Annals of Mathematics and Artificial Intelligence (2005) 44: 157–177

Correlations between Horn fractions, satisfiability and solver performance for fixed density random 3-CNF instances

Hans van Maaren and Linda van Norden

Faculty of Electrical Engineering, Mathematics and Computer Science, Department of Software Technology, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

E-mail: {H.vanMaaren,L.vanNorden}@ewi.tudelft.nl

An enhanced concept of sub-optimal reverse Horn fraction of a CNF-formula was introduced in [18]. It was shown that this fraction is very useful in effectively (almost) separating 3-colorable random graphs with fixed node-edge density from the non-3-colorable ones. A correlation between this enhanced sub-optimal reverse Horn fraction and satisfiability of random 3-SAT instances with a fixed density was observed. In this paper, we present experimental evidence that this correlation scales to larger-sized instances and that it extends to solver performances as well, both of complete and incomplete solvers. Furthermore, we give a motivation for various phases in the algorithm aHS, establishing the enhanced sub-optimal reverse Horn fraction, and we present clear evidence for the fact that the observed correlations are stronger than correlations between satisfiability and sub-optimal MAXSAT-fractions established similarly to the enhanced sub-optimal reverse Horn fraction. The latter observation is noteworthy because the correlation between satisfiability and the optimal MAXSAT-fraction is obviously 100%.

Keywords: satisfiability, Horn, 3-SAT, density

AMS subject classification: 90C05, 03B99, 68Q01, 68W01

1. Introduction

Horn clauses are clauses with at most one positive literal. Reverse Horn clauses are clauses with at most one negative literal. The reverse Horn fraction of a CNF-formula is the number of reverse Horn clauses divided by the total number of clauses. In this paper, we present extensive empirical evidence to show that an enhanced version of the reverse Horn fraction is correlated to the probability of fixed density 3-SAT instances being satisfiable as well as to the computational effort for a variety of DPLL-solvers in solving them. This correlation even turns out to be observable for incomplete SAT-solvers.

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 problems are with high

probability satisfiable and easily shown to be satisfiable with DPLL techniques. High density problems are with high probability unsatisfiable.

Friedgut [11] showed the existence of a function $d_0(n)$ for random 3-SAT such that the phase transition is within an ϵ-neighbourhood of $d_0(n)$. Generally it is assumed that a threshold d_0 independent of n exists for random 3-SAT, but this is not proven yet. In the report underlying [14] is shown that $d_0 = 4.247$ is reasonably close to the phase transition for random 3-SAT instances with $n \leqslant 400$.

There is a vast literature on phase-transitions for the random 3-SAT problem and related problems. Monasson et al. [15] and Nishimura et al. [3] studied threshold phenomena with respect to the density parameter in random 3-SAT and the class of random $(2 + p)$-SAT problems. In a $(2 + p)$-SAT-formula a fraction p of the clauses has three literals while the others have two literals.

The density parameter is strongly correlated to satisfiability for random 3-SAT instances as is the node-edge density for 3-colorability of random graphs. However, for a sample of instances with equal density a second parameter is necessary to (weakly) separate satisfiable from unsatisfiable instances. In [18] an enhanced version of the reverse Horn fraction was introduced. This notion turned out to effectively weakly separate 3-colorable from non-3-colorable instances for a class of graphs near threshold node-edge-density. We presented experimental evidence for a similar correlation with satisfiability for random 3-SAT instances near threshold density. In this paper, we want to focus on the correlation between this enhanced reverse Horn fraction, satisfiability and computational effort. We extend the earlier results with respect to the following 'features':

- The MAXSAT-fraction is defined as the maximal number of clauses that can be satisfied. The correlation between satisfiability and MAXSAT-fraction is obviously 100%. The formula is satisfiable if and only if the MAXSAT-fraction equals 1. This would suggest that a greedy sub-optimal approximation of this fraction is correlated to the satisfiability of the instance involved. Section 3 shows that this correlation is much weaker than the correlation with the enhanced reverse Horn fraction.
- Correlation between the enhanced reverse Horn fraction and computational effort of complete SAT-solvers. In section 4 we present evidence for the fact that instances with high enhanced reverse Horn fraction are easier to solve in case they are satisfiable and harder to solve in case they are unsatisfiable, and vice versa. We selected a number of state-of-the-art solvers to illuminate this feature.
- In section 5 we give evidence for the fact that the correlation between the enhanced reverse Horn fraction and computational effort of complete solvers extends to two state-of-the-art incomplete SAT-solvers.
- The correlation between the enhanced reverse Horn fraction and satisfiability was already observed in [18]. In section 6 we present large size experiments which support the fact that this correlation scales with respect to size and is not just a small size artifact.

Section 2 gives the definitions of the relevant concepts. The algorithm aHS for calculating the enhanced reverse Horn fraction is described in section 7. The conclusions are presented in section 8.

2. Preliminaries and concepts of Horn fraction

In this section we give the definitions of the variants of reverse Horn fraction from [18], we will use in this paper.

A literal is an atomic propositional variable or its negation (indicated by $\neg$). A clause is a disjunction of literals. 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 satisfying truth assignment exists and unsatisfiable otherwise. Cook [7] proved that the satisfiability problem is $\mathcal{NP}$-complete. 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 [7] also proved that the class of 3-SAT problems is $\mathcal{NP}$-complete.

A clause is Horn if at most one literal is positive. A clause is *reverse Horn* if at most one literal is negative.

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

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 reverse Horn fraction of a CNF-formula might change. Recognizing whether a formula $\mathcal{F}$ is renamable Horn, i.e. all clauses can be made Horn by flipping, can be done in linear time, for example by the algorithm of [1]. CNF-formulae with only Horn clauses can be solved in time linear in the number of literals by the algorithm of [9]. When renaming is involved Horn and reverse Horn are equivalent concepts: by renaming all variables, all Horn clauses get reverse Horn clauses and vice versa. We chose to work with the reverse Horn fraction because of the comparability with the MAXSAT-fraction.

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

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

For a maximization problem the performance ratio of an algorithm is defined as the solution value found by the algorithm divided by the true optimal value. Boros [5] presents a rounding procedure for computing a sub-optimal Horn fraction after renaming that has a guaranteed performance ratio of $\frac{40}{67}$ on 3-SAT formulae. This procedure starts with solving a particular linear program. If the optimum value of this linear program is reasonably large, the rounding procedure uses the linear programming solution as input, otherwise the vector with all 0.5's is used to determine which variables are to be flipped. aHS, the SAT-solver from [18], reviewed in section 7, uses a deterministic greedy local search procedure to compute a sub-optimal reverse Horn fraction. This procedure is described in subsection 2.1.

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 reverse Horn fraction over the search tree (in short average reverse Horn fraction), $AH_{\text{so}}(\mathcal{F}, \mathcal{R}, T)$, is the average of the $H_{\text{subopt}}(\mathcal{F}_t, \mathcal{R})$ over all nodes t in the search tree T, with $\mathcal{F}_t$ the CNF-formula at node t of T.

To compute its average reverse Horn fraction, an instance has to be solved completely. For the next variant of reverse Horn fraction this is not necessary.

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 sub-optimal reverse Horn fraction, $AH_{\text{so}}(\mathcal{F}, \mathcal{R}, T, k)$, is the average reverse Horn fraction over the nodes in T up to depth k.

In this paper it is understood that $\mathcal{R}$ is the routine described in subsection 2.1 and that the algorithm aHS from section 7 generates the search tree T involved. This k-level average sub-optimal reverse Horn fraction is the enhanced reverse Horn fraction mentioned in the introduction. In the remainder, we write k-level reverse Horn fraction for k-level average sub-optimal reverse Horn fraction. The above definitions are sound because the routine and search tree we use are deterministic. The renaming heuristic and search tree of aHS guarantee that the k-level reverse Horn fraction can be computed in polynomial time for fixed k. As a tool for (weakly) separating satisfiable from unsatisfiable instances appropriate choices for k depend on the size and nature of the instances. Dealing with instances of the size we investigated in this paper, $k = 1$ or $k = 2$ seem to be appropriate choices to reveal the correlation mentioned.

2.1. Greedy Horn heuristic

The optimal reverse Horn fraction is approximated by a greedy local search heuristic. After computing the current reverse Horn fraction, for every variable X_i, the resulting increase in reverse 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 total number

of positive literals. If several variables yield the same increase or increase with the same number of positive literals, the variable with the smallest index number is chosen.

The algorithm for computing the optimal non-negative fraction, defined and used in section 3, is analogous. In the algorithm above, one needs to replace only reverse Horn fraction by non-negative fraction.

3. Comparing reverse Horn and non-negative fraction

Here we motivate why we study k-level reverse Horn fraction as a parameter for weakly separating satisfiable from unsatisfiable instances. One could ask whether for instance a greedy MAXSAT-approximation could serve this purpose equally well, because an optimal MAXSAT-procedure of course shows complete separation. First, we present some arguments for studying MAXSAT-approximation for separation. Then, we present convincing experimental evidence which shows that the correlation between k-level reverse Horn fraction and satisfiability is stronger than the correlation between a greedy approximation of the MAXSAT-fraction and satisfiability. In order to obtain an impression of the influence of the density parameter, we vary this density over the test set. Finally, we illustrate that the correlation of k-level reverse Horn fraction and satisfiability gets stronger with increasing k.

3.1. Optimal reverse Horn fraction and MAXSAT-fraction

In this subsection, we present the concept of non-negative fraction and illustrate why this might be an appropriate alternative for the reverse Horn fraction. A clause is *non-negative* if it contains at least one positive literal. The *non-negative fraction* of a CNF-formula is the number of non-negative clauses divided by the total number of clauses. The optimal non-negative fraction is the non-negative fraction after applying an optimal renaming procedure. The k-level non-negative fraction is defined analogous to the k-level reverse Horn fraction (see definition 5).

The MAXSAT-fraction is equal to the optimal non-negative fraction. In a formula with a non-negative fraction p, one can satisfy p clauses by setting all variables to true. The other way around, a formula with MAXSAT-fraction p can be turned into a formula with non-negative fraction p by renaming all variables that are false in the MAXSAT-solution. For an exact 2SAT-formula the non-negative fraction and the reverse Horn fraction are equal by definition. Because finding the optimal reverse Horn fraction and the MAXSAT-fraction are the same for 2SAT, and MAX2SAT is $\mathcal{NP}$-hard [12], the known fact that finding the optimal reverse Horn fraction is $\mathcal{NP}$-hard follows (see for a different proof of this fact [8]).

In reverse Horn clauses at most one literal is negative. In a non-negative clause at least one literal must be positive. For strict 2-CNF formula reverse Horn and non-negativeness are equivalent. Every reverse Horn clause with two or more literals is a non-negative clause, but for clauses with three or more literals the reverse is not true. Hence, non-negativeness is a less stringent constraint for clauses with at least three literals.

Because the non-negative fraction and the reverse Horn fraction are equal for 2-SAT and because the optimal non-negative fraction is, in contrary to optimal reverse Horn fraction, directly related to satisfiability, the k-level non-negative fraction might be expected to weakly separate at least as good as the k-level reverse Horn fraction. However, our experiments show that the k-level reverse Horn fraction performs better in this respect. One explanation might be the fact that a reverse Horn clause stays reverse Horn when shortened in a search tree, while this is not true for non-negative clauses.

The greedy algorithms used to compute sub-optimal reverse Horn fraction and sub-optimal non-negative fraction are discussed in section 2.1. In fact, both algorithms only differ in the notion that is maximized. Because the algorithms are essentially the same, the difference in approximation algorithm does not play a role in the difference of separation possibilities.

To get an impression on the quality of the approximations, we computed our sub-optimal reverse Horn fraction and the sub-optimal non-negative fraction for 130 instances with 40 variables and density 4.25. For these instances we computed the optimal reverse Horn fraction with the integer linear program suggested in [5] and the MAXSAT-fraction with a complete MAXSAT-solver [4]. The sub-optimal reverse Horn fraction is on average about 1.9% less than the optimal reverse Horn fraction. For the sub-optimal non-negative fraction and the MAXSAT-fraction the difference is about 2.26% on average. This implies that the MAXSAT-fraction is slightly more difficult to approximate by our greedy sub-optimization procedure. We realize these instances are quite small compared to the instances in the remainder of this paper. Therefore, we compared the sub-optimal non-negative fraction and the MAXSAT-fraction also for 800 instances with 100 variables and density 4.25. In that case the difference between the sub-optimal and the optimal value is on average 2.28%, which is comparable to the result for the instances with 40 variables. Computing optimal reverse Horn fractions for the instances with 100 variables is not possible in reasonable time with the method used. The sub-optimal reverse Horn fraction does not decrease with size as is shown in table 1. In this table, μ_s is the average sub-optimal reverse Horn fraction of the satisfiable instances and μ_u is the average of the unsatisfiable instances. This might indicate that the performance of our greedy heuristic does not deteriorate with increasing size.

The better separation possibilities for the k-level reverse Horn fraction compared to k-level non-negative fraction in section 3.2 might be partly caused by the better approximation, but we do not believe this plays an important role because the difference in error percentages is not that large. The difference between the average MAXSAT-

Table 1
The sub-optimal Horn fraction for instances with increasing size.

#var	μ_s	μ_u
150	0.7102	0.7031
200	0.7108	0.7040
300	0.7106	0.7044
400	0.7105	0.7059

Table 2
40 variables, comparison of reverse Horn and non-negative fraction.

	μ_s	σ_s	μ_u	σ_u	d	$\widehat{\mu}_s$	$\widehat{\sigma}_s$	$\widehat{\mu}_u$	$\widehat{\sigma}_u$	$\widehat{d}$	ϵ
Horn	0.7151	0.0021	0.6979	0.0221	0.0172	0.7271	0.0175	0.7141	0.0158	0.0130	1.9%
NN	0.9772	0.0103	0.9716	0.0095	0.0056	1	0	0.9938	0.0012	0.0062	2.26%

fraction of satisfiable and unsatisfiable instances is 0.0062. For optimal reverse Horn fraction the difference between these two averages is about twice as large (0.013). This could explain why the separation is better for k-level reverse Horn fraction than for k-level non-negative fraction. More details about this experiment comparing reverse Horn and non-negative fraction are given in table 2. In this table, NN indicates non-negative, Horn represents reverse Horn. μ_s is the average sub-optimal fraction of the satisfiable instances, μ_u the average sub-optimal fraction of the unsatisfiable instances. σ_s and σ_u are the corresponding standard deviations. d is the difference between μ_s and μ_u. The parameters with a hat are the corresponding concepts with respect to the average optimal fraction. ϵ is the average difference between sub-optimal and optimal fractions.

3.2. Experiments on reverse Horn fraction and non-negative fraction

Now we will present experimental evidence that the k-level reverse Horn fraction has better separating possibilities than the k-level non-negative fraction. We used a set of random 3-SAT instances with 400 variables and densities 4.0, 4.1, 4.2, 4.3, 4.4 and 4.5, 2000 instances of each density.

Definition 6. The cumulative distribution function $FH_s(x)$ of the satisfiable instances is defined as the probability that a satisfiable instance has a k-level reverse Horn fraction smaller than x.

Definition 7. The probability density function $fH_s(x)$ of the satisfiable instances is the derivative of $FH_s(x)$.

Analogously, $FN_s(x)$ and $fN_s(x)$ are defined for the non-negative fraction. FH_u, FN_u, fH_u and fN_u are defined analogously for the unsatisfiable instances.

To compare the quality of the weak separation, we introduce the concept *overlap*. $FH_s(x)$ and $FH_u(x)$ are approximated by the fraction of instances with a reverse Horn fraction smaller than x. $FN_s(x)$ and $FN_u(x)$ are approximated by the fraction of instances with a non-negative fraction smaller than x. These are approximated with spline functions and differentiated to get fH_s, fH_u, fN_s and fN_u. C_H is defined as the point where the density functions of satisfiable and unsatisfiable instances cross.

The overlap equals the area below corresponding density functions of both satisfiable and unsatisfiable instances. In figure 1 the overlap is the shaded area. The overlap can be computed without using the probability density functions, only using the approximation of the probability distribution which is numerically more stable than its

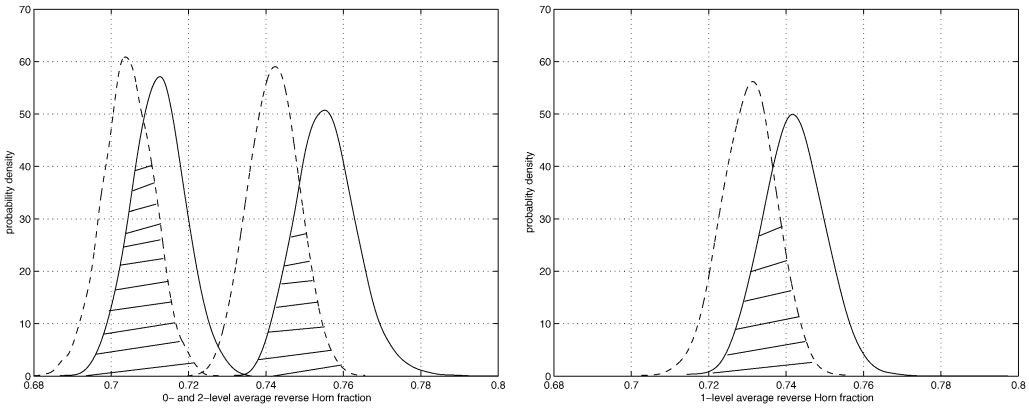

Figure 1. 400 variables, densities 4.0–4.5 joined, probability density functions for 0-, 1- and 2-level reverse Horn fraction.

Table 3
400 variables, densities 4.2, 4.3, and 4.4.

d	Level	μ_s	σ_s	μ_u	σ_u	$\mu_s - \mu_u$	Overlap	C_H
4.2	0	0.7109	0.0067	0.7058	0.0063	0.0051	0.674	0.709
	1	0.7396	0.0073	0.7331	0.0068	0.0065	0.628	0.737
	2	0.7526	0.0066	0.7454	0.0058	0.0072	0.559	0.748
4.3	0	0.7105	0.0065	0.7055	0.0063	0.0050	0.672	0.708
	1	0.7388	0.0067	0.7324	0.0068	0.0064	0.642	0.735
	2	0.7511	0.0061	0.7443	0.0060	0.0068	0.568	0.749
4.4	0	0.7103	0.0056	0.7047	0.0068	0.0056	0.638	0.705
	1	0.7378	0.0064	0.7308	0.0071	0.0070	0.593	0.735
	2	0.7498	0.0054	0.7422	0.0063	0.0076	0.500	0.746

derivative. To see this, note that the point where the density functions cross is exactly the point where the difference between both cumulative distributions is minimal. The smaller the overlap, the better the separation.

Below, we give detailed results for the densities 4.2, 4.3 and 4.4. For instances with density 4.0, 4.1 and 4.5 it is not meaningful to compute the overlap. With density 4.0 there is only 1 unsatisfiable instance, for density 4.1 only 54 unsatisfiable instances and for density 4.5 there are only 10 satisfiable instances.

Tables 4 and 5 give results for the complete data set with densities 4.0, 4.1, 4.2, 4.3, 4.4 and 4.5. Detailed results for $k = 0$, 1 and 2 for the k-level reverse Horn fraction are given in table 4. Table 5 gives the detailed results for the non-negative fraction for $k = 0$ and $k = 1$. In tables 3, 4 and 5 the average k-level reverse Horn (or non-negative) fraction for both the satisfiable (μ_s) and unsatisfiable (μ_u) instances are given, and their difference ($\mu_s - \mu_u$). Columns σ_s and σ_u indicate respectively the standard deviations.

For 0-level reverse Horn fraction the overlap measured over the test set turns out to be 0.571. For 0-level non-negative fraction the overlap is 0.732. Hence, the correlation

Table 4
400 variables, densities 4.0, 4.1, 4.2, 4.3, 4.4 and 4.5, k-level reverse Horn fraction.

#var	Level	μ_s	σ_s	μ_u	σ_u	$\mu_s - \mu_u$	Overlap	C_H
400	0	0.7122	0.0070	0.7044	0.0067	0.0078	0.571	0.708
	1	0.7420	0.0082	0.7305	0.0072	0.0115	0.446	0.736
	2	0.7557	0.0081	0.7419	0.0067	0.0138	0.344	0.749

Table 5
400 variables, densities 4.0, 4.1, 4.2, 4.3, 4.4 and 4.5, k-level non-negative fraction.

#var	Level	μ_s	σ_s	μ_u	σ_u	$\mu_s - \mu_u$	Overlap	C_H
400	0	0.9772	0.0033	0.9751	0.0033	0.0021	0.732	0.977
	1	0.9800	0.0030	0.9774	0.0030	0.0026	0.665	0.979

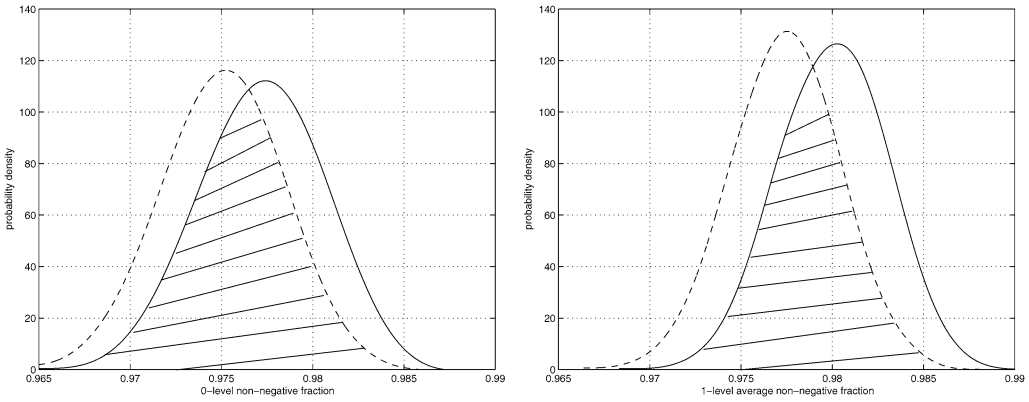

Figure 2. 400 variables, densities 4.0–4.5 joined, probability density functions for 0- and 1-level non-negative fraction.

between 0-level reverse Horn fraction and satisfiability turns out to be stronger. Around the cross point C_H, where the probability density function f_s and f_u cross, instances with any of the clause-variable-densities occur in comparable numbers, which shows that the differences in distributions of satisfiable and unsatisfiable instances are not due to clause-variable-density influences only but are essentially also a result of the parameter reverse Horn fraction itself.

The 1-level reverse Horn fraction improves on the results for the 0-level reverse Horn fraction; the overlap is only 0.446. For the 1-level non-negative fraction the overlap is 0.665. Hence, also on level 1 the reverse Horn fraction performs better than the non-negative fraction.

For the 2-level reverse Horn fraction, the overlap decreases to 0.344. Note that these instances with varying clause-variable densities are easier to separate than instances with a fixed density. For fixed density instances with 400 variables and density 4.25 we showed in [18] that the overlap with respect to the 0-level reverse Horn fraction

and 1-level reverse Horn fraction is respectively 0.718 and 0.672. The first picture in figure 1 shows the fH_s and fH_u with respect to the 0-level and 2-level reverse Horn fraction, the second one for the 1-level reverse Horn fraction. The solid curves are the f_s and the dashed curves the f_u. The left two are f_u and f_s with respect to the 0-level reverse Horn fraction and the right-hand side pair are f_s and f_u with respect to the 2-level reverse Horn fraction. Figure 1 shows that the k-level reverse Horn fractions are larger with search tree depth $k = 2$ than with $k = 0$.

4. Correlation with computational effort

In this section we present empirical evidence for the fact that k-level reverse Horn fraction and computational effort are correlated for a variety of DPLL solvers. To exclude influences from the density parameter, all random 3-SAT instances in these experiments have density 4.25, a density generally believed to be close to the threshold density for the sizes involved. First, we briefly discuss the selected solvers and estimate their average computational effort, in order to present the results independent of size. We give an illustration why scaling with the estimated average computational effort is suitable, and show the difference in the distributions of satisfiable and unsatisfiable instances. We conclude the section with empirical evidence for the correlation between k-level reverse Horn fraction and computational effort for the different solvers. For convenience, we will denote number of nodes by *run time*.

4.1. The solvers selected

The selected complete SAT-solvers are the OKsolver, kcnfs and aHS. The first two are selected because of their good performance on random CNF formulae. aHS is selected because it explicitly utilizes reverse Horn fractions. Each of these solvers is briefly described in this subsection.

- The OKsolver [14] 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. The OKsolver won the award for the best solver in the random category in the SAT-competition in 2002. It still belongs to the best solvers in this category but also performs quite well on different type of instances.
- kcnfs [10] is a complete DPLL-like SAT-solver with a very effective branching heuristic based on the search for variables in the backbone of the CNF-formula. A variable belongs to the backbone of a CNF-formula if it has the same value in every solution that satisfies the maximum number of clauses. kcnfs won the award for the best solver in the random category in the SAT-competition 2003. It is designed and optimized to perform well in this category. Performance outside this category turned out to be poor.

- aHS is a complete solver with as a special property that the search tree is not binary. It uses the reverse Horn fraction and linear programs to guide its search. A description of this solver can be found in section 7. aHS is our tool to compute k-level reverse Horn fractions if it is run up to level k, but aHS is also a complete SAT-solver if it is run till completion.

4.2. *Estimation of average run time*

We will briefly describe how the average run time can be estimated. Estimating the average run time for each solver and size is necessary to be able to present size independent results on the correlation between k-level reverse Horn fraction and run time. The *scaled run time* of an instance with a given size is defined as the run time of the instance divided by the estimated average run time of that particular size.

For estimating the average run time, we used 13,000 random 3-SAT instances with the number of variables between 125 and 400. These instances are solved by the OKsolver, kcnfs and the smaller instances among those by aHS. For each of the selected sizes, this results in the average run time for that particular size. For each solver, we compute an exponential fit to these data points (number of variables, average run time), which estimates the average run time. In figure 3 the exponential fits estimating run time are plotted. The horizontal axis gives the number of variables, the vertical axis the estimated average run time. The upmost curve is the average run time for aHS, the middle one is for the OKsolver and the lowest for kcnfs.

Each exponential fit is of the form

$$RT_{\text{av}} = \alpha e^{\beta n},$$

with RT_{av} the average run time, and n the number of variables. The parameters α and β are given in table 6 for each solver. The exponential fits constitute a good approximation of the data. The maximum gap between the function value of the fit and the observed data points for the OKsolver is 2.3%. For kcnfs the maximum gap equals 2.7% and for aHS 4.2%.

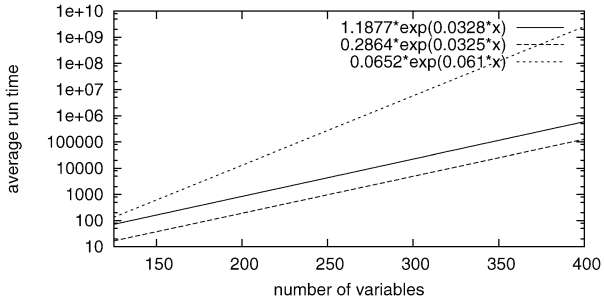

Figure 3. Average run times for OKsolver, kcnfs and aHS.

Table 6
Parameters of exponential fit for the three solvers.

Solver	α	β
OKsolver	1.1877	0.0328
kcnfs	0.2864	0.0325
aHS	0.0652	0.0610

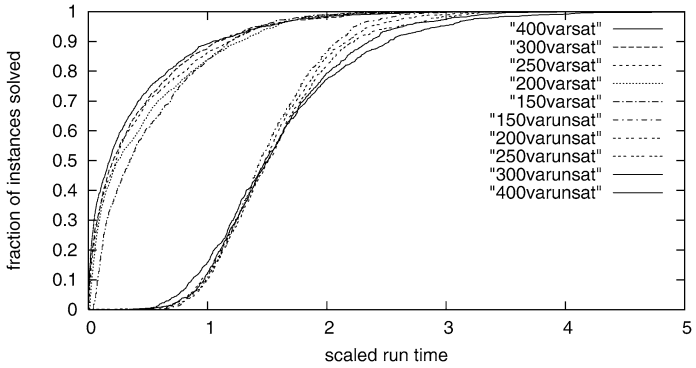

Figure 4. Scaled run time distribution, OKsolver, 150–400 variables.

4.3. Scaled run time distribution

In this subsection, we show that the estimation made above works well in scaling run times for different sizes and we observe that the probability distributions of run times of satisfiable instances have a different shape than those of unsatisfiable ones.

Figure 4 shows the probability distributions with respect to the scaled run time for the OKsolver for instances with 150–400 variables. On the horizontal axis is indicated the scaled run time. On the vertical axis is the fraction, or probability, $F(t)$ that an instance is solved before run time t. The logarithmic-shaped curves concern satisfiable instances and the S-shape curves the unsatisfiable instances. The names of the curves are given in top-down order in the figure. We conclude that the scaled run time distribution is quite size independent. Similar pictures emerged for the other solvers.

4.4. Correlation between reverse Horn fraction and run time

Now we show how k-level reverse Horn fraction and run time are correlated for the fixed-density problems considered. For the sizes considered we take $k = 2$ to illustrate these features. In our previous paper on this subject, we have shown a global trend in the correlation between k-level reverse Horn fraction and run time for OKsolver, kcnfs and aHS. Satisfiable instances get on average easier when k-level reverse Horn fraction increases and unsatisfiable instances get more difficult with increasing k-level reverse Horn fraction. The intuition behind satisfiable instances with a small k-level reverse Horn fraction being more difficult is that these instances probably have only a few satis-

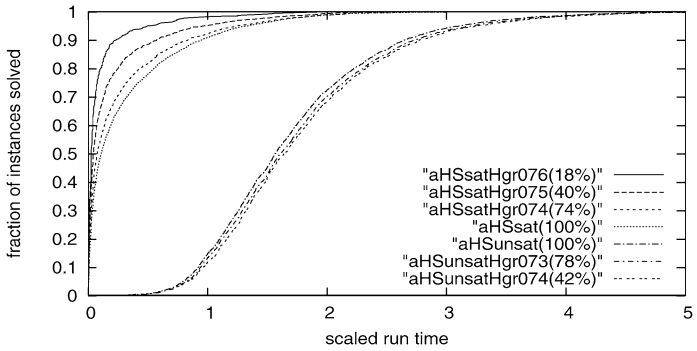

Figure 5. aHS, 125, 150, 175 and 200 variables, density 4.25.

fying assignments. On the other hand, unsatisfiable instances with a large k-level reverse Horn fraction are difficult because a large reverse Horn fraction indicates few conflicts in the formula and hence it is difficult to find these conflicts to prove unsatisfiability.

The instances used in figure 5 are random 3-SAT instances with 125, 150, 175 and 200 variables. On the horizontal axis is indicated the scaled run time, on the vertical axis the fraction of instances $F(t)$ that is solved before run time t. The names of the curves are in the top-down order of the curves. The lowest logarithmic-shaped curve corresponds to the complete set of satisfiable instances. If we restrict the set of satisfiable instances to the instances with a 2-level reverse Horn fraction larger than or equal to 0.74, 0.75 and 0.76 respectively, we observe that the curves become steeper. The percentages given in figure 5 are the percentages of instances that respect the corresponding lower bound on the 2-level reverse Horn fraction. The difference between the curves indicates that satisfiable instances with a larger 2-level reverse Horn fraction are easier for aHS to solve. Similarly, we plotted in figure 5 three curves for the unsatisfiable instances, the upmost S-shaped curve for the complete set of unsatisfiable instances, the second one for instances with a 2-level reverse Horn fraction larger than 0.73 and the lowest one for unsatisfiable instances with a 2-level reverse Horn fraction larger than 0.74. The order of the curves illustrates that contrarily to the satisfiable instances, unsatisfiable instances with a larger 2-level reverse Horn fraction are more difficult.

Similar pictures emerge for the OKsolver and kcnfs, also for larger instances (with 400 variables), as can be seen in figure 6. We observe a same shape as in figure 5. We conclude that the correlation between 2-level reverse Horn fraction and run time for the OKsolver and kcnfs is well observable.

5. Correlation between reverse Horn fraction and performance of incomplete solvers

In this section, we study the correlation between the k-level reverse Horn fraction and the performance of incomplete solvers. For this study we selected two state-of-the-art incomplete solvers: SP [6] and UnitWalk [13].

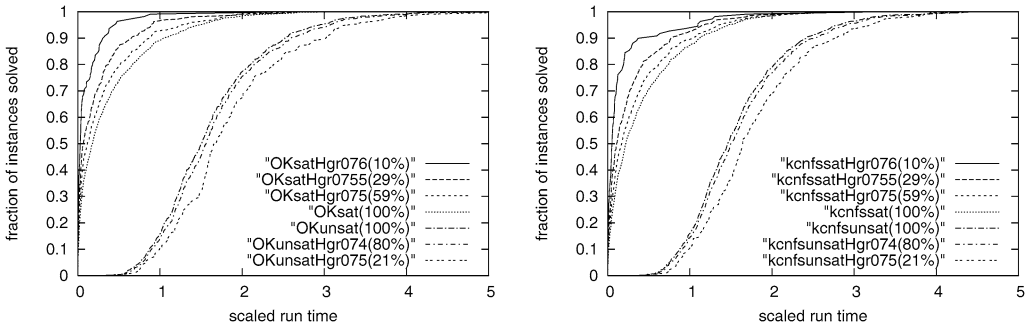

Figure 6. OKsolver and kcnfs, 400 variables, density 4.25.

SP (SurveyPropagation) is an incomplete solver which uses an improved message passing algorithm to solve instances close to the threshold of 3-SAT, which have clusters of solutions that are not very similar and many clusters of partial solutions. The authors report to be able to solve instances with 10^6 variables. SP is designed for really large instances (more than 10,000 variables). Hence, the performance on the small instances with 400 variables in our experiment is not representative for the performance of SP. However, we decided to include the results because they still give some insight in the correlation between k-level reverse Horn fraction and solver performance.

UnitWalk is an incomplete SAT-solver combining local search and unit clause elimination. In the SAT-2003 competition UnitWalk won the award in the satisfiable random benchmarks category.

5.1. SurveyPropagation and reverse Horn fractions

This subsection shows a correlation between 2-level reverse Horn fraction and the success rate of SP, in the sense that SP finds a solution more often for instances with a larger 2-level reverse Horn fraction than for ones with a smaller 2-level reverse Horn fraction. We ran SP on 1000 satisfiable instances with 400 variables and density 4.25. On the complete set of instances SP finds a solution for 38.8% of the instances. However, if we partition the instances in six groups of about the same number of instances with increasing 2-level reverse Horn fraction, the success rate of finding a solution increases from 26.9% in the first group to 68.8% in the last group. Table 7 gives the detailed results of this experiment. It shows a clear correlation between 2-level reverse Horn fraction and the performance of SP.

5.2. UnitWalk and reverse Horn fractions

For UnitWalk it turned out impossible to analyze its performance similarly as we did for SP, because the success rate on the same benchmark set is 100%. (We will make use of this excellent performance in section 6.) To get an impression of the influence of reverse Horn fraction on UnitWalk's performance, we let it try to solve 500 instances with 600 variables and density 4.25. These instances are currently too large to be solved

Table 7
SurveyPropagation on instances with 400 variables and density 4.25.

2-level reverse Horn fraction	Success rate
[0,0.745]	26.9%
(0.745,0.75]	29.6%
(0.75,0.755]	38.0%
(0.755,0.76]	45.7%
(0.76,1]	68.8%

in reasonable time with a complete solver. For 59.4% of the instances UnitWalk finds a solution within 600 seconds (clock time). For these proven satisfiable instances, a figure similar to figure 5 showed up (not included because of its similarity). Difference between the curves are somewhat less pronounced than for OKsolver, kcnfs and aHS.

6. Correlation between reverse Horn fraction and satisfiability scales

In this section, we show that the results on the correlation between k-level reverse Horn fraction and satisfiability in [18] with data up to 400 variables scale to larger instances with 600 variables. We used the excellent performance of UnitWalk to extend the data set.

We ran UnitWalk on 2000 instances with 600 variables and density 4.25 with a limit on (clock) time of 1800 seconds. UnitWalk finds a solution for 1149 instances (57.5%). Because the density is close to the 3-SAT-threshold it is likely that most of the remaining instances are unsatisfiable. We tested UnitWalk also on 500 instances of this size and density 4.27. From these instances, UnitWalk finds a solution in 44.6% of the cases. A second indication that at most for only a few satisfiable instances a solution is not found is the fact that the average 0-level reverse Horn fraction for the instances with 600 variables does not deviate much from that for the 400 variable instances for which is sure that all unsatisfiable instances are really unsatisfiable. A satisfiable instance that is not solved by UnitWalk is likely to have a 0-level reverse Horn fraction smaller than the average, because the satisfiable instances with a small 0-level reverse Horn fraction tend to be difficult. If many instances would not be solved by UnitWalk the average 0-level reverse Horn fraction would be considerably higher for the satisfiable instances with 600 variables than for the ones with 400 variables, which is not the case. From these facts, we conclude that in order to establish empirical evidence for the correlation between satisfiability and k-level reverse Horn fraction, one might read 'satisfiable' as 'solved by UnitWalk' and 'unsatisfiable' as 'not solved by UnitWalk' rather safely: the small number of errors made presumably will not essentially alter the probability density functions measured.

Table 8 gives the details on the distribution of the k-level reverse Horn fraction for instances with 600 variables. For comparison, we also included the data for 400 variable instances from [18]. The full data can be found in that paper. We computed the average k-level reverse Horn fraction for both the satisfiable (μ_s) and unsatisfiable (μ_u)

Table 8
400 and 600 variables, density 4.25.

#var	Level	μ_s	σ_s	μ_u	σ_u	$\mu_s - \mu_u$	Overlap	C_H
400	0	0.7105	0.0065	0.7059	0.0063	0.0046	0.718	0.708
	1	0.7390	0.0069	0.7333	0.0066	0.0057	0.672	0.736
	2	0.7519	0.0063	0.7454	0.0059	0.0065	0.591	0.749
600	0	0.7102	0.0055	0.7060	0.0054	0.0042	0.664	0.709
	1	0.7392	0.0063	0.7338	0.0060	0.0054	0.622	0.737
	2	0.7530	0.0053	0.7473	0.0050	0.0057	0.558	0.750

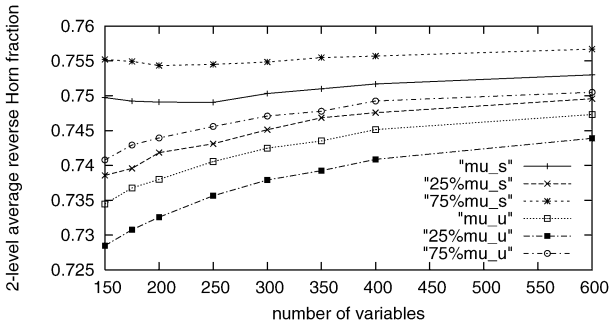

Figure 7. Expected 2-level reverse Horn fraction.

instances, and their difference ($\mu_s - \mu_u$). This difference is an indication for the quality of separation. σ_s and σ_u indicate respectively the standard deviations. Finally, the table gives the overlap and cross point C_H.

Behavior with respect to expected 1- and 2-level reverse Horn fraction, standard deviations and overlap is according to the trend for the smaller instances with up to 400 variables studied in [18]. Figure 7 illustrates this trend. We plotted the average 2-level reverse Horn fraction for both the satisfiable and unsatisfiable instances as function of size. Below each of these lines, we indicate the 25%-percentile, i.e. that fraction such that 75% of the instances has a larger 2-level reverse Horn fraction. Analogously, 75%-percentiles are indicated. Expected 2-level reverse Horn fractions μ_s and μ_u increase with size. It shows that the correlation between 2-level reverse Horn fraction and satisfiability does not get weaker with increasing size. This fact advocates that these correlations are not 'small size artifacts'.

7. The algorithm aHS

In this section we present some statistics about a typical run on a 3-SAT instance. Also, some aspects related to the choice of the objective function of the linear programming relaxation are discussed. First, we briefly describe the main steps of the algorithm aHS introduced in [18]. Steps 2, 4 and 5 are described in more detail below. We empha-

size that aHS is not designed to be an efficient SAT-solver, but to generate many nodes at lower levels to make the averaging process more reliable.

1. Identify all unit clauses; propagate the corresponding values through the formula until no unit clauses remain.
2. Apply failed literal tests (FLT) in order to simplify the formula as much as possible. If the formula is proved unsatisfiable or a satisfying assignment is found: STOP. Otherwise: step 3.
3. Greedy suboptimize the reverse Horn fraction using a greedy local search renaming heuristic as described in subsection 2.1.
4. Solve an associated linear program. Let $\omega = (\omega_1, \ldots, \omega_n)$ be a solution (splitting point) to this linear program.
5. Define $n + 1$ subproblems using a decomposition of the search space based on ω.
6. Go into breadth-first recursion.

The associated linear programming relaxation is described in subsection 7.2. Subsection 7.3 clarifies the branching strategy from step 5. In subsection 7.4 a justification of the linear program is given together with an example.

7.1. Failed literal tests

Iterative unit propagation is the process of propagating the literals in unit clauses till no unit clauses remain. In the failed literal tests, every variable is set temporarily to true. If this does lead to a contradiction after iterative unit propagation the variable can be fixed to false for the current node and all of its children. This value is propagated followed by iterative unit propagation. If this leads to a contradiction we can conclude that the formula is unsatisfiable and backtrack.

These failed literal tests are 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.

7.2. 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 an $m \times n$ 0–1 matrix, with $a_{ij} = 1$ if variable X_j is contained in clause i, $a_{ij} = -1$ if $\neg X_j$ is contained in clause i and 0 otherwise. Let x be a 0–1 vector with $x_j = 1$ if X_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 x satisfies

$$\left\{Ax \geqslant b, x \in \{0, 1\}^n\right\}.$$

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

The constraints $x \in \{0, 1\}^n$ are replaced by $0 \leqslant x \leqslant 1$ to obtain the associated linear programming relaxation in step 4. The objective of the linear program is to maximize $\sum x_j$. Setting all free variable to 0.5 satisfies all linear inequalities. With the objective function we increase the chance that the solution to the linear programming relaxation is integer-valued. If an integer-valued linear programming solution is found, this is a solution to the integer linear program (and hence, the CNF-formula) and the algorithm can be stopped.

7.3. Explanation of branching strategy

If the formula is not solved in a search tree node, $n + 1$ subproblems are defined using a decomposition of the search space based on the linear programming solution ω. The first region G_0 is defined as:

$$G_0 = \{x = (x_1, \ldots, x_n) \mid x_i \leqslant \lfloor \omega_i \rfloor, \text{ for all } i\}.$$

Each G_i, $i = 1, \ldots, n$, is defined by

$$x_i \geqslant \lfloor \omega_i \rfloor + 1, \tag{1}$$

$$x_i - x_k \geqslant \lfloor \omega_i - \omega_k \rfloor + 1, \quad k < i, \tag{2}$$

$$x_i - x_k \geqslant \lceil \omega_i - \omega_k \rceil, \quad k > i. \tag{3}$$

$\lfloor x \rfloor$ is defined as the largest integer smaller than or equal to x. $\lceil x \rceil$ is the smallest integer larger than or equal to x.

The branching strategy results in fact only in adding a number of unit clauses to the formula in each of the regions G_0 to G_k. Assume that we ordered the boolean variables $X_1, \ldots, X_n$ in non-decreasing order of the corresponding solution $(\omega_1, \ldots, \omega_n)$ of the linear programming solution. Let $X_{[i]}$ be the ith variable in this ordering. If the solution does not have all ones, region $G_{[1]}$ is simply obtained by fixing $X_{[1]}$ to 1. All other (free) variables remain free. In general, $G_{[i]}$ for each i such that $X_{[i]}$ is not fixed and $\omega_{[i]} < 1$ is as follows:

$$G_{[i]} = \{x \mid x_{[i]} = 1, x_{[k]} = 0 \text{ for each } k < i\}.$$

By constraint (1), region i is empty if $\omega_{[i]} = 1$. In each G_i these region constraints are propagated before FLT is applied and the linear programming relaxation in the node solved.

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

$$G_0 = \{x_1 \leqslant 1, x_2 = x_3 = x_4 = 0\}, \tag{4}$$

$$G_1 = \emptyset, \tag{5}$$

$$G_2 = \{x_2 = 1, x_4 = 0, x_1, x_3 \text{ free}\}, \tag{6}$$

$$G_3 = \{x_3 = 1, x_2 = x_4 = 0, x_1 \text{ free}\}, \tag{7}$$

$$G_4 = \{x_4 = 1, x_1, x_2, x_3 \text{ free}\}. \tag{8}$$

7.4. The objective of the linear program

The objective of the linear program is to maximize $\sum x_j$. We want to add a justification for using this objective function. The main intuition behind the search space splitting is: try to eliminate many non-reverse Horn clauses. By eliminating non-reverse Horn clauses the CNF-formula is supposed to get closer to a reverse Horn formula, which hopefully makes it easier to (partially) solve by the failed literal tests. A linear inequality is defined to be stringent to the optimal linear programming solution if the slack of this inequality is 0 with respect to this solution. The slack of the linear inequality $ax \geqslant b$ is $ax - b$. A clause is defined to be *stringent* if the corresponding linear inequality is stringent to the optimal linear programming solution. A clause is defined to be *affected* in region G_i at level 1 if it is either satisfied or shortened in region G_i. The objective function tries to get as many as possible non-reverse Horn clauses stringent to the optimal linear programming solution. These stringent clauses turn out to have a larger probability to be affected at the next level in the search tree.

In the following argument, the above is illustrated. We use a two-literal non-reverse Horn clause, but one could give similar arguments for non-reverse Horn clauses with more literals. Assume without loss of generality $\omega_p \leqslant \omega_q$. A clause $\neg X_p \vee \neg X_q$ in the formula is satisfied at the next level in all regions G_i with $\omega_i > \omega_p$ if $i < p$ and $\omega_i \geqslant \omega_p$ if $i > p$, because of the definitions of the regions. Now, for a *stringent* clause of the type $\neg X_p \vee \neg X_q$, it holds that $(1 - \omega_p) + (1 - \omega_q) = 1$, resulting in $\omega_q = 1 - \omega_p$. Hence, the clause is satisfied in each region G_i with $\omega_i > \min(\omega_p, \omega_q) = \min(\omega_p, 1 - \omega_p)$, which is always smaller than or equal to 0.5. The linear program maximizes $\sum x_i$ and $(x_1, \ldots, x_n) = (0.5, \ldots, 0.5)$ is feasible for the linear program. Hence, $\frac{1}{n}\sum_k \omega_k$ is at least 0.5. However, for random 3-SAT CNF formulae, the average ω_k of the root-node linear program turns out to be about 0.75. This implies that a stringent clause $\neg X_p \vee \neg X_q$ is very likely to be satisfied, because most of the ω_i-values will be larger than $\min(\omega_p, 1 - \omega_p)$. From the above, we might conclude that stringent non-reverse Horn clauses are satisfied in most nodes at the next level.

Example. The following example illustrates the above. We consider an arbitrary selected random 3-SAT instance with 200 variables and density 4.25. FLT is an abbreviation for failed literal tests. For this instance we solved the linear programming relaxation in the root node and examined the nodes at level 1. The average percentage of clauses affected is the average over the percentages of affected clauses in the non-empty regions G_i at level 1. Analogously, the average percentage of stringent clauses affected and the average percentage of affected stringent clauses that are not-reverse Horn are defined.The statistics of this instance are presented below.

From the above we see that for the root-node linear program 75% of the stringent clauses are non-reverse Horn, which means that non-reverse Horn clauses are more likely

Table 9
Statistics for an instance with 200 variables.

Objective value	156.5
$\frac{1}{n}\sum_i \omega_i$	0.78
% of stringent clauses	27%
% of stringent non-reverse Horn clauses	75%
Fraction of regions not empty after propagating (1)–(3)	8.5%
Fraction of regions not empty and not solved after FLT	6.0%
Average % of clauses affected	16.4%
Average % of stringent clauses affected	19.2%
Average % of affected stringent clauses that are non-reverse Horn	61.5%

to be stringent than non-reverse Horn clauses. At level 1 stringent clauses are more likely to be affected than arbitrary clauses. Notice that from the 201 possible child nodes at level 1 only 6% is not empty and not solved by the failed literals tests. It can be concluded that the branching strategy aims at affecting non-reverse Horn clauses, and hence increasing the (non-renamed) reverse Horn fraction in the child nodes compared to their parent node. The greedy heuristic can only increase the reverse Horn in a node by applying a sub-optimal renaming procedure.

8. Conclusions

We presented evidence for the fact that the k-level reverse Horn fraction may play a similar (albeit weaker) role as the density parameter in separating satisfiable random 3-SAT instances from unsatisfiable ones. If it is impossible to discriminate using the density parameter the k-level reverse Horn fraction seems suitable. From a statistical point of view it is favorable to combine weak separators. The k-level reverse Horn fraction can be combined with other different weak separators as for instance discussed in [2] and [19]. This combination could be subject of further study. It is remarkable that the k-level reverse Horn fraction indeed has weakly separating properties because random 3-SAT instances start with having expected reverse Horn fraction equal to 0.5 and hence seem rather insensitive with respect to this parameter. It turned out that the k-level reverse Horn fraction suits this purpose better than a similarly obtained sub-optimal MAXSAT-fraction. Performances of various state-of-the-art solvers (complete as well as incomplete ones) seem to be correlated to the k-level reverse Horn fraction as well, although any link to or use of reverse Horn clauses in these configurations is absent. Recently, some theoretical back-up for the correlations observed come from [16] and [17]. In the first paper, a relation between Horn fraction and the size of "backdoor" sets is established. The size of these sets are correlated to computational effort of DPLL based solving techniques. The second paper shows that larger Horn fractions yield lower complexity worst case bounds.

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 (1979) 121–123.

[2] H. Bennaceur and C.M. Li, Characterizing SAT problems with the row convexity property, in: *Principles and Practice of Constraint Programming – CP2002*, ed. P. Van Hentenrijck (2002).

[3] 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 (2002) 381–394.

[4] B. Borchers and J. Furman, A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems, Journal of Combinatorial Optimization 2(4) (1999) 299–306.

[5] E. Boros, Maximum renamable Horn sub-CNFs, Discrete Applied Mathematics 96–97 (1999) 29–40.

[6] A. Braunstein, M. Mezard and R. Zecchina, SurveyPropagation: an algorithm for satisfiability, Preprint (2002).

[7] S.A. Cook, The complexity of theorem proving procedures, in: *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing* (1971) pp. 151–158.

[8] Y. Crama, O. Ekin and P.L. Hammer, Variable and term removal from boolean formulae, Discrete Applied Mathematics 75 (1997) 217–230.

[9] W.F. Dowling and J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, Journal of Logic Programming 1 (1984) 267–284.

[10] O. Dubois and G. Dequen, A backbone-search heuristic for efficient solving of hard 3-SAT formulae, in: *Proceedings of IJCAI-01* (2001).

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

[12] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified NP-complete graph problems, Theory of Computer Science 1 (1976) 237–267.

[13] E.A. Hirsch and A. Kojevnikov, UnitWalk: A new SAT solver that uses local search guided by unit clause elimination, in: *Electronic Proceedings of SAT-2002* (2002).

[14] O. Kullmann, Investigating the behaviour of a SAT solver on random formulas, Annals of Mathematics and Artificial Intelligence (2002).

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

[16] N. Nishimura, P. Ragde and S. Szeider, Detecting backdoor sets with respect to Horn and binary clauses, in: *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing* (2004).

[17] S. Porschen and E. Speckenmeyer, Worst case bounds for some $\mathcal{NP}$-complete modified Horn-SAT problems, in: *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing* (2004).

[18] H. van Maaren and L. van Norden, Hidden threshold phenomena for fixed-density SAT-formulae, in: *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003, Selected Revised Papers*, eds. E. Giunchiglia and A. Tacchella, Lecture Notes in Computer Science, Vol. 2919 (2004) pp. 135–149.

[19] H. van Maaren and J.P. Warners, Solving satisfiability problems using elliptic approximations. A note on volumes and weights, Annals of Mathematics and Artificial Intelligence 37 (2003) 273–283.