The solving complexity of circular sudokus

Bram Kooiman & Florian Mohnert

University of Amsterdam

September 2017

1 Abstract

In an attempt to spice up the world of sudokus this paper introduces the "Circledoku", a circular sudoku. These CircleDokus have rules for rings, diagonals and wedges (which we will elaborate on shortly). The rules reflect those of the row-, column- and region-rules of regular sudokus. It is therefore believed that the difficulty of solving a circledoku is comparable to that of a square sudoku. This paper investigates that belief. All sudokus get harder to solve as they get bigger. The hypothesis therefore reads: "The relation between the amount of fields and the complexity of a circular sudoku puzzle is the same as the relation for square sudokus". To test this hypothesis the complexity of solving the two types of sudokus is measured by the average amounts of flips per assignment that the SAT-solver Walksat takes to solve the puzzles. We present some evidence that the complexity grows the same way for both types of puzzles, but that circular sudokus are generally harder to solve.

2 The Circular Sudoku

Regular sudokus are puzzles that need to be filled with digits 1 to 9. Every row must contain one and exactly one of every digit, as well as every column and some pre-defined 3-by-3 regions. There also exist sudokus with digits 1 to 4 or 1 to 16. The circledoku (of this example) is a puzzle that needs to be filled with digits 1 to 6. Every ring must contain every digit. The puzzle is also sliced up in 6 playfully named "pizza slices". A diagonal consists of two opposing pizza slices and a wedge consists of two predefined adjacent pizza slices. Wedges do not overlap.

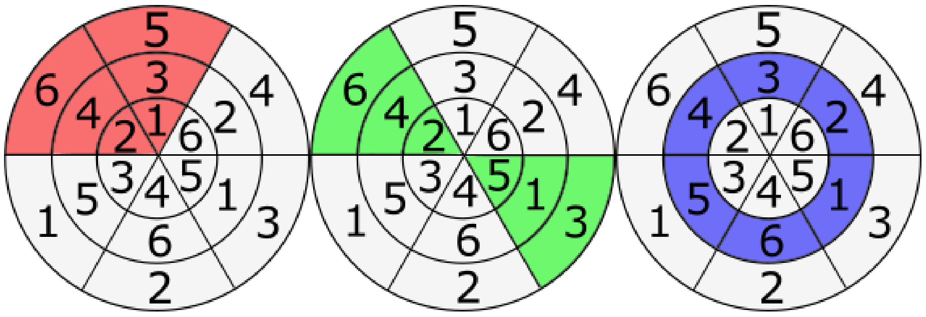

Figure 1: Rules for a ”circledoku”. All digits 1 to 6 must be present in each wedge(red), diagonal(green) and ring(blue).

These rules are similar to the row-, column- and region rules for regular sudokus, but not exactly the same. For instance, in a circledoku there are two fields where ring and diagonal overlap, but in regular sudokus there is only one field where a column and a row overlap. The hypothesis can be interpreted as our belief that this difference in rules won’t matter for the solving complexity of a circledoku. Acknowledging the difference, they are expected to be similar enough. Exactly that claim will be investigated in this paper.

3 Complexity and size

The most regular sudokus are square 9-by-9 sudokus. These will be refered to as square sudokus of order 3 (the order of a square sudoku is the square root of the length of the row). This is because there cannot exist an 8-by-8 sudoku (the region-rule would not work). A sudoku of order 2 is therefore 4-by-4. The relation between the order of a square sudoku and the amount of fields is $o = n^4$.
A circledoku’s order is its amount of rings. If a circular sudoku has 3 rings, it has 3 possible diagonals and 3 wedges. Note that this means that there are 6 “pizza slices”. The relation between the order of a circledoku and the amount of fields is $o = 2 * n^2$.
One could never justifiably compare a circledoku and a sudoku based directly on the complexity of solving them, for it will always be harder to solve a larger puzzle. In table 1 it can be shown that there exists no pair of sudoku and circledoku with the same amount of fields.

Order	Circular	Square
2	8	16
3	18	81
4	32	256
6	72	-
7	98	-
10	200	-

Table 1: amount of fields per order puzzle. There are no square or circular puzzles with the same amount of fields

Sudoku's are known to be solvable in polynomial time [1]. That means the relation between the upper bound of the complexity of solving a square sudoku and its number of fields obeys the formula k^n, where n the amount of variables and k a constant depending on the type of SAT solver used. In the worst-case scenario, k is equal to 2. Every variable is either true or false, and in the worst case, every variable-combination has been checked. Even though the upper bound is theoretically determined to be an exponential relation, it has been found time and time again that 2-SAT problems (problems with 2 variables per clause) are solvable in linear time. There seems to be a threshold at an average of 2.42 variables per clause below which the relation is linear and above exponential.

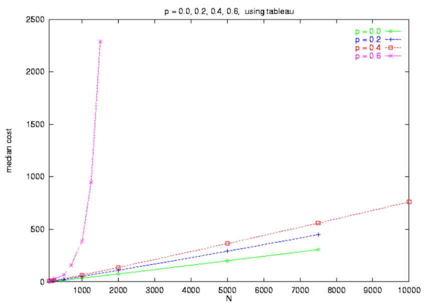

Figure 2: Shift from exponential to linear relation at 2.42 variables-to-clauses-ratio. Image gratefully borrowed from lecture slides

4 Hypothesis

A sudoku- and a circledoku problem have similar (note "similar" not "identical") rules. It is hypothesized that this similarity is reflected in the solving complexity for the puzzles. The two-part hypothesis reads as follows:

- "The relation between the amount of fields and the complexity of a circular sudoku puzzle is the same as the relation for square sudokus"

- "if the relation between the amount of fields and the complexity of a circular sudoku is exponential, they will agree on the same fit trough the data"

Because these puzzles have many 2-variable clauses, the average variables per clause is expected to be below 2.42, and therefore the relation is expected to be linear.

5 Method

The hypothesis is tested by plotting the data of amount of fields vs. solving complexity for both cases, and see to which degree they agree on the same relation. Puzzles to be compared are proper sudokus with half the fields open and half the fields given. 'Proper' sudokus are sudokus with only one possible solution. Puzzles are encoded in 'extended coding' [1]. In extended encoding we define all the constraints for the number 1 to 2n to be "at least"(d) and "at most"(u) present in each diagonal, wedge and ring. Also, every individual Cell should have only exactly one number. Lastly, the "Givens" of the puzzles need to be encoded in propositional logic.

$Encoding = Cell_d \wedge Cell_u \wedge Diagonal_d \wedge Diagonal_u \wedge Wedge_d \wedge Wedge_u \wedge Ring_d \wedge Ring_u \wedge Givens$

In the following we will present the propositional formulas for the constraints $Cell_d$ and $Cell_u$ to show how we thought about the encoding. We do not want to present too much notation and refer our readers to our github repository[1] for the exact encoding of the constraints. We represent every propositional variable by indices r,c representing rows and columns and v as possible values to take on. Hence we get,

$$Cell_u = \wedge_{r=1}^{n} \wedge_{c=1}^{2n} \vee_{v=1}^{2n} (r, c, v)$$

$$Cell_u = \wedge_{r=1}^{n} \wedge_{c=1}^{2n} \wedge_{v_1=1}^{v2n} \vee_{v_2=1}^{2n-v_1} \neg(r, c, v_1) \vee \neg(r, c, v_2)$$

For most of the encodings we have clauses of length 2 which stem from the "at most" part of the encodings, as we have to exclude all possibilities of each field in a constraint region to have the same value as another field in the same constraint region. There are also for each constraint region 2n clauses of length 2n that encode the at "least part" of the encodings. With this extended coding, the average variables per clause below 2.42. The ratio even decreases with size, which would mean that larger puzzles get (relatively) easier.

[1]github.com/florianmohnert/KR_CircleDoku.git

Order	Circular	Square
2	2.20	2.21
3	2.18	2.14
4	2.15	2.08
6	2.11	-
7	2.09	-
10	2.07	-

Table 2: ratio of clauses to variables for different order puzzles. Note that they are all below the threshold of 2.42

Solving complexity is measured in the average amount of flips required to assign a satisfiable solution by walksat[2]. Databases are generated by having the SAT solver pycosat return multiple possible assignments for an empty puzzle. These are all valid full puzzles. The puzzles are 'reduced' to a form with half the givens taken out. In order to still ensure properness, this is done one by one. Every step, one randomly determined given is taken out and the algorithm checks if there is still one and only one solution. If not, the given may not be taken out. If so, it can be removed and the next given is picked. Reducing the amount of givens is done randomly, and the resulting puzzles are not all the same.Pycosat is used to generate databases because it is easy to use and it is very fast. Walksat is used to solve the puzzles because it gives insight into the inner workings through metrics. Also, it is not used in the generating process which could possibly bias the results. Before a puzzle can be sent to a SAT solver, it must be represented in Conjuctive Normal Form (cnf). This is done in Python 3. In order to represent a circledoku with an array, we imaginatively 'cut it open' as demonstrated in Appendix 2.

6 Results

The red datapoints in figure 3 represent circledokus and the blue datapoints represent square sudokus. When plotted in exponential space, the data does not follow a straight line. The relation is not exponential (as was expected by the ¡2.4 variable-per-clause ratio). From the right hand plot in figure 3 it can be concluded that the relation is not linear either. The data seems to follow a root-shaped curve in exponential space. This may be a quadratic or log-linear relation. Whatever the relation, it is the same for the blue and the red data. They both construct the same root-in-exponential-space shape. The circledokus generally require more flips to assign and the variance within a given size of puzzles seems to be larger.

[2]www.cs.rochester.edu/u/kautz/walksat/

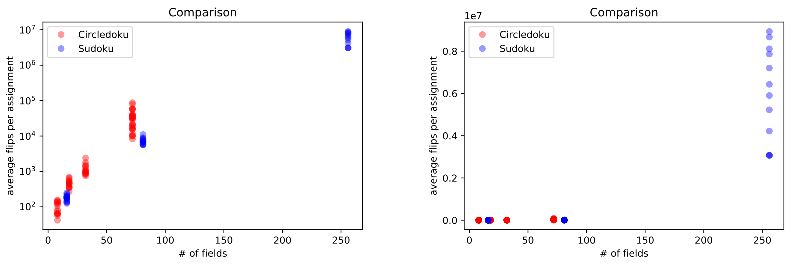

Figure 3: Comparison of complexity for circular and square sudokus on logarithmic (left) and non-logarithmic scale (right)

7 Conclusion

The solving complexity for circledoku's increases with size in the same way square sudoku complexity increases with size. This can be concluded from the shape in figure 3. From this figure it can also be concluded that circledokus are generally harder to solve than square sudoku's, because they require more flips per assignment with approximately the same number of fields.

8 Recommendations

- The conclusion is drawn from visual inspection of figure 4. It has however not been numerically determined. It may be appropriate to construct a fit through the blue data and a fit through the red data and see on which shape and parameters they agree best, or agree at all.

- Though it was initially the plan to evaluate circledokus with 10 rings (200 fields), these would not find assignment in reasonable time in WalkSat . WalkSat is a non-deterministic SAT solver. These generally work better on smaller problems. We feel this may be the heart of the problem with bigger puzzles. We recommend anyone to use a deterministic SAT solver on bigger puzzles.

- Due to time constrictions the dataset has been kept relatively small. We recommend fellow researchers to also solve circledokus of intermediate orders (we skipped 5-, 8- and 9-ring circledokus), go bigger (11- and 12-ring circledokus and use more puzzles per size. We used 15, which is very few. We recommend using 50 to 100, but in case of doubt even more will be even better.

9 Appendix

9.1 Notes to self

In the first attempts of generating full circledokus we used a brute-force algorithm. It picks a number at random, and places it on the field if the numbers on the other fields do not restrict this. In a later stage, when more of the fields were filled, it may be the case that none of the possible numbers qualify. In that case, the algorithm would start anew. This appeared to happen quite often, and the algorithm was hideously inefficient. In the end we chose to *use* the wheel instead of trying to re-invent it and use a SAT solver to find a satisfiable solution to an empty puzzle (of which it could find many). Generating a 4-ring puzzle with brute force could take anywhere between 1 and 90 minutes, whereas generating it with a SAT solver took a few seconds. The brute force algorithm was of course very simple and unoptimized, but we were still impressed with the sudden gain in efficiency and reliability of the process.

9.2 Numpy representation

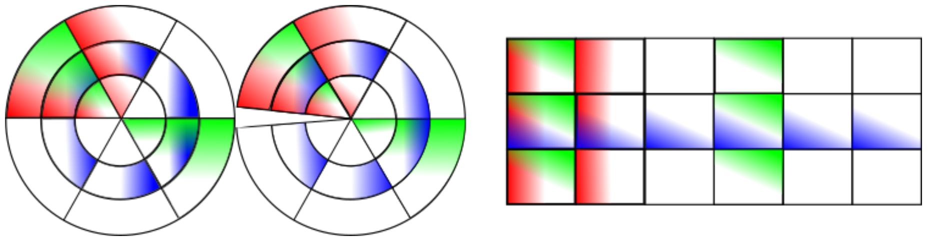

Figure 4: A circledoku with its rules for wedge(red), diagonal (green) and ring (blue) represented in a numpy.array

9.3 Github

Main files in the repo are:

- "Circledoku.ipynb": A jupyter notebook including all the functions to generate the databases for circular Sudokus (of different sizes) and encoding them to cnf. This file also exists as .py version.
- "Sudoku.ipynb" : jupyter notebook including all the functions to generate the databases for standard Sudokus (of different sizes) and encoding them to cnf.This file also exists as .py version.
- files like "CircleDokuDatabase_for_size_number_reduced" & "CircleDokuDatabase_for_size_number_full" , where "number" indicated the size of the respective puzzle and reduced or full refer to puzzles with half the fields as givens and fully solved puzzles respectively.

- The Results_plot file shows how we accumulated the data and build the plot from figure 3.

9.4 Youtube

A link to our presentation: www.youtube.com/watch?v=1zaF4vG2i2kfeature=youtu.be

References

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