
On the fundamentals of gradient-based SAT solvers

Nikolaos Karalias*

Saku Peltonen*

Nikita Kostin*

Stefanie Jegelka

Abstract

Gradient-based methods have received increasing attention for Boolean satisfiability, both in neural-network-based solvers and broadly as continuous local search techniques. In this work, we provide a deeper understanding of the interplay of optimization process, problem structure, common formula transformations and neural network properties. Our unified view treats both direct gradient descent over variable marginals and neural-parametrized gradient descent as instances of the same continuous SAT framework. We show that its objective, the expected number of unsatisfied clauses under independent variable marginals, equals (up to constant shift) the clausewise Fourier objective commonly used in continuous SAT methods, and characterize its relation to the exact product-measure probability of unsatisfiability. Then, we derive convergence conditions for gradient descent. A central theme is symmetry: gradient descent can become trapped in invariant subspaces induced by formula symmetries. To remedy this, we show that formula transformations commonly used as SAT preprocessing tools can be repurposed to modulate the optimization landscape. We analyze their effect on problem structure, smoothness, tightness, and expressivity barriers of the neural network. Empirically, across several benchmarks, these transformations can improve gradient-descent performance. Our experiments also indicate that transformations can improve data efficiency and inference-time performance in self-supervised neural SAT solving. Finally, we demonstrate that neural-parametrized gradient descent can improve over input-space gradient descent across several datasets.

1 Introduction

Modern advances in machine learning and differentiable programming have drawn increased attention to gradient-based methods for hard discrete optimization problems. Boolean satisfiability (SAT) is a canonical example in this category as a problem of central importance in computer science [25]. Its broad scope has produced a rich tapestry of insights from a wide range of perspectives and fields. This includes the study of symmetries and symmetry-breaking predicates for solvers [13, 51], community structure in industrial instances [42, 7], phase transitions and solution-space geometry [12, 35, 2, 28], continuous relaxations and semidefinite programming approaches [18, 31], heavy-tailed runtime behavior and restart strategies [19, 62], and probabilistic techniques for certifying satisfiability [38, 1, 17]. The dominant approaches to SAT are still highly specialized discrete algorithms, including conflict-driven clause-learning solvers and stochastic local search methods [8, 49, 16]. Their success depends strongly on the representation and structure of the input formula, including preprocessing, learned clauses, locality, and symmetry breaking.

At the same time, a broad landscape of diverse methods for gradient-based techniques has emerged. It includes continuous and Fourier/Walsh-Fourier formulations for SAT and hybrid Boolean constraints [29, 30, 47, 65, 11], differentiable MaxSAT layers and solvers [59, 34, 20, 36], learning-enhanced hybrid solvers and CDCL guidance [60, 54], and neural SAT architectures trained through self-supervised objectives [43, 32, 64]. Despite these developments, there is a lack of overall theoretical direction and understanding of fundamental design principles. Our goal in this work is to

*Equal contribution.

establish core insights that can guide the development of more successful gradient-based methods. Due to the scope of the problem, establishing insights that carry over to all the aforementioned settings is a nearly impossible undertaking. Consequently, we focus on a simple but general problem formulation: minimizing a multilinear polynomial over the hypercube; this objective represents the expected number of clause violations under a product distribution of independent variables. This minimization is performed with gradient descent, either directly over the variables, or through the parameters of a model. This view captures important parts of Fourier-expansion-based SAT methods and of many self-supervised neural combinatorial optimization methods [22, 24, 23, 43, 61], physics-inspired neural combinatorial optimization [48, 27], and dataless neural networks [3, 21, 57].

Many key factors that affect the continuous problem mirror those in the discrete setting: symmetry obstructions, initializations, the structure of the formula, and the dynamics of the search algorithm (gradient descent in this case). Our goal is to provide a unifying perspective through a comprehensive set of theoretical results that cover all those aspects. Our contributions are as follows.

- We relate our objective to the Fourier expansion objective used in continuous SAT methods. We show that it upper-bounds the exact product-measure probability of unsatisfiability, characterize the tightness of this bound and discuss its effect on optimization.
- We give sufficient conditions under which projected gradient descent converges to zero expected-unsat objective on satisfiable instances. These conditions indicate the role of initialization and yield explicit convergence certificates for certain formula families.
- We analyze symmetry obstructions and show that gradient descent can be trapped in invariant subspaces induced by formula symmetries, and that 1-WL-style message-passing neural parametrizations impose related restrictions through the architecture.
- We propose to use formula transformations to alter the optimization landscape and show how they affect the smoothness of the objective, bound tightness, product-measure probabilities, and symmetry structure. We also show how transformations can break symmetry barriers.
- Empirically, we validate these mechanisms on specific symmetric families of instances and on a wider range of benchmark instances. The experiments show that our convergence certificates can predict projected-gradient success and that formula transformations can improve gradient-descent performance. To investigate the effects of parametrization, we also compare neural-parametrized gradient descent with direct input-space optimization. Finally, we show that transformations can improve data efficiency and inference-time performance in self-supervised neural SAT solving.

2 Preliminaries

Definitions and notation. Throughout the paper, a CNF instance is denoted by $T = (V, \Phi)$, where $V = \{v_1, \dots, v_n\}$ is the set of Boolean variables and $\Phi = C_1 \wedge \dots \wedge C_m$ is a conjunction of clauses. A literal is either a variable v_i or its negation $\neg v_i$, and a clause is a disjunction of literals. For each clause C_j , let $N^+(j)$ be the set of indices appearing positively in C_j , let $N^-(j)$ be the set of indices appearing negated in C_j , and let $A(j) = N^+(j) \cup N^-(j)$, $a(C_j) = |A(j)|$. We use $\mathbf{x} \in \{0, 1\}^n$ for a Boolean assignment, with $x_i = 1$ denoting $v_i = \text{True}$ and $x_i = 0$ denoting $v_i = \text{False}$. We use $p \in [0, 1]^n$ for Bernoulli marginals, and μ_p for the product measure with independent coordinates satisfying $\Pr_{X \sim \mu_p}[X_i = 1] = p_i$. An assignment satisfies Φ if $\Phi(\mathbf{x}) = 1$. The feasible set of satisfying assignments is $\mathcal{C} = \{\mathbf{x} \in \{0, 1\}^n : \Phi(\mathbf{x}) = 1\}$. The SAT decision problem asks whether $\mathcal{C}$ is nonempty; algorithmically, one usually seeks an assignment in $\mathcal{C}$ when such an assignment exists. For ease of exposition, we focus our attention on CNF SAT, although many of the results below do not depend on this particular encoding. We also define the signed typed variable-clause graph G_T . Its nodes are the variables v_i and the clauses C_j , with node types distinguishing variables from clauses. There is an edge between v_i and C_j whenever $i \in A(j)$; the edge label is $+$ if $i \in N^+(j)$ and $-$ if $i \in N^-(j)$. Here, “typed” means that color refinement and message passing respect the variable/clause node types, in addition to the signed edge labels.

The standard optimization variant of the problem is MAX-SAT, which asks for an assignment satisfying as many clauses of a CNF formula as possible. For convenience, we work with the equivalent minimization variant, MinUNSAT, which counts violated clauses: for a CNF instance $T = (V, \Phi)$, define $V_T(\mathbf{x}) = \sum_{j=1}^m \mathbf{1}[C_j(\mathbf{x}) = 0]$. MinUNSAT asks for an assignment minimizing $V_T(\mathbf{x})$ over $\{0, 1\}^n$. A CNF formula is satisfiable when the minimum possible value of V_T is zero.

Continuous optimization setup. In the continuous optimization perspective used throughout the paper, we associate one marginal with each Boolean variable. These marginals are determined by real-valued optimization variables. Abstractly, let $z \in \mathcal{Z} \subseteq \mathbb{R}^k$ denote these continuous variables, and let $\pi : \mathcal{Z} \rightarrow [0, 1]^n$ map them to Bernoulli marginals $p = \pi(z)$. For direct marginal optimization one can take $\mathcal{Z} = [0, 1]^n$ and $\pi(z) = z$; for an unconstrained parametrization one may take $\mathcal{Z} = \mathbb{R}^n$ and $\pi_i(z) = \sigma(z_i)$. For neural parametrizations, z are the parameters of the neural network that outputs p . A continuous SAT formulation is then a differentiable surrogate $\mathcal{L}_T : [0, 1]^k \rightarrow \mathbb{R}$, viewed as a real-valued extension of a discrete SAT or MinUNSAT objective. Under a parametrization π , the objective optimized in the variables z is $F_T(z) := \mathcal{L}_T(\pi(z))$. Gradient descent updates the continuous variables by

$$z^{t+1} = z^t - \eta \nabla F_T(z^t),$$

possibly followed by projection when $\mathcal{Z}$ is constrained. The resulting marginals $p^t = \pi(z^t)$ can later be decoded into a Boolean assignment.

3 Gradient descent on multilinear polynomials for SAT

This section studies the continuous expected-unsat objective from three complementary angles: its probabilistic meaning, convergence conditions for gradient descent, and symmetry-induced invariant subspaces. Unless otherwise noted, proofs of main-text results are collected in Appendix A.

To obtain a continuous objective, we follow a common construction for continuous extensions [15, 24]. We treat a point $p \in [0, 1]^n$ as the marginals of a product measure μ_p over assignments and take the expectation of the discrete MinUNSAT objective V_T . For a clause C_j , the violation probability under μ_p is the monomial $q_j(p) := \Pr_{X \sim \mu_p}[C_j(X) = 0] = \prod_{i \in N^+(j)} (1 - p_i) \prod_{i \in N^-(j)} p_i$. This gives the expected-unsat objective:

$$\mathcal{L}_T(p) \triangleq \mathbb{E}_{X \sim \mu_p}[V_T(X)] = \sum_{j=1}^m q_j(p). \quad (1)$$

Several works in the literature on continuous SAT and CSP optimization use Walsh-Fourier expansions of Boolean constraints [29, 30, 11, 10]. A key observation is that, for CNF-SAT, the clausewise Fourier expansion often used in those works is essentially equivalent to Eq. (1). To see this, let $s \in [-1, 1]^n$ where $s_i = -1$ corresponds to Boolean true and $s_i = +1$ to Boolean false, and set $p_i = (1 - s_i)/2$. For a clause C , write $\psi_C(s)$ for its clausewise Fourier contribution.

Proposition 3.1 (Clausewise Fourier expansions and expected clause violations). *For a clause C , under the change of variables $p_i = (1 - s_i)/2$, the clausewise Fourier term satisfies $\psi_C(s) = 2q_C(p) - 1$, where $q_C(p) := \Pr_{X \sim \mu_p}[C(X) = 0]$.*

Summing over clauses gives $\sum_{C \in \Phi} \psi_C(s) = 2\mathcal{L}_T(p) - m$. This is the violated-clause form of the randomized-rounding identity in FourierSAT [29, Theorem 3], and it follows directly from our definition of $\mathcal{L}_T$ as the expected number of violated clauses. Hence, our results straightforwardly extend to those objectives. Additionally, $\mathcal{L}_T$ is the *multilinear extension* of V_T . It is a continuous proxy for unsatisfiability because its discrete minimizers correspond to assignments that minimize the number of violated clauses. Standard conditional-expectation rounding can decode any marginal vector p to a Boolean assignment whose violated-clause count is no worse than $\mathcal{L}_T(p)$ [4, 45].

The expectation perspective highlights connections with self-supervised machine-learning approaches to combinatorial optimization that use expectations or continuous extensions as smooth training objectives for neural networks [22, 58, 24, 9, 46, 40, 26, 39, 23]. We return to this parametrized viewpoint below.

3.1 Properties of the objective

As a multilinear polynomial, the objective (1) enjoys nice smoothness properties.

Proposition 3.2 (Global smoothness of the expected violated-clause objective). *Fix a CNF instance $T = (V, \Phi)$ with $\Phi = C_1 \wedge \dots \wedge C_m$. Let $w := \max_j a(C_j)$ be the maximum clause width and $\Delta := \max_i |\{j : i \in A(j)\}|$ the maximum variable degree. Then $\mathcal{L}_T$ is globally $(w - 1)\Delta$ -smooth on $[0, 1]^n$ with respect to the Euclidean norm, and in particular*

$$\|\nabla^2 \mathcal{L}_T(p)\|_2 \leq (w - 1)\Delta, \text{ for all } p \in [0, 1]^n. \quad (2)$$

L-smoothness is a standard regularity condition in first-order optimization [41]. Note the dependence on the of the bound on the structure of the formula.

Next, we focus on the probabilistic character of the objective.

Proposition 3.3 (The expected violated-clause objective upper-bounds the exact unsat probability). *Let $p \in [0, 1]^n$, and let $X \sim \mu_p$. Then $\Pr[\Phi(X) = 0] \leq \mathcal{L}_T(p)$.*

Thus, we can see that $\mathcal{L}_T$ is also a valid union bound for the true unsatisfiability probability. This raises the question of tightness: When is the union bound objective exactly equal to the product-measure probability that Φ is unsatisfied? Proposition 3.4 gives the characterization.

Proposition 3.4 (Tightness of the union-bound objective). *Let $T = (V, \Phi)$ be a CNF instance with non-tautological clauses, and let $P_\Phi(p) := \Pr_{X \sim \mu_p}[\Phi(X) = 0]$ be the exact product-measure unsat probability. Then $P_\Phi \equiv \mathcal{L}_T$ as multilinear polynomials if and only if Φ is a hitting CNF, meaning that every two distinct clauses contain some variable with opposite polarities.*

Intuitively, $\mathcal{L}_T$ counts every clause that an assignment violates, whereas P_Φ only records whether at least one clause is violated. The gap comes from assignments that violate several clauses and are therefore counted multiple times by $\mathcal{L}_T$. The bound is tight exactly when the formula structure rules this out, so every assignment violates at most one clause. Unlike $\mathcal{L}_T$, P_Φ is generally hard to optimize directly because exact evaluation under the uniform product measure recovers model counting and is therefore $\#P$ -hard [56]. A natural question is whether tightening the bound could yield optimization benefits. The answer depends on the instance. On satisfiable formulas, P_Φ can give a less biased signal because it does not over-penalize assignments that violate several clauses. On unsatisfiable formulas, however, $P_\Phi \equiv 1$, so it passes no informative gradient. In that setting, tightening the bound can weaken useful signal for reducing the number of violated clauses. Appendix A.4.1 gives a toy example illustrating both effects.

3.2 Convergence conditions for gradient descent

Beyond the probabilistic properties of the objective, a central question is when gradient dynamics can converge to a zero-objective basin for a satisfiable formula. A first step is to identify conditions under which gradient descent commits to a monotone trajectory toward a satisfying assignment. The following result encodes this idea through a simple directional condition. Fix a satisfying assignment $y \in \{0, 1\}^n$ and an initialization p^0 . Define

$$D_i^y(p) := -(2y_i - 1) \frac{\partial \mathcal{L}_T}{\partial p_i}(p).$$

Say that p lies between p^0 and y on $S \subseteq [n]$ if, for each $i \in S$, p_i lies between p_i^0 and y_i ; equivalently, $p_i \geq p_i^0$ if $y_i = 1$, and $p_i \leq p_i^0$ if $y_i = 0$.

Theorem 3.5 (Informal: certified basin theorem). *Suppose every clause C_j has a designated literal that is true under y , with selected variable $r_j \in S$. Assume that throughout the box between p^0 and y on S , $D_i^y(p) \geq 0$ for every $i \in S$, and whenever $q_j(p) > 0$, $D_{r_j}^y(p) > 0$. Then, for every step size $\eta > 0$, projected gradient descent on $\mathcal{L}_T$ initialized at p^0 satisfies $\mathcal{L}_T(p^t) \rightarrow 0$.*

The formal sign-vector version is Theorem A.1 in Appendix A. In short, along the box from p^0 to a satisfying assignment y , each clause must have a selected true literal with $D_i^y(p) \geq 0$, and with $D_i^y(p) > 0$ whenever that clause can still be violated. These inequalities force monotone progress of the selected coordinates and rule out positive-objective limit points. For private literals, each clause has an independent selected variable, so the condition holds from every initialization.

Corollary 3.6 (Private literals imply convergence). *If every clause C_j contains a private literal, meaning a literal whose variable does not appear in any other clause, then for every $p^0 \in [0, 1]^n$ and every $\eta > 0$, projected gradient descent on $\mathcal{L}_T$ satisfies $\mathcal{L}_T(p^t) \rightarrow 0$.*

Beyond such initialization-free cases, the same condition can be used as an initialization diagnostic. The exactly-one constraint gives a simple example. After choosing the coordinate that should become one-hot, the following condition certifies convergence.

Corollary 3.7 (Exactly-one certificate). *Let $T_{\text{EO}(n)} = (v_1 \vee \dots \vee v_n) \wedge \bigwedge_{1 \leq i < j \leq n} (\neg v_i \vee \neg v_j)$ be the CNF encoding of the exactly-one constraint. Fix $r \in [n]$. If the initialization satisfies*

$$p_r^0 > \frac{1}{2} \quad \text{and} \quad \prod_{i \neq r} (1 - p_i^0) > \sum_{i \neq r} p_i^0, \quad (\text{EO})$$

then projected gradient descent on $\mathcal{L}_T$ satisfies $\mathcal{L}_T(p^t) \rightarrow 0$.

3.3 Invariant subspaces for gradient descent

The preceding results highlight how a biased initialization can induce the gradient dynamics needed for global convergence to zero objective value. On the other hand, symmetry in the initialization can create an obstruction when satisfying assignments do not share the same symmetry. Gradient descent may be unable to realize them if the initialization lies in a symmetry-induced subspace. We formalize this first through a general invariant-subspace criterion.

Proposition 3.8 (Invariant affine subspaces). *Let $L_T : \mathbb{R}^n \rightarrow \mathbb{R}$ be differentiable, and let $M = \{p \in \mathbb{R}^n : Ap = b\}$ be a nonempty affine subspace. Suppose that $A\nabla L_T(p) = 0$ for every $p \in M$. Then M is invariant under gradient descent: if $p^0 \in M$, then every subsequent iterate p^t remains in M .*

The exactly-one formula from Corollary 3.7 gives a concrete example. Its objective is invariant under every permutation of the variables, so the diagonal subspace

$$M_{\text{sym}} = \{p \in [0, 1]^n : p_1 = \dots = p_n\}$$

is invariant under gradient descent. Let A be the $(n-1) \times n$ matrix with rows $e_i - e_n$. Then $M_{\text{sym}} = \{p : Ap = 0\}$. If $p = \alpha \mathbf{1}$, then all gradient coordinates are equal, and hence $A\nabla L_T(p) = 0$. But every satisfying assignment of $T_{\text{EO}(n)}$ is one-hot, so no satisfying assignment lies in M_{sym} when $n > 1$. Thus symmetric initialization traps gradient descent in an invariant subspace with no zero-objective point. Corollary 3.7 shows that this obstruction can be overcome with an appropriate asymmetric initialization, since the certified basin for exactly-one contains points that converge to a one-hot satisfying assignment.

A useful perspective on this phenomenon can be obtained through the signed typed variable-clause graph G_T defined in Section 2. Consider the stable colorings of G_T . A coloring is stable if nodes with the same color have the same type and the same signed multiset of neighboring colors. Equivalently, it is a fixed point of 1-Weisfeiler-Leman (WL) color refinement. These color classes are especially relevant for graph neural networks, a popular architectural choice for Boolean satisfiability and other constraint satisfaction problems [5, 6, 50, 43, 33, 32, 55, 53, 36, 48]. We return to these symmetry obstructions in the parametrized setting in Section 3.4, and later in Section 4 we will see how formula transformations can be used to help overcome them.

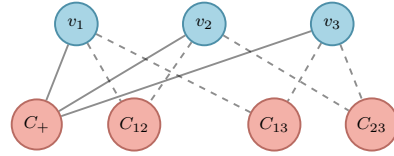


Figure 1: The variable-clause graph for the $n = 3$ exactly-one formula from Corollary 3.7: $(v_1 \vee v_2 \vee v_3) \wedge \bigwedge_{1 \leq i < j \leq 3} (\neg v_i \vee \neg v_j)$. Variable nodes are blue and clause nodes are red. The three variable nodes are automorphic, matching the diagonal invariant subspace discussed above.

Theorem 3.9 (Stable color classes are invariant). *Let $T = (V, \Phi)$ be a CNF instance with multilinear objective $\mathcal{L}_T$, and let χ be a stable coloring of G_T . Suppose that the initialization satisfies $p_u^0 = p_v^0$ whenever $u, v \in V$ and $\chi(u) = \chi(v)$. Then, under gradient descent, $p_u^t = p_v^t$ for every $t \geq 0$ whenever $u, v \in V$ and $\chi(u) = \chi(v)$.*

This means stable color classes are a fixed point not only for the 1-WL algorithm but also for gradient descent. If the initialization is constant on each color, gradient descent always assigns the same p_i to all variables in the same color class. Thus, stable color classes define effective degrees of freedom for direct gradient descent. If satisfying assignments require variables in one color class to take different Boolean values, such a run cannot reach those assignments without some symmetry-breaking mechanism. This obstruction is initialization-dependent for direct gradient descent. For 1-WL message-passing models, the same obstruction can be imposed by the architecture.

3.4 Parametrizing gradient descent with neural networks

The gradient update in Section 2 optimized continuous variables z through the objective $F_T(z) = \mathcal{L}_T(\pi(z))$. A neural parametrization replaces the variables z with network parameters θ and writes the resulting marginals as $p_\theta(G_T) \in [0, 1]^n$. Training then optimizes

$$F_T(\theta; G_T) := \mathcal{L}_T(p_\theta(G_T)), \quad \theta^{t+1} = \theta^t - \eta \nabla_\theta F_T(\theta^t; G_T).$$

We now ask what restrictions such a parametrization imposes on the marginal vectors that training can produce. The preceding invariant-subspace result, Theorem 3.9, concerns direct optimization

over the marginal variables. If the initialization lies in an invariant subspace M and the marginal-space gradient field is tangent to M , then gradient descent cannot leave M . Neural parametrizations introduce a different mechanism. Writing $\rho_T(\theta) := p_\theta(G_T)$, the attainable marginals are constrained not only by the objective $\mathcal{L}_T$, but also by the image of ρ_T , which is determined by the model. If $\rho_T(\Theta) \subseteq M$, then every parameter-space trajectory produces marginals in M , independently of the initialization and of the optimizer. Thus parametrization can turn an initialization-dependent invariant of the dynamics into an architecture- and feature-dependent restriction on the attainable marginal vectors. The message-passing result below is one instance of this general principle. For any coloring χ , let M_χ be the variable color-constant marginal subspace

$$M_\chi = \{p \in [0, 1]^n : p_i = p_k \text{ whenever } \chi(v_i) = \chi(v_k)\}.$$

Let χ^R and χ^∞ be the R -round and stable signed 1-WL colorings of G_T . Since WL-refinement only splits colors, $M_{\chi^R} \subseteq M_{\chi^\infty}$. We can relate this observation to the marginals $p_\theta(G_T)$ generated by 1-WL message-passing GNNs, including GIN-style architectures. Such networks are constant on 1-WL color classes [63, 37], and so is $p_\theta(G_T)$.

Theorem 3.10 (GNN training cannot leave the 1-WL subspace). *Consider a depth- R 1-WL message-passing GNN on G_T , with shared updates, permutation-invariant aggregation, signed edge labels, node types, and initial features that are constant on initial node colors. The network has no node identifiers that distinguish vertices inside the same initial color class. A shared variable readout produces the marginal vector $p_\theta(G_T)$. Then every parameter-space training trajectory $\{\theta^t\}_{t \geq 0}$ satisfies*

$$p_{\theta^t}(G_T) \in M_{\chi^R} \quad \text{and hence} \quad p_{\theta^t}(G_T) \in M_{\chi^\infty} \quad \text{for all } t \geq 0.$$

Theorem 3.10 makes the obstruction architectural: every parameter value already produces marginals in the WL color-constant subspace. In the following section we will see how common formula transformations can be used to overcome symmetry-related limitations.

4 Formula transformations

Formula transformations are a standard tool in SAT solvers, where preprocessing and inprocessing routines simplify CNF instances before or during search [16]. The same satisfiability-preserving operations have been examined as label-preserving augmentations for contrastive learning with neural SAT solvers [14]. We use the following transformations (details in Appendix B): unit propagation (UP), unit-literal augmentation (AU), pure literal elimination (PL), subsumed clause elimination (SC), subsumed clause addition (SA), clause resolution (CR), and variable elimination (VE).

Although these transformations preserve satisfiability of the formula, they are not neutral for continuous optimization. By changing the signed variable–clause graph, they can change invariant subspaces and sometimes break symmetries that would otherwise constrain gradient descent or a 1-WL GNN. By adding, deleting, or shortening clause-violation monomials, they can also change the tightness of $\mathcal{L}_T$ as a union-bound surrogate for exact unsatisfiability: redundant clause additions typically loosen the bound, while simplifications often tighten the transported unsatisfiability bound. Finally, transformations that introduce or eliminate variables change the product space itself. They preserve whether a satisfying assignment exists, but they can rescale or otherwise alter the product-measure satisfiability probability

by creating new satisfying and unsatisfying assignments over the extended variables. Table 1 summarizes these effects for the transformations; a detailed analysis is in Appendix D. Here, $\Pr[\Phi(X) = 0]$ tracks the product-measure unsatisfiability probability, *Objective* shows the unnormalized multilinear objective $\mathcal{L}_T$, *Tightness* the transported union-bound tightness relative to $\Pr[\Phi(X) = 0]$, and *Smoothness* the structural bound controlling $\|\nabla^2 \mathcal{L}_T\|_2$ in Eq. (2). Taken together, Table 1 shows

Transf.	$\Pr[\Phi(X) = 0]$	Objective	Tightness	Smoothness
AU	↑	–	↓	↑
SA	=	↑	↓	↑
CR	=	↑	↓	↑
UP	↓	–	↑	↓
SC	=	↓	↑	↓
PL	↓	↓	↑	↓

Table 1: Qualitative effect of label-preserving transformations. “Tightness”: transported union-bound tightness, “Smoothness”: structural Hessian bound underlying Eq. (2). Here = means unchanged, while – means that the effect can go in either direction.

that the size-increasing transformations considered here can loosen the union bound, worsen the structural smoothness bound, and, in the case of unit-literal augmentation reduce the probability of satisfiability.

Effect on symmetries. The neural-parametrization viewpoint gives a concrete reason why formula transformations can help neural solvers. They may introduce asymmetric structure that makes previously indistinguishable nodes distinguishable to WL-style message passing, addressing SAT-specific expressivity barriers identified in prior work [44]. The following informal theorem summarizes two precise transformation results, stated and proved as Theorems C.4 and C.6 in Appendix C.

Theorem 4.1 (Informal: formula transformations can break WL barriers). *Write $T \equiv_{\text{WL}} T'$ when their stable 1-WL color multisets agree. For a formula T , write T_{CR} for a one-step resolution transform, T_{VE} for a variable-elimination transform, and T_{AU} for its random unit-literal augmentation.*

- *There exist satisfiable and unsatisfiable formulas T, T' with $T \equiv_{\text{WL}} T'$ such that $T_{\text{CR}} \not\equiv_{\text{WL}} T'_{\text{CR}}$, and likewise $T_{\text{VE}} \not\equiv_{\text{WL}} T'_{\text{VE}}$.*
- *For a formula T with m clauses and no unit clauses, if T_{AU} is formed using the AU transformation with $O(\log m)$ random unit literals, then $G_{T_{\text{AU}}}$ is identified by 1-WL with high probability.*

Thus the same WL analysis has two sides. Without transformation, stable color classes describe a hard architectural invariance. With suitable formula transformations, asymmetric structure can create anchors that make clauses distinguishable by WL, so the color-constant restriction no longer forces the same collapse on the transformed representation.

5 Experiments

Our experiments probe the main phenomena discussed in previous sections, including initialization, symmetry breaking, formula transformations, and neural parametrization. Full experimental protocols, implementation details, and extended tables are given in Appendix E and Appendix G. Due to space constraints we omit the discussion of two empirical investigations, but their details can be found in the appendix. Optimization with exact probability vs union bound (Appendix E.5) and a design choice experiment that investigates ADAM vs GD, Symmetric vs Gaussian Initialization, and Projection vs Sigmoid (Appendix E.3).

5.1 Convergence, initialization, and transformations

We first test the certified-basin condition on two symmetric families where it gives an explicit boundary, the parity-one XOR cycle and the exactly-one CNF. For each formula, we run projected gradient descent from initializations on both sides of the certificate boundary and check whether the rounded endpoint satisfies the formula. The left part of Figure 2 verifies that positive certificate margins solve after rounding, while negative margins fail.

The right part of Figure 2 shows the same exactly-one family as a symmetry test. Symmetrically initialized projected GD stays in the diagonal subspace, so it cannot reach a one-hot satisfying assignment. A message-passing GNN without node identifiers has the same obstruction because all original variables lie in the same 1-WL class. Unit-literal augmentation changes the graph seen by the GNN and lets the neural model enter a one-hot basin. The same augmentation does not make symmetrically initialized projected GD solve the instances, suggesting that asymmetry by itself is not always enough.

5.2 Transformation effects in direct optimization

We test the effect of transformations on direct optimization with gradient descent on the G4SATBench [32] benchmark. Table 2 contains a short summary. Full results can be found in Table 6 in the appendix. Formulas with increased size that contain unit literals, subsumed clauses, and resolvents perform worse than the baseline that has all of them removed. This that transformations like unit propagation that are used in solvers could also be beneficial for gradient descent.

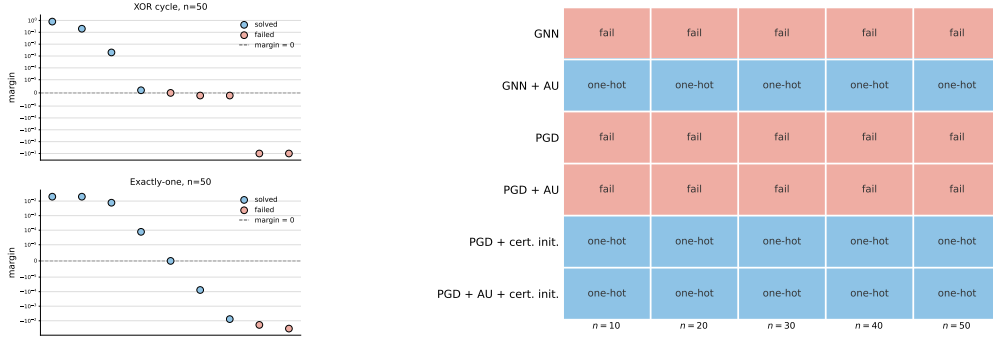


Figure 2: Initialization, symmetry, and augmentation diagnostics. The left part shows certified-basin boundary diagnostics for the $n = 50$ parity-one XOR cycle (top) and exactly-one CNF (bottom). Blue points round to satisfying assignments and red points do not. The right part reports whether each method converges to a one-hot assignment on the original EO variables.

Table 2: Results of gradient descent with formula transformations. Win % is the fraction of family/difficulty cells matching or improving on the untransformed baseline. Lower is better.

Method	Mean final original objective	Win %
Baseline	4.91 ± 8.41	—
Unit literals	19.15 ± 31.57	4.8
Subsumed clauses	5.09 ± 8.64	38.1
Resolvers	5.85 ± 8.46	0.0

5.3 Benefits of Parametrization

We next ask whether parametrizing gradient descent with a neural network can improve over vanilla gradient descent. We compare a simple sigmoid gradient descent baseline and FourierSAT against a simple 1-WL equivalent GNN and a more powerful handcrafted architecture which we call $\text{GNN}_{\text{walkSAT}}$ on a large collection of benchmark instances from G4SATBench. Figure 3 shows that the GD baseline is already strong across the mixed split, especially on medium and hard instances. A simple GNN_{Base} , which is a simple 1-WL-equivalent model, does not by itself reliably improve over the direct GD approach. By contrast, the bespoke $\text{GNN}_{\text{walkSAT}}$ architecture uses a larger architecture with various symmetry-breaking mechanisms and is able to achieve superior results. Architecture details are given in Appendix F.

5.4 Transformations in self-supervised neural SAT

Transformations help neural SAT solvers in two different ways. During training, they can create artificial data in a self-supervised setting where labeled instances are scarce. During inference, they can boost performance by performing forward passes over transformations of the same instance. For the inference-time experiment, we first train each neural architecture self-supervised on 1000 instances from a given G4SATBench dataset and evaluate on a separate test set of 100 instances. At test time, each instance is run once in its original form and then $\tau = 128$ additional times. In the Restarts baseline, those additional executions only change the random initial assignment in $\{0, 1\}^N$ that is given as input to the model. In the augmentation variants, they also apply a fresh graph transformation before the forward pass. The inference-time results show that transformations can improve over plain restarts. The full table is given in Appendix E, Table 8. For the training-time experiment, we focus on $\text{GNN}_{\text{walkSAT}}$ and compare scarce-data training sets built from original formulas only against mixed training sets that replace half of those originals with transformed copies. Concretely, we form original-only datasets of size 100, 200, and 400, and compare them with mixed datasets of size 200 and 400 in which exactly half of the formulas come from the original distribution and the other half are augmentations of the first half. We evaluate on 100 test instances. In this experiment we report the rounded literal predictions directly, rather than the discretization procedure used in the inference-time study. Figure 4 shows the training-time effect. On 3-sat, augmented data nearly matches the benefit of collecting more original formulas. On the highly symmetric



Figure 3: Direct GD and FourierSAT-style optimization on the mixed G4SATBench split. Cell annotations show mean final expected unsatisfied clauses and color shows the $\log_{10}$ relative gap to the best method in each column. Lower values and lighter cells are better.

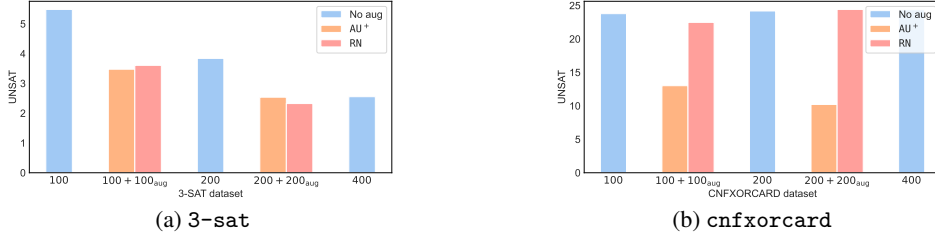


Figure 4: Training-time augmentation with $\text{GNN}_{\text{WalkSAT}}$. The augmented datasets mix original formulas with transformed copies. On 3-sat, augmentation nearly closes the gap to using more original samples. On the symmetric cnfxorcard family, AU^+ (see Appendix B for the definition) gives a clear improvement over original-only and random (non label-preserving) augmentation RN.

cnfxorcard family, unit-literal augmentation gives a larger gain and clearly beats both original-only training and random augmentation. Together with the inference-time results, this supports the view that transformations are not just preprocessing operations. They can also provide useful structural variation for parametrized gradient descent.

6 Conclusion and Limitations

The purpose of this work is to advance the understanding of gradient-based SAT solving by providing a collection of insights and experimental investigations. Our convergence conditions are sufficient rather than necessary, so they identify certified success cases but do not fully characterize when gradient descent on SAT formulas succeeds or fails. The transformation results should also be read as landscape-level and statistical statements rather than universal prescriptions: the same transformation can help some formulas and harm others based on the interactions between formula structure, initialization, and optimization parameters. Finally, the WL-symmetry invariance theorem is an exact-arithmetic statement: in finite precision, numerical errors can accumulate over long gradient trajectories, so symmetric initializations may sometimes lead to asymmetric outputs.

Conclusion. We have presented a comprehensive analysis of a standard continuous formulation of the satisfiability problem and proposed the use of transformations as a method of overcoming symmetry obstructions and improving optimization outcomes. We hope our work provides the basis for further developments in this area. There are many interesting topics that we were not able to cover that could be the subject of future investigations, such as the analysis of learning-augmented solvers or a study of the effect of different SAT encodings in the optimization process.

References

- [1] Dimitris Achlioptas and Fotis Iliopoulos. Focused stochastic local search and the Lovász local lemma. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2024–2038. SIAM, 2016.

- [2] Dimitris Achlioptas and Federico Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, pages 130–139, 2006.
- [3] Ismail R Alkhouri, George K Atia, and Alvaro Velasquez. A differentiable approach to the maximum independent set problem using dataless neural networks. *Neural Networks*, 155:168–176, 2022.
- [4] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2016.
- [5] Saeed Amizadeh, Sergiy Matushevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *International Conference on Learning Representations*, 2018.
- [6] Saeed Amizadeh, Sergiy Matushevych, and Markus Weimer. Pdp: A general neural framework for learning constraint satisfaction solvers, 2019.
- [7] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. Community structure in industrial SAT instances. *Journal of Artificial Intelligence Research*, 66:443–472, 2019.
- [8] Armin Biere et al. Lingeling, plingeling and treengeling entering the sat competition 2013. *Proceedings of SAT competition*, 2013:1, 2013.
- [9] Fanchen Bu, Hyeonsoo Jo, Soo Yong Lee, Sungsoo Ahn, and Kijung Shin. Tackling prevalent conditions in unsupervised combinatorial optimization: Cardinality, minimum, covering, and more. *arXiv preprint arXiv:2405.08424*, 2024.
- [10] Yunuo Cen, Zixuan Wang, Jintao Zhang, Zhiwei Zhang, and Xuanyao Fong. Fourierscp: Differentiable constraint satisfaction problem solving by walsh-fourier expansion. *arXiv preprint arXiv:2510.04480*, 2025.
- [11] Yunuo Cen, Zhiwei Zhang, and Xuanyao Fong. Massively parallel continuous local search for hybrid sat solving on gpus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 11140–11149, 2025.
- [12] Peter C Cheeseman, Bob Kanefsky, and William M Taylor. Where the really hard problems are. In *IJCAI*, volume 91, pages 331–337, 1991.
- [13] James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 148–159, 1996.
- [14] Haonan Duan, Pashootan Vaezipoor, Max B Paulus, Yangjun Ruan, and Chris Maddison. Augment with care: Contrastive learning for combinatorial problems. In *International Conference on Machine Learning*, pages 5627–5642. PMLR, 2022.
- [15] Shaddin Dughmi. Submodular functions: Extensions, distributions, and algorithms. a survey. *arXiv preprint arXiv:0912.0322*, 2009.
- [16] Niklas Eén and Armin Biere. Effective preprocessing in sat through variable and clause elimination. In *Theory and Applications of Satisfiability Testing*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005.
- [17] Heidi Gebauer, Tibor Szabó, and Gábor Tardos. The local lemma is asymptotically tight for sat. *Journal of the ACM (JACM)*, 63(5):1–32, 2016.
- [18] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.
- [19] Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24(1):67–100, 2000.

- [20] Abdelrahman Hosny and Sherief Reda. torchmsat: A gpu-accelerated approximation to the maximum satisfiability problem. *arXiv preprint arXiv:2402.03640*, 2024.
- [21] Sangram K. Jena, K. Subramani, and Alvaro Velasquez. Differentiable discrete optimization using dataless neural networks. In *International Conference on Combinatorial Optimization and Applications*. Springer Nature Switzerland, 2023.
- [22] Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672, 2020.
- [23] Nikolaos Karalias, Akbar Rafiey, Yifei Xu, Zhishang Luo, Behrooz Tahmasebi, Connie Jiang, and Stefanie Jegelka. Geometric algorithms for neural combinatorial optimization with constraints. *arXiv preprint arXiv:2510.24039*, 2025.
- [24] Nikolaos Karalias, Joshua Robinson, Andreas Loukas, and Stefanie Jegelka. Neural set function extensions: Learning with discrete functions in high dimensions. *Advances in Neural Information Processing Systems*, 35:15338–15352, 2022.
- [25] Richard M Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975.
- [26] Marcel Kollovich, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Expected probabilistic hierarchies. In *Advances in Neural Information Processing Systems*, volume 37, pages 13818–13850, 2024.
- [27] Martin Krutsky et al. Binarizing physics-inspired gnns for combinatorial optimization. *arXiv preprint arXiv:2507.13703*, 2025.
- [28] Florent Krzakala, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007.
- [29] Anastasios Kyrillidis, Anshumali Shrivastava, Moshe Vardi, and Zhiwei Zhang. Fouriersat: A fourier expansion-based algebraic framework for solving hybrid boolean constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1552–1560, 2020.
- [30] Anastasios Kyrillidis, Moshe Y. Vardi, and Zhiwei Zhang. On continuous local bdd-based search for hybrid sat solving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3841–3850, 2021.
- [31] James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pages 567–576, 2015.
- [32] Zhaoyu Li, Jinpei Guo, and Xujie Si. G4satbench: Benchmarking and advancing sat solving with graph neural networks. *arXiv preprint arXiv:2309.16941*, 2023.
- [33] Zhaoyu Li and Xujie Si. Nsnet: A general neural probabilistic framework for satisfiability problems. In *Advances in Neural Information Processing Systems*, volume 35, pages 25573–25585, 2022.
- [34] Minghao Liu, Fuqi Jia, Pei Huang, Fan Zhang, Yuchen Sun, Shaowei Cai, Feifei Ma, and Jian Zhang. Can graph neural networks learn to solve maxsat problem? *arXiv preprint arXiv:2111.07568*, 2021.
- [35] Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.
- [36] Sota Moriyama and Katsumi Inoue. Graph-based attention for differentiable maxsat solving. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

- [37] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.
- [38] Robin A Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):1–15, 2010.
- [39] Konrad Munding, Max Zimmer, Aldo Kiem, Christoph Spiegel, and Sebastian Pokutta. Neural discovery in mathematics: Do machines dream of colored planes? *arXiv preprint arXiv:2501.18527*, 2025.
- [40] Robert R. Nerem, Zhishang Luo, Akbar Rafiey, and Yusu Wang. Differentiable extensions with rounding guarantees for combinatorial optimization over permutations. *arXiv preprint arXiv:2411.10707*, 2024.
- [41] Yurii Nesterov. *Lectures on Convex Optimization*, volume 137. Springer, 2018.
- [42] Zack Newsham, Vijay Ganesh, Sebastian Fischmeister, Gilles Audemard, and Laurent Simon. Impact of community structure on sat solver performance. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 252–268. Springer, 2014.
- [43] 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.
- [44] Saku Peltonen and Roger Wattenhofer. On the expressive power of gnns for boolean satisfiability. In *International Conference on Learning Representations*, 2026.
- [45] Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.
- [46] Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 43346–43367, 2024.
- [47] Taisuke Sato and Ryosuke Kojima. Matsat: a matrix-based differentiable sat solver. *arXiv preprint arXiv:2108.06481*, 2021.
- [48] Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.
- [49] Bart Selman, Henry A Kautz, Bram Cohen, et al. Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability*, 26:521–532, 1993.
- [50] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a sat solver from single-bit supervision, 2018.
- [51] Ilya Shlyakhter. Generating effective symmetry-breaking predicates for search problems. *Discrete Applied Mathematics*, 155(12):1539–1548, 2007.
- [52] Arie Soeteman, Michael Benedikt, Martin Grohe, and Balder ten Cate. How expressive are graph neural networks in the presence of node identifiers? *arXiv preprint arXiv:2601.21882*, 2026.
- [53] Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:98, 2021.
- [54] Jan Tönshoff and Martin Grohe. Learning from algorithm feedback: One-shot sat solver guidance with gnns. *arXiv preprint arXiv:2505.16053*, 2025.
- [55] Jan Tönshoff, Berke Kisin, Jakob Lindner, and Martin Grohe. One model, any csp: Graph neural networks as fast global search heuristics for constraint satisfaction. *arXiv preprint arXiv:2208.10227*, 2022.

- [56] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
- [57] Alvaro Velasquez, Susmit Jha, and Ismail R. Alkhoury. On the dataless training of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 2026.
- [58] Haoyu Peter Wang, Nan Wu, Hang Yang, Cong Hao, and Pan Li. Unsupervised learning for combinatorial optimization with principled objective relaxation. In *Advances in Neural Information Processing Systems*, volume 35, pages 31444–31458, 2022.
- [59] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019.
- [60] Wenxi Wang, Yang Hu, Mohit Tiwari, Sarfraz Khurshid, Kenneth McMillan, and Risto Mikkulainen. Neurocomb: Improving sat solving with graph neural networks. *arXiv preprint arXiv:2110.14053*, 2021.
- [61] Frederik Wenkel et al. Towards a general recipe for combinatorial optimization with multi-filter gnns. *arXiv preprint arXiv:2405.20543*, 2024.
- [62] Ryan Williams, Carla P. Gomes, and Bart Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 222–230, 2003.
- [63] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.
- [64] Morris Yau, Nikolaos Karalias, Eric Lu, Jessica Xu, and Stefanie Jegelka. Are graph neural networks optimal approximation algorithms? *Advances in Neural Information Processing Systems*, 37:73124–73181, 2024.
- [65] Yu Zhang, Hui-Ling Zhen, Mingxuan Yuan, and Bei Yu. Diffsat: Differential maxsat layer for sat solving. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 114:1–114:7, 2024.

A Proofs for Section 3

A.1 Proof of Proposition 3.1

Proof. Fix a clause C , and let $\tau(\ell) = +1$ for a positive literal and $\tau(\ell) = -1$ for a negated literal. If $\ell = v_i$ is a positive literal, then

$$1 - p_i = 1 - \frac{1 - s_i}{2} = \frac{1 + s_i}{2} = \frac{1 + \tau(\ell)s_i}{2}.$$

If $\ell = \neg v_i$ is a negated literal, then

$$p_i = \frac{1 - s_i}{2} = \frac{1 + \tau(\ell)s_i}{2}.$$

Multiplying the appropriate factors across literals in C gives

$$q_C(p) = \frac{\prod_{\ell \in C} (1 + \tau(\ell)s_{|\ell|})}{2^{|C|}}.$$

By the normalized clausewise Fourier convention, the clause contribution is -1 on satisfying assignments and $+1$ on violating assignments. Substituting the preceding identity into this clause expansion yields

$$\psi_C(s) = 2q_C(p) - 1.$$

□

A.2 Proof of Proposition 3.2

Proof. For each clause C_j , define its violation probability by

$$q_j(p) := \prod_{i \in N^+(j)} (1 - p_i) \prod_{i \in N^-(j)} p_i. \quad (3)$$

Then

$$\mathcal{L}_T(p) = \sum_{j=1}^m q_j(p). \quad (4)$$

Let $A(j) := N^+(j) \cup N^-(j)$. For each variable index i , define

$$L_i(T) := \sum_{j: i \in A(j)} (a(C_j) - 1), \quad L(T) := \max_{i \in [n]} L_i(T). \quad (5)$$

By (4), $\mathcal{L}_T$ is multilinear. Therefore, for each $i \in [n]$,

$$\frac{\partial \mathcal{L}_T}{\partial p_i}(p) = \sum_{j: i \in A(j)} s_{j,i} \prod_{\ell \in N^+(j) \setminus \{i\}} (1 - p_\ell) \prod_{\ell \in N^-(j) \setminus \{i\}} p_\ell, \quad (6)$$

where

$$s_{j,i} = \begin{cases} -1, & i \in N^+(j), \\ 1, & i \in N^-(j). \end{cases}$$

Since $\mathcal{L}_T$ is affine in each coordinate separately, its diagonal second derivatives vanish. For distinct $i \neq k$,

$$\frac{\partial^2 \mathcal{L}_T}{\partial p_i \partial p_k}(p) = \sum_{j: \{i,k\} \subseteq A(j)} s_{j,i,k} \prod_{\ell \in N^+(j) \setminus \{i,k\}} (1 - p_\ell) \prod_{\ell \in N^-(j) \setminus \{i,k\}} p_\ell, \quad (7)$$

where $s_{j,i,k} \in \{-1, 1\}$ records whether i and k occur in C_j with the same or opposite polarity.

Every factor in the products in (7) lies in $[0, 1]$, so for all $p \in [0, 1]^n$,

$$\left| \frac{\partial^2 \mathcal{L}_T}{\partial p_i \partial p_k}(p) \right| \leq c_{ik}, \quad c_{ik} := |\{j : \{i, k\} \subseteq A(j)\}|. \quad (8)$$

Summing the absolute values across the i -th row of the Hessian and using that the diagonal entries are zero gives

$$\begin{aligned} \sum_{k=1}^n \left| \frac{\partial^2 \mathcal{L}_T}{\partial p_i \partial p_k}(p) \right| &= \sum_{k \neq i} \left| \frac{\partial^2 \mathcal{L}_T}{\partial p_i \partial p_k}(p) \right| \\ &\leq \sum_{k \neq i} \sum_{j: \{i,k\} \subseteq A(j)} 1 \\ &= \sum_{j: i \in A(j)} \sum_{k \in A(j) \setminus \{i\}} 1 \\ &= \sum_{j: i \in A(j)} (a(C_j) - 1) \\ &= L_i(T) \leq L(T). \end{aligned} \quad (9)$$

Therefore

$$\|\nabla^2 \mathcal{L}_T(p)\|_\infty \leq L(T). \quad (10)$$

Since $\mathcal{L}_T$ is C^2 , its Hessian is symmetric, and so the absolute column sums equal the absolute row sums. Thus (9) also yields

$$\|\nabla^2 \mathcal{L}_T(p)\|_1 \leq L(T). \quad (11)$$

Using the standard inequality

$$\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty},$$

and combining (10) with (11), we obtain

$$\|\nabla^2 \mathcal{L}_T(p)\|_2 \leq \sqrt{\|\nabla^2 \mathcal{L}_T(p)\|_1 \|\nabla^2 \mathcal{L}_T(p)\|_\infty} \leq L(T).$$

Finally, since $a(C_j) \leq w$ for every clause C_j , we have

$$L_i(T) = \sum_{j: i \in A(j)} (a(C_j) - 1) \leq \sum_{j: i \in A(j)} (w - 1) = d_i(w - 1) \leq \Delta(w - 1).$$

Taking the maximum over i gives $L(T) \leq (w-1)\Delta$, and the bound $\Delta \leq m$ yields $L(T) \leq (w-1)m$. Hence

$$\|\nabla^2 \mathcal{L}_T(p)\|_2 \leq L(T) \leq (w-1)\Delta \leq (w-1)m.$$

This proves (2). $\square$

A.3 Proof of Proposition 3.3

Proof. For each clause C_j , let E_j be the event that C_j is violated by $X \sim \mu_p$. Then $\Phi(X) = 0$ if and only if at least one clause is violated, i.e.,

$$\{\Phi(X) = 0\} = \bigcup_{j=1}^m E_j.$$

By Boole's inequality,

$$\Pr[\Phi(X) = 0] = \Pr\left[\bigcup_{j=1}^m E_j\right] \leq \sum_{j=1}^m \Pr[E_j].$$

Using the definition of the clause violation monomials $q_j(p)$, the right-hand side is exactly

$$\sum_{j=1}^m \Pr[C_j(X) = 0] = \mathcal{L}_T(p),$$

which proves the claim. $\square$

A.4 Proof of Proposition 3.4

Proof. For each clause C_a , let

$$E_a := \{x \in \{0, 1\}^n : C_a(x) = 0\}$$

denote its falsification event. Write

$$C_a = \bigvee_{i \in P_a} x_i \vee \bigvee_{j \in N_a} \neg x_j, \quad C_b = \bigvee_{i \in P_b} x_i \vee \bigvee_{j \in N_b} \neg x_j,$$

with $P_a \cap N_a = P_b \cap N_b = \emptyset$. To falsify C_a , one must impose

$$x_i = 0 \ \forall i \in P_a, \quad x_j = 1 \ \forall j \in N_a,$$

and similarly for C_b . Hence $E_a \cap E_b \neq \emptyset$ if and only if these two partial assignments are compatible, i.e. if and only if

$$P_a \cap N_b = \emptyset \quad \text{and} \quad N_a \cap P_b = \emptyset.$$

Equivalently,

$$E_a \cap E_b = \emptyset \iff (P_a \cap N_b) \cup (N_a \cap P_b) \neq \emptyset.$$

Thus two distinct clauses have disjoint falsification sets if and only if they contain a complementary literal pair. This proves that Φ is hitting if and only if the events $E_1, \dots, E_m$ are pairwise disjoint, which is in turn equivalent to saying that every assignment falsifies at most one clause.

Write $B_\Phi(p) := \mathcal{L}_T(p)$ for the union-bound polynomial. It remains to show that this is equivalent to $P_\Phi \equiv B_\Phi$. Both P_Φ and B_Φ are multilinear polynomials in $p_1, \dots, p_n$, so it is enough to compare

them on the Boolean cube $\{0, 1\}^n$. Fix $x \in \{0, 1\}^n$ and evaluate at $p = x$. Then μ_x is the point mass at the assignment x , so

$$P_\Phi(x) = \begin{cases} 1, & \text{if } x \text{ falsifies at least one clause of } \Phi, \\ 0, & \text{otherwise,} \end{cases}$$

whereas

$$B_\Phi(x) = \sum_{a=1}^m \mathbf{1}_{E_a}(x) = \#\{a \in [m] : x \in E_a\},$$

namely the number of clauses falsified by x . Therefore

$$P_\Phi(x) = B_\Phi(x) \text{ for all } x \in \{0, 1\}^n$$

if and only if every assignment falsifies at most one clause. Since multilinear polynomials are determined by their values on $\{0, 1\}^n$, this is equivalent to $P_\Phi \equiv B_\Phi$. $\square$

A.4.1 A toy example for tightness and optimization signal

The equality $P_\Phi \equiv \mathcal{L}_T$ has different implications depending on whether the goal is recovering a satisfying assignment or reducing the number of violated clauses. The following two-variable examples separate these effects in closed form. Throughout, $p_i = \Pr[X_i = 1]$.

A satisfiable formula. Let

$$F_{\text{sat}} = (v_1) \wedge (v_1 \vee v_2), \quad H_{\text{sat}} = (v_1 \vee v_2) \wedge (v_1 \vee \neg v_2).$$

Both CNFs are logically equivalent to (v_1) , with satisfying assignments $(1, 0)$ and $(1, 1)$, so $P_{F_{\text{sat}}}(p) = P_{H_{\text{sat}}}(p) = 1 - p_1$. For the non-hitting representation,

$$\begin{aligned} \mathcal{L}_{F_{\text{sat}}}(p) &= (1 - p_1) + (1 - p_1)(1 - p_2) = (1 - p_1)(2 - p_2), \\ \nabla \mathcal{L}_{F_{\text{sat}}}(p) &= (-(2 - p_2), -(1 - p_1)), \quad \nabla P_{F_{\text{sat}}}(p) = (-1, 0). \end{aligned}$$

Thus $\mathcal{L}_{F_{\text{sat}}}$ pushes p_2 , even though v_2 is irrelevant to satisfiability. In contrast, the hitting representation satisfies

$$\mathcal{L}_{H_{\text{sat}}}(p) = (1 - p_1)(1 - p_2) + (1 - p_1)p_2 = 1 - p_1 = P_{H_{\text{sat}}}(p) = P_{F_{\text{sat}}}(p),$$

so the spurious p_2 -gradient disappears.

An unsatisfiable formula. Now let

$$F_{\text{unsat}} = (v_1) \wedge (\neg v_1) \wedge (v_2).$$

This formula is unsatisfiable, but setting $v_2 = 1$ still avoids the extra violation of the unit clause (v_2) . The expected violated-clause objective keeps this signal.

$$\begin{aligned} \mathcal{L}_{F_{\text{unsat}}}(p) &= (1 - p_1) + p_1 + (1 - p_2) = 2 - p_2, \\ \nabla \mathcal{L}_{F_{\text{unsat}}}(p) &= (0, -1). \end{aligned}$$

Minimizing $\mathcal{L}_{F_{\text{unsat}}}$ pushes p_2 toward 1. By contrast, since F_{unsat} is unsatisfiable, the exact probability is constant, $P_{F_{\text{unsat}}}(p) \equiv 1$, with $\nabla P_{F_{\text{unsat}}}(p) = 0$. The logically equivalent hitting CNF for false over (v_1, v_2) is

$$H_{\text{unsat}} = (v_1 \vee v_2) \wedge (v_1 \vee \neg v_2) \wedge (\neg v_1 \vee v_2) \wedge (\neg v_1 \vee \neg v_2).$$

Every assignment violates exactly one clause, and therefore

$$\mathcal{L}_{H_{\text{unsat}}}(p) \equiv 1, \quad \nabla \mathcal{L}_{H_{\text{unsat}}}(p) = 0.$$

Thus, for reducing violated clauses on an unsatisfiable formula, a perfectly tight representation can remove useful optimization signal.

A.5 Formal certified-basin theorem (Proof of Theorem 3.5)

For $y \in \{0, 1\}^n$, define $s_i := 2y_i - 1$ and

$$D_i^y(p) := -s_i \frac{\partial \mathcal{L}_T}{\partial p_i}(p).$$

Thus $D_i^y(p) > 0$ means that the negative gradient direction moves p_i toward the Boolean value y_i . For $S \subseteq [n]$, define the S -restricted forward box

$$\mathcal{B}_S(p^0, y) := \{p \in [0, 1]^n : s_i(p_i - p_i^0) \geq 0 \text{ for every } i \in S\}.$$

Equivalently, for $i \in S$, this box requires $p_i \geq p_i^0$ if $y_i = 1$, and $p_i \leq p_i^0$ if $y_i = 0$. No restriction is imposed on coordinates outside S .

Theorem A.1 (Subset certified-basin theorem). *Let $T = (V, \Phi)$ be a CNF instance. Fix $p^0 \in [0, 1]^n$, $y \in \{0, 1\}^n$, a set $S \subseteq [n]$, and a map $h : [m] \rightarrow S$ such that $h(j) \in A(j)$ and the literal of $v_{h(j)}$ in C_j is true under y . Assume*

$$D_i^y(p) \geq 0 \quad \forall i \in S, \forall p \in \mathcal{B}_S(p^0, y), \quad (\text{CB})$$

and that

$$q_j(p) > 0 \implies D_{h(j)}^y(p) > 0 \quad \forall j \in [m], \forall p \in \mathcal{B}_S(p^0, y). \quad (\text{SP})$$

Then, for every step size $\eta > 0$, projected gradient descent on $\mathcal{L}_T$ initialized at p^0 satisfies $\mathcal{L}_T(p^t) \rightarrow 0$.

Proof. For $i \in S$, the update can be written as

$$p_i^{t+1} = \Pi_{[0,1]}(p_i^t + \eta s_i D_i^y(p^t)).$$

As long as $p^t \in \mathcal{B}_S(p^0, y)$, condition (CB) implies

$$s_i(p_i^{t+1} - p_i^t) \geq 0 \quad \forall i \in S. \quad (1)$$

Indeed, if $y_i = 1$, then $s_i = 1$ and the update attempts to increase p_i ; if $y_i = 0$, then $s_i = -1$ and the update attempts to decrease p_i . Projection onto $[0, 1]$ preserves this one-dimensional monotonicity.

Since $p^0 \in \mathcal{B}_S(p^0, y)$, equation (1) implies by induction that

$$p^t \in \mathcal{B}_S(p^0, y) \quad \forall t \geq 0.$$

Moreover, for each $i \in S$, the distance

$$d_i^t := |p_i^t - y_i|$$

is nonincreasing and bounded below by 0, hence convergent.

We now prove that the objective converges to zero. Suppose otherwise. Since $[0, 1]^n$ is compact, there exists a subsequence $p^{t_k} \rightarrow p^* \in [0, 1]^n$ such that

$$L_T(p^*) > 0.$$

Because $\mathcal{B}_S(p^0, y)$ is closed and all iterates lie in it, we have

$$p^* \in \mathcal{B}_S(p^0, y).$$

Since $L_T(p^*) > 0$, there is some clause C_j with

$$q_j(p^*) > 0.$$

Let $i = h(j)$. By (SP),

$$D_i^y(p^*) > 0.$$

Also, because the literal of v_i in C_j is true under y , the corresponding violation factor in q_j vanishes if $p_i = y_i$. Since $q_j(p^*) > 0$, we must have

$$p_i^* \neq y_i.$$

Therefore there exist constants $\gamma > 0$, $\delta > 0$, and K such that for all $k \geq K$,

$$D_i^y(p^{t_k}) \geq \gamma, \quad |p_i^{t_k} - y_i| \geq \delta.$$

At each such time t_k , the projected update decreases the distance to y_i by at least

$$d_i^{t_k} - d_i^{t_k+1} \geq \min\{\eta\gamma, \delta\} > 0. \quad (2)$$

But d_i^t is nonincreasing and bounded below, so the total decrease

$$\sum_{t=0}^{\infty} (d_i^t - d_i^{t+1})$$

is finite. This contradicts the existence of infinitely many indices t_k satisfying (2). Hence no positive-objective limit point exists, and therefore

$$L_T(p^t) \rightarrow 0.$$

□

A.6 Proof of Corollary 3.6

Proof. For each clause C_j , choose one private literal and let $h(j)$ be its variable. Since $v_{h(j)}$ appears in no other clause, the chosen private variables are pairwise distinct.

Define $y \in \{0, 1\}^n$ so that every chosen private literal is true under y . This is consistent because the chosen private variables are distinct. Let

$$S := \{h(j) : j \in [m]\}.$$

Fix a clause C_j , and write $i = h(j)$. Let

$$r_j(p) := \prod_{\ell \in A(j) \setminus \{i\}} q_{j,\ell}(p),$$

where

$$q_{j,\ell}(p) = \begin{cases} 1 - p_\ell, & \ell \in N^+(j), \\ p_\ell, & \ell \in N^-(j). \end{cases}$$

Because v_i appears in no other clause, the only dependence of L_T on p_i comes from q_j . If the private literal is positive, then $y_i = 1$, $q_j(p) = (1 - p_i)r_j(p)$, and

$$D_i^y(p) = -\frac{\partial L_T}{\partial p_i}(p) = r_j(p) \geq 0.$$

If the private literal is negative, then $y_i = 0$, $q_j(p) = p_i r_j(p)$, and

$$D_i^y(p) = \frac{\partial L_T}{\partial p_i}(p) = r_j(p) \geq 0.$$

Thus (CB) holds for all $i \in S$ on all of $[0, 1]^n$.

Moreover, if $q_j(p) > 0$, then both the private literal's violation factor and $r_j(p)$ are positive. Hence

$$D_{h(j)}^y(p) = r_j(p) > 0.$$

So (SP) holds. The result follows from Theorem A.1. □

A.7 Formal symmetric-family certificates

Proof of Corollary 3.7. For $T_{\text{EO}(n)}$, the multilinear objective is

$$L_T(p) = \prod_{i=1}^n (1 - p_i) + \sum_{1 \leq i < j \leq n} p_i p_j.$$

Let $y = e_r$ and use Theorem A.1 with $S = [n]$. On the forward box $p_r \geq p_r^0$ and $p_i \leq p_i^0$ for $i \neq r$. For the winner coordinate,

$$D_r^{e_r}(p) = -\frac{\partial L_T}{\partial p_r}(p) = \prod_{i \neq r} (1 - p_i) - \sum_{i \neq r} p_i.$$

Thus, throughout the box,

$$D_r^{e_r}(p) \geq \prod_{i \neq r} (1 - p_i^0) - \sum_{i \neq r} p_i^0 > 0$$

by condition (EO).

For any loser coordinate $j \neq r$,

$$D_j^{e_r}(p) = \frac{\partial L_T}{\partial p_j}(p) = \sum_{\ell \neq j} p_\ell - \prod_{\ell \neq j} (1 - p_\ell).$$

This expression is increasing in each coordinate other than p_j , so its minimum over the box is attained at $p_r = p_r^0$ and $p_\ell = 0$ for $\ell \neq j, r$. Hence

$$D_j^{e_r}(p) \geq p_r^0 - (1 - p_r^0) = 2p_r^0 - 1 > 0.$$

Therefore $D_i^{e_r}(p) > 0$ for every coordinate i throughout the forward box. Each clause has at least one literal true under e_r : the long positive clause is certified by v_r , and every pairwise negative clause is certified by at least one loser variable. Thus the hypotheses of Theorem A.1 hold, and

$$L_T(p^t) \rightarrow 0.$$

□

Corollary A.2 (Parity-one XOR-cycle certificate). *Let n be even, and let $T_{\text{XOR}(n)}$ be the CNF*

$$\bigwedge_{i=1}^n [(v_i \vee v_{i+1}) \wedge (\neg v_i \vee \neg v_{i+1})], \quad v_{n+1} = v_1.$$

Fix an alternating satisfying assignment y , so $y_i \neq y_{i+1}$ for all i . Suppose the initialization has two levels:

$$p_i^0 = p_T^0 \text{ if } y_i = 1, \quad p_i^0 = p_F^0 \text{ if } y_i = 0,$$

with $p_T^0 > 1/2$ and $p_F^0 < 1/2$. Then projected gradient descent on $\mathcal{L}_T$ satisfies $\mathcal{L}_T(p^t) \rightarrow 0$. Equivalently, the two-level margin $\min\{2p_T^0 - 1, 1 - 2p_F^0\}$ is positive.

Proof of Corollary A.2. For the XOR cycle,

$$L_T(p) = \sum_{i=1}^n ((1 - p_i)(1 - p_{i+1}) + p_i p_{i+1}), \quad p_{n+1} = p_1,$$

with indices understood modulo n . Use Theorem A.1 with $S = [n]$. If $y_i = 1$, then both neighbors of i have target value 0, and

$$D_i^y(p) = -\frac{\partial L_T}{\partial p_i}(p) = (1 - 2p_{i-1}) + (1 - 2p_{i+1}).$$

Throughout the forward box, $p_{i-1}, p_{i+1} \leq p_F^0 < 1/2$, so $D_i^y(p) > 0$. If $y_i = 0$, then both neighbors of i have target value 1, and

$$D_i^y(p) = \frac{\partial L_T}{\partial p_i}(p) = (2p_{i-1} - 1) + (2p_{i+1} - 1).$$

Throughout the forward box, $p_{i-1}, p_{i+1} \geq p_T^0 > 1/2$, so again $D_i^y(p) > 0$. Thus (CB) holds, and in fact every coordinate has strictly positive $D_i^y(p)$ throughout the forward box.

Each edge has one endpoint true and one endpoint false under y . For the positive clause $(v_i \vee v_{i+1})$, choose the literal at the true endpoint as the designated literal; for the negative clause $(\neg v_i \vee \neg v_{i+1})$, choose the literal at the false endpoint as the designated literal. The corresponding $D_i^y(p)$ is strictly positive whenever the clause is active, so (SP) holds. The result follows from Theorem A.1. □

A.8 Proof of Proposition 3.8

Proof. Let $p \in M$. Then $Ap = b$. By assumption,

$$A\nabla L_T(p) = 0.$$

Therefore, for one step of gradient descent,

$$A(p - \eta\nabla L_T(p)) = Ap - \eta A\nabla L_T(p) = b.$$

Hence $p - \eta\nabla L_T(p) \in M$. Thus M is invariant under the unprojected update. By induction, if $p^0 \in M$, then $p^t \in M$ for all $t \geq 0$.

The same computation also yields the projected variant. If the coordinatewise projection $\Pi_{[0,1]^n}$ satisfies $\Pi_{[0,1]^n}(M) \subseteq M$, then for any $p \in M \cap [0, 1]^n$,

$$y := p - \eta\nabla L_T(p) \in M$$

by the argument above, and therefore

$$\Pi_{[0,1]^n}(y) \in M \cap [0, 1]^n.$$

So $M \cap [0, 1]^n$ is invariant under projected gradient descent as well. $\square$

A.9 Proof of Theorem 3.9

Proof. Let the clauses be

$$\Phi = C_1 \wedge \cdots \wedge C_m.$$

Recall that

$$\mathcal{L}_T(p) = \sum_{j=1}^m q_j(p), \quad q_j(p) = \prod_{i \in N^+(j)} (1 - p_i) \prod_{i \in N^-(j)} p_i.$$

We assume the usual simplification that no variable appears twice in the same clause. The signed typed variable–clause graph G_T has vertex set

$$V \cup \{C_1, \dots, C_m\}.$$

There is an edge between $i \in V$ and C_j whenever $i \in N^+(j) \cup N^-(j)$. The edge has sign $+$ if $i \in N^+(j)$, and sign $-$ if $i \in N^-(j)$.

Stability of χ means that vertices of the same color have the same signed color-neighborhood counts. In particular, variables of the same color see the same number of incident clauses of each clause color through positive and negative edges. Similarly, clauses of the same color see the same number of incident variables of each variable color through positive and negative edges.

Let

$$M_\chi = \{p \in \mathbb{R}^n : p_u = p_v \text{ whenever } \chi(u) = \chi(v)\}.$$

We will show that

$$\nabla \mathcal{L}_T(p) \in \mathcal{T}_p M_\chi \quad \text{for every } p \in M_\chi.$$

Here $\mathcal{T}_p M_\chi$ denotes the tangent space of M_χ at p . Since M_χ is a linear subspace, $\mathcal{T}_p M_\chi = M_\chi$, so this is equivalent to showing that

$$\frac{\partial \mathcal{L}_T}{\partial p_u}(p) = \frac{\partial \mathcal{L}_T}{\partial p_v}(p) \quad \text{whenever } \chi(u) = \chi(v).$$

Fix $p \in M_\chi$. Thus p is constant on variable color classes. For each variable color α , let q_α denote the common value of p_i on variables of color α . Thus

$$p_i = q_{\chi(i)} \quad \text{for every } i \in V.$$

For a clause color β , stability implies that the following quantities are well-defined:

$$a_{\beta,+,\gamma} = \#\{i \in N^+(j) : \chi(i) = \gamma\}, \quad a_{\beta,-,\gamma} = \#\{i \in N^-(j) : \chi(i) = \gamma\},$$

where C_j is any clause with $\chi(C_j) = \beta$. These counts do not depend on the choice of C_j .

Likewise, for a variable color α , stability implies that

$$b_{\alpha,+,\beta} = \#\{j : u \in N^+(j), \chi(C_j) = \beta\}, \quad b_{\alpha,-,\beta} = \#\{j : u \in N^-(j), \chi(C_j) = \beta\},$$

are well-defined for any variable u with $\chi(u) = \alpha$.

Now fix a variable u of color α . If $u \in N^+(j)$, then

$$\frac{\partial q_j}{\partial p_u}(p) = - \prod_{i \in N^+(j) \setminus \{u\}} (1 - p_i) \prod_{i \in N^-(j)} p_i.$$

If $\chi(C_j) = \beta$, then because $p \in M_\chi$, this contribution equals

$$D_{\alpha,+,\beta}(q) := - \prod_{\gamma} (1 - q_\gamma)^{a_{\beta,+,\gamma} - \mathbf{1}_{\gamma=\alpha}} \prod_{\gamma} q_\gamma^{a_{\beta,-,\gamma}}.$$

This quantity depends only on α, β , and the vector of color-values q , not on the particular choice of u or C_j .

Similarly, if $u \in N^-(j)$ and $\chi(C_j) = \beta$, then

$$\frac{\partial q_j}{\partial p_u}(p) = \prod_{i \in N^+(j)} (1 - p_i) \prod_{i \in N^-(j) \setminus \{u\}} p_i,$$

which equals

$$D_{\alpha,-,\beta}(q) := \prod_{\gamma} (1 - q_\gamma)^{a_{\beta,+,\gamma}} \prod_{\gamma} q_\gamma^{a_{\beta,-,\gamma} - \mathbf{1}_{\gamma=\alpha}}.$$

Therefore, for any variable u with $\chi(u) = \alpha$,

$$\frac{\partial \mathcal{L}_T}{\partial p_u}(p) = \sum_{\beta} b_{\alpha,+,\beta} D_{\alpha,+,\beta}(q) + \sum_{\beta} b_{\alpha,-,\beta} D_{\alpha,-,\beta}(q).$$

The right-hand side depends only on the color $\alpha = \chi(u)$. Hence, if $\chi(u) = \chi(v)$, then

$$\frac{\partial \mathcal{L}_T}{\partial p_u}(p) = \frac{\partial \mathcal{L}_T}{\partial p_v}(p).$$

Thus $\nabla \mathcal{L}_T(p) \in \mathcal{T}_p M_\chi$ for all $p \in M_\chi$.

By Proposition 3.8, M_χ is invariant under gradient descent. Hence if $p^0 \in M_\chi$, then $p^t \in M_\chi$ for every $t \geq 0$, which proves the claim.

The same argument gives the projected extension. Coordinatewise projection preserves equality constraints: if $y_u = y_v$, then

$$\Pi_{[0,1]^n}(y)_u = \Pi_{[0,1]^n}(y)_v.$$

Therefore $\Pi_{[0,1]^n}(M_\chi) \subseteq M_\chi$, and the projected version follows from the projected extension of Proposition 3.8. $\square$

A.10 Proof of Theorem 3.10

Proof. We first prove the pointwise claim that every parameter value θ produces marginals in the R -round WL subspace. Write the signed 1-WL refinement as

$$\chi^{r+1}(a) = \text{HASH}(\chi^r(a), \{(\lambda(a, b), \chi^r(b)) : b \in N(a)\}),$$

and write the GNN update as

$$h_a^{r+1} = \Psi_r(h_a^r, \text{AGG}_r\{\{\Phi_r(h_a^r, h_b^r, \lambda(a, b)) : b \in N(a)\}\}).$$

Here the message and update maps are shared across nodes, and AGG_r is permutation-invariant. The shared variable readout has the form $z_i = \rho(h_{v_i}^R)$, followed either by $p_{\theta,i}(G_T) = \sigma(z_i)$ or by the coordinatewise clamp $p_{\theta,i}(G_T) = \Pi_{[0,1]}(z_i)$.

We prove the stronger layerwise claim that, for every $0 \leq r \leq R$,

$$\chi^r(a) = \chi^r(b) \implies h_a^r = h_b^r$$

for all nodes a, b of the signed typed variable–clause graph G_T .

The base case $r = 0$ holds because the initial node features are constant on χ^0 -classes. Assume the claim holds at layer r , and let $\chi^{r+1}(a) = \chi^{r+1}(b)$. By the signed 1-WL refinement rule, this means

$$\chi^r(a) = \chi^r(b)$$

and the signed color-neighborhood multisets agree:

$$\{(\lambda(a, s), \chi^r(s)) : s \in N(a)\} = \{(\lambda(b, t), \chi^r(t)) : t \in N(b)\}.$$

By the induction hypothesis, nodes with the same χ^r -color have the same r -th layer embedding. Therefore the message multisets received by a and b are equal:

$$\{\{\Phi_r(h_a^r, h_s^r, \lambda(a, s)) : s \in N(a)\}\} = \{\{\Phi_r(h_b^r, h_t^r, \lambda(b, t)) : t \in N(b)\}\}.$$

The aggregator is permutation-invariant, so the aggregated messages are equal. Also, $\chi^r(a) = \chi^r(b)$ implies $h_a^r = h_b^r$ by the induction hypothesis. Since the same update map Ψ_r is used at every node,

$$h_a^{r+1} = h_b^{r+1}.$$

This completes the induction.

In particular, if variable nodes v_i, v_k satisfy $\chi^R(v_i) = \chi^R(v_k)$, then $h_{v_i}^R = h_{v_k}^R$. The readout ρ is shared, so $z_i = z_k$. Applying either the sigmoid or the coordinatewise clamp preserves equality, hence

$$p_{\theta, i}(G_T) = p_{\theta, k}(G_T).$$

Thus $p_{\theta}(G_T) \in M_{\chi^R}$. Since 1-WL refinement only splits color classes as the number of rounds increases, χ^R is coarser than χ^∞ , and therefore $M_{\chi^R} \subseteq M_{\chi^\infty}$. Hence

$$p_{\theta}(G_T) \in M_{\chi^\infty}.$$

Any training algorithm produces a sequence of parameter values $\theta^0, \theta^1, \theta^2, \dots$. Applying the pointwise claim to each θ^t gives

$$p_{\theta^t}(G_T) \in M_{\chi^R} \quad \forall t \geq 0.$$

The inclusion $M_{\chi^R} \subseteq M_{\chi^\infty}$ gives the second claim. $\square$

B Label-preserving augmentations

We describe a set of label-preserving augmentations (LPAs) for Boolean satisfiability problems. Each transformation preserves the satisfiability status of a formula φ , i.e., for the augmented instance $\widehat{\varphi}$ we have

$$f(\widehat{\varphi}) = f(\varphi),$$

where f is the satisfiability function. Thus, if φ is satisfiable (resp. unsatisfiable), the augmented instance $\widehat{\varphi}$ is also satisfiable (resp. unsatisfiable).

Unit Propagation (UP). If a formula contains a unit clause ℓ , one may set $\ell = 1$ and simplify accordingly by removing all clauses satisfied by ℓ and deleting $\neg\ell$ from all remaining clauses. *Example:* $\varphi = (x_1) \wedge (\neg x_1 \vee x_2) \longrightarrow \varphi' = (x_2)$.

Add Unit Literal (AU). One can introduce a new literal as a unit clause. Furthermore its negation is added to some predetermined number of existing clauses. At last, new random clauses that include the literal are also added. In this work, we examine this augmentation with both random clause addition skipped (notated with AU) and included (notated with AU⁺). *Example (AU⁺):* $\varphi = (x_1 \vee x_2) \longrightarrow \varphi' = (x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1)$.

Pure Literal Elimination (PL). A variable is pure if it occurs only with one polarity in the formula. In this case, all clauses containing the variable can be safely removed. *Example:* $\varphi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \longrightarrow \varphi' = \top$.

Subsumed Clause Elimination (SC). If one clause is a subset of another, the larger clause is redundant and may be deleted. *Example:* $\varphi = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee x_3) \longrightarrow \varphi' = (x_1 \vee x_2)$.

Add Subsumed Clause (SA). Create a new clause subsuming an existing one in the formula. Note that this is a reverse SC operation. *Example:* $\varphi = (x_1 \vee x_2) \longrightarrow \varphi' = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee \neg x_3)$.

Clause Resolution (CR). Given two clauses containing complementary literals, one may combine them into a new resolvent clause implied by both. *Example:* $(x_1 \vee x_2), (\neg x_1 \vee x_3) \longrightarrow \text{resolvent } (x_2 \vee x_3)$.

Variable Elimination (VE). For a variable x , one considers the set of clauses containing x and those containing $\neg x$, and replaces them with all possible resolvents. This effectively removes x from the formula while maintaining satisfiability. *Example:* $\varphi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \longrightarrow \varphi' = (x_2 \vee x_3)$.

These transformations provide effective label-preserving augmentations for SAT. In the context of contrastive learning, resolution-based augmentations such as CR and VE often yield harder positive pairs, which improve representation quality (see more in [14]). In our work, we use the following augmentation parameters: 5% for added literals and clauses in AU and AU⁺; 20% for added resolvents in CR; 10% for added clauses in SA; and 5% to remove an existing variable in VE.

C WL expressivity of label-preserving augmentations

This appendix records the formal statements and proofs behind Theorem 4.1, with the two claims proved separately in Section C.1 and Section C.2. Throughout, G_T denotes the signed typed variable–clause graph of a CNF formula T , with variable and clause node types and signed edges for positive and negative literal occurrences.

Definition C.1 (WL indistinguishability and identification). *The 1-WL test compares two signed typed graphs by running color refinement on their disjoint union, respecting node types and signed edge labels. We write $T \equiv_{\text{WL}} T'$ if the stable color multisets of G_T and $G_{T'}$ agree. A signed typed graph G is identified by 1-WL if 1-WL distinguishes G from every non-isomorphic signed typed graph.*

Definition C.2 (WL-separating LPA). *A label-preserving augmentation A is WL-separating if there exist CNF formulas T, T' such that T is satisfiable, T' is unsatisfiable, $T \equiv_{\text{WL}} T'$, but*

$$A(T) \not\equiv_{\text{WL}} A(T').$$

Definition C.3 (Resolution and variable elimination). *One-step resolution augments a CNF formula by adding all non-tautological resolvents obtainable from one resolution step between two existing clauses. Variable elimination on x_i removes the clauses containing x_i or $\neg x_i$ and adds all non-tautological resolvents between opposite-polarity clauses on x_i . Both transformations preserve satisfiability.*

C.1 WL-separating resolution and variable elimination

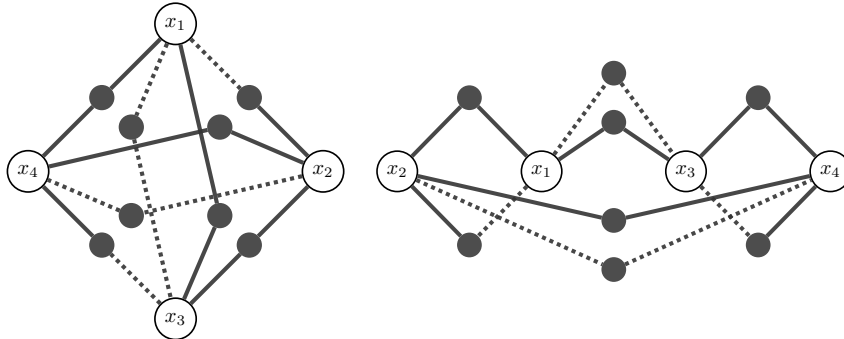


Figure 5: Variable-clause graphs of T (right) and T' (left) for Theorem C.4.

In this section, we show that some label-preserving augmentations are WL-separating. Informally, this implies that they expose structural facts about the formula that were previously unobservable for 1-WL GNNs.

Theorem C.4 (Resolution and variable elimination are WL-separating). *One-step resolution and variable elimination are WL-separating label-preserving augmentations: there exist CNF formulas T and T' such that T is satisfiable, T' is unsatisfiable, G_T and $G_{T'}$ are 1-WL-indistinguishable, but applying the corresponding augmentation yields formulas whose signed typed variable–clause graphs are distinguished by 1-WL. This holds regardless of any randomness in the augmentation.*

Proof. It is enough to exhibit one satisfiable formula and one unsatisfiable formula that are initially indistinguishable by 1-WL, but become distinguishable after the augmentation.

Let

$$T = (\neg x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_3 \vee x_4) \wedge (x_4 \vee x_1) \\ \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_2 \vee \neg x_4),$$

and

$$T' = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (\neg x_3 \vee x_4) \\ \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_2 \vee \neg x_4).$$

The formula T is satisfiable, for instance by $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$. The formula T' is unsatisfiable. Indeed, if $x_1 = 1$, then $\neg x_1 \vee \neg x_3$ forces $x_3 = 0$, then $x_3 \vee x_4$ forces $x_4 = 1$, and then $\neg x_2 \vee \neg x_4$ forces $x_2 = 0$, contradicting $\neg x_1 \vee x_2$. If $x_1 = 0$, then $x_1 \vee x_3$ forces $x_3 = 1$, then $\neg x_3 \vee x_4$ forces $x_4 = 1$, and again $\neg x_2 \vee \neg x_4$ forces $x_2 = 0$, contradicting $x_1 \vee x_2$.

The signed typed variable–clause graphs G_T and $G_{T'}$ (Figure 5) are 1-WL-indistinguishable. In both formulas, color refinement stabilizes with the same variable color classes

$$V_A = \{x_1, x_3\}, \quad V_B = \{x_2, x_4\}.$$

The clause color multiset is also the same in the two graphs: one clause has signed profile $(+V_A, +V_A)$, one has $(-V_A, -V_A)$, one has $(+V_B, +V_B)$, one has $(-V_B, -V_B)$, two have $(+V_A, +V_B)$, and two have $(-V_A, +V_B)$. These profiles define a stable partition, because every variable in V_A and every variable in V_B sees the same signed multiset of clause-profile colors as the other variable in its class. Hence the stable color histograms agree, so $T \equiv_{\text{WL}} T'$.

Now apply one-step resolution. In T' , resolving $(x_1 \vee x_2)$ with $(\neg x_1 \vee x_2)$ produces the unit clause (x_2) , and resolving $(x_3 \vee x_4)$ with $(\neg x_3 \vee x_4)$ produces (x_4) . In T , no one-step resolution produces a unit clause: every non-tautological resolvent from the displayed binary clauses is again binary. Thus the one-step resolution augmentations are distinguished by 1-WL, since the augmented graph for T' contains unit-clause nodes while the augmented graph for T does not. Hence one-step resolution is WL-separating.

Variable elimination is WL-separating using the same pair. Eliminating x_1 from T' includes the resolvent (x_2) from $(x_1 \vee x_2)$ and $(\neg x_1 \vee x_2)$. In T , there is no variable such that eliminating it yields a unit clause (this follows from the same resolvent inspection above). Therefore the variable-elimination augmentations are also distinguished by 1-WL. $\square$

C.2 Identification by random unit-literal augmentation

Definition C.5 (Random unit-literal augmentation). *Let T have original clauses $C_1, \dots, C_m$. For each $i \in [s]$, introduce a fresh variable y_i , add the unit clause (y_i) , and independently for every original clause C_j , add the literal $\neg y_i$ to C_j with probability $p \in (0, 1)$. The resulting formula is denoted $\hat{T}$.*

Theorem C.6 (Random unit-literal augmentation identifies formulas). *Let T be a CNF formula with m clauses and no unit clauses. Form $\hat{T}$ by introducing $s = \lceil \gamma \log m \rceil$ fresh unit variables $y_1, \dots, y_s$, adding the unit clauses (y_i) , and adding $\neg y_i$ independently to each original clause with a fixed probability $p \in (0, 1)$. For a sufficiently large constant $\gamma = \gamma(p)$, the signed typed variable–clause graph $G_{\hat{T}}$ is identified by 1-WL with probability $1 - o(1)$ as $m \rightarrow \infty$.*

Lemma C.7. *Let G_T be a signed typed variable–clause graph. If the stable 1-WL coloring is discrete on the clause nodes, then G_T is identified by 1-WL.*

Proof. Suppose G_T is not identified. Then there is a non-isomorphic signed typed graph H with the same stable WL color multiset. Since clause nodes are initially colored differently from variable nodes, clause colors match only clause colors. Discreteness on the clauses of G_T therefore gives a unique color-preserving bijection from the clause nodes of G_T to the clause nodes of H .

For a variable node v , define its signed clause fingerprint by

$$\text{fp}(v) = (\{\chi^*(c) : c \in N_+(v)\}, \{\chi^*(c) : c \in N_-(v)\}),$$

where χ^* is the stable coloring. Because the clause colors are unique, this fingerprint records exactly which clauses contain v positively and negatively. Stable 1-WL colors determine these fingerprints. Hence any bijection between same-colored variable nodes, together with the unique clause bijection above, preserves signed adjacency. This yields an isomorphism, contradicting the choice of H . $\square$

Lemma C.8. Fix $p \in (0, 1)$, let $D_1, \dots, D_s$ be i.i.d. $\text{Bin}(m, p)$, and let $s = O(\log m)$. Then

$$\Pr(D_1, \dots, D_s \text{ are pairwise distinct}) = 1 - o(1).$$

Proof. For $X, Y \stackrel{\text{i.i.d.}}{\sim} \text{Bin}(m, p)$,

$$\Pr(X = Y) = \sum_k \Pr(X = k)^2 \leq \max_k \Pr(X = k).$$

By Stirling's formula, $\max_k \Pr(X = k) \leq C_p m^{-1/2}$ for a constant C_p depending only on p . A union bound over the $\binom{s}{2}$ pairs gives

$$\Pr(\exists i < j : D_i = D_j) \leq \binom{s}{2} C_p m^{-1/2} = o(1),$$

because $s = O(\log m)$. $\square$

Lemma C.9. Fix $p \in (0, 1)$, and let $s = \lceil \gamma \log m \rceil$. For every original clause C_j , define its unit-literal fingerprint

$$\sigma(C_j) = (\mathbf{1}[\neg y_1 \in C_j], \dots, \mathbf{1}[\neg y_s \in C_j]) \in \{0, 1\}^s.$$

For sufficiently large $\gamma = \gamma(p)$, the fingerprints $\sigma(C_1), \dots, \sigma(C_m)$ are pairwise distinct with probability $1 - o(1)$.

Proof. For two fixed distinct clauses C_j, C_k , the two bits contributed by a fixed fresh variable y_i agree with probability

$$q := p^2 + (1 - p)^2 < 1.$$

The s fresh variables are sampled independently, so $\Pr(\sigma(C_j) = \sigma(C_k)) = q^s$. By a union bound,

$$\Pr(\exists j < k : \sigma(C_j) = \sigma(C_k)) \leq \binom{m}{2} q^s.$$

Choosing $\gamma > 2/(-\log q)$ makes this upper bound $o(1)$. $\square$

Proof of Theorem C.6. Let D_i be the number of original clauses to which $\neg y_i$ was added. Then $D_i \sim \text{Bin}(m, p)$. Since $s = \lceil \gamma \log m \rceil$, Lemma C.8 implies that, with probability $1 - o(1)$, all values D_i are distinct. Also, by increasing γ if needed, Lemma C.9 implies that, with probability $1 - o(1)$, all original clauses have distinct unit-literal fingerprints.

Condition on these two high-probability events. Since T originally has no unit clauses, the newly added unit clauses (y_i) are distinguished from the original clauses by signed color refinement: they are clause nodes of signed degree one, adjacent positively to a fresh variable. Once unit clauses are distinguished, each fresh variable y_i is distinguished from every original variable because it is adjacent to such a unit clause. The fresh variables are also distinguished from one another because y_i has exactly D_i negative edges to original clauses, and the D_i 's are pairwise distinct. Therefore the y_i 's have unique stable WL colors, and then their unit clauses (y_i) also have unique stable colors.

Now consider the original clauses after augmentation. Their neighborhoods include the uniquely colored fresh variables y_i for precisely the indices appearing in their fingerprints $\sigma(C_j)$, all through

negative edges. Because the fingerprints are pairwise distinct, the original clause nodes have pairwise distinct stable WL colors. The new unit clauses also have pairwise distinct stable colors, as argued above, and are distinguished from original clauses. Hence the stable coloring is discrete on all clause nodes of $G_{\hat{T}}$.

By Lemma C.7, $G_{\hat{T}}$ is identified by 1-WL. The failure probability is the sum of the two $o(1)$ error probabilities above, so the identification holds with probability $1 - o(1)$. $\square$

Note on Canonical vs. Non-canonical Labeling.

Theorem C.6 shows that repeated unit literal addition produces a formula which is identified by WL with probability $1 - o(1)$. The mechanism is that random degrees and adjacency patterns act as unique node identifiers. From the perspective of GNNs, this added information exposes structure that message passing could not otherwise access. Even though the identifiers are random instead of canonical, recent work studying identifier-agnostic GNNs [52] shows that they increase expressivity in a generalizable way, as long as the GNN remains invariant to the particular identifiers used. In our setting, label preservation provides a justification for this non-canonical representation: although the mapping from the original formula to the augmented formula is not one-to-one, it still represents the same SAT decision problem. Thus, a robust neural SAT solver may exploit the additional structure while still making consistent predictions across equivalent augmentations, even in cases where message passing on the original VCG lacks the expressivity to do so [44].

D Effects of label-preserving augmentations

This appendix records how several label-preserving augmentations change the multilinear objective, the exact product-measure unsatisfiability probability, and the smoothness parameters used in Proposition 3.2. We keep the notation of Section 2 and Eq. (1): a CNF instance is $T = (V, \Phi)$, with $\Phi = C_1 \wedge \dots \wedge C_m$, clause violation probabilities $q_j(p)$, and objective $\mathcal{L}_T(p) = \sum_j q_j(p)$. We write

$$P_\Phi(p) := \Pr_{X \sim \mu_p} [\Phi(X) = 0], \quad S_\Phi(p) := 1 - P_\Phi(p),$$

and define the union-bound gap

$$\Gamma_T(p) := \mathcal{L}_T(p) - P_\Phi(p) \geq 0.$$

For any displayed clause D , $q_D(p)$ denotes the same violation monomial as the main-text clause terms $q_j(p)$. To track smoothness more precisely, define

$$d_T(i) := |\{j : i \in A(j)\}|, \quad \lambda_T(i) := \sum_{j: i \in A(j)} (a(C_j) - 1),$$

and

$$\Delta(T) := \max_i d_T(i), \quad \Lambda(T) := \max_i \lambda_T(i).$$

The proof of Proposition 3.2 shows the sharper bound $\|\nabla^2 \mathcal{L}_T(p)\|_2 \leq \Lambda(T)$, and $\Lambda(T) \leq (w - 1)\Delta(T)$ for width at most w .

D.1 Size-increasing transformations

Fresh unit-clause augmentation. Let u be a fresh variable, and let $J \subseteq [m]$ be a set of clauses to modify. The augmented instance $\hat{T} = (\hat{V}, \hat{\Phi})$ is obtained by adding the unit clause (u) , replacing C_j by $C_j \vee \neg u$ for $j \in J$, and leaving the other clauses unchanged. Write

$$Q_J(p) := \sum_{j \in J} q_j(p),$$

and let p_u be the Bernoulli marginal of the fresh variable.

Proposition D.1. *The augmented formula satisfies*

$$S_{\hat{\Phi}}(p, p_u) = p_u S_\Phi(p), \quad P_{\hat{\Phi}}(p, p_u) = 1 - p_u S_\Phi(p),$$

and

$$\mathcal{L}_{\hat{T}}(p, p_u) = \mathcal{L}_T(p) + (1 - p_u)(1 - Q_J(p)).$$

Consequently,

$$\Gamma_{\hat{T}}(p, p_u) - \Gamma_T(p) = (1 - p_u)(P_\Phi(p) - Q_J(p)).$$

Proof. For $j \in J$, $(u) \wedge (C_j \vee \neg u)$ is equivalent to $(u) \wedge C_j$, so $\widehat{\Phi}$ is equivalent to $u \wedge \Phi$. This gives the exact satisfiability identity. The new unit clause contributes $1 - p_u$ to the objective, while a modified clause $C_j \vee \neg u$ is violated with probability $p_u q_j(p)$. Summing the unchanged and modified terms gives the objective identity, and subtracting the exact unsatisfiability probabilities gives the gap identity. $\square$

Thus the transformed formula's own gap can move in either direction. However, if the lower bound is transported back to the original formula using $S_\Phi = p_u^{-1} S_{\widehat{\Phi}}$, then

$$p_u^{-1}(1 - \mathcal{L}_{\widehat{T}}) - (1 - \mathcal{L}_T) = -\frac{1 - p_u}{p_u} \sum_{j \notin J} q_j(p) \leq 0 \quad (p_u > 0),$$

so this augmentation never improves the transported upper bound on the original formula's unsatisfiability probability.

Let $s_i := |\{j \in J : i \in A(j)\}|$. For the original variables,

$$d_{\widehat{T}}(i) = d_T(i), \quad \lambda_{\widehat{T}}(i) = \lambda_T(i) + s_i,$$

while the fresh variable satisfies

$$d_{\widehat{T}}(u) = |J| + 1, \quad \lambda_{\widehat{T}}(u) = \sum_{j \in J} a(C_j).$$

Hence the unnormalized smoothness bound typically increases, and the normalized bound can move either way because both the numerator and the clause count have changed.

Subsumed clause addition. Suppose D is a non-tautological clause subsumed by an existing clause, i.e., some C_a contains only literals already contained in D . Let $\widehat{T}$ be obtained by adding D while keeping all original clauses. Then $\widehat{\Phi}$ is logically equivalent to Φ , so

$$S_{\widehat{\Phi}} = S_\Phi, \quad P_{\widehat{\Phi}} = P_\Phi,$$

but

$$\mathcal{L}_{\widehat{T}} = \mathcal{L}_T + q_D, \quad \Gamma_{\widehat{T}} = \Gamma_T + q_D.$$

Thus the transported unsat bound becomes looser, except at points where $q_D = 0$. Moreover,

$$\Delta(\widehat{T}) \leq \Delta(T) + 1, \quad \Lambda(\widehat{T}) \leq \Lambda(T) + a(D) - 1,$$

so the unnormalized smoothness bound can only worsen or stay fixed.

Resolution addition with parents kept. Let $C_a = (u \vee A)$ and $C_b = (\neg u \vee B)$, and assume the resolvent $R = A \vee B$ is non-tautological. If $\widehat{T}$ is obtained by keeping the parents and adding R , then $C_a \wedge C_b \Rightarrow R$, so the Boolean function is unchanged:

$$S_{\widehat{\Phi}} = S_\Phi, \quad P_{\widehat{\Phi}} = P_\Phi.$$

The objective and gap change by

$$\mathcal{L}_{\widehat{T}} = \mathcal{L}_T + q_R, \quad \Gamma_{\widehat{T}} = \Gamma_T + q_R.$$

The transported unsat bound therefore becomes looser or equal. The structural smoothness parameters obey

$$\Delta(\widehat{T}) \leq \Delta(T) + 1, \quad \Lambda(\widehat{T}) \leq \Lambda(T) + a(R) - 1.$$

D.2 Size-decreasing transformations

Unit propagation. Assume T contains the unit clause (u) . Apart from this unit clause, partition the clauses containing u as

$$S_+ = \{u \vee A_j\}_j, \quad S_- = \{\neg u \vee B_k\}_k,$$

and let R denote the clauses that do not mention u . Unit propagation produces T' with clauses $R \cup \{B_k\}_k$. Define

$$Q_+(p) := \sum_j q_{A_j}(p), \quad Q_-(p) := \sum_k q_{B_k}(p), \quad \mathcal{L}_R(p) := \sum_{C \in R} q_C(p).$$

Then

$$S_{\Phi}(p, p_u) = p_u S_{\Phi'}(p),$$

and

$$\mathcal{L}_T(p, p_u) = (1 - p_u)(1 + Q_+(p)) + p_u Q_-(p) + \mathcal{L}_R(p), \quad \mathcal{L}_{T'}(p) = Q_-(p) + \mathcal{L}_R(p).$$

Consequently,

$$\mathcal{L}_{T'} - \mathcal{L}_T = (1 - p_u)(Q_- - 1 - Q_+).$$

Although the transformed formula's own gap need not change monotonically, the transported upper bound on the original formula's unsatisfiability probability is always tighter or equal:

$$p_u(1 - \mathcal{L}_{T'}) - (1 - \mathcal{L}_T) = (1 - p_u)(Q_+ + \mathcal{L}_R) \geq 0.$$

All surviving clauses are either unchanged or shortened, and some clauses are removed. Therefore, for every surviving variable $i \neq u$,

$$d_{T'}(i) \leq d_T(i), \quad \lambda_{T'}(i) \leq \lambda_T(i),$$

so $\Delta(T') \leq \Delta(T)$ and $\Lambda(T') \leq \Lambda(T)$.

Subsumed clause elimination. If $C_a \subseteq C_b$, deleting C_b gives an equivalent formula T' . Thus

$$S_{\Phi'} = S_{\Phi}, \quad P_{\Phi'} = P_{\Phi},$$

while

$$\mathcal{L}_{T'} = \mathcal{L}_T - q_{C_b}, \quad \Gamma_{T'} = \Gamma_T - q_{C_b}.$$

The transported unsat bound becomes tighter or equal. The same deletion decreases the structural smoothness parameters:

$$\Delta(T') \leq \Delta(T), \quad \Lambda(T') \leq \Lambda(T).$$

Pure literal elimination. Assume u is pure positive and appears in clauses

$$u \vee D_1, \dots, u \vee D_t.$$

Let T' be obtained by deleting these clauses, and let

$$H := T' \cup \{D_1, \dots, D_t\}$$

be the formula obtained by setting $u = 0$. Then

$$S_{\Phi}(p, p_u) = p_u S_{\Phi'}(p) + (1 - p_u) S_H(p), \quad S_H(p) \leq S_{\Phi'}(p).$$

Writing $Q_D(p) := \sum_{r=1}^t q_{D_r}(p)$, the objective relation is

$$\mathcal{L}_T(p, p_u) = \mathcal{L}_{T'}(p) + (1 - p_u) Q_D(p),$$

so

$$\mathcal{L}_{T'} - \mathcal{L}_T = -(1 - p_u) Q_D \leq 0.$$

The gap satisfies

$$\Gamma_{T'} - \Gamma_T = (1 - p_u)((S_{\Phi'} - S_H) - Q_D) \leq 0,$$

because $S_{\Phi'} - S_H \leq Q_D$ for the deleted residual clauses. Thus pure literal elimination tightens the transported unsat bound or leaves it unchanged. It also removes clauses and literal occurrences, so

$$\Delta(T') \leq \Delta(T), \quad \Lambda(T') \leq \Lambda(T).$$

Across the size-decreasing transformations above, the unnormalized smoothness bound improves or stays fixed. For normalized objectives, monotonicity is not guaranteed because the clause count changes at the same time as the row-sum parameter.

Table 1 summarizes the qualitative trend from these calculations using the transported-bound convention.

The compressed entries in Table 1 use the following convention. The codes are those of Appendix B. The ‘‘Objective’’ column refers to the unnormalized multilinear objective $\mathcal{L}_T$, and the ‘‘Smoothness’’ column refers to the unnormalized structural smoothness bound $\Lambda(T)$, the sharper row-sum version of Eq. (2). The arrows denote qualitative movement of the named quantity: $\uparrow$ means increase, $\downarrow$

means decrease, = means unchanged, and – means ambiguous. For the “Tightness” column, $\uparrow$ means that the transported upper bound on the original formula’s unsatisfiability probability becomes tighter; for the “Smoothness” column, $\downarrow$ means a smaller curvature/smoothness constant.

The removed assignment-map column distinguished two cases. Same-variable transformations such as subsumed clause addition, resolution addition, and subsumed clause elimination preserve the original assignments exactly. Transformations that introduce or eliminate variables are compared through a unique-extension or projection identity. Thus, for unit injection, unit propagation, and pure-literal elimination, the $\Pr[\Phi(X) = 0]$ column describes the transformed instance, while the original formula’s probability is recovered by the corresponding assignment map described in the proofs above. This is why the transported-bound column is the quantity that is most directly comparable across all rows: for same-variable transformations it coincides with the ordinary multilinear upper bound, whereas for variable introducing or eliminating transformations it accounts for the map back to the original formula. Several arrows also hide equality cases, for example when an added or deleted clause term vanishes at the current marginals.

E Detailed experimental protocols and results

E.1 Certified-basin boundary diagnostics

We first isolate the certified-basin theorem as a boundary diagnostic, separate from augmentation. The goal of this experiment is to compare the sign and size of the certified-basin margin with the empirical projected-GD basin on two fixed symmetric formulas: the parity-one XOR cycle and the exactly-one CNF, both with $n = 50$. All runs in this diagnostic use projected GD directly on the marginal vector p , the exact multilinear objective on the unaugmented CNF clause list, exact analytic gradients in double precision, CPU execution, projection onto $[0, 1]^{50}$ after every update, learning rate $\eta = 1$, and 20,000 full-batch steps. We plot only the certified-basin margin. A point is colored blue when the final continuous objective is zero and deterministic threshold rounding $p_i \geq 1/2$ satisfies the formula, and red otherwise.

For the XOR-cycle experiment, the formula is

$$\bigwedge_{i=1}^{50} [(x_i \vee x_{i+1}) \wedge (\neg x_i \vee \neg x_{i+1})], \quad x_{51} = x_1.$$

We target an alternating satisfying assignment, initialize all variables true under that target to p_T^0 , and initialize all variables false under that target to p_F^0 . For this two-level family, direct differentiation gives the certified-basin margin

$$\min\{2p_T^0 - 1, 1 - 2p_F^0\}.$$

The complementary alternating assignment has the same expression with the two classes swapped; the top-left panel of Figure 2 reports the better of the two margins. Positive margin means one alternating solution is certified by the condition, zero is the boundary, and negative margin violates this two-level certified-basin check.

For the exactly-one experiment, the formula is

$$(x_1 \vee \dots \vee x_{50}) \wedge \bigwedge_{1 \leq i < j \leq 50} (\neg x_i \vee \neg x_j).$$

We fix target coordinate $r = 1$, initialize $p_r^0 = 0.51$, and set all other coordinates to a common value q . The certified-basin margin used in the bottom-left panel of Figure 2 is

$$\min \left\{ 2p_r^0 - 1, \prod_{i \neq r} (1 - p_i^0) - \sum_{i \neq r} p_i^0 \right\}.$$

For this one-parameter slice, the certified-basin boundary is $q_\star = 0.0115500425716365$, where $(1 - q_\star)^{49} = 49q_\star$. The scatter uses the same grid as the XOR-cycle experiment: four positive-margin initializations, one boundary row, and four negative-margin initializations.

Table 3: Symmetry and augmentation sweep on the exactly-one family. Entries are final original EO multilinear objective values for GNN and PGD variants across n . The GNN row is the unaugmented degree-feature message-passing baseline. Both augmented rows use only standard unit-literal append augmentation, which adds marker units (y_a) and appends $\neg y_a$ to existing EO clauses. The GNN+AU row uses $s = \lceil 0.1m \rceil$ markers and $k = 5\lceil n/20 \rceil$ decorated clauses per marker. The PGD+AU rows use $s = \lceil 0.1m \rceil$ markers and $k = \lceil 0.1m \rceil$ decorated clauses per marker. PGD rows use 20,000 projected-gradient steps with $\eta = 1$; GNN+AU uses 10,000 Adam steps.

Setting	Method	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
<i>No augmentation</i>	GNN	0.700238	0.714765	0.719304	0.721521	0.722834
	PGD	45	190	435	780	1225
<i>Unit-literal augmentation</i>	GNN+AU	9.29×10^{-7}	2.84×10^{-4}	1.03×10^{-5}	5.62×10^{-5}	2.43×10^{-5}
	PGD+AU	45	1	1	1	1
<i>Certified initialization</i>	PGD+cert. init.	0	0	0	0	0
	PGD+AU+cert. init.	0	0	0	0	0

E.2 Symmetry and augmentation on exactly-one

Table 3 is a separate experiment from the certified-basin margin diagnostic above. Here the goal is to test how exact symmetry and unit-literal augmentation affect optimization on the exactly-one family as n varies. We write EO for the unaugmented exactly-one CNF, which has $m = 1 + \binom{n}{2}$ clauses, and EO+AU for the standard unit-literal append augmentation. In EO+AU we add marker variables $y_1, \dots, y_s$, add the unit clauses (y_a), and decorate existing EO clauses by appending negative marker literals $\neg y_a$. This is the only augmentation used in the EO+AU rows of Table 3. The Boolean formula is satisfiability-preserving under unit propagation, but the continuous objective optimized in the augmented rows is changed.

The PGD rows probe two initialization regimes: a fully symmetric start and a certified asymmetric start. Each PGD run optimizes the exact multilinear objective of its input formula using analytic gradients in double precision on CPU, projection to $[0, 1]^d$ after every step, learning rate $\eta = 1$, and 20,000 full-batch steps. The PGD+AU rows use $s = \lceil 0.1m \rceil$ markers, and each marker decorates exactly $\lceil 0.1m \rceil$ original EO clauses chosen uniformly without replacement. The symmetric initialization sets every input marginal, including markers, to $1/2$. The certified initialization sets $p_r^0 = 0.51$, $p_i^0 = 0.01$ for $i \neq r$, and initializes each marker at 0.51, toward the truth value enforced by its unit clause. This certified initialization satisfies $(1 - 0.01)^{n-1} > (n - 1)0.01$ for every n in the sweep. All PGD objective values reported in the table are final original EO objective values, even when the optimized objective is the augmented one.

The GNN rows test the architectural symmetry limitation and the effect of AU on that limitation. The unaugmented GNN uses signed message passing without node identifiers, so it is constrained by 1-WL color classes of the input graph. It uses degree and signed-degree features, signed sum aggregation, ReLU MLP updates, a sigmoid output head, Adam with learning rate 10^{-4} , and 2,000 full-batch steps on the original EO objective. For GNN+AU, the input and training objective are both augmented. We use normalized degree/count features, signed mean aggregation, ReLU MLP updates, a sigmoid output head over all input variables including markers, Adam with learning rate 10^{-4} , and 10,000 full-batch steps. Its augmentation schedule keeps $s = \lceil 0.1m \rceil$ markers but uses sparser wiring: each marker decorates $k = 5\lceil n/20 \rceil$ original clauses. The GNN+AU objective values are again evaluated only on the original EO clauses.

The sweep confirms the qualitative behavior seen at $n = 50$. With a large step size, symmetric projected GD does not recover the one-hot solution: in EO the even-time endpoint is the all-ones assignment, while the standard EO+AU symmetric runs for $n \geq 20$ end with all original variables at zero, leaving only the large positive clause unsatisfied. Thus standard append AU by itself does not make symmetric PGD solve these instances. The certified initialization converges to zero original objective value in every EO and EO+AU run. The unaugmented GNN remains trapped in the 1-WL symmetry class of the original EO formula, whereas scheduled append AU breaks the architectural color-class tie strongly enough for the GNN to enter a one-hot basin and drive the original EO objective close to zero. This is an optimization observation, not a guarantee over all random augmentations or initializations.

Table 4: Held-out unaugmented G4SATBench comparison aggregated by family. Rows correspond to initialization, projection rule, and optimizer. Each entry is the mean $\pm$ standard deviation of the final original multilinear-extension objective over all 84 formulas from the corresponding family, pooled across the three difficulty levels, after 20,000 full-batch gradient steps. Lower is better.

Method	sr	3-sat	ca	ps	k-clique	k-domset	k-vercov
Symmetric / Sigmoid / GD	4.87 $\pm$ 4.30	4.96 $\pm$ 3.90	11.72 $\pm$ 16.79	8.85 $\pm$ 8.98	1.02 $\pm$ 0.63	1.20 $\pm$ 0.76	1.74 $\pm$ 0.93
Symmetric / Sigmoid / Adam	5.55 $\pm$ 5.25	5.52 $\pm$ 4.24	12.86 $\pm$ 17.93	9.90 $\pm$ 9.88	1.01 $\pm$ 0.61	0.95 $\pm$ 0.67	1.51 $\pm$ 0.94
Symmetric / Projection / GD	5.14 $\pm$ 4.64	5.43 $\pm$ 4.21	12.21 $\pm$ 17.39	9.42 $\pm$ 9.64	29.04 $\pm$ 22.18	1.62 $\pm$ 1.02	3.98 $\pm$ 12.32
Symmetric / Projection / Adam	6.79 $\pm$ 6.37	7.08 $\pm$ 5.49	14.26 $\pm$ 19.44	11.10 $\pm$ 10.56	0.87 $\pm$ 0.58	1.21 $\pm$ 0.70	1.93 $\pm$ 1.06
Gaussian / Sigmoid / GD	6.56 $\pm$ 5.55	7.07 $\pm$ 5.47	13.90 $\pm$ 19.12	10.96 $\pm$ 10.31	1.09 $\pm$ 0.62	1.68 $\pm$ 0.88	2.33 $\pm$ 1.04
Gaussian / Sigmoid / Adam	6.71 $\pm$ 5.81	6.39 $\pm$ 4.82	14.29 $\pm$ 19.66	11.21 $\pm$ 10.77	1.12 $\pm$ 0.63	1.57 $\pm$ 0.85	1.85 $\pm$ 1.09
Gaussian / Projection / GD	7.01 $\pm$ 5.91	7.12 $\pm$ 5.35	14.71 $\pm$ 20.54	11.54 $\pm$ 10.86	29.06 $\pm$ 22.16	1.85 $\pm$ 0.78	4.28 $\pm$ 12.18
Gaussian / Projection / Adam	8.04 $\pm$ 6.97	7.87 $\pm$ 5.88	15.44 $\pm$ 21.21	12.58 $\pm$ 11.81	0.94 $\pm$ 0.55	1.57 $\pm$ 0.78	1.93 $\pm$ 1.06

E.3 Held-out G4SATBench optimization comparison

We next report a controlled study on the public G4SATBench benchmark [32]. The purpose of this experiment is not yet to test augmentations, but to establish a clean optimization baseline for the unaugmented multilinear objective and to isolate the effect of three design choices: initialization, parameterization, and optimizer.

Protocol. We use a held-out subset of 588 formulas: 7 families, 3 difficulty levels, and 28 formulas per family/difficulty cell. The families are *sr*, *3-sat*, *ca*, *ps*, *k-clique*, *k-domset*, and *k-vercov*. Each run optimizes only the original multilinear objective in Eq. (1), with no augmentation clauses or selector variables. We compare two initializations, two parameterizations, and two optimizers. The *symmetric* initialization sets all logits to zero, so all marginals start at 0.5. The *Gaussian* initialization draws $h \sim \mathcal{N}(0, I)$ with a deterministic per-formula seed; under the sigmoid parameterization we set $z^{(0)} = h$, while under the projected parameterization we set $p^{(0)} = \sigma(h)$. The parameterization is either *sigmoid*, which optimizes unconstrained logits with $p_i = \sigma(z_i)$, or *projection*, which optimizes marginals directly and projects them back to $[0, 1]$ after each step. The optimizer is either vanilla gradient descent or Adam. All runs use learning rate 0.1, 20,000 deterministic full-batch steps, CPU execution, and `torch.float64`. Table 4 aggregates over difficulty and reports family-level means and standard deviations over the 84 formulas from each family.

Results. Table 4 reports the mean and standard deviation of the final multilinear objective for each method-family pair. The main pattern is that symmetric initialization consistently outperforms Gaussian initialization. Within the symmetric runs, sigmoid parameterization with vanilla GD is best on *sr*, *3-sat*, *ca*, and *ps*. The graph-derived families differ somewhat: *k-clique* is best under symmetric projection with Adam, while *k-domset* and *k-vercov* are best under symmetric sigmoid with Adam. We use this experiment as the optimization baseline that later augmentation experiments should improve upon.

E.4 Held-out G4SATBench augmentation ablation

We next revisit the same held-out G4SATBench subset to test whether label-preserving transformations improve exact full-batch optimization of the multilinear objective. Unlike the previous subsection, which selected an optimization baseline, here we fix the benchmark and ask whether optimizing a transformed formula yields a lower final objective on the *original* formula.

Protocol. We use the same 588 held-out formulas as in the unaugmented comparison above: 7 families, 3 difficulty levels, and 28 formulas per family/difficulty cell. Every run starts from the symmetric initialization $p_i^{(0)} = 0.5$ for all original variables. We compare two exact full-batch optimization rules. The *sigmoid* variant optimizes logits $z \in \mathbb{R}^n$ with $p_i = \sigma(z_i)$, whereas the *projected* variant optimizes marginals directly and applies coordinatewise projection back to $[0, 1]$ after each step. Both use learning rate 0.1, 20,000 deterministic full-batch steps, CPU execution, and exact clausewise objective and gradient evaluation in double precision. The baseline row uses the original CNF without augmentation.

We compare three label-preserving transformations. AU is the append-only unit-literal transform without extra random clauses: it adds $s = \lceil 0.05n \rceil$ fresh marker variables, rewires exactly $\lceil 0.05m \rceil$ original clauses per marker by appending $\neg y_a$, and adds the positive unit clause (y_a) . SA samples $\lceil 0.1m \rceil$ original clauses and adds one non-tautological literal to each selected clause, thereby cre-

Table 5: Held-out G4SATBench augmentation summary on the 588-formula subset. Each entry reports the mean $\pm$ standard deviation of the final original multilinear-extension objective after 20,000 exact full-batch steps. The “Mean Δ ” column reports the mean paired difference against the corresponding unaugmented baseline under the same optimization rule, and “Win %” is the percentage of the 21 family/difficulty cells whose mean original objective is no larger than that baseline. Lower objective values and higher win percentage are better.

Method	Mean final original objective	Mean Δ vs same baseline	Win %
Sigmoid GD / Baseline	4.91 $\pm$ 8.41	—	—
Sigmoid GD / AU	19.15 $\pm$ 31.57	+14.24	4.8
Sigmoid GD / CR	5.85 $\pm$ 8.46	+0.94	0.0
Sigmoid GD / SA	5.09 $\pm$ 8.64	+0.18	38.1
Projected GD / Baseline	9.55 $\pm$ 15.06	—	—
Projected GD / AU	23.31 $\pm$ 32.46	+13.76	23.8
Projected GD / CR	6.38 $\pm$ 8.78	−3.17	33.3
Projected GD / SA	10.64 $\pm$ 17.94	+1.09	38.1

Table 6: Mean final original multilinear-extension objective on the same held-out G4SATBench subset, grouped by difficulty and family, after 20,000 exact full-batch steps. Rows compare the unaugmented baseline against three label-preserving augmentations under sigmoid GD and projected GD. All reported values are evaluated on the original clause list, not on the transformed objective used during optimization. Lower is better.

Method	Easy Datasets						Medium Datasets						Hard Datasets								
	SR	3-SAT	CA	PS	k-Clique	k-Domset	k-Vercov	SR	3-SAT	CA	PS	k-Clique	k-Domset	k-Vercov	SR	3-SAT	CA	PS	k-Clique	k-Domset	k-Vercov
Sigmoid GD / Baseline	1.12	1.19	3.37	1.05	0.69	0.95	1.34	3.26	4.55	11.41	10.51	1.19	1.15	1.81	10.24	9.14	20.40	15.00	1.18	1.49	2.07
Sigmoid GD / AU	1.30	1.26	3.58	1.33	0.69	1.49	1.90	11.47	16.23	22.17	23.25	1.19	3.33	5.12	74.12	56.17	99.25	60.27	1.36	6.81	9.91
Sigmoid GD / CR	1.19	1.36	3.69	1.12	1.75	1.23	1.84	3.29	4.80	11.73	10.76	3.90	1.92	4.07	10.34	9.60	21.12	16.07	4.71	2.62	5.81
Sigmoid GD / SA	1.15	1.19	3.40	1.08	0.69	0.88	1.45	3.15	4.91	11.19	10.37	1.12	1.29	2.06	10.20	9.75	21.17	16.46	1.60	1.70	2.13
Projected GD / Baseline	1.04	1.36	3.57	1.25	5.10	1.07	1.39	3.46	5.04	11.68	10.89	28.82	1.71	1.79	10.93	9.89	21.39	16.11	53.18	2.07	8.75
Projected GD / AU	1.35	1.41	3.47	1.18	4.23	1.84	2.53	12.46	17.96	23.37	24.66	21.54	4.30	6.90	79.18	60.62	104.23	64.32	34.64	7.52	11.72
Projected GD / CR	1.18	1.32	3.74	1.11	1.98	1.32	2.13	3.54	4.89	12.03	10.77	4.32	2.14	5.89	11.35	10.10	21.69	16.89	4.75	2.64	10.12
Projected GD / SA	1.25	1.29	3.57	1.18	5.06	1.11	1.36	3.64	5.18	11.59	10.84	29.88	1.75	1.71	11.00	10.29	21.96	17.21	60.97	2.11	20.40

ating a weaker subsumed clause while keeping the parent clause. CR adds up to $\lceil 0.2m \rceil$ distinct non-tautological resolvents while keeping the parent clauses. All random augmentation choices use deterministic per-instance seeds. In augmented runs, optimization is performed on the transformed formula, but all reported objective values are evaluated on the original clause list.

Results. Table 5 shows that the best overall method remains unaugmented sigmoid GD. The clearest negative result is the append-only unit-literal transform: its mean objective value is much larger than baseline under both optimization rules, and it matches or improves only 4.8% of family/difficulty cells under sigmoid GD and 23.8% under projected GD. Under sigmoid GD, clause resolution matches or improves 0.0% of cells, while subsumed-clause addition reaches 38.1%. The projected comparison is more mixed: clause resolution improves the *overall* mean relative to the projected baseline, largely because it stabilizes the `k-clique` cells where the projected baseline is poor, but it matches or improves only 33.3% of cells. Overall, on this exact held-out optimization benchmark, the tested label-preserving augmentations do not improve the strongest unaugmented baseline and often increase the final objective on the original formula.

E.5 Exact unsatisfiability probability versus expected violated clauses

We next compare the multilinear extension in Eq. (1) against the true product-measure unsatisfiability probability on a separate exact-evaluable benchmark. For clause-unsat events $A_1, \dots, A_m$, the exact objective is

$$P_{\text{unsat}}(\mathbf{p}) = \Pr_{\mathbf{x} \sim \mathbf{p}} \left[\bigcup_{a=1}^m A_a \right],$$

which we compute exactly by the full inclusion-exclusion expansion over all nonempty clause subsets. The surrogate is

$$E[\#\text{unsat}](\mathbf{p}) = \sum_{a=1}^m \Pr_{\mathbf{x} \sim \mathbf{p}} [A_a] = \mathcal{L}_T(\mathbf{p}),$$

the expected number of violated clauses already used above.

Table 7: Family-level comparison between optimizing the exact unsat probability P_{unsat} and optimizing the expected number of violated clauses $E[\#\text{unsat}]$ on the 420 exact-evaluable formulas. Each entry is the mean over the 60 formulas from the corresponding family. In both panels, lower is better for P_{unsat} and $E[\#\text{unsat}]$, while higher is better for the rounded satisfaction rate.

Panel A: Optimize P_{unsat}						
Family	GD			Adam		
	P_{unsat}	$E[\#\text{unsat}]$	Rounded sat.	P_{unsat}	$E[\#\text{unsat}]$	Rounded sat.
3-SAT	0.035	0.059	0.967	0.033	0.056	0.967
CA	0.044	0.044	0.917	0.033	0.033	0.933
k -Clique	0.004	0.005	1.000	0.000	0.000	1.000
k -Domset	0.483	1.905	0.517	0.417	1.754	0.583
k -Vercov	0.884	3.897	0.117	0.883	3.896	0.117
PS	0.002	0.002	1.000	0.000	0.000	1.000
SR	0.501	1.554	0.500	0.500	1.552	0.500

Panel B: Optimize $E[\#\text{unsat}]$						
Family	GD			Adam		
	P_{unsat}	$E[\#\text{unsat}]$	Rounded sat.	P_{unsat}	$E[\#\text{unsat}]$	Rounded sat.
3-SAT	0.035	0.035	0.967	0.033	0.033	0.967
CA	0.044	0.044	0.917	0.050	0.050	0.900
k -Clique	0.127	0.128	0.750	0.242	0.242	0.517
k -Domset	0.684	0.784	0.317	0.683	0.781	0.317
k -Vercov	0.884	1.020	0.117	0.883	1.017	0.117
PS	0.001	0.002	1.000	0.017	0.017	0.983
SR	0.517	0.519	0.483	0.550	0.567	0.450

Protocol. Because exact inclusion-exclusion is exponential in the number of clauses, we construct a dedicated small benchmark with 420 formulas: the same seven G4SATBench families, the same three difficulty levels, and 20 newly sampled instances per family/difficulty cell, with rejection sampling to keep the exact representation tractable. Every run uses the same Bernoulli product distribution with logits $z \in \mathbb{R}^n$ and marginals $p_i = \sigma(z_i)$. We compare two objectives, P_{unsat} and $E[\#\text{unsat}]$, each optimized with either vanilla GD or Adam. All runs start from $z = 0$, use learning rate 0.1, run for 20,000 deterministic full-batch steps in double precision, and use exact analytic gradients. For P_{unsat} we evaluate the complete inclusion-exclusion formula, with no truncation, no Monte Carlo estimation, and no SAT-solver shortcut. After optimization, we also form a discrete assignment by rounding each marginal with the rule $p_i \geq 0.5 \mapsto 1$ and $p_i < 0.5 \mapsto 0$, and report the fraction of instances whose rounded assignment satisfies the original CNF.

Results. Across all 420 formulas, optimizing P_{unsat} gives higher rounded satisfaction rates than optimizing $E[\#\text{unsat}]$ under both optimizers: $301/420 = 0.717$ for GD and $306/420 = 0.729$ for Adam, versus $273/420 = 0.650$ for GD and $255/420 = 0.607$ for Adam when optimizing $E[\#\text{unsat}]$. Table 7 reports the family-level means. The largest gap appears on the graph-derived families k -clique and k -domset, where directly optimizing P_{unsat} substantially lowers the true unsat probability and improves the rounded satisfaction rate. By contrast, the two objectives are nearly identical on 3-sat and k -vercov. Adam further widens the advantage of the exact objective on the rounded metric.

E.6 Benefits of using augmentations for neural baselines

In this section, we report the influence of using augmentation in the setting of neural architectures. In particular, we take a look at the two following scenarios: (i) using augmentations to improve performance at inference time and (ii) using augmentations as a method for generating artificial data samples to improve training performance. To ensure the robustness of our observations, we look at two different GNN architectures: a lightweight GNN_{Base} consisting of several GINE layers and a $\text{GNN}_{\text{WalkSAT}}$ that is inspired by the WalkSAT algorithm and improves the current solution iteratively.

Table 8: Average number of UNSAT clauses ($\downarrow$) depending on the number of additional executions τ . The numbers are reported on a 100-instance test set after the discretization procedure.

Method	Easy Datasets							Medium Datasets						
	SR	3-SAT	CA	PS	k-Clique	k-Domset	k-Vercov	SR	3-SAT	CA	PS	k-Clique	k-Domset	k-Vercov
WalkSAT ($\tau = 128$)	0.50	0.60	1.80	0.80	0.51	0.53	0.60	0.53	0.59	9.07	4.96	0.51	0.60	0.74
GNN_{WalkSAT}														
Restarts ($\tau = 128$)	0.57	0.78	2.10	0.84	0.51	0.53	0.61	1.15	1.13	8.78	5.06	0.55	0.61	0.84
Aug _{AU} ⁺ ($\tau = 128$)	0.55	0.67	1.94	0.85	0.51	0.53	0.61	0.84	0.86	8.06	4.53	0.69	0.62	0.95
Aug _{CR} ($\tau = 128$)	0.55	0.65	1.86	0.88	0.60	0.53	0.65	0.97	0.76	7.92	4.83	0.70	0.61	0.92
Aug _{SA} ($\tau = 128$)	0.55	0.65	1.93	0.87	0.51	0.53	0.60	0.93	0.82	7.98	4.63	0.52	0.60	0.77
Aug _{RN} ($\tau = 128$)	0.54	0.65	1.91	0.85	0.51	0.53	0.60	0.98	0.93	8.13	4.52	0.52	0.60	0.80
GNN_{Base}														
Restarts ($\tau = 128$)	0.85	0.90	2.13	1.09	0.52	0.67	0.84	4.55	4.43	12.02	8.00	0.54	0.93	1.44
Aug _{AU} ⁺ ($\tau = 128$)	0.82	0.79	1.98	1.10	0.90	0.64	1.69	4.38	4.61	11.24	7.00	1.22	1.01	1.21
Aug _{CR} ($\tau = 128$)	0.77	0.84	1.97	1.06	1.01	0.96	1.84	4.55	4.08	11.25	7.18	1.26	1.65	1.41
Aug _{SA} ($\tau = 128$)	0.81	0.77	1.98	1.02	0.51	0.61	0.98	4.46	4.08	11.23	7.24	0.58	0.70	1.01
Aug _{RN} ($\tau = 128$)	0.80	0.79	2.01	1.06	0.62	0.68	1.52	4.45	4.03	11.31	7.36	0.53	0.92	0.97

See Appendix F for detailed architecture descriptions. All model training here is executed using the multilinear loss extension from Eq. (1) and a learning rate that decreases from 10^{-3} to 10^{-5} during training. All experiments in this subsection were executed on a single Nvidia A100/H100 GPU.

Inference-time augmentations. For evaluation of the inference methods, we first train both of the models for each dataset using the SSL approach. We use 1000 instances in our training set and 100 instances in our test set. The gist of our neural methods is as follows: (i) Restarts uses a random initial assignment from $\{0, 1\}^N$ where N is the number of variables and (ii) Aug uses random initial assignments plus the structural perturbation of the graph obtained using the selected augmentation. We report the results on both label-preserving augmentations (see Appendix B for augmentation definitions) or non-label-preserving (denoted as Aug_{RN}), in which case the original formula is extended via adding 10% of the random existing literals to random existing clauses. We further report the results on WalkSAT as the baseline.

In Table 8 we report the empirical results of each method on *easy* and *medium* datasets from G4SATBench. Note: the results are reported after the discretization procedure. The number of restarts τ corresponds to the number of additional executions performed on top of the inference on the original data. Augmentations outperform the Restarts method across the board. The best-performing augmentation is Aug_{SA}, having the average rank of 2.00. The best performing label-preserving augmentation (excluding RN) per dataset ranks 1.25 on average. Pure restarts come with the average rank of 4.43. Appendix G provides results on additional augmentations and a further breakdown of the results per restart count.

Training-time augmentations. When training the network on a problem with scarce data, augmentations can provide a way to generate artificial samples. In this experiment, we take a look at the effectiveness of this method on the 3-sat and the extremely symmetric cnfxorcard dataset. To do so, we generate 3 sets of size 100, 200, and 400 containing only formulas from the original distribution, and compare them to 2 sets of size 200, and 400, where only half of the formulas come from the original distribution, and the other half is augmented from the first. We evaluate the performance of GNN_{WalkSAT} trained on these datasets on a test set of 100 instances. Note: the results in this section are based on rounding the literal predictions to the nearest integer instead of using the discretization procedure. As can be seen in Figure 4, using the augmented formulas on 3-sat allows to practically close the gap to their original counterparts. Notably, on cnfxorcard usage of AU⁺ allows the model to break symmetry and clearly outperform both the dataset of original formulas and the RN-augmented version.

E.7 FourierSAT comparison on the mixed G4SATBench split

We finally compare the logit-space multilinear objective against a FourierSAT-style continuous SAT baseline [29] on a larger mixed split of G4SATBench. This experiment uses 2,100 DIMACS CNF instances: the same seven families, three difficulty levels (*easy*, *medium*, and *hard*), and 100 formulas per family/difficulty cell. Each cell contains both satisfiable and unsatisfiable instances. All methods

are evaluated on the original CNF clause lists, with no augmentation clauses or SAT preprocessing beyond tautology removal.

Protocol. The GD row is the deterministic multilinear baseline from Eq. (1): it optimizes logits $z \in \mathbb{R}^n$ with $p_i = \sigma(z_i)$, starts from $z = 0$, and runs exact full-batch GD with learning rate 0.1 for 20,000 steps in double precision. FourierSAT optimizes continuous variables $x \in [-1, 1]^n$, where $x_i = -1$ means Boolean true and $x_i = +1$ means Boolean false. It optimizes the clausewise FourierSAT objective with one SciPy SLSQP call, analytic gradients, box constraints $[-1, 1]^n$, and tolerance 10^{-12} . By Proposition 3.1, under the change of variables $p_i = (1 - x_i)/2$ this objective is an affine transform of the clause-failure probability, so for comparison to GD we report all FourierSAT variants using the same expected number of unsatisfied clauses $\mathcal{L}_T(p)$ from Eq. (1). The *FourierSAT* row uses the deterministic zero initialization $x = 0$, matching $p_i = 0.5$, length-weighted clauses, and a maximum of 100 SLSQP iterations. The *FourierSAT (Random Init)* row uses one deterministic per-instance draw from $\text{Uniform}[-1, 1]^n$ with global seed 0, again with length-weighted clauses and maximum 100 iterations. The *FourierSAT (Unweighted)* row uses the same symmetric zero initialization as FourierSAT, but optimizes uniform clause weights for up to 1000 SLSQP iterations.

Results. The figure reports mean final expected unsatisfied clauses grouped by difficulty and family. Lower is better. The GD baseline is best in most cells, especially on medium and hard instances, while optimizing FourierSAT with uniform clause weights improves the reported unweighted objective relative to the length-weighted FourierSAT baseline on most family/difficulty cells, with the largest gains on graph-derived families.

F Architecture details.

Base GNN. To explore the influence of augmentations, we first introduce a simple shallow network with GINE layers, SiLU activations, and LayerNorm, which is intended to serve as the baseline. We use 32 randomized $\{0, 1\}^N$ assignments concatenated with positional encodings obtained with Random Walk as the input features to the model and set the hidden dimension to 128. To add robustness, the features are perturbed by small $\mathcal{N}(0, 10^{-4})$ Gaussian noise. In order to convert the logits to soft probabilities, the model uses sigmoid.

WalkSAT GNN. Inspired by the structure of the WalkSAT algorithm, we design a custom neural architecture that iteratively improves the solution using $T = 5$ differentiable local-search iterations ran over parallel soft assignments. Similar to GNN_{Base} , the input features to the model are the 32 randomized assignments sampled uniformly from $\{0, 1\}^N$ and the positional Random Walk encodings. The hidden dimension is set to 128. Each iteration consists of three stages:

1. **Exploration block.** To escape local minima, the network generates 10 assignments out of each input assignment with random 25% of the variables flipped. The 32 best-scoring candidates according to the self-supervised loss are chosen to pass forward together with the 32 input assignments.
2. **Parallel operation blocks.** In order to improve the solution from iteration to iteration, we let the assignment pass through 5 parallel learnable operation blocks. Each operation is controlled with a data-dependent per-variable parameter λ chosen by the network. The 5 operations are as follows: (i) $\text{NOT}(x, \lambda) := \lambda(1 - x) + (1 - \lambda)x$; (ii) $\text{SET}_0(x, \lambda) := \lambda \cdot 0 + (1 - \lambda)x$; (iii) $\text{SET}_1(x, \lambda) := \lambda \cdot 1 + (1 - \lambda)x$; (iv) $\text{RESET}(x, \lambda) := \lambda \cdot 0.5 + (1 - \lambda)x$; (v) $\text{NOOP}(x, \lambda) := x$. The union of the resulting assignments is passed to the next block.
3. **Decider block.** To prevent the number of assignments exploding, this stage reduces the candidates back to 32 assignments through a learned convolution block.

$\text{GNN}_{\text{WalkSAT}}$ uses a mix of GINE and linear layers, SiLU activations and LayerNorm. Similar to GNN_{Base} , this model uses sigmoid to obtain the soft probabilities from logits. To ensure robustness the input to each iteration is perturbed by $\mathcal{N}(0, 10^{-4})$ Gaussian noise.

G Extended empirical results

This section reports the complete augmentation experiment results. Table 9 reports the training loss on individual datasets, and Tables 10–16 give a breakdown of inference results per dataset and number of additional executions $\tau \in \{1, 4, 32, 128\}$. *Hard* problem instances are omitted for inference due to time constraints. We further report the training loss of the tested models on a mixed 2100-instance dataset with 100 instances per group: 3.50 for GNN_{Base} and 20.32 for $\text{GNN}_{\text{WalkSAT}}$.

Table 9: Train loss per dataset and difficulty level.

Dataset	Level	$\text{GNN}_{\text{WalkSAT}}$	GNN_{Base}
3-SAT	Easy	0.75	2.91
3-SAT	Medium	1.49	15.75
3-SAT	Hard	2.61	33.09
CA	Easy	2.53	5.11
CA	Medium	11.31	30.72
CA	Hard	21.82	77.03
K-Clique	Easy	0.58	2.13
K-Clique	Medium	0.73	2.50
K-Clique	Hard	1.07	2.93
K-Domset	Easy	0.57	3.04
K-Domset	Medium	0.64	5.59
K-Domset	Hard	0.77	7.06
K-Vercov	Easy	0.73	3.69
K-Vercov	Medium	0.95	7.35
K-Vercov	Hard	1.23	10.86
PS	Easy	1.14	4.41
PS	Medium	5.46	22.65
PS	Hard	10.65	49.58
SR	Easy	0.70	3.65
SR	Medium	1.43	15.95
SR	Hard	3.01	44.26

Table 10: Average number of UNSAT clauses ($\downarrow$) on 3-SAT. The deviation is reported in superscript.

Method	Easy Dataset				Medium Dataset			
	$\tau = 1$	$\tau = 4$	$\tau = 32$	$\tau = 128$	$\tau = 1$	$\tau = 4$	$\tau = 32$	$\tau = 128$
WalkSAT	$1.08^{\pm 0.90}$	$0.64^{\pm 0.67}$	$0.60^{\pm 0.67}$	$0.60^{\pm 0.67}$	$2.71^{\pm 2.16}$	$1.55^{\pm 1.21}$	$0.79^{\pm 0.80}$	$0.59^{\pm 0.62}$
$\text{GNN}_{\text{WalkSAT}}$								
Restarts	$0.88^{\pm 0.76}$	$0.83^{\pm 0.75}$	$0.80^{\pm 0.72}$	$0.78^{\pm 0.72}$	$1.44^{\pm 1.02}$	$1.37^{\pm 0.99}$	$1.21^{\pm 0.88}$	$1.13^{\pm 0.82}$
Aug _{AU+}	$0.88^{\pm 0.77}$	$0.80^{\pm 0.74}$	$0.68^{\pm 0.68}$	$0.67^{\pm 0.68}$	$1.46^{\pm 1.02}$	$1.35^{\pm 1.07}$	$1.12^{\pm 0.96}$	$0.86^{\pm 0.82}$
Aug _{AU}	$0.87^{\pm 0.75}$	$0.85^{\pm 0.72}$	$0.75^{\pm 0.69}$	$0.70^{\pm 0.69}$	$1.47^{\pm 1.03}$	$1.38^{\pm 1.01}$	$1.17^{\pm 0.91}$	$1.04^{\pm 0.85}$
Aug _{CR}	$0.82^{\pm 0.73}$	$0.77^{\pm 0.71}$	$0.68^{\pm 0.68}$	$0.65^{\pm 0.67}$	$1.35^{\pm 0.98}$	$1.20^{\pm 0.85}$	$0.87^{\pm 0.72}$	$0.76^{\pm 0.68}$
Aug _{VE}	$0.87^{\pm 0.72}$	$0.76^{\pm 0.67}$	$0.68^{\pm 0.66}$	$0.65^{\pm 0.67}$	$1.49^{\pm 1.04}$	$1.38^{\pm 1.01}$	$1.09^{\pm 0.90}$	$0.93^{\pm 0.79}$
Aug _{SA}	$0.86^{\pm 0.71}$	$0.79^{\pm 0.70}$	$0.69^{\pm 0.69}$	$0.65^{\pm 0.67}$	$1.33^{\pm 0.90}$	$1.17^{\pm 0.85}$	$0.88^{\pm 0.70}$	$0.82^{\pm 0.70}$
Aug _{ERN}	$0.83^{\pm 0.73}$	$0.77^{\pm 0.74}$	$0.69^{\pm 0.69}$	$0.65^{\pm 0.69}$	$1.73^{\pm 1.19}$	$1.59^{\pm 1.09}$	$1.19^{\pm 0.91}$	$0.93^{\pm 0.74}$
GNN_{Base}								
Restarts	$1.66^{\pm 1.08}$	$1.31^{\pm 0.90}$	$1.03^{\pm 0.78}$	$0.90^{\pm 0.75}$	$6.70^{\pm 3.24}$	$6.08^{\pm 3.08}$	$4.98^{\pm 2.69}$	$4.43^{\pm 2.57}$
Aug _{AU+}	$1.74^{\pm 1.09}$	$1.40^{\pm 1.01}$	$0.98^{\pm 0.77}$	$0.79^{\pm 0.74}$	$7.12^{\pm 3.58}$	$6.57^{\pm 3.52}$	$5.52^{\pm 3.32}$	$4.61^{\pm 2.99}$
Aug _{AU}	$1.77^{\pm 1.21}$	$1.33^{\pm 1.04}$	$0.99^{\pm 0.78}$	$0.88^{\pm 0.77}$	$6.69^{\pm 3.34}$	$6.01^{\pm 3.05}$	$4.91^{\pm 2.72}$	$4.18^{\pm 2.34}$
Aug _{CR}	$1.67^{\pm 1.08}$	$1.30^{\pm 0.90}$	$0.94^{\pm 0.75}$	$0.84^{\pm 0.71}$	$6.67^{\pm 3.42}$	$6.08^{\pm 3.15}$	$4.66^{\pm 2.67}$	$4.08^{\pm 2.60}$
Aug _{VE}	$1.76^{\pm 1.16}$	$1.38^{\pm 1.06}$	$0.97^{\pm 0.80}$	$0.76^{\pm 0.71}$	$7.06^{\pm 3.52}$	$6.81^{\pm 3.54}$	$5.83^{\pm 3.40}$	$5.13^{\pm 3.15}$
Aug _{SA}	$1.61^{\pm 1.11}$	$1.32^{\pm 0.92}$	$0.95^{\pm 0.74}$	$0.77^{\pm 0.71}$	$6.97^{\pm 3.36}$	$5.98^{\pm 3.15}$	$4.72^{\pm 2.63}$	$4.08^{\pm 2.44}$
Aug _{ERN}	$1.75^{\pm 1.15}$	$1.36^{\pm 0.97}$	$0.96^{\pm 0.76}$	$0.79^{\pm 0.71}$	$6.74^{\pm 3.35}$	$6.05^{\pm 3.11}$	$4.77^{\pm 2.76}$	$4.03^{\pm 2.46}$

Table 11: Average number of UNSAT clauses ($\downarrow$) on CA. The deviation is reported in superscript.

Method	Easy Dataset				Medium Dataset			
	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$
WalkSAT	2.17 ± 2.74	1.90 ± 2.31	1.80 ± 2.16	1.80 ± 2.16	13.68 ± 16.62	11.88 ± 14.64	10.06 ± 12.13	9.07 ± 11.05
GNN_{WalkSAT}								
Restarts	2.27 ± 2.47	2.21 ± 2.46	2.14 ± 2.40	2.10 ± 2.35	9.80 ± 10.20	9.53 ± 10.15	9.08 ± 9.96	8.78 ± 9.64
Aug _{AU+}	2.28 ± 2.49	2.14 ± 2.36	2.01 ± 2.30	1.94 ± 2.26	9.91 ± 10.31	9.43 ± 10.19	8.52 ± 9.62	8.06 ± 9.16
Aug _{AU}	2.37 ± 2.53	2.18 ± 2.39	1.99 ± 2.32	1.96 ± 2.27	9.97 ± 10.28	9.60 ± 10.12	9.02 ± 9.93	8.53 ± 9.75
Aug _{CR}	2.29 ± 2.52	2.06 ± 2.36	1.92 ± 2.27	1.86 ± 2.20	9.70 ± 10.18	9.24 ± 9.87	8.24 ± 9.21	7.92 ± 8.95
Aug _{VE}	2.37 ± 2.56	2.22 ± 2.47	2.01 ± 2.37	1.95 ± 2.32	10.21 ± 10.28	10.13 ± 10.30	9.97 ± 10.34	9.65 ± 10.27
Aug _{SA}	2.29 ± 2.51	2.12 ± 2.38	1.96 ± 2.30	1.93 ± 2.26	9.58 ± 10.08	9.05 ± 9.87	8.26 ± 9.24	7.98 ± 9.03
Aug _{RN}	2.22 ± 2.47	2.12 ± 2.45	1.94 ± 2.25	1.91 ± 2.22	9.75 ± 10.13	9.28 ± 9.93	8.48 ± 9.41	8.13 ± 9.14
GNN_{Base}								
Restarts	2.76 ± 2.82	2.54 ± 2.74	2.21 ± 2.53	2.13 ± 2.41	14.59 ± 12.66	13.76 ± 12.24	12.62 ± 11.79	12.02 ± 11.67
Aug _{AU+}	2.75 ± 2.85	2.46 ± 2.62	2.11 ± 2.40	1.98 ± 2.31	14.78 ± 12.63	13.75 ± 12.22	11.95 ± 11.84	11.24 ± 11.60
Aug _{AU}	2.77 ± 2.82	2.50 ± 2.74	2.17 ± 2.48	2.05 ± 2.32	15.50 ± 13.18	15.17 ± 13.34	14.63 ± 13.38	14.11 ± 13.48
Aug _{CR}	2.77 ± 2.95	2.39 ± 2.71	2.04 ± 2.38	1.97 ± 2.30	14.88 ± 12.74	13.57 ± 12.46	12.16 ± 11.79	11.25 ± 11.38
Aug _{VE}	2.84 ± 2.92	2.62 ± 2.83	2.08 ± 2.39	1.99 ± 2.31	15.70 ± 13.10	15.56 ± 13.16	15.05 ± 13.36	14.67 ± 13.20
Aug _{SA}	2.77 ± 2.88	2.36 ± 2.59	2.06 ± 2.39	1.98 ± 2.31	14.57 ± 12.57	13.76 ± 12.52	12.05 ± 12.06	11.23 ± 11.48
Aug _{RN}	2.67 ± 2.81	2.46 ± 2.63	2.11 ± 2.41	2.01 ± 2.30	14.58 ± 12.63	13.50 ± 12.11	12.04 ± 11.80	11.31 ± 11.50

Table 12: Average number of UNSAT clauses ($\downarrow$) on K-Clique. The deviation is reported in superscript.

Method	Easy Dataset				Medium Dataset			
	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$
WalkSAT	0.59 ± 0.51	0.52 ± 0.52	0.51 ± 0.52	0.51 ± 0.52	0.66 ± 0.52	0.51 ± 0.52	0.51 ± 0.52	0.51 ± 0.52
GNN_{WalkSAT}								
Restarts	0.59 ± 0.51	0.55 ± 0.52	0.53 ± 0.52	0.51 ± 0.52	0.73 ± 0.53	0.61 ± 0.53	0.58 ± 0.54	0.55 ± 0.52
Aug _{AU+}	0.68 ± 0.55	0.62 ± 0.51	0.53 ± 0.52	0.51 ± 0.52	0.81 ± 0.54	0.81 ± 0.54	0.75 ± 0.48	0.69 ± 0.49
Aug _{AU}	0.64 ± 0.50	0.58 ± 0.52	0.51 ± 0.52	0.51 ± 0.52	0.79 ± 0.54	0.77 ± 0.55	0.61 ± 0.53	0.54 ± 0.52
Aug _{CR}	0.72 ± 0.53	0.70 ± 0.52	0.64 ± 0.54	0.60 ± 0.53	0.76 ± 0.53	0.76 ± 0.53	0.76 ± 0.53	0.70 ± 0.52
Aug _{VE}	0.71 ± 0.56	0.71 ± 0.56	0.69 ± 0.56	0.66 ± 0.52	0.83 ± 0.57	0.83 ± 0.57	0.82 ± 0.58	0.77 ± 0.57
Aug _{SA}	0.66 ± 0.55	0.57 ± 0.52	0.52 ± 0.52	0.51 ± 0.52	0.77 ± 0.53	0.67 ± 0.53	0.54 ± 0.52	0.52 ± 0.52
Aug _{RN}	0.63 ± 0.51	0.58 ± 0.52	0.51 ± 0.52	0.51 ± 0.52	0.79 ± 0.48	0.73 ± 0.49	0.58 ± 0.54	0.52 ± 0.52
GNN_{Base}								
Restarts	0.94 ± 0.55	0.76 ± 0.55	0.55 ± 0.54	0.52 ± 0.52	1.18 ± 0.58	0.95 ± 0.41	0.67 ± 0.49	0.54 ± 0.52
Aug _{AU+}	1.10 ± 0.54	1.02 ± 0.53	0.92 ± 0.54	0.90 ± 0.56	1.43 ± 0.64	1.42 ± 0.65	1.32 ± 0.57	1.22 ± 0.52
Aug _{AU}	0.98 ± 0.49	0.92 ± 0.51	0.59 ± 0.55	0.54 ± 0.52	1.39 ± 0.63	1.27 ± 0.63	0.89 ± 0.53	0.68 ± 0.53
Aug _{CR}	1.12 ± 0.57	1.11 ± 0.57	1.07 ± 0.54	1.01 ± 0.56	1.44 ± 0.64	1.43 ± 0.64	1.37 ± 0.56	1.26 ± 0.52
Aug _{VE}	1.04 ± 0.51	0.97 ± 0.50	0.71 ± 0.48	0.66 ± 0.50	1.42 ± 0.64	1.41 ± 0.62	1.26 ± 0.68	1.18 ± 0.69
Aug _{SA}	1.06 ± 0.51	0.97 ± 0.48	0.61 ± 0.55	0.51 ± 0.52	1.20 ± 0.55	0.98 ± 0.45	0.66 ± 0.50	0.58 ± 0.52
Aug _{RN}	1.06 ± 0.51	0.98 ± 0.47	0.82 ± 0.46	0.62 ± 0.51	1.17 ± 0.59	0.98 ± 0.47	0.69 ± 0.51	0.53 ± 0.52

Table 13: Average number of UNSAT clauses ($\downarrow$) on K-Domset. The deviation is reported in superscript.

Method	Easy Dataset				Medium Dataset			
	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$
WalkSAT	$0.68^{\pm 0.58}$	$0.55^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.93^{\pm 0.74}$	$0.63^{\pm 0.69}$	$0.60^{\pm 0.68}$	$0.60^{\pm 0.68}$
GNN_{WalkSAT}								
Restarts	$0.57^{\pm 0.57}$	$0.55^{\pm 0.58}$	$0.53^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.63^{\pm 0.71}$	$0.62^{\pm 0.69}$	$0.62^{\pm 0.69}$	$0.61^{\pm 0.69}$
Aug _{AU+}	$0.58^{\pm 0.57}$	$0.54^{\pm 0.58}$	$0.53^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.64^{\pm 0.70}$	$0.64^{\pm 0.70}$	$0.62^{\pm 0.69}$	$0.62^{\pm 0.69}$
Aug _{AU}	$0.56^{\pm 0.57}$	$0.54^{\pm 0.58}$	$0.53^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.66^{\pm 0.70}$	$0.65^{\pm 0.70}$	$0.62^{\pm 0.69}$	$0.61^{\pm 0.69}$
Aug _{CR}	$0.58^{\pm 0.57}$	$0.55^{\pm 0.56}$	$0.54^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.65^{\pm 0.70}$	$0.65^{\pm 0.70}$	$0.62^{\pm 0.69}$	$0.61^{\pm 0.68}$
Aug _{VE}	$0.59^{\pm 0.57}$	$0.59^{\pm 0.57}$	$0.58^{\pm 0.57}$	$0.56^{\pm 0.56}$	$0.65^{\pm 0.70}$	$0.65^{\pm 0.70}$	$0.65^{\pm 0.70}$	$0.65^{\pm 0.70}$
Aug _{SA}	$0.55^{\pm 0.56}$	$0.54^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.62^{\pm 0.69}$	$0.62^{\pm 0.69}$	$0.60^{\pm 0.68}$	$0.60^{\pm 0.68}$
Aug _{RN}	$0.57^{\pm 0.57}$	$0.55^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.53^{\pm 0.56}$	$0.63^{\pm 0.71}$	$0.62^{\pm 0.71}$	$0.61^{\pm 0.69}$	$0.60^{\pm 0.68}$
GNN_{Base}								
Restarts	$1.36^{\pm 0.77}$	$1.08^{\pm 0.66}$	$0.73^{\pm 0.60}$	$0.67^{\pm 0.59}$	$1.51^{\pm 0.92}$	$1.25^{\pm 0.83}$	$1.01^{\pm 0.72}$	$0.93^{\pm 0.69}$
Aug _{AU+}	$1.42^{\pm 0.75}$	$1.21^{\pm 0.70}$	$0.80^{\pm 0.65}$	$0.64^{\pm 0.56}$	$1.70^{\pm 0.97}$	$1.58^{\pm 0.93}$	$1.27^{\pm 0.89}$	$1.01^{\pm 0.81}$
Aug _{AU}	$1.37^{\pm 0.71}$	$1.00^{\pm 0.67}$	$0.74^{\pm 0.54}$	$0.62^{\pm 0.56}$	$1.55^{\pm 0.88}$	$1.34^{\pm 0.77}$	$0.97^{\pm 0.74}$	$0.77^{\pm 0.71}$
Aug _{CR}	$1.55^{\pm 0.83}$	$1.43^{\pm 0.82}$	$1.17^{\pm 0.74}$	$0.96^{\pm 0.68}$	$1.76^{\pm 1.02}$	$1.74^{\pm 0.99}$	$1.71^{\pm 0.98}$	$1.65^{\pm 0.98}$
Aug _{VE}	$1.46^{\pm 0.72}$	$1.25^{\pm 0.67}$	$0.89^{\pm 0.57}$	$0.72^{\pm 0.53}$	$1.72^{\pm 0.96}$	$1.64^{\pm 0.92}$	$1.45^{\pm 0.82}$	$1.32^{\pm 0.78}$
Aug _{SA}	$1.22^{\pm 0.69}$	$0.93^{\pm 0.61}$	$0.64^{\pm 0.58}$	$0.61^{\pm 0.58}$	$1.37^{\pm 0.86}$	$1.11^{\pm 0.72}$	$0.78^{\pm 0.69}$	$0.70^{\pm 0.70}$
Aug _{RN}	$1.45^{\pm 0.77}$	$1.18^{\pm 0.80}$	$0.74^{\pm 0.66}$	$0.68^{\pm 0.58}$	$1.60^{\pm 0.94}$	$1.34^{\pm 0.86}$	$1.12^{\pm 0.84}$	$0.92^{\pm 0.75}$

Table 14: Average number of UNSAT clauses ($\downarrow$) on K-Vercov. The deviation is reported in superscript.

Method	Easy Dataset				Medium Dataset			
	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$
WalkSAT	$0.83^{\pm 0.68}$	$0.61^{\pm 0.68}$	$0.60^{\pm 0.68}$	$0.60^{\pm 0.68}$	$1.07^{\pm 1.01}$	$0.77^{\pm 0.91}$	$0.74^{\pm 0.91}$	$0.74^{\pm 0.91}$
GNN_{WalkSAT}								
Restarts	$0.66^{\pm 0.73}$	$0.62^{\pm 0.69}$	$0.61^{\pm 0.68}$	$0.61^{\pm 0.68}$	$0.92^{\pm 1.06}$	$0.86^{\pm 1.03}$	$0.85^{\pm 1.04}$	$0.84^{\pm 1.01}$
Aug _{AU+}	$0.69^{\pm 0.72}$	$0.68^{\pm 0.72}$	$0.64^{\pm 0.70}$	$0.61^{\pm 0.68}$	$0.95^{\pm 1.06}$	$0.95^{\pm 1.06}$	$0.95^{\pm 1.06}$	$0.95^{\pm 1.06}$
Aug _{AU}	$0.65^{\pm 0.70}$	$0.64^{\pm 0.70}$	$0.61^{\pm 0.69}$	$0.60^{\pm 0.68}$	$0.93^{\pm 1.03}$	$0.90^{\pm 1.02}$	$0.89^{\pm 1.01}$	$0.85^{\pm 1.01}$
Aug _{CR}	$0.71^{\pm 0.73}$	$0.70^{\pm 0.73}$	$0.68^{\pm 0.72}$	$0.65^{\pm 0.72}$	$0.92^{\pm 1.06}$	$0.92^{\pm 1.06}$	$0.92^{\pm 1.06}$	$0.92^{\pm 1.06}$
Aug _{VE}	$0.70^{\pm 0.73}$	$0.70^{\pm 0.73}$	$0.68^{\pm 0.72}$	$0.67^{\pm 0.71}$	$0.95^{\pm 1.07}$	$0.95^{\pm 1.07}$	$0.95^{\pm 1.07}$	$0.95^{\pm 1.07}$
Aug _{SA}	$0.65^{\pm 0.70}$	$0.63^{\pm 0.71}$	$0.62^{\pm 0.69}$	$0.60^{\pm 0.68}$	$0.96^{\pm 1.05}$	$0.95^{\pm 1.03}$	$0.82^{\pm 0.95}$	$0.77^{\pm 0.93}$
Aug _{RN}	$0.68^{\pm 0.71}$	$0.63^{\pm 0.69}$	$0.60^{\pm 0.68}$	$0.60^{\pm 0.68}$	$0.94^{\pm 1.03}$	$0.93^{\pm 1.01}$	$0.84^{\pm 0.93}$	$0.80^{\pm 0.93}$
GNN_{Base}								
Restarts	$1.70^{\pm 0.92}$	$1.37^{\pm 0.92}$	$0.96^{\pm 0.78}$	$0.84^{\pm 0.75}$	$1.79^{\pm 1.27}$	$1.61^{\pm 1.06}$	$1.49^{\pm 0.99}$	$1.44^{\pm 0.98}$
Aug _{AU+}	$2.02^{\pm 1.07}$	$1.96^{\pm 1.12}$	$1.75^{\pm 1.19}$	$1.69^{\pm 1.23}$	$1.93^{\pm 1.27}$	$1.75^{\pm 1.05}$	$1.43^{\pm 1.08}$	$1.21^{\pm 1.05}$
Aug _{AU}	$1.90^{\pm 1.09}$	$1.65^{\pm 1.12}$	$1.21^{\pm 0.97}$	$0.93^{\pm 0.84}$	$2.01^{\pm 1.40}$	$1.92^{\pm 1.40}$	$1.58^{\pm 1.04}$	$1.40^{\pm 0.96}$
Aug _{CR}	$2.06^{\pm 1.07}$	$2.04^{\pm 1.08}$	$1.95^{\pm 1.14}$	$1.84^{\pm 1.19}$	$2.02^{\pm 1.41}$	$1.97^{\pm 1.37}$	$1.75^{\pm 1.13}$	$1.41^{\pm 1.13}$
Aug _{VE}	$2.01^{\pm 1.10}$	$1.90^{\pm 1.11}$	$1.71^{\pm 1.09}$	$1.55^{\pm 1.10}$	$2.02^{\pm 1.37}$	$2.02^{\pm 1.37}$	$1.99^{\pm 1.27}$	$1.98^{\pm 1.26}$
Aug _{SA}	$1.93^{\pm 1.06}$	$1.61^{\pm 1.11}$	$1.28^{\pm 1.05}$	$0.98^{\pm 0.92}$	$1.81^{\pm 1.24}$	$1.57^{\pm 1.07}$	$1.19^{\pm 0.98}$	$1.01^{\pm 0.96}$
Aug _{RN}	$2.04^{\pm 1.06}$	$1.98^{\pm 1.07}$	$1.69^{\pm 1.21}$	$1.52^{\pm 1.23}$	$1.69^{\pm 1.08}$	$1.47^{\pm 1.09}$	$1.10^{\pm 1.02}$	$0.97^{\pm 0.96}$

Table 15: Average number of UNSAT clauses ($\downarrow$) on PS. The deviation is reported in superscript.

Method	Easy Dataset				Medium Dataset			
	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$
WalkSAT	$1.30^{\pm 1.62}$	$0.89^{\pm 1.21}$	$0.80^{\pm 1.03}$	$0.80^{\pm 1.03}$	$8.50^{\pm 12.50}$	$6.78^{\pm 10.90}$	$5.40^{\pm 9.24}$	$4.96^{\pm 8.71}$
GNN_{WalkSAT}								
Restarts	$1.34^{\pm 1.43}$	$1.12^{\pm 1.31}$	$0.91^{\pm 1.16}$	$0.84^{\pm 1.09}$	$5.89^{\pm 6.63}$	$5.63^{\pm 6.52}$	$5.20^{\pm 6.34}$	$5.06^{\pm 6.29}$
Aug _{AU+}	$1.31^{\pm 1.45}$	$1.09^{\pm 1.32}$	$0.92^{\pm 1.13}$	$0.85^{\pm 1.10}$	$5.85^{\pm 6.52}$	$5.40^{\pm 6.34}$	$4.77^{\pm 6.20}$	$4.53^{\pm 6.09}$
Aug _{AU}	$1.35^{\pm 1.47}$	$1.13^{\pm 1.29}$	$0.92^{\pm 1.10}$	$0.85^{\pm 1.06}$	$5.91^{\pm 6.53}$	$5.63^{\pm 6.53}$	$5.02^{\pm 6.22}$	$4.72^{\pm 6.12}$
Aug _{CR}	$1.28^{\pm 1.40}$	$1.13^{\pm 1.32}$	$0.97^{\pm 1.18}$	$0.88^{\pm 1.09}$	$5.92^{\pm 6.81}$	$5.56^{\pm 6.58}$	$5.08^{\pm 6.30}$	$4.83^{\pm 6.16}$
Aug _{VE}	$1.37^{\pm 1.54}$	$1.19^{\pm 1.35}$	$1.01^{\pm 1.23}$	$0.91^{\pm 1.13}$	$6.11^{\pm 6.75}$	$5.95^{\pm 6.72}$	$5.60^{\pm 6.49}$	$5.38^{\pm 6.33}$
Aug _{SA}	$1.32^{\pm 1.42}$	$1.12^{\pm 1.27}$	$0.93^{\pm 1.13}$	$0.87^{\pm 1.12}$	$5.78^{\pm 6.65}$	$5.50^{\pm 6.54}$	$4.84^{\pm 6.10}$	$4.63^{\pm 6.07}$
Aug _{RN}	$1.37^{\pm 1.47}$	$1.18^{\pm 1.39}$	$0.92^{\pm 1.14}$	$0.85^{\pm 1.07}$	$5.86^{\pm 6.57}$	$5.57^{\pm 6.52}$	$4.93^{\pm 6.31}$	$4.52^{\pm 6.18}$
GNN_{Base}								
Restarts	$1.86^{\pm 1.70}$	$1.65^{\pm 1.59}$	$1.27^{\pm 1.33}$	$1.09^{\pm 1.21}$	$10.29^{\pm 8.63}$	$9.40^{\pm 8.24}$	$8.41^{\pm 8.04}$	$8.00^{\pm 7.92}$
Aug _{AU+}	$1.87^{\pm 1.69}$	$1.63^{\pm 1.52}$	$1.17^{\pm 1.30}$	$1.10^{\pm 1.23}$	$10.34^{\pm 8.87}$	$9.34^{\pm 8.25}$	$7.70^{\pm 7.48}$	$7.00^{\pm 7.15}$
Aug _{AU}	$1.99^{\pm 1.83}$	$1.63^{\pm 1.61}$	$1.22^{\pm 1.31}$	$1.12^{\pm 1.22}$	$10.40^{\pm 8.93}$	$9.60^{\pm 8.38}$	$8.24^{\pm 7.90}$	$7.50^{\pm 7.39}$
Aug _{CR}	$1.95^{\pm 1.78}$	$1.68^{\pm 1.56}$	$1.18^{\pm 1.33}$	$1.06^{\pm 1.20}$	$10.42^{\pm 8.96}$	$9.43^{\pm 8.27}$	$7.95^{\pm 7.59}$	$7.18^{\pm 7.25}$
Aug _{VE}	$1.88^{\pm 1.72}$	$1.65^{\pm 1.60}$	$1.23^{\pm 1.38}$	$1.10^{\pm 1.30}$	$10.91^{\pm 8.97}$	$10.63^{\pm 8.86}$	$9.68^{\pm 8.19}$	$9.09^{\pm 7.96}$
Aug _{SA}	$1.92^{\pm 1.73}$	$1.60^{\pm 1.56}$	$1.16^{\pm 1.27}$	$1.02^{\pm 1.15}$	$10.20^{\pm 8.69}$	$9.21^{\pm 8.52}$	$7.83^{\pm 7.82}$	$7.24^{\pm 7.62}$
Aug _{RN}	$1.89^{\pm 1.67}$	$1.56^{\pm 1.41}$	$1.18^{\pm 1.26}$	$1.06^{\pm 1.20}$	$10.15^{\pm 8.58}$	$9.24^{\pm 8.31}$	$7.88^{\pm 7.86}$	$7.36^{\pm 7.60}$

Table 16: Average number of UNSAT clauses ($\downarrow$) on SR. The deviation is reported in superscript.

Method	Easy Dataset				Medium Dataset			
	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$	$\tau=1$	$\tau=4$	$\tau=32$	$\tau=128$
WalkSAT	$0.88^{\pm 0.77}$	$0.53^{\pm 0.52}$	$0.50^{\pm 0.50}$	$0.50^{\pm 0.50}$	$2.25^{\pm 1.57}$	$1.12^{\pm 0.77}$	$0.67^{\pm 0.47}$	$0.53^{\pm 0.50}$
GNN_{WalkSAT}								
Restarts	$0.78^{\pm 0.60}$	$0.73^{\pm 0.58}$	$0.60^{\pm 0.49}$	$0.57^{\pm 0.50}$	$1.53^{\pm 0.88}$	$1.40^{\pm 0.77}$	$1.21^{\pm 0.70}$	$1.15^{\pm 0.67}$
Aug _{AU+}	$0.79^{\pm 0.57}$	$0.66^{\pm 0.54}$	$0.59^{\pm 0.51}$	$0.55^{\pm 0.50}$	$1.42^{\pm 0.81}$	$1.24^{\pm 0.79}$	$0.95^{\pm 0.59}$	$0.84^{\pm 0.56}$
Aug _{AU}	$0.79^{\pm 0.57}$	$0.68^{\pm 0.55}$	$0.56^{\pm 0.50}$	$0.54^{\pm 0.50}$	$1.52^{\pm 0.95}$	$1.35^{\pm 0.90}$	$1.00^{\pm 0.64}$	$0.89^{\pm 0.55}$
Aug _{CR}	$0.79^{\pm 0.61}$	$0.70^{\pm 0.58}$	$0.58^{\pm 0.50}$	$0.55^{\pm 0.50}$	$1.47^{\pm 0.83}$	$1.29^{\pm 0.70}$	$1.11^{\pm 0.63}$	$0.97^{\pm 0.64}$
Aug _{VE}	$0.80^{\pm 0.62}$	$0.70^{\pm 0.54}$	$0.59^{\pm 0.49}$	$0.57^{\pm 0.50}$	$1.60^{\pm 0.97}$	$1.60^{\pm 0.97}$	$1.55^{\pm 0.96}$	$1.49^{\pm 0.95}$
Aug _{SA}	$0.75^{\pm 0.58}$	$0.68^{\pm 0.53}$	$0.57^{\pm 0.50}$	$0.55^{\pm 0.50}$	$1.40^{\pm 0.82}$	$1.21^{\pm 0.66}$	$1.06^{\pm 0.57}$	$0.93^{\pm 0.56}$
Aug _{RN}	$0.82^{\pm 0.63}$	$0.69^{\pm 0.58}$	$0.55^{\pm 0.50}$	$0.54^{\pm 0.50}$	$1.56^{\pm 0.84}$	$1.43^{\pm 0.82}$	$1.13^{\pm 0.68}$	$0.98^{\pm 0.59}$
GNN_{Base}								
Restarts	$1.58^{\pm 0.97}$	$1.31^{\pm 0.79}$	$1.00^{\pm 0.64}$	$0.85^{\pm 0.59}$	$7.17^{\pm 2.87}$	$6.30^{\pm 2.49}$	$5.18^{\pm 2.16}$	$4.55^{\pm 2.07}$
Aug _{AU+}	$1.72^{\pm 1.11}$	$1.33^{\pm 0.85}$	$0.95^{\pm 0.67}$	$0.82^{\pm 0.59}$	$7.24^{\pm 2.81}$	$6.48^{\pm 2.63}$	$5.02^{\pm 2.17}$	$4.38^{\pm 2.03}$
Aug _{AU}	$1.62^{\pm 1.03}$	$1.31^{\pm 0.87}$	$1.02^{\pm 0.68}$	$0.86^{\pm 0.59}$	$7.29^{\pm 2.86}$	$6.41^{\pm 2.64}$	$5.15^{\pm 2.43}$	$4.73^{\pm 2.26}$
Aug _{CR}	$1.62^{\pm 0.94}$	$1.33^{\pm 0.83}$	$0.99^{\pm 0.64}$	$0.77^{\pm 0.58}$	$7.28^{\pm 3.05}$	$6.43^{\pm 2.73}$	$5.21^{\pm 2.25}$	$4.55^{\pm 2.14}$
Aug _{VE}	$1.74^{\pm 1.02}$	$1.51^{\pm 0.92}$	$1.08^{\pm 0.72}$	$0.89^{\pm 0.62}$	$7.90^{\pm 3.22}$	$7.77^{\pm 3.17}$	$7.29^{\pm 3.05}$	$6.72^{\pm 2.94}$
Aug _{SA}	$1.65^{\pm 0.96}$	$1.26^{\pm 0.82}$	$0.94^{\pm 0.68}$	$0.81^{\pm 0.56}$	$7.03^{\pm 2.87}$	$6.35^{\pm 2.59}$	$5.10^{\pm 2.25}$	$4.46^{\pm 1.93}$
Aug _{RN}	$1.65^{\pm 1.04}$	$1.35^{\pm 0.85}$	$0.92^{\pm 0.63}$	$0.80^{\pm 0.59}$	$7.30^{\pm 2.93}$	$6.34^{\pm 2.65}$	$5.10^{\pm 2.28}$	$4.45^{\pm 2.06}$

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: Yes, all claims are either theoretically or empirically validated.

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: Section 7 explicitly discusses the present paper’s limitations.

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: The full set of assumptions and a complete proof are provided in either the main paper or the appendix.

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: The paper provides detailed descriptions of the experiments in the Appendix.

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: We provide detailed explanations of the experimental setup, and upon publication we plan to release the code.

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: The experiments section specifies the benchmark families, the number of formulas per family and difficulty, whether a sigmoid is used, the initialization, the optimizer and the learning rate.

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: Error bars are provided (e.g., see table 2,4,5,6)

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: [No]

Justification: We report the main experimental setup and hardware used for the main experiments, but we did not track the memory and total runtime across all experiments.

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: This paper is methodological in scope and does not involve human subjects, hidden data collection, or anonymity-breaking disclosures.

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: [N/A]

Justification: This paper studies theoretical properties of gradient-based SAT solving. Any broader impact is therefore indirect and depends on future uses of these techniques.

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 does not release models, datasets, or other assets that create a special misuse risk.

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: [N/A]

Justification:

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: The 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: The paper does not involve crowdsourcing or 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 or 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 are not part of the paper's core method, and no non-standard methodological use of LLMs is claimed in the research content.

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.