Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

Learning to Generate Industrial SAT Instances

Haoze Wu
Department of Computer Science
Stanford University
Stanford, CA 94305
U.S.A.
haozewu@stanford.edu

Raghuram Ramanujan
Dept. of Mathematics and Computer Science
Davidson College
Davidson, NC 28035
U.S.A.
raramanujan@davidson.edu

Abstract

In this paper, we present SATGEN, the first implicit model that generates Boolean Satisfiability formulas which resemble instances that arise in real-world industrial settings. Our approach uses unsupervised machine learning techniques to create new formulas by mimicking the structural properties of a given input formula Φ. We proceed in two phases: first, we construct the Literal Incidence Graph (LIG) of Φ. This is used by a Generative Adversarial Network to generate new LIGs that exhibit graph-theoretic properties similar to those of the LIG of Φ. In the second phase, we extract a formula Φ' whose LIG would correspond to the generated graph. Generating such a formula is equivalent to finding a minimal clique edge cover of the given graph, which we tackle efficiently using a greedy hill-climbing algorithm. We verify experimentally that our approach generates formulas that closely resemble a given real-world SAT instance, as measured by a range of different metrics.

1 Introduction

The *propositional Boolean Satisfiability problem* (SAT) is the canonical NP-complete problem and is of central importance to computer science. Developing and evaluating practical SAT-solvers relies on extensive empirical testing on a diverse suite of benchmark problems. SAT instances that arise in a variety of real-world settings, such as circuit verification and planning, form a key component of such test sets and have very different structural properties from instances that are synthetically generated (Ansótegui, Bonet, and Levy 2009). However, the supply of such "industrial" SAT problems is limited. Indeed, the problem of developing an instance generator that could create arbitrarily many artificial SAT problems that display the same characteristics as their real-world counterparts has been identified as one of ten key challenges in propositional reasoning and search (Selman, Kautz, and McAllester 1997). We study this very problem in this paper.

2 The SATGEN System

Prior work in the area of pseudo-industrial SAT instance generation has focused on developing parameterized models that capture some specific graph-theoretic property, like modularity (Giráldez-Cru and Levy 2015) or the presence of scale-free structures (Giráldez-Cru and Levy 2017). While such prescribed models permit theoretical analysis, they do not simultaneously capture all the essential characteristics of industrial SAT formulas. In this work, we take an implicit modeling approach instead, and present a system that captures a wide range of essential (possibly yet unknown) graph-based features without specifically targeting any one of them. Our SATGEN system takes as input a real-world industrial SAT instance Φ and generates new formulas that mimic the structure of the original. It works in four stages:

1. We first extract the Literal Incidence Graph (LIG) of Φ. In this graph G, each node corresponds to a literal in Φ, with edges connecting two literals that co-occur in the same clause in Φ.
2. We then train a *Generative Adversarial Network* (GAN) using the NetGAN algorithm (Bojchevski et al. 2018), to model this graph's topology.
3. The learned model is used to generate a new LIG G', that exhibits similar structural properties to G.
4. Finally, we extract a new SAT formula Φ' from G' using a greedy hill-climbing approach.

We note that step 4 above presents a difficulty: an LIG does not uniquely map to a SAT formula for it only captures which literals co-occur in a clause, and not what the clauses themselves are. For instance, the formulas $\Phi_1 = (v_1 \vee v_2 \vee \neg v_3)$ and $\Phi_2 = (v_1 \vee v_2) \wedge (v_2 \vee \neg v_3) \wedge (v_1 \vee \neg v_3)$ have the same LIG. However, imposing some sensible constraints — specifically, that the generated formula cannot contain duplicated clauses, unit clauses, or subsumed clauses — allows us to extract reasonable SAT formulas given an LIG, a problem that is equivalent to finding a minimal clique edge cover of the generated graph (proof omitted).

However, not all minimal clique edge covers correspond to desirable formulas. We require that the generated formula Φ' contain the same number of clauses as the original formula Φ, while also matching the diversity of clause lengths exhibited by Φ (which often follows a power-law distribution in real-world formulas). The following result (proof omitted) suggests a way towards achieving this goal.

Lemma 1. *In a SAT formula Φ, for any clause C of length k ($k > 2$), there exist three clauses C_1, C_2, and C_3, each of*

	# clauses	LIG Clust.	LIG Mod.	VCG Mod.	α_v	λ_v	α_c	λ_c
ssa2670-141	377	0.351	0.559	0.647	4.84	0.265	3.56	0.783
SATGEN	**377**	**0.340**	**0.542**	**0.632**	**4.64**	0.183	5.28	1.012
PS	352	0.464	0.584	0.731	4.39	**0.210**	**4.07**	**0.693**
CA	375	0.247	0.499	0.629	7.06	0.394	-	-

Table 1: Median clustering coefficients, modularities, and estimated degree distribution parameters for variables (α_v, λ_v) and clauses (α_c, λ_c) of formulas generated to mimic benchmark ssa2670-141, using three different generators. For each property, the model with the closest statistics to the original formula is identified in boldface.

length $k-1$, such that if we replace C with $C_1 \wedge C_2 \wedge C_3$ in Φ, the LIG of Φ remains unchanged.

Specifically, we start by enumerating all cliques in the generated LIG that are of size smaller than some n (we set $n = 15$ as the real-world formulas we considered rarely contained longer clauses). We then use greedy hill-climbing over the set of generated cliques to find a minimal clique edge cover. This edge cover is then repeatedly expanded by breaking down a clique chosen uniformly at random according to the procedure described in Lemma 1, until the desired number of clauses is reached.

3 Experiments and Results

We evaluated SATGEN on industrial SAT benchmarks from SATLIB (Hoos and Stützle 2000) and past SAT competitions. We used the SatElite pre-processor (Eén and Biere 2005) to remove subsumed, unit, and duplicated clauses from these formulas. The number of nodes in the LIGs that we trained on ranged from 182 to 2244, with between 919 to 12582 edges. We compare the properties of the formulas generated by SATGEN to those generated by two other pseudo-industrial SAT formula generators: the Community Attachment (CA) model that generates formulas with a desired modularity (Giráldez-Cru and Levy 2015) and the Popularity-Similarity (PS) model that generates formulas with scale-free structures (Giráldez-Cru and Levy 2017). In the interests of space, we only present our results on one instance — ssa2670-141, a circuit fault analysis benchmark — though similar results were obtained with other formulas as well. Benchmark ssa2670-141 originally contained 986 variables and 2315 clauses. After pre-processing with SatElite, the formula is reduced to 91 variables and 377 clauses. The clause lengths range from 2 to 8 and the average clause length is about 3. The LIG of the benchmark contains 182 nodes and 1062 edges. Table 1 summarizes our results. The statistics shown for each generator represent the median of measurements made on 100 generated formulas.

Overall, SATGEN is the only generator that consistently captures all of the key graph theoretic properties of the original formula. Since SATGEN learns from the LIG of the original formula, it is not surprising that the formulas it generates have a similar LIG clustering coefficient and modularity statistics as the original. Interestingly, however, those formulas also match the statistics of the original formula when other graph projections, such as the Variable-Clause Graph (VCG), are employed. Table 1 also includes measures of scale-free structure exhibited by the variable degrees (α_v and λ_v) and clause degrees (α_c and λ_c) in the VCG of the generated formulas. The definitions of these measures are found in (Ansótegui, Bonet, and Levy 2009). While not as effective as PS, which directly fits these statistics, the formulas generated by SATGEN nonetheless capture scale-free characteristics well. We finally also note that SATGEN generates diverse formulas: the average clause overlap between a generated formula and the original is only 14%, while the average clause overlap between two generated formulas is only 12%.

4 Future Work and Conclusions

In this paper, we introduced SATGEN which, in contrast to previous pseudo-industrial SAT formula generators such as the CA and PS models, captures all the graph-based properties of a given SAT formula without targeting any individual one. There are two avenues for further research. First, the NetGAN learning algorithm can currently only handle graphs with a few thousand nodes; we are thus unable to generate very large SAT instances with tens or hundreds of thousands of variables. It should however be noted that many seemingly large industrial benchmarks undergo significant shrinkage when pre-processed. Second, the generated formulas are often easier to solve than the original real-world formulas on which they are modeled, though this is also true of comparable formulas generated by CA and PS. Nevertheless, we believe that our preliminary exploration of a graph-based implicit SAT formula generator shows promise.

References

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2009. On the structure of industrial sat instances. In *CP 2009*, 127–141. Springer Berlin Heidelberg.

Bojchevski, A.; Shchur, O.; Zügner, D.; and Günnemann, S. 2018. NetGAN: Generating graphs via random walks. In *ICML 2018*, volume 80, 609–618. PMLR.

Eén, N., and Biere, A. 2005. Effective preprocessing in sat through variable and clause elimination. In *SAT '05*, 61–75. Springer Berlin Heidelberg.

Giráldez-Cru, J., and Levy, J. 2015. A modularity-based random sat instances generator. In *IJCAI '15*, 1952–1958. AAAI Press.

Giráldez-Cru, J., and Levy, J. 2017. Locality in random sat instances. In *IJCAI '17*, 638–644. AAAI Press.

Hoos, H. H., and Stützle, T. 2000. SATLIB: An online resource for research on sat. In *SAT '00*, 283–292. IOS Press.

Selman, B.; Kautz, H.; and McAllester, D. 1997. Ten challenges in propositional reasoning and search. In *IJCAI '97*, 50–54. Morgan Kaufmann Publishers Inc.