# Comparative Difficulty of X-Sudoku and Sudoku Stripe

Caitlin Lagrand and Hunter McKnight

October 2, 2017

**Abstract**

We hypothesize that, given a set of underdetermined puzzles and a SAT solver, the puzzles will be more difficult in both the average and worst case to solve when encoded as sudoku stripe than as x-sudoku. We test this hypothesis using one large dataset and two different SAT solvers. Our results offer some support for our hypothetical claim about average-case difficulty but refute our claim about worst-case difficulty.[1]

## 1 Introduction

X-sudoku and sudoku stripe are two variations on sudoku with additional constraints. X-sudoku requires that each digit occur exactly once along each of the puzzle's two main diagonals. Sudoku stripe requires that the digits in at least one row, column, or region must appear in strictly ascending or strictly descending order. Although a human player may reason that the stripe constraint is in some sense 'easier' because there are many ways to satisfy it, we reasoned that the relative strictness of the x constraint should actually make x-sudoku relatively easier for a SAT solver. As a solver attempts to build a model decision by decision, it cannot tell that a partial model will ultimately fail until that partial model actually violates an encoded constraint. Because the x constraint is so simple and so strict, it should be violated relatively early in model construction if it is violated at all. On the other hand, there are so many ways to satisfy the stripe constraint that a solver might get quite deep into model construction before all those possibilities are ruled out. For similar reasons, the solver may need to backtrack more (in addition to searching deeper) before either finding a model or determining the the stripe constraint cannot be satisfied at all.

Depth of search and number of backtracks—quantified by maximum decision level reached, total number of decisions made, and number of conflict clauses added—is the obvious metric for our notion of puzzle difficulty. Of course, most sudoku puzzles are designed to be solved with no guessing or backtracking, whether by humans or SAT solvers; such puzzles are equally difficult for as x-sudoku or sudoku stripe by our metric, since the solver will make no decisions at all in either case. We thus restrict our attention to puzzles that require the solver to make at least one decision, which we call "underdetermined" puzzles. Furthermore, to account for differences between solvers (such as random seeds or heuristics for decision-making), we focused on worst-case and average-case performance for a set of puzzles rather than directly comparing performance directly between individual puzzles. We therefore hypothesized that, for a given set of underdetermined $9 \times 9$ sudoku puzzles, the average and maximum values for maximum decision level reached, number of decisions made, and number of conflict clauses added should be lower when a SAT solver encodes x-sudoku rules than when it encodes sudoku stripe rules.

## 2 Experimental Setup

### 2.1 Dataset and Solver Selection

For our dataset, we chose Dr. Gordon Royle's publicly available collection of 49151 sudoku puzzles[2]. Each puzzle in this dataset has exactly 17 hints, and the puzzles are pairwise mathematically inequivalent under transposition, permutations of the digits, permutations of columns in a region, permutations of rows in a region, and both column- and row-wise permutations of the regions. Since these puzzles contain so few hints, we expected many of these puzzles to be underdetermined

[1] Watch a brief video presentation on YouTube: https://youtu.be/msFuuGF-dZs.

[2] This dataset is publicly available at http://staffhome.ecm.uwa.edu.au/~00013890/sudokumin.php.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *4* | 7 | 8 | 1 | 9 | 5 | 3 | 2 | *6* |
| 5 | *2* | 6 | 4 | 8 | 3 | 7 | *1* | 9 |
| 9 | 3 | *1* | 6 | 7 | 2 | *4* | 5 | 8 |
| 2 | 5 | 9 | *8* | 6 | *7* | 1 | 3 | 4 |
| 3 | 6 | 4 | 2 | *5* | 1 | 9 | 8 | 7 |
| 1 | 8 | 7 | *3* | 4 | *9* | 5 | 6 | 2 |
| 7 | 1 | *2* | 9 | 3 | 8 | *6* | 4 | 5 |
| 6 | *9* | 3 | 5 | 2 | 4 | 8 | *7* | 1 |
| *8* | 4 | 5 | 7 | 1 | 6 | 2 | 9 | *3* |

Solved X-Sudoku

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | 2 | 7 | 8 | 9 | 1 | 3 | 4 | 5 |
| 5 | 4 | 3 | 6 | 7 | 2 | 1 | 9 | 8 |
| 9 | 1 | 8 | 3 | 4 | 5 | 7 | 2 | 6 |
| 4 | 5 | 6 | *1* | *2* | *3* | 8 | 7 | 9 |
| 7 | 8 | 9 | *4* | *5* | *6* | 2 | 3 | 1 |
| 2 | 3 | 1 | *7* | *8* | *9* | 6 | 5 | 4 |
| 3 | 6 | 5 | 9 | 1 | 7 | 4 | 8 | 2 |
| 8 | 7 | 2 | 5 | 6 | 4 | 9 | 1 | 3 |
| 1 | 9 | 4 | 2 | 3 | 8 | 5 | 6 | 7 |

Solved Sudoku Stripe

Figure 1: Example Solutions

for both x-sudoku and sudoku stripe and therefore relevant to the support or refutation of our hypothesis.

For our principal solver, we chose Princeton University's zChaff (version 2007.3.12, 64 bit)[3]. Since our hypothesis makes no particular assumptions about the solver itself (except perhaps that it employs conflict-driven clause learning), this decision was largely arbitrary. For comparison, we also used Armin Biere's CaDiCaL (version sc17)[4]. As we found CaDiCaL's default metric for decision depth not properly comparable with zChaff's, we do not report maximum decision level for the CaDiCaL experiment.

The python scripts we wrote to parse data, encode and decode puzzles, and interact with solvers are publicly available on our GitHub repository[5].

## 2.2 Encoding

A SAT solver determines if a set of clauses is satisfiable. These clauses are Boolean expressions, using variables (literals) the can be either true or false, and the operators AND ($\wedge$), OR ($\vee$) and NOT ($\neg$). The set of clauses must be in Clausal Normal Form (CNF), which is a conjunction of one or more clauses, where a clause is a disjunction of literals. To be able to solve sudoku puzzles using a SAT solver, a sudoku must be encoded into Boolean expressions in CNF, which means that each possible cell and its value (1-9) must be represented with its own literal that can be either true or false.

### 2.2.1 Normal sudoku

Sudoku puzzles can be encoded in several ways [1]. This research used the *minimal encoding* [2] which describes a sudoku as a conjunction of the rules for cells, rows, columns, regions and the hints (1). A normal sudoku is encoded using 729 ($9 \times 9 \times 9$) literals (Fig. 2). Starting from the left top, each cell is assigned nine literals. When these literals are represented as tuples (row, column, value), the rules for cells can be encoded as (2). This encoding means that each cell must contain a digit from one to $n$, where $n = 9$ for a $9 \times 9$ sudoku. The rules for rows are such that each row must contain the digits one to $n$ once (3). The rules for columns and regions are defined in a similar way (4, 5) and all hints are encoded as unit clauses (6).

[3] ZChaff is publicly available at https://www.princeton.edu/~chaff/zchaff.html.
[4] CaDiCaL's repository is maintained at https://github.com/arminbiere/cadical.
[5] https://github.com/huntermcknight/x-stripe

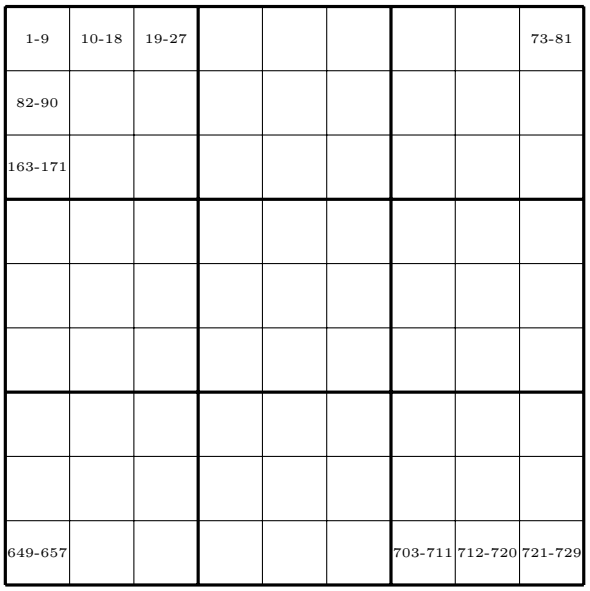


Figure 2: Some of the literals used to encode the possible digits in each cell.

$$Sudoku = Cell \bigwedge Row \bigwedge Column \bigwedge Region \bigwedge Hints \tag{1}$$

$$Cell = \bigwedge_{r=1}^{n} \bigwedge_{c=1}^{n} \bigvee_{v=1}^{n} (r, c, v) \tag{2}$$

$$Row = \bigwedge_{r=1}^{n} \bigwedge_{v=1}^{n} \bigvee_{c_i=1}^{n-1} \bigvee_{c_j=c_i+1}^{n} \neg(r, c_i, v) \vee \neg(r, c_j, v) \tag{3}$$

$$Column = \bigwedge_{c=1}^{n} \bigwedge_{v=1}^{n} \bigvee_{r_i=1}^{n-1} \bigvee_{r_j=r_i+1}^{n} \neg(r_i, c, v) \vee \neg(r_j, c, v) \tag{4}$$

$$Region = \bigwedge_{r_b=1}^{\sqrt{n}} \bigwedge_{c_b=1}^{\sqrt{n}} \bigwedge_{v=1}^{n} \bigwedge_{r_i=1}^{\sqrt{n}} \bigwedge_{c_i=1}^{\sqrt{n}} \bigwedge_{r_j=r_i+1}^{\sqrt{n}} \bigwedge_{c_j=1}^{\sqrt{n}} \neg(r_b * \sqrt{n} + r_i, c_b * \sqrt{n} + c_i, v) \vee \neg(r_b * \sqrt{n} + r_j, c_b * \sqrt{n} + c_j, v) \tag{5}$$

$$Hints = \bigwedge^{\#Hints} (r, c, v) \tag{6}$$

### 2.2.2 X-Sudoku

The x-sudoku requires that the two main diagonals must contain each digit exactly once. These rules can be encoded as (7), which is similar to the encoding of the rows, columns and blocks, but with different indices.

$$\begin{aligned} Diagonal\_Left\_Right &= \bigwedge_{i_i=1}^{n} \bigwedge_{v=1}^{n} \bigwedge_{i_j=i_i+1}^{n} \neg(i_i, i_i, v) \vee \neg(i_j, i_j, v) \\ Diagonal\_Right\_Left &= \bigwedge_{i_i=1}^{n} \bigwedge_{v=1}^{n} \bigwedge_{i_j=i_i+1}^{n} \neg(i_i, n - i_i, v) \vee \neg(i_j, n - i_j, v) \end{aligned} \tag{7}$$

### 2.2.3 Sudoku Stripe

The sudoku stripe requires that the digits in at least one row, column, or region must appear in strictly ascending or strictly descending order. To ensure this, stripe can be defined as (8). For a row that is striped, the row can be ascending or descending (9a). These ascending (10a)

and descending (11a) rules can be described using new literals (10b, 11b). Each row can now be described with a new literal using the literals for ascending and descending (12). Combining these new literals for each row results in a disjunction of all rows (9b). The encoding for a striped column or region is similar to the encoding of a striped row, only the indices differ, since it will be looping over the columns or regions instead of the rows.

$$Stripe = RowsStripe \vee ColumnsStripe \vee RegionsStripe \tag{8}$$

$$RowsStripe = \bigvee_{r=1}^{n} (\bigwedge_{c=1}^{n} (r, c, c)) \vee (\bigwedge_{c=1}^{n} (r, c, n-c)) \tag{9a}$$

$$RowsStripe = \bigvee_{r=1}^{n} row_r \tag{9b}$$

$$Ascending = \bigwedge_{c=1}^{n} (r, c, c) \leftrightarrow (r, 1, 1) \wedge y_1 \tag{10a}$$

$$y_1 \rightarrow \bigwedge_{c=2}^{n-1} (r, c, c) \leftrightarrow \neg y_1 \vee (\bigwedge_{c=2}^{n-1} (r, c, c)) \leftrightarrow \bigwedge_{c=2}^{n-1} (\neg y_1 \vee (r, c, c)) \tag{10b}$$

$$Descending = \bigwedge_{c=1}^{n} (r, c, n-c) \leftrightarrow (r, 1, n-1) \wedge y_2 \tag{11a}$$

$$y_2 \rightarrow \bigwedge_{c=2}^{n-1} (r, c, n-c) \leftrightarrow \neg y_2 \vee (\bigwedge_{c=2}^{n-1} (r, c, n-c)) \leftrightarrow \bigwedge_{c=2}^{n-1} (\neg y_2 \vee (r, c, n-c)) \tag{11b}$$

$$row_r \rightarrow ((r, 1, 1) \wedge y_1) \vee ((r, 1, n) \wedge y_2) \leftrightarrow (\neg row_r \vee (r, 1, 1) \vee y_2) \wedge (\neg row_r \vee (r, 1, n) \wedge y_1) \tag{12}$$

## 3 Experimental Results

Of the 49151 puzzles, 0 are valid instances of x-sudoku, 7 are valid instances of sudoku stripe, 32772 are underdetermined for both puzzle types using zChaff, and 32781 are underdetermined for both puzzle types using CaDiCaL. Figure 3 and Figure 4 below are aggregated visualizations of the difficulty of the underdetermined puzzles for zChaff and CaDiCaL, respectively. Table 1 and Table 2 show the average and maximum statistics for zChaff and CaDiCaL, respectively.

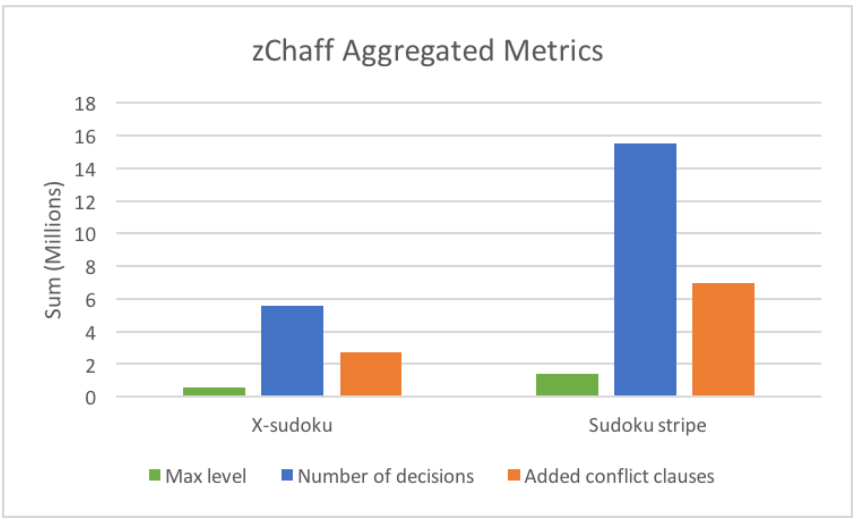


Figure 3: Aggregated metrics for the underdetermined subset of the puzzles using the zChaff solver. The vertical axis shows the sum (in millions) of the individual metrics for all 32772 puzzles. In each category, the sum for sudoku stripe exceeds the sum for x-sudoku.

**zChaff Statistics**

| | Maximum depth | | Number of decisions | | Added conflict clauses | |
|---|---|---|---|---|---|---|
| | x | stripe | x | stripe | x | stripe |
| Mean | 17.78 | 28.51 | 169.37 | 319.64 | 83.09 | 142.76 |
| Standard deviation | 9.10 | 9.06 | 223.79 | 249.05 | 118.11 | 135.93 |
| Max | 67 | 77 | 7876 | 3535 | 2900 | 1791 |

Table 1: Statistics for the underdetermined subset of the puzzles using the zChaff solver. As the aggregated metrics suggest, the mean for each metric is lower for x-sudoku than for sudoku stripe. Standard deviations are quite high for both puzzle types due a few puzzles requiring thousands of decisons and discovering thousands of conflicts. Notice that the maximum for number of decisions and added conflict clauses is actually *lower* for sudoku stripe than for x-sudoku.

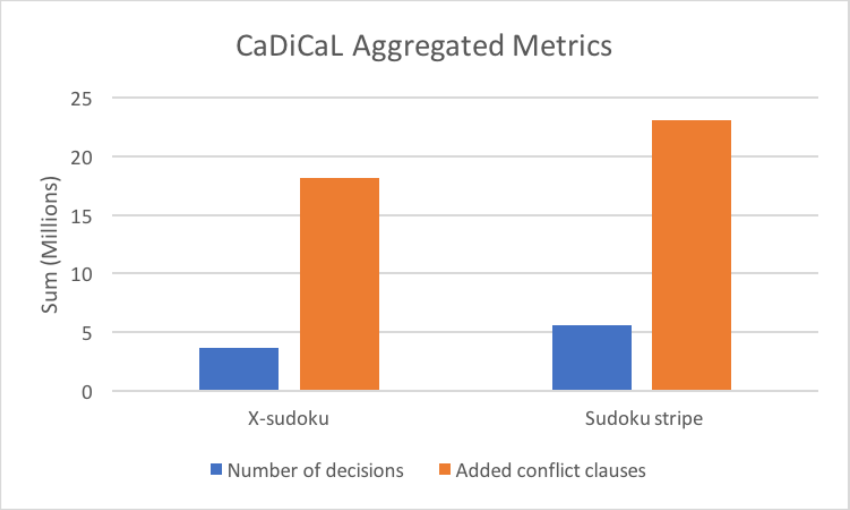


Figure 4: Aggregated metrics for the underdetermined subset of the puzzles using the CaDiCaL solver. The vertical axis shows the sum (in millions) of the individual metrics for all 32781 puzzles. In each category, the sum for sudoku stripe exceeds the sum for x-sudoku.

**CaDiCaL Statistics**

| | Number of decisions | | Added conflict clauses | |
|---|---|---|---|---|
| | x | stripe | x | stripe |
| Mean | 112.72 | 171.84 | 553.62 | 704.84 |
| Standard deviation | 161.77 | 173.01 | 1033.87 | 932.15 |
| Max | 2715 | 2849 | 20669 | 19236 |

Table 2: Statistics for the underdetermined subset of the puzzles using the CaDiCaL solver. As the aggregated metrics suggest, the mean for each metric is lower for x-sudoku than for sudoku stripe. Standard deviations are quite high for both puzzle types due a few puzzles requiring thousands of decisons and discovering thousands of conflicts. Notice that the maximum added conflict clauses is actually *lower* for sudoku stripe than for x-sudoku.

# 4 Interpretation

Our hypothesis appears too strong and is only partially validated by our evidence. We correctly predicted that average metrics would be lower for x-sudoku than for sudoku stripe, but our prediction that the same would be true of maximum metrics has been spectacularly refuted. Even when we restrict our attention to only those 32772 puzzles that require zChaff to make a decision, there are puzzles that are pathologically difficult as x-sudoku (in terms of number of decisions and added conflict clauses), but none are so pathologically difficult as sudoku stripe. It is not absolutely clear why these puzzles are so difficult as x-sudoku, but we note that relatively few hints lie on

either diagonal in such puzzles compared to the rest of the dataset. For CaDiCaL, there was little difference between the puzzle types in terms of the difficulty of the hardest puzzles in the set.

It is also interesting that, of the 49151 puzzles, 7 were valid instances sudoku stripe, but none were valid instances of x-sudoku. This accords with our notion of the comparative strictness of the x-condition: Since there are many more ways to satisfy the stripe condition than the x-condition, one should also expect there to be more puzzles that satisfy the stripe condition.

# 5 Conclusion

Our experiment supports our hypothetical claim that, given an arbitrary fixed set of puzzles, SAT-solving the puzzles as sudoku stripe is more difficult on average than as x-sudoku. Our experiment refutes our hypothetical claim that it is also more difficult in the worst case relative to that set. However, our experiment's support only extends to two solvers with default settings and one dataset.

Since our hypothesis is a general claim about solvers, a natural direction for future work is to repeat this experiment with different solvers or different settings. Since our hypothesis is also a general claim about puzzles, one could also repeat the experiment with different datasets, perhaps providing only a pathologically small number of givens for sudoku. It seems intuitive that variations of our average-case hypothesis should hold for puzzles larger than $9 \times 9$, but future experiments could demonstrate this rigorously.

# References


[1] Gihwon Kwon and Himanshu Jain. Optimized cnf encoding for sudoku puzzles. In *Proc. 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR2006)*, pages 1–5, 2006.

[2] Inês Lynce and Joël Ouaknine. Sudoku as a sat problem. In *ISAIM*, 2006.