CSC 320 – Foundations of Computer Science Project Summary Report

Luuk Veenis
Nikita Malhotra

1.0 Introduction

For the final project we implemented a program in Ruby that converts partially solved Sudoku puzzles into the standard DIMACS SAT-challenge format that can be inputted to the MiniSAT SAT solver. If a solution to the puzzle exists, it is converted back and presented to the user. Several extended tasks were also explored, and will be discussed in this report, including alternate puzzle encodings and the use of SAT solvers other than MiniSAT.

2.0 Discussion

2.1 Different Encodings

For the first extended task, we considered the extended encoding outlined in the project specification of Sudoku as a SAT Problem. This encoding added redundant clauses to the minimal encoding and is claimed to be the intuitive encoding one might come up with at first. Since this encoding adds redundant clauses, we can expect the size of the encoding to be larger than that of the minimal encoding. Specifically, the minimal encoding resulted in 8,829 clauses plus the additional unit clauses representing the given values in the puzzle. The extended encoding resulted in 11,988 clauses plus the additional unit clauses. This difference amounts to a 35.8% increase in the number of clauses. When comparing the performances of the 2 encodings, the same puzzle was solved a fixed number of times using the MiniSAT SAT solver to account for any small variances. A shell script was executed to determine the total time required for the solver to run for specific iterations and these timings were reported after the execution was complete. Figure 1 and Figure 2 illustrate the performance between the minimal and extended encoding.

	1	100	1,000	10,000
Minimal	0.012s	0.498s	4.763s	47.910s
Extended	0.013s	0.460s	4.537s	46.557s

Figure 1: MiniSAT performance for minimal and extended encodings.

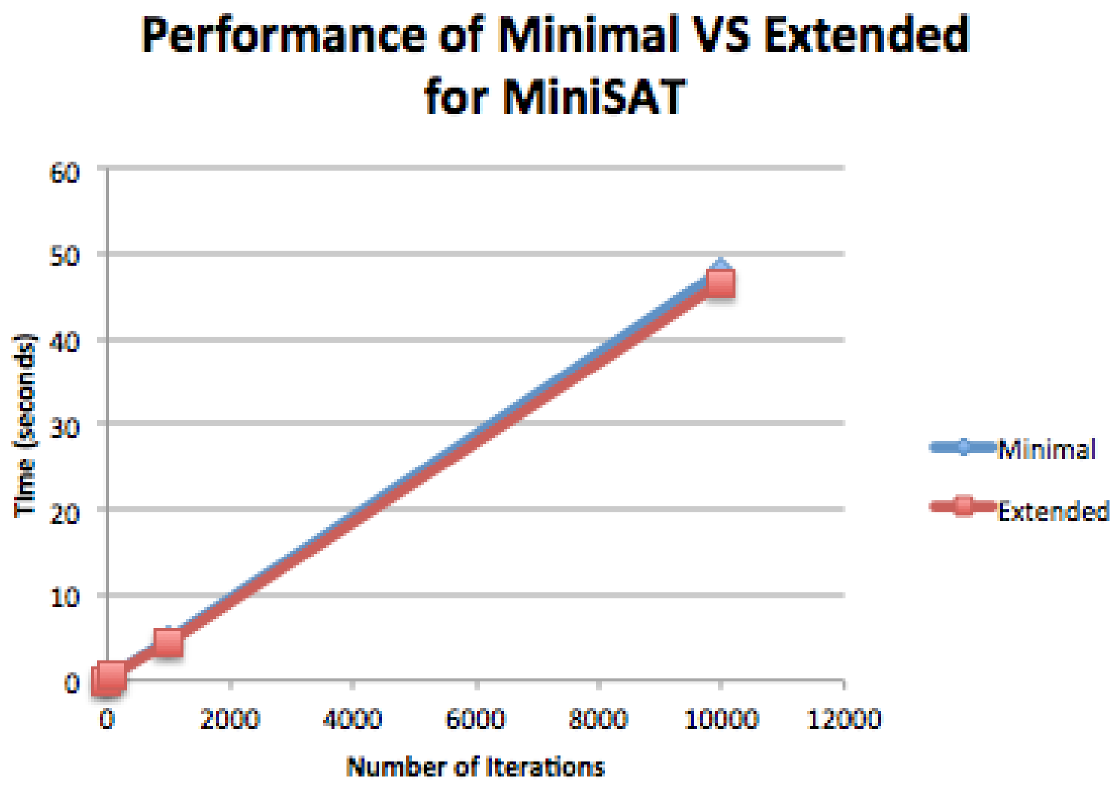

Figure 2: Performance of Minimal VS Extended for MiniSAT

As shown by the table and graph above, we saw a minor performance improvement using the extended encoding. However, the performance only increased by approximately 2.8% for a 35.8% increase in the number of clauses. We argue that the performance difference is insignificant compared to the increase in input size and do not recommend the extended encoding for MiniSAT.

2.2 RSat

For the second extended task, we compared the performance of two different SAT solvers. Specifically, we compared our results above with the RSat SAT solver. Figure 3 and Figure 4 below summarize the same tests when run using RSat.

	1	100	1,000	10,000
Minimal	0.022s	0.752s	7.587s	78.59s
Extended	0.019s	0.916s	9.132s	90.43s

Figure 3: RSAT performance for minimal and extended encodings.

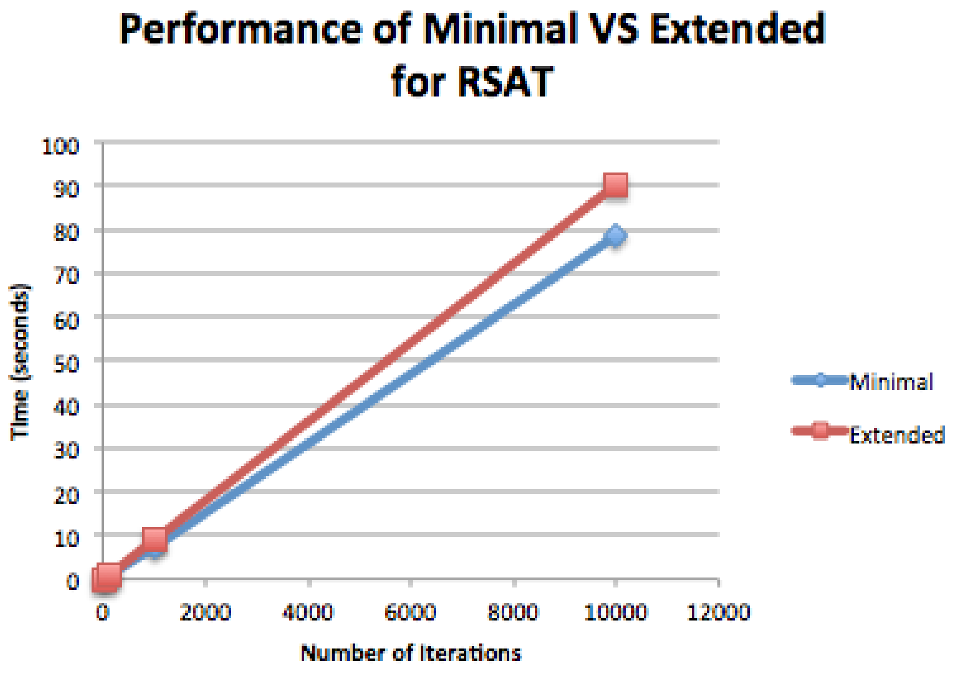

Figure 4: Performance of Minimal VS Extended for RSAT

Again, we see that for a single puzzle, the difference in solution time is insignificant. However, the performance decreased by approximately 15% over 10,000 iterations using the extended encoding over the minimal encoding. From this observation, we can conclude that the extended encoding is not useful with RSat since the input size increases and the solution time decreases.

Note that for both types of encodings shown in this paper, MiniSAT performs significantly better than RSat. This is to be expected, as RSat won gold in the SAT competition in 2007, whereas MiniSAT is a new solver that won gold in 2014. We can expect that many improvements and optimizations to the algorithms used occurred during that time.

2.3 Special Purpose Solver

For the third extended task, we explored the performance of Peter Norvig's special purpose Sudoku solver. This solver is written in Python and is not a SAT-based solver. It uses the encoding we originally provide to our program of the form "1..3..5.6...9..3.1.. etc" and solves the puzzle directly. The table below shows the performance of Peter Norvig's solver for the same puzzle tested in previous sections.

	1	100	1,000	10,000
Standard Encoding	0.030s	0.716s	6.891s	69.38s

Figure 5: Norvig's Python Solver performance for standard encoding.

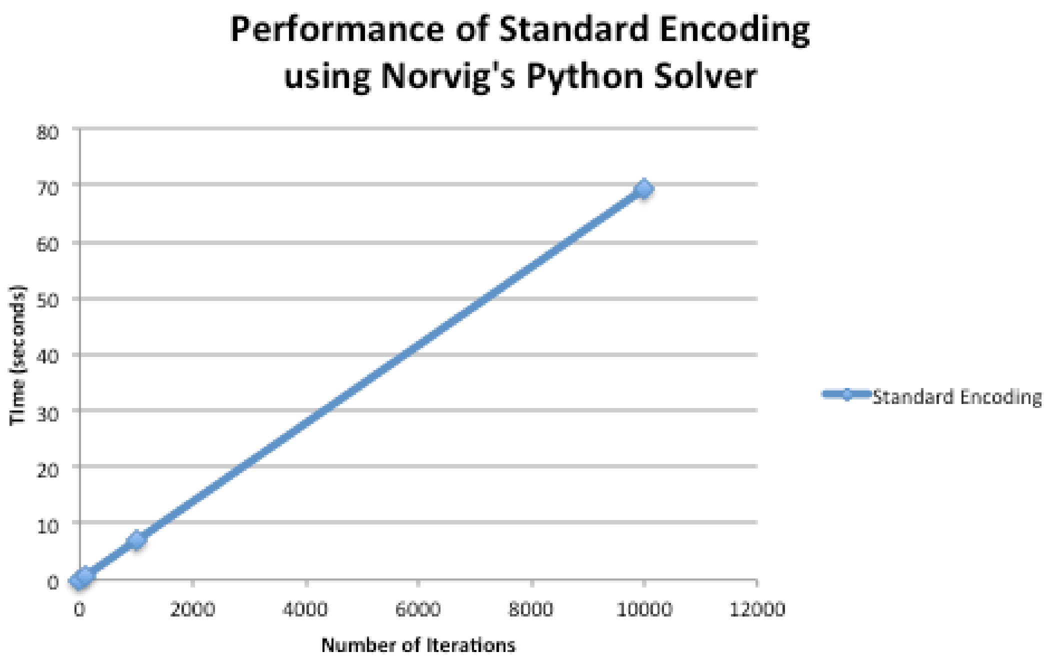

Figure 6: Performance of Standard Encoding using Norvig's Python Solver

The results above indicate that this special purpose solver performs better than RSat, but still significantly slower than MiniSAT. This leads us to recommend the use of a SAT-based solver for anyone looking to implement a Sudoku solver. Since most SAT solvers such as MiniSAT or RSat use a standardized input format, it is trivial so swap out the solver used by the program if a newer, better performing one is released.

3.0 Conclusion

The primary goal of this project was to implement a program that converted a standard representation of Sudoku puzzles into an input format for SAT solvers and then convert the solution returned by solvers back into a human-readable format. Our solution is able to solve any puzzle we give it in fractions of a second and can also report if the SAT representation is not satisfiable.

Additionally, we compared the performance of different encodings and different solvers. We discovered that the extended encoding outlined in the provided document Sudoku as a SAT Problem did not provide significant performance advantages over the minimal encoding and in the case of RSat even decreased performance. It also increased the number of clauses in the SAT encoding drastically and is therefore recommended to use the minimal encoding in all cases. Although Peter Norvig's special purpose Sudoku solver performed better than an older SAT solver RSat, it failed to perform better than the newer SAT solver, MiniSAT. Since most SAT-based solvers are interchangeable due to a standard input format, it is more practical to use these for Sudoku solvers so they can be swapped out for better performing ones in the future.

4.0 Acknowledgements

This paper is part of the Spring 2015 Foundations of Computer Science course and it was presented as a project for the course. We would like to thank Bruce Kapron for his guidance and help during the course.