On Probability Density Distribution of Randomized Algorithms Performance

Jan Schmidt, Rudolf Blažek, and Petr Fišer
Czech Technical University in Prague
email: schmidt@fit.cvut.cz, rblazek@fit.cvut.cz, fiserp@fit.cvut.cz

Abstract

EDA tools employ randomized algorithms for their favorable properties. Deterministic algorithms have been found sensitive to details in the input, and thus they also behave as the randomized ones. In both cases, performance analysis and comparison shall be made by histograms, or more precisely, by their probability density distribution. We present a modeling of ABC performance and, to gain understanding, a simple randomized algorithm solving the 3MAX-SAT problem. Also, a simulated annealing algorithm solving the same problem is studied. We claim that Gaussian Mixtures are suitable for such models and that truncated models must be considered.

1 Introduction

EDA tools must appear deterministic to the user. In practice, the design process must be repeatable: given the same input and the same constraints, the output shall be identical when run multiple times. Also, small changes in the input shall cause small changes in the output.

When randomized algorithms such as simulated annealing are used, the randomness is masked by pseudo-random generators with a fixed seed. Recently, however, multiple authors [1],[2],[3],[4], [5] reported, that the solution quality of almost all tools depends on semantically irrelevant changes in the input descriptions, such as the order of variables declaration. This can be understood as (undesirable) *bias*. Because the existence of the dependence is hidden from the users, the bias must be considered random, and in this sense the algorithm becomes (involuntarily) randomized.

A fair comparison of EDA algorithms thus becomes problematic [3],[4], as a single value of quality does not suffice, and requires to measure the *probability density distribution* of quality. Thus, an algorithm is characterized not only by quality (e.g. the mean value of the distribution), but also by robustness (e.g. the deviation of the distribution) [5].

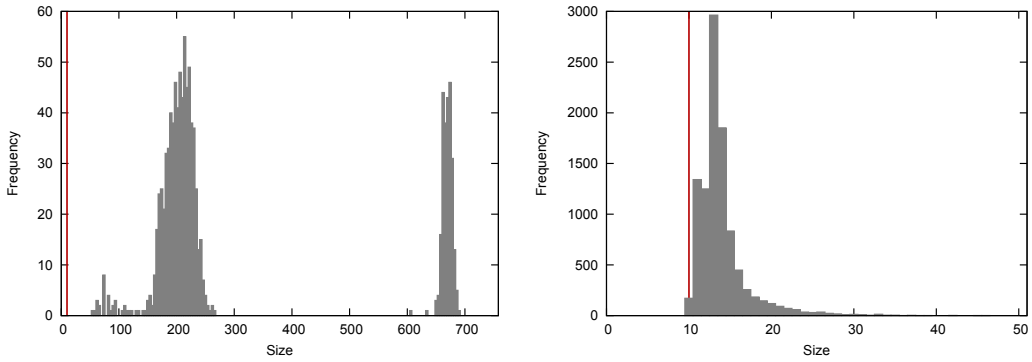
After a motivation statement in Section 2, the proposed stochastic models are given. Statistical analyses of different randomized algorithms are presented in sections 3–6. Section 7 concludes the paper.

2 Motivation

To get a more relevant picture of the tools' performance, we used probability density distribution to evaluate the proposed algorithms. Let us briefly introduce some of them to illustrate the problem.

Figure 1a shows the result of 1,000 independent runs of the academic EDA tool ABC [6] on the *cordic* benchmark from the IWLS'93 Benchmark Set [7], where inputs and outputs were randomly permuted in each run. ABC command sequence `strash; dch; if; mfs` for look-up table (LUT) mapping has been used, and the number of LUTs used as measure. For comparison, the known minimum-size implementation of the circuit is represented as a vertical line. When the same process is iterated 40 times, we get much improved results, as shown in Figure 1b.

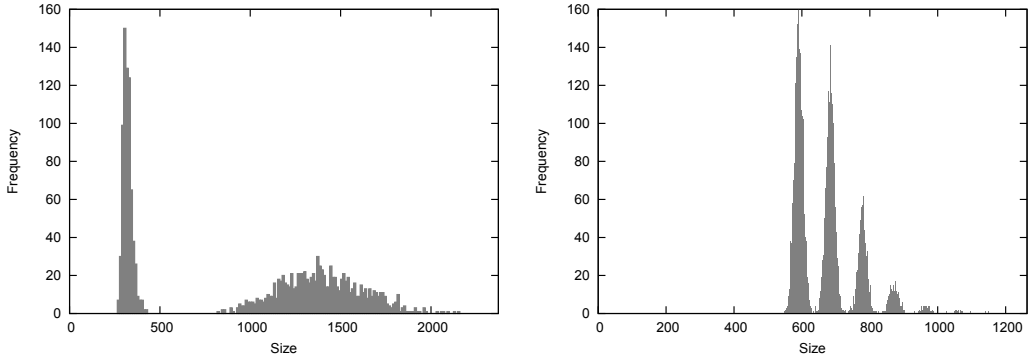
The next example is from a different area, namely test compression. The SAT-Compress algorithm [8] was repeatedly run 1,000-times on the ITC'99 *b04* [15] benchmark circuit, whereas



(a) 1 iteration (b) 40 iterations

Figure 1: Solution quality of ABC randomized by signals declaration, on the *cordic* circuit.

different initial test patterns were chosen randomly each time. Again, results of different sizes (measured as the number of the resulting compressed test bitstream bits) were observed.



(a) SAT Compress, randomized by initial test pattern, on the *b04* circuit (b) Randomized term expansion on a synthetic circuit

Figure 2: Solution quality of other processes

Yet another example in Figure 2b comes from two-level (SOP) logic minimization. Cubes of a randomly generated incompletely specified Boolean function having 100 inputs, 20 outputs, and 100 specified terms have been randomly expanded, and then the function cover has been solved. The result size has been measured as the number of literals.

The problem we present here is, what information can the probability density distribution tell us? Can we *qualitatively* judge the algorithm using such measurements?

In this paper, we focus on size. Certainly, this is not the only and perhaps not the most important criterion. As the above examples show, very different processes exhibit similar behavior, and further experiments show that this is the case of other quality criteria too. Our aim is to understand the behavior of randomized algorithms, and for this the choice of the criterion does not matter much.

In the rest of the paper, we show that a stochastic model can be constructed reliably in most cases. Then we discuss possible ways of robustness measures. As the algorithms studied are complex, we present also probability density distributions of very simple randomized algorithms.

3 Stochastic Models

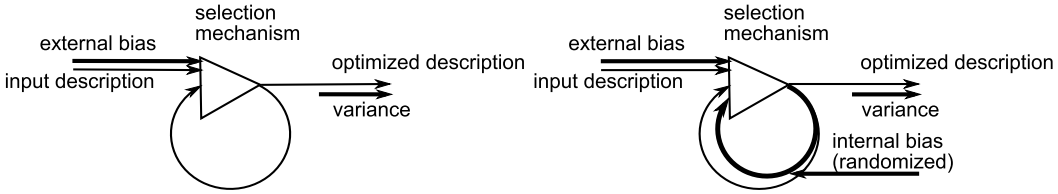
To measure the performance and to understand the operation of an algorithm, we need to characterize the detailed data (as in Figures 1 and 2) by a small number of parameters. An approximation (model) by continuous (parametric) distribution can also serve this purpose. The model, however, can be considered valid for restricted arguments values only. First, the arguments are integer (circuit size is discrete). Secondly, the arguments shall be restricted to

the interval of actually observed values, or values that are known as possible. Such a restriction is known as *truncation* [9]. We have chosen the sum of weighted normal distributions, known as Gaussian Mixtures (GM) [10] as our model.

For example, the distribution in Figure 2b has a seven-component model, as verified by Kolmogorov-Smirnov test [12].

4 ABC performance

ABC [6], [13] is an academic logic synthesis and optimization tool. Because research studies (and design practice as well) require reproducible results, ABC behaves deterministically. Recently, however, it has been discovered that many logic synthesis tools are sensitive to seemingly innocent and certainly semantic-preserving changes [1], [2], [3], [4], [5]. The performance of the tool shall be evaluated as if the tool has two inputs – one comprehensible to the designer and one hidden (Figure 3a). The hidden input can be (mis-)used to *randomize* the originally determinis-



(a) with input influence only (b) with randomized iterations

Figure 3: An iterative design tool sensitive to hidden details

tic tool. This is essential for a fair evaluation of the tool as well as for estimations of worst case behavior. For tools working iteratively, such a randomness injection can be performed during iteration (Figure 3b), similarly to truly randomized algorithms. For details and more thorough analysis of ABC commands robustness see [2], [16].

ABC has a modular architecture. Each module (called *command*) implements a particular design task. A sequence of commands constitutes a synthesis *script*. The commands used in our measurement performed combinational optimization and mapping to standard cells and to FPGA look-up tables (LUTs). We used two scripts. *Gate mapping* performs standard cells mapping (`strash; dch; map` commands sequence), while *LUT mapping* performs 4-LUT mapping (`strash; dch; if; mfs`).

264 benchmarks from the IWLS [7] and LGSynth [14] sets were processed 1,000 times for each circuit, inputs and outputs have been randomly permuted in the source file, and the resulting design size (gates, LUTs) has been measured.

As recommended by the ABC authors [6], [17], the result quality can be sometimes significantly improved by running the synthesis in ABC iteratively, as *resynthesis*. Here the synthesis scripts (including technology independent optimization and technology mapping) are run repeatedly. Therefore, we iterated the two above described scripts 40 times in otherwise identical conditions. We did not inject randomness into individual iterations. For the differences it may cause, see [5] and [16].

We modeled the data from the ABC experiments. To find the GM parameters, we employed the `em4gmm` implementation [18] of the EM algorithm [10], [11]. The types of results are summarized in Table 1. We tested the goodness of fit by comparing the values of empirical and statistical cumulative distribution function (CDF). The cases where those functions were completely dissimilar, were marked as cases of "bad fit".

For some benchmarks, the result was not affected by the randomized bias at all, hence such cases are labeled *invariant* in the table. In other cases, the randomized algorithm produced only a small number of different sizes, sometimes distant, and GM modeling failed. Such cases are denoted *discrete* in the table. In the rest of cases, we were able to estimate the goodness of fit. This leads us to the general conclusion that about half of the cases *are* Gaussian mixtures. The GM models enable us to discriminate cases such as in Figure 1a by the number of local maxima besides the number of components. A summary is in Table 2, with invariant cases omitted. We can see that instances giving more than two local maxima are sparse, yet they do

Table 1: GM fit of experimental data, numbers and percents of benchmarks by result types

result type	LUT mapping		Gate mapping		LUT mapping, 40 it.	
invariant	43	16%	28	10%	59	28%
discrete	41	15%	30	11%	37	17%
bad fit	33	13%	26	10%	10	4%
good fit	147	56%	179	68%	106	50%

Table 2: Number of components and local maxima in the ABC experiments

Local max.	Components	LUT mapping		Gate mapping	
1	1	203	35	217	53
	2		150		148
	3		18		16
2	2	21	10	11	10
	> 2		7		1
> 2	> 2		4		6

occur. Surprisingly, the most frequent case is *two* components forming a *single* local maximum. A typical case is shown in Figure 4. The crucial question here is the qualitative interpretation

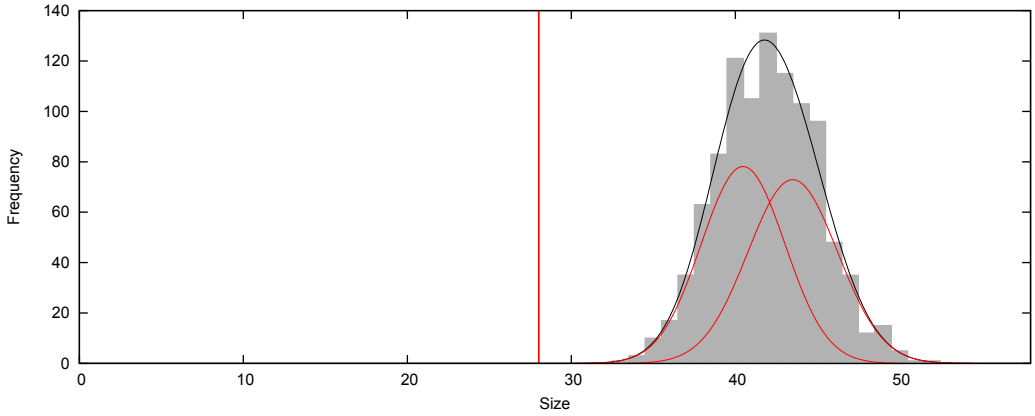


Figure 4: LUT mapping on *br2*, with GM model, its components and known minimum

of the results. The data may have a non-Gaussian distribution, which the algorithm tries to approximate by Gaussian mixtures. Alternatively, the two components may correspond to two solution clusters.

As we were unable to find any result in this direction, we performed experiments with processes much simpler than the complex, layered heuristics of ABC.

5 Randomly Valued 3MAX-SAT Formulas

The Maximum 3-Satisfiablity (3MAX-SAT) optimization problem belongs to the family of Satisfiability problems. Given a Boolean formula in conjunctive normal form and with exactly 3 literals in each clause, the goal is to find a valuation of the formula's variables which satisfies the largest number of clauses [19].

The simplest randomized algorithm for this problem is to set the values randomly, with equal probability of 0 and 1. The expected number of satisfied clauses is 7/8 of maximum, for satisfiable formulas [20]. For this problem, the maximum (optimum) value is known and equal to the number of clauses.

We used 3SAT instances form the SATLIB example collection [21], namely 1,000 satisfiable instances with 50 variables and 218 clauses. Formulas with this ratio of clauses to variables are known to have only a small number of solutions (if satisfiable at all), and to be the hardest ones to solve [22]. For each instance, we measured the number of satisfied clauses for 1,000 random

valuations, and estimated the distribution as in the previous experiments. The character of the instances implies, that the probability of obtaining the maximum value is only 2^{-50}.

All instances permitted a good fit. The models had a single local maximum, which agrees with the predicted mean value. The numbers of components are in Table 3. The situation observed

Table 3: Number of components in randomly valuated 3SAT formulas

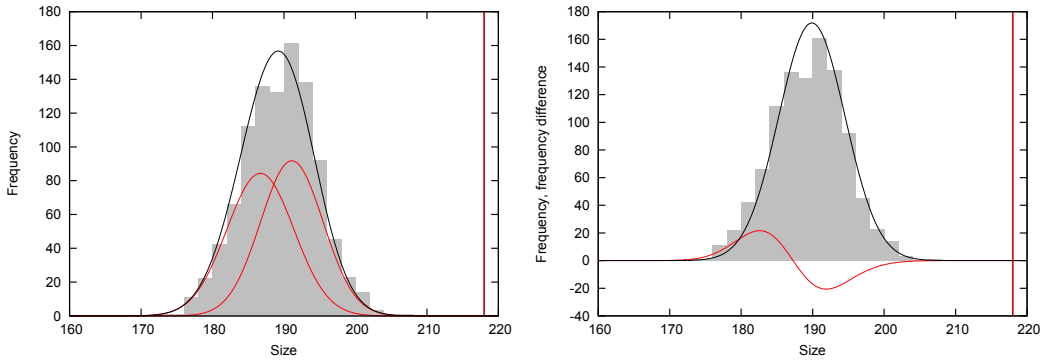
components	count
1	187
2	732
3	81

in the ABC experiments repeats here. The case with 1 local maximum and 2 components is the most frequent one. We forced the EM algorithm to produce a single component model for each case, but the results had worse quality of fit. Figure 5 illustrates the reason: the distribution is asymmetric.

This intuition is supported by Log Likelihood Ratio (LLR) statistics. We used the single-component model as the null hypothesis, and the two-component model as the alternative hypothesis. The log likelihood difference gave the value of 20.6. As the critical p value for significance level of 0.05 is in the order of 10^{-4}, we reject the null hypothesis in favor of the alternative hypothesis.

Other, asymmetric distributions (e.g. Weibull) did not yield any good fit, with negative log likelihood difference against the two-component model.

Backtrack search has been reported to have a (right) heavy-tailed distribution [23], that is, that the number of poor-quality solutions does not fade exponentially with the quality. No observed distribution in our experiments had heavy tails (left or right).



(a) A two-component model

(b) A single-component model and the difference from the 2-component model

Figure 5: Random valuation on the *uf50-0828* instance, and the achievable maximum

Pure normal distribution cannot be expected here, as the process producing it is not fully additive; the contribution of one variable to satisfied clauses depends on the value of other variables. The strength of variable linkage, however, can be regulated by omitting clauses.

Table 4: GM models of randomly valuated and shortened *uf50-0828* instance

Clauses	Component 1			Component 2		
	weight	rel. mean	rel. std. dev	weight	rel. mean	rel. std. dev
30	0.515	0.893	0.0445	0.485	0.835	0.0571
50	1	0.864	0.0498			
100	0.512	0.887	0.0246	0.488	0.851	0.0273
150	0.502	0.883	0.0220	0.498	0.857	0.0227
218	0.498	0.884	0.0182	0.502	0.858	0.0183

We shortened the clauses to 150, 100, 50 and 30 clauses. The resulting models are in Table 4.

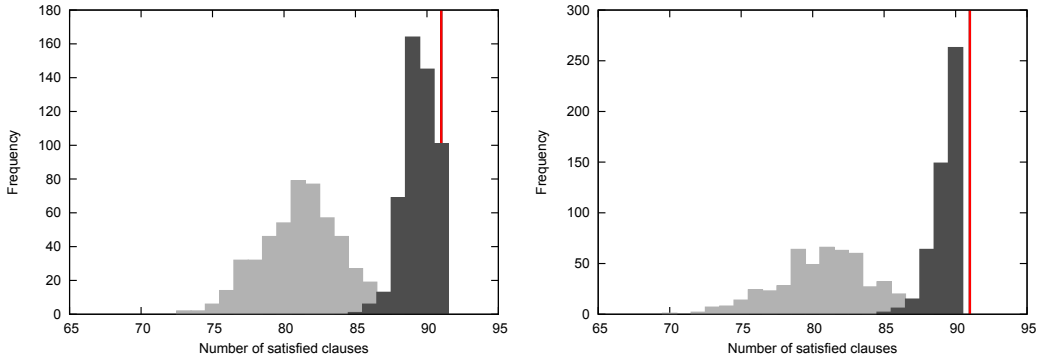
The mean values and standard deviations are taken relatively to the number of clauses. The general character of the models is fairly stable, with the exception of 50 clauses, where the EM algorithm converged to a different solution. For the rest of clause counts, there are two components with almost equal weights. With decreasing number of clauses, there appears small but systematic asymmetry. Standard deviations increase with decreasing number of clauses, while mean values remain constant. Both these observations may indicate that, in reality, truncation takes place, as shown in the next section.

6 Simulated Annealing

In the previous experiments, no improvement has been attempted and the results were distant from the optimum. To lean the distributions closer to the optimum, we have measured the standard Simulated Annealing (SA) algorithm [24] applied to the same 3MAX-SAT problem. SA is a randomized algorithm, so that Figure 3b applies here.

We have conducted two experiments with 500 runs. In the first one, the initial solution was always "111...11" and the algorithm has been randomized internally by random seeding of the pseudo-random numbers generator. In the other set, the algorithm was randomized both internally and externally by its initial solution. 4×10^6 steps were performed in each run, with slow cooling. Satisfiable SATLIB formulas with 20 variables and 91 clauses were used.

From the modeling point of view, the middle phase, where most of the optimization happens, is the most interesting one. Figure 6 shows the histograms after 10^6 and 2.4×10^6 steps. In the first case, the distribution is truncated at the expected maximum of 91 clauses. In the other case, however, the truncation happens *earlier*, at the value of 90. The same behavior also occurs with larger clauses, where the truncation point is 214 as compared to the number of clauses, which is 218. Such a behavior illustrates the fact that truncation plays an important role. Although



(a) "Magic" initial solution ("111...11") (b) Random initial solutions

Figure 6: Simulated annealing on the *uf20-09* instance, after 10^6 (gray) and 2.4×10^6 steps

the *entire* distribution is truncated in this case, examples can be constructed where *components* are truncated.

7 Conclusions

Most of EDA tools are affected by semantically irrelevant changes in the input description, and can be randomized that way. For a particular circuit, the tool may be immune to the input changes, or may produce several distinct solutions. In the majority of cases, the distribution of result quality can be modeled by Gaussian mixtures with acceptable goodness of fit. The prevalent type of model has two components, a single local maximum and corresponds to a slightly asymmetric distribution. For some tools, the distribution may have multiple local maxima for less than 5% of benchmark circuits.

The qualitative interpretation of the model components remains uncertain. While distant local maxima seem to correspond to different solution clusters, the frequent case of two close components is probably related to the asymmetry of the distribution.

Because the random values studied are measures of solution quality coming from optimization problems, the models have natural bounds. Some algorithms exhibit a behavior suggesting even tighter bounds within the algorithm itself. Whether those bounds can be incorporated in the model simply by stating its valid range, or whether they affect the character of the model, remains again an unsolved problem.

References

[1] P. Fišer and J. Schmidt: "How Much Randomness Makes a Tool Randomized?," in Proc. of the 20th International Workshop on Logic and Synthesis (IWLS), San Diego, California, USA, June 3–5, 2011, pp. 136–143.

[2] P. Fišer, J. Schmidt, and J. Balcárek, "Sources of Bias in EDA Tools and Its Influence," in Proc. of 17th IEEE Symposium on Design and Diagnostics of Electronic Systems (DDECS), Warsaw (Poland), April 23–25, 2014.

[3] A. Puggelli, T. Welp, A. Kuehlmann, and A. Sangiovanni-Vincentelli, "Are Logic Synthesis Tools Robust?" in Proc. of the 48th ACM/EDAC/IEEE Design Automation Conference (DAC), 5-9 June 2011, pp. 633–638.

[4] W. Shum and J. H. Anderson, "Analyzing and predicting the impact of CAD algorithm noise on FPGA speed performance and power," in Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays (FPGA '12), pp. 107–110.

[5] P. Fišer, J. Schmidt, and J. Balcárek, "On Robustness of EDA Tools," in Proceedings 17th Euromicro Conference on Digital Systems Design (DSD), Verona (Italy), August 27–29, 2014.

[6] Berkeley Logic Synthesis and Verification Group, "ABC: A System for Sequential Synthesis and Verification". [Online]. Available: http://www.eecs.berkeley.edu/ alanmi/abc/

[7] "IWLS'93 Benchmark Set: Version 4.0", distributed as part of the IWLS'93 benchmark distribution.

[8] J. Balcárek, P. Fišer, and J. Schmidt, "Techniques for SAT-based Constrained Test Pattern Generation," in Microprocessors and Microsystems, Vol. 37, Issue 2, March 2013, Elsevier, pp. 185–195.

[9] A.C. Cohen: "Truncated and Censored Samples: Theory and Applications." Marcel Dekker, New York, 1991.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: "Gaussian Mixture Models and k-Means Clustering". In Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York, Cambridge University Press, 2007. ISBN 978-0-521-88068-8.

[11] D. Titterington, A. Smith, U. Makov, "Statistical Analysis of Finite Mixture Distributions". Wiley. 1985, pp. 243.

[12] M. A. Stephens: "EDF Statistics for Goodness of Fit and Some Comparisons." Journal of the American Statistical Association 69 (347), 1974, pp. 730–737.

[13] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial Strength Verification Tool," in Proc. of the 22nd International Conference on Computer Aided Verification, Edinburgh, UK, July 15–19, 2010, LNCS 6174 6174, Springer 2010, pp. 24–40.

[14] K. McElvain, "LGSynth93 Benchmark Set: Version 4.0", Mentor Graphics, May 1993

[15] F. Corno, M.S. Reorda, and G. Squillero, "RT-level ITC99 benchmarks and first ATPG results," in Proc. of the IEEE Design and Test of Computers (2000), pp. 44–53.

[16] P. Fišer and J. Schmidt: "On Using Permutation of Variables to Improve the Iterative Power of Resynthesis," in Proc. of 10th Int. Workshop on Boolean Problems (IWSBP), Freiberg (Germany), September 19–21, 2012, pp. 107–114.

[17] A. Mishchenko, R. Brayton, J.-H. R. Jiang, and S. Jang, "Scalable don't-care-based logic optimization and resynthesis", ACM Trans. Reconfigurable Technology and Systems (TRETS), Vol. 4(4), April 2011, Article 34.

[18] Juan Daniel Valor Mir: "Fast clustering Expectation Maximization algorithm for Gaussian Mixture Models." Accessed at https://github.com/juandavm/em4gmm

[19] G. Ausiello et al.: "Complexity and Approximation". Berlin etc., Springer, 1994, pp. 455–456.

[20] ibid, Theorem 2.19, p. 75

[21] H. H. Hoos and T. Stützle: SATLIB: An Online Resource for Research on SAT. In: I. P.Gent, H. v. Maaren, T. Walsh, editors, SAT 2000, pp. 283–292, IOS Press, 2000. SATLIB is available online at `www.satlib.org`.

[22] B. Selman, D. Mitchell, H. Levesque: "Generating Hard Satisfiability Problems". In Artificial Intelligence, 1996, vol. 81, pp. 17–29.

[23] H. Chen, C. Gomes, B.Selman: "Formal Models of Heavy-Tailed Behavior in Combinatorial Search". In AAAI Technical Report FS-01-04, AAAI, 2001.

[24] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi: "Optimization by Simulated Annealing". Science 220 (4598), 1983, pp. 671-680.