Can Graph Neural Networks Learn to Solve the MaxSAT Problem? (Student Abstract)

Minghao Liu[1,3], **Pei Huang**[1,3], **Fuqi Jia**[1,3], **Fan Zhang**[2,3], **Yuchen Sun**[4], **Shaowei Cai**[1,3], **Feifei Ma**[1,2,3], **Jian Zhang**[1,3]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[2] Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences
[4] Inspir.ai
{liumh, maff, zj}@ios.ac.cn

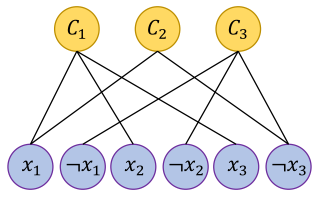
Abstract

The paper presents an attempt to bridge the gap between machine learning and symbolic reasoning. We build graph neural networks (GNNs) to predict the solution of the Maximum Satisfiability (MaxSAT) problem, an optimization variant of SAT. Two closely related graph representations are adopted, and we prove their theoretical equivalence. We also show that GNNs can achieve attractive performance to solve hard MaxSAT problems in certain distributions even compared with state-of-the-art solvers through experimental evaluation.

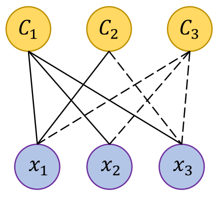
Introduction

The satisfiability problem of propositional logic formulas (denoted as SAT) has a significant impact on many areas of computer science and artificial intelligence, which is also the first problem proved to be NP-complete. The Maximum Satisfiability (MaxSAT) problem is an optimization variant of SAT, which aims to maximize the number of satisfied clauses. Thanks to recent advances in deep learning, especially those for non-Euclidean structures, there have been initial efforts to represent and solve combinatorial problems including SAT through data-driven approaches (Bengio, Lodi, and Prouvost 2021). Since the characteristics of problem instances in practice are usually domain-specific, these approaches are possible to produce better results than general-purpose solvers by learning from benchmarks.

NeuroSAT (Selsam et al. 2019) is a pioneering work which shows that GNNs can learn to predict the satisfiability of SAT problems, and leads to a series of improvements. In order to better explore the ability of GNNs on this task, we target the work as *learning to predict the solution of the MaxSAT problem*. The advantage is that as an optimization problem, we can measure the quality of solutions more quantitatively. First, we build two GNN models, called *GMS-N* and *GMS-E*, based on two kinds of clause variable-incidence graphs separately. These two GNNs are proven to be theoretically equivalent. Next, we train both models on synthetic MaxSAT benchmarks produced by different generating functions. The results show that both models achieve >0.99 approximation ratios on small instances with tens of variables, and they also have attractive generalization to much larger and more difficult problems.



(a) NS-CVIG



(b) ES-CVIG

Figure 1: Two kinds of clause variable-incidence graphs to represent the CNF formula $\phi := (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$ with 3 variables and 3 clauses.

Model Description

Clause Variable-Incidence Graph (CVIG) is a common representation for CNF formulas, which is a bipartite structure to establish the relationship between literals and clauses. There are mainly two kinds of CVIGs which have been used in previous works. The first one is *node-splitting CVIG (NS-CVIG)*, which represents the two relevant literals $(x_i, \neg x_i)$ as two nodes. The other one is *edge-splitting CVIG (ES-CVIG)*, which contains two types of edges, connecting the clauses with positive and negative literals separately. An example is shown in Figure 1.

The GNN models generally follow the message passing process. Considering the bipartite structure, in each layer the process is divided into two steps executed in sequence.

For NS-CVIG, the k-th layer of GNN is formalized as

$$\begin{aligned} C_j^{(k)} &= \mathrm{UPD_C}(C_j^{(k-1)}, \mathrm{AGG_L}(\{L_i^{(k-1)} | (l_i, c_j) \in E\})), \\ (L_i^{(k)}, \neg L_i^{(k)}) &= \mathrm{UPD_L}(L_i^{(k-1)}, \neg L_i^{(k-1)}, \\ &\qquad \mathrm{AGG_C}(\{C_j^{(k-1)} | (l_i, c_j) \in E\})), \end{aligned}$$

where $L_i^{(k)}, C_j^{(k)}$ are the embeddings of literal l_i and clause c_j in the k-th layer, and E is the set of edges.

For ES-CVIG, there are two sets of edges, which can be

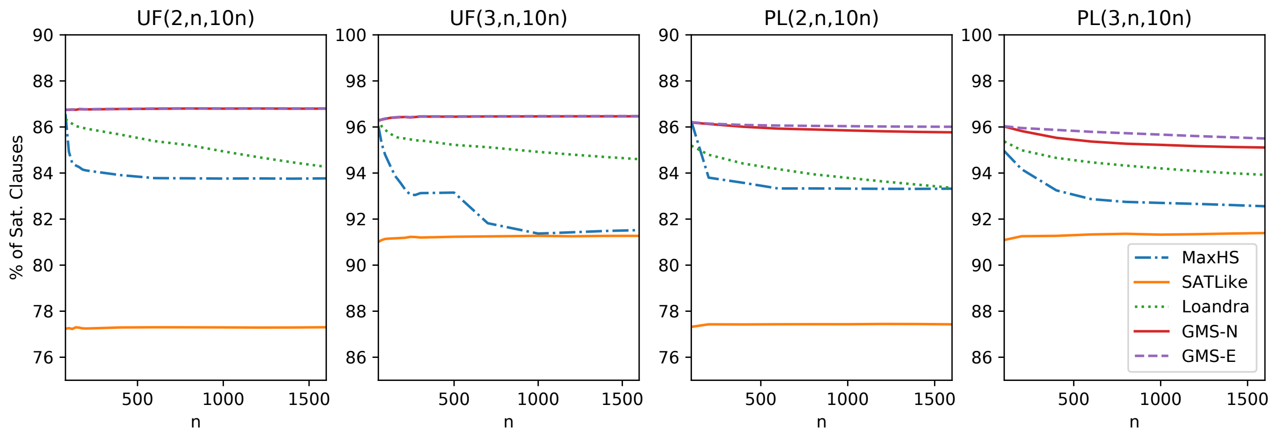


Figure 2: The average percentage of satisfied clauses by our GNN models and state-of-the-art MaxSAT solvers on a set of testing problems with up to 1,600 variables. GNNs are trained on benchmarks coming from the same distribution with the testing ones, but with only 60 variables. The results indicate that GNNs are promising and generalize well to much larger problems.

denoted as E^+ and E^-. The k-th layer of GNN is

$$\begin{aligned} C_j^{(k)} &= \mathrm{UPD}_\mathrm{C}(C_j^{(k-1)}, \mathrm{AGG}_\mathrm{L}^+(\{L_i^{(k-1)} | (l_i, c_j) \in E^+\}), \\ &\qquad \mathrm{AGG}_\mathrm{L}^-(\{L_i^{(k-1)} | (l_i, c_j) \in E^-\})), \\ L_i^{(k)} &= \mathrm{UPD}_\mathrm{L}(L_i^{(k-1)}, \mathrm{AGG}_\mathrm{C}^+(\{C_j^{(k-1)} | (l_i, c_j) \in E^+\}), \\ &\qquad \mathrm{AGG}_\mathrm{C}^-(\{C_j^{(k-1)} | (l_i, c_j) \in E^-\})). \end{aligned}$$

Finally, to adapt the models to solve the MaxSAT problem, we use a binary classifier PRED to generate a number $p_i \in [0, 1]$ for each variable x_i after K GNN layers. The models are trained by minimizing the binary cross entropy loss against the ground truth.

Specifically, we establish the equivalence of these two GNN models based on NS-CVIG and ES-CVIG:

Proposition 1 *Given a GNN $\mathcal{N}_1$ based on NS-CVIG, there exists a GNN $\mathcal{N}_2$ based on ES-CVIG, such that for every MaxSAT instance ϕ, $\mathcal{N}_1(\phi) = \mathcal{N}_2(\phi)$, and vice versa.*

Summary and Future Work

We implement the above two GNN models and abbreviate them as *GMS-N* and *GMS-E*. Given the number of variables n, the number of clauses m and the clause size k, we produce the synthetic MaxSAT problems by running two representative generators with different distributions: (a) *Uniform distribution* (Mitchell, Selman, and Levesque 1992), which samples the clauses uniformly and independently. The datasets are denoted by `UF(`k,n,m`)`; (b) *Power-Law distribution* (Ansótegui, Bonet, and Levy 2009), a non-uniform generating function which aims to simulate the characteristics of real-world (industrial) instances. The datasets are denoted by `PL(`k,n,m`)`.

By training GMS-N and GMS-E on 4 datasets ($n = 60$) separately, we find that the average approximation ratios are >0.99 on the validation sets in all cases, and both models generalize well to a broad range of clause-variable proportions. We also generate a series of testing sets with the same distribution as the training sets, but the number of variables $n \in [100, 1600]$ is much larger. Since the optimal objectives of these large problems are clearly unsolvable, we compare the percentage of satisfied clauses instead of the approximation ratio. In addition to the GNN models, we also run 3 efficient MaxSAT solvers: a complete solver *MaxHS* and 2 incomplete solvers *Loandra* and *SATLike*, which have won top awards in the unweighted track of the MaxSAT competition. For each instance, every solver is executed for 5 minutes and the best found solution is recorded. From Figure 2, we surprisingly find that GMS-N and GMS-E could generalize very well on the large problems, even compared with these state-of-the-art solvers. This suggests that GNNs are expected to be promising alternatives to help solve more challenging MaxSAT problems.

For future work, we plan to investigate the capability of GNNs to solve partial MaxSAT problem, a more challenging variant in which all hard clauses must be satisfied. Extension, proof and implementation details can be found at https://github.com/minghao-liu/GMS.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (NSFC) under grant No.61972384, and the Youth Innovation Promotion Association CAS under grant No.Y202034.

References

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2009. Towards industrial-like random SAT instances. In *IJCAI*, 387–392.

Bengio, Y.; Lodi, A.; and Prouvost, A. 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2): 405–421.

Mitchell, D. G.; Selman, B.; and Levesque, H. J. 1992. Hard and easy distributions of SAT problems. In *AAAI*, 459–465.

Selsam, D.; Lamm, M.; Bünz, B.; Liang, P.; de Moura, L.; and Dill, D. L. 2019. Learning a SAT solver from single-bit supervision. In *ICLR*.