Benchmarks Description

Jesús Giráldez-Cru[*1] and Jordi Levy[†1]

[1]Artificial Intelligence Research Institute (IIIA-CSIC), Spain

Introduction

It is accepted that random k-CNF and industrial SAT instances have a distinct nature. While random formulas can be easily generated on demand, the set of industrial instances, which encode real-world problems, is limited. The problem of generating realistic pseudo-industrial random instances is stated in [9, 6, 4] as one of the most important *challenges* for the next few years. The main motivation of this challenge is improving the process of development and test of SAT solvers, and their possible specialization.

Industrial SAT formulas exhibit a clear community structure (i.e. high modularity Q) [1]. This means that, representing formulas as graphs, we can find a partition of the formula into communities with many edges between nodes of the same community (i.e., many clauses relating variables of the same community), and few edges connecting distinct communities. This property is very characteristic in real-world problems in contrast to randomly generated instances, where modularity is very low. In the context of SAT, it has been shown that the community structure is correlated with the runtime of CDCL SAT solvers [8]. Moreover, it has also been used to improve the performance of some solvers [7, 10].

Community Attachment

In [5], it is presented a method to generate realistic pseudo-industrial random SAT instances: the **Community Attachment** model. This generator is based on the community structure, and it generates formulas for any given value of modularity. Therefore, for a high value of modularity, the resulting formula is more adequate to model industrial problems than classical random k-CNF. However, it also generates SAT instances similar to the random k-CNF when the value of the modularity is low.

The Community Attachment model is parametric in a probability P and a partition C of the set of variables. In this model all variables of a clause belong to the same

*jgiraldez@iiia.csic.es
†levy@iiia.csic.es

community with probability P, and with probability $1 - P$ they all belong to distinct communities. In particular, the probability P is taken as:

$$P = Q + \frac{1}{c} \tag{1}$$

Using Eq. 1, the generated formulas will have an expected modularity close to Q. Recall that industrial SAT instances are characterized by a high modularity $Q > 0.7$ in most cases, while the modularity of random SAT formulas is very small $Q \approx 0.3$. Moreover, the number of communities c is usually in the interval $(10, 100)$ [1].

We address the reader to [5] for further references about the Community Attachment generation method.

Set of Submitted Instances

Using the Community Attachment model described in the previous section[1], we create a family of 200 pseudo-industrial random SAT instances, with a modularity $Q = 0.8$, a number of communities $c = 40$, a number of variables $n = 2200$, and a number of clauses $m = 9086$. This clause/variable ratio $m/n = 4.13$ is the phase transition point of this family (found experimentally).

When the value of modularity Q used to generate the instances is very high, there exist some instances having a very small refutation. This happens because most of the clauses relate variables of the same community, and hence it is more likely to find a small unsatisfiable set of clauses which only contains variables of one or few communities. Therefore, it is recommendable to filter this family in order to remove those *easy* instances. To do so, we use solvers Lingeling [3] and Glucose [2], and we remove those SAT instances solved by any of them in less than 20 minutes.

The remaining set of SAT instances contains a total 44 instances: 26 are known to be satisfiable, 4 are unsatisfiable, and the remaining 14 instances are indeterminate by Lingering and Glucose in a timeout of 3 hours.

References

[1] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. The community structure of SAT formulas. In *Proc. of the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT'2012)*, pages 410–423, 2012.

[2] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI'2009)*, pages 399–404, 2009.

[3] Armin Biere. Lingeling essentials, A tutorial on design and implementation aspects of the the SAT solver Lingeling. In *Proc. of Pragmatics of SAT (POS'2014)*, 2014.

[1]This generator is available in `http://www.iiia.csic.es/~jgiraldez/software`.

[4] Rina Dechter. *Constraint Processing.* Morgan Kaufmann, 2003.

[5] Jesús Giráldez-Cru and Jordi Levy. A modularity-based random SAT instances generator. In *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI'2015)*, 2015.

[6] Henry A. Kautz and Bart Selman. Ten challenges redux: Recent progress in propositional reasoning and search. In *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP'2003)*, pages 1–18, 2003.

[7] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Community-based partitioning for maxsat solving. In *Proc. of the 16th International Conference on Theory and Applications of Satisfiability Testing (SAT'2013)*, pages 182–191, 2013.

[8] Zack Newsham, Vijay Ganesh, Sebastian Fischmeister, Gilles Audemard, and Laurent Simon. Impact of community structure on SAT solver performance. In *Proc. of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT'2014)*, pages 252–268, 2014.

[9] Bart Selman, Henry A. Kautz, and David A. McAllester. Ten challenges in propositional reasoning and search. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI'1997)*, pages 50–54, 1997.

[10] Tomohiro Sonobe, Shuya Kondoh, and Mary Inaba. Community branching for parallel portfolio SAT solvers. In *Proc. of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT'2014)*, pages 188–196, 2014.