Hidden threshold phenomena for fixed-density SAT-formulae

Hans van Maaren
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
Email: H.vanMaaren@its.tudelft.nl, L.vanNorden@its.tudelft.nl

Abstract. Experimental evidence is presented that hidden threshold phenomena exist for fixed density random 3-SAT formulae. At such fixed densities the (sub-optimal) 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 density and size on the observed phenomena.
Keywords: satisfiability, Horn, thresholds, 3-SAT

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 (Friedgut (1999)), 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. Monasson et al. (1999) and Biroli et al. (2002) 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 show experimental evidence that the class of random 3-SAT problems with a fixed density again is subject to similar phenomena with respect

to a different parameter: the Horn fraction. Horn 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 the coming full version of this paper. 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 points. 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 the experimental tests are generated by the OK-generator (Kullmann (2002a)).

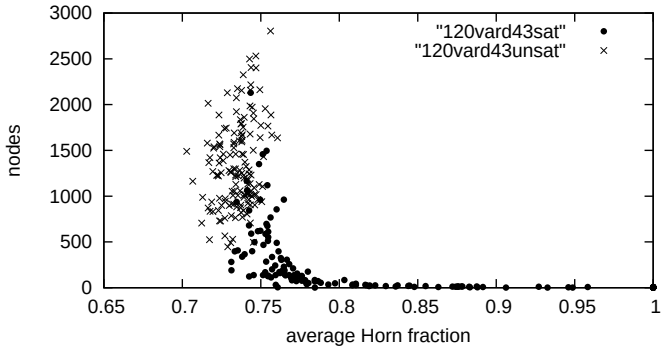

Fig. 1. 120 variables, d=4.3

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 its average Horn fraction?
b. Is this 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.

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. Cook (1971) 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). Cook (1971) 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 HF 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 of Dowling and Gallier (1984). Renaming, or flipping, a variable 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 (Crama et al. (1997)). 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 Aspvall et al. (1979). Chandru and Hooker (1992) 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 by applying routine $\mathcal{R}$.*

Boros (1999) 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 largest instances with density 4.25 for which the optimal Horn fraction after renaming can be computed in reasonable time, the sub-optimal Horn fraction after renaming computed by our heuristic is only about 1.9% from the optimal Horn fraction after renaming.

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.*

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 the full version of the paper, 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 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. On the vertical axis is the number of nodes it took to solve the instance. 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 Boros (1999) for a set of random 3-SAT instances with 40 variables. Plotting this optimal Horn fraction against

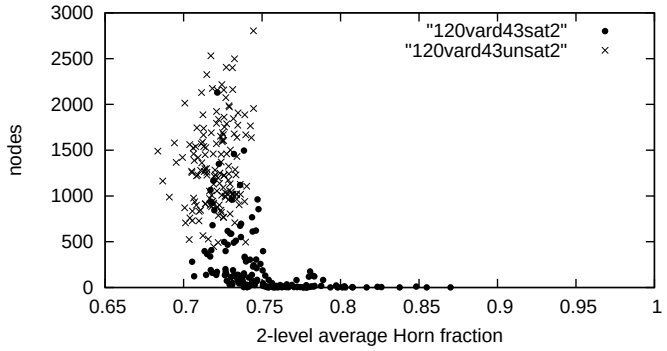

Fig. 2. 120 variables, d=4.3

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).

4 Solver independency

In this section we will gather evidence that the correlations between average Horn fraction, running time and satisfiability are features that are not specific for aHS, but generalize, although to a slightly weaker extend, to other type of algorithms. In the remainder, 2-level average Horn fractions are always established using aHS. We selected two other SAT-solvers with a completely different architecture: the OKsolver and limmat. The OKsolver (Kullmann (2002b)) 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. Limmat (Biere (2002)) is a zchaff-like solver with a very efficient implementation of unit propagation and an optimized ordering of decision variables.

In the first figure in Figure 3 we indicate on the vertical axis the number of nodes in the search tree created by the OKsolver. On the horizontal axis is indicated the 3-level average Horn fraction computed with aHS. The instances concerned are random 3-SAT instances with 400 variables and density 4.25, generated by the OKgenerator. Instances of this size are difficult or even unsolvable in reasonable time for most solvers. Because the instances are much larger than the ones used in Figures 1 and 2 we chose the 3-level average Horn fraction in this case. The second figure in Figure 3 shows a similar type of distribution when using limmat. In this case, we used instances with 150 variables and density 4.25,

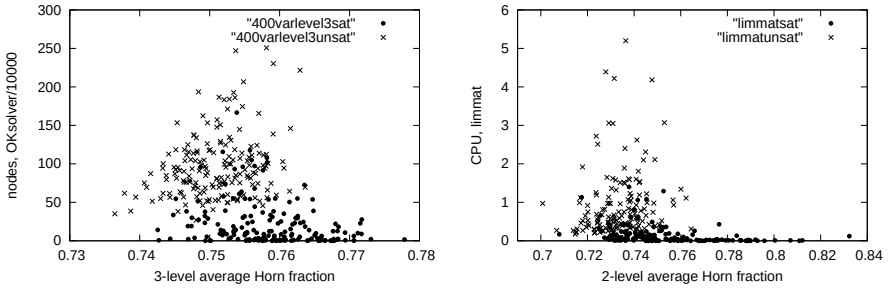

Fig. 3. 400 variables, d=4.25, OKsolver and 150 variables, d=4.25, limmat

because the instances with 400 variables would take too much time to solve with the version of limmat we used.

From both figures it is clear that a similar distribution of the instances appears as in Figure 1 and 2. The correlation between Horn fraction and satisfiability is of course independent of the solver used to solve the instance, because the average Horn fraction is always computed by aHS, and the satisfiability of an instance is obviously independent of the solver. It is clear from the figures that also with respect to the OKsolver and limmat, a weak linear separation between satisfiable and unsatisfiable instances is possible, although to a weaker extent than using aHS. We might conclude that the parameter k-level average Horn fraction is correlated with computation times with respect to different solvers. Correlation with the computation times with respect to aHS however seems to be more prominent.

5 Size independency

The phenomena discussed in the introduction seem to scale to larger random 3-SAT instances, which will be shown in this Section. We use random 3-SAT instances with 100, 120, 150 and 400 variables and density 4.25. For each of these sizes we generated 150 satisfiable and 150 unsatisfiable instances.

5.1 Correlation between Horn fraction and satisfiability

We will examine the correlation between Horn fraction and satisfiability with observed probability density functions over a sample of random 3-SAT instances of various sizes. These probability densities are shown in Figure 4.

The vertical axis gives the observed probability that an instance is satisfiable among a sample in which the number of satisfiable instances is equal to the number of unsatisfiable ones. The horizontal axis gives the 2-level average Horn fractions. The curves of the probability densities seem to move a bit to the right with increasing size, but they show the same shape. Note that the deviating value of the 150-variable-curve at 0.71 is caused by small-number effects. There were

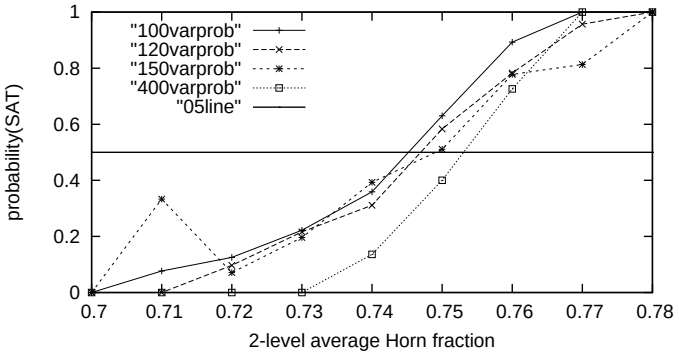

Fig. 4. 100, 120, 150 and 400 variables, probability densities

only three instances in the sample with 2-level average Horn fraction between 0.7 and 0.71, one of which was unsatisfiable. Figure 4 gives reasons to believe that for density 4.25, the 2-level average Horn fraction parameter shows threshold properties in the vicinity of 0.75.

5.2 Linear separation

A different graphical representation shows the weak linear separation mentioned in the introduction. Figure 5 concerns the above mentioned random 3-SAT instances with 100, 120 and 150 variables and density 4.25 and is also based on the 2-level average Horn fraction.

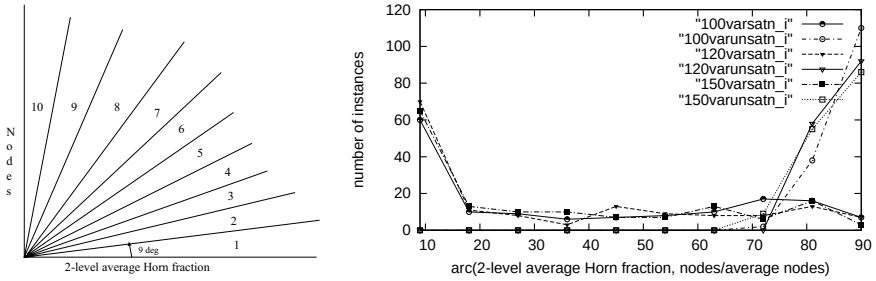

Fig. 5. Creating an arc-bin-plot and 100, 120 and 150 variables, d=4.25, arc-bin-plot, origin 0.69

The idea is to split the dataset by rays with a certain arc with the horizontal axis, as shown in Figure 5. For this purpose, we chose an origin for the rays on the horizontal axis at the largest integer 2-level average Horn percentage

that is smaller than the 2-level average Horn percentages of the instances in the dataset. For our test set, this is 69%. From this point we draw a ray with an arc of every multiple of 9° and count the number of satisfiable and unsatisfiable instances between each two consecutive rays. On the horizontal axis of Figure 5 are the arcs. For example, 9 stands for all instances with an arc smaller than 9°. On the vertical axis of Figure 5 is given the number of instances for which the coordinates fall in the corresponding region. For purposes of representation, we have scaled the number of nodes of each instance with n variables by dividing by the average number of nodes it took to solve the unsatisfiable instances with n variables. The 'cross-over' points, where the curves of satisfiable and unsatisfiable instances cross, are very close for the three sizes: 71°, 72° and 75°. Choosing 72°, it is easily verified that the estimated separation line for the complete dataset is

$$N = 9.93H - 6.85, \tag{1}$$

where N is the scaled number of nodes and H is the 2-level average Horn fraction. Over the data set involved, 92% of the instances are correctly classified by this formula. Hence, we can conclude that we can establish a weak linear separation between satisfiable and unsatisfiable instances, that is independent of the size of the instances, within a certain range.

To test the quality of this separation for other instances, we generated for 100, 120 and 150 variables 250 satisfiable and 250 unsatisfiable instances. 93% of these 1500 instances is classified correctly by (1), which is comparable with the results on the data set that is used to compute the line.

5.3 Correlation between Horn fraction and size of search tree

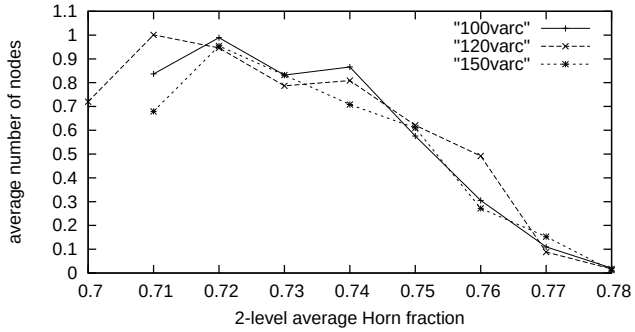

Fig. 6. 100, 120 and 150 variables, d=4.25, scaled average number of nodes per Horn fraction

Besides the correlations mentioned above there is also an observed correlation between the average Horn fraction and the average number of nodes needed to solve an instance by aHS. Figure 6 demonstrates this observed correlation. It gives a graphical representation of the correlation between the 2-level average Horn fraction of an instance and the scaled number of nodes it took aHS to

solve it. We observe that instances with a 2-level average Horn fraction larger than 0.72 are easier with respect to aHS as their 2-level average Horn fractions increase. The shape and maxima of the graphs for the instances of different size are similar.

6 Density independency

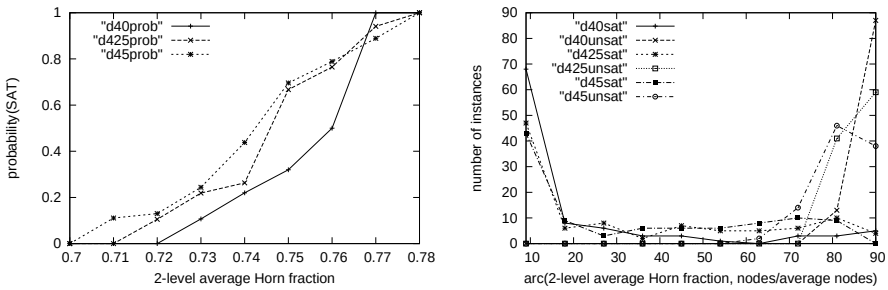

Fig. 7. 120 variables, d=4.0, 4.25 and 4.5

The separation properties do not seem to be specific for the densities considered above but seem to be a characteristic that extends through different densities. We consider three datasets of random 3-SAT instances with different densities, 4.0, 4.25, and 4.5 with 120 variables. For each of the densities we generated 100 satisfiable and 100 unsatisfiable instances.

The correlation between 2-level average Horn fraction and satisfiability is again illustrated by probability density functions just as in Figure 4. In the first picture of Figure 7 the graph belonging to density 4.0 differs somewhat from the other two, but this might be caused by the fact that unsatisfiable instances are very scarce for this density.

The linear separation is shown in an arc-bin-plot, designed as in Figure 5, with origin 0.69. In the second picture of Figure 7 the three graphs are very similar. For increasing densities, the ray constituting the best separation seems to become steeper.

7 Conclusion

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. This feature calls for more extensive experiments to establish more accurate separations, with more accurate statistical methods. Besides, theoretical arguments must be sought to understand the nature of the observed features. In the full version of this paper it is shown that these features extend to other classes of CNF-formulae as well. Another weak predictor for satisfiability of random 3-SAT instances is discussed in Bennaceur and Li (2002).

8 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 of an instance can predict quite well whether the 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 clauses that is Horn 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.

The graphs used for the tests are at threshold density and have 80 nodes and 176 edges. 235 satisfiable and 235 unsatisfiable instances were generated with a random graph generator.

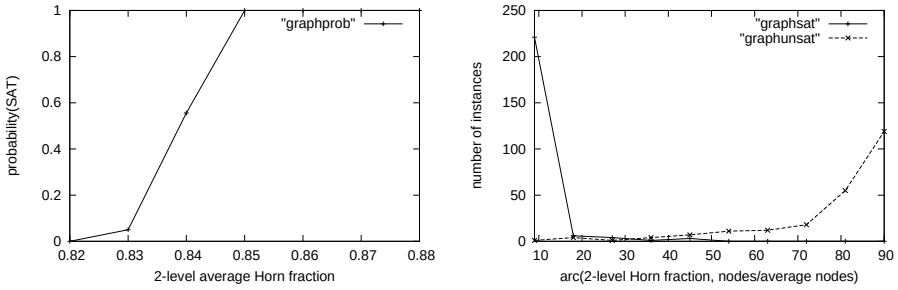

Fig. 8. graph coloring, 80 nodes, 176 edges

For these pictures, we used the weighted variant of the 2-level average Horn fraction. The weight of the clauses with two literals is taken equal to one, and the clauses with three literals have weight four. The intuition behind the fact of giving the longer clauses a larger weight is that it is more important that they are Horn because they generally disappear from the formula deeper in the search tree.

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.

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 is more prominent for the problem without unicity clauses, but it also present for the problem with unicity clauses. Consider Figure 9. On the vertical axis is indicated the 2-level Horn fraction. On the horizontal axis is number of instances in the 2-level Horn fraction interval divided by the total number of instances. The figure concerns graph coloring instances without the unicity clauses with 80 nodes and 176 edges. The drawn lines represents the satisfiable instances and the dashed line the unsatisfiable instances. The first picture concerns the case with the weight of the clauses of length 3 equal to 1. For the second picture this weight is 7.

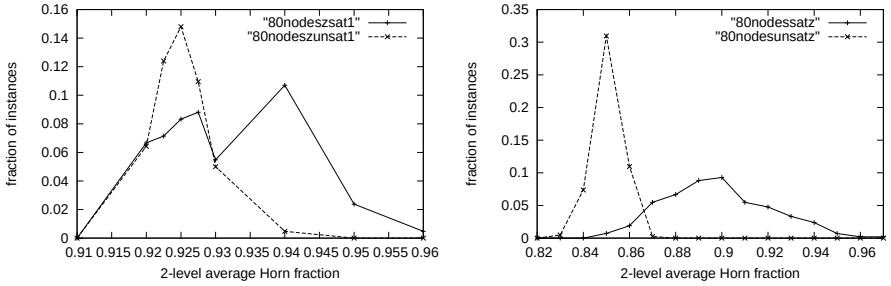

Fig. 9. graph coloring, 80 nodes, 176 edges, without unicity clauses, weights 1 and 7

It is obvious that the separation between satisfiable and unsatisfiable instances is much better for the weight 7.

As a measure of separation we introduce the weighted number of so-called unsure instances. An "unsure" instance is an instance that has a 2-level Horn fraction that falls in an interval with observed probability of being satisfiable between 0 and 1. The weighted number of unsure instances is the sum of the products of the minimum of p and $1-p$, with p the observed probability of being satisfiable, and the number of instances in the interval. This measure takes large values for small weights of the three-literal clauses and decreasing when this weights increases, till it reaches a point after which there is not much difference in the weighted number of unsure instances. This point is about 7 for the above mentioned instances.

Experiments with varying weights on random 3-SAT instances showed that the separation of satisfiable and unsatisfiable instances is hardly influenced by the weights.

8.1 Correlation between Horn fraction and satisfiability

In Figure 8, the first picture is the probability density function. This illustrates that the correlation between Horn fraction and satisfiability is even stronger for graph 3-coloring than for random 3-SAT instances. This is seen from the fact that the density function is much steeper.

8.2 Linear separation

The second picture in Figure 8 is again an arc-bin-plot and gives for each arc of 9° the number of instances that falls in the corresponding region. The rays are drawn from origin 0.82 in this case. It illustrates that also the linear separation between satisfiable and unsatisfiable instances is for these graph-3-coloring instances stronger than for the random 3-SAT instances. All but one of the satisfiable instances are in the first region, whilst none of the unsatisfiable ones is in that region.

9 An algorithm for computing fast converging average Horn fractions

In this section we describe an algorithm, which is typically designed for the purpose of fast convergence of the average Horn fraction. Our algorithm aims to provide a fast to compute approximation 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, or splitting point, ω .
5. Define subproblems of $\mathcal{F}_2$ using a decomposition of the search space based on ω
6. Go into depth-first recursion.

It must be said that our algorithm is not fast, partly because solving linear programming relaxations which take about 20 to 40% of the computation time depending on the problem size. It was not our goal to design a fast satisfiability solver. We therefore did not try to write a code optimized with respect to required CPU-time and memory use.

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

9.1 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.

9.2 Single look-ahead unit propagation

Single-lookahead unit resolution is described in Franco (1997).

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.

9.3 Greedy Horn heuristic

Because unit propagation is an important aspect of our algorithm, it is expected that the performance will be better if the fraction of Horn clauses is larger. Below, we resume a greedy local search heuristic to increase the fraction of Horn clauses in the formula. The length of a clause is the number of free variables in it. Let H_l be the number of Horn clauses of length l.

The heuristic works as follows

1. Check which of the clauses are satisfied, and compute the number of free variables in every clause that is not satisfied.
2. Compute H_l, for every l considering only clauses that are not satisfied.
3. For every free variable i, compute the number of Horn clauses of every length l in case this variable would be flipped. This equals the previous number of Horn clauses of length l - minus the number of non-satisfied clauses of length l with one negative literal and containing X_i plus the number of non-satisfied clauses of length l with $\neg X_i$ that have two negative literals. Compute the (weighted) number of Horn clauses.
4. If the largest increase in the (weighted) number of Horn clauses is positive we flip the corresponding variable and update the number of negative occurrences, go to step 3, Otherwise, go to step 5.
5. Find the variable among the variables, which give an equal number of Horn clauses when they are flipped, that has the largest increase in number of positive literals. If this is positive, flip this variable and go to step 3, otherwise, STOP.

The complexity of this heuristic is $\mathcal{O}(nm)$ in which n is the number of variables and m is the number of clauses. In each iteration for every free variable, maximal n, the increase in Horn clauses and the increase in the number of positive literals are computed. This can be done in $\mathcal{O}(m)$ over all variables. In each iteration at least one clause gets Horn or there is at most one positive literal more. A lower bound on the number of iterations is hence m plus $n + m$.

9.4 Linear program

Let n be the number of variables and m the number of clauses of a satisfiability instance. 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 $x_p - x_q + x_r \geq 0$.

It is not obvious which objective function is most suitable for computing the splitting points. For the moment, we chose the objective coefficient of each of the variables to be 1. This can be refined by choosing objective coefficients that take into account characteristics of the formula at hand, like the number of times a variables occurs as a positive literal.

9.5 Search tree design

The search tree design described in this Subsection is based on Van Maaren and Dang (2002). Most constraints that belong to non-Horn clauses are expected to be active close to the linear programming solution. Hence, it seems favorable to split the search space at the linear programming solution $\omega = (\omega_1, \ldots, \omega_n)$ to get less-constrained, and simpler, subproblems that are closer to Horn problems. The search space is split into $n + 1$, possible empty, disjoint parts. Obviously, none of these regions contains the linear programming solution. Our splitting method aims at getting the formulae in the nodes as close as possible to Horn formulae. The linear programming solution is the vertex of the polytope, describing the feasible region of the linear program, that is closest to the vector with all ones. By the splitting strategy we present below, the larger regions are only constrained by Horn clauses and only the smaller regions are bounded by linear inequalities from clauses that are not Horn.

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. Besides, we mention the clauses the CNF-formula has to satisfy in each of the regions, following from their definitions.

The first, and often smallest region, G_0 is

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

Only lattice points can be feasible solutions to the satisfiability problem. Hence, 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 $\omega_i < 1$.

Elements x in the second region G_1 satisfy

$$x_1 \geq \omega_1 \tag{2}$$
$$x_1 - x_2 \geq \lceil \omega_1 - \omega_2 \rceil \tag{3}$$
$$\ldots$$
$$x_1 - x_n \geq \lceil \omega_1 - \omega_n \rceil \tag{4}$$

$x_1 \geq \omega_1$ may be replaced by $x_1 \geq \lfloor \omega_1 \rfloor + 1$ because $x_1 = \omega_1$ is already included in G_0 and we are interested in finding integer solutions of the linear program. 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

$$x_i \geq \lfloor \omega_i \rfloor + 1 \tag{5}$$
$$x_i - x_k \geq \lfloor \omega_i - \omega_k \rfloor + 1, k < i \tag{6}$$
$$\ldots$$
$$x_i - x_k \geq \lceil \omega_i - \omega_k \rceil, k > i \tag{7}$$

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, generally, only sets of unit literal clauses are added to the CNF-formula in each of the regions.

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

$$G_0 = \{x_1 \leq 1, x_2 = x_3 = x_4 = 0\} \tag{8}$$
$$G_1 = \emptyset \tag{9}$$
$$G_2 = \{x_2 = 1, x_4 = 0, x_1, x_3 \text{ free}\} \tag{10}$$
$$G_3 = \{x_3 = 1, x_2 = x_4 = 0, x_1 \text{ free}\} \tag{11}$$
$$G_4 = \{x_4 = 1, x_1, x_2, x_3 \text{ free}\} \tag{12}$$

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

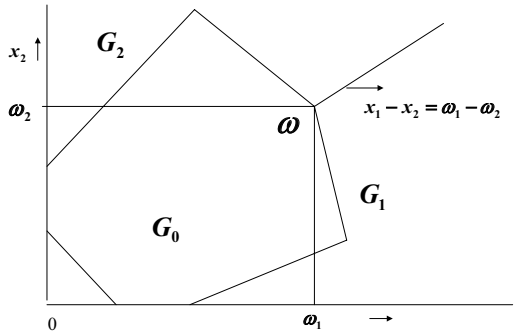

Fig. 10. Splitting the search space into regions

9.6 Branching strategy

In this section, we will present a branching strategy that will improve the convergence of the average Horn, as compared to more straightforward branching strategies. Also, with respect to computation time the branching strategy performs quite well.

Consider a node in the search tree in which we cannot find a satisfying assignment or prove unsatisfiability, even after a full loop of single look-ahead unit propagation. For every region, we check if it is not empty. For every non-empty region, we propagate the assignments of the variables that should be made in the region. For all regions, for which this does not lead to a contradiction, we compute the (weighted) fraction of Horn clauses. The nodes are ordered in non-increasing order of these fractions. Nodes with the largest (weighted) Horn fraction are considered first. The intuition behind this is that subproblems with a larger fraction of Horn clauses tend to be easier to solve. Hence, in this way we avoid to solve the most difficult subproblems for satisfying instances for which the solution is in one of the easier regions.

10 Conclusion

In this paper, we presented experimental evidence that the class of fixed-density random 3-SAT and graph-3-coloring problems are again subject to threshold

phenomena with respect to the parameter Horn fraction. There are correlations between Horn fraction and satisfiability of an instance and a correlation between the Horn fraction and the number of nodes in the search tree. Besides this there is a weak linear separation possible of the satisfiable and unsatisfiable instances based on Horn fraction and the number of nodes in the search tree. It is, as far as we know the first time that this kind of threshold within a class of problems that have the same characteristic with respect to another threshold is studied with extensive experimental tests. Another weak predictor for satisfiability of random 3-SAT instances in discussed in Bennaceur and Li (2002). However, these authors only study the differences in average value of their secondary threshold, row convexity, for satisfiable and unsatisfiable instances.

Bibliography

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

Bennaceur H. and Li C., 2002. Characterizing SAT problems with the row convexity property. In P. Van Hentenrijck, editor, *Principles and Practice of Constraint Programming - CP2002*.

Biere A., 2002. Limmat. http://www.inf.ethz.ch/personal/biere/projects/limmat/.

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

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

Chandru V. and Hooker J., 1992. Detecting embedded horn structure in propositional logic. *Information Processing Letters*, 42, 109–111.

Cook S., 1971. The complexity of theorem proving procedures. In *Proceedings of the 3rd annual ACM symposium on the Theory of Computing*, 151–158.

Crama Y., Ekin O., and Hammer P., 1997. Variable and term removal from boolean formulae. *Discrete Applied Mathematics*, 75, 217–230.

Dowling W. and Gallier J., 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1, 267–284.

Franco J., 1997. *Relative size of certain polynomial time solvable subclasses of satisfiability*, volume 35 of *DIMACS Series Discrete Math.Theoret.Computer Science*, 211–233. Amer.Math.Soc., Piscataway, NJ.

Friedgut E., 1999. Sharp thresholds of graph proporties and the k-sat problem. *Journal of the American Mathematical Society*, 12(4), 1017–1054.

Kullmann O., 2002a. 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.

Kullmann O., 2002b. Investigating the behaviour of a sat solver on random formulas. Submitted to Annals of Mathematics and Artificial Intelligence.

Monasson R., Zecchina R., Kirkpatrick S., Selman B., and Troyansky L., 1999. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400, 133–137.

Van Maaren H. and Dang C., 2002. Simplicial pivoting algorithms for a tractable class of integer programs. *Journal of Combinatorial Optimization*, 6, 133–142.