COS 516: Class Project Outline
Testing SAT Heuristics

Natalia Perina, Pranjit Kalita

October 27, 2016

1 Introduction

Describe in a few sentences WHAT problem you will be addressing. Feel free to use the text in the list of projects as a starting point, but make it specific to the scope of your project.

In our project, we will be studying the effects of changing decision heuristics and formula processing techniques on three different classic NP-complete problems. Specifically, we will study the effects of random or VSIDS decision making, 2 literal watch scheme lazy structure and formula pre-processing into the miniSAT solver individually for the three aforementioned classic problems. We intend to use the finding of our experiments to draw parallels between termination and solvability in terms of efficient memory usage (if applicable) and time taken to execute.

2 Proposed Work

Describe briefly HOW you will be solving the problem. Provide as many relevant details as you can about the following items (not all of these might apply to your project).

- Tools: MiniSAT solver, Trace Generator and Trace Analyzer developed by Princeton Senior Thesis
- Examine the heuristics on 3 NP-Complete problems: traveling salesman, K-coloring, and longest common subset problem.
- Modules added would be the implemented for each heuristic that we test along with miniSAT implementation for the 3 problems. Implement modules/code for random decision, VSIDS decision, pre-processing on/off, 2 literal watching scheme on/off, and other potential heuristics.

  For the trace generator, since the main objective is to understand the design of the decision graph, we would want to follow our understanding of its implementation with our own more memory-efficient implementation. Doing so will probably require using newer and more efficient data structures.
- Experiments that we conduct will be running each of the problems with the varying heuristics for the SAT solver and comparing across the boards. Experiments can include the random decision and VSIDS decision heuristics, pre-processing turned on and off, 2 Literal watching scheme on / off. We can test varying outputs for these experiments which would include running time, termination of program, variable assignments, and potentially memory usage. For the trace analyzer/generator, we can run them on problems of increasing benchmarks and look at the useful work done and memory usage. Larger benchmark problems would be problems with a larger amount of clauses to be satisfied to determine UNSAT.
- Deliverables will be the code for each module/heuristics. Deliverables would also include the additional code for memory usage improvement in the trace generator/analyzer. Modules that we would

implement for this would be different ways of improving the decision tree storage – so potentially only storing parts of the decision graph at a time or using a different data structure that would require less storage. Scripts for running the tests on each of the heuristics and for testing large values will also be added to the deliverables. Code for traveling salesman and for largest common subset implementation of miniSAT would also be a deliverable. For example, this could include working examples, working code, scripts for running tests/experiments, result output from tools, etc. Result output will be the output of the tests, and we could potentially visualize some of the outputs with graphs.

3 Proposed Schedule

- Nov. 7th: Research heuristics/different options and data structure memory improvements.
- Nov. 14th: miniSAT implementations of 3 test problems, finish understanding the trace generator implementation
- Nov. 27th: Have implemented code for different heuristics/options in miniSAT, trace analyser
- Dec 7th: created Testing scripts/start tests and produce results
- Dec 12th: Project presentations
- Jan 17th: Final Project Report