Encoding a Sudoku as a k-SAT problem

Firstname Lastname

***Abstract*—This paper aims to examine the influence of the clauses length inside a Sudoku CNF encoding on the performance of a SAT solver.**

I. Introduction

II. Hypothesis

Our hypothesis is: *"Encoding the Sudoku as a k-SAT problem, for $k_3, \ldots,$ will get better performances (in terms of CPU time, number of decisions and conflict) during the resolution with a SAT solver, compared to the naive encoding"*. We were motived to investigate in this direction by the results of a paper dealing with general k-SAT problems and the search for hard problems.

III. Experimental setup

A. The naive encoding: The naive encoding was implementend by defining the functions *at least one* and *at most one*, which encode the concept of having one and only one value among the considered set. Then we applied this function to each single cell, to ensure only one value was assigned, to lines, to rows and finally to blocks. This approach returns clauses with length 2, that avoid the presence of more than one value in the cell, and clauses of length n, namely the dimension of the Sudoku.

B. The k-SAT encoding: In order to riformulate the problem as a k-SAT problem, we introduced new dummy variables in the following way:

$$(a \vee b \vee c \vee d) = (a \vee b \vee z) \wedge (c \vee d \vee \neg z).$$

This increases the number of clauses and variables, but the length is reduced. We tried to state the problem as a k-SAT for $k \in \{3, 4\}$.

C. The SAT solver: We choose to use the SAT solver miniSAT. It accepts files written in CNF format and returns the solution and a number of useful statistics. We were interested in the CPU time, the number of conflicts and the number of decisions.

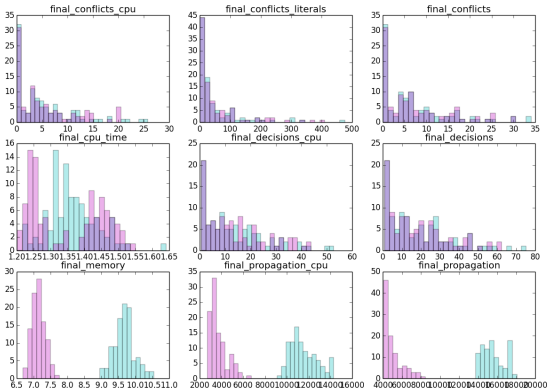
D. The database: We used a database collected from a website. We were interested in analyzing the problem for Sudoku of different sizes, hence we scraped sudoku of size 9, 16 and 25.

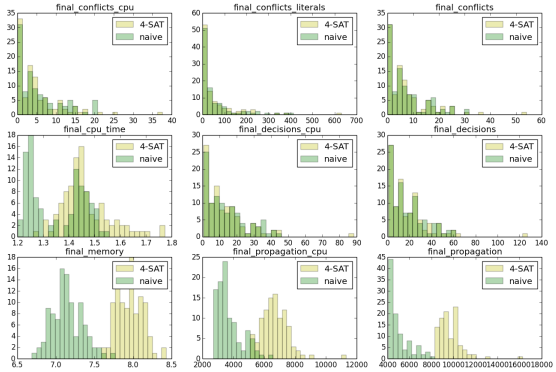
D. The code: The code we wrote scrapes the sudoku from the website, then it encodes them in CNF format and reduces the encoding to a k-SAT problem. Finally miniSAT is used to solve the sudoku and gather statistics.

IV. Experimental results

For each sudoku size, we applied the procedure for k = 3,4 on 500 sudoku. Then we represented the statistics comparison in some histogram plots. The plot show that the performance is not increased with the new encoding. For what concerns the conflicts, there is almost no difference, but for the CPU time and the memory usage, it is clear that the SAT solver performance gets worse. Some further research confirmed that the length of the clauses is much less important than the number of clauses and variables.

Here we can see the comparison between 16x16 sudokus encoded in the naive way and as a 3-SAT and a 4-SAT.





V. Interpretation

VI. Conclusion, summary, future work

References

[1] k-SAT paper
[2] miniSAT
[3] Sudoku database