Testing SAT Heuristics

Natalia Perina, Pranjit Kalita

Introduction

Motivation - Learn how SAT solver decision heuristics affect solver performance on different instances.

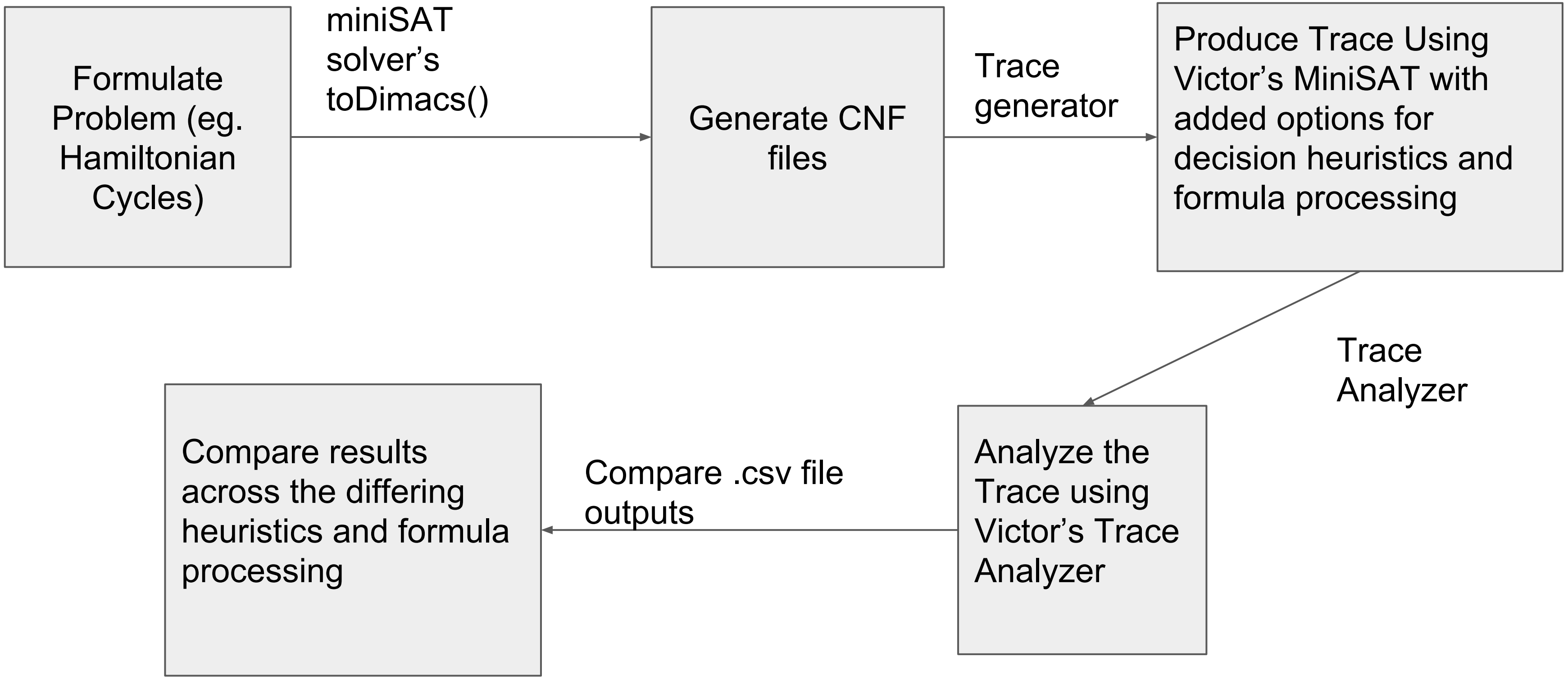
Problem Statement - Use Victor Ying’s trace generator and trace analyzer to measure SAT performance when using different decision heuristics and different formula processing techniques on various benchmark problems:

VSIDS vs. DLIS vs. Random Decision

Pre Processing vs. No Pre Processing

Hamiltonian Cycles & K-Coloring, SAT & UNSAT

Overview


Formulate Problem (eg. Hamiltonian Cycles)
miniSAT solver's toDimacs()
Generate CNF files
Trace generator
Produce Trace Using Victor's MiniSAT with added options for decision heuristics and formula processing
Trace Analyzer
Analyze the Trace using Victor's Trace Analyzer
Compare .csv file outputs
Compare results across the differing heuristics and formula processing

Project Tasks

- Formulate Benchmark NP-complete problems (k-Coloring, Hamiltonian Cycles)
- Understand Victor Ying’s thesis with major attention to his description of measurements (required and wasted branches, events, implications) and how they related to performance
- Understand miniSAT’s solver code with an eye towards manipulation pickBranchLit() function for implementing random, VSIDS (default), DLIS heuristics
- Understand turning on and off of pre-processing
- Learn how to use Victor’s trace generator and trace analyzer
- Implement random decision heuristic (100%), DLIS, and test thoroughly

Results (and Discussions)

- Created model for Hamiltonian Cycles Problem
- Random Decision Making
    - Have tested with ~70% random decisions
        - Differs from VSIDS on larger instances (eg: k-Coloring problem with 20 variables and 67 clauses)
    - Altering MiniSAT code to implement fully random decision making
    - Victor’s largest benchmark file (three.cnf - 8 clauses, 3 variables) worked 100% random but produced no differing results from VSIDS
- Pre Processing On vs. Pre Processing Off
    - Debugging the build to run with pre processing off
- DLIS
    - Need to implement over the break
- Compare for each decision heuristic the following
    - required and wasted branches, events, implications
    - Both SAT and unSAT instances
    - Also measure the impact on runtime statistics using Solver.cc::printStats()

Final Steps

- Finish Random Decision & Pre Processing Off implementation
  - Current Status: Random is not 100% random, i.e, sometimes falls on VSIDS when randomizing process fails
  - Debugging code - currently working on method to remove out already predetermined set variables, since otherwise without that the last iteration goes into an infinite loop
  - Pre-processing off seg faulting on files except for easy.cnf, and not sure why
- Implement code for Dynamic Largest Individual Sum (DLIS) decision heuristic
  - Choose the variable and value that satisfies the maximum number of unsatisfied clauses
  - Must go through all of the clauses
- Interpret our results & compare to Victor's
  - Generate appropriate graphs for visualization