The Community Structure of SAT Formulas*

Carlos Ansótegui¹, Jesús Giráldez-Cru², and Jordi Levy²

¹ DIEI, Univ. de Lleida
carlos@diei.udl.cat
² IIIA-CSIC
{jgiraldez,levy}@iiia.csic.es

Abstract. The research community on complex networks has developed techniques of analysis and algorithms that can be used by the SAT community to improve our knowledge about the structure of industrial SAT instances. It is often argued that modern SAT solvers are able to exploit this hidden structure, without a precise definition of this notion.

In this paper, we show that most industrial SAT instances have a high modularity that is not present in random instances. We also show that successful techniques, like learning, (indirectly) take into account this community structure. Our experimental study reveal that most learnt clauses are local on one of those modules or communities.

1 Introduction

In recent years, SAT solvers efficiency solving industrial instances has undergone a great advance, mainly motivated by the introduction of lazy data-structures, learning mechanisms and activity-based heuristics [11,18]. This improvement is not shown when dealing with randomly generated SAT instances. The reason for this difference seems to be the existence of a structure in industrial instances [25].

In parallel, there have been significant advances in our understanding of complex networks, a subject that has focused the attention of statistical physicists. The introduction of these network analysis techniques could help us to understand the nature of SAT instances, and could contribute to further improve the efficiency of SAT solvers. Watts and Strogatz [24] introduce the notion of small world, the first model of complex networks, as an alternative to the classical random graph models. Walsh [23] analyzes the small world topology of many graphs associated with search problems in AI. He also shows that the cost of solving these search problems can have a heavy-tailed distribution. Gomes et al. [14,15] propose the use of randomization and rapid restart techniques to prevent solvers from falling on the long tail of such kinds of distributions.

The notion of structure has been addressed in previous work [14,16,13,17,3]. In [22] it is proposed a method to generate more realistic random SAT problems based on the notions of characteristic path length and clustering coefficient. Here

* This research has been partially founded by the CICYT research projects TASSAT (TIN2010-20967-C04-01/03) and ARINF (TIN2009-14704-C03-01).
we use a distinct notion of modularity. In \cite{6}, it is shown that many SAT instances can be decomposed into connected components, and how to handle them within a SAT solver. They discuss how this component structure can be used to improve the performance of SAT solvers. However, their experimental investigation shows that this is not enough to solve more efficiently SAT instances. The notion of community is more general than the notion of connected components. In particular, it allows the existence of (a few) connections between communities. As we discuss later, industrial SAT instances use to have a connected component containing more than the 99% of the variables. Also, in \cite{1} some techniques are proposed to reason with multiple knowledge bases that overlap in content. In particular, they discuss strategies to induce a partitioning of the axioms, that will help to improve the efficiency of reasoning.

In this paper we propose the use of techniques for detecting the community structure of SAT instances. In particular, we apply the notion of modularity \cite{19} to detect these communities. We also discuss how existing conflict directed clause learning algorithms and activity-based heuristics already take advantage, indirectly, of this community structure. Activity-based heuristics \cite{18} rely on the idea of giving higher priority to the variables that are involved in (recent) conflicts. By focusing on a sub-space, the covered spaces tend to coalesce, and there are more opportunities for resolution since most of the variables are common.

2 Preliminaries

Given a set of Boolean variables $X = \{x_1, \ldots, x_n\}$, a literal is an expression of the form $x_i$ or $\neg x_i$. A clause $c$ of length $s$ is a disjunction of $s$ literals, $l_1 \lor \ldots \lor l_s$. We say that $s$ is the size of $c$, noted $|c|$, and that $x \in c$, if $c$ contains the literal $x$ or $\neg x$. A CNF formula or SAT instance of length $t$ is a conjunction of $t$ clauses, $c_1 \land \ldots \land c_t$.

An (undirected) graph is a pair $(V, w)$ where $V$ is a set of vertices and $w : V \times V \to \mathbb{R}^+$ satisfies $w(x, y) = w(y, x)$. This definition generalizes the classical notion of graph $(V, E)$, where $E \subseteq V \times V$, by taking $w(x, y) = 1$ if $(x, y) \in E$ and $w(x, y) = 0$ otherwise. The degree of a vertex $x$ is defined as $\deg(x) = \sum_{y \in V} w(x, y)$. A bipartite graph is a tuple $(V_1, V_2, w)$ where $w : V_1 \times V_2 \to \mathbb{R}^+$.

Given a SAT instance, we construct two graphs, following two models. In the Variable Incidence Graph model (VIG, for short), vertices represent variables, and edges represent the existence of a clause relating two variables. A clause $x_1 \lor \ldots \lor x_n$ results into $\binom{n}{2}$ edges, one for every pair of variables. Notice also that there can be more than one clause relating two given variables. To preserve this information we put a higher weight on edges connecting variables related by more clauses. Moreover, to give the same relevance to all clauses, we ponderate the contribution of a clause to an edge by $1/\binom{n}{2}$. This way, the sum of the weights of the edges generated by a clause is always one. In the Clause-Variable Incidence Graph model (CVIG, for short), vertices represent either variables or clauses, and edges represent the occurrence of a variable in a clause. Like in the VIG model, we try to assign the same relevance to all clauses, thus every edge