IEEE/ACM
2024 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN
ICCAD

DiffSAT: Differential MaxSAT Layer for SAT Solving

Yu Zhang[1], Hui-Ling Zhen[2], Mingxuan Yuan[2], Bei Yu[1]

[1] The Chinese University of Hong Kong
[2] Huawei Noah's Ark Lab

Oct. 29, 2024

Outline

Introduction

Introducing SAT

- Variables: $v1, v2, v3, \ldots, vn$
- Literals: $v1, \neg v1, v2, \neg v2, \ldots$
- Clauses: disjunction ($\vee$) of literals, such as $v1 \vee v2 \vee v3$
- Conjunctive Normal Form (CNF): conjunction ($\wedge$) of clauses
- Objective: assign variables to $\{-1, 1\}$ to make the CNF satisfied
- Formula can be SAT / UNSAT

SAT CNF example: $(v1 \vee \neg v2 \vee v3) \wedge (v2 \vee \neg v1 \vee \neg v3) \wedge (\neg v1 \vee \neg v3)$ can be satisfied with v1 = True, v2 = True, and v3 = False.

UNSAT CNF example: $(v1) \wedge (\neg v1) \wedge (\neg v1 \vee v2) \wedge (\neg v2)$ can not be satisfied with an unsat core $\{(v1), (\neg v1)\}$.

Background

	Algorithm	Approaches	Pros	Cons
Heuristic-based SAT Solver	CDCL	Kissat,CaDiCal, Glucose,...	Fast Search	Exponential search space, Endless loop
Learning-based SAT Solver	GNN+Classifier	NeuroSAT[1],DeepGate2[2], SATformer[3],...	No runtime limit	Low accuracy

Motivation:

Can we develop a SAT solving algorithm that is both accurate and fast?

[1]Daniel Selsam et al. (2018). "Learning a SAT Solver from Single-Bit Supervision". In: *ICLR*.

[2]Zhengyuan Shi, Hongyang Pan, et al. (2023). "Deepgate2: Functionality-aware circuit representation learning". In: *2023 ICCAD*. IEEE, pp. 1–9.

[3]Zhengyuan Shi, Min Li, et al. (2023). "SATformer: Transformer-Based UNSAT Core Learning". In: *2023 ICCAD*. IEEE, pp. 1–4.

Contribution

We introduce `DiffSAT`, a differential approach aimed at progressively searching for satisfying assignments for SAT problems. Our DiffSAT algorithm mainly includes three steps:

1. MaxSAT generalization
2. Semidefinite initialization
3. Solution refinement by differential MaxSAT layer

DiffSAT Algorithm

From SAT to MaxSAT

The MaxSAT problem serves as the counterpart to the SAT problem, aiming to maximize the number of satisfied clauses:

$$\max_{\tilde{\boldsymbol{v}} \in \{-1,1\}^n} \sum_{j=1}^{m} \bigvee_{i=1}^{n} \mathbf{1}\{s_{ij} \tilde{\boldsymbol{v}}_i > 0\}, \tag{1}$$

For conversing to loss, we formulate Eq. 1 in its minimization, or *unsatisfiability*, form as

$$\min_{\tilde{\boldsymbol{v}} \in \{-1,1\}^n} \sum_{j=1}^{m} \bigwedge_{i=1}^{n} \mathbf{1}\{s_{ij} \tilde{\boldsymbol{v}}_i < 0\}, \tag{2}$$

Construct Loss for MaxSAT

The minimization problem in Eq. (2) can be solved by introducing a quadratic loss function as

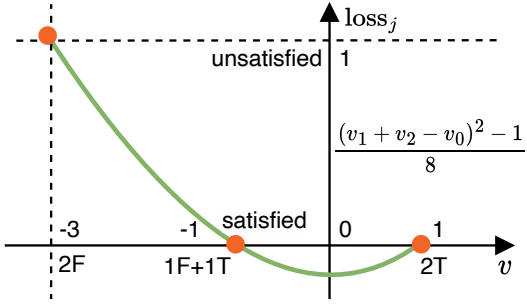
$$\text{loss}_j = \frac{(\sum_{i=0}^{n} s_{ij}\tilde{v}_i)^2 - (m_j - 1)^2}{4m_j}, \tag{3}$$

$$\text{loss} = \sum_{j=1}^{m} \text{loss}_j. \tag{4}$$

We introduce $v_0 = 1$ and $s_{0j} = -1$ to make the purely quadratic loss function.

Construct Loss for MaxSAT

Take a simple SAT problem with clauses $(v_1 \vee v_2)$ as an illustrating example. For this clause, the loss quantity is equal to +1 if no literal is satisfied, and 0 or less if at least one literal is True.



SDP Initialization: From MaxSAT to SDP Relaxation

Relax discrete variable $\tilde{\boldsymbol{v}}_i \in \{-1, 1\}$ to a unit vector $\boldsymbol{v}_i \in \mathbb{R}^k$, $\|\boldsymbol{v}_i\| = 1$, the MaxSAT problem becomes the following SDP problem:

$$\min_{\boldsymbol{V} \in \mathbb{R}^{k \times (n+1)}} \langle \boldsymbol{S}^\top \boldsymbol{S}, \boldsymbol{V}^\top \boldsymbol{V} \rangle, \quad \text{s.t. } \|\boldsymbol{v}_i\| = 1, i \in \{0, 1, 2, \ldots, n\}, \tag{5}$$

There is a simple closed-form solution for the remaining $\boldsymbol{v}_i$ given by:

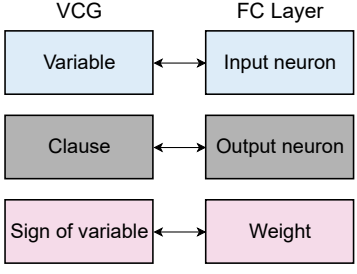
$$v_i := \text{normalize}(-\sum_{k \neq i} (\boldsymbol{S}^\top \boldsymbol{S})_{ik} v_k) = -\frac{\boldsymbol{z}_i}{\|\boldsymbol{z}_i\|}, \tag{6}$$

High Approximation Ratio

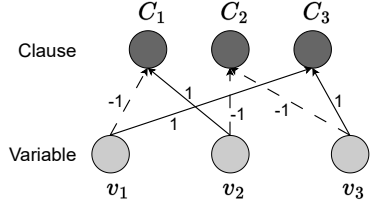
SDP achieves 87.5% approximation to original MaxSAT.

Solution Refinement: Differential MaxSAT Layer

We draw an analogy between the FC layer and variable-clause graph (VCG) and develop a novel network layer architecture, `MaxSAT Layer`.



Analogy between VCG and FC layer



$\phi := (\neg v_1 \vee v_2) \wedge (\neg v_2 \vee \neg v_3) \wedge (v_1 \vee v_3)$

Illustration of VCG

Forward and Backward Pass of MaxSAT Layer

Forward pass:

- Check the satisfiability of CNF

Algorithm 1 The forward pass of MaxSAT layer

Input: Variable assignment $v^k \in \mathbb{R}^n$ at k-th epoch, conjunctive normal formula ϕ.
Output: y^k, loss$_j$, solution v^*.

$\tilde{v}^k \leftarrow [v_i^k > 0 : i = 1, \ldots, n]$;
$\phi' \leftarrow \phi(\tilde{v}_1^k, \ldots, \tilde{v}_n^k)$;
if ϕ' is satisfiable **then**
  $y^k \leftarrow$ True;
  $v^* \leftarrow v^k$;
else
  $y^k \leftarrow$ False;
end if

Backward pass:

- Select one variable based on absolute gradient, and perform gradient descent.

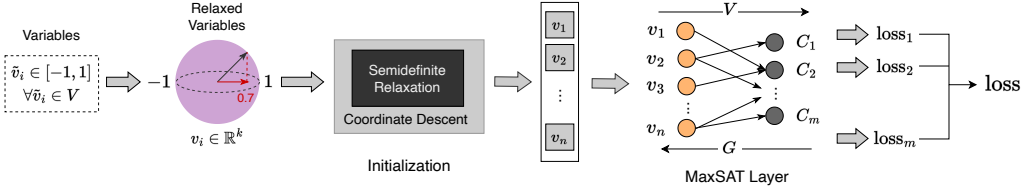
Algorithm 2 The backward pass of MaxSAT layer

Input: $v^k \in \mathbb{R}^n$ from forward pass, y^k from forward pass, CNF ϕ, learning rate λ.
Output: Gradient g^k of v^k, updated assignment v^{k+1}.

$g^k \leftarrow 0$;
if y^k is False **then**
  $\tilde{v}^k \leftarrow [v_i^k > 0 : i = 1, \ldots, n]$;
  $\phi' \leftarrow \phi(\tilde{v}_1^k, \ldots, \tilde{v}_n^k)$;
  $\bar{I} \leftarrow \{i \in [1, n] | v_i \in \bar{\phi}'\}$;
  for $i \in \bar{I}$ **do**
    $g_i \leftarrow \partial_{v_i^k}$loss;
  end for
  if $\exists\, g_i \neq 0$ **then**
    $g_i^k \leftarrow \arg\max_{i \in \bar{I}} \|g_i\|$
  else
    $v_i :=$ a random variable in a random falsified clause;
    $g_i^k :=$ sign(v_i);
  end if
end if
Update $v^{k+1} \leftarrow v^k - \lambda g^k$;

DiffSAT Flow

DiffSAT first generalizes SAT to MaxSAT, relaxes MaxSAT to SDP, which is solved by coordinate descent, and then refines the SDP solution to the final solution with the differential MaxSAT layer.



DiffSAT Overview

Experimental Results

Experimental Design

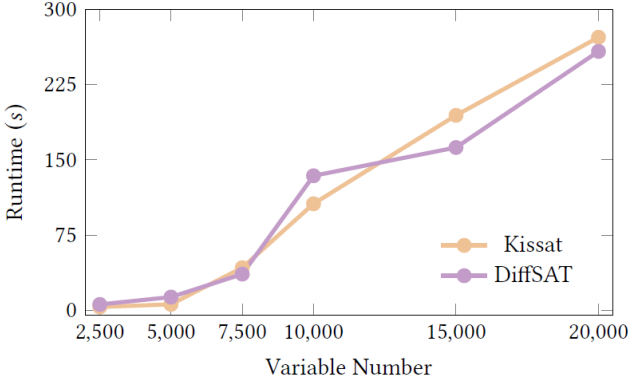
We aim to answer the following three questions through experiment:

RQ1: Can DiffSAT be generalized to handle large-scale SAT problems?

RQ2: Can DiffSAT demonstrate improved performance on complex datasets compared to the SOTA SAT solver?

RQ3: Is semidefinite initialization necessary for achieving the desired outcomes?

Performance on Random Dataset



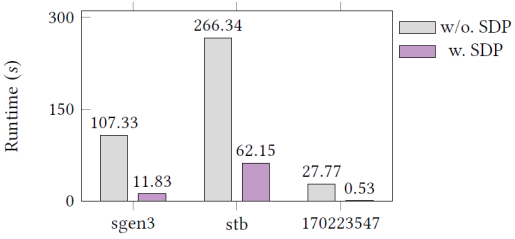
Performance comparison of Kissat and DiffSAT on random datasets.

Performance on Hard Dataset

Benchmark	# Variables	# Clauses	*CV*	Runtime (s)	
				SBVA-CaDiCaL	DiffSAT
rbsat-v1150c84314gyes10	1150	84314	73.32	>10000	98.76
sgen3-n260-s62321009-sat	260	884	3.4	>10000	11.83
Schur_160_5_d34	756	28445	37.63	>10000	23.81
SCPC-900-31	900	41714	46.35	>10000	0.23
SCPC-900-27	900	41618	46.24	700.78	0.34
em_11_3_4_cmp	8713	99699	11.44	663.96	8.65
170223547	322	1449	4.5	490.06	0.53
SCPC-1000-20	1000	51095	51.10	480.68	0.14
stb_588_138.apx_2_DS-ST	1176	14821	12.60	335.19	62.15
SCPC-700-86	700	27447	39.21	134.18	0.20
SCPC-800-44	800	34495	43.12	86.45	0.56

Table: Hard test cases in SAT Competition 2023 with extremely high CV ratio or tree/cyclic structures. SBVA-CaDiCaL is the TOP-1 solver in SATCOMP 2023.

Ablation Study on SDP Initialization



Comparison of DiffSAT with and without SDP initialization.

Table: Runtime of SDP initialization.

	sgen3	stb	170223547
Runtime(s)	3.45e-3	3.40e-1	1.45e-2
Ratio	< 1‰	5.47‰	2.74%

Conclusion

Conclusion

- DiffSAT demonstrates comparable scalability to traditional heuristic-based SAT solvers;
- DiffSAT is effective in finding solutions for complex SAT problems on which vanilla SAT solvers easily get stuck.
- Relaxation can be a good starting point.

THANK YOU!