Defeating Sudoku: A comparison between Sudoku difficulty

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081HV Amsterdam, The Netherlands

Abstract. This research aimed to get a better understanding about different indications of Sudoku difficulty. Depending on the research field, Sudoku difficulty commonly gets operationalized as Sudoku dimensions or the number of input numbers. Therefore, the applicability of both difficulty indications was tested on 1707 Sudoku using running time and number of backtracks as a representation. It was hypothesized that both Sudoku dimensions and number of input numbers have an effect on running time and the number of backtracks. Several independent t-tests were conducted to test the hypotheses. Against expectations, only the Sudoku dimension had a significant effect on the running time but not on number of backtracks. Neither did number of input variables have a significant effect. Therefore, based on our results, only Sudoku dimension can be used as an indication for Sudoku difficulty. Given the contradiction of our results with the findings of previous research we suggest to interpret them with caution.

Keywords: Sudoku · SAT-solver · DPLL-Algorithm.

1 Introduction

Sudoku puzzles were introduced in 1979 and have been fascinating puzzle solvers ever since. A Sudoku is a logic-based, combinatorial puzzle in which numbers are placed on a grid following specific rules. In classic Sudoku, the numbers 1-9 are placed in 81 cells on a 9x9 grid. The following rules have to be considered while placing the numbers: each number from 1-9 must occur only once in every column, each number from 1-9 must occur exactly once in every row, and each number from 1-9 must occur only once in each 3x3 block regions. Several input numbers, so-called givens, are included in the puzzle to be solved. The amount of givens varies, and well-posed Sudoku puzzles only have a single solution that has to be found.

Sudoku not only inherit an entertainment factor for ambitious puzzlers, they are additionally of interest for researchers of several fields. For example, on a neurocognitive level, the solving of number puzzles including Sudoku was linked to reasoning and problem solving capabilities [1]. Furthermore, artificial intelligence has also been tested using Sudoku. The solving of Sudoku puzzles on n^2xn^2 grids on nxn blocks was found to be NP-complete [2]. One technique that was found to reliably solve Sudoku is the SAT-solving method. This method interprets Sudoku as a Boolean satisfiability problem and determines whether there is an interpretation that satisfies a given Boolean formula. The theory behind SAT-solving algorithms will be addressed in detail in the next chapter.

1.1 Sudoku Difficulty

One important classification when solving Sudoku for humans and machines alike is the difficulty of the puzzle. For more complex puzzles, humans need a higher cognitive load and demonstrate reduced performance, as highlighted by van Gog and colleagues [3]. Their operationalization of the complex and simple puzzles was set as 4x4 grid Sudoku (easy task) vs. 9x9 grid Sudoku (complex task). The researchers found a significant difference between these conditions in terms of cognitive load and performance which emphasises the importance of dimensions of Sudoku.

On the other hand, research in Artificial Intelligence focused on Sudoku difficulty conceptualised as number of input numbers, or givens. For instance, [2] and [4] both mentioned the givens as an indication for difficulty.

Both approaches enhance difficulty in a different way. First, increasing the dimension of the Sudoku, for example from a 4x4 to a 9x9 grid, heightens the number of possible input for each cell. Therefore, a higher amount of data has to be processed to solve the puzzle. Second, reducing the number of givens increases the ambiguity of the puzzle and therefore makes it harder to place the correct number.

This research aims to investigate both approaches and compare the two in order to get a better understanding of the conceptualisation of Sudoku difficulty, as research in psychology and AI seem to differ in their approach of measuring difficulty. Therefore, the question arises whether the approaches are interdisciplinary applicable and both conceptualisations can be used in AI. In our research, we assume that dimensions as well as the number of givens can be conceptualised as an accurate conceptualisation of difficulty. The following hypothesis can be established from the mentioned research question.

H1: There is a significant difference in difficulty between Sudoku with a differing number of givens.

H2: There is a significant difference in difficulty between Sudoku with a differing dimensions.

In order to test this research question, a SAT solving algorithm, namely the Davis-Putnam-Logemann-Loveland algorithm (DPLL) was used. The running time and the number of backtracking calls of the DPLL with two different implemented heuristics were evaluated as difficulty measurement. We chose the running time and the number of backtracking calls as an operationalization of difficulty because it quantifies the effort that the algorithm has to invest to solve the Sudoku. The hypotheses were tested by comparing two Sudoku dimensions and two groups with a high and a low proportion of number of givens with each other.

2 Related Work

As already stated above, humans solve Sudoku differently as they do not follow one consistent solving strategy like an algorithm [5]. Several strategies are combined and therefore these solving method does not bear much resemblance to the backtracking method used by computers. As aforementioned, one way to classify Sudoku difficulty for humans is to increase the dimension of a Sudoku [1], with the same proportion of givens. Moreover, machines were found to have an increased running time with a

decreasing number of givens [2], since the algorithm has fewer information to base the backtracking search on it needs a higher number of backtracking steps to solve the puzzle. Furthermore, the puzzle gets increasingly difficult the greater the gap in-between the input numbers as well [6] but has to have at least 17 input numbers to be solvable [7].

2.1 Solving Sudoku with SAT solving

SAT solving was proven to be an effective method to solve Sudoku [2]. SAT-solvers work with a Boolean formula, or clause set. This set can be described as a conjunction consisting of a number of disjunctions, also called CNF clauses. Each clause has several variables that get assigned either *True* or *False* values. If the clause set can be evaluated to *True*, the problem is satisfiable, and thus solvable. On the other hand, if the clause set evaluates to *False*, the problem is unsatisfiable and thus cannot be solved. Complete SAT-solvers can give out the values of the assign values, which in our case are the number for each Sudoku cell. One clause gets satisfied if at least one variable in it is assigned the value *True*. The clause set is satisfied if all clauses are assigned the value *True*. In the representation of a Sudoku, the clause-set can be seen as the Sudoku rules and potential numbers for each cell. For this research, the most common SAT algorithm, the DPLL, was used.

The approach SAT-solvers use for evaluating the given problem as satisfiable or unsatisfiable is to reduce the complexity of the clause set by reducing and simplifying it in each search step. Several simplification rules can be used.

- **Tautology Rule**: If a clause contains $\neg$ P $\vee$ P the clause, is satisfied.
- **Pure Literal Rule**: If a literal is only occurring in the positive or negative form, it can be assigned *True* (or *False* if the literal is negated)
- **Unit Clause Rule**: If there is a clause consisting of only one literal, the literal can be assigned *True* (or *False* if the literal is negated).

After applying the simplification rules, DPLL uses tree search combined with unit resolution. The search tree splits into the different literals found in the clause-set. One literal is picked and gets a specific value assigned. For each newly assigned variable, the unit rule gets applied again for the clauses. After simplifying the clause-set. Then, the search tree moves to a new level and another variable gets assigned and the same procedure takes place. If a variable needs to be assigned different values in order to satisfy two different clauses, a conflict occurs. If this happens, the search tree is moved up one level and the previous variable gets assigned a different value. A visualization of the described process can be found below.

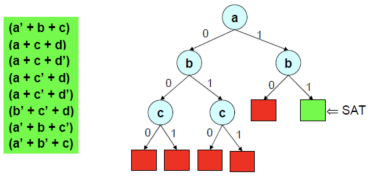

Fig. 1: Example of a Search Tree

3 Design Decisions

3.1 DPLL Algorithm

The SAT solver chosen for this experiment is the DPLL algorithm. This procedure is well known for its efficiency in SAT solving, and works via a branching procedure to reduce the search space [8]. This algorithm was implemented in Python through a recursion function in which simplifies the clauses, selects new literals of which to branch from, and checks for the solution. The aforementioned pure literal rule was not used. The rules and games were both used in DIMACS format, in which the integers represent the row, column, and value of the squares respectively. These rules as implemented above increase efficiency of the algorithm through the reduction of the search space, and decreased amounts of backtracking.

Algorithm 1 DPLL Recursion

```
1: DPLL recursion (clauses, literal, sol)
2:     Create copy of sol
3:     Add literal to copy
4:     Apply simplification rules to clauses
5:     if empty list of clauses then
6:         no solution,
7:     else 1 remaining clause
8:         solution found, terminates
9:         checks for unit clauses
10:        if unit clause then
11:            select as next literal
12:        else
13:            heuristic determines next literal
14:        end if
15:    end if
16:    DPLL recursion (clause, -literal, sol)
17:    if not true then
18:        return DPLL recursion (clause, literal, sol)
```

4 Heuristics

Branching heuristics are often implemented in DPLL algorithms as they can increase the efficiency in finding a solution. The standard DPLL method simply chooses a literal to branch from probabilistically, whereas heuristics choose the literal with decision criterion. The two heuristics implemented include Jeroslow-Wang (One-Sided)(JWOS) and the Dynamic Largest Individual Sum (DLIS). Two different heuristics are implemented separately due to their difference in selection mechanisms. Although both have been proven to be effective methods, there is no way to know which is better a prior

to application. Thus, both are used. JWOS is a formula in which weights are assigned to each literal based on its frequency and the lengths of the clauses in which it occurs. In every clause in which a literal occurs, the negative length of this clause is taken to the power of 2. The sums for each literal are calculated, and the literal in which has the greatest weight is selected to begin the next branching sequence. This formula can be seen directly below.

$$J(l) = \sum_{l \in w, w \in \varphi} 2^{-|w|} \tag{1}$$

The DLIS heuristic begins by selecting the literal in which occurs the most, regardless of the assigned truth value. However, the truth value of this literal moving forward is determined by whether this value occurs more as true, or as false. Whichever truth value of this literal is the most frequent is the truth value that is assigned.

5 Experimental Design

As independent variables, the effect of increasing dimension and number of givens on the performance of the algorithm were investigated. To test these effects, 1000 Sudoku with the dimensions 4x4 and 707 Sudoku with the dimension 9x9 were taken into account. Furthermore, within the data set of the dimensions, the Sudoku were split into two groups, one with fewer givens and one with many givens. The proportion of the number of givens was similar for both dimensions. For the 4x4 Sudoku, the group with fewer givens had 4 input numbers, which equals 25%. The group with many input numbers had 6 input numbers, which equals 38%. For the 9x9 Sudoku, the group with fewer givens had the input numbers 21-24, which equals 25-27%. The group with many input numbers had 26-29 input numbers, which equals 33-37%. We chose to include a range of givens in the two groups in the 9x9 dimension to ensure a sufficient amount of testing data. A visualisation of our experimental set-up can be found in Figure 2. The dependent variables are the performance of the algorithm, operationalized as running time of the algorithm and the number of backtracks used.

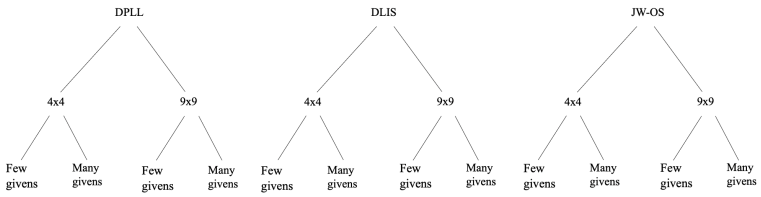

Fig. 2: Experimental set-up

For the statistical analysis, we first investigated the descriptive statistics of the two assessed dependent variables. Then, several independent t-tests were conducted to analyse whether the different experimental groups significantly differ from each other in terms of performance.

6 Experimental Results

6.1 Descriptive Statistics

A table of the summarized descriptive results can be found below in Table 1. As can be seen, the descriptives highlight the crucial difference between the number of dimensions for both measurements. The 4x4 Sudoku have a considerably faster running time and zero number of backtracks while both measurements increase significantly for the 9x9 Sudoku. Also, the standard deviation increases much for the higher dimension.

Not only the performance differences between the dimensions but also between the heuristics gives interesting insights. In respect to the heuristics, the JWOS heuristic led to an increase in running time in comparison to the standard DPLL algorithm and the DLIS. The running times of the DPLL and DLIS are indistinguishable. Moving on, the number of backtracks displays a much more salient distinction in performance, with JWOS having led to a substantial higher amount of backtracking for the 9x9 dimension than the DPLL and DLIS. Interestingly, the running time and backtracks for the 4x4 Sudoku are (very) similar for all three heuristics.

Table 1: Description of Results

Inputs Dimension		**Runtime**			**Backtracks**		
		DPLL	**DLIS**	**JWOS**	**DPLL**	**DLIS**	**JWOS**
4x4	**Group 1**	0.13, (0.06)	0.16, (0.77)	0.16, (0.08)	0	0	0
	Group 2	0.05, (0.12),	0.15, (0.07)	0.15, (0.07)	0	0	0
9x9	**Group 1**	45.34, (17.17)	44.86, (17.86)	55.67, (74.71)	3.15, (4.71)	3.15, (5.43)	86.18, (317.86)
	Group 2	45.98, (14.63)	47.23, (16.12)	51.25, (38.89)	3.56, (6.26)	4.76, (5.77)	75.14, (169.06)

Note. The results in the above described table are presented as mean (SD)

6.2 Hypothesis Testing

To test the hypotheses, we conducted 18 independent t-tests to compare the measure running time and number of backtracks between two Sudoku dimensions, two amounts of givens, and three heuristics. An overview of the testing results can be found in Figure3. As highlighted in the figure, several significant differences can be distinguished. Regarding running time as a measurement, all t-tests comparing the two dimensions had a significant p-value of 0.0. This emphasizes that the running time of the 9x9 Sudoku was significantly slower than the 4x4 Sudoku. Regarding the two groups of number of givens, there was only one significant difference between the groups of the 4x4 Sudoku of the DPLL algorithm.

Regarding the number of backtracks as a measurement, there was also only one significant difference between the two input groups of the 9x9 Sudoku with the DLIS heuristic. All other p-values were not significant and therefore did not imply a significant difference in number of backtracking for these groups.

7 Discussion

Sudoku difficulty for humans has been operationalized in some research according to dimensionality[3], whereas AI research has instead operationalized difficulty according to the amount of given inputs[4].

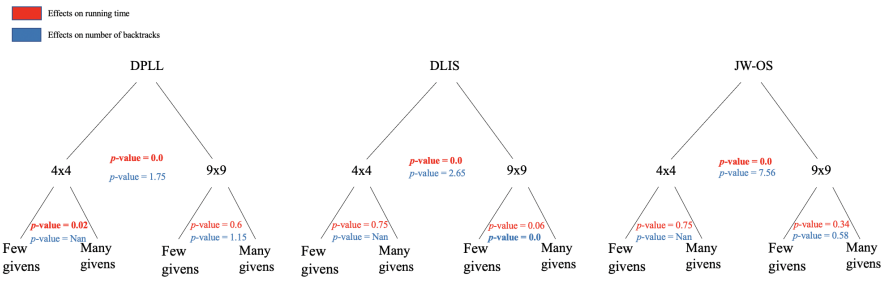

Fig. 3: Experimental Results

The results seem to indicate potential commonalities between the two in terms of how puzzle difficulty may be conceptualized, namely concerning dimensionality. Dimensionality did show a significant effect, as the 9x9 Sudoku had substantially longer running times. As the 9x9 puzzles containing roughly 5 times the amount of cells, therefore a highly increased number of clauses, the increase in running time is not necessarily unexpected. However, dimensionality did not have a significant effect on the number of backtracks. This may mean that the higher dimensionality took more time due to the increased amount of clauses, but did not run into significantly more errors. Considering the puzzles have identical rules, it is reasonable that an increase in more cells would not necessarily lead to more conflicts. However, these results may still contribute to the field of AI in terms of how SAT solvers may further conceptualize Sudoku difficulty: although an increase in dimensionality did not lead to more conflicts, it did indeed lead to far greater running times. Although a greater number of clauses seems likely to lead to longer running times, due to the significance in this difference for all instances, dimensionality may indeed be a way to operationalize difficulty. Therefore, in accordance with our operationalization of difficulty, we move to accept Hypothesis 1.

In contrast however, the results did not demonstrate overall significant effects on the number of givens. The number of givens only proved to significantly effect two conditions: Sudoku of 4x4 dimensionality as solved with standard DPLL, and Sudoku of 9x9 dimensionality in which were solved with DLIS. Therefore, we move to reject Hypothesis 2 and contribute these two significant results to unspecifiable reasons.

7.1 Limitations and future research

As in other research, we encountered several limitations. First, the operationalization of Sudoku difficulty could have been too simple. Even though we found statistically significant results for dimensions as a classification as difficulty, it is surprising that the number of givens did not show more significant results, especially since it is a common classification for difficulty in previous AI research [2]. One possible explanation for this surprising finding is that the groups we established did not differ sufficiently in the distinction of the proportional amount of givens. We included a range of number of givens for the 9x9 Sudoku, in order to have a sufficient amount testing data. Thus, the gap between between the two groups could be too close to the same proportion of

input numbers. Future research could increase the gap between both groups in order to have a clearer distinction between easy and difficult Sudoku.

Moreover, another limitation of this research were surprising results in terms of running time. Even though the hypotheses indicated an increase in running time (and number of backtracks) and more CNF clauses have to be taken into consideration, the amount of increase was astonishing. The 9x9 Sudoku increase the running time by a factor of around 350 in comparison to the 4x4 Sudoku. One possible explanation could be an inefficient representation of the problem in the code. Due to time constraints, we did not find a better solution for improving our algorithm. In the future, changing the representation and functions to efficiently apply the algorithm could help improving the running time.

Last but not least, the descriptive results imply noticeably higher number in backtracks for the 9x9 Sudoku with the JWOS heuristic. This means that the JWOS heuristic ran into a much higher number of conflicts while solving the algorithm and hints on the JWOS heuristic to be unsuitable for our problem representation, namely solving a Sudoku. It is advised for future research with the same aim to consider different types of heuristics.

8 Conclusion

To conclude, this research offered interesting insights into the operationalization of Sudoku difficulty. In favour of our hypotheses, an increasing dimension of the Sudoku can be seen as indication for a higher difficulty measured in running time, however not in terms of number of backtracks. Second, the second hypothesis that indicates number of givens as an indication for Sudoku difficulty was rejected. The mixed results of this research make it hard to draw a clear conclusion. Our research implies Sudoku dimension as an indicator for difficulty but not number of givens. Given the successful operationalization of difficulty as number of givens by previous research [2][4] and the astonishing descriptive statistics of this research, we suggest that more research is needed in this matter to give a precise conclusion about the indication of Sudoku difficulty.

References

1. O. Thompson-Bradley, S. Barrett, C. Patterson, and D. Craig, "Examining the neurocognitive validity of commercially available, smartphone-based puzzle games," *Psychology*, vol. 3, pp. 525–526, 07 2012.
2. M. Henz and H.-M. Truong, *SudokuSat—A Tool for Analyzing Difficult Sudoku Puzzles*, pp. 25–35. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.
3. T. Van Gog, L. Kester, and F. Paas, "Effects of concurrent monitoring on cognitive load and performance as a function of task complexity," *Applied Cognitive Psychology*, vol. 25, no. 4, pp. 584–587, 2011.
4. C. Xu and X. Weng, "The model and algorithm to estimate the difficulty levels of sudoku puzzles," *Journal of Mathematics Research*, vol. 1, 08 2009.
5. R. Pelánek, "Difficulty rating of sudoku puzzles: An overview and evaluation," 03 2014.
6. R. Béjar, C. Fernández, C. Mateu, and M. Valls, "The sudoku completion problem with rectangular hole pattern is np-complete," *Discrete Mathematics*, vol. 312, no. 22, pp. 3306–3315, 2012.

7. G. McGuire, B. Tugemann, and G. Civario, "There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem," 2013.
8. B. P. Frank Van Harmelen, Vladimir Lifschitz, *Handbook of Knowledge Representation*. Amsterdam ; Boston: Elsevier, 2008.