Czech Technical University in Prague

Faculty of Information Technology

Comparison of GSAT and ProbSAT Algorithms

Anastasiia Solomiia Hrytsyna

November 24, 2022

B221 - Winter semester 2022/2023

EXECUTIVE SUMMARY

This examination is dedicated to the problem of effectiveness of 2 different algorithms. The ProbSAT and GSAT (with the selected parameters - see **Introduction**) have been used to solve 3-SAT hard instances and then their output performance has been compared with the use of numerous metrics. In the end, our study proves that the ProbSAT algorithm does not have worse performance than the GSAT, but in most cases its efficiency is even better.

INTRODUCTION

In this study the comparison of 2 different algorithms (GSAT and ProbSAT) was carried out. The main goal of the research is to determine which of the algorithms has a better performance (faster finds the solution for the problems with different complexity). So, the central question is next:

Does Prob-SAT algorithm have better performance in comparison to the GSAT for solving 3-SAT hard instances (where clauses-to-variables ratio is approximately 4.4) with the next parameters:

GSAT: p = 0.4, num_of_tries = 10000

ProbSAT: c_m = 0, c_b = 2.3, num_of_tries = 10000, max_num_of_flips = ∞

More details about the task may be found on the link.

Therefore, to answer this particular question a few experiments have been performed. We ran both of the algorithms on various datasets (which have different number of clauses and variables) and compared the results of their performance down below.

MATERIAL

Input Data

As an input data (stored in *data* folder) we used formulas in CNF and DIMACS format with max clause length of 3 variables. Algorithms run on these datasets: *uf20-91.zip*, *uf50-218.zip* and *uf75-325.zip (reduced)* (source here from SATLIB Collection) 10000 times (on the last dataset 10 times more to balance smaller dataset size) to avoid any biased results. Selected input data corresponds to the main task and has different solving complexity (number of clauses and variables: 20/91, 50/218, 75/325), so the algorithm's output was fair.

Necessary Programs

As it was already mentioned 2 non-complete algorithms (stored in *algorithms* folder) have been used: GSAT (link for the pseudocode and some explanation how the algorithm works) and ProbSAT (link for the pseudocode and some explanation how the algorithm works). All of the constant input parameters for them were discussed above.

To run both of the algorithms a short script (*generate_results.sh*) was used. Then the output results have been cleaned (some commands for it may be found in *probsat_data_cleaning.txt* and a helper file *rows_to_delete*) and stored in the *results* folder. Finally, performance comparison was implemented in Python and may be found in *results_comparison.ipynb*.[1]

[1] All of these programs are available in the submitted *hrytsyna_DU1.zip* archive.

All in all, the main output value to compare was the number of iterations (or flips) to find a successful solution (if it exists at all).

In addition, to perform a better numerical and graphical comparison these metrics have been chosen: **difference of iteration, number of iterations, average values and other basic metrics, cumulative distribution function** (see below). All of these metrics may clearly define what algorithm is a dominant one - that is why they have been chosen.

RESULTS

For the final evaluation of performance we have 30 million records, where each record consists of a number of clauses both algorithms solve at once, number of GSAT iterations and ProbSAT flips to find a solution (see *results_comparison.ipynb*).

Notice that not every GSAT run found a solution to the problem. Due to the limitation (max. number of acceptable iterations) some of the runs finished unsuccessfully, so the ratio of partly solved clauses is shown on Fig. 1 (on the left). On the other hand, ProbSAT solves all of the problems, as it does not have any limitations. That is why on the right side of Fig. 1 is shown ratio of clauses the solution of which takes more iterations than the limit.

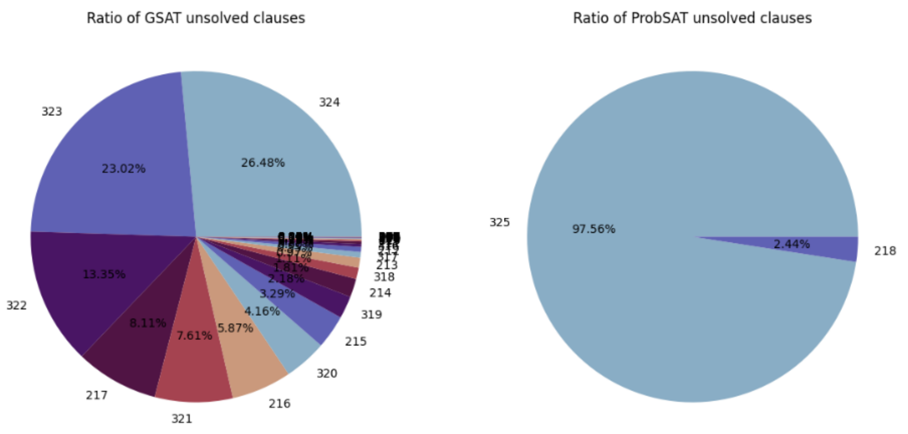


Fig. 1: Ratio of Unsolved Clauses

All of the rows, which contain unsolved clauses or the solution of which is out of the limit have been deleted. Totally 9.31% of the outputs have been deleted and the remaining dataset has the next ratio:

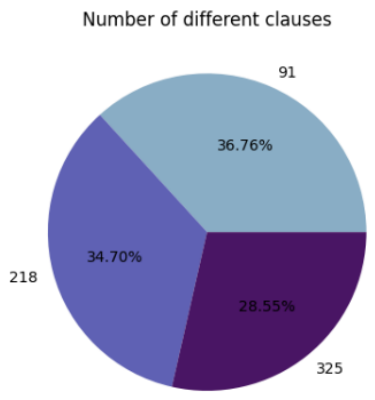


Fig. 2: Ratio of Solved Clauses in the Examined Dataset

In the next steps the dataset with successful solutions has been tested with such metrics:

Difference of Iterations

In this section we counted the difference between ProbSAT flips and GSAT iterations. As we may see from Fig. 3 the more clauses we have, the greater the difference in the algorithm performance is. Also, interquartile range (IQR) seems to have more negative values than the positive... It means that **ProbSAT needs less flips for finding the right solution, so it is faster.**

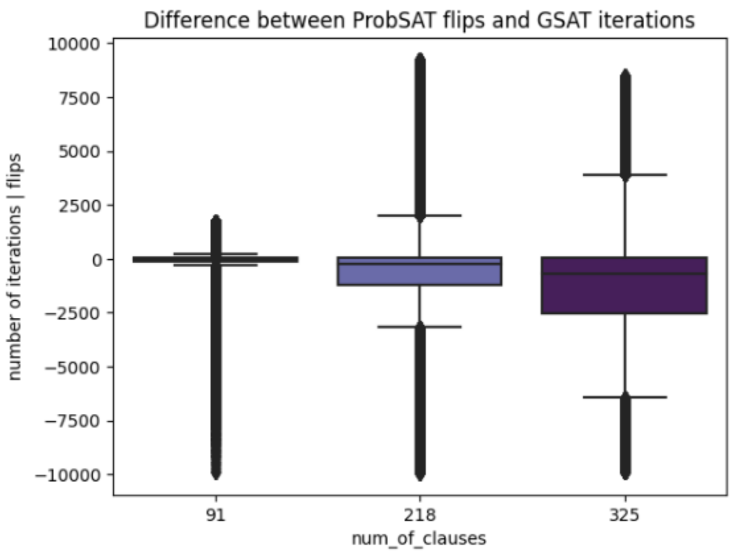


Fig. 3: Boxplot for the Difference (iterations - flips) Depending on the Clauses

In addition, we may notice the same trend in the separate boxplots for each algorithm (see Fig. 4).

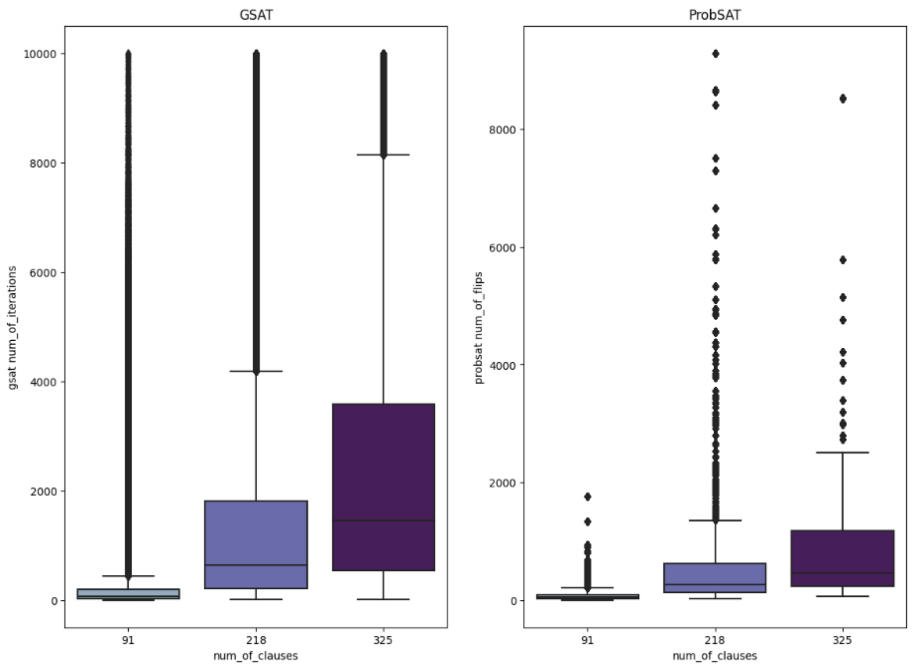


Fig. 4: Boxplot for the Each Algorithm Depending on the Clauses

Number of Iterations

In this step (see Fig. 5) we may notice that mostly **ProbSAT needs less steps to find a solution than the GSAT. It also has a lower probability to perform a lot of flips, but to finish solving the problems sooner.**

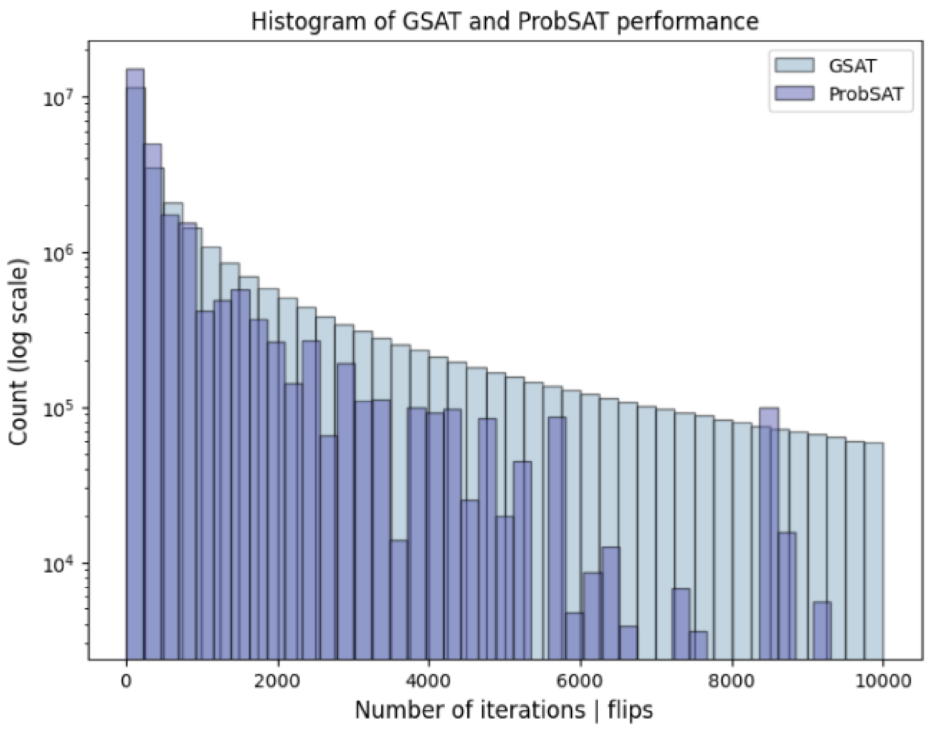
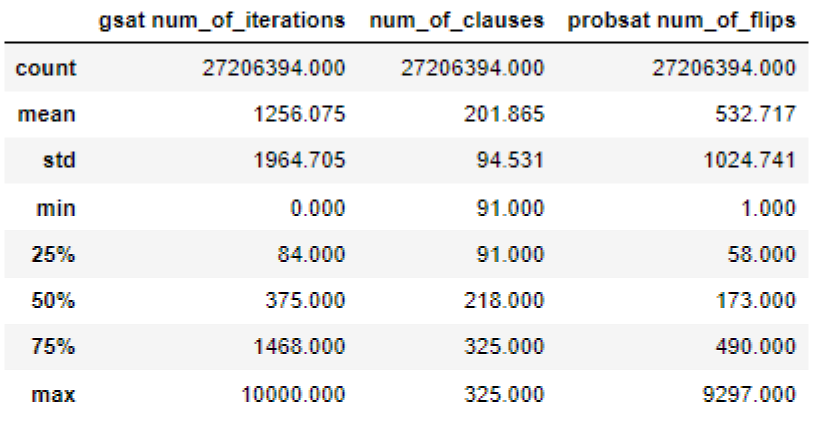


Fig. 5: Histogram of Steps Needed to Find a Solution

Average Values and Other Basic Metrics

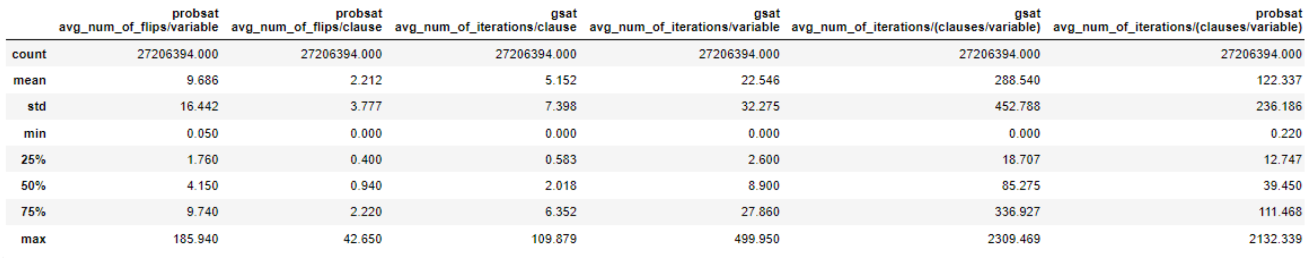
Down below the table of some fundamental metrics is displayed (see Fig. 6). **We may notice that on average ProbSAT needs less steps to find a solution** (see mean value, 25%, 50%, 75%)**. Further the standard deviation of flips distribution is smaller than the GSAT one.**



	gsat num_of_iterations	num_of_clauses	probsat num_of_flips
count	27206394.000	27206394.000	27206394.000
mean	1256.075	201.865	532.717
std	1964.705	94.531	1024.741
min	0.000	91.000	1.000
25%	84.000	91.000	58.000
50%	375.000	218.000	173.000
75%	1468.000	325.000	490.000
max	10000.000	325.000	9297.000

Fig. 6: Table 1 of Some Fundamental Metrics

Additionally, in this section I counted an average number of steps on variable, clause or both (see Fig. 7). **Here we may notice the same trends as it was mentioned above.**



	probsat avg_num_of_flips/variable	probsat avg_num_of_flips/clause	gsat avg_num_of_iterations/clause	gsat avg_num_of_iterations/variable	gsat avg_num_of_iterations/(clauses/variable)	probsat avg_num_of_iterations/(clauses/variable)
count	27206394.000	27206394.000	27206394.000	27206394.000	27206394.000	27206394.000
mean	9.686	2.212	5.152	22.546	288.540	122.337
std	16.442	3.777	7.398	32.275	452.788	236.186
min	0.050	0.000	0.000	0.000	0.000	0.220
25%	1.760	0.400	0.583	2.600	18.707	12.747
50%	4.150	0.940	2.018	8.900	85.275	39.450
75%	9.740	2.220	6.352	27.860	336.927	111.468
max	185.940	42.650	109.879	499.950	2309.469	2132.339

Fig. 7: Table 2 of Some Fundamental Metrics

Cumulative Distribution Function

Last but not least, we tested both algorithms for their CDF. And, as we may see from the graph (see Fig. 8), at first these graphs have some intersection points (no one is dominant), but in general **ProbSAT has the same (at the beginning) or even bigger probability to get a successful solution than the GSAT.**

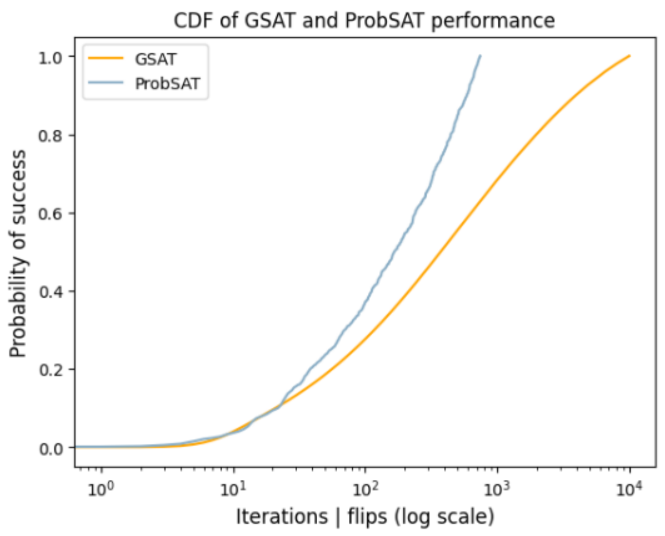
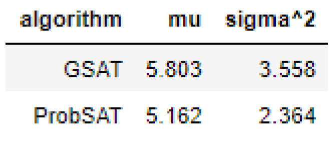


Fig. 8: Graph for a Cumulative Distribution Function

Moreover, I counter mu and sigma^2 parameters (Fig. 9) to see how my distribution varies from the normal one. For this I used the next formulas $\mu = \frac{\sum_{i=1}^{n} \log x_i}{n}$ and $\sigma = \sqrt{\frac{\sum_{i=1}^{n} (\log x_i - \mu)^2}{n-1}}$.



algorithm	mu	sigma^2
GSAT	5.803	3.558
ProbSAT	5.162	2.364

Fig. 9: Metrics of Normal Distribution

As we may see from Fig. 9 ProbSAT has lower mu and sigma^2 values (so most of the values are located on the left side from GSAT mu - the number of steps is smaller, and most of the values have closer value to the mu). **Therefore ProbSAT dominates (is better) GSAT** (see Fig. 10).

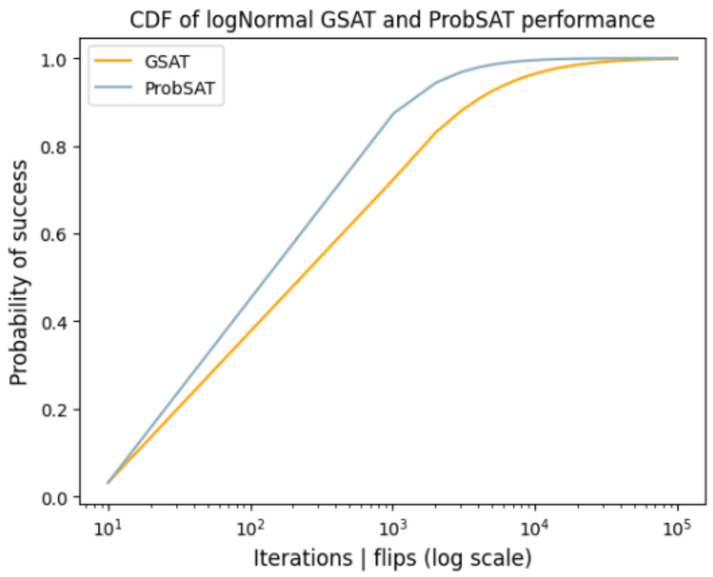


Fig. 10: Theoretical Graph for a Cumulative Distribution Function

DISCUSSION

Due to the selected metrics and previous conclusions we may clearly say that **ProbSAT used to have a better performance on the selected parameters (c_m, c_b, p and num_of_tries) solving 3-SAT hard instances than the GSAT (so, our hypothesis is correct).** As an input we used a lot of values, so the outcome decision is not influenced by random anomalies.

For the future examinations will be great to check other values of algorithm parameters and try to improve them to the optimal one.