**James Davidson - V00812527 | Bridget Rassell - V00804180 | Wilfred Lynch - V00809085**

**CSC 320 – SAT based Sudoku solving Fall 2015**

**Our solver can be found at:** [**https://github.com/jamesthomasdavidson/SudokuSolver**](https://github.com/jamesthomasdavidson/SudokuSolver)

**☺**

**Introduction:**

We implemented our SAT based Sudoku solver using an encoding and decoding program sudokusolver.py and Minisat, a well-respected SAT solver that takes as input a file specifying a CNF expression in DIMACS format and outputs whether or not the expression is satisfiable, along with the solution if it exists. Our program implements the encoding based on the article “Sudoku as a SAT problem” from connex, then decodes the Minisat output into readable format for the solved Sudoku, which then prints to the console.

**Usage:**

To run, simply

**Ouput:**

Our SAT solver prints the solution to the console, as well as the execution time.

**Testing:**

To test our Sudoku solver, we built test files that differ in difficulty then timed how long the the program took to print the solution. We were interested in finding out if problems deemed to be “difficult” or “evil” to humans would reflect on the performance of the SAT-based Sudoku solver. The different levels of difficulty are: easy, medium, hard and evil. Our results are as follows:

Recorded times:

Easy: 0.014, 0.016

Medium: 0.016, 0.017

Hard: 0.018, 0.015

Evil: 0.018, 0.016

**Discussion:**

The human difficulty categorization of the Sudoku puzzles did not appear to reflect on the performance of the Sudoku solver. We reasoned that this might be the case because SAT solvers would produce similar number of clauses for each category, and then the time to solve them would be very close. Humans usually use a trial and error method when solving Sudoku’s, which differs from the way say solvers would solve the puzzle.

Notes:

* We realize more (many many more..) input files are necessary to draw more concrete conclusions
* We would need to also define the human categories more concretely