CS227: Reasoning Methods in AI
Assignment 1 Feedback for:

Harry Robertson Todd Sullivan Pavani Vantimitta

April 27, 2008

I Overview

Project 1 was designed to give everyone hands-on experience with the propositional satisfiability algorithms that were presented in the first part of the course. Every group implemented solvers for these problems and hopefully the insights gained from the experience enhanced your understanding of the algorithms.

Section II provides general feedback and observations for all assignments. Section III provides specific feedback for your group. Additional feedback has also been left on a hardcopy of your group's report.

II General Feedback and Observations

Section 1 describes the overall performance results. The important aspects of the most successful reports is detailed in Section 2, while some general presentation advice appears in Section 3. Finally, Sections 4, 5, and 6 present unique and interesting approaches, engineering, and presentation techniques taken by various groups (as well as a few which would have interesting to see).

1 Performance

The maximum number of problems which could be solved by any group's solver was 151. No group was able to solve more than one of the `spin-glass` problems. Other problem classes which stumped many solvers were the `qcp` and `qwh` problems. *WalkSAT*, as expected had the best overall performance for most groups, though some found *GSAT-RW* or even *HSAT* to perform the best.

2 Report Organization

The reports, though dependent on the implementation, are the most important part of the project because they provide each group with an opportunity to explain their understanding of algorithms and why they performed as they did on the diverse problem classes. The experimental process is complicated and effectively reporting results can be difficult - as many of you found out when writing your reports. In general, the best reports covered the following important areas.

- **Problem Introduction** - What is the problem this paper is solving?
- **Algorithms** - What algorithms were used to solve the problem?
- **Tuning** - What values were used for tuning parameters (`p`, `max-flips`)? Why were these values used? What experimentation was done to justify these values? Under what circumstances might these tuning values be valid, and when might they not work well?
- **Experimental Method** - It is important to communicate *how* experiments were performed. In particular, how many trials were run? What environment were the experiments run in? What aspects of the program were included in the execution times (e.g. was parsing the file a significant cost)?
- **Results** - Results are more than a few tables and graphs. Data should be presented concisely when possible. Try to chunk it into meaningful pieces which can be independently analyzed. Identify interesting and unusual results and hypothesize about what caused them. Consider designing additional experiments to better explore quirks, and carry them out to prove (or just as well disprove!) your hypotheses.
- **Discussion - What We Learned and Future Work** - Analyze each algorithm and compare it to the others. What were the algorithms' relative strengths? Also, what insights were gained from the experiments? What hypotheses can you make from results? Again, consider designing and conducting additional experimentation if it is needed justify your views.

3 Presentation

Though you are primarily graded on the depth of your analysis, a good presentation is essential to effectively communicate all the interesting insights you have had. Here are some suggestions to maker the reader's life a little easier:

- **Bold the best performer** to draw the reader's eye to best results. When bolding, consider bolding multiple values if they are sufficiently close.
- **Timeouts** should be signified with an asterisk when presenting run times, rather than specifying the maximum running time.

- **Appropriate graphs** should be used to highlight interesting patterns or results from the data. They are often more effective at illustrating a point than a data table alone.
- Use **page numbers**.
- **Justified, two-column** text is often easier to read for scientific reports.

4 Good Approaches

4.1 Benchmark Analysis

Not all SAT problems are created equal. In other words, there are a variety of properties which may make a problem more difficult. The number of variables or clauses in a problem is one heuristic for determining difficulty, but you likely found that some problem classes (like `spin-glass`) were much harder than other problems which had more variables. One group went to great lengths to better understand the specific properties of each problem class and explained those properties and how they affected each solver. This knowledge is valuable and may be exploited, as another group suggested, with *problem-specific* heuristics.

4.2 Consistency

Examining the variance of an algorithm on various problem classes and comparing it to other algorithms would be quite informative. Furthermore, when evaluating stochastic algorithms, consider recording the random seed used to initialize the random number generator. This way, experiments are reproducible.

4.3 Per-Problem Comparison

It is difficult to concisely and effectively compare results for two algorithms on a problem-by-problem basis - many groups simply reported aggregate results for different classes of problems. One way more fine-grained results might be reported is through the use of a scatter plot (or histogram) which plots the difference between two algorithms' run times on each problem.

4.4 Plateaus

A common hill-climbing problem is to become "stuck" on a plateau. A number of groups developed heuristics to identify when they were stuck on a plateau so that they could try to escape it.

5 Good Engineering

5.1 Amdahl's Law and Profiling

Amdahl's Law states that the overall speedup which may be achieved by improving some subsystem is limited by the fraction of the work done by subsystem. Several groups took this to heart and used profilers to identify bottlenecks. Groups which used profilers discovered opportunities for optimization such as avoiding *Java*'s `Collections` framework due to the overhead of autoboxing integers. Another group found that returning a list of possibilities was rather expensive and that it was much more efficient to integrate the final variable choosing function with the one which identified the initial candidate variables.

5.2 Architecture-Related Analysis

Though each group analyzed performance, there was little discussion of how various components of the machine were being stressed by each algorithm. It would have been interesting to see an analysis of the algorithms in terms of processor and memory footprints, and how these requirements limited performance. It would have been even more interesting to have seen how such an analysis might have been used to further optimize the implementations. A comparison of such "optimizations" with the original implementations would be an essential part of this analysis.

5.3 Meta-Statistics

One group devised new "meta-statistics" to investigate various aspects of solvers' behavior. One such statistic tracked how many flips occurred between the best variable assignment and a restart. This data was used to determine good tuning parameter heuristics. Other statistics were used to evaluate a solver's fairness or determine the effectiveness of various tie-breaking strategies.

5.4 Simplification

One group undertook the difficult task of pre-processing inputs and trying to simplify them. This strategy is an effective tactic which is often used when trying to solve SAT problems in practice.

5.5 Tweaking Algorithms

Several groups presented modified versions of the basic *GSAT* and *HSAT* algorithms. Tweaking algorithms is a valuable technique. They are most effective when used to gather data about a possible improvement to a specific problem which has been observed. However, such tweaks are less valuable if they are not targeted at a specific problem.

6 Good Presentation

6.1 Explanatory Diagrams

When explaining a complicated process, it may be helpful to use a diagram to concisely capture the complexity. One group created diagrams to help illustrate their data structures and it definitely helped us better understand their approach.

6.2 Speedups versus Run Times

When presenting run times, it is a good idea to use the best implementation of the naive algorithm as a baseline, and then compare others' performance relative to that baseline. This is also helpful when comparing two algorithms - it allows the reader to focus on a single column of speedups instead of various run times. Speedups are also easier to compare and reason about that run times.

III Feedback for the Robertson-Sullivan-Vantimitta Group

1 The Good

- **p Tuning**. Your experimentation with the `p` tuning parameter was very thorough. However, it would probably be more effective to determine a single value or function for computing `p` by comparing various values head-to-head at the beginning of the report. Presenting all the different values of `p` throughout the report tends to clutter later graphs and obfuscate some of the good work you have done. Also, by eliminating poor values at the beginning, it may enable you to run fewer or other more important experiments.

- **Tweaked Algorithms**. You present a number of tweaks to existing algorithms. While these tweaks seem reasonable, it would even better if you had a motivating reason for trying these tweaks and could hypothesize about what kinds of improvements these tweaks might reap. That said, the *WalkSAT-M* solver seemed to perform quite well in a number of circumstances (in particular, on the `ransat` problem class).

- **Code-Level Optimizations**. The attention to detail you gave the underlying implementation was excellent. The simple test driver you wrote to illustrate the gains was a great idea, though it would have been even better if you had presented these results in the report. Better still would have been to quantify how much these optimizations improved the algorithms on various problem classes. It would have been very interesting to see if the optimizations enabled some of the more difficult problems to be solved.

- **Good Engineering**. Your group had the best performing solver out of all the groups. You were very careful about exploring many implementation details and investigated a number of algorithmic improvements as well. Good work! It would be even better if you had more discussion and insights into all this hard work.

2 Areas for Improvement

- **Results**. Unfortunately, this important section did not provide the detailed analysis we were looking for. This section would have been far stronger if you had presented the run time data and more carefully compared the results of the various solvers. In particular, why does *WalkSAT* win? Also, which approach solves the most problems and why? We were able to derive some of these answers from your tables, but tables are no substitute for a good, in-depth analysis of the data.
- **Missing Data**. Though you did not present data for `ransat`'s hardest instances, we did not penalize you for it since you had such great success with some of the harder instances. In the future, be more cautious with your *leland* accounts and more discriminatory when choosing which problems are most interesting to run!

3 Grade

Your letter grade on this assignment: *B+*