Project 1

Ruben Polak (11389095) Krsto Proroković (11391839)

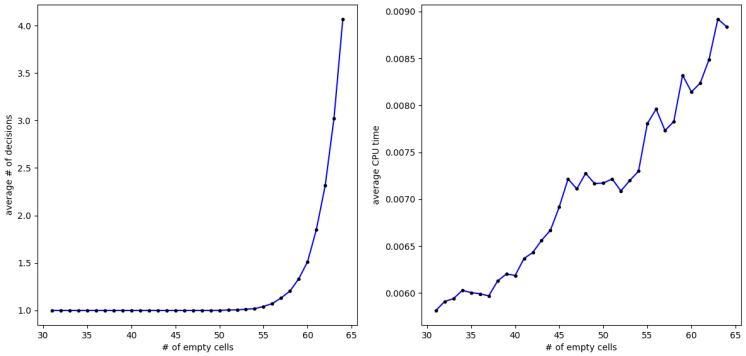
Hypothesis and Approach

In this project we investigate how the performance of SAT solvers scales with the number of empty cells in a **proper** Sudoku. Our hypothesis is that the performance decreases exponentially with the number of empty cells. This should justify the belief that proper Sudokus with more hints are easier to solve.

There are examples of proper sudoku with 17 hints, but no proper sudoku with 16 hints exists [1]. Gordon Royle has compiled a huge database of 17-hint Sudokus [2]. We took a random sample of 2400 of these 17-hint sudokus. We solved each of them using SAT solver and randomly replaced a single empty cell on every unsolved 17-hint Sudoku with the filled cell from the corresponding solution to obtain 2400 18-hint Sudokus. Note that doing this preserves properness. Then we solved 18-hint Sudokus and randomly replaced a single empty cell on each of them with a filled cell from the corresponding solution to obtain 2400 19-hint Sudokus. We continued the procedure until we solved 50-hint Sudokus. After each iteration we recorded the performance of the SAT solver. We used extended encoding for Sudoku as in [3] and Minisat as SAT solver. To quantify its performance we used number of restarts, number of decisions and CPU time. We collaborated via Github. Corresponding code and data can be found at `https://github.com/rubenpolak/KR_2017` and the presentation can be seen at ...

Results

Apparently, Minisat makes no restarts when solving these Sudokus, regardless of the number of hints. Average number of decisions stays at one until the number of empty cells passes 50, then it increases quickly. Still, it does not get much above 4. Average CPU time increases, but it seems that it does not increase exponentially, rather linearly. The figure below displays the results of our work.



Conclusion and Further Work

The results counter our hypothesis, so we can refute it. Interesting question is whether the same holds for every $n \times n$ Sudoku. However, answering this would not be easy at all. First, there is no database containing proper Sudokus for $n > 9$, so the method used over here would not work (even though our code supports $n \times n$ Sudoku). Second, better encodings may be needed as the number of clauses in minimal or extended encoding grows as $\mathcal{O}(n^4)$. For example, see [4].

One thing we noticed is that the number of decisions made by the SAT solver could be used as a measure of difficulty of Sudoku. It would be nice to see how this compares with human performance or rating.

References

[1] Gary McGuire, Bastian Tugemann and Gilles Civario. There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem. arXiv:1201.0749

[2] http://staffhome.ecm.uwa.edu.au/ 00013890/sudokumin.php

[3] Ines Lynce and Joel Ouaknine. Sudoku as a SAT Problem.

[4] Gihwon Kwon and Himanshu Jain. Optimized CNF Encoding for Sudoku Puzzles.