From Deterministic to Random:

Evaluating the effect of Randomness in SAT-solver heuristics

Knowledge Representation

Prof. dr. F.A.H. van Harmelen

October 2, 2019

1. Introduction

Propositional satisfiability (SAT) is the problem of deciding, for a formula in propositional logic, whether there exists a satisfying truth assignment for its variables. The problem has numerous applications in Artificial Intelligence (e.g. [1]), as well as in many other areas of Computer Science. An example of a set of problems that can be written in propositional logic are Sudokus, a well-known logic-based puzzle game. Sudokus are an NP-complete problem [2], as first shown in [3] via a reduction from the Latin Square Completion problem [4].

In the last decennia great progress has been made on SAT solvers based on the backtrack-search algorithm of Davis, Putnam, Logemann and Loveland (DPLL) [5]. One key aspect of backtracking-based search SAT algorithms is how value assignments are selected at each step of the algorithm, i.e. the branching heuristics [6]. In the past several heuristics have been applied, analyzed and compared to each other. In this report several different alterations of branching heuristics will be compared to gain insight in the influence of randomness on a very basic and deterministic heuristic.

2. Design of algorithm

The algorithm that was used in this study was a generic, recursive DPLL algorithm. The DPLL algorithm consists of different steps. First, checks on satisfiability or unsatisfiability were implemented. Next the clause set was simplified by checking and setting variables for tautologies (only in the first iteration), pure literals, and unit clauses. If the clause set could not be simplified anymore a split was selected by using one of the implemented heuristics, after which the clause set was simplified again. If a certain split lead to an unsatisfiable clause set basic backtracking was used to undo the ‘wrong’ split. This process was repeated until the problem was solved. The input for the DPLL were files in DIMACS format containing the sudoku rules and the sudoku puzzles. The output of the puzzle was also in DIMACS format.

3. Heuristics

Besides the RANDOM heuristic the decision was made to implement four other heuristics, i.e. DLIS, RDLIS, PDLIS and RPDLIS (explained in table 1). It is interesting to compare these different alterations of heuristics as they have a different degree of randomness and scientific research on these specific variations is quite scarce. Only a few studies, such as the one by Marques-Silva, made a comparison of RAND, DLIS and RDLIS [6], but in that

study gaining knowledge on the impact of randomness was not the goal. Most heuristics applied in SAT solvers are based on a deterministic approach, therefore it may be interesting to compare this approach to more probabilistic ones. Table 1 contains a description of the heuristics.

Table 1. Definition of heuristics

Heuristic	*Description*
RANDOM	Randomly select one of the yet unassigned variables and assign a randomly chosen truth value to it.
DLIS	Dynamic Largest Individual Sum. Count the number of unresolved clauses in which a given variable *x* appears as a positive literal (C_p) and as negative literal (C_n). Select the variable with the largest individual sum, assigning it to true, if $C_p \geq C_n$ or false, if $C_n \geq C_p$.
RDLIS	Random Largest Individual Sum. A variation of DLIS randomly selecting the truth value to be assigned to a given selected variable instead of comparing C_p and C_n.
PDLIS	Probabilistic Dynamic Largest Individual Sum. A variation of DLIS, where variables get a chance proportional to their relative frequency, assigning it to true, if $C_p \geq C_n$ or false, if $C_n \geq C_p$.
RPDLIS	Random Probabilistic Dynamic Largest Individual Sum. A variation of PDLIS randomly selecting the truth value to be assigned to a given selected variable instead of comparing C_p and C_n.

4. Hypothesis

Previous research has shown that branching heuristics do have impact on the performance of SAT solvers [6], [7], although scientific research on specific probabilistic variations appears to be quite scarce. Thus, the decision was made to compare five specific branch heuristics (DLIS, RDLIS, PDLIS, RPDLIS and RANDOM) within the plain DPLL algorithm. The goal of this experiment is to investigate the influence of gradually adding randomness to a heuristic.

Marques-Silva found that, when comparing the DLIS, RDLIS and RANDOM heuristics, the RDLIS outperformed both DLIS and RANDOM. The worse performance of DLIS was attributed towards being a greedy algorithm, on the other hand RANDOM probably had a worse performance due to the unstructured nature of the heuristic. Therefore, the hypothesis was that there is a certain optimal randomness in using branching heuristics, where no randomness and complete randomness perform worse than everything in between.

5. Method

In the current study several different alterations of heuristics were used to gain insight in the influence of randomness on the very basic and deterministic DLIS heuristics. To test the hypothesis a set containing 1000 9x9 Sudokus was used. Furthermore, the metrics that were used included Time, Putnam Calls, Splits, and Backtracks. The total number of Putnam Calls for a given Sudoku is equal to the number of Splits + the number of Backtracks + 1 (for the initial simplification). After solving each Sudoku, the metrics were saved and were afterwards used to make comparisons between the heuristics.

All of the 5 (DLIS, RDLIS, PDLIS, RPDLIS, and RANDOM) heuristics were tested using the exact same test set consisting of 200 randomly selected Sudokus from the 1000 Sudokus set. The order of these heuristics is shown in this way because of the randomness in each heuristic. DLIS is completely deterministic, while RDLIS has a bit of randomness. The randomness is obviously the highest in the RANDOM heuristic. For the statistical analyses 4 one-way ANOVA’s with post-hoc tests were used to test for significant differences between the heuristics. For these analyses the alpha was set to .05.

6. Results

Firstly, each of the metrics used were visually inspected on homoscedasticity, normality, and outliers. The data from all the heuristics had several outliers. These outliers were not removed, because they are to be expected since the greediness of the more deterministic approaches may cause it to be forced to traverse more of the search space. On the contrary the more probabilistic heuristics may have a hard time finding the right split in some Sudokus. The results of the experiment are schematically shown in table 1.

	Time		Putnam Calls		Splits		Backtracks	
	M	SD	M	SD	M	SD	M	SD
DLIS	0.898	0.67	175.6	234.9	92.3	119.7	82.3	115.3
RDLIS	0.886	0.73	151.2	252.6	77.8	127.7	72.3	125.0
PDLIS	0.528	0.13	15.3	20.8	8.2	11.0	6.1	10.0
RPDLIS	0.524	0.13	13.5	19.7	7.1	10.2	5.5	9.5
RANDOM	0.505	0.12	12.7	20.7	6.6	10.7	5.1	10.0

The ANOVA's were carried out with the different heuristics as independent variables and the four metrics as dependent variables. The tests showed significant differences between the heuristics for every metric, Time, $F(995, 4) = 40.58$, $p < .000$, Putnam Calls, $F(995,4) = 56.43$, $p < .000$, Splits, $F(995, 4) = 59.46$, $p < .000$, and Backtracks, $F(995, 4) = 53.31$, $p < .000$. Post-hoc tests revealed that the DLIS and RDLIS performed significantly worse than the other three algorithms, all $p < .000$. Between DLIS and RDLIS, $p > .352$, and between PDLIS, RPDLIS and RANDOM, $p > .987$, no significant differences were found.

7. Conclusion

The goal of this experiment was to determine if added randomness had an influence on the performance of the basic and deterministic DLIS heuristic in solving Sudokus. The results confirmed the expectation that DLIS is a greedy heuristic, which consequently leads to a relatively bad performance. When considering the research from Marques-Silva it was expected that the RDLIS would perform better than the DLIS due to being less greedy. This was not the case in the current study, since the results showed that RDLIS had a similar performance to the greedy DLIS.

Conversely, the probabilistic PDLIS and RPDLIS performed a lot better than their deterministic counterparts. The added randomness in the form of a probabilistic split had a positive effect on the total performance of these heuristics. However, the performance of the RANDOM heuristic was as good as the PDLIS and RPDLIS, which means that in solving Sudokus an actual 'smart' heuristic may not be needed for a good performance. This is an

unexpected result since most heuristics have been developed to improve the performance of a SAT solver by making 'smart' splits. It is possible that the Sudoku-problem might require a completely different strategy than the strategies that are normally used for SAT-problems. Based on the results of the current study it might be interesting to select heuristics that are specifically designed for solving Sudokus. For example, based on how humans solve sudokus it might be interesting to research if a literal count heuristic that selects the lowest count would perform better.

References

[1] H. Kautz, B. Selman. Planning as satisfiability. In Proceedings of the 10th European conference on Artificial Intelligence, 1992.

[2] Michael R Garey and David S Johnson. Computers and Intractability: A Guide to the Theory of NP-Completness. WH Freeman: New York, 1979.

[3] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. IEICE Transac- tions on Fundamentals of Electronics, Communications and Computer Sciences, 86(5):1052–1060, 2003.

[4] Charles J Colbourn. The complexity of completing partial Latin squares. Discrete Applied Mathematics, 8(1):25–30, 1984.

[5] M. Davis, G. Logemann and D. Loveland. A machine program for theorem proving. In Communications of the ACM, (5): 394-397, 1962.

[6] João Marques-Silva: The Impact of Branching Heuristics in Propositional Satisfiability Algorithm, 1999.

[7] Marc Herbstritt and Bernd Becker, Conflict-Based Selection of Branching Rules. In International Conference on Theory and Applications of Satisfiability Testing, p. 441-451, 2003.