Typical-case complexity and the SAT competitions*

Zoltán Ádám Mann
Department of Computer Science and Information Theory
Budapest University of Technology and Economics

Abstract

The aim of this paper is to gather insight into typical-case complexity of the Boolean Satisfiability (SAT) problem by mining the data from the SAT competitions. Specifically, the statistical properties of the SAT benchmarks and their impact on complexity are investigated, as well as connections between different metrics of complexity. While some of the investigated properties and relationships are "folklore" in the SAT community, this study aims at scientifically showing what is true from the folklore and what is not.

1 Introduction

Modern SAT solvers routinely solve problem instances with hundreds of thousands of variables and millions of clauses. This and similar claims can be found in several papers [29, 2, 12, 11]. Indeed, *some* problem instances of this size can be solved by modern SAT solvers, but by far not all of them; in fact, there are problem instances that are orders of magnitude smaller but still cannot be solved in practice. Being an NP-complete problem, the worst-case complexity of all known exact SAT solvers is super-polynomial, leading to a huge gap between the worst-case complexity and the complexity of typical instances. The main goals of typical-case complexity research are to understand the factors that impact the complexity of typical problem instances and to develop algorithms that are efficient on as many instances as possible [19].

Such results are usually achieved either through mathematical analysis of expected algorithm complexity on random instances or through comprehensive empirical measurements. In this paper, the focus is on the latter. However, since large-scale computational experiments are costly (see below), the aim is to reuse existing data from past SAT competitions, instead of running new measurements.

SAT competitions are held since 1992, with the aim of providing an objective evaluation of contemporary SAT solvers against each other [15]. The exact format of the competitions has varied over the years (for example, in some years, there was a SAT race instead of a SAT competition, putting more focus on industrial application problems), but the SAT competitions have consistently acted as a driving force in the development of ever more efficient SAT solvers.

This paper mostly relies on data from the last SAT competition, held in 2013 (SC 2013), as a satellite event of the SAT 2013 conference [7]. The benchmarks are grouped in three categories: application, hard combinatorial, and random. In all benchmark categories, there are both satisfiable and unsatisfiable instances. In the first two categories, there are 300 instances each, and 400 in the random category. In this paper, only complete, sequential SAT solvers are considered.

SC 2013 was a giant set of computational experiments: the 93 submitted solvers used altogether roughly 100,000 hours of CPU time, which would cost approximately 50,000 EUR on Amazon EC2, used about 6,000 kWh energy, generating roughly 3 tons of carbon-dioxide [8].

The aim of this paper is to investigate the following questions, based on the SC 2013 results:

*This paper was published in *Proceedings of the 5th Pragmatics of SAT Workshop (POS-14)*, EasyChair Proceedings in Computing, volume 27, pp. 72-87, 2014.

- What characterizes typical problem instances? Are there significant differences between instances of the three benchmark categories?
- What metrics can be used to characterize the complexity of a problem instance, and how do they relate to each other?
- How do size and other metrics of problem instances impact their hardness?

The presented data should provide valuable insight for algorithm designers, competition organizers, and other researchers working in the field of SAT. In particular, we use a substantial amount of data to prove or disprove by rigorous statistical analysis some of the common beliefs in the SAT "folklore."

2 Previous work

SAT competitions are quite well documented, see e.g. [7, 4] for the last two and [15] for a recent overview of the series. The website http://www.satcompetition.org has (links to) all the details about the competitions, including benchmarks, solvers, and results.

There have also been some scientific works inspired by the competitions. In particular, some researchers have questioned the adequacy of the used solver ranking schemes [22, 25] and benchmark selection process [14], and came up with more sophisticated solutions. Other work dealt with the detailed evaluation of solvers' performance [27, 28], and the machinery needed to smoothly implement the competition [24].

In contrast to most previous work, which focussed on solver performance, this paper takes a *problem instance* centered approach, in which solvers are just instruments to obtain information about the instances.

3 Preliminaries

In the *SAT problem*, we are given n Boolean variables and m clauses, where each clause is the disjunction of a set of literals, and a literal is a variable or its negation. The aim is to decide whether there is an interpretation of the variables that satisfies all clauses. The ratio m/n will be denoted by α.

In the subsequent analysis, the *correlation* of two datasets will often be investigated. For this purpose, the most widely-used metric is the Pearson product-moment correlation coefficient, commonly called simply "correlation coefficient." For a pair of data series $X = (X_1, \ldots, X_n)$, $Y = (Y_1, \ldots, Y_n)$, the correlation coefficient is given by the following formula:

$$r(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}},$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$. The correlation coefficient is always between -1 and 1; $r = 0$ means no correlation, $r = 1$ means a perfect positive correlation, whereas $r = -1$ means perfect negative correlation.

A shortcoming of the Pearson correlation coefficient is that it measures the *linear* correlation between the two datasets, and is less sensitive to non-linear dependence relationships. Therefore, in this paper, the *Spearman rank correlation coefficient* (ϱ) will be used instead. Also ϱ is between -1 and 1, with similar interpretation as r. A high $|\varrho|$ value means that the relationship between the two datasets can be well described by a *monotonic* function (which need not be linear). $\varrho > 0$ means a monotonically increasing, $\varrho < 0$ a monotonically decreasing trend between the two datasets. As a rule of thumb, the correlation is considered *strong* if $|\varrho| \geq 0.6$, *moderate* if $0.6 > |\varrho| \geq 0.4$ and *weak* otherwise.

To calculate $\varrho(X,Y)$, one has to first create the corresponding *ranked variables* X' and Y': e.g., X'_i is the rank of X_i in the sorted (ascending) order of the X dataset. If multiple values are equal, they receive the ranking number which is the average of their positions in the sorted order. Then, the Spearman rank correlation coefficient is calculated as the Pearson coefficient of the ranked variables, i.e., $\varrho(X,Y) = r(X',Y')$.

Whether a correlation can be considered *significant*, depends not only on the value of ϱ, but also on the sample size from which ϱ was calculated. It is common to characterize the significance using the *p-value*, which gives the probability that the given $|\varrho|$ value or higher is obtained by chance, i.e. if there were no correlation. A p-value below 1 percent is usually considered sufficient for significance. For the Spearman correlation coefficient, the p-value can be approximated by means of a t-test with $t = \varrho\sqrt{(n-2)/(1-\varrho^2)}$ [16]. Specifically for a sample of size 300 (the size of the application and hard combinatorial benchmark categories), the correlation is significant if $|\varrho| \geq 0.14$; for a sample size of 400 (the size of the random benchmark category), it is significant if $|\varrho| \geq 0.12$.

4 Characteristics of problem instances

First, let us look at some basic statistical properties of the benchmarks. Several researchers pointed out the importance of short clauses and that binary and ternary clauses may need special treatment, partly because they form the majority of clauses in real-world problem instances [9, 3, 18]. To verify this, Fig. 1 shows a histogram of the clause length distribution of the three benchmark categories. Moreover, Table 1 shows the average clause length and the tail probabilities of the clause distributions for the three benchmark categories.

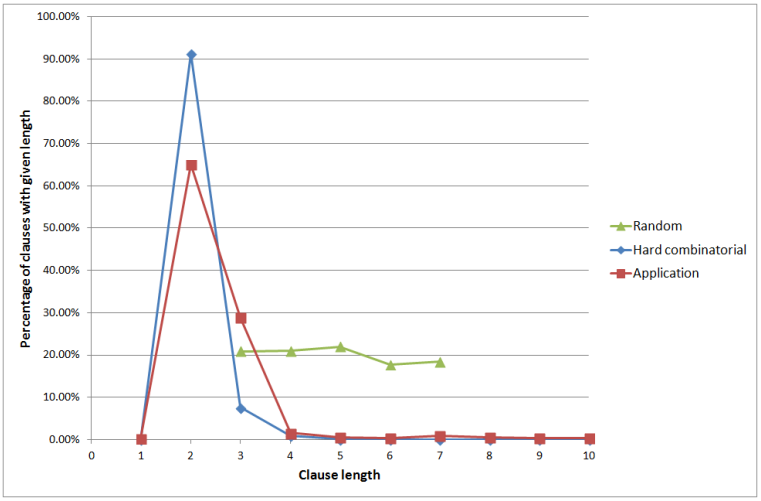

Figure 1: Clause length distribution of the three benchmark categories of SC 2013

Table 1: Average and tails of the clause length distribution

	Average clause length	Percentage of clauses with length=1	Percentage of clauses with length>10
Application	2.78	0.30%	1.82%
Hard combinatorial	2.11	0.01%	0.04%
Random	4.92	0.00%	0.00%

The characteristics of the random benchmarks are quite different from the other two categories. This is no surprise, since the random benchmarks were specifically generated with clause lengths of $3, \ldots, 7$ [6]. However, it also means that the random benchmarks do not model real-world instances well, so that insight from the analysis of uniform random k-SAT instances does not automatically apply to real-world instances as well. For the application and the hard combinatorial categories, the distributions are similar, both with a clear peak at 2. However, the concentration at 2 is much stronger for hard combinatorial instances; for application instances, also the length 3 occurs often. It is also worth mentioning that there is a non-negligible number of unit clauses in the benchmarks: even the 0.01% in the case of hard combinatorial instances means that, on average, such an instance contains roughly 103 unit clauses.

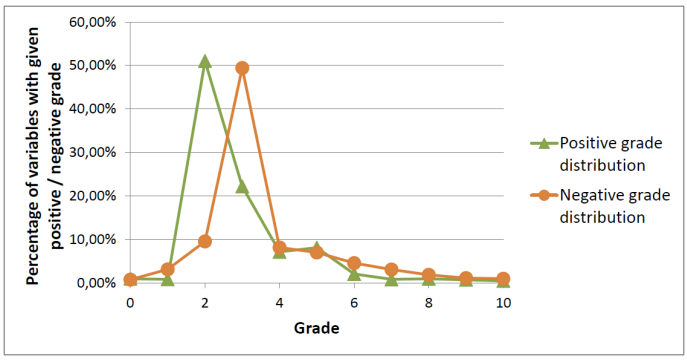

Figure 2: Variable grade distribution of the Application benchmark category

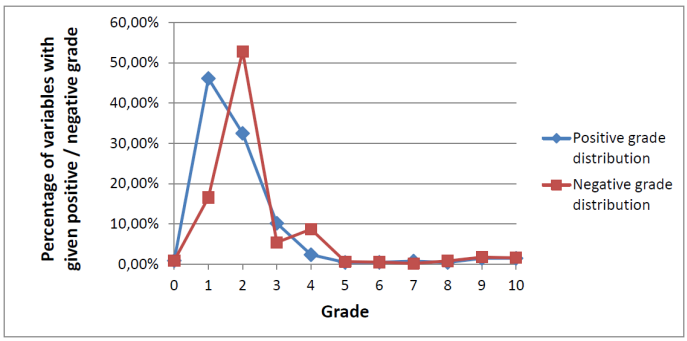

Figure 3: Variable grade distribution of the Hard combinatorial benchmark category

For a variable x, the *positive grade* of x is the number of clauses containing the literal x; its *negative grade* is the number of clauses containing the literal $\bar{x}$. As shown in Figures 2-4, also the grade distributions concentrate in the low part of the possible values, especially for application instances, where the positive grade peaks at 2 and the negative grade at 3, as well as the hard combinatorial instances, where the positive grade peaks at 1 and the negative grade at 2, all of them with a frequency of roughly 50%. Interestingly, in both cases, negative grades tend to be higher than positive grades. This seemingly counter-intuitive finding may be explained by some common idioms in SAT encoding: for example, the fact that exactly one of k variables should be true is usually expressed by a subformula in which each variable has positive grade 1 and negative grade $k-1$. Another example is the Tseitin encoding of an AND gate: the variable introduced for the AND gate has positive grade 1 and negative grade 2.

The behaviour of random instances is again quite different from the other two categories. In this case, the positive and negative distributions are almost identical, as could be expected. The

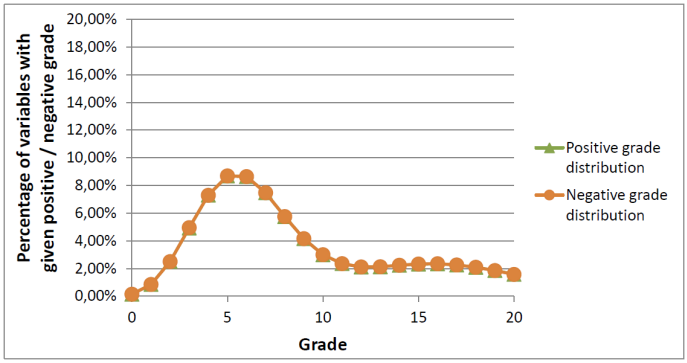

Figure 4: Variable grade distribution of the Random benchmark category

peak at the grade 5 is rather weak: less than 9% of variables have this grade. The distribution is clearly less concentrated than in the other two cases. The form of the curve with its second local maximum at 16 is a bit surprising, but can be explained as the superposition of multiple binomial distributions with different parameters.

Table 2: Average and tails of the variables' grade distribution

	Average grade	Percentage of variables with grade=0	Percentage of variables with grade>20
Application			
positive grade	6.54	0.96%	2.31%
negative grade	9.94	0.74%	5.04%
Hard combinatorial			
positive grade	3.28	0.88%	2.18%
negative grade	32.86	0.88%	9.11%
Random			
positive grade	25.07	0.14%	25.28%
negative grade	25.07	0.15%	25.27%

Similar conclusions can be drawn from Table 2. Even for the application and hard combinatorial benchmarks, which exhibit a high peak for values $1 \ldots 3$, the average is much higher, especially for negative grades. This also shows that the variable grade distribution is significantly less concentrated than the clause length distribution. Moreover, the grade 0 has a non-negligible frequency in all cases, justifying the usefulness of the pure literal heuristic [12].

5 Metrics of complexity

One of the fundamental questions of typical-case complexity is how to measure the complexity of a problem instance. What is easy to measure is the complexity of a specific algorithm on a given problem instance, but the results obtained this way are algorithm-specific. The data from the SAT competitions offer a great opportunity to decrease the dependence of the results on the peculiarities of a specific algorithm by considering a bigger set of algorithms[1]. The main aim of

[1] A limitation of this approach is that many similar solvers participate in the SAT competition. It is an interesting direction for future research to quantify and compensate the effect of solver homogeneity on the complexity metrics presented here.

the SAT competitions is to use the benchmark instances as tools to investigate the complexity of the algorithms in relation to each other, but a dual look at the competitions means using the algorithms as tools to obtain insight into the complexity of the benchmark instances.

Suppose that n algorithms are run on a given benchmark instance, with a timeout of T. The result is the set of runtimes $0 \leq t_1, t_2, \ldots, t_n \leq T$ (for a timeout, $t_i = T$). Based on these data, the following metrics can be defined to characterize the hardness of the given instance:

- Average of the runtimes (AVG)
- Median of the runtimes (MED)
- Number of timeouts (#TOUT)
- Average runtime of the runs that did not time out (AVG-NOTOUT)

Although AVG is probably the most intuitive metric, it is not always appropriate. If a distribution has no expected value, then the average of a sample can behave in an erratic way; the median should be preferred in such cases [13]. Also, AVG is distorted by the timeouts being counted as T, whereas this does not influence the median as long as at most half of the runs timed out. This is also the motivation for AVG-NOTOUT, because the average is only accurate for those runs that did not time out. #TOUT is meaningful because there are many timeouts in SC 2013, and hence the primary question of each run is whether it finishes before the timeout. (This is also the basis for the ranking of solvers [15].)

Table 3: Correlation between different metrics of complexity

	Application	Hard combinatorial	Random
AVG – MED	0.95	0.95	0.87
AVG – #TOUT	0.93	0.97	0.94
AVG – AVG-NOTOUT	0.85	0.64	0.92
MED – #TOUT	0.86	0.93	0.82
MED – AVG-NOTOUT	0.81	0.56	0.75
#TOUT – AVG-NOTOUT	0.68	0.52	0.94

As can be seen in Table 3, a strong positive correlation can be observed among the different complexity metrics on all three benchmark categories. This means that there are indeed instances that are hard, no matter how "hard" is defined, and similarly there are clearly easy instances. The correlation is strongest between AVG and #TOUT. This is no surprise, because the higher the value of #TOUT, the more runs have $t_i = T$, increasing also AVG. The weakest correlations can be observed between AVG-NOTOUT and the other metrics; in particular, for both application and hard combinatorial instances, the correlation between #TOUT and AVG-NOTOUT is lowest. This means that the fact that only a few solvers could solve a given instance makes it probable, but not at all definite, that it took a long time for those solvers to solve it. This also suggests that the pair (#TOUT, AVG-NOTOUT) can be used as an intuitive metric of complexity, in which both components provide useful information: an instance is really hard if few solvers could solve it, *and* it took them a long time.

6 Dichotomy of complexity

A challenge faced by algorithm engineers is to find good benchmarks for evaluation of algorithm performance. This can be quite difficult because of an often-perceived dichotomy of complexity: many benchmarks are either too easy or too hard, with only very few in between. This phenomenon is demonstrated in Figs. 5-7, showing the distribution of runtimes for the three benchmark categories. The horizontal axis corresponds to the runtimes, in seconds. In SC 2013, a timeout of

$T = 5000s$ was used; hence the last bar corresponds to the timeouts. The height of the bars represents the number of runs whose runtime falls into the given interval. (For the Random benchmark category, the last bar is actually much higher, but it is chopped off to enhance visibility of the rest of the diagram.) As can be seen, the pattern is the same for all three benchmark categories: the majority of runs is either quite short or experiences a timeout. Runs with a runtime between 1000 and 5000 seconds are relatively rare[2].

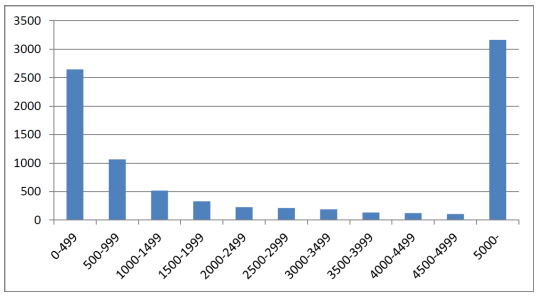

Figure 5: Histogram of runtimes in the Application benchmark category

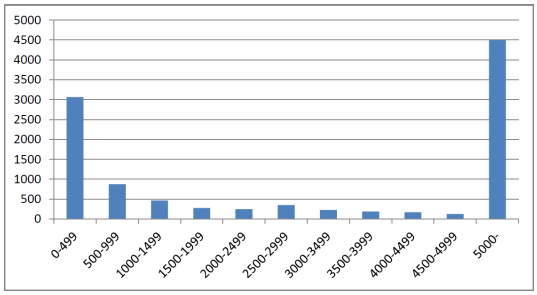

Figure 6: Histogram of runtimes in the Hard combinatorial benchmark category

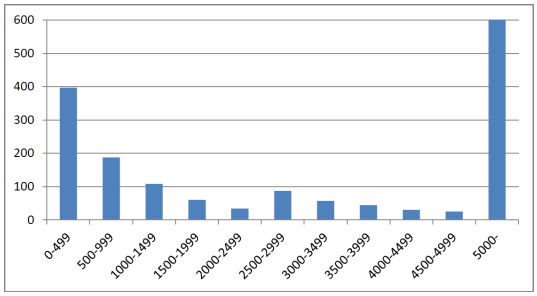

Figure 7: Histogram of runtimes in the Random benchmark category

This dichotomy is in line with the notion of heavy-tailed runtime distributions [13], but it makes benchmarking difficult. Also, it is worth mentioning that (i) SAT solvers routinely use frequent restarts to mitigate the effect of heavy tails [12] and (ii) the selection of the benchmarks for SC 2013 was carried out in a way so as to minimize the number of too easy and too hard

[2] Although less strikingly, but this phenomenon can also be recognized in the usual cactus plots of the competition, see e.g. `http://satcompetition.org/edacc/SATCompetition2013/experiment/19/cactus/`.

instances [5]. Even with these counter-measures, the complexity dichotomy remains very clear. Thus, the mitigation of the heavy-tailed runtime distribution of solvers, as well as the testing of solvers with a heavy-tailed runtime distribution, both remain important research topics.

7 The satisfiability threshold

It is well-known that at some critical value of α, denoted by $\alpha_0(k)$, the probability of solvability of random k-SAT formulae abruptly changes from almost 1 to almost 0; moreover, the average-case complexity of random k-SAT also peaks at $\alpha_0(k)$ [10, 17]. The question is to what extent these probabilistic properties apply also to specific real-world instances.

The random benchmarks used in SC 2013 come from random k-SAT distributions with $k = 3, \ldots, 7$ [6]. The threshold values for these distributions are quite precisely known [21]. For the other benchmarks, clause lengths are not uniform, so the average clause length can be used as a substitute for k. For a given value of k – which may be fractional in this case – the best known approximation of the satisfiability threshold is derived from the cavity method [21]. Based on the calculated estimation of the satisfiability threshold, denoted by α_0^*, the following two metrics can be derived to characterize, in a normalized way, the position of the given instance with respect to the threshold: (i) *constrainedness* is defined as $\mathrm{sgn}(\alpha - \alpha_0^*)$ and (ii) *relative density* is defined as α/α_0^*. The numerical property *satisfiability* is defined as 1 if the instance is satisfiable, -1 if it is unsatisfiable, and 0 if it is not known whether it is satisfiable.

Table 4: Correlation between metrics of density and satisfiability

	Application	Hard combinatorial	Random
constrainedness – satisfiability	-0.06	-0.03	0.01
relative density – satisfiability	0.12	-0.25	-0.01

As can be seen from Table 4, only very weak correlations can be observed between satisfiability and density. Actually, only the value of -0.25 is significant at the 99% confidence level, the others are not. This result may be surprising, as one would have expected a clear negative correlation (the higher the density, the less formulae will be satisfiable). For application and hard combinatorial benchmarks, there are two possible reasons for this. First, the phase transition phenomenon that is known for random k-SAT may not be applicable to structured instances with non-uniform clause length. Second, the used approximation for the threshold is only accurate for large values of k, whereas the average clause length of these benchmarks is very low (see Section 4). For random benchmarks, these issues are not applicable. Here, the reason seems to be rather that the random benchmarks were deliberately generated with α always being very close to the threshold [6]. Specifically, the relative density of the random benchmarks of SC 2013 varies between 0.9998 and 1.0003. This lack of a real variance practically prohibits any meaningful conclusions regarding the dependence on density.

Table 5: Correlation of relative distance from satisfiability threshold and hardness

Application	Hard combinatorial	Random
-0.16	0.12	-0.26

The other question is how the distance from the satisfiability threshold impacts hardness. For this, let the *relative distance from the satisfiability threshold* be defined as $|\alpha - \alpha_0^*|/\alpha_0^*$. As metric of hardness, the number of timeouts (#TOUT) is used. The resulting correlations are shown in Table 5. As can be seen, the correlations are rather weak, which is not surprising, given the

above issues with α_0^* and the benchmark selection process. Only the correlations of -0.16 and -0.26 are significant, and they are negative as expected: instances further away from the satisfiability threshold tend to be slightly easier.

8 The impact of problem instance size

Usually, bigger instances are expected to be harder. Size can be measured for example by the number of variables, number of clauses, or file size. Furthermore, the compressed file size has also been suggested [20]. The correlations among these metrics are shown in Table 6.

Table 6: Correlation between different metrics of size in the three benchmark categories

	Application	Hard combinatorial	Random
$n - m$	0.88	0.83	0.64
n – file size	0.85	0.83	0.55
n – compressed file size	0.90	0.85	0.60
m – file size	0.96	1.00	0.98
m – compressed file size	0.99	0.99	1.00
file size – compressed file size	0.97	0.99	1.00

As can be seen, all size metrics have a strong positive correlation. In particular, the correlations between m, file size, and compressed file size[3] are very strong, which is not surprising. More interestingly, also n has a strong correlation with these metrics, especially for application and hard combinatorial benchmarks: for example, formulae with more variables clearly tend to have more clauses. The correlation for random benchmarks is less strong.

Table 7: Correlation between metrics of size and complexity

	Application	Hard combinatorial	Random
Correlation with #TOUT			
n	-0.38	-0.16	0.38
m	-0.39	0.01	0.36
file size	-0.33	0.01	0.34
compressed file size	-0.39	-0.01	0.35
Correlation with AVG			
n	-0.42	-0.13	0.35
m	-0.43	0.03	0.32
file size	-0.37	0.03	0.31
compressed file size	-0.41	0.01	0.32

Investigating the impact of size on complexity, and using #TOUT and AVG as complexity metrics, the results are summarized in Table 7. As can be seen, there is no clear connection between size and complexity. For the hard combinatorial benchmarks, the correlations are very weak, most of them not even significant. For random benchmarks, a weak but significant positive correlation can be observed in all cases. Quite surprisingly, the application benchmarks give rise to weak or moderate but significant *negative* correlation in all cases.

Searching for an explanation of this counter-intuitive phenomenon, it is worth looking graphically at the connection between size and complexity. Figs. 8 and 9 show this dependence for the

[3]Compression was done with `gzip`.

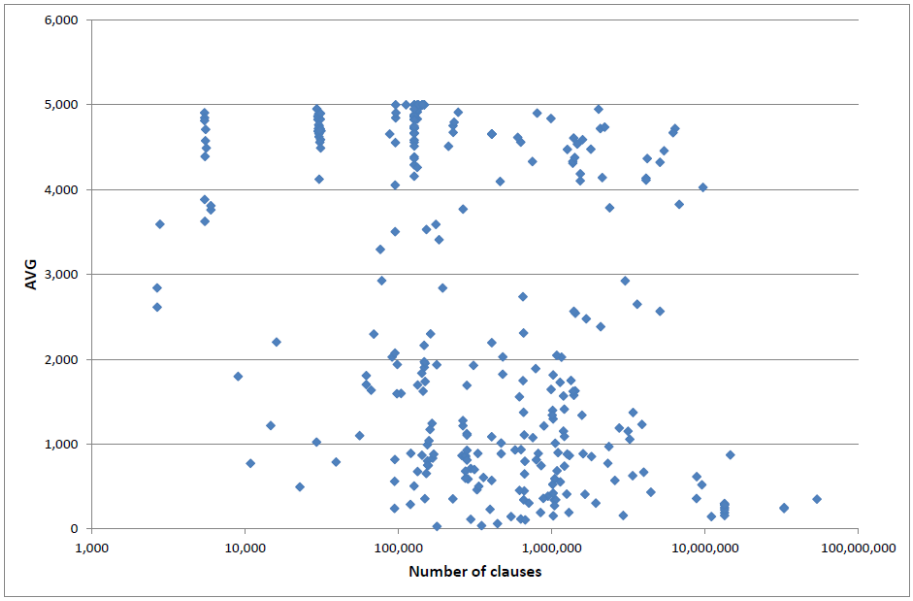

Figure 8: Dependence of average runtime on m for application benchmarks

application and random benchmark families, respectively. (The horizontal axis has logarithmic scale in both cases in order to improve visibility.) For the application benchmarks, the plot is rather chaotic, without any clear trend. However, there is a big cluster of very hard instances in the left half of the diagram, while in the right half, the majority of data points correspond to lower AVG values. This might be rooted in the way benchmarks are selected because huge instances must be relatively easy, otherwise no solver could solve them and they would have been discarded. This may explain the negative correlation. For the random benchmarks, the k-SAT instances with different k values can be quite clearly differentiated, and for each k, a steeply ascending trend can be observed, leading to the positive correlation.

Table 8: Extreme sizes (size is measured by the number of clauses)

	Application	Hard combinatorial	Random
Biggest solved SAT instance	53,616,734	1,827,900	52,057
Biggest solved UNSAT instance	9,676,386	15,834,160	6,145
Smallest unsolved instance	87,670	672	1,907

As a final experiment on the impact of size, Table 8 shows the size (in terms of m) of the biggest solved and smallest unsolved instances. It is again interesting to observe that size is hardly relevant: although an instance with over 50 million clauses could be solved, another one with some hundreds of clauses – i.e., 5 orders of magnitude smaller – could not.

The fact that problem instance size has limited impact on complexity is not new to the SAT community, because it has been known for a long time that large application instances are often easier than smaller random or hard combinatorial instances. However, referring back to the first sentences of this paper, the size of the biggest solved instances is still a key element in the communication about the efficiency of solvers, showing that this fallacy is still widely accepted. Moreover, our results show that even within the same benchmark category, size has no clear impact on complexity, and this may have been less widely known.

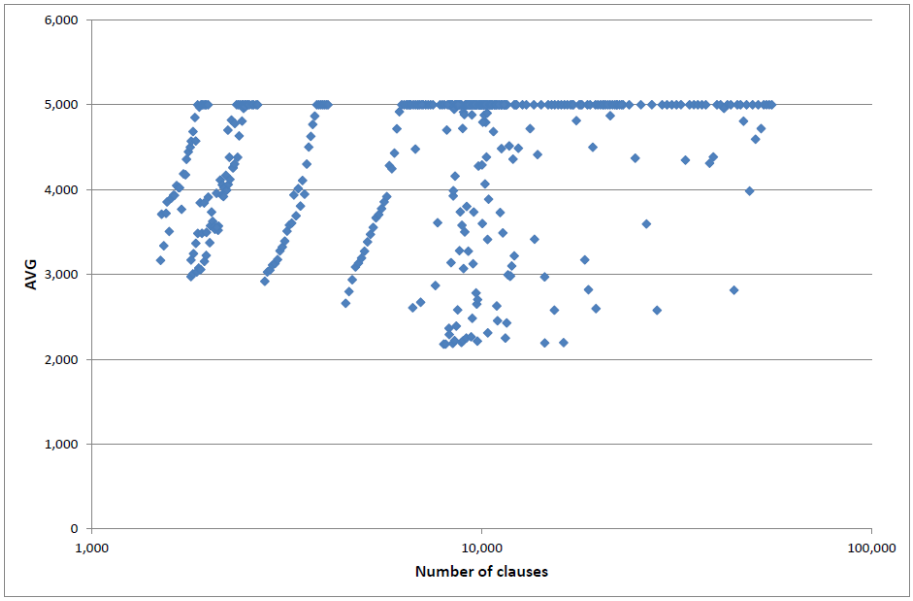

Figure 9: Dependence of average runtime on m for random benchmarks

9 The impact of some statistical features on complexity

Since size seems to be of little relevance to the complexity of problem instances, the next question is whether there are other, easily calculated features of the problem instances that have higher impact on complexity.

Table 9: Correlation between some statistical features of problem instances and their complexity (AVG)

	Application	Hard combinatorial	Random
α	0.18	0.25	-0.15
average clause length	0.42	0.14	-0.14
average variable grade	0.24	0.25	-0.15
average difference of positive and negative grade	0.30	-0.02	-0.02
compression ratio	-0.28	-0.23	0.27

As Table 9 shows, none of the investigated metrics exhibits a strong correlation with complexity; only the average clause length has moderate correlation with complexity on application benchmarks. The correlations are particularly weak for random benchmarks, and they have in most cases exactly the opposite sign as the corresponding correlations on application and hard combinatorial benchmarks. This shows once again the heterogeneity of the benchmarks and the large difference between the three benchmark categories.

It is worth noticing that such features are used (among others) in portfolio-based solvers like SATzilla to select, from a set of available solvers, the most appropriate solver for a given problem instance [26]. Although these features are useful for solver selection, apparently none of them is – at least alone – appropriate for predicting complexity. This is in line with previous observations from Nudelman et al., showing that from the features used in SATzilla, the ones that are useful for predicting complexity are only those that are either variations of the clauses-to-variables ratio

or information from partial runs of SAT solving algorithms [23].

10 Robustness of ranking

The usual way of ranking solvers based on the number of instances that they can solve within a given time frame may lead to some anomalies. The ranking methodology contains some arbitrary choices, in particular the set of benchmarks and the timeout value, that can significantly influence the results. As shown in Table 10, halving the timeout value T significantly changes the ranking results. This is in accordance with previous observations [25]. It can also be seen that halving the set of benchmarks, i.e. using only the first half of the set of benchmarks w.r.t. some random order, has even bigger impact on the ranking in the application and hard combinatorial tracks. Hence, small differences in the ranking will probably not have any statistical significance [22]. The displacements are smallest for random benchmarks; this is probably due to the very high percentage (82%) of the random benchmarks that were not solved by any solver, which makes the ranking in the random track relatively more resilient to such changes.

Table 10: Impact of halving the timeout value or the set of benchmarks: average and maximum displacement of solvers in the ranking

	Application	Hard combinatorial	Random
Halving T			
average displacement	11%	12%	8%
maximum displacement	45%	37%	36%
Halving the set of benchmarks			
average displacement	16%	13%	5%
maximum displacement	72%	54%	14%

Table 11: Impact of halving the timeout value or the set of solvers: average and maximum displacement of benchmarks in the ranking

	Application	Hard combinatorial	Random
Halving T			
average displacement	8%	5%	5%
maximum displacement	48%	28%	21%
Halving the set of solvers			
average displacement	6%	4%	10%
maximum displacement	33%	17%	23%

Analogously to the solver ranking, one can also create a ranking of benchmarks based on the number of solvers that could solve the instance in the given time frame (the #TOUT metric). The robustness of this ranking can be investigated by similar means as that of the solver ranking: what happens if T is halved or the set of solvers is halved? The results are shown in Table 11. As can be seen, the impact of changes is usually smaller for this ranking than for the solver ranking. This may be intuitively explained as follows: the drive to improve solvers has led to the good solvers having similar performance (just like the difference between the performance of top athletes is small), whereas there is no such drive for benchmarks.

11 Conclusions

The presented statistics on the basis of SC 2013 lead to the following conclusions:

- Random instances behave quite differently from the other two, more practical benchmark categories. Thus, the practical applicability of insight from random SAT (e.g., [1, 17, 21, 23]) is quite limited.
- Clause length distribution: for application benchmarks, 94% of clauses have length 2 or 3. For hard combinatorial benchmarks, this ratio is 99% (91% have length 2). For random benchmarks, the distribution is very different.
- Variable grade distribution: for application and hard combinatorial instances, the peak is at low values (1-3), but the distribution is less concentrated. Negated literals appear more frequently. There is a significant number of pure literals. Again, the distribution is quite different for random benchmarks.
- Metrics of complexity: the correlation between all investigated complexity metrics was quite strong.
- Dichotomy of complexity: for all three benchmark categories, the majority of instances is either very hard or relatively easy.
- Satisfiability threshold: the sudden phase transition and accompanying complexity peak that were previously observed for random k-SAT are not apparent in the given set of benchmarks.
- Problem instance size: different metrics of size correlate with each other; however, they have no clear impact on complexity. Also, while it is true that instances with tens of millions of clauses can be solved, there are also instances with only some hundreds of clauses that no solver could solve.
- Other statistical features: none of the investigated statistical problem instance features has a strong correlation with complexity.
- Robustness of ranking: changes in the arbitrary features of the experimental methodology may substantially distort the resulting ranking of solvers. The impact on the ranking of benchmarks is significantly smaller.

In some cases, these results reinforce our existing knowledge about the SAT problem, in some other cases, they enrich and extend it. This better understanding of problem characteristics will hopefully pave the way to improved solution and analysis techniques in the future.

Acknowledgements

This work was partially supported by the Hungarian Scientific Research Fund (Grant Nr. OTKA 108947) and the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

[1] Dimitris Achlioptas and Cristopher Moore. Random k-SAT: Two moments suffice to cross a sharp threshold. *SIAM Journal on Computing*, 36(3):740–762, 2006.

[2] Fadi A. Aloul, Karem A. Sakallah, and Igor L. Markov. Efficient symmetry breaking for Boolean satisfiability. *IEEE Transactions on Computers*, 55(5):549–558, 2006.

[3] Fahiem Bacchus. Enhancing Davis Putnam with extended binary clause reasoning. In *18th National Conference on Artificial Intelligence*, pages 613–619, 2002.

[4] Adrian Balint, Anton Belov, Daniel Diepold, Simon Gerber, Matti Järvisalo, and Carsten Sinz, editors. *Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions*, volume B-2012-2 of *Department of Computer Science Series of Publications B*. University of Helsinki, 2012.

[5] Adrian Balint, Anton Belov, Marijn J.H. Heule, and Matti Järvisalo. The application and the hard combinatorial benchmarks in SAT Competition 2013. In Adrian Balint, Anton Belov, Marijn J.H. Heule, and Matti Järvisalo, editors, *Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions*, volume B-2013-1 of *Department of Computer Science Series of Publications B*, pages 99–100. University of Helsinki, 2013.

[6] Adrian Balint, Anton Belov, Marijn J.H. Heule, and Matti Järvisalo. Generating the uniform random benchmarks for SAT Competition 2013. In Adrian Balint, Anton Belov, Marijn J.H. Heule, and Matti Järvisalo, editors, *Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions*, volume B-2013-1 of *Department of Computer Science Series of Publications B*, pages 97–98. University of Helsinki, 2013.

[7] Adrian Balint, Anton Belov, Marijn J.H. Heule, and Matti Järvisalo, editors. *Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions*, volume B-2013-1 of *Department of Computer Science Series of Publications B*. University of Helsinki, 2013.

[8] Adrian Balint, Anton Belov, Marijn J.H. Heule, and Matti Järvisalo. SAT competition 2013. Presentation at the SAT 2013 conference, 2013.

[9] Ronen I. Brafman. A simplifier for propositional formulas with many binary clauses. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(1):52–59, 2004.

[10] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *12th International Joint Conference on Artificial Intelligence (IJCAI '91)*, pages 331–337, 1991.

[11] Kenneth D. Forbus. AI and cognitive science: The past and next 30 years. *Topics in Cognitive Science*, 2(3):345–356, 2010.

[12] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*, pages 89–134. Elsevier, 2008.

[13] Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24(1-2):67–100, 2000.

[14] Holger H. Hoos, Benjamin Kaufmann, Torsten Schaub, and Marius Schneider. Robust benchmark set selection for Boolean constraint solvers. In *Proceedings of the 7th International Conference on Learning and Intelligent Optimization (LION 7)*, pages 138–152, 2013.

[15] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. *AI Magazine*, 33(1):89, 2012.

[16] Maurice G. Kendall and Alan Stuart. *The Advanced Theory of Statistics, Vol. 2: Inference and Relationship*. Griffin, 4th edition, 1979.

[17] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random Boolean expressions. *Science*, 264(5163):1297–1301, 1994.

[18] Tracy Larrabee. Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(1):4–15, 1992.

[19] Zoltán Ádám Mann. *Optimization in computer engineering – Theory and applications*. Scientific Research Publishing, 2011.

[20] Zoltán Ádám Mann and Pál András Papp. Predicting algorithmic complexity through structure analysis and compression. *Applied Soft Computing*, 13(8):3582–3596, 2013.

[21] Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k-SAT from the cavity method. *Random Structures & Algorithms*, 28(3):340–373, 2006.

[22] Mladen Nikolic. Statistical methodology for comparison of SAT solvers. In *Theory and Applications of Satisfiability Testing – SAT 2010*, pages 209–222, 2010.

[23] Eugene Nudelman, Kevin Leyton-Brown, Holger H. Hoos, Alex Devkar, and Yoav Shoham. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Principles and Practice of Constraint Programming (CP-04)*, pages 438–452, 2004.

[24] Olivier Roussel. Behind the scene of solvers competitions: the evaluation experience. In Vladimir Klebanov, Bernhard Beckert, Armin Biere, and Geoff Sutcliffe, editors, *Comparative Empirical Evaluation of Reasoning Systems (Compare 2012) – Proceedings of the International Workshop*, pages 66–77, 2012.

[25] Allen Van Gelder. Careful ranking of multiple solvers with timeouts and ties. In *Theory and Applications of Satisfiability Testing – SAT 2011*, pages 317–328. Springer, 2011.

[26] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008.

[27] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Detailed SATzilla results from the data analysis track of the 2011 SAT competition. In *14th International Conference on Theory and Applications of Satisfiability Testing*, 2011.

[28] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Evaluating component solver contributions to portfolio-based algorithm selectors. In *Theory and Applications of Satisfiability Testing – SAT 2012*, pages 228–241. Springer, 2012.

[29] Lintao Zhang and Sharad Malik. The quest for efficient Boolean satisfiability solvers. In Ed Brinksma and Kim Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 641–653. Springer Berlin / Heidelberg, 2002.