iSat: Structure Visualization for SAT Problems

Ezequiel Orbe, Carlos Areces, and Gabriel Infante-López[*]

Grupo de Procesamiento de Lenguaje Natural
FaMAF, Universidad Nacional de Córdoba, Argentina
{orbe,areces,gabriel}@famaf.unc.edu.ar

Abstract. We present **iSat**, a Python command line tool to analyze and find structure in propositional satisfiability problems. **iSat** offers an interactive shell to control propositional SAT solvers and generate graph representations of the internal structure of the search space explored by them for visualization, with the final aim of providing a unified environment for propositional solving experimentation. **iSat** was designed to enable simple integration of both new SAT solvers and new visualization graphs and statistics with a minimum of coding overhead.

1 Introduction

iSat[1] (interactive SAT) is a command line tool implemented in Python that helps users to analyze and find structure in propositional satisfiability problems. It can be used, for example, to investigate the behavior of different provers over a given test set. The main service offered by **iSat** is a unified interface for experimentation with different propositional SAT solvers and visualization graphs. Moreover, it can be use to mechanize the repetitive tasks often performed during the development of SAT solvers (e.g., fine tuning heuristics) or the selection of the appropriate configuration options for a given solver working on a particular satisfiability problem. **iSat** computes different visualization graphs (e.g., Variable-Clause , Variable, Interaction, etc.) over the current clause set at different points during the exploration of the search space, and computes related statistics over these graphs (degree mean/max/min/standard deviation, clique number, clustering, number of cliques, etc.). **iSat** was designed to facilitate the integration of new SAT solvers, visualization graphs and statistics with a minimum of coding overhead. **iSat** is distributed under a GPL license and currently supports two SAT solvers out of the box: Minisat [3] and CryptoMinisat [11].

1.1 A Brief Overview on SAT Solving

Propositional satisfiability is the problem of deciding whether there exists a Boolean assignment to variables, such that all clauses in a given propositional formula evaluate to true. Despite its complexity, current SAT solvers (e.g., [3,8,5]) efficiently solve many instances of the SAT problem.

[*] Consejo Nacional de Investigaciones Científicas y Técnicas.

[1] Available at `https://cs.famaf.unc.edu.ar/~ezequiel/software/isat/`

Current SAT solvers can be classified in two broad classes: *incomplete* and *complete* systems. Incomplete solvers perform different kinds of stochastic local search to find a satisfying valuation; if the search is successful, satisfiability is established, but search failure does not imply unsatisfiability. Complete SAT solvers, on the other hand, perform an exhaustive, systematic search, and hence can establish both satisfiability and unsatisfiability. Most of them implement variants of the Davis-Putnam-Logemann-Loveland algorithm (DPLL) [2,1]. Complete SAT solvers can be further classified into conflict-driven and look-ahead. Conflict-driven SAT solvers augment DPLL with conflict analysis, clause learning, non-chronological backtracking and restarts as its principal components. Look-ahead SAT solvers, are also based on DPLL but invest substantial efforts choosing first the branching variable to be used (the different choice options are called decision heuristics) and then the truth value this variable will be assigned (using so called direction heuristics) aiming to achieve the largest reduction of the remaining search space. [4] provides an excellent overview of the area.

The SAT solving community is large and very active, with strong industrial involvement on application areas like planning [6], test pattern generation [7], etc. As a result of this demand, new algorithms and heuristics are being constantly developed, and the available solvers tuned to obtain the best behavior on particular problem domains. But interacting with solvers to gather statistics and explore their behavior when solving particular problems in order to find the best configuration parameters for a given problem class is a burdensome task.

iSat is a command line tool developed in Python that provides an interactive shell for multiple SAT solvers and is capable of producing visualization graphs and statistics, with the final aim of providing a unified environment for SAT solving experimentation. Currently, **iSat** provides access to the Minisat [3] and the CryptoMinisat [11] solvers; produces Variable-Clause graphs, Variable graphs, Literal-Clause graphs, Literal graphs and Interaction graphs that can be exported in `gml` and `dot` format and can be visualized using, for example, Cytoscape[2]; and computes statistics over these graphs like degree mean/max/min/standard deviation, clique number, clustering and number of cliques. Moreover, the architecture of **iSat** has been designed to enable simple integration of new SAT solvers and new analysis tools (i.e., new visualizations and statistics).

2 A Sample Session

We will describe a typical session with **iSat** to illustrate its capabilities. The screen capture of the interaction can be seen in Figure 1.

Consider the case of a researcher who wants to visualize how the structure of a pigeon hole problem instance evolves during search. She suspects that if she can identify some structural properties of the problem, she could develop new

[2] `http://www.cytoscape.org/`