Hidden Threshold Phenomena for Fixed-Density SAT-formulae

Hans van Maaren and Linda van Norden

Technical University Delft
Faculty of Information Technology and Systems
Department of Information Systems and Algorithms
Mekelweg 4, 2628 CD Delft
The Netherlands
H.vanMaaren@ewi.tudelft.nl, L.vanNorden@ewi.tudelft.nl

Abstract. Experimental evidence is presented that hidden threshold phenomena exist for fixed density random 3-SAT and graph-3-coloring formulae. At such fixed densities the average Horn fraction (computed with a specially designed algorithm) appears to be a parameter with respect to which these phenomena can be measured. This paper investigates the effects of size on the observed phenomena.
Keywords: satisfiability, Horn, thresholds, 3-SAT, graph-3-coloring

1 Introduction

It is well-known that the class of random 3-SAT problems is subject to threshold phenomena with respect to the density parameter. Given a random 3-SAT instance, without unit clauses and two-literal clauses, its density d is defined as the ratio of the number of clauses and the number of variables. Low density problem are (with high probability) satisfiable and easily shown to be satisfiable with DPLL techniques. High density problems are (with high probability) unsatisfiable. At a certain value $d_0(n)$, the threshold value for instances with n variables, a phase-transition occurs: instances generated with this particular density are satisfiable with probability 0.5 and solving them with DPLL techniques is very expensive relative to instances with larger or smaller densities. The fact that threshold $d_0(n)$ exists is proven ([10]), but determining the exact position of this threshold for a certain size n is a difficult problem. A large variety of experiments supports the claim that $d_0(n)$ is located in the neighborhood of 4.27.

There is a vaste literature on phase-transitions in the 3-SAT problem and related problems. [13] and [2] study threshold phenomena with respect to the density parameter in 3-SAT and the class of $(2+p)$-SAT problems. In a $(2+p)$-SAT-formula a fraction p of the clauses has three literals while the others have two literals.

In this paper we provide experimental evidence that the class of random 3-SAT problems with fixed density again is subject to similar phenomena with respect to a different parameter: an enhanced version of the Horn fraction. Horn

E. Giunchiglia and A. Tacchella (Eds.): SAT 2003, LNCS 2919, pp. 135–149, 2004.
© Springer-Verlag Berlin Heidelberg 2004

clauses are clauses in which at most one variable occurs in negated form. The Horn fraction of a CNF-formula is the number of Horn clauses divided by the total number of clauses. Various notions of Horn fraction will be discussed more extensively in Section 3.

Before going into details, we consider Figure 1. At this moment it is enough to know that Figure 1 is based on the use of an algorithm, which we will refer to as average-Horn SAT (aHS for short), described in Section 7. In solving an instance it creates a search tree in which at each node the Horn fraction of the current CNF is computed. On the vertical axis is indicated the number of nodes aHS needs to solve the problem. On the horizontal axis is indicated the average of the Horn fractions over all nodes of the search tree. The unsatisfiable instances are indicated by crosses and the satisfiable ones by dots. The test set is a set of 150 unsatisfiable and 150 satisfiable random 3-SAT instances with 120 variables and density 4.3. All random 3-SAT instances used in this and the next experimental tests are generated by the OK-generator ([11]).

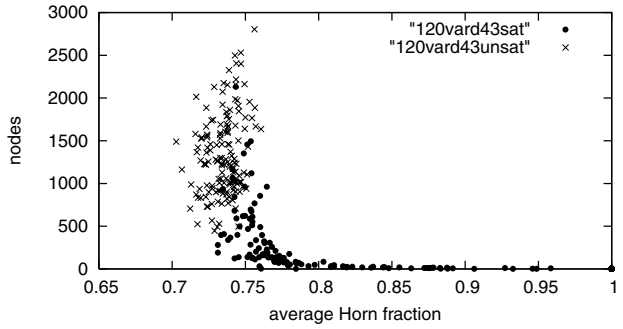

Fig. 1. 120 variables, d=4.3

Examining Figure 1 we can distinguish a few interesting features:

1. A correlation between average Horn fraction and satisfiability
2. A weak form of linear separation between satisfiable and unsatisfiable instances seems possible, in the sense that a line $N = \alpha H + \beta$, in which N is the number of nodes and H the average Horn fraction, can be drawn which constitutes a weak separation.
3. A correlation between average Horn fraction and the number of nodes it took to solve an instance (independent of the satisfiability of the instance).

The obvious questions to ask at this stage are

a. Is it necessary to solve an instance completely to establish a good approximation of its average Horn fraction?
b. Is the correlation between computational effort and average Horn fraction a phenomenon which is typically solver dependent?

c. Are these phenomena size dependent?
d. Do similar pictures arise for models with other densities?
e. Is the phenomenon typical for 3-SAT, or can we expect similar behavior for other classes of CNF-formulae?

We ran several experiments to gain more evidence for the observed features and to get a first impression of what the answers to these questions might be. In this paper we focus on the size independency of the correlations between the Horn fraction, satisfiability and computational effort for random 3-SAT formulae and graph-3-coloring problems at the threshold. We also established density independency and linear separation possibilities for these type of problems, but on a less extensive experimental scale. These latter results were presented at the SAT03 conference. However, because of space limitations we do not present them here.

2 Preliminaries

A clause is a disjunction of literals. A literal is an atomic propositional variable or its negation (indicated by $\neg$). A CNF-formula is a logical conjunction of clauses. The satisfiability problem is to find an assignment of the truth-values true and false to the propositional variables such that all clauses in the CNF-formula are true or to prove that such truth assignment does not exist. A formula is satisfiable if a truth assignment exists and unsatisfiable otherwise. [5] proved that the satisfiability problem is $\mathcal{NP}$-hard. In this paper, we use 3-SAT to denote the class of problems where the CNF-formula consists only of clauses with exactly three literals (some authors use {3}-SAT for notation). [5] also proved that the class of 3-SAT problems is $\mathcal{NP}$-hard.

3 Horn Fractions

In this section, we will describe different variants of the Horn fraction of an instance. Our aim is to define a concept of 'Horn fraction' of a CNF-formula $\mathcal{F}$ which can be computed efficiently and reveals similar distributions as in Figure 1. A Horn clause is a clause with at most one negated variable.

Definition 1. *The Horn fraction $H(\mathcal{F})$ of a CNF-formula $\mathcal{F}$ is the number of Horn clauses divided by the total number of clauses.*

CNF-formulae with only Horn clauses can be solved in time linear in the number of literals by the algorithm in [7]. Renaming, or flipping, a variable X_i means that every occurrence of X_i is replaced by $\neg X_i$ and every $\neg X_i$ is replaced by X_i. By renaming, the Horn fraction might change.

Definition 2. *The optimal Horn fraction of a CNF-formula $\mathcal{F}$ after renaming, $H_{opt}(\mathcal{F})$ is the largest fraction of Horn clauses that can be obtained by renaming, or flipping, variables in $\mathcal{F}$.*

The problem of computing this optimal Horn fraction after renaming is $\mathcal{NP}$-hard ([6]). Recognizing whether a formula $\mathcal{F}$ is renamable Horn, i.e. $H_{opt}(\mathcal{F}) = 1$, however, can be done in linear time, for example by the algorithm of [1]. [4] proved $\mathcal{NP}$-completeness of the related problem of finding a renamable Horn subproblem with as many variables as possible.

Definition 3. *Given a CNF-formula $\mathcal{F}$ and a deterministic renaming routine $\mathcal{R}$, the sub-optimal Horn fraction after renaming, $H_{subopt}(\mathcal{F}, \mathcal{R})$, is the Horn fraction obtained after applying routine $\mathcal{R}$.*

[3] presents a rounding procedure to approximate $H_{opt}(\mathcal{F})$ that has a guaranteed performance ratio of $\frac{40}{67}$ on 3-SAT problems. This procedure rounds the solution of a linear program or the vector with all 0.5, depending on the objective value of the linear program, to an integer valued solution indicating which variables have to be flipped. The routine $\mathcal{R}$ we use in aHS to sub-optimize the Horn fraction, is a deterministic greedy local search procedure. It is simple and its performance on random 3-SAT problems seems to be quite good: for instances with 40 variables, the sub-optimal Horn fraction after renaming computed by our heuristic is only about 1.9% from the optimal Horn fraction after renaming. Computing the optimal Horn fraction after renaming for larger instances cannot be done in reasonable time.

Definition 4. *Given a CNF-formula $\mathcal{F}$, a deterministic renaming routine $\mathcal{R}$, and a deterministic search tree T which solves $\mathcal{F}$, the sub-optimal average Horn fraction over the search tree (in short average Horn fraction), $AH_{so}(\mathcal{F}, \mathcal{R}, T)$, is the average of the $H_{subopt}(\mathcal{F}_t, \mathcal{R})$ over all nodes t in the search tree T, with $\mathcal{F}_t$ the CNF-formula at node t of T.*

The design of the search tree we use for computing $AH_{so}(\mathcal{F}, \mathcal{R}, T)$ is described in Subsection 7.5. To compute $AH_{so}(\mathcal{F}, \mathcal{R}, T)$, the instance has to be solved completely. As already mentioned in question a in the introduction, we would like to have a method for computing a Horn fraction without having to solve $\mathcal{F}$ completely.

Definition 5. *Given a CNF-formula $\mathcal{F}$, a deterministic renaming routine $\mathcal{R}$, and a deterministic search tree T which solves $\mathcal{F}$, the k-level average Horn fraction, $AH_{so}(\mathcal{F}, \mathcal{R}, T, k)$, is the average Horn fraction over the nodes in T up to level k.*

The above definitions are sound because the routine and search tree are deterministic. As we shall show in Section 7, our choices of $\mathcal{R}$ and T guarantee that $AH_{so}(\mathcal{F}, \mathcal{R}, T, k)$ can be computed in polynomial time for fixed k. An appropriate choice for k depends on the size and nature of the instances. Dealing with problems of the size we investigated in this paper, $k = 2$ or $k = 3$ seem to be appropriate choices.

Figure 2 illustrates the distribution of the same instances as in Figure 1, but now with respect to their 2-level average Horn fractions. On the horizontal axis is indicated the 2-level average Horn fraction of the instance, computed by

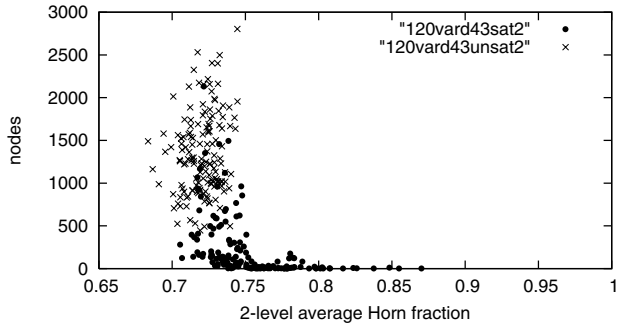

Fig. 2. 120 variables, d=4.3

aHS. On the vertical axis is the number of nodes it took to solve the instance completely by aHS. Figure 2 shows that it is not necessary to solve an instance completely to have an appropriate concept of average Horn fraction which shows a similar distribution.

To test whether this kind of distribution also occurs with respect to the optimal Horn fraction after renaming, we computed this optimal Horn fraction, using the Integer Linear Program proposed by [3] for a set of random 3-SAT instances with 40 variables. Plotting this optimal Horn fraction against the number of nodes it took aHS to solve these instances, revealed no clear correlation. However, if we plot the same instances against the 1-level average Horn fraction the correlations discussed show up. Hence, this type of distribution seems to be specific for the way aHS computes the k-level average Horn fraction (that is, the way aHS decomposes the search space).

3.1 Weighted and Max-Horn Variants

Each of the above notions can be extended to weighted variants. In these variants, the weighted fraction of Horn clauses is optimized or approximated. This weighted fraction is defined as

$$\frac{\sum_{l=1}^{n} w_l H_l}{\sum_{l=1}^{n} w_l S_l}$$

in which n is the number of variables, w_l is a weight of clauses of length l, S_l is the number of clauses of length l and H_l is the number of Horn clauses of length l. Varying weights might influence computation times of our algorithm and the observed distributions significantly, but experiments showed that this is not the case for random 3-SAT problems.

One might consider also the k-level max-Horn fraction which is defined as the maximum of the Horn fractions in the nodes of the search tree up till level k. These variants are used in Section 6.

4 Solver Independency and Computational Effort

In this Section we will present evidence that there is a relation between the k-level average Horn fraction of an instance and the computational effort necessary to solve it for various current best DPLL techniques. In the remainder, the k-level average Horn fractions are always established using aHS. We selected three different SAT-solvers: OKsolver, kcnfs (award winning solvers in the random category of the SAT-competition in 2002 and 2003 respectively) and aHS. The OKsolver ([12]) is a DPLL-like algorithm, with reduction by failed literals and autarkies. The branching heuristic chooses a variable creating as many new two-literal clauses as possible and the first branch is chosen maximizing an approximation of the probability that the subproblem is satisfiable. kcnfs ([8]) is a complete, deterministic solver optimized for solving random k-SAT formulae. kcnfs contains a new heuristic searching variables belonging to the backbone of a CNF-formula.

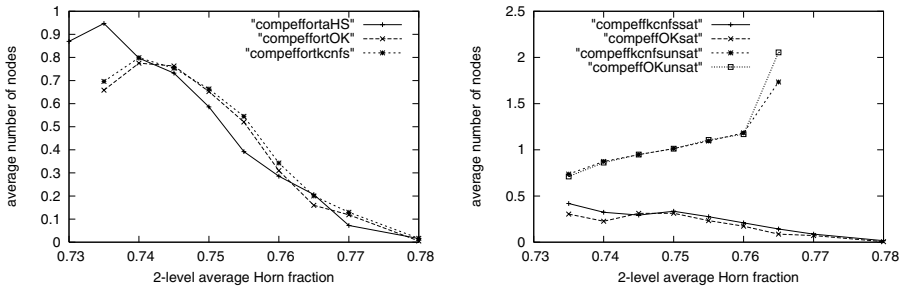

Fig. 3. 400 variables, d=4.25, OKsolver and kcnfs and 150 variables, d=4.25, aHS

In the first picture in Figure 3 we indicate on the vertical axis the average number of nodes in the search tree created by the OKsolver, kcnfs and aHS, scaled with the average number of nodes it took the solver concerned to solve the unsatisfiable instances of the indicated size. On the horizontal axis is indicated the 2-level average Horn fraction computed with aHS. The instances concerned are 2000 random 3-SAT instances with 400 variables, for the OKsolver and kcnfs, and 300 instances of 150 variables for aHS. All instances have density 4.25 and are generated by the OKgenerator. Notice that instances of 400 variables are difficult or even unsolvable in reasonable time for most solvers. The second figure in Figure 3 represents the average number of nodes for the satisfiable and unsatisfiable instances separately for the OKsolver and kcnfs.

The figure shows that there is an obvious correlation between the size of the search tree and the 2-level average Horn fraction established using aHS. From the first figure it is clear that for all of the three solvers the average size of the search tree shows an increase/decrease pattern peaking in the vicinity of the

transition point (see Section 5). From the second figure it can be concluded that satisfiable instances get easier when the 2-level average Horn fraction increases while unsatisfiable instances contrarily get more difficult to solve.

5 Size Independency of the Correlation between Horn Fraction and Satisfiability

In this Section we will examine the correlation between Horn fraction and satisfiability for random 3-SAT instances with the number of variables between 150 and 400 and the density equal to 4.25. For each size we use 1000 satisfiable and 1000 unsatisfiable instances. The instances are all close to the threshold. Hence, based on the density alone, an instance has a probability of being satisfiable being equal to 50%. For each of the instances we computed the Horn fraction of the formula in the root node (level 0), the 1-level and 2-level average Horn fraction. From these data we computed the average k-level Horn fraction for both the satisfiable (column μ_s) and unsatisfiable (column μ_u) instances, and their difference (column $\mu_s - \mu_u$). This difference is an indication for the separation between the distributions of the k-level average Horn fraction of satisfiable and unsatisfiable instances. Columns σ_s and σ_u indicate respectively the involved standard deviations. For each size and level, a spline function is fit to the discrete cumulative distribution function, and the intersection point (column 'transition') between both density functions is computed. At this point the probability of an instance being satisfiable is 0.5. The 'satCE' is the probability that a satisfiable instance is classified incorrectly by the test T that classifies an instance as satisfiable if its Horn fraction is larger than or equal to the k-level average Horn fraction in the intersection point and unsatisfiable otherwise. Analogously, the 'unsatCE' is the probability of an unsatisfiable instance being classified incorrectly similarly. The 'lefttail' is the probability that a satisfiable instance has a k-level average Horn fraction smaller than μ_u (the expected value of the k-level average Horn fraction of the unsatisfiable instances).

From Table 1, we can conclude various trends. First, for fixed k, the difference $\mu_s - \mu_u$ decreases, but the probabilities quoted show stable. Further, for fixed size, separation improves with increasing k. To illustrate this somewhat more we added $k = 3$ for the 400 variable instances.

The first picture in Figure 4 shows the probability density functions of satisfiable and unsatisfiable instances with 400 variables and density 4.25 over the 2-level average Horn fraction. The transition point and μ_u are indicated by the two vertical lines. The second picture gives for each value of the 2-level average Horn fraction the probability that an instance with the corresponding fraction is satisfiable. The vertical lines indicate mu_s and mu_u. As mentioned, for the complete data set the probability of being satisfiable is about 50%, but for example for instances with a 2-level average Horn fraction larger than 0.756 this probability of being satisfiable is 80%.

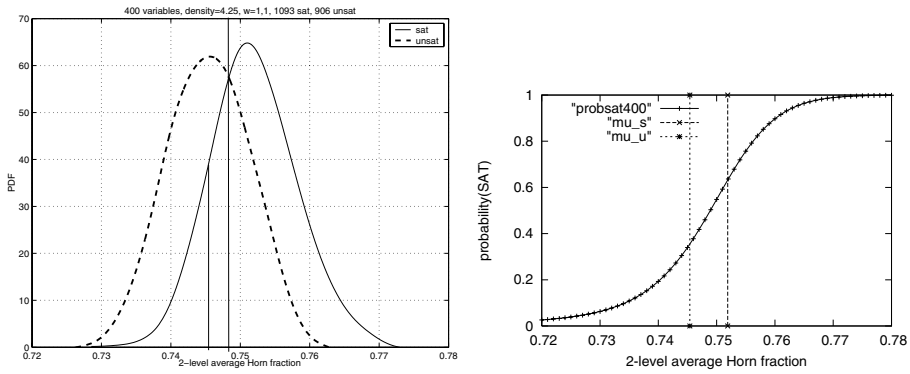

Fig. 4. 400 variables, probability density and probability of being satisfiable

6 Graph 3-Coloring

To investigate whether the phenomena above are typical for random 3-SAT we ran several experiments with graph 3-coloring instances, coded as satisfiability problems. The problem is to check whether the nodes in a graph can be colored with three colors such that no two adjacent nodes have the same color. Let N be the number of nodes in the graph and E the number of edges. To encode the graph-3-coloring problem as a satisfiability problem variables X_{nk} are introduced being true if node n has color k and false otherwise. The encoding of the graph 3-coloring problem has N clauses of length three, expressing that every node has to be assigned one of the three colors, $3N$ clauses of length two formulating for each node and pair of colors that the node does not have both colors, called unicity clauses, and finally there are $3E$ clauses of length two ensuring for every edge that both end points do not have the same color.

For this type of instances the average Horn fraction can predict quite well whether an instance is satisfiable or not, but the optimal Horn fraction in the root node does not give any hint, because the latter is equal for every instance of a certain size. By renaming all variables, all clauses of length two have two positive literals and the clauses of length three have three negative literals. Hence, the number of Horn clauses, after this renaming, is $3N + 3E$, independent of the particular instance. We cannot improve on this, because to get one of the clauses of length three Horn, we need to rename at least two variables. These two variables occur together in one of the $3N$ unicity clauses, so one of the clauses with two literals ceases to be Horn.

Besides the version of the problem described above, we also considered a variant without unicity clauses. These clauses are not strictly necessary to find a feasible coloring of the nodes, because we can always choose one the colors a node gets if it is multiple-colored. This variant turns out to be easier, and the fact that all instances have the same Horn fraction at the root node with weights equal to 1 does not hold, in absence of the unicity clauses.

#var	level	μ_s	σ_s	μ_u	σ_u	$\mu_s - \mu_u$	satCE	unsatCE	lefttail	transition
150	0	0.7102	0.0113	0.7031	0.0112	0.0071	40.0%	32.0%	25.0%	0.708
	1	0.7381	0.0164	0.7260	0.0108	0.0121	39.0%	25.0%	20.3%	0.733
	2	0.7489	0.0155	0.7349	0.0099	0.0140	37.1%	20.5%	15.1%	0.743
200	0	0.7108	0.0094	0.7040	0.0095	0.0068	30.0%	40.0%	22.8%	0.706
	1	0.7378	0.0107	0.7287	0.0094	0.0091	32.0%	32.3%	18.8%	0.7344
	2	0.7489	0.0102	0.7383	0.0085	0.0106	33.5%	23.1%	12.5%	0.7446
250	0	0.7102	0.0086	0.7044	0.0082	0.0058	39.0%	34.0%	25.2%	0.7078
	1	0.7378	0.0092	0.7302	0.0084	0.0076	36.6%	29.7%	20.4%	0.7347
	2	0.7493	0.0085	0.7409	0.0074	0.0084	33.8%	25.5%	16.1%	0.7458
300	0	0.7106	0.0078	0.7050	0.0077	0.0056	36.3%	35.0%	23.3%	0.7079
	1	0.7385	0.0083	0.7315	0.0081	0.0070	32.6%	34.4%	19.5%	0.7557
	2	0.7506	0.0078	0.7428	0.0072	0.0078	32.7%	27.4%	14.4%	0.7471
350	0	0.7105	0.0069	0.7051	0.0067	0.0054	35.9%	33.3%	22.0%	0.708
	1	0.7388	0.0074	0.7321	0.0073	0.0067	31.4%	33.6%	17.6%	0.735
	2	0.7512	0.0072	0.7438	0.0067	0.0074	31.7%	27.4%	15.3%	0.7478
400	0	0.7105	0.0065	0.7059	0.0063	0.0046	37.2%	34.6%	23.9%	0.7084
	1	0.7390	0.0069	0.7333	0.0066	0.0057	35.4%	31.8%	20.0%	0.7364
	2	0.7519	0.0063	0.7454	0.0059	0.0065	31.6%	27.5%	13.8%	0.7489
	3	0.7585	0.0061	0.7517	0.0056	0.0068	27.5%	31.1%	11.9%	0.755

Table 1. Random 3-SAT from 150 to 400 variables

We experimented with the weight of the clauses. It turned out that the weight has a significant influence on both computation times and the shape of the distribution. This influence turned out to be more prominent for the problem without unicity clauses, but it is also present for the problem with unicity clauses. Since the CNF-formulae have roughly seven times as much 2-literal clauses as 3-literal ones, we chose the weighted variant where 'being Horn' of a 3-literal clause is considered seven times more important than 'being Horn' of a 2-literal clause. We confirmed experimentally that this weight indeed showed the best separation properties.

nodes	edges	level	μ_s	σ_s	μ_u	σ_u	$\mu_s - \mu_u$	satCE	unsatCE	lefttail	transition
100	222	1	0.8674	0.0204	0.844	0.0046	0.0234	28.8%	3.3%	11.6%	0.852
		2	0.9029	0.0248	0.8491	0.0053	0.0538	5.6%	1.3%	0.4%	0.86
150	333	1	0.8523	0.0102	0.8445	0.0039	0.0078	43.1%	15.6%	17.5%	0.848
		2	0.8662	0.015	0.8487	0.0041	0.0175	18.9%	7.2%	3.5%	0.854
		3	0.8882	0.0253	0.8532	0.0042	0.035	10.3%	1.8%	0.9%	0.861
150	339	1	0.8503	0.0109	0.8431	0.004	0.0072	59.1%	6.5%	25.5%	0.849
		2	0.8649	0.018	0.8483	0.0041	0.0166	29.9%	7.9%	6.4%	0.854
		3	0.8813	0.0261	0.8505	0.0043	0.0308	11.0%	5.4%	0.6%	0.857

Table 2. Graph-3-coloring with respect to the k-level average Horn fraction

In the next experiment we generated 500 instances with 100 nodes and 222 edges. For this density the probability for an instance being satisfiable is approximately 50%. We generated also 600 instances with 150 nodes with 333 edges, i.e. with the same density as the instances with 100 nodes, but also with 339 edges, seemingly closer to the threshold density for this size. The results are given in the next table, which illustrates that the correlation between k-level average Horn fraction and satisfiability is much more prominent than for random 3-SAT instances. Considering the k-level max-Horn fraction in the search tree at level k, we even get better results, as is shown in Table 3.

nodes	edges	level	μ_s	σ_s	μ_u	σ_u	$\mu_s - \mu_u$	satCE	unsatCE	lefttail	transition
150	333	1	0.8985	0.0536	0.8555	0.0075	0.043	42.2%	10.2%	15.4%	0.864
		2	0.9809	0.0353	0.865	0.0083	0.1159	7.9%	0.6%	1.6%	0.887
		3	0.9989	0.0085	0.869	0.0084	0.1299	0.2%	0.0%	0.2%	0.9
150	339	1	0.8931	0.0542	0.8549	0.0077	0.0382	54.5%	9.0%	26.1%	0.865
		2	0.9688	0.0492	0.8635	0.0079	0.1053	17.1%	1.1%	3.8%	0.881
		3	0.9967	0.0125	0.8682	0.0084	0.1285	0.6%	0.3%	0.0%	0.892

Table 3. Graph-3-coloring with respect to k-level max-Horn fraction

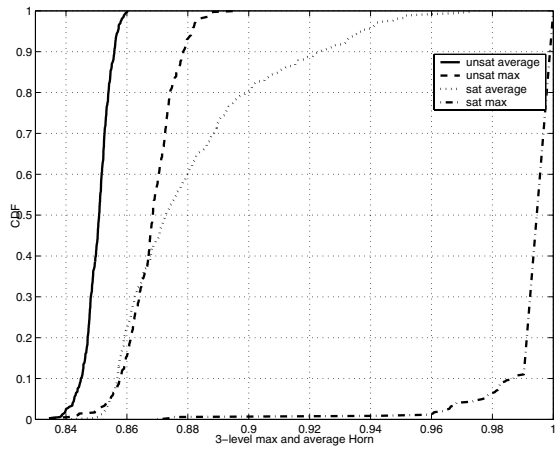

Fig. 5. Graph-coloring, 150 nodes, 339 edges, cumulative distribution functions

In Figure 5 the cumulative distribution functions for the satisfiable and unsatisfiable instances with 150 nodes and 339 edges are shown, both with respect to the 3-level average Horn fraction and the 3-level max-Horn fraction. On the vertical axis are the distribution functions $F(x)$, indicating the probability that an instance has a 3-level average Horn or max-horn fraction smaller than

x. These distributions illustrate that the correlation between satisfiability and 3-level Horn fraction is more prominent (almost complete separation over the sample) for max-Horn than for average Horn. The phenomena show much more convincing than for random 3-SAT.

7 The Algorithm aHS

In this section we describe an algorithm, which is typically designed for the purpose of fast convergence of the average Horn fraction. Before going into the details of our algorithm we first briefly summarize the main steps

1. Read the CNF-formula from the input and identify all unit clauses and propagate the corresponding values through the formula until no unit clauses remain. The resulting formula is $\mathcal{F}$.
2. Apply to $\mathcal{F}$ single look-ahead unit propagation in order to simplify the formula as much as possible. If the formula is proved unsatisfiable or a satisfying assignment is found: STOP. Otherwise the resulting formula is $\mathcal{F}_1$
3. Greedy suboptimize the Horn fraction to obtain $\mathcal{F}_2$
4. Transform $\mathcal{F}_2$ to a linear program and solve this linear program to get a solution vector (splitting point) ω .
5. Define subproblems of $\mathcal{F}_2$ using a decomposition of the search space based on ω
6. Go into breadth-first recursion.

Each of the relevant steps is described in the corresponding subsection.

7.1 Propagating the Unit Clauses

For each of the unit clauses in the input, the variable is fixed to the appropriate truth-assignment. In this step, formulae with only Horn clauses are already solved.

7.2 Single Look-Ahead Unit Propagation

Single-lookahead unit resolution is described in [9]. In full single look-ahead unit resolution, every variable is set temporarily to true. If this does lead to a contradiction the variable can be fixed to false for the current node and all of its children. This value is propagated. If this propagation does lead to a contradiction we can conclude that the formula is unsatisfiable and backtrack. This loop is continued until none of the variables can be concluded to be false in a full loop over the variables. After this loop, an analogous loop for fixing the variables to false is made. We repeat the process until none of the variables could be fixed during the last loop over the positive and negative assignments.

7.3 Greedy Horn Heuristic

Below, we resume a greedy local search heuristic to increase the fraction of Horn clauses in the formula. After computing the (weighted) Horn fraction, for every variable X_i, the resulting increase in (weighted) Horn fraction is computed in case the variable would be flipped. If the largest increase is positive, the corresponding variable is flipped. If the largest increase is 0, flip, if possible, the variable with largest positive increase in the number of positive literals.

Notice that the complexity of this heuristic is $\mathcal{O}(nm)$ in which n is the number of variables and m is the number of clauses.

7.4 Linear Program

Let n be the number of variables and m the number of clauses of a satisfiability instance. As is common knowledge, an instance of the satisfiability problem can be described by a binary integer linear program. Let A be a $m \times n$ 0-1 matrix, with $a_{ij} = 1$ if variable X_j is contained in clause i, $a_{ij} = -1$ if the negation of variable X_j is contained in clause i and 0 otherwise. Let x be a 0-1 vector with $x_j = 1$ if variable j is set to true and 0 otherwise. For the vector b of right hand sides, b_i equals 1 minus the number of negated variables in clause i. With this notation a satisfying assignment corresponds to a lattice point of the polytope

$$\{Ax \geq b, x \in \{0,1\}^n\}$$

To illustrate this formulation the clause $X_p \vee \neg X_q \vee X_r$ gives the linear constraint $x_p - x_q + x_r \geq 0$.

As objective function $\sum x_j$ is used. The constraints $x \in \{0,1\}^n$ are replaced by $0 \leq x \leq 1$.

7.5 Search Tree Design

The search tree design described in this Subsection is based on [14]. The authors of this paper show that integer programs having only inequalities satisfying the so-called generalized Horn condition

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \geq \alpha,$$

with at most one α_i negative, can be solved effectively for feasibility using an homotopy procedure and a certain subdivision of Euclidean space. This subdivision now is precisely what we use in our splitting of the search space.

The region G_i is the cone spanned by the vectors from $\{-e_0, -e_1, \ldots, -e_n\} \backslash \{-e_i\}$, in which e_j is the unit vector with a 1 in the j-th row, e_0 is the vector with one in all components and e_i is one of the vectors from $\{e_0, e_1, \ldots, e_n\}$. The regions that follow from this splitting are described below.

The first region is

$$G_0 = \{x_i \leq \omega_i, \text{ for all } i\}.$$

Since only lattice points are involved, we may replace $x_i \leq \omega_i$ by $x_i \leq \lfloor \omega_i \rfloor$. The clauses that correspond to these linear inequalities are $\neg X_i$ for all i for which $\omega_i < 1$.

Region G_1 is defined by

$$x_1 \geq \omega_1, \tag{1}$$
$$x_1 - x_2 \geq \lceil \omega_1 - \omega_2 \rceil, \tag{2}$$
$$\dots$$
$$x_1 - x_n \geq \lceil \omega_1 - \omega_n \rceil. \tag{3}$$

$x_1 \geq \omega_1$ may be replaced by $x_1 \geq \lfloor \omega_1 \rfloor + 1$ because $x_1 = \omega_1$ is already included in G_0. Note that this region is empty if $\omega_1 = 1$. Otherwise, X_1 is added to the CNF-formula for this region. Note that or region 1 is empty or $x_1 = 1$, i.e. X_1 is true. In case the region is not empty, we use the fact that $x_1 = 1$ to get $x_k = 0$ if $\omega_1 - \omega_k \in (0, 1]$.

In general, region G_i is defined as

$$x_i \geq \lfloor \omega_i \rfloor + 1, \tag{4}$$
$$x_i - x_k \geq \lfloor \omega_i - \omega_k \rfloor + 1,\ k < i, \tag{5}$$
$$\dots$$
$$x_i - x_k \geq \lceil \omega_i - \omega_k \rceil,\ k > i. \tag{6}$$

Clearly, the region is empty if $\omega_i = 1$. Otherwise, we have $x_i = 1$ or X_i is true. For the case $k > i$ we have $\neg X_k$ if $\lceil \omega_i - \omega_k \rceil = 1$. For the case $k < i$, we have $\neg X_k$ if $\lfloor \omega_i - \omega_k \rfloor = 0$ and we can conclude that the region is empty if it is 1.

Thus, finally, only sets of unit literal clauses are added to the CNF-formula for each of the regions, resulting in a number of at most $n + 1$ child nodes.

Example To illustrate this approach, take for example, $\omega = (1, \frac{3}{4}, \frac{3}{4}, 0)$. In this case the regions are

$$G_0 = \{x_1 \leq 1, x_2 = x_3 = x_4 = 0\} \tag{7}$$
$$G_1 = \emptyset \tag{8}$$
$$G_2 = \{x_2 = 1, x_4 = 0, x_1, x_3 \text{ free}\} \tag{9}$$
$$G_3 = \{x_3 = 1, x_2 = x_4 = 0, x_1 \text{ free}\} \tag{10}$$
$$G_4 = \{x_4 = 1, x_1, x_2, x_3 \text{ free}\} \tag{11}$$

To see that G_1 is empty note that $\omega_1 = 1$. $x_1 \geq \lfloor \omega_1 \rfloor + 1$ implies $x_1 \geq 2$. For example, for G_2, $x_4 = 0$ follows from $\lceil \omega_2 - \omega_4 \rceil = 1$, hence $x_2 - x_4 \geq 1$. Combining with $x_2 = 1$ this gives $x_4 = 0$. Note that G_4 is much larger than the other regions.

In the two dimensional space the splitting of the search space graphically look as follows

Notice that ω is determined by the only two inequalities not satisfying the generalized Horn condition. This is the main intuition behind the splitting: try to eliminate many non-Horn clauses by splitting the search space at ω. Notice

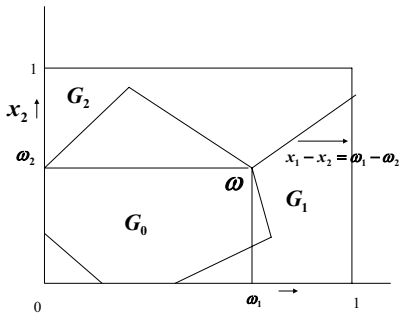

Fig. 6. Splitting the search space into regions

that in higher dimensions ω will be determined by more inequalities being not generalized Horn.

8 Conclusions and Future Research

In this paper, we presented a first experimental evidence that the class of fixed-density random 3-SAT problems is again subject to threshold phenomena with respect to the newly introduced parameter k-level average Horn fraction. We also established similar (even stronger) phenomena for random graph-3-coloring. This feature calls for more extensive experiments to establish more accurate separations, with more accurate statistical methods. Also, theoretical back-up is necessary (sofar, the only back-up comes from [14]). However, the algorithm aHS used to compute the Horn fractions contains elements (greedy optimization, linear programming relaxation) which makes it almost impossible to analyze. Theoretical back-up must come either from a clear mathematical statement showing separability possibilities for an $\mathcal{NP}$-hard problem class or from a much more transparent notion of 'average Horn fraction'. Whether the observed phenomena can be made useful in order to predict satisfiability of a certain 'difficult' and structured CNF-formula of particular interest is under current research.

References

[1] B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8:121–123, 1979.

[2] G. Biroli, S. Cocco, and R. Monasson. Phase transitions and complexity in computer science: an overview of the statistical physics approach to the random satisfiability problem. *Physica A*, 306:381–394, 2002.

[3] E. Boros. Maximum renable horn sub-CNFs. *Discrete Applied Mathematics*, 96-97:29–40, 1999.

[4] V. Chandru and J.N. Hooker. Detecting embedded horn structure in propositional logic. *Information Processing Letters*, 42:109–111, 1992.
[5] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd annual ACM symposium on the Theory of Computing*, pages 151–158, 1971.
[6] Y. Crama, O. Ekin, and P.L. Hammer. Variable and term removal from boolean formulae. *Discrete Applied Mathematics*, 75:217–230, 1997.
[7] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.
[8] O. Dubois and G. Dequen. A backbone-search heuristic for efficient solving of hard 3-sat formulae. IJCAI-01, 2001.
[9] J. Franco. *Relative size of certain polynomial time solvable subclasses of satisfiability*, volume 35 of *DIMACS Series Discrete Math.Theoret.Computer Science*, pages 211–233. Amer.Math.Soc., Piscataway, NJ, 1997.
[10] E. Friedgut. Sharp thresholds of graph proporties and the k-sat problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, 1999.
[11] O. Kullmann. First report on an adaptive density based branching rule for DLL-like SAT-solvers, using a database of mixed random conjunctive normal forms created using the advanced encryption standard (aes). Technical Report CSR 19-2002, University of Wales Swansea, Computer Science Report Series, 2002.
[12] O Kullmann. Investigating the behaviour of a sat solver on random formulas. Submitted to Annals of Mathematics and Artificial Intelligence, 2002.
[13] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400:133–137, July 1999.
[14] H. Van Maaren and C. Dang. Simplicial pivoting algorithms for a tractable class of integer programs. *Journal of Combinatorial Optimization*, 6:133–142, 2002.