Problem Statement

How do various types of Sudoku puzzles (based on difficulty and symmetry) solved using a satisfiability (SAT) solver affect several variables. The first is the time it takes to solve the puzzle. The second is the number of times three internal functions are called in the solving.

Hypothesis

I had predicted that both solve time and calls to all three functions would have a direct relationship with difficulty and no relationship with symmetry.

Procedure

First, I coded the SAT solver and the wrapper to use it to solve Sudoku puzzles in the Python programming language; this took most of the time. I then used an online Sudoku generator to get puzzles of various types. These were solved using the SAT solver and timed. I then collected the data and analyzed it.

Conclusion

From the data gathered in this investigation, I have concluded that both the time to solve a Sudoku puzzle using a satisfiability (SAT) solver and the calls to *satr* required vary directly with perceived difficulty, and symmetric puzzles are easier, computationally and time-wise, to solve. The data for calls to *simplify* and *deduce* is inconclusive.

Acknowledgements

I thank Dr. Aaron Bradley for his supervision on coding the SAT solver and advice on this poster. I also thank my parents for feedback on various drafts of the poster.

A Sudoku puzzle can be simplified into a Boolean expression. This expression will evaluate to true if the puzzle is valid, and false if the puzzle cannot be solved. Some of the variables will be set to values corresponding to the given numbers, and then the SAT solver will solve it. In addition to returning the satisfiability (hopefully TRUE), it will also return the variables it assigned, and from this the solution to the puzzle can be deduced.

Each variable corresponds to whether a single cell is equal to a number from 1 to 9. For instance, the S_{111} variable will correspond to whether row 1, column 1 of the puzzle is equal to 1; the S_{845} variable will correspond to whether row 8, column 4 of the puzzle is equal to 5. This is part of the equation:

Symmetry	Difficulty
No symmetry 120 cases	Difficulty 1 30 cases
	Difficulty 2-9 30 cases
	Difficulty 10-99 30 cases
	Difficulty 100-999 30 cases
Diagonal symmetry 120 cases	Difficulty 1 30 cases
	Difficulty 2-9 30 cases
	Difficulty 10-99 30 cases
	Difficulty 100-999 30 cases
Orthogonal (horizontal and vertical) symmetry 120 cases	Difficulty 1 30 cases
	Difficulty 2-9 30 cases
	Difficulty 10-99 30 cases
	Difficulty 100-999 30 cases
Both symmetries 120 cases	Difficulty 1 30 cases
	Difficulty 2-9 30 cases
	Difficulty 10-99 30 cases
	Difficulty 100-999 30 cases

References

Bernal, Raoul A. "De Morgan's Laws Revisited: To Be AND/OR NOT To Be." SAS PharmaSUG Papers. Amgen, Inc., 2005. Web. 31 Dec. 2011. <http://www.lexjansen.com/pharmasug/2005/posters/po25.pdf>.

Gurevich, Yuri. "The Logic in Computer Science Column." Computing Science. Simon Fraser University. Web. 1 Jan. 2012. <http://www.cs.sfu.ca/~mitchell/papers/colLogCS85.pdf>.

Lewis, Rhyd. "Metaheuristics can Solve Sudoku Puzzles." *Journal of Heuristics* Volume 13 (2007): 387-401. Print.

Lynce, I. and Ouaknine. "Sudoku as a SAT problem." *9th Symposium on Artificial Intelligence and Mathematics* (2006). Print.

Norvig, Peter. "Solving Every Sudoku Puzzle." *Peter Norvig*. Web. 18 Sept. 2011.

Scherphius, Jaap. "Sudoku Generator, Version 3." Jaap's Scratch Pad. 2006. Web. 31 Dec. 2011. <http://www.jaapsch.net/sudoku.htm>.

Simonis, H. "Sudoku as a Constraint Problem." *CP Workshop on Modelling and Reformulating Constraint Satisfaction Problems* (2005): 13-27. Print.

Yato, T. and T. Seta. "Complexity and Completeness of Finding Another Solution and Its Application to Puzzles." *IEICE Trans. Fundamentals* Volume E86-A, No. 5 (2003): 1052-1060.