Artificial Intelligence 238 (2016) 119–134

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Generating SAT instances with community structure ☆

Jesús Giráldez-Cru *, Jordi Levy *

Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB, Bellaterra, Spain

ARTICLE INFO

Article history:
Received 1 December 2015
Received in revised form 31 May 2016
Accepted 2 June 2016
Available online 7 June 2016

Keywords:
Satisfiability
SAT solver
SAT generator
Graph modularity

ABSTRACT

Nowadays, modern SAT solvers are able to efficiently solve many *industrial*, or *real-world*, SAT instances. However, the process of development and testing of new SAT solving techniques is conditioned to the finite and reduced number of known industrial benchmarks. Therefore, new models of random SAT instances generation that capture realistically the features of real-world problems can be beneficial to the SAT community. In many works, the structure of industrial instances has been analyzed representing them as graphs and studying some of their properties, like *modularity*.

In this work, we use the notion of modularity to define a new model of generation of random SAT instances with community structure, called *Community Attachment*. For high values of modularity (i.e., clear community structure), we realistically model pseudo-industrial random SAT formulas. This model also generates SAT instances very similar to classical random formulas using a low value of modularity. We also prove that the phase transition point, if exists, is independent on the modularity. We evaluate the adequacy of this model to real industrial SAT problems in terms of SAT solvers performance, and show that modern solvers do actually exploit this community structure. Finally, we use this generator to observe the connections between the modularity of the instance and some components of the solver, such as the variable branching heuristics or the clause learning mechanism.

1. Introduction

The Boolean Satisfiability Problem (SAT) is one of the most studied problems in Computer Science. It is the first known NP-complete problem, as proved by S. Cook in 1971 [15]. However, many application problems are nowadays encoded into SAT instances, and efficiently solved by modern SAT solvers. Namely, these solvers are the so known Conflict-Driven Clause Learning (CDCL) SAT solvers. In the last years, these solvers have been the dominant *technique* in the SAT community to solve application problems, including a wide variety of domains, such as cryptography, hardware and software verification, planning, or scheduling, among others.

It is accepted that random and industrial SAT instances have a distinct nature. While random formulas can be easily generated on demand, the set of industrial benchmarks, which encode real-world problems, is limited. The problem of generating realistic pseudo-industrial random instances is stated in [39] as one of the most important *challenges* for the next few years: *"Challenge 10: Develop a generator for problem instances that have computational properties that are more similar to*

☆ This work is partially supported by the MINECO/FEDER project RASO (TIN2015-71799-C2-1-P) and the CSIC project 201450E045.
* Corresponding authors.
E-mail addresses: jgiraldez@iiia.csic.es (J. Giráldez-Cru), levy@iiia.csic.es (J. Levy).
URLs: http://www.iiia.csic.es/~jgiraldez (J. Giráldez-Cru), http://www.iiia.csic.es/~levy (J. Levy).

http://dx.doi.org/10.1016/j.artint.2016.06.001
0004-3702/

real world instances". This challenge is also stated in [27,16]. The main motivation of this challenge is improving the process of development and testing of SAT solvers, and their possible specialization.

There exist a wide variety of works on the analysis of the nature of industrial SAT instances. The intuition is that these formulas have some kind of *structure*, which is exploited by CDCL SAT solvers. In many of these works, SAT formulas are represented as graphs, and some (graph) features are studied. The classical Erdös–Rényi model has been extensively used for generating random graphs. In this model, the degree of nodes follows a binomial distribution, with small variability. This is exactly the case of classical random formulas. However, the structure of most real-world networks cannot be described with this classical model, and therefore, new models have been defined. For instance, *Preferential Attachment* [12] is used to explain the scale-free structure of some real networks, where the nodes degree follows a power-law distribution, with big variability. *Similarity* has also been proposed as a second mechanism, that together with *popularity* or preferential attachment, may result into complex networks with an hyperbolic topology [38]. In [2] it is described an hyperbolic graph generator. When SAT instances are modeled as graphs, many graph properties can be analyzed, such as the small-world property [43], the scale-free structure [5,6], the eigenvector centrality [26], or the self-similarity [3]. Moreover, some notions of structure have been already used to better understand certain components of modern SAT solvers [29,28,4]. The structure of industrial SAT instances has also been proposed as responsible for the smaller tree-like resolution space needed to refute them [8].

In this paper, we focus on the notion of *modularity* of a graph [34]. In a graph with high modularity (i.e., clear *community structure*), we can find a partition of its nodes into communities such that most of its edges connect nodes of the same community. In [9], it is analyzed the modularity of the industrial benchmarks used in SAT competitions. In that paper, it is shown that most industrial SAT formulas exhibit a high modularity. On the contrary, randomly generated instances have a very low modularity. The community structure has been shown correlated with the runtime of CDCL SAT solvers [36,37]. Moreover, it has been used to improve the performance of some solvers [31,42,10,33].

One important motivation for the development of *pseudo-industrial* SAT instances generators is to *isolate* some known properties of these real-world problems. This allows us to study the impact of these properties on the performance and behavior of SAT solvers. This approach has already been used in [7]. In that work, authors present a random SAT instances generator which takes into account the scale-free structure of real-world SAT instances to generate formulas in which the number of variable occurrences follows a power-law distribution. Using it, they observe that CDCL SAT solvers focus their decisions on the most frequent variables. In the case of community structure, similar questions also arise. For instance, do SAT solvers concentrate their decisions on variables of the same (or few) communities? Do the conflicts found by the solver relate variables of the same community? How does the activity of each community evolve along the execution of the search? Answering these questions may help to better understand the different ingredients of modern SAT solvers and their impact on the solving process, with the long-term aim of improving them.

The main contribution of this work is a new model of generation of random SAT instances based on the notion of modularity.[1] With this new model, we can generate formulas for any given value of modularity. For a high modularity, the resulting instance is more adequate to model industrial problems than classical random formulas. On the contrary, with a low modularity we can generate SAT instances very similar to classical random ones. We show that this model works appropriately for different input values of number of variables n and clauses m. We also show that, if there exists a phase transition point SAT-UNSAT when the ratio clause/variable is increased, then it does not depend on the modularity. We give empirical evidence that the performance of SAT solvers is consistent with the expected properties of the generated formulas, i.e. SAT solvers *specialized* in industrial problems perform better in high modular instances than SAT solvers *specialized* in random formulas, and vice versa. Finally, we use this generator to answer the questions stated in the previous paragraph. In particular, we analyze the relations between the community structure and the branching decision heuristics, and the mechanism of learning new clauses from the conflicts the solver finds along its execution. A preliminary version of this paper was published as [20].

The rest of the paper proceeds as follows. We first reference some related works on the generation of pseudo-industrial SAT instances in Section 2. After some preliminaries on modularity of graphs in Section 3, we describe the generation model in Section 4. In Section 5, we show that this model works appropriately for different input values. In Section 6, we analyze the phase transition point of the instances generated by our model. In Section 7, we show the adequacy between the performance of SAT solvers and the properties of the generated instances. In Section 8, we use this generator to analyze some components of a CDCL SAT solver. Finally, we conclude in Section 9 and propose some future work.

2. Related work

There already exist some works facing the problem of generating *pseudo-industrial* random instances. We distinguish between those works focused on particular problems domains from those others that generalize common properties of industrial instances.

In the case of particular problems domains, we have the following works. Gomes and Selman [21] proposed a new model of SAT benchmarks resulting from introducing random perturbations into structured problems. These benchmarks encode

[1] This modularity-based random SAT instances generator is available in http://www.iiia.csic.es/~jgiraldez/software.