

HyperSAT: Unsupervised Hypergraph Neural Networks for Weighted MaxSAT Problems

Qiyue Chen^{1,2} Shaolin Tan² Suixiang Gao^{1,2} Jinhu Lü^{3,2}

Abstract

Graph neural networks (GNNs) have shown promising performance in solving both Boolean satisfiability (SAT) and Maximum Satisfiability (MaxSAT) problems due to their ability to efficiently model and capture the structural dependencies between literals and clauses. However, GNN methods for solving Weighted MaxSAT problems remain underdeveloped. The challenges arise from the non-linear dependency and sensitive objective function, which are caused by the non-uniform distribution of weights across clauses. In this paper, we present HyperSAT, a novel neural approach that employs an unsupervised hypergraph neural network model to solve Weighted MaxSAT problems. We propose a hypergraph representation for Weighted MaxSAT instances and design a cross-attention mechanism along with a shared representation constraint loss function to capture the logical interactions between positive and negative literal nodes in the hypergraph. Extensive experiments on various Weighted MaxSAT datasets demonstrate that HyperSAT achieves better performance than state-of-the-art competitors.

1. Introduction

The Boolean satisfiability (SAT) problem is a fundamental computational problem that asks whether there exists an assignment of true or false values to a set of variables such that a given propositional Boolean formula evaluates to true. It is the first problem proven to be NP-complete (Cook, 1971) and has important applications in various fields, including planning (Kautz et al., 1992), hardware and software verification (Clarke et al., 2001), and cryptanalysis (Sun et al.,

2020). The Maximum Satisfiability (MaxSAT) problem and its generalization, the Weighted MaxSAT problem, are the variants of SAT with a goal of maximizing the number of satisfied clauses or their weighted sum. By incorporating the notion of prioritizing or weighing different clauses, the Weighted MaxSAT problem provides a more general and flexible version of SAT and MaxSAT to tackle a wide range of real-world problems, such as scheduling (Thornton & Sattar, 1998), action model learning (Yang et al., 2007), computational protein design (Allouche et al., 2012) and resource allocation (Timm & Botha, 2022).

Over the past several decades, SAT problems and their extensions have become central topics in computational logic and theoretical computer science. A wide array of solvers has been developed, with most of them relying on meticulously designed search algorithms (Audemard & Simon, 2018; Cai & Zhang, 2021; Biere & Fleury, 2022). With the rapid progress of deep learning, there has been a growing shift towards developing learning-based solvers. These learning-based methods aim to effectively extract hidden structures within problem instances and uncover underlying patterns from the data, thus reducing reliance on manually crafted and domain-specific strategies, and providing more adaptable and scalable solvers to tackle large and complex problem instances.

Specifically, the applications of deep learning techniques to SAT problems have led to two main classes of approaches: end-to-end neural solvers and neural-guided solvers. The end-to-end approaches are typically designed to predict satisfiability or optimal assignments directly from the raw problem input. For example, the groundbreaking work NeuroSAT (Selsam et al., 2019) introduced an end-to-end framework that predicts satisfiability on random instances using a message passing neural network (MPNN). The study by (Cameron et al., 2020) represented SAT instances as permutation-invariant sparse matrices and compared the performances of the exchangeable architecture and MPNN. For the first time, it was demonstrated that end-to-end learning methods could achieve competitive performance in terms of satisfiability prediction. QuerySAT (Ozolins et al., 2022) proposed a neural SAT solver with a query mechanism that allows the network to make multiple solution trials and get feedback on the loss value towards better solutions.

¹School of Mathematical Sciences, University of Chinese Academy of Science, Beijing, China ²Zhongguancun Laboratory, Beijing, China ³School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. Correspondence to: Shaolin Tan <shaolintan@hnu.edu.cn>.

The neural-guided solvers, on the other hand, integrate neural networks with traditional search frameworks to improve both the efficiency and solution quality of SAT solvers. Along this line, NeuroCore (Selsam & Bjørner, 2019) leveraged Graph Neural Networks (GNNs) to predict unsatisfiable cores in SAT instances and further applied this prediction to periodically update the activity score of each variable in the Conflict-Driven Clause Learning (CDCL) solver. NLocalSAT (Zhang et al., 2021) employed a gated graph convolutional network (GGCN) to guide the initialization of assignments in the Stochastic Local Search (SLS) solver. Recently, NeuroBack (Wang et al., 2024) proposed a new approach by utilizing a graph transformer architecture to make offline neural predictions on backbone variable phases for refining the phase selection heuristic in CDCL solvers.

For the MaxSAT problem, (Liu et al., 2023) was the pioneering work in exploring the use of GNNs for solving the MaxSAT problem. This work represented the MaxSAT problem as two kinds of factor graphs and studied the capability of typical GNN models in solving the MaxSAT problem from a theoretical perspective. Overall, it can be observed that GNNs have shown promising performance in solving both SAT and MaxSAT problems, however, the extension of these methods to the Weighted MaxSAT problem remains underdeveloped to our best knowledge. The main challenge comes from the uneven weight distribution in the Weighted MaxSAT problem. The addition of weights increases the complexity of the search space and also makes the design of effective heuristics more difficult. The neural model needs to learn how to prioritize high-weight clauses over others. Moreover, since a small change in the assignment could result in a significant change in the weighted sum, the hidden structural patterns within Weighted MaxSAT instances are much harder to learn.

In this work, we consider the problem of designing a neural network solver for Weighted MaxSAT problems. Considering the intrinsic difficulties in learning the non-linear dependency and sensitive objective function in the Weighted MaxSAT problem, we formulate our learning-based framework HyperSAT by integrating hypergraph representation, cross-attention mechanism, and multi-objective loss design. We model the Weighted MaxSAT instance as a hypergraph. The positive and negative literals of a variable are represented as separate nodes, and each clause is represented as a hyperedge, with an associated weight equal to the weight of the clause. The hypergraph convolutional network (HyperGCN) is utilized as the learning model, and a specific cross-attention mechanism is designed to capture the logical interplay between the positive and negative literal node features of the same variable. In addition, we design an unsupervised multi-objective loss function to optimize the learning model. Besides the intrinsic optimization objective of the Weighted MaxSAT problem, a shared representation constraint ob-

jective is introduced to ensure that the representations of positive and negative literals of a variable are as distinct as possible in the feature space. We have tested our model on several random Weighted MaxSAT datasets in different settings. The experimental results demonstrate that our model outperforms baseline methods across various datasets, achieving significantly better performance. This work provides a new perspective on solving the Weighted MaxSAT problem using learning-based methods, with the hope that the results can offer preliminary knowledge into the capability of neural networks for solving Weighted MaxSAT problems in the future.

In summary, we make the following contributions:

- We propose HyperSAT, an innovative neural approach that uses an unsupervised hypergraph neural network model to solve Weighted MaxSAT problems. This is the first work to predict the solution of the Weighted MaxSAT problem with HyperGCN in an end-to-end fashion.
- We propose a hypergraph representation for Weighted MaxSAT instances and design a cross-attention mechanism along with a shared representation constraint loss function to capture the logical relationships between positive and negative literal nodes within the hypergraph.
- We conduct an extensive evaluation of the proposed HyperSAT on multiple Weighted MaxSAT datasets with different distributions. The experimental results demonstrate the superior performance of HyperSAT.

2. Preliminaries

We now briefly introduce some relevant preliminaries on Weighted MaxSAT and hypergraph neural networks that will be leveraged in later sections.

2.1. Weighted MaxSAT

A Weighted MaxSAT instance is represented by a triple $\varphi = (\mathcal{X}, \mathcal{C}, \mathbf{w})$. Here, the set of variables is denoted by $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$, where each $x_i \in \{0, 1\}$ is a Boolean variable and n is the total number of variables involved in the instance. The set of clauses is given by $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$, where each $C_j \in \mathcal{C}$ is a disjunction of literals and m is the number of clauses. The weight vector is given by $\mathbf{w} = (w_1, w_2, \dots, w_m)$ where w_j represents the weight of C_j . Take the clause $C_j \in \mathcal{C}$ for example. Suppose it contains k_j literals. Then, it can be represented by $C_j = l_{j1} \vee l_{j2} \vee \dots \vee l_{jk_j}$, where each literal l_{ji} is either a variable or the negation of a variable in \mathcal{X} . The clause C_j is satisfied if at least one of its literals evaluates to true. The clause C_j is unsatisfied if all its literals evaluate to false.

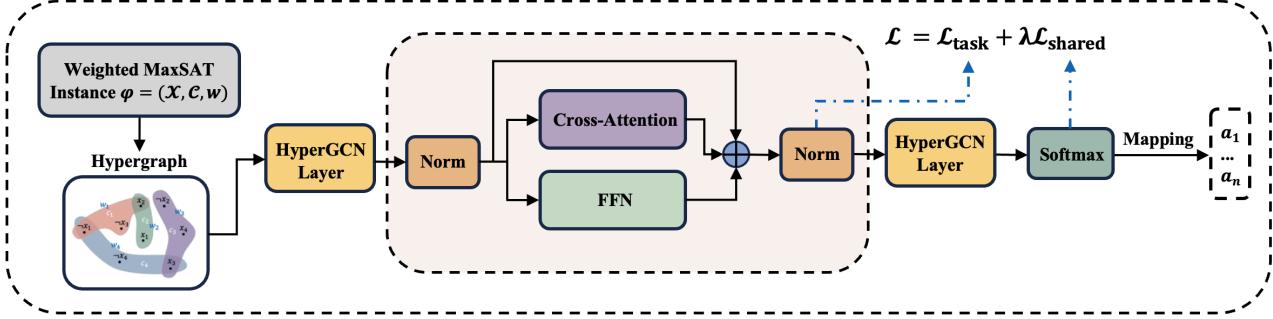


Figure 1. An overview of HyperSAT framework.

An assignment $\mathbf{A} = \{a_1, a_2, \dots, a_n\}$ is to assign each variable $x_i \in \mathcal{X}$ with a value $a_i \in \{0, 1\}$. Given a Weighted MaxSAT instance $\varphi = (\mathcal{X}, \mathcal{C}, \mathbf{w})$, the objective is to find an assignment \mathbf{A} for the Boolean variables \mathcal{X} such that the total weight of satisfied clauses is maximized. The optimization problem is given by

$$\begin{aligned} \max_{\mathbf{A}} \quad & \sum_{j=1}^m w_j \cdot \mathbf{1}(C_j(\mathbf{A})) \\ \text{s.t.} \quad & \mathbf{A} \in \{0, 1\}^n. \end{aligned}$$

Here,

$$\mathbf{1}(C_j(\mathbf{A})) = \begin{cases} 1, & \text{if } C_j \text{ is satisfied by } \mathbf{A}, \\ 0, & \text{otherwise.} \end{cases}$$

2.2. Hypergraph Neural Networks

A hypergraph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ which includes a set of nodes $\mathcal{V} = \{v_1, v_2, \dots, v_{|\mathcal{V}|}\}$ and a set of hyperedges $\mathcal{E} = \{e_1, e_2, \dots, e_{|\mathcal{E}|}\}$. \mathbf{W} is a diagonal matrix of edge weights which assigns a weight to each hyperedge. Each hyperedge e_j is a non-empty subset of nodes (i.e., $\emptyset \neq e_j \subseteq \mathcal{V}$). The hypergraph \mathcal{G} can be represented as a $|\mathcal{V}| \times |\mathcal{E}|$ incidence matrix \mathbf{H} , where $\mathbf{H}_{i,j} = 1$ if $v_i \in e_j$ and 0 otherwise. For a vertex $v_i \in \mathcal{V}$, its degree is defined as $d(v_i) = \sum_{j=1}^{|\mathcal{E}|} \mathbf{H}_{i,j} \mathbf{W}_{j,j}$. For an edge $e_j \in \mathcal{E}$, its degree is defined as $\delta(e_j) = \sum_{i=1}^{|\mathcal{V}|} \mathbf{H}_{i,j}$. Additionally, \mathbf{D}_v and \mathbf{D}_e denote the diagonal matrices of the vertex degrees and the edge degrees, respectively.

Hypergraph Neural Networks (HGNN) (Feng et al., 2019) perform data representation learning by utilizing a hypergraph structure to capture higher-order dependencies between nodes. In HGNN, hyperedge convolution operations are used to extract features by leveraging the hypergraph Laplacian for spectral convolution. To reduce computational complexity, Chebyshev polynomials are applied to approximate the spectral convolution and avoid the need to compute high-order eigenvectors explicitly.

Through the hyperedge convolution operation, the l -th layer of HGNN can be formulated by

$$\mathbf{X}^{(l+1)} = \sigma \left(\mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \mathbf{X}^{(l)} \Theta^{(l)} \right), \quad (1)$$

where $\mathbf{X}^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times d_l}$ is the signal of the hypergraph at l layer with $|\mathcal{V}|$ nodes and d_l dimensional features, $\mathbf{X}^{(0)}$ is the original signal of the hypergraph. $\Theta^{(l)}$ is the learnable filter parameter and σ denotes the nonlinear activation function.

3. HyperSAT

In this section, we propose HyperSAT, a neural approach for Weighted MaxSAT problems. Figure 1 illustrates the workflow of HyperSAT. It mainly consists of three modules: the hypergraph modeling module, the neural network module, and the probabilistic mapping module.

Given a Weighted MaxSAT instance, HyperSAT first applies its hypergraph modeling component to represent the instance as a hypergraph. Then, the hypergraph is solved by the neural network module, which is responsible for processing and optimizing the hypergraph. Finally, through a mapping operation, the probabilistic output of the neural network module is mapped into Boolean values, which serves as a solution to the Weighted MaxSAT instance.

Note that the uneven weight of clauses increases the non-linear dependency and sensitivity among variables, the neural network is required to learn how to prioritize the contributions of different clauses. To this end, we propose a specific cross-attention mechanism and introduce an unsupervised multi-objective loss function to capture the logical interplay between the positive and negative literal node representations. These two mechanisms are integrated with the hypergraph convolutional network to form the neural network module of our HyperSAT.

3.1. Hypergraph Modeling

Given a Weighted MaxSAT instance $\varphi = (\mathcal{X}, \mathcal{C}, \mathbf{w})$, we construct the hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ as follows:

- For the construction of the vertex set \mathcal{V} , each variable $x_i \in \mathcal{X}$ is represented by two nodes v_i and v_{n+i} , which correspond to the positive and negative literals of the variable x_i (i.e., x_i and $\neg x_i$), respectively.
- For the construction of the hyperedge set \mathcal{E} , each clause $C_j \in \mathcal{C}$ is represented by a hyperedge $e_j \in \mathcal{E}$, which connects the nodes corresponding to the literals in C_j . Specifically, if clause C_j consists of the literals $l_{j1}, l_{j2}, \dots, l_{jk_j}$, then the corresponding hyperedge e_j connects the nodes corresponding to these literals.
- For the construction of the weight matrix \mathbf{W} , the weight of each hyperedge is equal to the weight of the corresponding clause, i.e., $\mathbf{W}_{j,j} = w_j$.

Through the above modeling process, the Weighted MaxSAT instance can be uniquely represented by a hypergraph, where each literal is a node, and each clause is a hyperedge connecting the literals involved. The weight of each hyperedge is the weight of the corresponding clause in the instance. Figure 2 gives an illustration of the hypergraph modeling process.

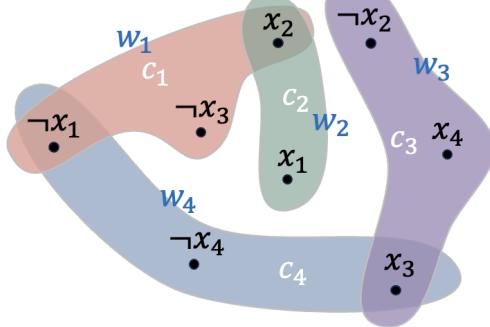


Figure 2. Hypergraph Modeling of a Weighted MaxSAT instance $(\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4)$ with 8 literals, 4 clauses and weights $\{w_1, w_2, w_3, w_4\}$.

In contrast to existing methods that represent Conjunctive Normal Form (CNF) formulas as factor graphs (Guo et al., 2023), which limit their ability to model relationships beyond pairwise connections, we construct Weighted MaxSAT instances as hypergraphs. These hypergraphs can encode higher-order variable dependencies through their degree-free hyperedges, and thus offer more powerful representational capacity and enable more efficient handling

of higher-order relationships in combinatorial optimization problems. In particular, we treat literals rather than variables as nodes, which addresses a major issue present in HypOp (Heydaribeni et al., 2024). Since the logical relationships between positive and negative literals are central to the Weighted MaxSAT problem, it is essential to treat them as distinct nodes, rather than merging them into a single node. In addition, we encode the weights of the clauses in the Weighted MaxSAT problem as the weights of the corresponding hyperedges in the hypergraph.

3.2. Neural Network Architecture

The neural network architecture of our proposed HyperSAT comprises the HyperGCN, a transformer module with the cross-attention mechanism, and a softmax layer.

3.2.1. HYPERGRAPH CONVOLUTIONAL NETWORKS

In HyperSAT, we introduce a T -layer HyperGCN for message passing among different nodes. The operation at the l -th layer of the HyperGCN is formally expressed as follows:

$$\mathbf{L}^{(l+1)'} = \sigma \left(\mathbf{D}_v^{-\frac{1}{2}} \tilde{\mathbf{Q}} \mathbf{D}_v^{-\frac{1}{2}} \mathbf{L}^{(l)} \mathbf{R}^{(l)} \right). \quad (2)$$

In this formulation, the matrix $\mathbf{L}^{(l+1)'}$ represents the output of the l -th layer; \mathbf{D}_v is the diagonal matrix of the vertex degrees in the hypergraph; $\mathbf{L}^{(l)} \in \mathbb{R}^{2n \times d_l}$ is the matrix of node representations at the l -th layer, where d_l is the dimension of the l -th layer node representations; $\mathbf{R}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is the l -th layer learnable weight matrix; and $\tilde{\mathbf{Q}}$ is given by

$$\tilde{\mathbf{Q}} = \mathbf{H} \tilde{\mathbf{D}}_e^{-1} \mathbf{H}^\top - \text{diag}(\mathbf{H} \tilde{\mathbf{D}}_e^{-1} \mathbf{H}^\top), \quad (3)$$

where \mathbf{H} is the hypergraph incidence matrix, and $\tilde{\mathbf{D}}_e = \mathbf{D}_e - \mathbf{I}$. Specifically, σ denotes the nonlinear activation function and $\mathbf{L}^{(0)}$ is a learnable input embedding of HyperGCN.

Compared to the updating rule in Eq. (1) in traditional HGNN, our HyperGCN focuses the convolutional layer's computation more on the influence of adjacent nodes by removing the diagonal elements of $\tilde{\mathbf{Q}}$. This adjustment allows the representation updates of each node to better align with the higher-order relationships of the adjacency structure.

3.2.2. CROSS-ATTENTION MECHANISM

The core of the Weighted MaxSAT problem lies in the logical constraints among variables. Positive and negative literal nodes (e.g., x and $\neg x$) are logically mutually exclusive and strongly correlated, and their relationships directly reflect the underlying structural characteristics of the problem.

Considering this, we leverage a cross-attention mechanism to dynamically assign importance weights to the relationships between each pair of complementary literals. This mechanism enables the model to adaptively capture critical logical properties such as mutual exclusivity, dependency, and the process of information exchange between positive and negative literal nodes. Therefore, after updating each node representation in the hypergraph convolution layer, a cross-attention layer is introduced to enhance the representations of positive and negative literal nodes.

Specifically, given the l -th layer output of the Hyper-GCN $\mathbf{L}^{(l+1)'} \in \mathbb{R}^{n \times d_{l+1}}$, we divide it into two parts: $\mathbf{L}^{(l+1)'} = [\mathbf{L}_+^{(l+1)'}, \mathbf{L}_-^{(l+1)'}]$, where $\mathbf{L}_+^{(l+1)'} \in \mathbb{R}^{n \times d_{l+1}}$ represent the positive literal node representations and $\mathbf{L}_-^{(l+1)'} \in \mathbb{R}^{n \times d_{l+1}}$ represent the negative literal node representations. The cross-attention mechanism is mathematically represented as follows:

$$\mathbf{L}_+^{(l+1)} = \text{softmax} \left(\frac{\mathbf{Q}_+^{(l+1)} (\mathbf{K}_-^{(l+1)})^\top}{\sqrt{d_{l+1}}} \right) \mathbf{V}_-^{(l+1)}, \quad (4)$$

$$\mathbf{L}_-^{(l+1)} = \text{softmax} \left(\frac{\mathbf{Q}_-^{(l+1)} (\mathbf{K}_+^{(l+1)})^\top}{\sqrt{d_{l+1}}} \right) \mathbf{V}_+^{(l+1)}. \quad (5)$$

In this formulation,

$$\begin{aligned} \mathbf{Q}_+^{(l+1)} &= \mathbf{W}_Q^{(l+1)} \mathbf{L}_+^{(l+1)'}, \mathbf{Q}_-^{(l+1)} = \widetilde{\mathbf{W}}_Q^{(l+1)} \mathbf{L}_-^{(l+1)'}, \\ \mathbf{K}_+^{(l+1)} &= \mathbf{W}_K^{(l+1)} \mathbf{L}_+^{(l+1)'}, \mathbf{K}_-^{(l+1)} = \widetilde{\mathbf{W}}_K^{(l+1)} \mathbf{L}_-^{(l+1)'}, \\ \mathbf{V}_+^{(l+1)} &= \mathbf{W}_V^{(l+1)} \mathbf{L}_+^{(l+1)'}, \mathbf{V}_-^{(l+1)} = \widetilde{\mathbf{W}}_V^{(l+1)} \mathbf{L}_-^{(l+1)'}, \end{aligned}$$

where $\mathbf{W}_Q^{(l+1)}$, $\mathbf{W}_K^{(l+1)}$, $\mathbf{W}_V^{(l+1)}$ are the learnable query, key and value projection matrices for the positive literal nodes at the l -th layer, and $\widetilde{\mathbf{W}}_Q^{(l+1)}$, $\widetilde{\mathbf{W}}_K^{(l+1)}$, $\widetilde{\mathbf{W}}_V^{(l+1)}$ are the learnable query, key and value projection matrices for the negative literal nodes at the l -th layer.

The final node representation at the l -th layer is obtained as $\mathbf{L}^{(l+1)} = [\mathbf{L}_+^{(l+1)'}, \mathbf{L}_-^{(l+1)'}]$. The cross-attention mechanism explicitly constructs the interaction between positive and negative literal nodes by enabling each node to incorporate the features of its counterpart. This facilitates a more comprehensive representation of node features for Weighted MaxSAT problems.

Building upon this, the transformer module in HyperSAT incorporates the previously discussed cross-attention layer, along with other components to further enhance the modeling capability. Inspired by the recent advancements in Vision Transformer architecture (ViT-22B (Dehghani et al., 2023)), the transformer module in HyperSAT includes a LayerNorm layer (Ba, 2016), followed by a combination of a cross-attention layer and a Feed-Forward Network (FFN) layer, along with a residual connection and an additional

LayerNorm layer. The FFN and cross-attention layers operate in parallel to enhance the efficiency. The residual connection is introduced by adding the outputs from the cross-attention layer, the FFN layer, and the LayerNorm layer. The resulting sum is then passed through another LayerNorm layer to stabilize the representations before proceeding to the next layer.

The final layer of the network is a softmax layer. We reshape the iterated $\mathbf{L}^{(T)'} \in \mathbb{R}^{2n \times 1}$ into $\hat{\mathbf{L}}^{(T)'} \in \mathbb{R}^{n \times 2}$, which serves as the input to the softmax layer. After applying the softmax function, we obtain $\tilde{\mathbf{L}} = \text{softmax}(\hat{\mathbf{L}}^{(T)'})$, which provides the soft node assignments for each literal, interpreted as class probabilities. The probability of assigning a variable to a truth value is recorded by $\mathbf{Y} = \tilde{\mathbf{L}}_{\cdot 1}$, which is the first column of $\tilde{\mathbf{L}}$.

3.2.3. LOSS FUNCTION

Given a Weighted MaxSAT instance $\varphi = (\mathcal{X}, \mathcal{C}, \mathbf{w})$ and its hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, we relax the Boolean variables \mathcal{X} into continuous probability parameters $\mathbf{Y}(\gamma)$ where $\gamma = (\mathbf{R}, \mathbf{L}^{(0)}, \mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \widetilde{\mathbf{W}}_Q, \widetilde{\mathbf{W}}_K, \widetilde{\mathbf{W}}_V)$ represent the learnable parameters. The relaxation is defined as follows:

$$\mathcal{X} \in \{0, 1\}^n \longrightarrow \mathbf{Y}(\gamma) \in [0, 1]^n. \quad (6)$$

With this relaxation, we can design a differentiable loss function to optimize the learnable parameters γ , enabling gradient-based optimization.

In this paper, we propose an unsupervised multi-objective loss function that consists of two components: a primary task loss $\mathcal{L}_{\text{task}}$ and a shared representation constraint loss $\mathcal{L}_{\text{shared}}$. This unsupervised loss function obviates the necessity for large, labeled training datasets, which are conventionally indispensable in supervised learning paradigms. The specific form is as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{shared}}, \quad (7)$$

where $\lambda \geq 0$ is a balancing hyperparameter.

The primary task loss function is the relaxed optimization objective of the Weighted MaxSAT problem, as shown below:

$$\mathcal{L}_{\text{task}}(\mathbf{Y}) = \sum_{j=1}^m w_j V_j(\mathbf{Y}), \quad (8)$$

where

$$V_j(\mathbf{Y}) = 1 - \prod_{i \in C_j^+} (1 - y_i) \prod_{i \in C_j^-} y_i. \quad (9)$$

Here, C_j^+ and C_j^- are the index sets of variables appearing in the clause C_j in the positive and negative form, respectively. The term $V_j(\mathbf{Y})$ represents the satisfaction of clause C_j , where a value of 1 indicates that the clause is satisfied, and 0 indicates it is not. The weight w_j ensures that

more important clauses are prioritized during optimization. Minimizing the primary task loss function corresponds to maximizing the weighted sum of satisfied clauses in the Weighted MaxSAT problem.

The shared representation constraint loss function focuses on the representations of positive and negative literal nodes in the penultimate layer of the HyperSAT network. Its form is given by

$$\mathcal{L}_{\text{shared}} = \left\| \mathbf{L}_+^{(T-1)} + \mathbf{L}_-^{(T-1)} \right\|_F^2, \quad (10)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. By applying the shared representation constraint loss function, the network encourages the positive and negative literal nodes to develop distinct feature representations in the learning process. This constraint enhances the separation of the two types of nodes in the feature space.

It should be noted that most neural network-based SAT methods employ supervised learning. However, supervised learning approaches are not particularly well-suited for solving the Weighted MaxSAT problem. On the one hand, a Weighted MaxSAT instance may have multiple satisfying assignments. On the other hand, the non-uniform distribution of weight across the clauses makes it challenging for supervised learning models to detect the hidden structural patterns within the Weighted MaxSAT instance. From this perspective, our proposed unsupervised learning approach offers an effective alternative approach that enhances generalization capabilities while eliminating the need for labeled instances.

3.3. Probabilistic Mapping

The neural network module takes the hypergraph as input and produces a probability vector \mathbf{Y} as output, which indicates the probability of each variable taking the truth value. Accordingly, to convert the probability vector \mathbf{Y} into a Boolean assignment, we transform \mathbf{Y} into n Bernoulli distributions, where the i -th distribution $\mathcal{B}(y_i)$ is over the discrete values $\{0, 1\}$. Finally, the node assignments are generated by sampling with the corresponding probability distributions.

4. Experiment

4.1. Experimental Settings

We compare the performance of HyperSAT against GNN-based methods for solving Weighted MaxSAT problems.

Baseline Algorithms. We compare our proposed HyperSAT against GNN-based methods. The following algorithms are considered as baselines: (i) HypOp (Heydaribeni et al., 2024): an advanced unsupervised learning framework

that solves constrained combinatorial optimization problems using hypergraphs; (ii) (Liu et al., 2023): an innovative supervised GNN-based approach that predicts solutions in an end-to-end manner by transforming CNF formulas into factor graphs. We apply these baselines to solve Weighted MaxSAT problems.

Datasets. We evaluate the algorithms using random 3-SAT benchmarks from the SATLIB dataset (Hoos & Stützle, 2000). The SATLIB dataset is one of the standard datasets for evaluating SAT solvers. We utilize a range of datasets with varying distributions, from which SAT and UNSAT instances are generated for each distribution. The number of variables in the dataset ranges from 100 to 250, while the number of clauses varies between 430 and 1065. More details on the datasets are provided in Table 1. In particular, to generate the required Weighted MaxSAT instances, we assign weights to the clauses in each CNF file within the dataset. These weights are sampled from the integers in the range [1, 10] uniformly at random.

Table 1. The parameters of the datasets.

DATASET	INSTANCE	VARS	CLAUSES	TYPES
UF100-430	1000	100	430	SAT
UUF100-430	1000	100	430	UNSAT
UF200-860	100	200	860	SAT
UUF200-860	100	200	860	UNSAT
UF250-1065	100	250	1065	SAT
UUF250-1065	100	250	1065	UNSAT

Model Settings. HyperGCN is a two-layer network, with the input dimension set to the square root of the number of variables, and the hidden layer dimension set to half of the square root of the number of variables. The dimension of the cross-attention layer is the square root of the number of variables. The dropout used in the cross-attention layer is 0.1. The model is optimized using the Adam optimizer (Kingma, 2014) with a learning rate of 7×10^{-2} . The balancing hyperparameter of loss function λ is 2×10^{-3} . The FFN consists of two linear transformations and a ReLU activation function, with both the input and output dimensions set to the square root of the number of variables, and the hidden layer dimension set to half of the square root of the number of variables. An early stopping strategy is employed, with a tolerance of 10^{-4} and a patience of 50 iterations, terminating training if no improvement is observed over this period. Finally, in the random sampling stage, we perform 5 independent samplings and return the best solution obtained.

Evaluation Configuration. All experiments are conducted on an NVIDIA A100-SXM4-40GB GPU, using Python 3.9.30 and PyTorch 1.13.0.

4.2. Analytical Experiment

We first evaluate the performance of unsupervised methods, HyperSAT and HypOp, with a focus on their convergence. We evaluate the convergence using the primary task loss $\mathcal{L}_{\text{task}}$ from Eq. (8), which represents the sum of the weights of unsatisfied clauses. We illustrate the evolution curves of loss for HyperSAT and HypOp on the uuf250-1065 dataset in Figure 3 as an example. All models can converge within 300 epochs. Moreover, the loss of HyperSAT is around 52, while the loss of HypOp is around 139. We can observe that HyperSAT achieves better performance than HypOp. Specifically, HyperSAT decreases the loss more quickly and achieves a lower loss value. The experimental results demonstrate that HyperSAT can be used in learning to solve Weighted MaxSAT problems.

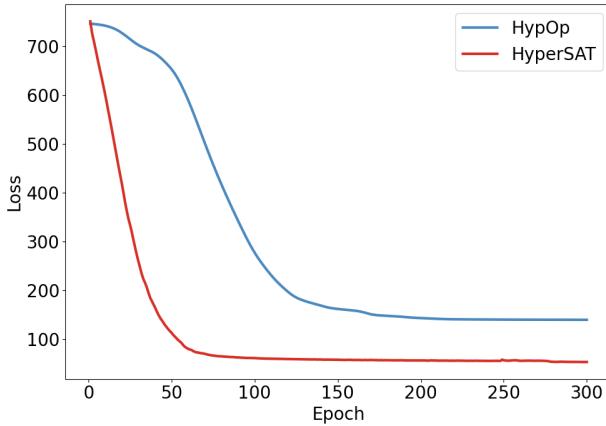


Figure 3. The evolution of loss for HyperSAT and HypOp during an inference process of 300 epochs on the uuf250-1065 dataset.

4.3. Result

We evaluate the performance of HyperSAT against baseline algorithms HypOp and (Liu et al., 2023) on various datasets. The primary evaluation metric is the average weighted sum of unsatisfied clauses. The experiments are conducted on datasets with the number of variables ranging from 100 to 250, and the number of clauses varying between 430 and 1065. The results are shown in Table 2.

As presented in the table, HyperSAT consistently achieves lower values for the average weighted sum of unsatisfied clauses compared to the baseline algorithms (Liu et al., 2023) and HypOp across multiple datasets. For example, on the uf100-430 dataset, HyperSAT significantly outperforms both baselines with a result of 15.64. This result represents a substantial improvement over the results of (Liu et al., 2023) (32.48) and HypOp (99.15). This trend is observed across all datasets, where HyperSAT always exceeds the

Table 2. The average weighted sum of unsatisfied clauses of Weighted MaxSAT problems.

DATASET	LIU ET AL. (2023)	HYPOP	HYPERSAT
UF100-430	32.48	99.15	15.64
UUF100-430	41.65	102.44	20.46
UF200-860	67.38	158.46	28.98
UUF200-860	81.68	171.34	35.55
UF250-1065	79.06	170.60	33.24
UUF250-1065	100.04	182.39	41.64

performance of the baselines. The average weight of unsatisfied clauses is reduced by approximately 50% compared to (Liu et al., 2023) and over 80% compared to HypOp. On the uf200-860 and uuf200-860 datasets, HyperSAT achieves reductions of 56.99% and 56.48% compared to (Liu et al., 2023), respectively, and reductions of 81.71% and 79.25% compared to HypOp. Similarly, on the uf250-1065 and uuf250-1065 datasets, HyperSAT continues to outperform the baselines. The reductions compared to (Liu et al., 2023) are 57.96% and 58.38%, respectively, while the reductions compared to HypOp are 80.52% and 77.17%. Importantly, even with the larger datasets, HyperSAT maintains or even enhances its efficacy. This demonstrates that HyperSAT scales well and performs better. These results show that HyperSAT consistently provides substantial improvements over both baseline algorithms across a range of datasets, including those with larger problem sizes. As a result, it validates its robustness and effectiveness in reducing the weighted sum of unsatisfied clauses.

4.4. Ablation Study

We conduct an ablation study to evaluate the contribution of each key component in our proposed model. By systematically removing components, we analyze their impact on performance and highlight the importance of each component. The experiments are designed to isolate the impact of the following components: (i) the hypergraph modeling of literal nodes rather than variable nodes; (ii) the transformer module with the cross-attention mechanism; (iii) the shared representation constraint loss. The results of the ablation study are shown in Table 3.

Effect of Hypergraph Modeling of Literal Nodes: The performance drops by 52.77% when the hypergraph modeling of variable nodes is used instead of literal nodes. The results demonstrate the importance and superiority of modeling the Weighted MaxSAT instance as a hypergraph with literal nodes.

Effect of Transformer with Cross-Attention: We disable the transformer module with the cross-attention mechanism to assess its importance. Without this module, the model experiences a performance drop of 11.54%. This signifi-

Table 3. The results of the ablation study. We consider three components: (i) HGM-L: the hypergraph modeling of literal nodes rather than variable nodes; (ii) Transformer: the transformer module with the cross-attention mechanism; (iii) SRCL: the shared representation constraint loss. Specifically, the first row in the table represents the transformation of the Weighted MaxSAT instance into the hypergraph of variables.

HGM-L	TRANSFORMER	SRCL	RESULT
×	×	×	182.39
✓	×	×	86.14
✓	✓	×	64.08
✓	×	✓	47.07
✓	✓	✓	41.64

cantly reduces its ability to effectively capture dependencies across each pair of complementary literals, leading to lower overall accuracy.

Effect of Share Representation Constraint Loss: We further remove the shared representation constraint loss and optimize the model with the primary task loss function to investigate the role of the unsupervised multi-objective loss. The results show that without the shared representation constraint loss, the model achieves a performance of 64.08. This performance is lower than that of the original configuration. Therefore, it highlights the importance of the shared representation constraint loss, which encourages the positive and negative literal nodes to develop distinct feature representations.

In summary, the full model consistently outperforms the ablated versions, demonstrating the synergistic effect of integrating the hypergraph modeling of literal nodes, cross-attention mechanism, and shared representation constraint loss design.

5. Conclusion

In this work, we propose HyperSAT, a novel neural approach for Weighted MaxSAT problems, addressing the challenges associated with learning the complex non-linear dependencies and sensitive objective function of this NP-hard problem. By modeling Weighted MaxSAT instances as hypergraphs, a hypergraph convolutional network is introduced as the learning model, coupled with a cross-attention mechanism to capture the logical relationships between positive and negative literals. The proposed framework incorporates an unsupervised multi-objective loss design, which not only optimizes the intrinsic objectives of the Weighted MaxSAT problem but also ensures that the representations of positive and negative literals remain distinct in the feature space.

Extensive experiments demonstrate the potential of Hyper-

SAT. The experimental results show that HyperSAT is effective in solving Weighted MaxSAT instances and outperforms state-of-the-art competitors in terms of performance. This work offers a fresh perspective on solving the Weighted MaxSAT problem through learning-based methods. Future work will focus on further integrating the model with heuristic solvers, improving the design of well-crafted solvers, and exploring its applications in other complex combinatorial problems.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

- Allouche, D., Traoré, S., André, I., De Givry, S., Katsirelos, G., Barbe, S., and Schiex, T. Computational protein design as a cost function network optimization problem. In *International Conference on Principles and Practice of Constraint Programming*, pp. 840–849. Springer, 2012.
- Audemard, G. and Simon, L. On the glucose sat solver. *International Journal on Artificial Intelligence Tools*, 27(1):1–25, 2018.
- Ba, J. L. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- Biere, A. and Fleury, M. Gimsatul, isasat and kissat entering the sat competition 2022. *Proceedings of SAT Competition*, pp. 10–11, 2022.
- Cai, S. and Zhang, X. Deep cooperation of cdcl and local search for sat. In *Theory and Applications of Satisfiability Testing—SAT 2021: 24th International Conference*, pp. 64–81. Springer, 2021.
- Cameron, C., Chen, R., Hartford, J., and Leyton-Brown, K. Predicting propositional satisfiability via end-to-end learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3324–3331, 2020.
- Clarke, E., Biere, A., Raimi, R., and Zhu, Y. Bounded model checking using satisfiability solving. *Formal methods in system design*, 19:7–34, 2001.
- Cook, S. A. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151–158, 1971.
- Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., et al. Scaling vision transformers

- to 22 billion parameters. In *International Conference on Machine Learning*, pp. 7480–7512. PMLR, 2023.
- Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3558–3565, 2019.
- Guo, W., Zhen, H.-L., Li, X., Luo, W., Yuan, M., Jin, Y., and Yan, J. Machine learning methods in solving the boolean satisfiability problem. *Machine Intelligence Research*, 20 (5):640–655, 2023.
- Heydaribeni, N., Zhan, X., Zhang, R., Eliassi-Rad, T., and Koushanfar, F. Distributed constrained combinatorial optimization leveraging hypergraph neural networks. *Nature Machine Intelligence*, pp. 1–9, 2024.
- Hoos, H. H. and Stützle, T. SATLIB: An online resource for research on sat. *Sat*, 2000:283–292, 2000.
- Kautz, H. A., Selman, B., et al. Planning as satisfiability. In *ECAI*, volume 92, pp. 359–363. Citeseer, 1992.
- Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Liu, M., Huang, P., Jia, F., Zhang, F., Sun, Y., Cai, S., Ma, F., and Zhang, J. Can graph neural networks learn to solve the maxsat problem? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 16264–16265, 2023.
- Ozolins, E., Freivalds, K., Draguns, A., Gaile, E., Zakovskis, R., and Kozlovics, S. Goal-aware neural sat solver. In *2022 International Joint Conference on Neural Networks*, pp. 1–8. IEEE, 2022.
- Selsam, D. and Bjørner, N. Guiding high-performance sat solvers with unsat-core predictions. In *Theory and Applications of Satisfiability Testing—SAT 2019: 22nd International Conference, SAT 2019*, pp. 336–353. Springer, 2019.
- Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., and Dill, D. L. Learning a sat solver from single-bit supervision. In *7th International Conference on Learning Representations*, 2019.
- Sun, L., Gerault, D., Benamira, A., and Peyrin, T. Neurogft: Using a machine learning based sat solver for cryptanalysis. In *Cyber Security Cryptography and Machine Learning: Fourth International Symposium, Proceedings 4*, pp. 62–84. Springer, 2020.
- Thornton, J. and Sattar, A. Dynamic constraint weighting for over-constrained problems. In *Pacific Rim International Conference on Artificial Intelligence*, pp. 377–388. Springer, 1998.
- Timm, N. and Botha, J. Synthesis of cost-optimal multi-agent systems for resource allocation. *arXiv preprint arXiv:2209.09473*, 2022.
- Wang, W., Hu, Y., Tiwari, M., Khurshid, S., McMillan, K., and Miikkulainen, R. Neuroback: Improving CDCL SAT solving using graph neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.
- Yang, Q., Wu, K., and Jiang, Y. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, 171(2-3):107–143, 2007.
- Zhang, W., Sun, Z., Zhu, Q., Li, G., Cai, S., Xiong, Y., and Zhang, L. Nlocalsat: boosting local search with solution prediction. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 1177–1183, 2021.