Random 3-SAT: The Plot Thickens *

Cristian Coarfa, Demetrios D. Demopoulus, Alfonso San Miguel Aguirre, Devika Subramanian, Moshe Y. Vardi
Department of Computer Science
Rice University
6100 S. Main St MS 132
Houston TX 77005-1892

June 3, 2001

Abstract.
This paper presents an experimental investigation of the following questions: how does the average-case complexity of random 3-SAT, understood as a function of the order (number of variables) for fixed density (ratio of number of clauses to order) instances, depend on the density? Is there a phase transition in which the complexity shifts from polynomial to exponential in the order? Is the transition dependent or independent of the solver? Our experiment design uses three complete SAT solvers embodying different algorithms: GRASP, CPLEX, and CUDD. We observe new phase transitions for all three solvers, where the median running time shifts from polynomial in the order to exponential. The location of the phase transition appears to be solver-dependent. While GRASP and CUDD shift from polynomial to exponential complexity at a density of about 3.8, CUDD exhibits this transition between densities of 0.1 and 0.5. This experimental result underscores the dependence between the solver and the complexity phase transition, and challenges the widely held belief that random 3-SAT exhibits a phase transition in computational complexity very close to the crossover point.

1. Introduction

The last decade has seen an intense focus on the complexity of randomly generated combinatorial problems. This interest was stimulated by the discovery of a fascinating connection between the *density* of combinatorial problems and their computational complexity, see [11, 39]. A problem that has received a lot of attention in this area is the *3-satisfiability problem* (3-SAT), a paradigmatic combinatorial problem that is important also for its own sake. An instance of 3-SAT consists of a conjunction of clauses, each one a disjunction of three literals. The goal is to find a truth assignment that satisfies all clauses. The

* Demopolous and Vardi are supported in part by NSF grants CCR-9700061 and IIS-9908435, and by a grant from the Intel Corporation. San Miguel Aguirre's current address is: Dept. of Computer Science, Instituto Tecnologico Autonomo de Mexico, Rio Hondo 1, 01000 Mexico City. He did this work while on sabbatical at Rice University, supported in part by CONACyT grant 145502. Subramanian is supported in part by NSF grant IRI-9796046.

density of a 3-SAT instance is the ratio of the number of clauses to the number of Boolean variables. We call the number of variables the *order* of the instance. Clearly, a low density suggests that the instance is under-constrained, and therefore is likely to be satisfiable, while a high density suggests that the instance is over-constrained and is unlikely to be satisfiable. Experimental research [15, 39] has shown that for ratio below (roughly) 4.26, the probability of satisfiability of a random 3-SAT instance goes to 1 as the order increases, while for ratio above 4.26 the probability goes to 0. At 4.26, the probability of satisfiability is 0.5. This satisfiability threshold density has been called the *crossover point*. Theoretically establishing the density at the crossover point is difficult, and is the subject of continuing research, cf. [20, 19, 1].

The experiments in [15, 39], which applied algorithms based on the so-called *Davis-Longemann-Loveland method* (abbr., DLL method) (a depth-first search with unit propagation [18]), also show that the density of a 3-SAT instance is intimately related to its computational complexity. Intuitively, under-constrained instances are easy to solve, as a satisfying assignment can be found fast, and over-constrained instances are also easy to solve, as all branches of the search terminate quickly. Indeed, the data displayed in [15, 39] demonstrate a peak in running time essentially at the crossover point. Using finite-size scaling techniques, [31] demonstrated a *phase transition* at the crossover point, viz., a marked qualitative change in the structural properties of the problem. This pattern of computational behavior with a peak at the crossover point is called the *easy-hard-easy* pattern in [38].

This picture, however, is quite simplistic for various reasons. First, the terms "easy" and "hard" do not carry any rigorous meaning. The computational complexity of a problem is typically studied on an infinite collection of instances, and is specified as a function of problem size or order. The easy-hard-easy pattern, however, is observed when the order is fixed while the density varies, but once the order is fixed, there are only finitely many possible instances. For that reason, theoretical analyses of the random 3-SAT problem focus on collections of fixed-density instances, rather than on collections of fixed-order instances, cf. [3]. Second, in the context of a concrete application, e.g., bounded model checking [4], planning [30], or scheduling [16], it is typically the order that tends to grow while the density stays fixed, for example, as we search for longer and longer counterexamples in bounded model checking. Thus, experimental results that vary density while fixing the order tell us little about the computational complexity of 3-SAT in such settings. Finally, it is not clear where the boundaries between the so-called "easy", "hard", and "easy" regions are. A widely held belief [39, 38] is that random 3-SAT problems are "hard" only for densities very close to the crossover point. Much work has therefore focused on explaining the "jump" in computational complexity around the crossover point using finite-size scaling [38] and backbones [36]. This alleged "jump", however, has not been documented experimentally. In fact, there is almost no experimental work that studies how the running time of a SAT solver varies as a function of the order for fixed-density instances (a few results of this nature, though not a systematic study, are reported in [14, 15, 38]). Further, the experiments reported in [39, 15] are based solely on DLL algorithms. While these are indeed the most popular algorithms for the satisfiability problem, one cannot jump to conclusions about the inherent and practical complexity of random 3-SAT based solely on experiments using these algorithms. We may observe different phenomena by experimenting with SAT solvers that embody different algorithms.

The goal of our experimental algorithmic research reported here is to determine how the average-case complexity of random 3-SAT, understood as a function of the order for fixed density instances, depends on the density. Is there a phase transition in which the complexity shifts from polynomial to exponential? Is such a transition dependent or independent of the solver?

To explore these questions, we set out to obtain a good coverage of an initial quadrangle of the two-dimensional $d \times n$ quadrant, where d is the density and n is the order. We explored the range $0 \leq d \leq 15$. We attempted to maximize the order of the sampled instances, given our resource constraints. We used three different SAT solvers, embodying different underlying algorithms. GRASP (`vinci.inesc.pt/~jpms/grasp/`) is based on the DLL method, but it augments the search with a conflict-analysis procedure that enables it to backtrack non-chronologically and record the causes of conflict. Experimental results [33] show that GRASP is very efficient for a large number of realistic SAT instances, and it has proven to be a very effective SAT solver in the context of automated hardware design [37]. The CPLEX MIP Solver is a commercial optimizer for linear-programming problems with integer variables (`www.cplex.com`). It employs a branch-and-bound technique using linear-programming relaxations that can be complemented with the dynamic generation of cutting planes. While branch-and-bound is related to depth-first-search, the cutting-planes technique is more powerful than resolution [28]. CUDD (`bessie.colorado.edu/~fabio/CUDD`) implements functions to manipulate Reduced Ordered Binary Decision Diagrams (ROBDDs), which provides an efficient representation for Boolean functions [8]. Unlike GRASP and CPLEX, CUDD does not search for a single satisfying truth assignment. Rather, it constructs a compact symbolic representation of the set of satisfying truth assignments and then checks whether this set is nonempty. Uribe and Stickel [44] compared ROBDDs with the DLL method for SAT solving, concluding that the methods are incomparable, and that ROBDDs dominate the DLL method on many examples. Recent work by Groote and Zantema proved the incomparability of ROBDDs and resolution [26]. ROBDDs have proven recently to be very effective in the context of hardware verification [9, 29].

Our aim was not to directly compare the performance of the different solvers in order to see which one has the "best" performance, but rather to understand their behavior in the $d \times n$ quadrant in order to make qualitative observations on how the complexity of random 3-SAT is viewed from different algorithmic perspectives. It is important to note that the algorithms used in GRASP, CPLEX, and CUDD do not explicitly refer to the density of the input instances. Thus, a qualitative change in the behavior of the algorithm, as a result of changing the density, indicates a genuine structural change in the SAT instances from the perspective of the algorithm.

In analyzing our experimental results we focus on measuring the *median* running time as a function of the order for a set of instances of fixed density.[1] This gives us a measure of the running time of the algorithm for that density. Our findings show that for GRASP and CPLEX the easy-hard-easy pattern is better described as an *easy-hard-less-hard* pattern, where, as is the standard usage in computational complexity theory, "easy" means *polynomial time* and "hard" means *exponential time*. When we start with low-density instances and then increase the density, we go from a region of polynomial running time, to a region of exponential running time, where the exponent first increases and then decreases as a function of the density. Thus, we observe at least *two* phase transitions as the density is increased: a transition at around density 3.8 from polynomial to exponential running time and a transition at around density 4.26 (the crossover point) from an increasing exponent to a decreasing exponent. The region between 3.8 and 4.26 is also characterized by the prevalence of very hard instances, the so called "heavy-tail phenomenon", cf. [23, 27, 34, 38]. Our results indicate one or more phase transitions in this region, where the ratio of the mean to median running time peaks. For CPLEX we also observe

[1] It is easy to see that 3-SAT is NP-complete for instances of each fixed density, as the generic reduction of NP to 3-SAT [22] produces instances of fixed density and each density can be obtained by adding redundant variables or redundant clauses. Thus, we'd expect the worst-cases running time to be exponential for all densities. Consequently, one would expect to find hard instances in the "easy" region, cf. [23]. The issue of median vs. mean running time is discussed later.

another phase transition at around density 1.7 from linear running time to quadratic running time.

A very different picture emerges for CUDD. Here the algorithm is exponential (in both time and space) for densities between 0.5 and 15. There is, however, no peak around the crossover point and no heavy-tail phenomenon was observed. We observed, however, a peak in the size of the final BDDs constructed by the algorithm at around density 2, indicating a phase transition at around this density. At a very low density (0.1) we did observe polynomial (cubic) behavior, which suggests that another phase transition is "lurking" between densities 0.1 and 0.5.

There are two conclusions that can be drawn from our experiments. First, the "phase transition" in average case computational complexity of random 3-SAT is not correlated with the "phase transition" in satisfiability. The sharp shift of average case complexity from polynomial to exponential occurs well before the crossover point. This implies that explanations for shifts in computational complexity cannot center around phenomena observed at the crossover point [38, 36] Second, unlike earlier predictions (cf. [32, 13]), phase transitions in average-case complexity (unlike the one for satisfiability) are not solver-independent. This implies that any theory attempting to explain the sharp shift in computational complexity must take the characteristics of the solver into account, as in [2].

2. Related Work

The fact that the "easy-hard-easy" pattern is quite simplistic is known, though rather under-emphasized, cf. [40, 2]. For example, in the high-density region (above density 5.2), an exponential lower bound on the length of resolution proofs is proved in [12]. This entails an exponential lower bound on the running time of DLL algorithms, implying that the high-density region can, at best, be described as "less hard". (Note that this lower bound does not apply to algorithms that are based on cutting planes or ROBDDs [13, 28, 26].) It is also known that the probability crossover is not the only phase transition involving random 3-SAT and that phase transitions can be solver dependent. In [42], the authors demonstrated experimentally a change from exponentially fast to power law relaxation at around density 3. In [7, 35], the authors proved linear median running time of the pure-literal algorithm at the low-density region (below 1.63) and showed a phase transition at 1.63 for this algorithm. In [21], the authors proved a linear median running time for another heuristic algorithm for low-density instances and showed a phase transition near density 3 for this algorithm.

These latter results indicate that the low-density region is indeed in some sense"easy", but they do not establish that complete SAT solvers have polynomial median running time in this region. The analytical results of Franco and his collaborators suggest that in this region we might expect a polynomial median running time for certain heuristic algorithms, cf. [10], but they do not prove it definitively. In [14], the authors reported linear median running time of Tableau, their SAT solver, for densities 1, 2, and 3, and an exponential median running time for densities 4.26 and 10. In [38], the authors reported linear median running time of their DLL SAT solver for density 3, and an exponential median running time for density 4.26. Neither of these papers, however, systematically explores the dependence on the density of the running time as a function of the order.

The performance of integer-programming algorithms on random SAT instances is studied in [28], but the author did not systematically study how the running time depends on the density of the instances. Similarly, in [6, 25] the authors studied the behavior of ROBDDs on random SAT instances, but did not study how this behavior varies as a function of the density.

While we focus in this paper on the study of collections of fixed-density instances, it would also be interesting to study the behavior of SAT solvers on instances where the order and the density vary simultaneously; for example, the density may increase together with the order. For DLL solvers, the results in [3] show that unless the density increases linearly with the order we should still expect to see exponential running time. Indeed, if one considers a logarithmic increase of the density as a function of the order, then our data (e.g., using Figure 1) shows that the median running time for GRASP is still exponential.

While the finite-size scaling studies in [24, 38] do aim to explore how both the density and the order affect the running time of SAT solvers, they do not reveal the same detailed picture that emerges from our experiments on the density-order quadrant. First, the finite-size scaling studies for SAT are limited to DLL solvers. Second, finite-size scaling studies show a very good fit only around the crossover point, but the fit gets worse as the scale value gets further from it. This makes it very difficult to draw conclusions on the dependence on the order for fixed-density instance in regions with density far below the crossover point. For example, it is not at all clear how one can obtain linear-time behavior at density 3 reported in [38] from their normalized and rescaled results. Finally, the fact that the running time of DLL is exponential in the high-density region and polynomial in the low-density region makes it rather unlikely that the scale factor observed in [38] applies anywhere but very near the crossover point. We elaborate on this point below.

Beame at al. [3] showed that in the high-density region, a certain variant of DLL terminates with high probability within time $2^{an/d+blogn}$, where a and b are some positive constants, n is the order and d is the density of the instance. Thus, the *normalized* running time (i.e., the ratio of the running time to the running time at the crossover point) is $2^{an(1/d-1/t)}$, where t is the crossover density. Thus, in the high-density region the finite-size scale factor is $n(1/d - 1/t)$. More generally, let the running time of a SAT solver in the high-density region be $2^{an^b/d^c}$, where a, b, c, and d are positive constants. Then the scale factor will be $n^b(1/b^c - 1/t^c)$. Note that in the high-density region, order and the density have opposing effects. Increasing the order increases the running time, but increasing density decreases the running time. The scale factor of $n^\alpha(d/t - 1)$ proposed in [38] does not reflect this opposition, as it increases with both n and d, which explains why the study in [38] shows a very good fit only around the crossover point.

On the other hand, our experiments, as well as other experiments mentioned above, provide evidence to the fact that in the low-density region the running time of DLL solvers is polynomial, i.e., $f(d)n^e$, where f is some function and e is a positive constant. Thus, the normalized running time is $f(d)n^e/2^{an^b/t^c}$. Therefore, the scale factor $n^\alpha(d/t - 1)$ of [38] does not appear to be a reasonable scale factor in this region. For the low-density region, it makes more sense to normalize the running time with respect to some other thresold s in the low-density region. The normalized running time is then $f(d)/f(s)$, which implies that d is the appropriate scale factor in this region. The bottom line regarding finite-size scaling is that running times in the low- and high-density regions are very different functions of density and order. Expecting the same scale factor to work in both regions is unrealistic.

As noted above, since the sharp shift of average-case complexity from polynomial to exponential is solver dependent and occurs well before the crossover point for all the solvers we tested, explanations for this shift cannot be solver independent and cannot center around phenomena observed at the crossover point [38, 36]. In an interesting recent development, Achlioptas, Beame, and Molly,[2] showed that mixtures of 2-clauses with density $1 - \epsilon$ and 3-clauses with density 2.28, which are unsatisfiable with high probability, take DLL solvers to take an exponential time to refute. If $(1 - \epsilon, 2.28)$ 2-clause/3-clause mixtures occur during the solution of a satisfiable 3-SAT instance, then a DLL solver will take time exponential in the order of the instance to solve it. Achlioptas et al. used this to show that a certain DLL solver behaves exponentially at density 3.81. This could also provide an explanation for our observed exponential behavior of GRASP (which is a modified DLL solver) in the low-density region.

3. Experimental Setup

Our experimental setup is identical to that of [15, 39]. We generate dn clauses, each by picking three distinct variables (out of n) at random and choosing their polarity uniformly. For each studied point in the $d \times n$ quadrant we generate at least 100 random instances and apply our solvers. Our experiments were run on Sun Ultra 1 machines. As in [39], we chose to focus on median running time rather than mean running time. The difficulty of completing the runs on very hard instances makes it less practical to measure the mean. Furthermore, the median and the mean are typically quite close to each other, except for the regions that display heavy-tail phenomena, where the median and the mean diverge dramatically [38]. It would be interesting to analyze our data at percentiles other than the 50th percentile (the median) (cf. [38]), though a meaningful analysis for high percentiles would require many more sample points than we have in our experiments.

For the statistical analysis and plotting of data, we used MATLAB, which is an integrated technical computing environment that combines numeric computation, advanced graphics and visualization, and a high-level programming language. The MATLAB (`www.mathworks.com`) functions we used for statistical analysis were:

- *polyfit*, for computing the best linear, quadratic, or cubic fit to the data (or the logarithm of the data) using polynomial regression, and

- *corrcoef*, for computing r^2, the square of correlation (r^2 is the fraction of the variance of one variable that is explained by regression on the other variable).

Unless stated otherwise, for the results reported in this paper, r^2 exceeded 0.98. This establishes high confidence in the validity of the fit of the curve to the data points.

4. Random 3-SAT and GRASP

GRASP [33] is a SAT solver that augments the basic backtracking search with a conflict-analysis procedure. In order to cut down on the search space, a dynamic-learning mechanism based on diagnosing the causes of the conflicts is used. By analyzing conflicts and discovering their causes, GRASP can backtrack non-chronologically to earlier levels in the search tree, potentially pruning large portions of the search space. Moreover, by recording the causes of conflicts, GRASP can avoid running into similar conflicts later during the search.

The experiments described in this section were run on a Sun Ultra 1 with a 167MHz UltraSPARC processor and 128MB RAM. Some changes were made to the default GRASP configuration; we increased the maximum number of backtracks allowed to 1,000,000 and the maximum number of conflicts allowed to 2,000,000. CPU time limit was set to 10,800 seconds. These changes were necessary in order to limit the portion of SAT instances on which GRASP aborted. This artificially lowers our measurements of mean running time, but does not affect our measurements of median running time.

The goal of the experiments was to evaluate GRASP's performance on an initial quadrangle of the $d \times n$ quadrant. We explored densities from 0.9 to 15. The order of the instances explored depends on the density:

- Density 0.9: 2000 variables (25 variables per step)
- Densities 1, 2 and 3: 1000 variables (10 variables per step)
- Density 3.6: 800 variables (10 variables per step)
- Density 3.7: 480 (10 variables per step)
- Density 3.8: 450 variables (10 variables per step)
- Density: 4.26: 170 variables (10 variables per step)
- Density 5: 210 variables (10 variables per step)
- Densities 4, 6-15: 250 variables (10 variables per step)

In Figure 1 the median running time is shown on a logarithmic (base 2) scale. (For densities 4.26 and 5 we extrapolated the data up to 250 variables).

We analyzed the median running time as a function of the order for fixed density instances. For low densities (at or below 3.6), our data indicate a quadratic running time. See Figures 1 and 2, where we plot the median running time as a function of the order for instances of density 0.9 and 3.6, respectively. The quadratic behavior of GRASP at low densities should be contrasted with the linear running time at

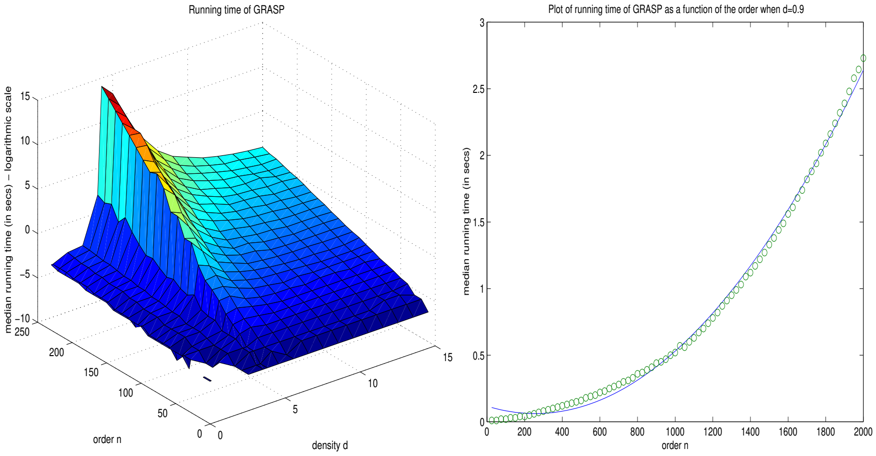

Figure 1. GRASP – (left) 3-D Plot of median running time, and (right) median running time for density 0.9 as a function of the order of the instances. A quadratic function fits these points better (with an $r^2 > 0.98$) than an exponential function.

low densities that was reported in [14, 38]. It seems that GRASP's conflict-analysis component has a quadratic overhead.

At densities 3.8 and above, the median running time is exponential in the order, i.e., it behaves as $2^{\alpha n}$, where the exponent α depends on d (see discussion below). See Figure 2 where we plot the median running time as a function of the order for instances of density 3.8 (the r^2 for this plot is 0.95). Thus, a phase transition seems to occur between densities

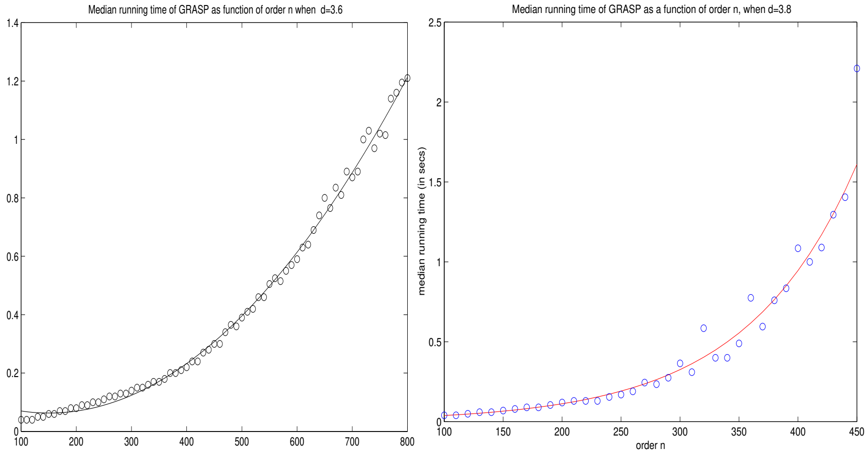

Figure 2. GRASP – median running time for density 3.6 (left) and density 3.8 (right) as a function of the order of the instances. At density 3.6, the best fit curve is quadratic in the order, while at 3.8, the best fit curve is exponential in the order.

3.6 and 3.8, where the median running time shifts from polynomial to exponential. As the density is increased beyond 3.8, the exponent α also increases. It peaks at around density 4.26, after which it declines with increased density. Thus, we observe two phase transitions. The second one, in which the exponent reaches its peak, essentially coincides with the crossover point, at which the probability of satisfiability is 0.5. This is the phase transition that was reported at [39] and then studied extensively. This transition, however, is preceded by another one, in some sense a more significant one, at around density 3.8, where we observe a qualitative shift in the behavior of GRASP. A transition from polynomial to exponential behavior in graph coloring was conjectured in [27] and counter-conjectured in [17]. Such a transition in random 3-SAT near the crossover point is claimed in [14]; this claim, however, was removed in a later paper [15]. We believe that we are the first to demonstrate such a transition in random 3-SAT, and to show that it occurs significantly below the crossover point.

The phase transition at around 3.8 is accompanied by a "heavy-tail phenomenon", which is a prevalence of *outliers*, i.e., instances on which the actual running time is at least an order of magnitude (10) larger than the median running time, as well as a divergence of the mean and the median. See Figure 3, where we plot the mean to median ratio and the proportion of outliers as a function of the density. The plots show a

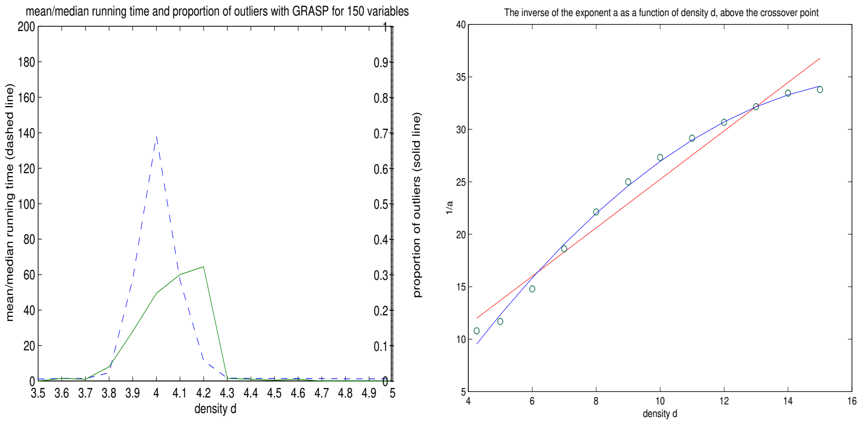

Figure 3. GRASP – (left) Ratio of mean to median running time and the proportion of outliers, and (right) the exponent $1/\alpha$ of median running time as a function of density.

drastic change in the region between density 3.7 and density 4.3. Both plots show a quick rise and decline. The mean to median ratio peaks at around density 4.0 and the proportion of outliers peaks at around density 4.2. For densities between 3.7 and 4.0 we found it quite difficult to analyze the median running time as a polynomial (of low degree) or exponential function of the order (note the lower r^2 reported above for density 3.8).

Our data suggest that as the density increases from 3.7 to 4.3, random 3-SAT formulas go through a series of changes and perhaps more than one phase transition. The heavy-tail phenomenon for random 3-SAT deserves further study (with many more samples per point in the $d \times n$ quadrant) to confirm our findings. In particular, the divergence of the two peaks in Figure 3 needs to be reconfirmed or refuted.

As noted above, beyond density 4.26 the exponent α declines. A theoretical analysis suggests that for DLL solvers α may decline inversely linearly, i.e., as $\frac{c}{d}$, for some constant c, see [3]. Our data, however, suggest a slower decline, even though one may expect GRASP to be faster than DLL solvers. See Figure 3, where we plot $\frac{1}{\alpha}$ as a function of d. Thus, GRASP is not as efficient in the high-density region as it could be. (We should caution, however, that we only have 11 data points, and these data points themselves have been obtained by fitting a linear curve to the logarithm of the median running time. Thus, the finding of a slower decline should be viewed as quite preliminary.)

5. Random 3-SAT and CPLEX

The CPLEX MIP Solver is a commercial linear-programming solver for integer variables. It employs a branch-and-bound technique starting from a linear-programming relaxation of the given integer-programming problem. This may be complemented with the dynamic generation of cutting planes [5].

The experiments described in this section were run on a Sun Ultra 1 with a 167MHz UltraSPARC processor and 64 MB RAM. SAT problems were encoded as 0-1 integer-programming problems. Values true and false are represented as 1 and 0. For a clause to be true the sum of the representations of the literals has to be greater or equal to 1. For example, the clause $\neg x_1 \vee x_2 \vee \neg x_3$ is represented by the inequality $(1 - x_1) + x_2 + (1 - x_3) \geq 1$.

We used CPLEX to solve problems for densities from 0.9 to 15. The order of the instances was chosen according to the density:

- Densities 0.9, 1.5, 1.6, 1.7 and 1.8: 2000 variables (25 variables per step)
- Density 1: 10000 variables (50 variables per step)
- Density 2: 1800 variables (25 variables per step)

– Densities 3, 4, 4.26, and 5-15: 120 variables (10 variables per step)

In Figure 4, the median running time is shown on a logarithmic (base 2) scale. Note that the peak at the crossover point is much less pronounced than the one in Figure 1.

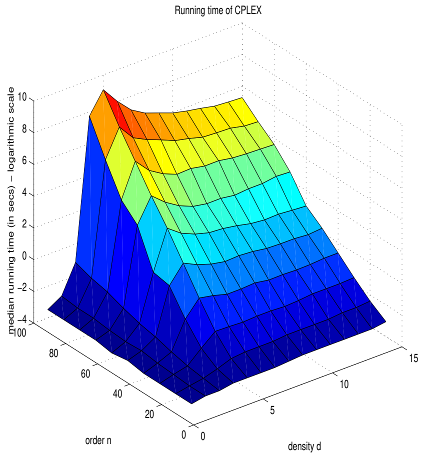

Figure 4. CPLEX – 3-D Plot of median running time

The median running time was analyzed as a function of the order for fixed density-instances. For low densities (below 1.7) our data indicate a linear running time. See Figure 5 for median running times for instances of density 1 with up to 10000 variables. For densities 2, 3, and up to 3.5,

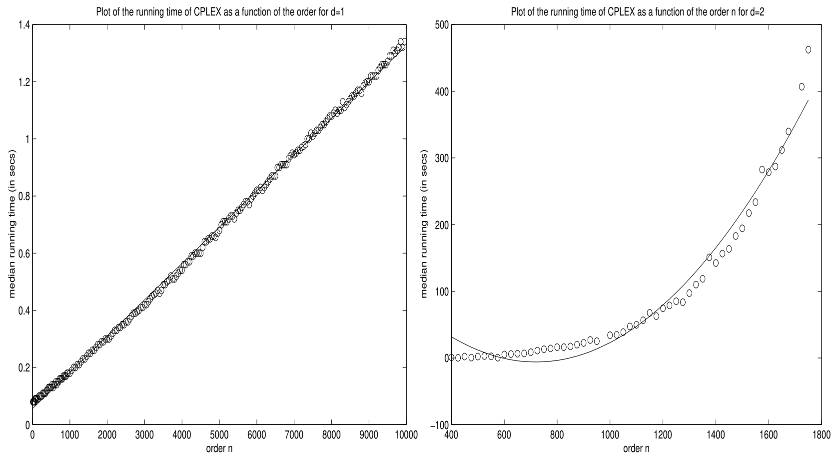

Figure 5. CPLEX – median running time for density 1 (left) and density 2 (right) as a function of the order of the instances.

the median running time is quadratic. See Figure 5 for median running time for instances of density 2, where for order above 400 the behavior is quadratic. Thus we seem to have a phase transition, corresponding to a shift from linear to quadratic behavior, between densities 1 and 2. This may coincide with the phase transition proved in [7, 35] around density 1.63, as described in Section 2.

At densities 4.0 and above, the median running time is exponential in the order, i.e., it behaves as $2^{\alpha n}$, where the exponent α depends on d. As with GRASP, a phase transition seems to occur between densities 3.6 and 4.0. It corresponds to the shift from polynomial to exponential behavior, and is accompanied by heavy-tail phenomena. See Figure 6, where we plot the mean to median ratio and the proportion of outliers as a function of the density. Note that the heavy-tail phenomenon for

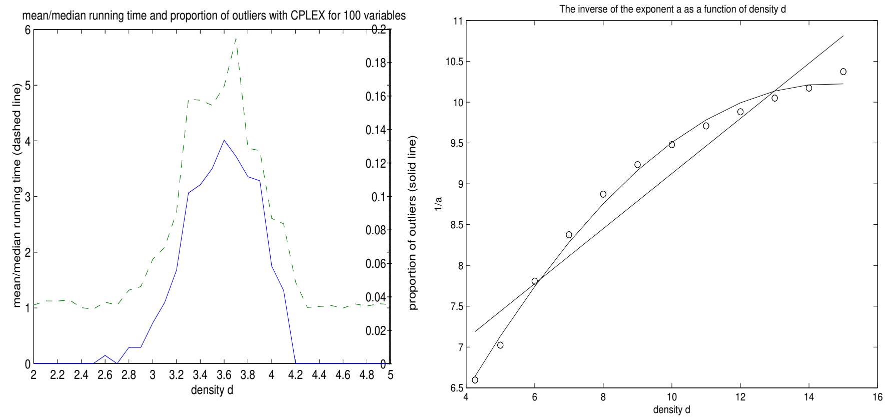

Figure 6. CPLEX – (left) Ratio of mean to median running time and proportion of outliers, and (right) the exponent $1/\alpha$ of median running time as a function of density.

CPLEX is not as marked as with GRASP; both peak mean-to-median ratio and peak proportion of outliers are lower for CPLEX than for GRASP. Intriguingly, the peak for CPLEX occurs at lower densities (around 3.6) then for GRASP (around 4.0).

As with GRASP, the exponent α peaks at density 4.26 and then declines. Again, our data show a slower decline than $\frac{c}{d}$, as suggested in [3] (though the analysis there is for resolution-based procedures, which are weaker than the cutting-planes method used in CPLEX.) See Figure 6, where we plot $\frac{1}{\alpha}$ as a function of d.

6. Random 3-SAT and CUDD

CUDD [41] is a package that provides functions for the manipulation of Boolean functions, based on the reduced, ordered, binary decision diagram (ROBDD) representation [8]. A binary decision diagram (BDD) is a rooted directed acyclic graph that has only two terminal nodes labeled 0 and 1. Every non-terminal node is labeled with a Boolean variable and has two outgoing edges labeled 0 and 1. An ordered binary decision diagram (OBDD) is a BDD with the constraint that the input variables are ordered and every path in the OBDD visits the variables in ascending order. An ROBDD is an OBDD where every node represents a distinct logic function.

Unlike GRASP and CPLEX, CUDD does not search for a satisfying truth assignment. Rather, it constructs a compact symbolic representation of the set of *all* satisfying truth assignments. Then, the resulting ROBDD is compared against the predefined constant 0 in order to find if an instance is (un)satisfiable. It is important to note that very large sets of truth assignments can have very compact ROBDD representation [8], which explains the effectiveness of ROBDDs in hardware verification [9, 29]. As we see later, CUDD performs well in the very-low-density region, where the set of satisfying truth assignment is very large.

The experiments described in this section were run on a Sun Ultra 1 with a 167MHZ UltraSPARC processor and 64MB RAM. The CUDD package has been used through the GLU C–interface [43], a set of low-level utilities to access BDD packages. It is well known that the size of the ROBDD for a given function depends on the variable order chosen for that function. We have used automatic dynamic reordering during the tests with the default method for automatic reordering of CUDD.

As in the preceding two sections, the goal of the experiments was to evaluate CUDD's performance on an initial quadrangle of the $d \times n$ quadrant. We explored densities 0.1, 0.5, and 1 to 15. The order of the instances explored depends on the density:

- Density 0.1: 1470 variables (10 variables per step)
- Density 0.5: 136 variables (2 variables per step)
- Density 0.9 and 1: 68 variables (2 variables per step)
- Densities 1.5, 2-4, 4.26, 5-15: 46 variables (2 variables per step)

In Figure 7 the median running time is shown on a logarithmic (base 2) scale. Note the absence of a peak (contrast with Figures 1 and 4).

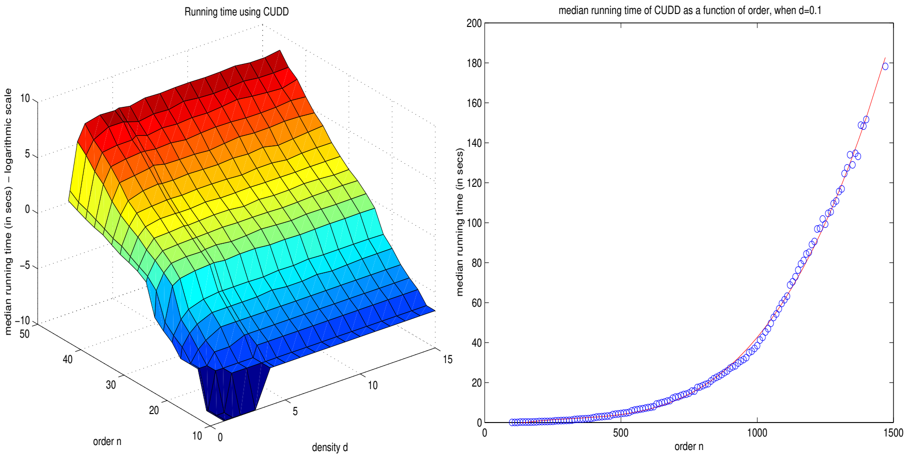

Figure 7. CUDD – (left) 3-D Plot of median running time and (right) median running time for density 0.1

We analyzed the median running time as a function of the order for fixed-density instances. At densities 0.5 and above, the median running time is exponential in the order, i.e., it behaves as $2^{\alpha n}$. At density 2 and above the exponent α is independent of the density. In particular, there seems to be nothing special about the crossover point at density 4.26. The explanation for this behavior is that the running time of ROBDD-based algorithms is determined mostly by the size of the manipulated ROBDDs. Our algorithm involves dn product operations between a possibly large ROBDD (representing all truth assignments of the clauses processed so far) and a small ROBDD (representing seven truth assignments of the currently processed clause). Thus, the running time of our algorithm is determined by the largest intermediate ROBDD constructed. As is shown in Figure 8, the peak in ROBDD size is attained after processing about $2n$ clauses, which explains the flattening of the running-time plot at density 2, and suggests that a phase transition in terms of ROBDD size occurs at round this density.

As ROBDDs are symmetrical with respect to the set they represent and its complement, both very small sets and very large sets can be represented by small ROBDDs [8]. This suggests that we may see polynomial behavior for very low density instances, which have a large number of satisfying truth assignments. To check this conjecture we measured the median running time of CUDD for instances of density 0.1. Our results indicate a cubic-time behavior, see Figure 7. This suggests the existence of another phase transition between densities 0.1 and 0.5. This result should be contrasted with that of [40], in which

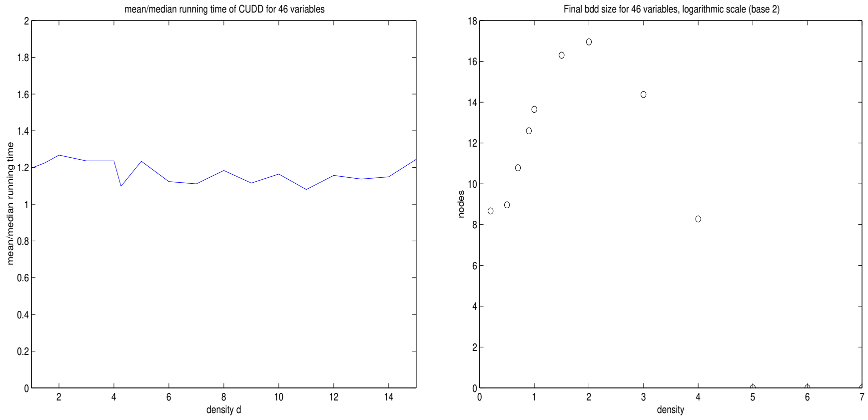

Figure 8. CUDD – (left) Ratio of mean to median running time and (right) median ROBDD size as a function of density

the running time for explicitly enumerating all solutions of random constraint-satisfaction instances increases as the density decreases.

Unlike with GRASP and CUDD, we did not observe a heavy-tail phenomenon with CUDD: there are no outliers and the mean to median ratio is independent of the density (see Figure 8).

7. Discussion

We provide experimental evidence for the following hypotheses. First, there is no connection between the phase transition in computational complexity and the phase transition in satisfiability. It is not the case that the shift from polynomial to exponential complexity occurs at or very close to the crossover point, as has been widely believed [38, 39]. Second, not only does the density at which the shift from polynomial to exponential time complexity vary with the choice of solver, but the very shape of the surface of the median running time (an experimental surrogate for average-time complexity), as a function of the density d and the order n, changes with the solver. Finally, the density-order quadrant contains several phase transitions; in fact, the region between density 0 and density 4.26 seems to be rife with phase transitions, which are also solver dependent. In essence, each solver provides us with a different tool with which to study the complexity of random 3-SAT. This is analogous to astronomers observing the sky using telescopes that operate at different wave lengths. We thus hope to alleviate the "fixation" with DLL solvers and the crossover point at 4.26.

Our experiments reveal a marked difference between solvers like GRASP and CPLEX, which are search based and display interesting similarities in the shapes of the median running time surface despite their different underlying algorithmic techniques, and ROBDD-based solvers, like CUDD, which are based on compactly representing all satisfying truth assignments. While the interesting region for GRASP and CPLEX is between 3.7 and 4.3, the interesting region for CUDD occurs below density 2. This refutes earlier conjectures (cf. [32]) that the peak in median running time around the crossover point is essentially solver independent. For both GRASP and CPLEX, we observed a new phase transition at around density 3.8, where the median running time shifts from being polynomial in the order to being exponential in the order. From the perspective of average-time complexity this is a significant phase transition because it corresponds to a qualitative shift in the behavior of the solver. We also observed several other phase transitions for CPLEX and for CUDD. This suggests that it would be interesting to explore the behavior of other SAT solvers, such as RELSAT or SATZ, on the $d \times n$ quadrant.

With fine grained sampling of the density parameter, and by exploring a greater range in the number of variables, we can start to document for each solver, phase transitions that correspond to significant shifts in the shape of the running time of the solver. These phase transitions are important to our understanding of the computational complexity of random 3-SAT, and can be used as a justification to develop density-based solvers for 3-SAT, i.e., solvers which use information about the density of an instance, to choose the most appropriate algorithmic technique.

While our results are purely empirical, as the lack of success with formally proving a sharp complexity threshold at the crossover point indicates (cf. [20, 19, 1]), providing rigorous proof for our qualitative observations may be a very difficult task, especially for sophisticated solvers like the ones studied in this paper.

Acknowledgement. The last author is grateful to Phokion G. Kolaitis for stimulating discussions.

References

1. D. Achlioptas. Setting two variables at a time yields a new lower bound for random 3-SAT. In *Proc. 32th ACM Symp. on Theory of Computing*, pages 28–37, 2000.
2. D. Achlioptas, P. Beame, and M. Molloy. A sharp threshold in proof complexity. In *Proc. 33th ACM Symp. on Theory of Computing*, 2001.
3. P. Beame, R. M. Karp, T. Pitassi, and M. E. Saks. On the complexity of unsatisfiability proofs for random k-CNF formulas. In *Proc. 30th ACM Symp. on Theory of Computing*, pages 561–571, 1998.

4. A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. 36th Conf. on Design Automation*, pages 317–320, 1999.
5. R. E. Bixby. Implementing the Simplex method: The initial basis. *ORSA J. on Computing*, 4(3):267–284, 1992.
6. F. Bouquet. *Gestion de la dynamicité et énumération d'implicants premiers: une approche fondée sur les Diagrammes de Décision Binaire*. PhD thesis, Université de Provence, France, 1999.
7. A. Z. Broder, A. M. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. In *Proc. 4th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 322–330, 1993.
8. R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Computers*, 35(8):677–691, 1986.
9. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.
10. M. Chao and J. V. Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem. *Information Sciences*, 51:289–314, 1990.
11. P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proc. 12th Int'l Joint Conf. on Artificial Intelligence (IJCAI '91)*, pages 331–337, 1991.
12. V. Chvátal and E. Szemerédi. Many hard examples for resolution. *J. of the ACM*, 35(4):759–768, 1988.
13. S. A. Cook and D. G. Mitchell. Finding hard instances of the satisfiability problem: A survey. In Du, Gu, and Pardalos, editors, *Satisfiability Problem: Theory and Applications*, volume 35 of *Dimacs Series in Discrete Math. and Theoretical Computer Science*. AMS, 1997.
14. J. M. Crawford and L. D. Auton. Experimental results on the crossover point in satisfiability problems. *AAAI*, pages 21–27, 1993.
15. J. M. Crawford and L. D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1-2):31–57, 1996.
16. J. M. Crawford and A. B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. *AAAI*, 2:1092–1097, 1994.
17. J. Culberson and I.P. Gent. Frozen development in graph coloring. Technical report, Department of Computer Science, University of St. Andrews, 2000. APES Research Report APES-19-20004.
18. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Comm. of the ACM*, 5:394–397, 1962.
19. O. Dubois, Y. Boufkhad, and J. Mandler. Typical random 3-SAT formulae and the satisfiability theshold. In *Proc. 11th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 126–127, 2000.
20. E. Friedgut. Necessary and sufficient conditions for sharp threshold of graph properties and the k-SAT problem. *J. Amer. Math. Soc.*, 12:1017–1054, 1999.
21. A. Frieze and S. Suen. Analysis of two simple heuristics for random instances of k-SAT. *J. of Algorithms*, 20(2):312–355, 1996.
22. M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.
23. I. P. Gent and T. Walsh. Easy problems are sometimes hard. *Artificial Intelligence*, 70(1-2):335–345, 1994.

24. I.P. Gent and T. Walsh. The SAT phase transition. In *Proc. European Conf. on Artificial Intelligence*, pages 105–109, 1994.
25. J. F. Groote. Hiding propositional constants in BDDs. *Formal Methods in System Design*, 8:91–96, 1996.
26. J.F. Groote and H. Zantema. Resolution and binary decision diagrams cannot simulate each other polynomially. Technical report, Department of Computer Science, Utrecht University, 2000. Technical Report UU-CS-2000-14.
27. T. Hogg and C. P. Williams. The hardest constraint problems: A double phase transition. *Artificial Intelligence*, 69(1-2):359–377, 1994.
28. J. N. Hooker. Resolution vs. cutting plane solution of inference problems: Some computational experience. *Operations Research Letters*, 7:1–7, 1988.
29. S. Jha, Y. Lu, M. Minea, and E.M. Clarke. Equivalence checking using abstract BDDs. In *Proc. Int'l Conf. on Computer Design (ICCD'97)*, pages 332–337, 1997.
30. H. Kautz and B. Selman. Planning as satisfiability. In *Proc. European Conference on Artificial Intelligence*, pages 359–379, 1992.
31. S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random formulas. *Science*, 264:1297–1301, 1994.
32. T. Larrabee and Y. Tsuji. Evidence for a satisfiability threshold for random 3CNF formulas. Technical report, University of California, Santa Cruz, 1992. Technical report USCS-CRL-92042.
33. J. P. Marques Silva and K. A. Sakallah. GRASP–A search algorithm for propositional satisfiability. *IEEE Trans. on Computers*, 48(5):506–521, 1999.
34. D. G. Mitchell and H. J. Levesque. Some pitfalls for experimenters with random SAT. *Artificial Intelligence*, 81(1-2):111–125, 1996.
35. M. Mitzenmacher. Tight thresholds for the pure literal rule. Technical report, Digital System Research Center, Palo Alto, California, 1997. SRC Technical Note 1997 - 011.
36. R. Monasson, R Zecchina, S. Kirkpatrick, B.Selman, and L.Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400:133–137, 1999.
37. G.J. Nam, K.A. Sakallah, and R. A. Rutenbar. Satisfiability-based layout revisited: Detailed routing of complex FPGAs via search-based boolean sat. In *Proc. ACM Int'l Symp. on Field-Programmable Gate Arrays (FPGA'99)*, pages 167–175, 1999.
38. B. Selman and S. Kirkpatrick. Critical behavior in the computational cost of satisfiability testing. *Artificial Intelligence*, 81(1-2):273–295, 1996.
39. B. Selman, D. G. Mitchell, and H. J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81(1-2):17–29, 1996.
40. B.M. Smith and M.E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence Journal*, 8(1–2):155–181, 1996.
41. F. Somenzi. CUDD: CU Decision Diagram package. release 2.3.0., 1998. Dept. of Electrical and Computer Engineering. University of Colorado at Boulder.
42. P. Svenson and M.G. Nordahl. Relaxation in graph coloring and satisfiability problems. *Phys. Rev. E*, 59(4):3983–3999, 1999.
43. The VIS Group. VIS: A system for verification and synthesis. In *Proc. 8th Int'l Conf. on Computer Aided Verification (CAV '96)*, pages 428–432, 1996. LNCS 1102. Ed. by R. Alur and T. Henziger.
44. T. E. Uribe and M. E. Stickel. Ordered binary decision diagrams and the Davis-Putnam procedure. In *First Int'l Conf. on Constraints in Computational*

Logics, volume 845 of *Lecture Notes in Computer Science*, pages 34–49, Munich, September 1994. Springer-Verlag.