DualSAT: A Dual-Branch GNN-Transformer Framework for SAT Solving

Wenzhu Yang[1,2], Zhanshan Li[1,2*], Jingyao Li[1,2*]
[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
Email addresses: yangwz24@mails.jlu.edu.cn, lizs@jlu.edu.cn, jingyao@jlu.edu.cn

Abstract

Existing graph neural network-based methods for Boolean satisfiability (SAT) solving are generally constrained by limited local receptive fields, making it difficult to adequately capture the long-range dependencies between variables and clauses in SAT instances. Moreover, these methods overlook the multi-solution nature of SAT instances. Consequently, their expressive power and generalization performance remain limited. To address these issues, we propose DualSAT, a novel neural framework for the SAT problem, designed to enhance the phase selection heuristic of SAT solvers. Specifically, DualSAT adopts a novel dual-branch graph neural network that separately encodes local structural information and global structural information in the graph, and then fuses them through an attention-based cross-branch feature fusion module. An assignment decision head is further introduced to generate the initial assignment prediction of variables. This prediction process requires only a single forward pass, achieving a favorable balance between expressive power and inference efficiency. Furthermore, to account for the multiple solutions nature of SAT instances, we design a two-stage training strategy that combines supervised and unsupervised learning to improve the generalization performance of the model. Experimental results on multiple datasets show that, as an end-to-end assignment prediction model, DualSAT significantly outperforms NeuroSAT and NeuroBack. In addition, we integrate DualSAT into the classical SAT solver CaDiCaL, where it increases the number of solved instances by up to 11.6% and reduces the number of conflicts by up to 9.21% on SAT competition problem sets.

1 Introduction

The Boolean Satisfiability (SAT) problem aims to determine whether a given Boolean formula has a satisfying assignment. Modern SAT solvers are primarily built on the Conflict Driven Clause Learning (CDCL) framework [1]. Owing to its strong empirical performance, this framework has become one of the most widely adopted paradigms for SAT solving [2–4]. In CDCL-based solvers, the branching heuristic [5] and the phase selection heuristic [6, 7] are two key variable-related heuristic strategies that directly influence the decision process. However, the design of traditional heuristic strategies often relies heavily on extensive empirical tuning and hand-crafted rules, resulting in substantial engineering cost and limited transferability across different problem distributions. To address this limitation, recent research has increasingly explored the introduction of neural network models into SAT solving, seeking to learn more adaptive and transferable heuristic strategies through data-driven approaches [8].

Neural network-based research on SAT can be broadly categorized into two directions. The first is standalone neural solvers, which directly predict the satisfiability [9–11] or satisfying assignment [12–

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

14] of a given SAT instance. However, such methods are only effective on small-scale SAT instances and fail to outperform modern SAT solvers. The second direction is learning-aided solvers, which integrate neural networks with SAT solvers to improve the solving effectiveness and efficiency by enhancing phase selection [15, 16], variable state independent decaying sum scores [17, 18], or initial solutions of local search [19]. This line of research has emerged as a more promising research direction [8].

However, existing learning-aided methods still exhibit limited expressive power and poor scalability when solving industrial and large-scale SAT instances [4, 8]. (1) Message passing-based graph neural network methods [20, 15, 16] iteratively update node representations via multiple rounds of message passing to produce more informative node embeddings. Yet, for large-scale instances, expanding the receptive field typically requires increasing the number of message passing iterations. This not only incurs substantial computational and memory overhead, but also risks representation degradation, such as over-smoothing. (2) The graph structure of the SAT instance contains not only local constraint relations between variables and clauses, but also long-range dependencies induced by Boolean Constraint Propagation (BCP) [8]. Nevertheless, existing methods typically aggregate only local neighborhood information [9, 19, 16], lacking the capacity to capture such long-range dependencies. Although Transformer-based methods can model long-range dependencies through global attention, their time and space complexity increase quadratically with respect to the number of nodes, posing significant scalability challenges when applied to large-scale SAT instances. (3) Existing neural network SAT methods that enhance the phase selection heuristic of SAT solvers [16] typically use a single satisfying assignment of a satisfiable Boolean formula as the supervision signal. However, SAT instances often have multiple satisfying assignments, and training with supervision from a single reference satisfying assignment can bias the model toward a specific assignment, making it insufficient to reflect the overall structure of the solution space, thereby limiting the generalization ability of the model.

To address the above problems, we propose DualSAT, a novel neural network framework for SAT solving that predicts the initial assignment of a given SAT instance. DualSAT designs two independent branches, the structural embedding branch and the semantic embedding branch, which encode local structural information from neighboring nodes and long-range semantic information over the graph, respectively. The embeddings from the two branches are ultimately fused through an Attention-based Cross-Branch Fusion (ACBF) module. Our model requires only a single forward pass to predict the initial assignment of a SAT instance. This design avoids the computational overhead associated with the repeated iterative updates required by Graph Neural Network (GNN)-based methods, thereby offering superior time and memory efficiency on large-scale SAT instances. In addition, we adopt a two-stage training strategy. In the first stage, supervised and unsupervised losses are jointly optimized to guide the model toward the feasible solution region. In the second stage, training proceeds using only the unsupervised loss, enabling the model to further adapt within the feasible solution space and thereby achieve better generalization.

Finally, we integrate DualSAT into the CDCL SAT solver [21]. During the initialization stage, the initial assignment predicted by DualSAT is used to enhance the phase selection heuristic of the solver, thereby reducing the backtracking search caused by conflicts. To our knowledge, DualSAT is the first learning-aided SAT framework that explicitly leverages both structural and semantic information during the encoding process to guide variable assignment prediction. We further validate the effectiveness of DualSAT on the modern SAT solver CaDiCaL [21].

Appendix A includes a full discussion of **related work**.

2 DualSAT

2.1 Overview

To reduce the computational overhead associated with multi round message passing in GNN-based SAT methods while improving SAT solving performance, DualSAT adopts a multi-layer GNN architecture that predicts variable phases in a single forward pass, thereby enhancing the phase-selection heuristic of CDCL SAT solvers. Figure 1 shows the overall framework of DualSAT. First, DualSAT converts the input Conjunctive Normal Form (CNF) formula into the Variable-Clause Graph (VCG), and uses the structural branch and the semantic branch to model the local structural information and global structural information of the graph, respectively. Subsequently, the two embeddings are fed into the ACBF module for cross-branch information alignment and complementary fusion. Finally, the assignment decision head outputs the phase prediction for each

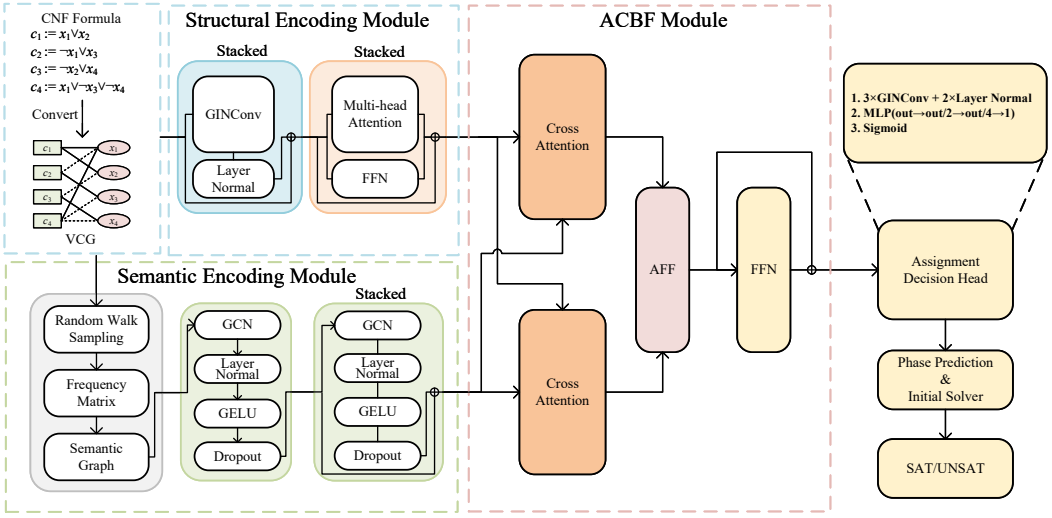


Figure 1: The overview of DualSAT framework.

variable and provides it to the SAT solver as an initial assignment, thereby guiding the solving process. Since model inference is executed only once before the solving process begins, DualSAT is able to provide effective guidance while maintaining low overhead.

2.2 Problem formulation

A Boolean formula ϕ consists of a set of Boolean variables $\{x_i\}_{i=1}^{n} \in \{\text{true}, \text{false}\}$ and Boolean operators (AND ($\wedge$), OR ($\vee$), NOT ($\neg$), ...). The goal of the SAT problem is to determine whether there exists an assignment to the variables such that ϕ evaluates to true. Any Boolean formula can be transformed into CNF in linear time [22], where the formula is expressed as a conjunction of clauses, $\phi = (c_1 \wedge c_2 \wedge \cdots \wedge c_m)$. Each clause c_j is a disjunction of literals, $c_j = (\ell_{j1} \vee \ell_{j2} \vee \cdots \vee \ell_{j|c_j|})$, $j = 1, \ldots, m$, and each literal $\ell \in \{x_i, \neg x_i\}$. A CNF formula has a satisfying assignment if and only if every clause contains at least one literal that is true. Eq. (1) provides a CNF example, it contains four clauses $c_1 = (x_1 \vee x_2),\ c_2 = (\neg x_1 \vee x_3),\ c_3 = (\neg x_2 \vee x_4),\ c_4 = (x_1 \vee \neg x_3 \vee \neg x_4)$, the literals include $x_1, \neg x_1, x_2, \neg x_2, x_3, \neg x_3, x_4, \neg x_4$, and the variable set is $\{x_1, x_2, x_3, x_4\}$. One satisfying assignment for ϕ is $x_1 = \text{true},\ x_2 = \text{false},\ x_3 = \text{true},\ x_4 = \text{true}$.

$$\phi := (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4). \tag{1}$$

To fully exploit the structural information implied in CNF, we represent a SAT instance ϕ as a VCG. A VCG is a bipartite graph representation of a CNF, where two types of nodes are used to represent variables and clauses, respectively. Specifically, a SAT instance is represented as an undirected graph with edge relation types $G = (V, E, R)$, where V is the vertex set and is partitioned into two disjoint subsets V_1 and V_2. Here, V_1 denotes the set of variable nodes, and V_2 denotes the set of clause nodes. The edge set $E \subseteq V_1 \times V_2$, for any $x_i \in V_1$ and $c_j \in V_2$, if $(x_i, c_j) \in E$ means that variable x_i appears in clause c_j. To characterize the polarity of a variable in a clause, we introduce a relation $R : E \to \{1, 0\}$ on each edge, where $R(x_i, c_j) = 1$ indicates that the positive literal x_i appears in clause c_j, and $R(x_i, c_j) = 0$ indicates that the negative literal $\neg x_i$ appears in clause c_j. Figure 2 shows the VCG representation corresponding to Eq. (1). For notational convenience in the subsequent model formulation, we use $v_i, v_j \in V$ to denote generic nodes and $(v_i, v_j) \in E$ to denote a generic edge.

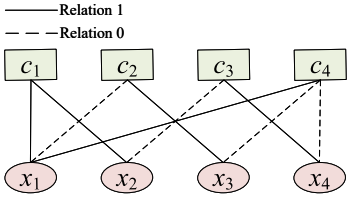


Figure 2: Example of variable-clause graph.

2.3 Structural encoding for local structure

The structural encoding module is designed to extract local structural information from the VCG of SAT instances, as shown in Figure 1. It consists of two main components. First, a relation-aware GINConv [23] is employed to capture discriminative local constraint relationships. Then, a relation-aware multi-head graph attention mechanism is introduced to adaptively weight the neighborhood structural information, enabling selective aggregation of key information. In addition, residual connections and normalization strategies are incorporated in deep stacking to stabilize training and alleviate the over-smoothing problem.

2.3.1 Relation aggregation

Given a SAT instance represented as a VCG $G = (V, E, R)$, the node features are denoted as $H \in \mathbb{R}^{n \times d}$, where $n = |V|$ and d is the feature dimension of each node. The feature vector of the node v_i is denoted as $h_i \in H$. For any edge $(v_i, v_j) \in E$, its discrete relation type is $r_{i,j} \in R$.

Given the initial node features H and the set of edge relation types R, we update the node representations using a relation-aware GINConv. Although standard GINConv is expressive, it treats all edges equally during neighborhood aggregation by default, making it difficult to model diverse relation types in VCG. Inspired by NeuroBack, we parameterize each relation type independently. In the l-th layer of relation-aware GINConv, the feature representation of node v_i is updated as:

$$h_i^{(l)} = \sum_{r \in R} \mathrm{MLP}_r^{(l)} \left(\left(1 + \epsilon_r^{(l)}\right) h_i^{(l-1)} + \sum_{j \in \mathcal{N}_r(i)} h_j^{(l-1)} \right), \tag{2}$$

where $\mathcal{N}_r(i)$ denotes the set of neighbors of v_i connected via relation r , and ϵ is a learnable parameter.

2.3.2 Graph self-attention enhancement

Relying solely on the node embeddings obtained above is still insufficient to fully model the complex structural information in the SAT instance. We therefore stack multiple relation-aware multi-head graph attention layers to aggregate key information selectively. Building on the dynamic attention scoring mechanism of GAT [24], we introduce relation-specific projection and attention parameters, and adopt intra-relation normalization to prevent neighbors associated with different relation types from competing with each other during the attention allocation.

In GAT attention scoring, for each relation type r, we learn a set of linear transformations to construct the query, key, and value representations, $q_i^{(r)} = \Theta_q^{(r)} h_i, \quad k_j^{(r)} = \Theta_k^{(r)} h_j, \quad v_j^{(r)} = \Theta_v^{(r)} h_j$, where $\Theta_q^{(r)}, \Theta_k^{(r)}, \Theta_v^{(r)} \in \mathbb{R}^{d_{\mathrm{head}} \times d}$ are relation-specific parameters, and d_{head} and d denote the output dimension of a single attention head and the input feature dimension, respectively.

To define relation-aware attention, we use $(v_j \to v_i, r)$ to denote an edge from node v_j to node v_i with relation type r. The attention on edge $(v_j \to v_i, r)$ can then be defined as:

$$e_{i,j}^{(r)} = \left(a^{(r)}\right)^{\top} \sigma\left(q_i^{(r)} + k_j^{(r)}\right), \tag{3}$$

where $a^{(r)} \in \mathbb{R}^{d_{\mathrm{head}}}$ is a relation-specific attention vector, and $\sigma(\cdot)$ denotes a nonlinear activation function.

We use intra-relation normalization, applying the softmax independently to the neighborhood $\mathcal{N}_r(i)$ of node v_i for each relation type r:

$$\alpha_{i,j}^{(r)} = \frac{\exp\left(e_{i,j}^{(r)}\right)}{\sum_{j' \in \mathcal{N}_r(i)} \exp\left(e_{i,j'}^{(r)}\right)}. \tag{4}$$

For each relation r, we first fuse node features via intra-relation attention-weighted aggregation, and then fuse the contributions from all relation types:

$$h_i' = \sum_{r \in R} \sum_{j \in \mathcal{N}_r(i)} \alpha_{i,j}^{(r)} v_j^{(r)}. \tag{5}$$

Under the multi-head attention setting, for each relation r we replicate M groups of parameter $\{\Theta_q^{(r,m)}, \Theta_k^{(r,m)}, \Theta_v^{(r,m)}, a^{(r,m)}\}_{m=1}^{M}$ compute the output of each attention head $h_i^{'(r,m)}$, and finally take the average over all heads:

$$\hat{h}_i = \sum_{r \in R} \frac{1}{M} \sum_{m=1}^{M} h_i^{'(r,m)}. \tag{6}$$

Finally, we combine the above attention with a Feed Forward Network (FFN) to form a structural attention block, and introduce residual connections within each block to enhance the expressive power of the model, stabilizing gradient propagation under deep stacking, and alleviating over-smoothing. Specifically, the node representation update within the structural attention block is defined as:

$$H^{(b+1)} = H^{(b-1)} + \hat{H}^{(b)} + \mathrm{FFN}\left(H^{(b-1)}\right), \tag{7}$$

where $H^{(b-1)} = \{ h_i \mid i = 1, \ldots, n \}$ denotes the output features of the $(b-1)$-th attention block, and $\hat{H}^{(b)} = \{ \hat{h}_i \mid i = 1, \ldots, n \}$ denotes the multi-head attention output features within the b-th attention block. We denote the output of the final structural attention block as H_{struct}.

2.4 Semantic encoding for global structure

To enhance the ability of the model to capture long-range dependencies, we introduce a semantic encoding module in addition to the local structural embedding to complement global graph structural information, as shown in Figure 1. Specifically, we perform random walks on the graph and randomly sample a large number of fixed-length node sequences to obtain richer contextual information. We then count the co-occurrence times of node pairs within the sliding window to construct a co-occurrence frequency matrix F. Based on F, we estimate the co-occurrence probability of node pairs and convert it into a semantic association matrix $S \in \mathbb{R}^{n \times n}$ using Positive Pointwise Mutual Information (PPMI) [25], which highlights highly correlated node pairs while suppressing noise arising from incidental co-occurrences. Finally, we construct a semantic graph based on S and feed it into the GNN for message passing and feature aggregation to preserve global graph structural information.

2.4.1 Random walk-based context construction

To capture contextual information beyond local neighborhoods, we perform random walks on the VCG G to sample node sequences. A random walk can be viewed as a Markov chain over the graph, where each state transition depends only on the connectivity of the one-hop neighbors of the current node. We use the adjacency matrix $A \in \mathbb{R}^{n \times n}$ to represent the graph structure, and define the one-step transition probability from node v_i to its neighbor v_j as follows:

$$P(s(t+1) = v_j \mid s(t) = v_i) = \frac{A_{i,j}}{\sum_{j'} A_{i,j'}}, \tag{8}$$

where $s(t) = v_i$ indicates that the random walk is at node v_i at time step t.

We use the node sequences generated by random walks as sequential samples for context modeling, thereby characterizing node co-occurrence relationships in the sequence space. Starting from each node, we sample K random walk sequences of length T, and use a sliding window of size w on each sequence to extract the context nodes for every target node in the sequence.

2.4.2 PPMI-based semantic association

Let $\pi = (v_1, \ldots, v_T)$ denote a node sequence obtained by random walk sampling. For the node v_i at position i in the sequence, its context window is defined as $\mathcal{C}(v_i) = \{ v_j \mid 1 \le j \le T,\ j \ne i,\ |i-j| \le w \}$.

We initialize a frequency matrix $F \in \mathbb{R}^{n \times n}$ to zero to record the co-occurrence counts of node pairs within the window, and update it as $F_{i,j} \leftarrow F_{i,j} + 1,\ \forall\, v_j \in \mathcal{C}(v_i)$, where $F_{i,j}$ denotes the number of times that node v_j appears in the context window of node v_i. The co-occurrence counts are subsequently normalized into co-occurrence probabilities, i.e., $p_{i,j} = \frac{F_{i,j}}{\sum_{i',j'} F_{i',j'}}$, based on which PPMI is applied to encode the semantic information of the original graph. We then define the semantic association matrix $S \in \mathbb{R}^{n \times n}$ as:

$$S_{i,j} = \max\left(\log \frac{p_{i,j}}{\sum_{j'} p_{i,j'} \sum_{i'} p_{i',j}},\ 0 \right), \tag{9}$$

where $\sum_{j'} p_{i,j'}$ and $\sum_{i'} p_{i',j}$ denote the marginal co-occurrence probabilities of node v_i and node v_j, respectively. To reduce the computational and storage overhead, we sparsify S by retaining only the top-k neighbors with the highest semantic association scores for each node.

2.4.3 Semantic embedding layer with global structure

After obtaining the semantic association matrix S, we binarize it and treat it as the global semantic adjacency relation, which is used to characterize whether a semantic association exists between a pair of nodes. Specifically, we define the adjacency matrix of the semantic graph as $\tilde{A} \in \{0,1\}^{n \times n}$, where $\tilde{A}_{i,j} = \mathbb{I}(S_{i,j} > 0)$ and $\mathbb{I}(\cdot)$ is the indicator function. This yields a semantic graph $\tilde{G} = (V, \tilde{E})$, where $\tilde{E} = \{(v_i, v_j) \mid \tilde{A}_{ij} = 1\}$. Based on $\tilde{G}$, we further apply a GNN to perform message passing and neighborhood aggregation, thereby extracting semantic features. We adopt the GCNII [26] for message passing, and its update rule is:

$$\tilde{H}^{(l+1)} = \sigma\left(\left((1-\gamma)\hat{P}\tilde{H}^{(l)} + \gamma \tilde{H}^{(0)} \right) \left((1-\beta_l)\mathbf{I} + \beta_l \Theta^{(l)} \right) \right), \tag{10}$$

where $\hat{P} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ is the graph convolution matrix with the renormalization trick, $\hat{A} = \tilde{A} + \mathbf{I}$, $\hat{D} = \tilde{D} + \mathbf{I}$, $\tilde{D}$ is the diagonal degree matrix of $\tilde{G}$, $\Theta^{(l)}$ is the l-th weight matrix, $\tilde{H}^{(0)}$ denotes the

initial node features, γ controls the strength of the initial residual, and β_l controls the interpolation between the identity mapping and the learnable transformation. This design supports deep stacking to model long-range dependencies, while effectively alleviating over-smoothing and stabilizing training. After multiple convolutional layers, we obtain the final semantic embedding H_{sem}.

2.5 Aggregation module

The structural encoding module and the semantic encoding module output the local structural embedding H_{struct} and the semantic embedding H_{sem} of the VCG, respectively. To fully exploit their complementary information and learn effective representations for downstream tasks, we feed H_{struct} and H_{sem} into the ACBF module, which uses an attention mechanism to align and fuse the two embeddings, producing final representations that capture both local consistency and global semantic awareness.

2.5.1 Bidirectional cross-attention for feature alignment

Since H_{struct} and H_{sem} are produced by different branches, they often lie in different feature distributions. To achieve cross-branch feature alignment, we introduce bidirectional cross-attention. For clarity, we denote the branch being updated as the target branch tg, and the branch providing information to it as the source branch sr.

In each cross-attention unit, we perform attention-based neighborhood aggregation and apply separate linear transformations to the target branch and the source branch. For a target node v_i , the cross-branch attention is computed as $q_i = W_q h_i^{(\mathrm{tg})}, k_j = W_k h_j^{(\mathrm{sr})}, v_j = W_v h_j^{(\mathrm{sr})}$, where $h_i^{(\mathrm{tg})}$ denotes the feature of node v_i in the target branch, and $h_j^{(\mathrm{sr})}$ denotes the feature of node v_j in the source branch. The attention scoring is as follows:

$$\alpha_{i,j} = \mathrm{softmax}_{j\in\mathcal{N}_t(i)}\left(a^\top \sigma\left([q_i \parallel k_j]\right)\right), \tag{11}$$

where $\mathcal{N}_t(i)$ denotes the neighborhood of node v_i in the graph corresponding to the target branch, a is a learnable attention parameter and $\parallel$ denotes vector concatenation. Under multi-head attention, the cross-branch update of a target node is defined as the average of the outputs from all attention heads:

$$\tilde{h}_i^{(\mathrm{tg})} = \frac{1}{M}\sum_{m=1}^{M}\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(m)} v_j^{(m)}. \tag{12}$$

To stabilize training, we apply normalization before and after the cross-attention operation, followed by residual connections. The bidirectional updates are as follows:

$$Z_{\mathrm{sem}\to\mathrm{str}} = \mathrm{LN}\left(H_{\mathrm{struct}} + \mathrm{CA}\left(\mathrm{LN}(H_{\mathrm{struct}}), \mathrm{LN}(H_{\mathrm{sem}}); E\right)\right), \tag{13}$$

$$Z_{\mathrm{str}\to\mathrm{sem}} = \mathrm{LN}\left(H_{\mathrm{sem}} + \mathrm{CA}\left(\mathrm{LN}(H_{\mathrm{sem}}), \mathrm{LN}(H_{\mathrm{struct}}); \tilde{E}\right)\right), \tag{14}$$

where LN denotes layer normalization, and $\mathrm{CA}(\cdot)$ denotes the cross-attention operation.

2.5.2 Attention-based feature fusion and FFN

The two enhanced representations obtained by bidirectional cross-attention may contain inconsistent information in different dimensions. Therefore, we adopt an Attention-based Feature Fusion (AFF) module to adaptively fuse them.

For each node v_i, we concatenate the two representations and feed the result into an MLP to obtain a fused representation:

$$\bar{H} = \mathrm{MLP}\left([\, Z_{\mathrm{sem}\to\mathrm{str}} \parallel Z_{\mathrm{str}\to\mathrm{sem}} \,]\right). \tag{15}$$

We then compute fusion weights for the two branches as:

$$[\omega_{\mathrm{str}}, \omega_{\mathrm{sem}}] = \mathrm{softmax}\left([W_1\bar{H}, W_2\bar{H}]\right), \tag{16}$$

where ω_{str} and ω_{sem} denote the branch-wise fusion weights, and W_1 and W_2 are learnable parameters. The final fused output is:

$$H_{\mathrm{fuse}} = \omega_{\mathrm{str}} Z_{\mathrm{sem}\to\mathrm{str}} + \omega_{\mathrm{sem}} Z_{\mathrm{str}\to\mathrm{sem}}. \tag{17}$$

This allows the model to dynamically select more reliable information sources at the feature level, achieving complementary fusion.

Finally, we apply a standard FFN module to H_{fuse}. The FFN consists of a two-layer MLP with GELU activation, together with dropout, residual connections, and layer normalization, to further enhance representation capacity and stabilize training. The resulting representations are then passed to the subsequent assignment decision head, which outputs variable phase predictions.

2.6 Training strategies

We propose a two-stage training strategy. In the first stage, we train the model with a combination of a BCE loss $\mathcal{L}_{\mathrm{sup}}$ and an unsupervised loss $\mathcal{L}_{\mathrm{unsup}}$ defined in terms of the degree of constraint

violation. The supervised term provides a clear learning objective for the model at the early stage of training, thereby accelerating convergence toward regions near feasible solutions. The unsupervised loss mitigates overfitting of the BCE loss to a single reference satisfying assignment. It guides the optimization process toward reducing constraint violations, thereby alleviating the loss fluctuations caused by the transition from supervised to unsupervised training in the second stage. The supervised loss is defined as:

$$\mathcal{L}_{\text{sup}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right], \tag{18}$$

where p_i denotes the predicted probability that variable x_i is assigned true. For a literal ℓ, if $\ell = x_i$, then $p(\ell = 1) = p_i$, if $\ell = \neg x_i$, then $p(\ell = 1) = 1 - p_i$. Here, $y_i \in \{0, 1\}$ denotes the ground truth label of the reference satisfying assignment for variable x_i. We define the unsupervised loss based on the degree of constraint violation for SAT instances (see Appendix B):

$$\mathcal{L}_{\text{unsup}} = -\log \mathbb{P}(\phi = 1) = -\sum_{j=1}^{m} \log\left(1 - \exp\left(s_{c_j}\right)\right), \tag{19}$$

where $s_{c_j} = \sum_{\ell \in L_j} \log\left(1 - \mathbb{P}(\ell = 1)\right)$, c_j denotes the j-th clause, L_j denotes the set of literals in clause c_j, and $\mathbb{P}(\ell = 1)$ denotes the probability that literal ℓ is true. Moreover, $\mathbb{P}(\phi = 1)$ denotes the probability that the CNF formula ϕ is satisfied under the variable assignment distribution predicted by the model. Therefore, the total loss in the first stage is:

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{unsup}}. \tag{20}$$

In the second stage, after the model reaches a predefined epoch, we remove the supervised term $\mathcal{L}_{\text{sup}}$ and continue training using only the unsupervised loss $\mathcal{L}_{\text{unsup}}$:

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{unsup}}. \tag{21}$$

This stage enables the model to reduce its reliance on a single reference satisfying assignment and to perform self-correction under the constraint-driven objective, thereby learning more generalizable solving representations and inference behaviors.

Overall, the first stage provides effective initialization guidance, while the second stage improves the generalization ability of the model through pure constraint-based optimization.

3 Combination with SAT solver

Most modern SAT solvers [7, 21, 27, 28] are built on the CDCL framework. We integrate DualSAT with the classic SAT solver CaDiCaL, using the phase predictions generated by DualSAT to enhance the phase selection heuristic of CaDiCaL and thereby improve solving performance. Further details of the CDCL workflow and the integration of DualSAT with CaDiCaL are provided in Appendix C.

4 Experiments

In this section, we conduct four sets of experiments. First, we compare DualSAT with two representative neural SAT methods, NeuroSAT and NeuroBack, to evaluate its effectiveness as an end-to-end assignment prediction model. Second, we further assess its generalization performance on larger scale datasets. Third, we perform ablation studies to analyze the contribution of each module and the effectiveness of the overall design. Fourth, we integrate DualSAT with the SAT solver CaDiCaL and evaluate its ability to improve solver performance.

4.1 Experimental setting

In this section, we describe the evaluation metrics used in our experiments. Detailed experimental data and the hyperparameter configuration of DualSAT are provided in Appendix D. Unless otherwise specified, all models in the following experiments use the hyperparameter settings described here.

Evaluation metric. According to the experimental objectives, we evaluate our method using two metrics. (1) Since DualSAT is designed to generate an initial assignment for a given SAT instance, we report the solving accuracy of the initial assignments predicted by DualSAT. (2) After integrating DualSAT with CaDiCaL, we evaluate its effect on solving efficiency and search behavior under different initial assignment settings from two aspects, the number of instances solved successfully and the number of conflicts generated.

4.2 Performance comparison with existing methods

We compare DualSAT with other neural baselines for satisfiable assignment prediction, including NeuroSAT and NeuroBack. NeuroSAT is a pioneering method and has been widely adopted as a baseline in prior studies [15, 18, 29, 30]. Since the original NeuroSAT is designed for SAT/UNSAT

Table 1: Solving accuracy (%) of generated initial assignments on the easy and medium datasets. "Avg." denotes the average solving accuracy. The best results are indicated in bold.

Method	Easy Datasets					Medium Datasets				
	SR	CA	k-Clique	k-Domset	Avg.	SR	CA	k-Clique	k-Domset	Avg.
NeuroSAT	49.6	2.0	64.1	62.4	44.5	5.3	0.2	**50.4**	45.0	25.2
NeuroBack	10.4	4.4	12.3	14.1	10.3	1.1	0.1	7.8	10.3	4.8
DualSAT	**56.0**	**69.0**	**75.5**	**91.4**	**73.0**	**14.5**	**28.4**	45.6	**74.7**	**40.8**

Table 2: Solving accuracy (%) of generated initial assignments under different generalization settings.

Method	Easy → Medium					Easy → Hard					Medium → Hard				
	SR	CA	k-Clique	k-Domset	Avg.	SR	CA	k-Clique	k-Domset	Avg.	SR	CA	k-Clique	k-Domset	Avg.
NeuroSAT	2.5	0.1	25.0	32.0	14.9	1.3	0.0	2.0	5.7	2.3	1.7	0.0	14.6	19.0	8.8
NeuroBack	0.7	0.0	3.2	6.6	2.6	0.1	0.0	0.5	1.4	0.5	0.3	0.0	1.8	3.1	1.3
DualSAT	**12.7**	**19.4**	**40.1**	**41.9**	**28.5**	**5.8**	**8.1**	**11.4**	**13.2**	**9.6**	**7.3**	**13.4**	**21.5**	**38.2**	**20.1**

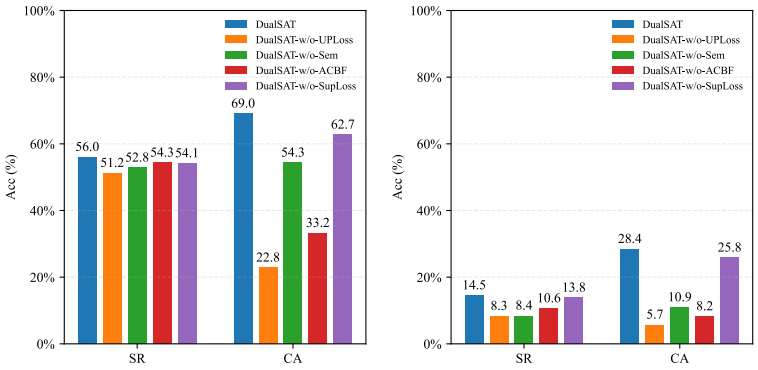
classification, we follow NSNet [15] to adapt its output head for satisfying assignment prediction and employ the same training strategy as DualSAT. NeuroBack is the latest neural-aided SAT solving approach for enhancing the phase selection heuristic in CDCL SAT solvers. In the evaluation, since NeuroBack predicts phases only for backbone variables, to ensure fairness, we adapt its original implementation to additionally predict phases for non-backbone variables, while keeping all other settings the same as in the original paper. Table 1 reports the solving accuracy of the initial assignments generated by each method on the easy and medium datasets.

The following observations can be drawn from Table 1. Compared with NeuroSAT and NeuroBack, DualSAT consistently achieves stable and significant performance advantages across all tasks. On the easy datasets, the average accuracies of NeuroSAT and NeuroBack over the four problem types are 44.5% and 10.3%, respectively. Whereas DualSAT reaches 73.0%, improving over them by 28.5% and 62.7%. As the problem scale increases, i.e., on the medium difficulty datasets, the accuracy of all three methods drops overall. But DualSAT remains the best, with an average accuracy of 40.8%, outperforming NeuroSAT (25.2%) by 15.6% and NeuroBack (4.8%) by 36.0%. It is worth noting that DualSAT performs particularly well on the pseudo-industrial CA task, where the other two methods achieve poor results. Specifically, on CA, DualSAT already significantly outperforms NeuroSAT and NeuroBack on the easy dataset, and the advantage is further expanded on the medium dataset. This suggests that as instances become larger and structurally more complex, the advantage of DualSAT becomes increasingly pronounced. We attribute this to the fact that conventional GNN-based methods mainly rely on local neighborhood aggregation for representation learning, which makes it difficult to adequately capture the complex long-range dependencies between variables and clauses in SAT. This limitation is further amplified in large-scale instances. In contrast, the semantic branch introduced in DualSAT can capture long-range semantic correlations and better leverage the global graph structure, leading to superior performance on the medium difficulty datasets, which further validates the advantages of our architecture.

4.3 Generalization to larger instances

To evaluate the generalization ability of DualSAT across instance scale, we train the model on the easy and medium datasets, respectively, and test it on larger scale datasets. Specifically, a model trained on the easy dataset is evaluated on the medium and hard datasets, and a model trained on the medium dataset is evaluated on the hard dataset. For a fair comparison, we train and evaluate NeuroSAT and NeuroBack using the same dataset. Table 2 presents the generalization performance of the three methods.

The results show that DualSAT significantly outperforms NeuroSAT and NeuroBack on all three generalization tasks. Moreover, the larger the scale gap between the training and test instances, the more pronounced the advantage of DualSAT becomes. This is consistent with our expectation that relying solely on supervised learning often causes the model to be biased toward a single reference satisfying assignment, thereby limiting its ability to explore other feasible solutions. Such bias can weaken the generalization capability of the model, especially when the solution space is larger and contains more feasible assignments. Since DualSAT introduces unsupervised learning in addition to supervised learning, it can alleviate the neural confusion caused by multiple solutions in SAT



(a) Easy dataset. (b) Medium dataset.

Figure 3: Performance of DualSAT and its variants.

instances. This phenomenon is particularly evident on the hard CA dataset, further validating the effectiveness of the two-stage training strategy.

In addition, we observe that training on the medium datasets generally yields better generalization than training on the easy datasets. This suggests that training on larger scale and structurally more complex instances can improve the ability of the model to adapt to more challenging SAT instances.

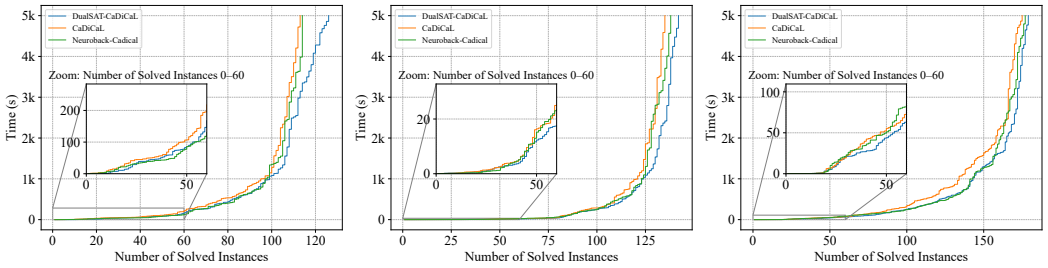
4.4 Ablation study

In this section, to evaluate the contribution of each component in DualSAT, we conduct ablation studies on its key components, including the semantic branch and the ACBF module, on the SR and CA datasets with easy and medium difficulty, respectively. For the variant without the semantic branch (DualSAT-w/o-Sem), we retain only the features from the structural branch in the subsequent fusion stage. For the variant without the ACBF module (DualSAT-w/o-ACBF), we replace the original ACBF module with a sum. In addition, to assess the effect of the two-stage training strategy, we train one variant using only the supervised BCE loss (DualSAT-w/o-UPLoss) and another using only the unsupervised loss defined based on the degree of constraint violation (DualSAT-w/o-SupLoss), while keeping all other settings unchanged. The results are reported in Figure 3.

Effectiveness of semantic branch. We compare DualSAT with DualSAT-w/o-Sem to evaluate the contribution of the semantic branch. As shown in Figure 3, compared with DualSAT, DualSAT-w/o-Sem shows varying degrees of performance degradation, with the performance gap becoming most pronounced on the medium datasets. This indicates that as instance scale increases, the semantic branch plays a more critical role, since it helps capture long-range dependencies and complements global graph structural information.

Effectiveness of ACBF module. The ACBF module is designed to align the representations from the structural and semantic branches and to perform complementary fusion of information. As shown in Figure 3, the performance of DualSAT-w/o-ACBF decreases significantly. This suggests that, compared with directly using sum for fusion, the ACBF module can more effectively align structural and semantic information within a unified representation space, thereby mitigating the representation shift and scale inconsistency caused by modeling the two branches independently. Moreover, through its adaptive fusion mechanism, ACBF can emphasize complementary information that is more relevant to the task objective while suppressing redundant or interfering signals. As a result, it learns a more discriminative and robust joint representation, thereby improving both representation quality and solving performance of the model.

Effectiveness of two-stage training strategy. Figure 3 shows the results for DualSAT-w/o-UPLoss and DualSAT-w/o-SupLoss. It can be observed that the performance degradation of DualSAT-w/o-UPLoss is the most significant, particularly on the CA dataset. Compared with DualSAT, DualSAT-w/o-UPLoss drops by 46.2% and 22.7% on the easy and medium difficulty datasets, respectively. We attribute this to the fact that CA instances generally admit more feasible solutions than SR instances [31]. In this case, relying solely on the supervised loss tends to bias the model toward a single reference satisfying assignment, whereas the unsupervised loss defined in terms of constraint violation helps alleviate this issue. In contrast, removing the supervised loss leads to only a modest performance drop, and the DualSAT-w/o-SupLoss remains close to the full model overall. This



(a) SATCOMP-2022. (b) SATCOMP-2023. (c) SATCOMP-2024.

Figure 4: Cactus plots of the solving progress of CaDiCaL, NeuroBack-CaDiCaL, and DualSAT-CaDiCaL on the SATCOMP 2022–2024 datasets, showing the cumulative number of solved instances over time.

indicates that, in the two-stage training strategy, the unsupervised loss plays a dominant role during model training, while the supervised loss mainly provides early stage guidance that helps the model reach regions with lower constraint violation more efficiently.

4.5 Combination with SAT solver

In this subsection, we integrate DualSAT and NeuroBack into CaDiCaL, denoted as DualSAT-CaDiCaL and NeuroBack-CaDiCaL, respectively, and compare DualSAT-CaDiCaL with NeuroBack-CaDiCaL and the baseline solver CaDiCaL to evaluate the effectiveness of DualSAT. We use all 1,200 CNF formulas from the main track of SATCOMP 2022–2024 as the test dataset, experimental details are provided in Appendix E.2. To satisfy the 10GB memory constraint, we retain only those instances with fewer than 2×10^5 nodes on which inference can be completed. Consequently, in the SATCOMP 2022–2024 datasets, 149, 214, and 224 instances are ultimately included in the experiments, respectively.

Time of solve instances. The cactus plots in Figure 4 shows the relationship between solving time and the cumulative number of instances solved by CaDiCaL, NeuroBack-CaDiCaL, and DualSAT-CaDiCaL. It is evident that DualSAT-CaDiCaL solves the highest number of instances across all three datasets, and this advantage becomes more pronounced on harder instances that require longer solving time. Specifically, on SATCOMP-2022, CaDiCaL solves 112 instances, NeuroBack-CaDiCaL solves 113, whereas DualSAT-CaDiCaL solves 125, corresponding to improvements of 11.6% and 10.6%, while also reducing the average solving time by 91.6 and 18.7 seconds compared with the two methods, respectively. On SATCOMP-2023, the three solvers solve 134, 137 and 141 instances, respectively, corresponding to improvements of 5.2% and 2.9%, and an average time reductions of 112.41 and 78.30 seconds. On SATCOMP-2024, they solve 174, 176 and 178 instances, respectively, corresponding to improvements of 2.3% and 1.1%, and an average time reductions of 208.55 and 44.46 seconds. These results suggest that the phase initialization provided by DualSAT is more effective than both the default phase initialization strategy of CaDiCaL and the NeuroBack-based strategy, leading to improvements in both solving capability and solving efficiency. In addition, although NeuroBack achieves relatively low assignment prediction accuracy, NeuroBack-CaDiCaL solves 1, 3, and 2 more instances than the original CaDiCaL on SATCOMP 2022–2024, respectively, indicating that its predictions can still improve the phase initialization of the solver. To further analyze how DualSAT influences the search behavior of CDCL SAT solvers, we report additional experiments in Appendix E.2, where we compare the original CaDiCaL and DualSAT-CaDiCaL in terms of the average number of conflicts generated when successfully solving satisfiable instances.

5 Conclusion

In this paper, we propose DualSAT, a dual-branch learning framework for generating satisfiable initial assignments for SAT solvers. The innovative architecture of DualSAT includes a dual-branch encoding module and an ACBF module, which substantially enhances the expressive power and generalization ability of graph representation learning for SAT solving. In addition, we design a two-stage training strategy that effectively mitigates the optimization challenges arising from the multi-solution nature of SAT instances. Experimental results demonstrate that DualSAT achieves strong performance in solving accuracy, generalization to larger instances, and integration with CaDiCaL, highlighting both its effectiveness and practical potential. In Appendix F, we discuss possible limitations and potential future work of DualSAT.

Acknowledgments and Disclosure of Funding

This work was supported by the Natural Science Research Foundation of Jilin Province of China under Grant No. 20260102306JC, and the National Natural Science Foundation of China under Grant No. 62607037.

References

[1] Joao Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. *Handbook of Satisfiability*, pages 131–153, 2009.

[2] Armin Biere, Hans van Maaren, and Toby Walsh. *Handbook of Satisfiability*. SAGE Publications Limited, 2009.

[3] Weiwei Gong and Xu Zhou. A survey of SAT solver. In *AIP Conference Proceedings*, volume 1836, page 020059. AIP Publishing LLC, 2017.

[4] Zhihui Xie, Xu Liu, and Shuai Li. Multi-armed bandit algorithms for the Boolean Satisfiability Problem: A survey. In *Proceedings of the International Symposium on Combinatorial Search*, volume 18, pages 221–229, 2025.

[5] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. Learning rate based branching heuristic for SAT solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 123–140. Springer, 2016.

[6] Armin Biere and Mathias Fleury. Chasing target phases. In *Workshop on the Pragmatics of SAT*, 2020.

[7] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proceedings of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[8] Wenxuan Guo, Hui-Ling Zhen, Xijun Li, Wanqian Luo, Mingxuan Yuan, Yaohui Jin, and Junchi Yan. Machine learning methods in solving the boolean satisfiability problem. *Machine Intelligence Research*, 20(5):640–655, 2023.

[9] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*, 2018.

[10] Sebastian Jaszczur, Henryk Michalewski, et al. Neural heuristics for SAT solving. *arXiv preprint arXiv:2005.13406*, 2020.

[11] Chris Cameron, Rex Chen, Jason Hartford, and Kevin Leyton-Brown. Predicting propositional satisfiability via end-to-end learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3324–3331, 2020.

[12] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-SAT: An unsupervised differentiable approach. In *International Conference on Learning Representations*, 2018.

[13] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. PDP: A general neural framework for learning constraint satisfaction solvers. *arXiv preprint arXiv:1903.01969*, 2019.

[14] Emils Ozolins, Karlis Freivalds, Andis Draguns, Eliza Gaile, Ronalds Zakovskis, and Sergejs Kozlovics. Goal-aware neural SAT solver. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[15] Zhaoyu Li and Xujie Si. NSNet: A general neural probabilistic framework for satisfiability problems. *Advances in Neural Information Processing Systems*, 35:25573–25585, 2022.

[16] Wenxi Wang, Yang Hu, Mohit Tiwari, Sarfraz Khurshid, Kenneth McMillan, and Risto Miikkulainen. NeuroBack: Improving CDCL SAT solving using graph neural networks. *arXiv preprint arXiv:2110.14053*, 2021.

[17] Ziyang Zheng, Shan Huang, Jianyuan Zhong, Zhengyuan Shi, Guohao Dai, Ningyi Xu, and Qiang Xu. DeepGate4: Efficient and effective representation learning for circuit design at scale. *arXiv preprint arXiv:2502.01681*, 2025.

[18] Zhengyuan Shi, Min Li, Yi Liu, Sadaf Khan, Junhua Huang, Hui-Ling Zhen, Mingxuan Yuan, and Qiang Xu. SATformer: Transformer-based unsat core learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–4. IEEE, 2023.

[19] Wenjie Zhang, Zeyu Sun, Qihao Zhu, Ge Li, Shaowei Cai, Yingfei Xiong, and Lu Zhang. NLocalSAT: Boosting local search with solution prediction. *arXiv preprint arXiv:2001.09398*, 2020.

[20] Wenxi Wang, Yang Hu, Mohit Tiwari, Sarfraz Khurshid, Kenneth McMillan, and Risto Miikkulainen. NeuroComb: Improving SAT solving with graph neural networks. *arXiv preprint arXiv:2110.14053*, 2021. URL `https://arxiv.org/abs/2110.14053`.

[21] Separate DECISION Queue. CaDiCaL at the SAT Race 2019. *SAT RACE*, 2019:8, 2019.

[22] Stephen A. Cook. The complexity of theorem-proving procedures. In *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, pages 143–152, 2023.

[23] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[25] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. *Advances in Neural Information Processing Systems*, 27, 2014.

[26] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

[27] Niklas Sorensson and Niklas Een. MiniSAT v1.13: A SAT solver with conflict-clause minimization. *SAT*, 53(2005):1–2, 2005.

[28] Gilles Audemard and Laurent Simon. On the Glucose SAT solver. *International Journal on Artificial Intelligence Tools*, 27(01):1840001, 2018.

[29] Tsz Ho Chan, Wenyi Xiao, Junhua Huang, Huiling Zhen, Guangji Tian, and Mingxuan Yuan. IB-Net: Initial branch network for variable decision in boolean satisfiability. *arXiv preprint arXiv:2403.03517*, 2024.

[30] Hongduo Liu, Peng Xu, Yuan Pu, Lihao Yin, Hui-Ling Zhen, Mingxuan Yuan, Tsung-Yi Ho, and Bei Yu. NeuroSelect: Learning to select clauses in SAT solvers. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.

[31] Zhaoyu Li, Jinpei Guo, and Xujie Si. G4SATBench: Benchmarking and advancing SAT solving with graph neural networks, 2023. URL `https://openreview.net/forum?id=8eVgdwKs2N`.

[32] Min Li, Zhengyuan Shi, Qiuxia Lai, Sadaf Khan, Shaowei Cai, and Qiang Xu. DeepSAT: An EDA-driven learning framework for SAT. *arXiv preprint arXiv:2205.13745*, 2022.

[33] Daniel Selsam and Nikolaj Bjørner. Guiding high-performance SAT solvers with unsat-core predictions. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 336–353. Springer, 2019.

[34] Vitaly Kurin, Saad Godil, Shimon Whiteson, and Bryan Catanzaro. Can Q-learning with graph networks learn a generalizable branching heuristic for a SAT solver? *Advances in Neural Information Processing Systems*, 33:9608–9621, 2020.

[35] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[36] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[37] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020.

[38] Le Wu, Peijie Sun, Richang Hong, Yanjie Fu, Xiting Wang, and Meng Wang. SocialGCN: An efficient graph convolutional network based model for social recommendation. *arXiv preprint arXiv:1811.02815*, 2018.

[39] Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*, pages 499–508. ACM, 2018. doi: 10.1145/3178876.3186116.

[40] Zhengyuan Shi, Ziyang Zheng, Sadaf Khan, Jianyuan Zhong, Min Li, and Qiang Xu. DeepGate3: Towards scalable circuit representation learning. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2024.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[42] Yongjing Yin, Junran Ding, Kai Song, and Yue Zhang. SemFormer: Transformer language models with semantic planning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18669–18680, 2024.

[43] Ali Hatamizadeh and Jan Kautz. MambaVision: A hybrid mamba-transformer vision backbone. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 25261–25270, 2025.

[44] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34: 21618–21629, 2021.

[45] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[46] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[47] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[48] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

[49] Md Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Triplet interaction improves graph transformers: accurate molecular graph learning with triplet graph transformers. *arXiv preprint arXiv:2402.04538*, 2024.

[50] Xiaohui Chen, Yinkai Wang, Jiaxing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Li-Ping Liu. Graph generative pre-trained transformer. *arXiv preprint arXiv:2501.01073*, 2025.

[51] Yisong Huang, Jin Li, Xinlong Chen, and Yang-Geng Fu. Training graph transformers via curriculum-enhanced attention distillation. In *The Twelfth International Conference on Learning Representations*, 2023.

[52] Jinsong Chen, Hanpeng Liu, John E. Hopcroft, and Kun He. Leveraging contrastive learning for enhanced node representations in tokenized graph transformers. *Advances in Neural Information Processing Systems*, 37:85824–85845, 2024.

[53] Armin Biere, Nils Froleyks, and Wenxi Wang. CadiBack: Extracting Backbones with CaDiCaL. In *26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023)*, pages 3:1–3:12, 2023.

[54] Jesús Giráldez-Cru and Jordi Levy. A modularity-based random SAT instances generator. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1952–1958. AAAI Press, 2015.

[55] Massimo Lauria, Jan Elffers, Jakob Nordström, and Marc Vinyals. CNFgen: A generator of crafted benchmarks. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 464–473. Springer, 2017.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[58] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Appendix

A Related work

A.1 Deep learning for SAT solving

Learning-based SAT solvers are generally divided into two categories, standalone neural solvers and learning-aided solvers. The former relies entirely on end-to-end deep learning models to predict satisfiability or generate variable assignments. For example, NeuroSAT [9] models SAT problems as a binary graph classification task and trains a neural network to predict the satisfiability of formulas. DeepSAT [32] represents SAT solutions as the product of conditional Bernoulli distributions and obtains variable assignments by maximizing the joint probability. However, compared with modern non-learning SAT solvers, the performance of such purely learning-based approaches still remains substantially inferior. The reason is that deep learning models rely on probability statistics for inference, while SAT problems are essentially strict deterministic logical reasoning tasks, creating a natural mismatch between the two.

In contrast, learning-aided approaches that integrate deep learning models with modern SAT solvers are widely regarded as a more promising research direction. Modern SAT solvers typically rely on heuristic strategies to guide the search process. Therefore, enhancing these heuristics with deep learning has become a key direction for improving solver performance. For example, NeuroCore [33] and NeuroComb [20] periodically predict UNSAT Cores to guide the solving process, significantly improving overall efficiency. Graph-Q-SAT [34] replaces the branching heuristic with a reinforcement learning agent, enabling the solver to make better decisions in the early stages of search, thus reducing the number of iterations required for SAT solving by approximately 2-3 times. NLocalSAT [19] and NeuroBack [16] both employ GNN to predict variable phases in CNF formulas, thereby generating high-quality initial assignments for the SAT solver, significantly reducing solver running time. Although existing learning-aided methods can improve the heuristic effect of solvers, they still face the problems of insufficient expressive power and weak scalability, particularly on industrial and large-scale instances.

A.2 Message passing graph neural networks

Message passing graph neural networks have become one of the most dominant approaches in graph representation learning in recent years. An early and pioneering representative is the graph convolutional network [35], whose core idea is to learn node representations by performing neighborhood aggregation over graph structures. Subsequently, [36] further unified earlier GNN models under the message passing framework, which has become the prevailing paradigm in current GNN research and applications, and has driven remarkable success in a wide range of graph learning tasks [37, 38]. In addition, subsequent studies have further explored the joint modeling of local and global structural information to enhance graph representations, as exemplified by Dual Graph Convolutional Networks [39].

In the SAT domain, existing works typically construct a CNF formula as an undirected bipartite graph and apply GNN to learn representations of variable nodes and clause nodes, using the resulting embedding for tasks such as satisfying assignment prediction [15, 19, 16], satisfiability prediction [9, 32], and branching variable selection [34, 18]. Existing GNN methods for SAT generally adopt a message passing framework, updating node representations by aggregating topological neighbor features at each layer and relying on multiple rounds of message passing to encode graph structure information [17, 15, 16, 40]. However, the graph structures corresponding to SAT instances often contain both local constraints and long-range dependencies between variables and clauses. Therefore, when the exploitable structural information is restricted to local neighborhoods while the global graph structure is ignored, the limited structural information can lead to performance degradation. In addition, multiple rounds of message passing also incur substantial computational and memory costs on large-scale instances.

A.3 Graph transformer

Transformer [41] is one of the most representative architectures for sequence modeling. Its self-attention mechanism has a natural advantage in modeling long-range dependencies, and has achieved remarkable success in fields such as natural language processing [42] and computer vision [43]. In recent years, researchers have begun extending transformers to graph-structured data [44] and combining them with GNN, proposing a series of graph transformer models [45, 46]. By leveraging the strength of self-attention in capturing long-range dependencies, these models alleviate the limitations of message passing GNNs in modeling distant relationships, such as oversmoothing [47, 48] and

over-squashing [44]. Graph transformers introduce attention into the node embedding update process while explicitly incorporating structural information, and they have outperformed traditional message passing GNNs on tasks such as molecular property prediction [49], graph generation [50], and node classification [51, 52].

In the SAT domain, NeuroBack is the first to introduce graph transformers into SAT solving. It represents a SAT instance as a VCG and applies a graph transformer to aggregate information across nodes. Meanwhile, the global self-attention mechanism in transformers leads to quadratic memory complexity with respect to the number of nodes, making it difficult to scale to large-scale graphs. NeuroBack replaces global self-attention by combining graph self-attention and local self-attention. However, training NeuroBack to predict backbone variables relies heavily on large-scale and labeled datasets such as DataBack [16], which are costly to construct. Since backbone labels must be generated through long offline runs of backbone extractors such as CadiBack [53], the data preparation process incurs substantial overhead, thereby limiting the practical deployment of the approach.

B Unsupervised loss function derivation

B.1 Problem definition and notation

We consider the SAT problem under the CNF formula representation. Let the set of Boolean variables be:

$$X = \{x_1, \ldots, x_n \mid x_i \in \{0, 1\}\}, \tag{22}$$

where n is the number of variables, and $x_i = 1$ and $x_i = 0$ indicate that variable x_i takes true and false, respectively. Let the clause set be:

$$C = \{c_1, \ldots, c_m\}, \tag{23}$$

where m is the number of clauses. Each clause c_j is the disjunction (OR) of a set of literals:

$$c_j = \bigvee_{\ell \in L_j} \ell, \tag{24}$$

where L_j denotes the set of literals in clause c_j. A literal ℓ is either x_i or $\neg x_i$, i.e., $\ell \in \{x_i, \neg x_i \mid i = 1, \ldots, n\}$. A CNF formula is the conjunction (AND) of all clauses:

$$\phi = \bigwedge_{j=1}^{m} c_j. \tag{25}$$

Given an assignment $\tau = \{\tau_1, \ldots, \tau_n \mid \tau_i \in \{0, 1\},\ i = 1, \ldots, n\}$, if $\phi(\tau) = 1$ (i.e., all clauses evaluate to true), then τ is said to satisfy the formula ϕ. The model outputs the probability of each variable being assigned to 1:

$$p_i = \mathbb{P}(x_i = 1) \in [0, 1]. \tag{26}$$

Then the probability that a literal is assigned to true is:

$$p(\ell = 1) = \begin{cases} p_i, & \ell = x_i, \\ 1 - p_i, & \ell = \neg x_i. \end{cases} \tag{27}$$

B.2 Unsupervised SAT constraint loss

Clause satisfaction probability. For a clause c_j, the clause is false if and only if all literals in the clause are false. Under the assumption that the variables are mutually independent, the probability that the clause is false is:

$$\mathbb{P}(c_j = 0) = \prod_{\ell \in L_j} \mathbb{P}(\ell = 0) = \prod_{\ell \in L_j} \big(1 - \mathbb{P}(\ell = 1)\big). \tag{28}$$

Therefore, the probability that the clause is true is:

$$\mathbb{P}(c_j = 1) = 1 - \mathbb{P}(c_j = 0) = 1 - \prod_{\ell \in L_j} \big(1 - \mathbb{P}(\ell = 1)\big). \tag{29}$$

To improve numerical stability, we move the product computation into the log space:

$$s_{c_j} = \sum_{\ell \in L_j} \log\big(1 - \mathbb{P}(\ell = 1)\big). \tag{30}$$

Then,

$$\mathbb{P}(c_j = 1) = 1 - \exp\big(s_{c_j}\big). \tag{31}$$

CNF satisfaction probability. Given a CNF formula $\phi = \bigwedge_{j=1}^{m} c_j$, it is true if and only if all clauses are true. Under the assumption that the clauses are mutually independent, the probability that

the CNF formula is true is:

$$\mathbb{P}(\phi = 1) = \prod_{j=1}^{m} \mathbb{P}(c_j = 1). \tag{32}$$

To improve numerical stability, we move the product computation into the log space:

$$L_{\text{unsup}} = -\log \mathbb{P}(\phi = 1) = -\sum_{j=1}^{m} \log \mathbb{P}(c_j = 1) = -\sum_{j=1}^{m} \log\big(1 - \exp(s_{c_j})\big), \tag{33}$$

where $s_{c_j} = \sum_{\ell \in L_j} \log\big(1 - \mathbb{P}(\ell = 1)\big)$.

Therefore, minimizing L_{unsup} encourages the model to output assignments that satisfy the formula ϕ as much as possible, i.e., $\mathbb{P}(\phi = 1)$ is encouraged to approach 1, thereby driving the model to generate assignments closer to satisfiable solutions.

C Combination with SAT solver

C.1 CDCL workflow

Most modern SAT solvers [7, 21, 27, 28] are built on the CDCL framework. Specifically, a CDCL SAT solver first selects a branching variable and then determines its phase using the phase selection heuristic, after which it performs BCP based on the decision. If a conflict occurs during propagation, this indicates that the current partial assignment is unsatisfiable, and the solver enters the conflict analysis stage. During conflict analysis, CDCL learns and adds a new clause that represents the cause of the conflict, thereby helping prevent the solver from repeating the same mistake in subsequent search steps. If the conflict occurs at the root level and remains unresolved, the formula is declared unsatisfiable. After conflict analysis, CDCL backjumps to the earliest decision level that can resolve the conflict and continues the search. This process iterates until all variables are assigned (SAT) or the problem is established as unsatisfiable (UNSAT).

In this process, the phase selection heuristic is a key strategy in CDCL that is directly related to the direction of variable assignments. It can significantly influence the subsequent propagation paths and the distribution of conflicts, thereby influencing the overall solving efficiency.

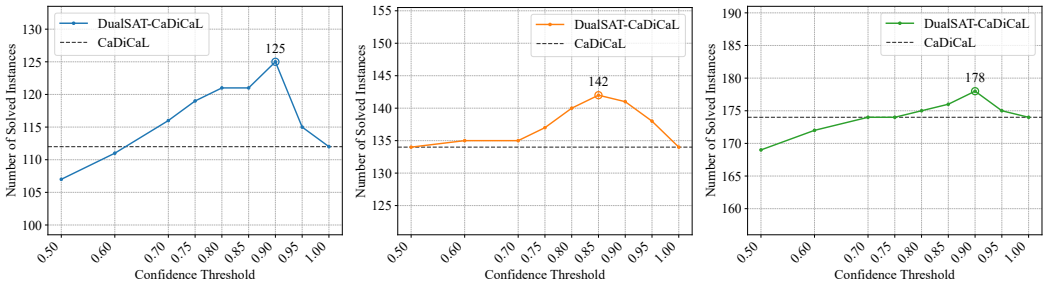
C.2 SAT solver with DualSAT

We integrate DualSAT with the classic SAT solver CaDiCaL, using the phase predictions generated by DualSAT to enhance the phase selection heuristic of CaDiCaL and thereby improve solving performance. Specifically, before invoking the solver, DualSAT performs a single inference pass on a given SAT instance, outputting the probability prediction that each variable takes the value 1. To avoid over-relying on the predictions of the neural network model, we adopt a confidence-thresholding strategy, under which only high-confidence predictions are used to initialize variable phases, while the remaining variables retain the default initial assignments of CaDiCaL. Let $\alpha \in (0.5, 1)$ denote the confidence threshold. If the predicted probability exceeds α, the variable is initialized to the positive phase. If it is below $1 - \alpha$, the variable is initialized to the negative phase. Otherwise, the default initial assignment of CaDiCaL is preserved. In this way, the model provides the solver with only a high-confidence partial initial phase assignment, allowing the search to be guided by reliable predictions while reducing the impact of low-confidence noise, and thus improving the overall robustness of solving. In this work, we set $\alpha = 0.9$, with its sensitivity analysis provided in Appendix E.1. The resulting phase predictions are then used as the initial assignment of CaDiCaL, after which the solver proceeds according to its default workflow.

It is known that using a neural network to predict the initial variable phases is the most direct and general way to integrate neural network models into CDCL SAT solvers. Therefore, DualSAT can be incorporated into this framework as a plug-in without modifying the internal mechanisms of the solver.

D Experimental setting

Dataset preparation. Following the experimental settings adopted in recent studies [14, 20, 15, 19, 16], we evaluate DualSAT on multiple SAT tasks and use four datasets from three domains, covering random instances, pseudo-industrial instances, and combinatorial problems. Specifically, random CNF formulas are generated by the SR generator [9]. Pseudo-industrial instances are generated by the CA generator [54], whose structure resembles that of real industrial SAT problems. And combinatorial problems are generated using CNFgen [55] by converting the k-Clique and k-Domset



(a) SATCOMP-2022. (b) SATCOMP-2023. (c) SATCOMP-2024.

Figure 5: Effect of the confidence threshold α on the number of solved instances across SATCOMP 2022–2024.

problems into SAT formulas. For finer-grained analysis, we partition each dataset into three difficulty levels, easy, medium, and hard, according to the difficulty of solving, where hard corresponds to instances with larger scale and more complex structure. Taking the SR dataset as an example, easy-SR contains 10–40 variables, medium-SR contains 40–200 variables, and hard-SR contains 200–400 variables. For each easy and medium dataset, we generate 8,000 satisfiable instances for training, 1,000 for validation, and 1,000 for testing. For each hard dataset, we generate 1,000 test instances to evaluate the generalization performance of the model on larger scale instances. Meanwhile, we control the generator parameters to avoid producing trivial instances. Ground truth assignment labels for all instances are obtained using the modern SAT solver CaDiCaL.

Implementation details. When constructing the global graph structure, we set the random walk length to $T = 30$, the context window size to $w = 10$, and the number of random walks per node to $K = 30$. In DualSAT, we use ReLU activation and layer normalization, set the number of heads in the multi-head self-attention mechanism to 8, and the feature dimensions of both the structural embedding H_{struct} and the semantic embedding H_{sem} to 64. We train the model for 200 epochs with a batch size of 64 on a single RTX 5090 GPU. The first stage lasts 50 epochs with joint training using both the supervised and unsupervised losses, after which we continue training using only the unsupervised loss. We use Adam [56] as the optimizer with a learning rate of 10^{-3} and a weight decay of 10^{-10}. All neural networks in this work are implemented using PyTorch [57] and PyTorch Geometric [58].

E Additional experiment

E.1 Sensitivity to confidence threshold

We evaluate the sensitivity of the solving performance of DualSAT-CaDiCaL to the confidence threshold α on the SATCOMP 2022–2024 datasets, with the results shown in Figure 5.

As α increases, the solving performance of DualSAT-CaDiCaL generally exhibits a non-monotonic trend. When α is too low, more low-confidence variable phase predictions are used for initialization, which may misguide the early search process of CaDiCaL. Conversely, when α is too high, too few phase predictions are retained to effectively guide the search, causing performance to gradually approach that of the original CaDiCaL. Although the optimal threshold varies slightly across datasets, the overall results indicate that a moderately high threshold provides a favorable balance between filtering out low-confidence predictions and retaining effective search guidance. Therefore, considering the overall performance across the evaluated datasets, we set the confidence threshold to $\alpha = 0.9$.

E.2 Implementation details and additional results for combination with SAT solver

Implementation details. In this subsection, we integrate DualSAT and NeuroBack into CaDiCaL denoted as DualSAT-CaDiCaL and NeuroBack-CaDiCaL, respectively, and compare DualSAT-CaDiCaL with NeuroBack-CaDiCaL and the baseline solver CaDiCaL to evaluate its effectiveness. We use all 1,200 CNF formulas from the main track of SATCOMP 2022–2024 as the test dataset. For inference, both DualSAT-CaDiCaL and NeuroBack-CaDiCaL employ the model trained on the medium-SR dataset. All solvers are run on a machine equipped with a 32-core CPU using 32 parallel processes, with a per-instance time limit of 5,000 seconds. To ensure a fair comparison, we report the end-to-end runtime of DualSAT-CaDiCaL and NeuroBack-CaDiCaL, including the overhead of model inference. Due to GPU memory constraints, inference for both DualSAT and NeuroBack is performed exclusively on the CPU. In addition, the memory limit for each process is set to 10

Table 3: Average number of conflicts for CaDiCaL and DualSAT-CaDiCaL on SATCOMP 2022–2024.

Dataset	CaDiCaL	DualSAT-CaDiCaL	Reduction (%)
SATCOMP 2022	22,287,495	20,669,510	7.26
SATCOMP 2023	28,745,032	26,384,327	8.21
SATCOMP 2024	20,949,397	19,019,055	9.21

GB. Since SATCOMP instances are typically very large, often containing millions of variables and clauses, the computational overhead increases significantly with graph size. Therefore, to satisfy the 10 GB memory constraint, we retain only those instances with fewer than 2×10^5 nodes on which inference can be completed. Consequently, in the SATCOMP 2022–2024 datasets, 149, 214, and 224 instances are ultimately included in the experiments, respectively.

Conflicts during SAT solving. We further evaluate solver performance by analyzing the average number of conflicts generated during the SAT solving process. The number of conflicts is generally positively correlated with the frequency of conflict analysis and backtracking. As shown in Table 3, compared with the original CaDiCaL, DualSAT-CaDiCaL incurs fewer conflicts on average when successfully solving satisfiable instances. This indicates that DualSAT-CaDiCaL reaches satisfying assignments more efficiently and at a lower conflict cost, which in turn allows it to solve more instances within the same time budget. Notably, DualSAT-CaDiCaL not only outperforms CaDiCaL in terms of the number of solved instances but also requires fewer conflicts to find satisfying assignments. Further demonstrating that the high-quality initialization phase assignments provided by DualSAT can effectively assist the CDCL SAT solver.

F Limitation and future work

Limitation. Despite DualSAT demonstrating strong empirical performance, it still has some limitations. DualSAT is specifically designed for satisfiable instances and is therefore not directly applicable to unsatisfiable instances. In addition, as discussed in Appendix E.2, DualSAT remains constrained by memory consumption as the instance scale continues to grow.

Future work. Future work may proceed in two directions. First, incremental models may offer a practical way to overcome current memory and GPU memory limitations, thereby enabling the solver to handle larger scale SAT instances. Second, it would be worthwhile to explore the application of DualSAT to other decision-making components in CDCL SAT solvers, such as variable selection and learned clauses management.

G Broader impact

Research on Boolean satisfiability (SAT) has broad applications across computer science and related fields, including hardware and software verification, automated planning, scheduling, and artificial intelligence. Improvements in SAT solving techniques can enhance the efficiency and scalability of these applications.

At the same time, as a general-purpose computational tool, SAT technology could be applied in domains with dual-use implications, such as cryptanalysis or security-related tasks. While our work focuses on algorithmic improvements without targeting any specific sensitive application, we encourage responsible use of these techniques in accordance with ethical guidelines and relevant regulations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We confirm that the main claims made in the abstract and introduction accurately reflect our contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix F

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We confirm that the paper provides the full set of assumptions and a complete (and correct) proof for each theoretical result, see Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We confirm that we have provided a detailed explanation of our experimental setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Codes of this work are in need of further polishing, and will be released if this paper is accepted.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See the implementation details in Sec. 4.1 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our experiments are conducted on multiple datasets, providing comprehensive quantitative results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided sufficient information on the computer resources in Sec. 4.1 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that our research meets the requirements.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix G.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The code and datasets used in this work have been properly attributed in accordance with the original authors' requirements and comply with the corresponding licenses and terms of use. See the experimental settings and the implementation details in Sec. 4.1 and Appendix D.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs did not contribute to the core methodology, experimental design, or scientific conclusions of this work. They do not constitute a component of the research methods and no further disclosure is required.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.