while a higher RMSE indicates worse performance. Chicco et al. [57] asserted that RMSE is highly sensitive to outliers, which can significantly inflate the error measurement. The RMSE is calculated in this study using equation (54).

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{1}{n} (\phi_{\max} - \phi_i)^2}, \tag{56}$$

where $\phi^{\max}$ defines the maximum fitness, ϕ_i denotes the maximum fitness from the network, and n is the number of data points.

10.2 *MAE*

Mean Absolute Error (MAE) is another widely used performance metric for model evaluation; it evaluates the accuracy and effectiveness of predictive models and shares an application domain with RMSE. It calculates the average absolute difference between actual values and predicted values. Equation (55) is used in calculating the MAE.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n\Sigma} |\phi_{\max} - \phi_i|, \tag{57}$$

where $\phi_{\max}$ and ϕ_i denotes the maximum fitness and the maximum fitness from the network, respectively, n is the number of solutions, and $|.|$ denotes the absolute value symbol. MAE is more robust and less sensitive to outliers; each absolute difference adds to the error and has a range of outputs in $[0, +\infty]$. MSE is mostly utilized in instances where datasets outliers could render a metric like RMSE unsuitable [57].

10.3 *SSE*

Sum Squared Error (SSE) calculates the difference between the actual and observed values. It evaluates the sum of the squared differences between each actual value and the corresponding observed value. SSE is sensitive to outliers, Wang and Lu [56], reported that large errors contribute disproportionately to SSE. It is computed using the formula below.

$$\text{SSE} = \sum_{i=1}^{n\Sigma^2} (\phi_{\max} - \phi_i), \tag{58}$$

where $\phi_{\max}$ and ϕ_i are the maximum fitness and the maximum fitness from the network, and. Its description of the domain and features is similar to that of the two metrics previously discussed; it can be scaled to normalise by the number of data points.

10.4 *SMAPE*

Another vital performance indicator for assessing the precision of a forecasting or predicting technique is the Symmetric Mean Absolute Percentage Error (SMAPE), which is especially relevant when analysing time series and machine learning. It is the Mean Absolute Percentage Error (MAPE) in symmetric form. Because it overcomes some of MAPE's shortcomings, particularly when handling situations in which the real values are near zero, SMAPE is a superior variant of MAPE. According to [57] the symmetric characteristic of SMAPE means that if the predicted and actual values are switched, the resulting SMAPE value remains unchanged. Since it is indifferent to qualities, it is a better option than MAPE. It is easily interpretable, expressed as a percentage, and remains unaffected by scale changes. It is calculated using equation (57)

$$SMAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{\phi_{\max} - \phi_i}{(|\phi_{\max} i| + |\phi_i|)/2} \right|. \tag{59}$$

10.5 *Energy analysis*

For most optimisation problems and search techniques, we can evaluate an algorithm's effectiveness in locating the global minimum in the energy contour using the global minima ratio. The energy landscape of the Hopfield neural network is characterised by several local minima. The lowest point on the energy contour often represents the optimal solution [2]. However, our proposed models are tested and evaluated for efficacy using energy analysis. We compute the global minimum ratio by dividing the total global minimum energy by the total number of runs.

$$Global\ minima\ ratio = \frac{Total\ global\ minimum\ energy}{Total\ number\ of\ run}. \tag{60}$$

Sathasivam et al. [58] facilitate better search for network state by refining the search space with the number of combinations (γ) and number of trials (ε) and crafted the formula for global minimum ratio and local minima ratio as shown in equation (59) and equation (60), respectively.

$$Z_m = \frac{1}{\gamma\varepsilon} \sum_{i=1}^{n} z_i, \tag{61}$$

$$Y_m = \frac{1}{\gamma\varepsilon} \sum_{i=1}^{n} y_i, \tag{62}$$

where z_i and y_i represent total global minimum and local minimum energy respectively. Z_m can be any real number between 0 and 1, and higher Z_m demonstrates the goodness of the optimisation algorithm in successfully finding lower energy across multiple runs.

10.6 *Computational time*

Computational time in neural networks refers to the time required for a neural network to analyse and evaluate input, in HNN it encompassing both the training and inference phases. The formula for finding the CPU time in HNN as presented by Sathasivam et al. [58].

$$CPU\ time = Training\ time\ (s) + Testing\ time\ (s). \tag{63}$$

A good CPU time is an indication of an effective model, such a model completes both learning and inference phases within a reasonable amount of time and is often considered a better option once it maintains such a good time while producing reasonable results.

11. Result discussion and analysis

The comparative performance analysis of the three proposed hybrid models, HNN-2SATFuzzy, HNN-2SATFuzzySA and HNN-2SATFuzzyMGWO with the standalone HNN-2SAT is carried out using simulated data, this aims at identifying the most effective model in solving the 2-SAT problem. The performance was carried out using RMSE, MAE, SSE, and SMAPE. The accuracy of the model was done using energy analysis. The energy of the network state in the CAM, where all clauses are satisfied, serves as the baseline for identifying optimal solutions. CPU time was deployed to investigate the balance between speed and efficiency of the models. The models' effectiveness is further tested by varying the number of neurons from 6 to 240. Figures 4 and 5 depict the graph of RMSE and MAE for the three hybrids.

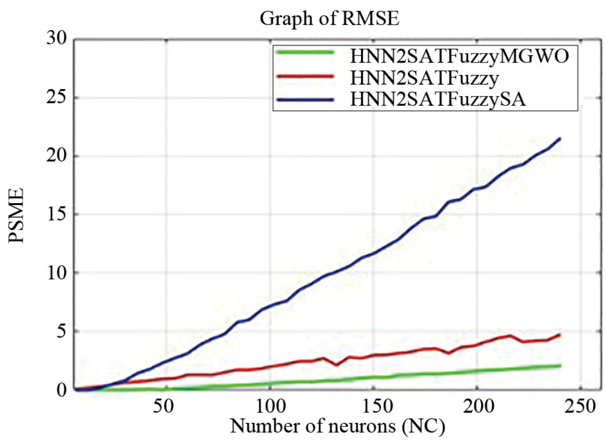


Figure 4. The graph of RMSE for the three hybrids

Figure 4 shows different RMSE values for each of the hybrid models under consideration. The RMSE in each of the hybrids increases as NC increases. HNN2SATFuzzySA lags behind the other models, this could be attributed to the inherent limitations of Simulated Annealing (SA) in handling high-dimensional data structures. Although HNN2SATFuzzySA maintains consistent performance as the simulation progresses. But it cannot match the effectiveness of the other two hybrids. The HNN2SATFuzzyMGWO model consistently outperforms the other two hybrids, showing superior performance due to the effectiveness of the Modified Grey Wolf Optimisation (MGWO) algorithm, which enhances the search process by iteratively replacing poor solutions with better ones and guides the solutions toward optimal regions within the search space, improving search efficiency and convergence. Figure 5 closely mirrors the RMSE graph in Figure 4, demonstrating the similarity in their formulations and result. While both metrics are effective in measuring error, MAE treats all errors equally, regardless of their magnitude or direction, providing better interpretability, computational simplicity, and greater resistance to outliers. Chai and Draxler [59] assert that machine learning often favours MAE over RMSE. The HNN2SATFuzzyMGWO lowest MAE values demonstrate minimal deviation from the desired output. It is followed by the HNN2SATFuzzy model, which shows reasonable alignment with the target configuration. Meanwhile, the HNN2SATFuzzySA model exhibits the highest MAE, reflecting its lower performance.

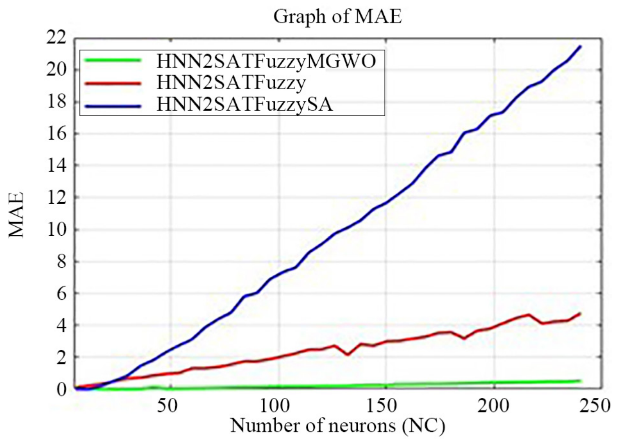


Figure 5. The graph of MAE for the three hybrids

Table 4. Comparative analysis of RMSE and MAE values for the 3 hybrid models with a standalone hybrid HNN2SAT proposed in [9]

NN	HNN2SAT		HNN2SATFuzzy		HNN2SATFuzzySA		HNN2SATFuzzyMGWO	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
18	2.287774	1.846285	0.347093	0.346919	0.209895	0.209790	0	0
30	3.883821	3.456849	0.652405	0.652079	0.209895	0.209790	0.003780	0.001429
42	5.564778	5.173571	0.855800	0.855372	0.849575	0.849150	0.035538	0.007593
54	7.089714	6.702856	1.023098	1.022586	1.829085	1.828170	0.029667	0.125159
66	8.632842	8.256275	1.320482	1.309787	2.758620	2.757240	0.203049	0.045958
78	10.106831	9.734722	1.544423	1.543650	3.878060	3.876120	0.310748	0.073408
90	11.613027	11.239000	1.750414	1.749539	4.807594	4.805190	0.431041	0.107554
102	13.134047	12.759140	2.060003	2.058973	6.036979	6.033960	0.529972	0.123280
114	14.613235	14.237720	2.476095	2.474856	7.336329	7.332660	0.650988	0.158137
126	16.135321	15.755090	2.720489	2.719129	8.565714	8.561430	0.708682	0.163598
138	17.613884	17.233070	2.821882	2.820471	9.755119	9.750240	0.878416	0.201902
150	19.103042	18.724980	3.000361	2.998861	10.574709	10.569420	1.020093	0.250618
162	20.586629	20.204180	3.171114	3.169528	11.684154	11.678310	1.121443	0.255548
174	22.116556	21.735670	3.518160	3.516401	12.893548	12.887100	1.296723	0.317975
186	23.608793	23.225520	3.180526	3.178935	14.632678	14.625360	1.369648	0.324702
198	25.097222	24.710860	3.791473	3.789577	16.071958	16.063920	1.543418	0.357304
210	26.605341	26.224390	4.459272	4.457042	17.141423	17.132850	1.688643	0.400305
222	28.112117	27.726440	4.126263	4.124199	18.260863	18.251730	1.844323	0.433354
234	29.607305	29.219230	4.292363	4.290216	19.280353	19.270710	1.973796	0.463476
240	30.372610	29.983970	4.757201	4.754822	20.599692	20.589390	2.036626	0.477032

Table 4 depicts the RMSE and MAE of the trends in Figures 4 and 5 and the RMSE and MAE of the standalone HNN2SAT proposed in [9]. HNN2SATFuzzyMGWO achieves the lowest RMSE and MAE values, demonstrating superior alignment with the optimal configuration. This performance highlights the effectiveness of the MGWO algorithm in refining solutions and efficiently guiding the search toward optimal regions. Similarly, the integration of fuzzy logic in HNN2SATFuzzy shows vital improvement over the standalone HNN2SAT model, which is prone to the local minima problem. All three hybrid models show lesser RMSE and MAE when compared with the standalone HNN2SAT. This demonstrates the effectiveness of enhancement in obtaining reasonable configuration when using HNN to solve satisfiability problems.

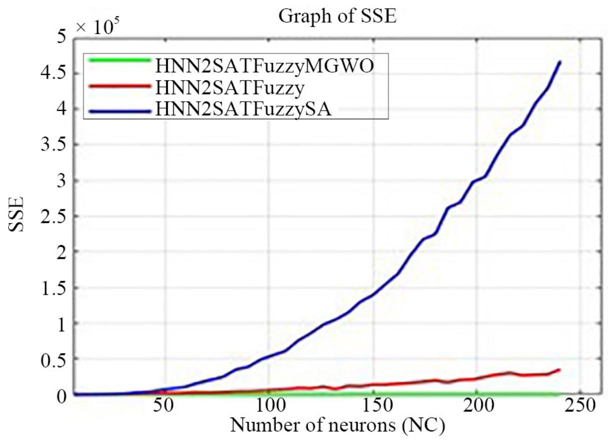


Figure 6. The graph of SSE for the three hybrids

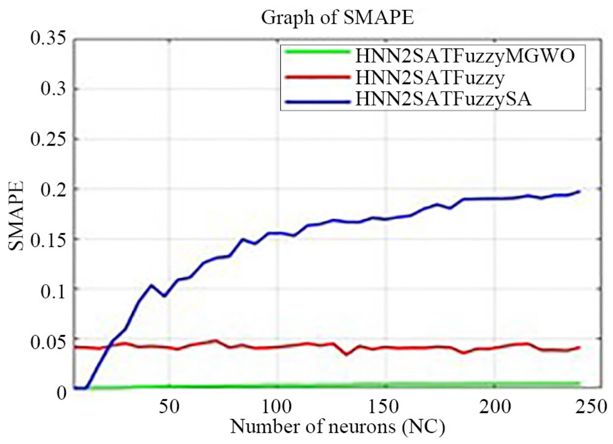


Figure 7. The graph of SMAPE for the three hybrids

Figure 6 depicts the Sum of Squared Errors (SSE). SSE quantifies the total squared error between the network's output and the baseline configuration. The result of SSE in the study aligns well with that of RMSE and MAE discussed above. The deviation of HNN2SATFuzzyMGWO is minimal, indicating that it once again affirms its outstanding performance. The same trend can be seen in Figure 7, where HNN2SATFuzzyMGWO consistently achieves the lowest SMAPE values throughout the simulation period. This reinforced our earlier findings using other performance metrics in the study, which consistently identified HNN2SATFuzzySA as the least effective hybrid model in the study.

Table 5. Comparative analysis of SSE and SMAPE for the 3 hybrid models with a standalone HNN2SAT proposed in [9]

NN	HNN2SAT		HNN2SATFuzzy		HNN2SATFuzzySA		HNN2SATFuzzyMGWO	
	SSE	SMAPE	SSE	SMAPE	SSE	SMAPE	SSE	SMAPE
18	5.525872	0.248051	179.7875	0.039694	209.790	0.024681	0.010	0.000168
30	15.531612	0.272204	617.6302	0.044894	1108.890	0.059207	0.200	0.000524
42	31.005031	0.288204	1113.7504	0.042040	3946.050	0.103226	0.880	0.001457
54	50.288051	0.288963	1642.0924	0.039039	8471.520	0.108309	1.890	0.001754
66	74.535405	0.290380	2852.1137	0.045476	16323.660	0.125498	3.650	0.002297
78	102.15533	0.288866	3570.5170	0.040795	24825.150	0.131986	6.520	0.002862
90	134.871299	0.288576	4680.0217	0.040074	38921.040	0.144487	10.380	0.002841
102	172.513720	0.288758	6174.4712	0.041600	55324.620	0.155287	14.460	0.003229
114	213.560580	0.287939	8972.1552	0.044850	75754.170	0.162871	17.630	0.002988
126	260.359490	0.288115	11000.1533	0.044597	97822.080	0.168186	26.800	0.003346
138	310.259730	0.287484	12054.6684	0.042188	114745.140	0.166285	33.940	0.003806
150	364.940480	0.287183	13691.7934	0.041245	139470.390	0.169205	43.800	0.003565
162	423.821500	0.286754	15222.0015	0.040330	169230.600	0.173155	53.220	0.004124
174	489.155390	0.287150	18086.7252	0.041654	217382.400	0.183832	62.100	0.003913
186	557.391220	0.286923	17149.8974	0.035209	261757.980	0.189338	80.600	0.004034
198	629.889860	0.286666	21265.1965	0.039400	298011.690	0.189949	94.680	0.004256
210	707.865750	0.286759	28289.1391	0.043775	337372.290	0.190622	111.540	0.004293
222	790.312880	0.286743	26889.8592	0.038284	376712.910	0.190396	128.490	0.004402
234	876.608310	0.286615	28456.9743	0.037743	429699.870	0.193230	139.030	0.004293
240	922.516330	0.286755	34885.9347	0.040879	467492.040	0.197022	145.880	0.004647

Table 5 depicts the performance of the three proposed hybrids in terms of SSE and SMAPE and it is contrasted with the standalone HNN2SAT model proposed in [9]. The network state produced by HNN2SATFuzzyMGWO shows much similarity with the ideal network state stored in the HNN's CAM.

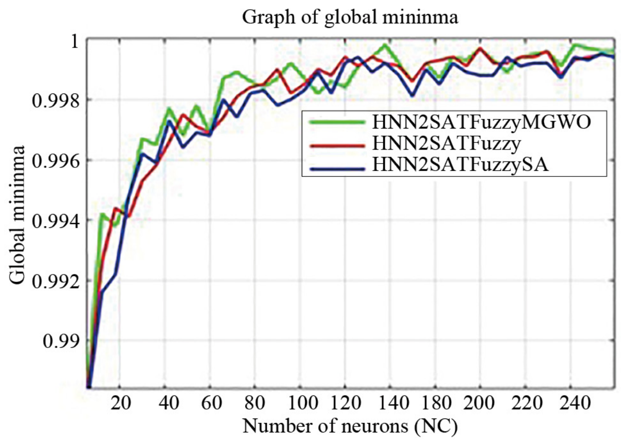


Figure 8. The graph of the global minimum ratio

The Global Minimum Ratio (GMR) is a critical performance metric used in the evaluation of optimisation algorithms, particularly for identifying global minimum in continuous or combinatorial optimisation problems. GMR quantitatively assesses the effectiveness of an algorithm by measuring its consistency in locating the global minimum across multiple independent runs. Figure 8 presents the GMR graph for the three proposed hybrid models under investigation. Equations (59) and (60) are the refined forms for global minimum and local minimum, respectively. Configuration that satisfies the condition $|Hconf_m in|$ is classified as a global minimum, where $H_m in$ is the minimum energy of the baseline configuration in the network (when all clauses are satisfied), and $H_c onf$ represents the energy of the configurations produced by the hybrid models. A GMR value of 1 indicates the highest reliability, demonstrating that the algorithm consistently finds the global minimum in every run, while a GMR of 0 indicates that the algorithm rarely locates the global minimum, reflecting poor performance. As illustrated in Figure 8, all three hybrid models exhibit relatively good performance. However, the hybrid HNN2SATFuzzySA underperformed relatively low when compared to the others in most instances. The HNN2SATFuzzy model achieved intermediate performance, HNN2SATFuzzyMGWO outperformed the other two, exhibiting higher GMR values across most runs. This suggests that the MGWO significantly improved the network states within the search space. A table of the comparison of the proposed model when compared with the standalone hybrid of HNN and 2SAT is shown in Table 6.

Table 6. Comparative analysis of GMR values for the 3 hybrid models with a standalone HNN2SAT

NN	HNN2SAT GMR	HNN2SATFuzzy GMR	HNN2SATFuzzySA GMR	HNN2SATFuzzyMGWO GMR
18	0.9936	0.9944	0.9952	0.9938
30	0.9957	0.9953	0.9962	0.9957
42	0.9970	0.9966	0.9973	0.9967
54	0.9969	0.9971	0.9969	0.9978
66	0.9974	0.9974	0.9980	0.9978
78	0.9975	0.9984	0.9982	0.9976
90	0.9971	0.9990	0.9978	0.9987
102	0.9972	0.9983	0.9985	0.9987
114	0.9981	0.9988	0.9982	0.9986
126	0.9983	0.9991	0.9994	0.9989
138	0.9989	0.9992	0.9998	0.9992
150	0.9983	0.9986	0.9991	0.9984
162	0.9981	0.9993	0.9995	0.9987
174	0.9987	0.9991	0.9989	0.9993
186	0.9991	0.9996	0.9988	0.9991
198	0.9990	0.9994	0.9991	0.9993
210	0.9991	0.9996	0.9992	0.9993
222	0.9990	0.9993	0.9994	0.9990
234	0.9998	0.9994	0.9995	0.9995
240	0.9992	0.9995	0.9995	0.9997

Table 6 depicts the performance of the three proposed hybrid models in terms of GMR. The solution of the three proposed hybrids contrasted with that of the GMR standalone HNN2SAT model. The three hybrid models demonstrate robust performance when compared with the standalone HNN2SAT model by returning better GMR values. This can be attributed to the fact that all the hybrids are enhanced with at least one form of enhancement that could improve the global minimum ratio. HNN2SATFuzzySA exhibits the lowest GMR across most trials when compared with other proposed models. The HNN2SATFuzzy model shows moderate performance, surpassing HNN2SATFuzzySA but falling short of the highest-performing hybrid.

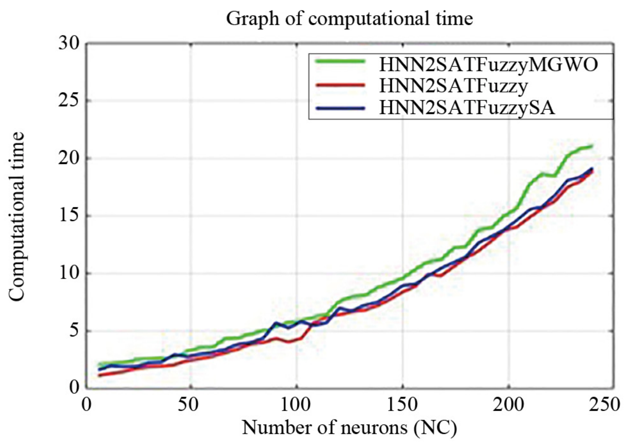


Figure 9. Depicts the comparative graph of computational time for the 3 hybrids

Table 7. Comparative analysis of computational time for the 3 hybrid models with a standalone HNN2SAT

NN	HNN2SAT	HNN2SATFuzzy	HNN2SATFuzzySA	HNN2SATFuzzyMGWO
	CPU Time	CPU Time	CPU Time	CPU Time
18	1.693048	1.515095	1.879299	1.797637
30	1.915635	1.755244	2.083074	2.338035
42	2.497386	2.028918	2.624331	2.542899
54	3.218197	2.550365	3.316325	2.855486
66	3.731941	3.056524	3.648125	3.438241
78	4.737024	3.328007	5.138205	4.138005
90	4.747937	4.407048	5.661468	4.991581
102	6.386023	5.365517	5.457552	5.480074
114	6.397776	5.417329	5.476026	5.819528
126	7.129434	5.972488	6.715594	6.574800
138	8.046515	6.961095	8.620929	8.489125
150	9.205783	7.924853	8.445312	8.522213
162	10.361246	10.500440	9.266918	9.479303
174	11.748086	10.545680	10.622722	10.840154
186	13.114224	11.841562	12.059469	11.953299
198	14.546283	13.507764	13.688606	13.792706
210	16.145518	14.657674	14.899354	15.262799
222	17.961669	16.983891	16.748409	16.765810
234	20.085045	18.889050	18.490746	19.118224
240	20.323748	18.931016	20.789034	20.849524

Figure 9 shows the graph of the time taken by the three hybrid models, which encompasses both the training and testing phases of the network. Conceptually, a consistent increase in the graph across three models underscores the direct relationship between network size and the time required for convergence. HNN2SATFuzzy achieves the shortest computational time in most instances during the simulation. This efficiency can be attributed to the defuzzification process, which classifies the network states into two sections based on the alpha condition as the only refinement, which classifies the network state based on a predefined alpha value. Once the alpha-cut condition is satisfied, the network promptly converges to a state that meets the alpha criteria, effectively halting the simulation process early. The computational time of the hybrid HNN2SATFuzzySA falls between the other two hybrids. Its moderate CPU time can be attributed to the large number of iterations required, similar to MGWO, but with fewer parameters and computational burdens. Unlike MGWO, HNN2SATFuzzySA's convergence is on a predefined few parameters. HNN2SATFuzzyMGWO has the highest computational time among the three hybrid models.

Table 7 shows the difference in CPU time across all models is less than two seconds, indicating that all three hybrid models demonstrate efficient computational performance.

12. Conclusion

In this research, we developed three hybrid models combining HNN with fuzzy logic and metaheuristic algorithms to effectively solve the 2SAT problem. We enhanced the fuzzy hybrid model by integrating both single-solution-based and population-based metaheuristic algorithms to evade local minima in the HNN's energy contour and drove the cost function towards zero during HNN training as defined in equation (12); this corresponds to minimal or near-minimal energy. Among the proposed models, HNN2SATFuzzyMGWO emerged as the most effective hybrid, demonstrating superior capability in navigating the rugged energy landscape and achieving global minima. The integration of fuzzy logic provided more flexibility in the optimisation process by generating diverse network states based on membership values. Lower membership values, which indicate local minima with fewer satisfied clauses, were further refined using optimisation algorithms such as Simulated Annealing (SA) and the Modified Grey Wolf Optimisation (MGWO). HNN2SATFuzzyMGWO consistently returned the lowest values for performance metrics such as RMSE, MAE, SSE, and SMAPE, reflecting minimal discrepancies between the baseline network state and the model's output. It also achieved a higher global minimum ratio across most runs, showcasing its ability to balance optimisation accuracy and energy efficiency. When compared to existing models like HNN-SAT and HNN-SATACO (a model enhanced with the Ant Colony Optimisation (ACO) algorithm as proposed in [60]), HNN2SATFuzzyMGWO demonstrated clear advantages. It produced lower RMSE, MAE, and computational time, which are the common performance metrics shared with that study. While both approaches achieved similar global minimum ratios as network complexity increased, our proposed model HNN2SATFuzzyMGWO maintained moderate computational times. The differences in computational time between the three proposed models were negligible, and it is within acceptable limits, as the variations are less than one second, attributed to additional time needed to carry out the metaheuristic's algorithms. This study highlights the potential of combining neuro-fuzzy models in solving satisfiability problems that erupt in real life, since most real-world problems can be modelled as SAT problems; solving them will benefit massively from this study. Fuzzy integration in the mode allows partial satisfiability, such as "mostly true," instead of strict true/false logic, thus facilitating logical consistency in decision-making processes that span every aspect of life. Future research could explore the integration of fuzzy rules that evolve through reinforcement learning to improve solving SAT problems using HNN and metaheuristic algorithms while examining their generalisability to Constraint Satisfaction Problems (CSPs) and testing their effectiveness in solving real-world optimisation problems.

Acknowledgement

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

Conflict of interest

The authors declare that they have no conflicts of interest.

References

[1] Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*. 1982; 79(8): 2554-2558. Available from: https://doi.org/10.1073/pnas.79.8.2554.

[2] Ramsauer H, Schäfl B, Lehner J, Seidl P, Widrich M, Adler T, et al. Hopfield networks is all you need. *arXiv:2008.02217*. 2020. Available from: https://doi.org/10.48550/arXiv.2008.02217.

[3] Daniele A, Serafini L. Knowledge enhanced neural networks for relational domains. 2022.*arXiv:2205.15762*. Available from: https://doi.org/10.48550/arXiv.2205.15762.

[4] Cunnington D, Law M, Lobo J, Russo A. FFNSL: Feed-forward neural-symbolic learner. *arXiv:2106.13103*. 2023. Available from: https://doi.org/10.48550/arXiv.2106.13103.

[5] Sathasivam S. Learning rules comparison in neuro-symbolic integration. *International Journal of Applied Physics and Mathematics*. 2011; 1(2): 129-132. Available from: https://doi.org/10.7763/IJAPM.2011.V1.25.

[6] Zhu L, Chen Y, Wang Y, Wei Y, Zheng H, Zhang Y. A comprehensive analysis of impacts of socio-economic development and land use on river water quality in a megacity-region: A case study. *Environmental Research Communications*. 2023; 5(2): 025006. Available from: https://doi.org/10.1088/2515-7620/acbbbd.

[7] Abdullah WATW. Logic programming on a neural network. *International Journal of Intelligent Systems*. 1992; 7(6): 513-519. Available from: https://doi.org/10.1002/int.4550070604.

[8] Alzaeemi SA, Salaudeen AA, Mohd Kasihmuddin MK, Mohd Mansor A, Sathasivam S. Use of genetic algorithm for hopfield neural network to do logic programming. *Bio-Genetics Journal*. 2017; 5(1): 101113.

[9] Kasihmuddin MS, Mansor MA, Sathasivam S. Discrete hopfield neural network in restricted maximum *k*-satisfiability logic programming. *Sains Malaysiana*. 2018; 47(6): 1327-1335. Available from: https://doi.org/10.17576/jsm-2018-4706-30.

[10] Mansor M, Kasihmuddin MSM, Saratha S. Artificial immune system paradigm in the hopfield network for 3-satisfiability problem. *Pertanika Journal of Science Technology*. 2017; 25(4): 1173-1188.

[11] Blocho M. Heuristics, metaheuristics, and hyperheuristics for rich vehicle routing problems smart delivery systems. *Smart Delivery Systems*. 2020; 101-156. Available from: https://doi.org/10.1016/B978-0-12-815715-2.00009-9.

[12] Gao J, Lv Y, Liu M, Cai S, Ma F. Improving simulated annealing for clique partitioning problems. *Journal of Artificial Intelligence Research*. 2022; 74: 1485-1513. Available from: https://doi.org/10.1613/jair.1.13382.

[13] Spears WM. Simulated annealing for hard satisfiability problems. *Cliques, Coloring, and Satisfiability*. 1993; 26: 533-558.

[14] Tambouratzis T. *Simulated Annealing Artificial Neural Networks for the Satisfiability (SAT) Problem Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer Vienna; 1995. p.340-343. Available from: https://doi.org/10.1007/978-3-7091-7535-4_89.

[15] Selman B, Kautz H. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In: *IJCAI'93: Proceedings of the 13th International Joint Conference on Artificial Intelligence, Vol 1*. France: International Joint Conferences on Artificial Intelligence; 1993. p.290-295.

[16] Kimura Y, Nishimori H. Convergence condition of simulated quantum annealing for closed and open systems. *Physical Review A*. 2022; 106(6): 062614. Available from: https://doi.org/10.1103/PhysRevA.106.062614.

[17] Malczewski J. Multicriteria analysis comprehensive geographic information systems. In: *Comprehensive Geographic Information Systems*. Amsterdam, Netherlands: Elsevier; 2018. p.197-217. Available from: https://doi.org/10.1016/B978-0-12-409548-9.09698-6.

[18] Kruger J. Simulated annealing: A tool for data assimilation into an almost steady model state. *Journal of Physical Oceanography*. 1993; 23(4): 679-688. Available from: https://doi.org/10.1175/1520-0485(1993)023<0679:SAATFD>2.0.CO;2.

[19] Sathasivam S, Wan Abdullah WAT. Logic mining in neural network: Reverse analysis method. *Computing*. 2011; 91(2): 119-133. Available from: https://doi.org/10.1007/s00607-010-0117-9.

[20] Azizan FL, Sathasivam S, Ali MKM, Roslan N, Feng C. Hybridised network of fuzzy logic and a genetic algorithm in solving 3-satisfiability hopfield neural networks. *Axioms*. 2023; 12(3): 250. Available from: https://doi.org/10.3390/axioms12030250.

[21] Stützle T, Hoos H, Roli A. *A review of the literature on local search algorithms for MAX-SAT*. Intellectics Group, Darmstadt University of Technology, Germany. Rapport technique: AIDA-01-02, 2001.

[22] Drechsler R, Junttila T, Niemelä I. Non-clausal SAT and ATPG. In: *Handbook of Satisfiability*. Netherlands: IOS Press Ebooks; 2022. p.655-693. Available from: https://doi.org/10.3233/978-1-58603-929-5-655.

[23] Ruiz JMD. SAT in P does not imply Chaos in the security system. *Research Square*. 2021. Available from: https: //doi.org/10.21203/rs.3.rs-678279/v1.

[24] Liang C, Wang X, Lu L, Niu P. A new method for 3-satisfiability problem solving space structure on structural entropy. *Symmetry*. 2021; 13(11): 2005. Available from: https://doi.org/10.3390/sym13112005.

[25] Bergel A. *The Traveling Salesman Problem Agile Artificial Intelligence in Pharo*. Berkeley, CA: Apress; 2020. p.209-224. Available from: https://doi.org/10.1007/978-1-4842-5384-7_10.

[26] Peng J, Xiao M. Further improvements for SAT in terms of formula length. *Information and Computation*. 2023; 294: 105085. Available from: https://doi.org/10.1016/j.ic.2023.105085.

[27] San Segundo P, Furini F, León R. A new branch-and-filter exact algorithm for binary constraint satisfaction problems. *European Journal of Operational Research*. 2022; 299(2): 448-467. Available from: https://doi.org/10.1016/j.ejor.2021.09.014.

[28] Kilani Y, Bsoul M, Alsarhan A, Khasawneh AA. A survey of the satisfiability-problems solving algorithms. *International Journal of Artificial Intelligence and Pattern Recognition*. 2013; 5(3): 233. Available from: https: //doi.org/10.1504/IJAIP.2013.056447.

[29] Dantsin E, Hirsch E. Worst-case upper bounds. In: *Handbook of Satisfiability*. Netherlands: IOS Press Ebooks; 2009. Available from: https://doi.org/10.3233/978-1-58603-929-5-403.

[30] Joya G, Atencia MA, Sandoval F. Hopfield neural networks for optimization: Study of the different dynamics. *Neurocomputing*. 2002; 43(1-4): 219-237. Available from: https://doi.org/10.1016/S0925-2312(01)00337-X.

[31] Kasihmuddin MSM, Mohd M, Saratha S. Robust artificial bee colony in the hopfield network for 2-satisfiability problem. *Pertanika Journal of Science Technology*. 2017; 25(2): 453-468.

[32] Mohamad S, Gopalsamy K. Dynamics of a class of discrete-time neural networks and their continuous-time counterparts. *Mathematics and Computers in Simulation*. 2000; 53(1-2): 1-39. Available from: https://doi.org/10.1016/S0378-4754(00)00168-3.

[33] Sampath S, Srivastava V. On stability and associative recall of memories in attractor neural networks. *PLoS ONE*. 2020; 15(9): e0238054. Available from: https://doi.org/10.1371/journal.pone.0238054.

[34] Zhou X, Huang J, Lu F, Liu J, Wang C. HNN-based generalized predictive control for turbofan engine direct performance optimization. *Aerospace Science and Technology*. 2021; 112: 106602. Available from: https://doi.org/10.1016/j.ast.2021.106602.

[35] Mansor MA, Kasihmuddin MSM, Sathasivam S. Enhanced hopfield network for pattern satisfiability optimization. *International Journal of Intelligent Systems and Applications*. 2016; 8(11): 27-33. Available from: https://doi.org/10.5815/ijisa.2016.11.04.

[36] Yadav JKPS, Singh L, Jaffery ZA. Optimization of hopfield neural network (HNN) using multiconnection and lyapunov energy function (LEF) for storage and recall of handwritten images. *Sādhanā-Academy Proceedings in Engineering Sciences*. 2023; 48(1): 26. Available from: https://doi.org/10.1007/s12046-023-02083-6.

[37] Charalambidis A, Handjopoulos K, Rondogiannis P, Wadge WW. Extensional higher-order logic programming. *ACM Trans Comput Logic*. 2013; 14(3): 1-40. Available from: https://doi.org/10.1145/2499937.2499942.

[38] Velavan M, Bin Yahya ZR, Bin Abdul Halif MN, Sathasivam S. Mean field theory in doing logic programming using hopfield network. *Modern Applied Science*. 2015; 10(1): 154. Available from: https://doi.org/10.5539/mas.v10n1p154.

[39] Zadeh LA. Fuzzy sets. *Information and Control*. 1965; 8(3): 338-353. Available from: https://doi.org/10.1016/S0019-9958(65)90241-X.

[40] Zadeh LA. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*. 1973; SMC-3(1): 28-44. Available from: https://doi.org/10.1109/TSMC.1973.5408575.

[41] Lehal MS. Fuzzy in the real world. *International Journal of Computers and Technology*. 2013; 7(1): 473-477. Available from: https://doi.org/10.24297/ijct.v7i1.3476.

[42] Mizumoto M, Tanaka K. Fuzzy sets and type 2 under algebraic product and algebraic sum. *Fuzzy Sets and Systems*. 1981; 5(3): 277-290. Available from: https://doi.org/10.1016/0165-0114(81)90056-7.

[43] De Campos Souza PV. Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature. *Applied Soft Computing*. 2020; 92: 106275. Available from: https://doi.org/10.1016/j.asoc.2020.106275.

[44] Jozi A, Pinto T, Praca I, Silva F, Teixeira B, Vale Z. Wang and Mendel's fuzzy rule learning method for energy consumption forecasting considering the influence of environmental temperature. In: *2016 Global Information Infrastructure and Networking Symposium*. Porto, Portugal: IEEE; 2016. Available from: https://doi.org/10.1109/GIIS.2016.7814944.

[45] Gou J, Hou F, Chen W, Wang C, Luo W. Improving Wang-Mendel method performance in fuzzy rules generation using the fuzzy C-means clustering algorithm. *Neurocomputing*. 2015; 151: 1293-1304. Available from: https://doi.org/10.1016/j.neucom.2014.10.077.

[46] Pourabdollah A, Mendel JM, John RI. Alpha-cut representation used for defuzzification in rule-based systems. *Fuzzy Sets and Systems*. 2020; 399: 110-132. Available from: https://doi.org/10.1016/j.fss.2020.05.008.

[47] Czogała E, Łęski J. *Fuzzy and Neuro-Fuzzy Intelligent Systems, Vol. 47*. Heidelberg: Physica-Verlag HD; 2000. Available from: https://doi.org/10.1007/978-3-7908-1853-6.

[48] Kirkpatrick SC, Daniel GJ, Mario PV. Optimization by simulated annealing. *Science*. 1983; 220(4598): 671-680.

[49] Lalaoui M, El Afia A, Chiheb R. Hidden Markov Model for a self-learning of Simulated Annealing cooling law. In: *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*. Marrakech, Morocco: IEEE; 2016. p.558-563. Available from: https://doi.org/10.1109/ICMCS.2016.7905557.

[50] Mateev V, Marinova I. Optimization of heat sink design by simulated annealing method. In: *2023 XXXII International Scientific Conference Electronics*. Sozopol, Bulgaria: IEEE; 2023. Available from: https://doi.org/10.1109/ET59121.2023.10279429.

[51] Park MW, Kim YD. A systematic procedure for setting parameters in simulated annealing algorithms. *Computers Operations Research*. 1998; 25(3): 207-217. Available from: https://doi.org/10.1016/S0305-0548(97)00054-3.

[52] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. *Advances in Engineering Software*. 2014; 69: 46-61. Available from: https://doi.org/10.1016/j.advengsoft.2013.12.007.

[53] Emary E, Zawbaa HM, Hassanien AE. Binary grey wolf optimization approaches for feature selection. *Neurocomputing*. 2016; 172: 371-381. Available from: https://doi.org/10.1016/j.neucom.2015.06.083.

[54] Banaie-Dezfouli M, Nadimi-Shahraki MH, Beheshti Z. R-GWO: Representative-based grey wolf optimizer for solving engineering problems. *Applied Soft Computing*. 2021; 106: 107328. Available from: https://doi.org/10.1016/j.asoc.2021.107328.

[55] Barbour B, Brunel N, Hakim V, Nadal JP. What can we learn from synaptic weight distributions. *Trends in Neurosciences*. 2007; 30(12): 622-629. Available from: https://doi.org/10.1016/j.tins.2007.09.005.

[56] Wang W, Lu Y. Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model. In: *IOP Conference Series: Materials Science and Engineering*. UK: Institute of Physics Publishing Limited; 2018. Available from: https://doi.org/10.1088/1757-899X/324/1/012049.

[57] Chicco D, Warrens MJ, Jurman G. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*. 2021; 7: e623. Available from: https://doi.org/10.7717/peerj-cs.623.

[58] Sathasivam S, Mustafa M, Mohd Shareduwan MK, Mohd AM. Metaheuristics approach for maximum *k*-satisfiability in restricted neural symbolic integration. *Pertanika Journal of Science & Technology*. 2020; 28(2): 545-564.

[59] Chai T, Draxler RR. Root mean square error (RMSE) or mean absolute error (MAE)?-Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*. 2014; 7(3): 1247-1250. Available from: https://doi.org/10.5194/gmd-7-1247-2014.

[60] Kho LC, Kasihmuddin MSM, Mansor MA, Sathasivam S. 2 satisfiability logical rule by using ant colony optimization in Hopfield Neural Network. In: *AIP Conference Proceedings*. USA: American Institute of Physics; 2019. Available from: https://doi.org/10.1063/1.5136441.