

The SAT 2005 solver competition on random instances

Oliver Kullmann*

O.Kullmann@Swansea.ac.uk

*Computer Science Department
University of Wales Swansea
Swansea, SA2 8PP, UK*

Abstract

An analysis of the SAT 2005 sub-competition on random instances is given. This year this (sub-)competition set-up was geared to establish a basic setting, focusing on the instances near the (infamous) “50% densities”, so first we have to establish a more quantitative understanding of phase transitions here. We present extended empirical results, clearly showing that the models used before, which are motivated by large-scale considerations, are inadequate at the relatively small scale considered here. Then the series’ and all individual instances used in the competition are described. We give a formal definition of the competition evaluation, and analyse the results of the competition, looking into the details of the scoring mechanism as well as into alternative evaluation methods.

KEYWORDS: *Random satisfiability problems, Phase transition, Threshold behaviour, Critical exponent, SAT competition*

Submitted October 2005; revised February 2006; published March 2006

1. Introduction

The SAT 2005 solver competition has three sub-competitions: the “i-competition” on so-called “industrial benchmarks”, the “h-competition” on so-called “handmade benchmarks”, and the “r-competition” on random CNF. Each sub-competition itself has three sub-competitions regarding the selection of benchmarks, namely considering all benchmarks or only satisfiable or only unsatisfiable benchmarks. In this article we discuss the r-competition in detail, considering first the choice of benchmarks, then discussing the evaluation criteria for the solver results, and finally interpreting the outcome. In the final section we give some recommendations for the next r-competition (planned for SAT 2007).

The goals for the build of the r-competition can be seen on the one hand in finding a basic selection of challenging benchmarks (which are “just right”, neither too hard nor too easy), and on the other hand in setting up a reasonably fair evaluation framework. Unlike the i- and h-competition with their wide ranges of very diverse benchmarks, the r-competition is based on quite predictable benchmarks (this is after all the main advantage in using random formulas), and thus an especially fine-tuned evaluation scheme for the r-competition is conceivable. However, all sub-competitions are using the same uniform evaluation scheme;

* Supported by EPSRC Grant GR/S58393/01

this year’s SAT competition actually introduced a new evaluation scheme, giving up the previously used more “qualitative” (and very simple) evaluation (see [4]), and setting up a purely quantitative (and more complex) scoring scheme. Within this framework, it seems fair to say that that both goals have been realised: The benchmarks allowed to clearly distinguish the strongest solvers, and the new evaluation scheme molded the multidimensional aspects of “the strongest solver” (which depend on several parameters) into one coherent picture. Further argumentation one finds in the rest of the paper.

1.1 The choice of benchmarks

For the selection of benchmarks we wanted to establish a new basic standard, so we considered only those formulas which have attracted most attention, namely random k -CNF near the assumed phase transition (from almost always satisfiable to almost always unsatisfiable). The case $k = 3$ is of greatest interest here, but we believe that a better understanding of random formulas needs to consider mixed clause-sizes (as was demonstrated with the well-known “ $2 + p$ ”-model; see [14]), and so as a first step we considered also $k = 5, 7$. Even asymptotically the notion of phase transition for SAT is yet not well understood, and for “finite sizes” the situation is even less clear; as we discuss in Section 2, a good “finite approximation” of the location of “the phase transition” is to consider the density $\rho = \mathfrak{d}_1(\frac{1}{2}, n, k)$ where 50% of all random k -CNF’s are satisfiable (for a given number n of variables), and so the benchmarks aim at giving a random sampling of random 3, 5, 7-CNF near the 50%-density for interesting n -values (which here always denotes the number of variables).

Due to time constraints, the r-competition can only consider very small sample sizes; for each pair of k, n, ρ considered (clause-length, number of variables, density), which makes a “series” in the language of the SAT competition, only 10 benchmarks are allowed. Just creating 10 random formulas, also for $\rho \approx \mathfrak{d}_1(\frac{1}{2}, n)$ with high chance some series are very biased towards having nearly only satisfiable or nearly only unsatisfiable instances, which is intolerable here since it would basically ruin the series (the incomplete solvers cannot handle the unsatisfiable instances at all, while having too many satisfiable instances does not allow to differentiate between complete solvers, which are more geared towards unsatisfiable instances). So the benchmarks had to be created in such a way that the right balance between satisfiable and unsatisfiable instances was guaranteed. Of course, randomness had also to be guaranteed, and thus selecting such instances were pre-decision was successful (leaving out other instances) is not an option. Complete solvers currently can handle only quite small instances (called “medium size instances” here), for which the spread of running times is still feasible, and thus all medium size instances have been pre-decided.¹

For incomplete solvers the situation is different however, since at problem sizes challenging for these solvers the variation of run times are enormous; as the reader can see with Table 5, the larger series with the large size benchmarks still contain instances which not only no solver from the competition could solve, but which have not been solved until

1. Randomness is guaranteed by just choosing the first 5 satisfiable and the first 5 unsatisfiable instances given by the random generator, *without skipping instances*; given enough time, this can be done.

today. So here a different approach had to be used: All instances should be satisfiable, and they should be satisfiable “by construction” (no pre-decision needed — currently likely no unsatisfiable instance of this size (and near the 50%-density) could be solved by any means provided on this planet). So densities ρ_k were chosen which seem to be well below the 50%-density so that all instances should be satisfiable, while still close enough to the 50%-density to be challenging.

1.2 The results

Progress with complete solvers on random formulas is slow and arduous. This year the winner of last year, `kcnfs-2004`, was still dominant. It seems that the second solver, `march_dl`, was aiming more to be an all-round-solver (much more so than `kcnfs-2004`, which outside its domain of random CNF does not show good performance) and so couldn’t mess with `kcnfs-2004` on random formulas (especially on longer clause-lengths, where `march_dl` shows quite weak performance, likely due to the fact, that `march_dl` in its preprocessing translates all formulas into 3-CNF).

Incomplete solvers (especially local search solvers) have always been strong on random formulas, and this year is no exception. Last year’s winner, `adaptnovelty`, was clearly beaten by the new champ, `ranov`, quite clearly dominating the competition on satisfiable (random) formulas. `ranov` also achieved the highest score on all random formulas, which seems not unreasonable, while `kcnfs-2004` achieved a creditable third rank here;² however, there are serious doubts whether it makes sense to compare solvers which are only capable of solving satisfiable instances with complete SAT solvers, and thus for the final official ranking only complete solvers have been taken into account in the all-instances sub-competition.

1.3 Outline of this article

In Section 2 we present an excerpt of our own experimental work on the satisfiability phase transition, which started with [9, 10]. The basic problem here is to get reasonable predictions for the 50% density function $\mathfrak{d}_1(\frac{1}{2}, n, k)$; we review shortly the existing experimental studies in the literature, and our material clearly demonstrates that none of these approaches, which are motivated by “large scale” considerations, gives a good approximation for the given “small scale”. In order to make the motivation for these approaches comprehensible, we also give a basic review on “scaling”. Table 1 gives by far the most comprehensive experimental data on the location of 50% densities so far, while with model (2) a numerically quite reasonable general model for the probability of satisfiability of *arbitrary* random 3-CNF is given; the derived model (3) for $\mathfrak{d}_1(p, n)$, the density of random 3-CNF with n variables having a satisfiability probability p , is for the special case of $p = \frac{1}{2}$ not much worse than the special models (7), (8), (9), (10) and (11) obtained for the 50% density (only for clause-length $k = 3$; for greater clause-length at this time not enough numerical information has been gathered to obtain a similarly detailed picture). While these special models either

2. That actually both incomplete and complete solvers were among the three highest scores in the all-formulas sub-competition can be seen as a success of the new evaluation method, which in a somewhat unpredictable way takes many factors into consideration.

stipulate the threshold density or the critical exponent (by using some external reasoning), with model (12) we were able to fit all parameters numerically, obtaining a model offering by far the best fit (as always in this article, this holds only for the range of n -values considered).

In Section 3 we explain the choices and properties of our benchmarks. Section 4 then discusses the evaluation of the competition. Subsection 4.1 precisely defines the scoring method, while Subsection 4.2 presents the solver evaluations in detail. We conclude in Section 5 with a resume on this year’s competition and an outlook on the next competition. The appendices contain comprehensive experimental data (including a detailed description of each single benchmark, and a complete alternative analysis, based on lexicographical ordering).

2. The satisfiability threshold for random formulas

Since we decided to focus this year’s r-competition on instances close to the “50% densities”, we need to know how they can be defined and how to compute them. In Subsection 2.1 we (curtly) review the model for random formulas used here and what is known (especially numerically) about the “threshold” and the “phase transition”. The extensive computer experiments performed by us so far led to a first (preliminary) general model (1) for the probability that a random 3-CNF is satisfiable. Then in Subsection 2.2 we have a closer look at the 50% densities for 3-CNF, based on the experimental data presented in Table 1.

Since our experimental results are only applicable to clause-size $k = 3$ and medium number of variables (and thus only here we have a quite precise knowledge about the 50% densities), here is what we have done for the other situations:

1. For the large size instances of random 3-CNF we have chosen the constant density 4.2. Recall that we want to make sure that all these formulas are satisfiable; now there are no indications that the 50% densities for 3-CNF will ever fall below 4.24, and thus, using the (assumed) increasing sharpness of the phase transition with increasing n , the probability of finding an unsatisfiable formula at density 4.2 for the values of n considered here should be very low.

2. Clause-size $k = 5$:

It is known that $18.79 < \mathfrak{T}(5) < 21.33$ holds for the (assumed) random 5-CNF threshold $\mathfrak{T}(5)$ (see [2, 1]); [13] predicts $\mathfrak{T}(5) \approx 21.117$.

- (a) For the medium size instances we have chosen a fixed density 21.3, which according to our experience is an estimation which is “good enough”.
- (b) For the large size instances we chose the fixed density 20.

3. Clause-size $k = 7$:

It is known that $84.82 < \mathfrak{T}(7) < 87.88$ holds for the (assumed) random 7-CNF threshold $\mathfrak{T}(7)$ (see [2, 1]); [13] predicts $\mathfrak{T}(7) \approx 87.79$.

- (a) For medium size instances the fixed density 89 was chosen; though outside of the (proven) interval for the (assumed) threshold value, according to our experimentations this density is close to the 50% density for the considered n -values, which is vindicated by Table 4.
- (b) For large size instances we chose the density 85.

2.1 The fixed clause-length model for random formulas

We use the standard model for fixed clause-length random formulas, where for given $k, c, n \in \mathbb{N}_0$ we consider all $2^k \cdot \binom{n}{k}$ clauses of fixed size k (without repetitions and without clashing literals) over variables $1, \dots, n$, and where all $(2^k \cdot \binom{n}{k})^c$ clause-sequences (“random CNF formulas”) of length c are equally likely. Now $\Pr_\varepsilon(k, c, n)$ is the probability, that such a random CNF is satisfiable ($\varepsilon = 1$) resp. unsatisfiable ($\varepsilon = 0$). Following the more general notation from [10], allowing also for mixed clause-sizes, I will actually use the notation $\Pr_\varepsilon(\rho \cdot \Delta_k, n) := \Pr_\varepsilon(k, \text{round}(\rho \cdot n), n)$, where $\text{round} : \mathbb{R}_{\geq 0} \rightarrow \mathbb{N}_0$ is standard rounding to the nearest integer; ρ here is called the *density*.

Figure 1 shows experimental data on the threshold behaviour of the satisfiability probabilities $\Pr_1(\rho \cdot \Delta_3, n)$ for variable numbers n in the range from 100 to 400 and densities ρ from 4.0 to 4.6. We see that the curves are turning from strictly increasing to strictly decreasing between densities 4.2 and 4.3; this behaviour is called the *threshold phenomenon*, and a generally accepted conjecture (the “Satisfiability Threshold Conjecture”) is, that for every $k \in \mathbb{N}$, $k \geq 2$ there is a *critical density* $\mathfrak{T}(k) \in \mathbb{R}_{>0}$, such that for all $\varepsilon > 0$ we have $\lim_{n \rightarrow \infty} \Pr_1((\mathfrak{T}(k) + \varepsilon) \cdot \Delta_k, n) = 0$ and $\lim_{n \rightarrow \infty} \Pr_1((\mathfrak{T}(k) - \varepsilon) \cdot \Delta_k, n) = 1$. Up to now only $\mathfrak{T}(2) = 1$ has been proven, while the existence of $\mathfrak{T}(k)$ for $k \geq 3$ is open; see [2, 1] for the currently best bounds for $k \geq 4$.

Actually a more striking picture is generally believed: In Figure 2 basically the same data as in Figure 1 is shown, but this time all data is entered into one diagram, with the density on the x-axis, while the number of variables is constant for each curve. We see a *critical point* emerging at around density 4.243, with a *critical probability* of around 0.565. The conjecture is, that for every $k \in \mathbb{N}$, $k \geq 2$ such a picture emerges for large n ; the critical density then must be the threshold $\mathfrak{T}(k)$. Moreover it is believed that there are *scaling functions* $\chi_k^{-1} : \mathbb{R} \rightarrow]0, 1[$ such that for “large n ” the following equation is “nearly precise”:

$$\Pr_1(\rho \cdot \Delta_k, n) = \chi_k^{-1}\left(\frac{\rho - \mathfrak{T}(k)}{n^{-1/\nu_k}}\right). \quad (1)$$

ν_k is called the *critical exponent*. See [10] for some general information on this approach, while [16] proves that if ν_k exists, then $\nu_k \geq 2$ must hold; only for the case $k = 2$ this approach has been verified (with $\nu_2 = 3$) in [5]. We remark here, that the critical probability 0.565 experimentally observed for $k = 3$ and small values for n is not too far away from $\frac{1}{2}$, and thus the general (often erroneous) association of the threshold density with the “50% density” is possibly not too far away from the truth.

Now how to get numerical information out of this general approaches? Ignoring the assumption of “large n ”, our best guess at this time is to use (1) for curve fitting. Purely

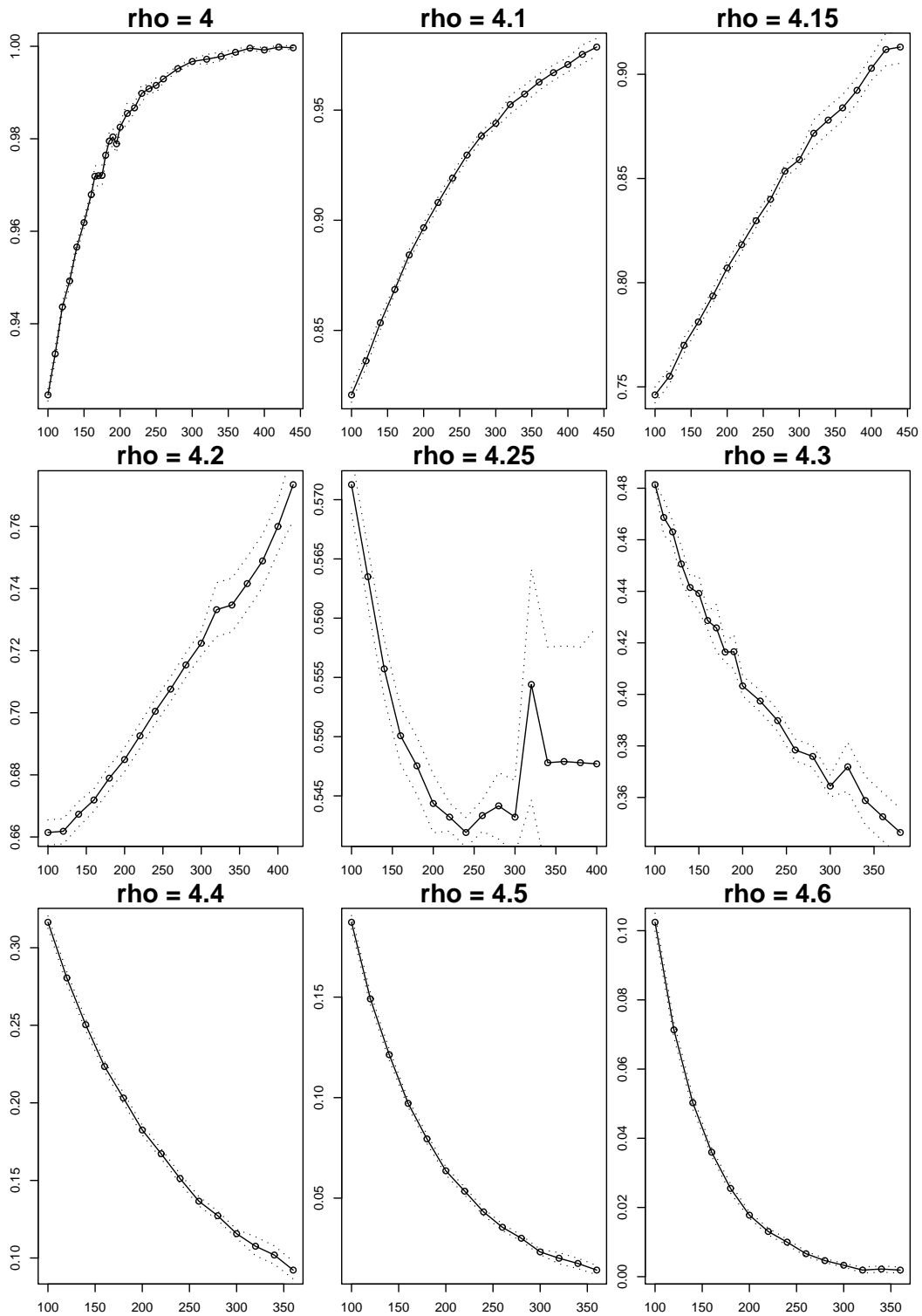


Figure 1. Curves $n \mapsto \Pr_1(\rho \cdot \Delta_3, n)$ for various densities ρ shown on top of each graph. The points are the means of the observations, joined by solid lines, while the dotted lines show the 95% confidence intervals.

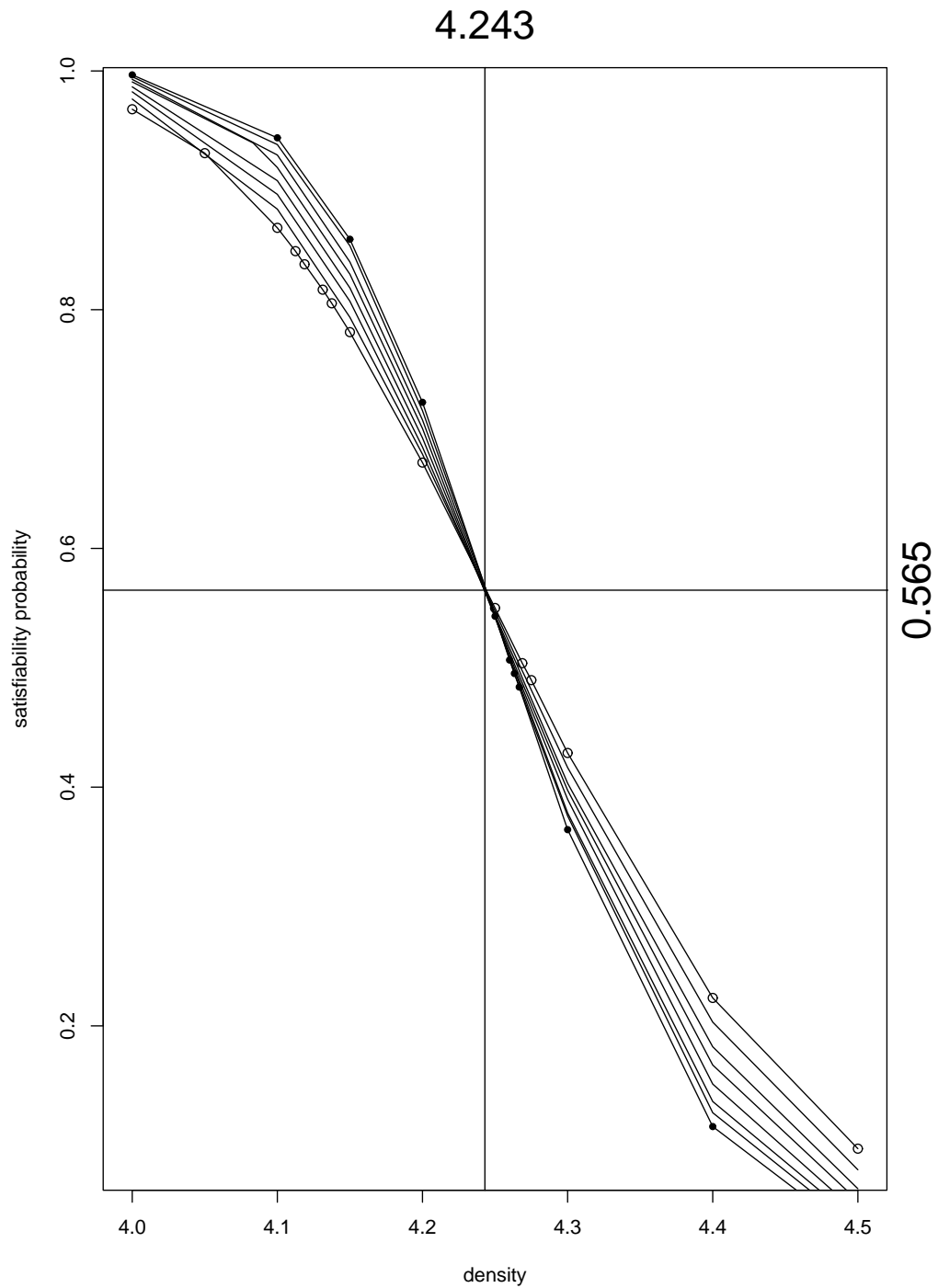


Figure 2. The functions $\rho \mapsto \Pr_1(\rho \cdot \Delta_3, n)$ for $n = 160$ to $n = 300$ in steps of $\Delta n = 20$; for all observations the sample size is at least 50000. The points plotted as empty circles are the observations for the minimal n , while the full circles are the observation for the maximal n ; the approximation for the critical point is obtained by “visual inspection”.

based on numerical considerations, our best guess for the scaling function χ_3 at this time is $\chi_3^{-1}(x) = \frac{1}{2}(1 - \tanh(x)) = \frac{e^{-x}}{e^x + e^{-x}}$. Using a considerable amount of data collected in the `OKdatabase` (see [10] for an early overview), curve fitting yields the following quite reasonable model³:

$$\Pr_1'((x \cdot \Delta_3, n)) = \frac{1}{2} \left(1 - \tanh(0.047 \cdot (x - 4.245) \cdot n^{1/1.11} + 0.155) \right). \quad (2)$$

We see that the threshold-approximation of 4.245 obtained by this approach is quite close to the above “critical density” 4.243; for recent claims coming from statistical physics that $\mathfrak{T}(3) \approx 4.267$ (see [13]) there must be a reversal of direction for larger n , which cannot be seen in the experimental data for the medium size values of n considered.

Stipulating that $\rho \mapsto \Pr_1(\rho \cdot \Delta_k, n)$ is continuous and strictly decreasing (for fixed n), we denote by $\mathfrak{d}_1^{[k]}(p, n)$ for $0 < p < 1$ the unique density ρ with $\Pr_1(\rho \cdot \Delta_k, n) = p$. Model (2) then yields the model

$$\mathfrak{d}_1^{[3]'}(p, n) = 4.245 + 21.1 \cdot (\operatorname{arctanh}(1 - 2p) + 0.155) \cdot n^{-1/1.11} \quad (3)$$

(where $\operatorname{arctanh}$ is the inverse of \tanh). For $p = \frac{1}{2}$ we obtain a reasonable model for the ((in)famous) 50% densities, which are closer examined in the following subsection.

2.2 Approximating the “50%-density”

Here we want to have a closer look at $\mathfrak{d}_1^{[3]}(\frac{1}{2}, n) = \mathfrak{d}_1(\frac{1}{2}, n)$ (following the tradition [8, 7]), the density where for given n the satisfiability probability is $\frac{1}{2}$ (we consider only $k = 3$ here); if n goes to infinity, then $\mathfrak{d}_1(\frac{1}{2}, n)$ tends to $\mathfrak{T}(3)$ (if existent), and, as already remarked, the “magical probability” $\frac{1}{2}$ is not too far away from the experimentally observed critical probability 0.565 (at least for small n). The approach is to compute approximations $\tilde{\mathfrak{d}}_1(0.5, n)$ by statistical sampling, and then to use curve fitting to obtain a model.

According to the definition $\Pr_1(\rho \cdot \Delta_3, n)$ actually is a step function (that is, piecewise constant), and thus can not be directly used for the definition of $\mathfrak{d}_1(p, n)$. Instead, we have to consider $\Pr_1(\rho \cdot \Delta_3, n)$ at $\rho = \frac{m}{n}$ for $m \in \mathbb{N}$, and we have to extend this function continuously so that the resulting function is injective. The natural choice is to use splines of order at least 1. In this article we consider the simplest case of order 1, that is, linear interpolation. So let $\rho_l(p, n)$ (the “left density”) for $0 < p < 1$ and $n \in \mathbb{N}$ be $\frac{m}{n}$ for the maximal $m \in \mathbb{N}$ such that $\Pr_1(\frac{m}{n} \cdot \Delta_3, n) > p$, and let $\rho_r(p, n) := \rho_l(p, n) + \frac{1}{n}$ (the “right density”), while $\tilde{\mathfrak{d}}_1(p, n)$ is the density where the line through the observed frequencies at densities $\rho_l(p, n)$ and $\rho_r(p, n)$ yields the frequency p .

In Table 1 our experimental observations on the computation of $\tilde{\mathfrak{d}}_1(0.5, n)$ are given, while the plotted points in Figure 3 show $\tilde{\mathfrak{d}}_1(0.5, n)$ itself (see also Figure 4 for an enlarged view). The minimal number of variables considered is the minimal sensible value at all, namely $n = 3$. We remark here, that by using the results from [15] we can precisely

3. the residual standard error is 0.014709 on 2707 degrees of freedom (where these observations are averages obtained from several hundred million runs of `OKsolver` on random formulas), and the coefficient of determination between observed and predicted values is $r^2 = 0.9988937$

calculate $\mathfrak{d}_1(0.5, 3) \approx 6.471570641$, where the precise value is rounded correctly to the given precision, validating the statistical data at this one point. See [10] for more information (including statistical quality indicators; an updated report using the extended data set will be made available in due course).

From the general model (1) for $\text{Pr}_1(\rho \cdot \Delta_k, n)$ we obtain the general basic approach

$$\mathfrak{d}_1^{[k]}(p, n) = \mathfrak{T}(k) + \chi_k(p) \cdot n^{-1/\nu_k}. \quad (4)$$

Values for ν_3 from the literature are:

1. [8] $\nu_3 \approx \frac{3}{2} = 1.5$ by finite size scaling;
2. [7] $\nu_3 \approx \frac{5}{3} = 1.\bar{6}$ by fitting a 50% curve;
3. [16] *proves* $\nu_3 \geq 2$ (provided that ν_3 makes sense at all; for our range this result might well not be applicable).

Fixing the critical exponent to one of these three values, we can use linear regression; all three regression curves are shown in Figure 3, and as the reader can see, none of these models gives a reasonable approximation of the observed data (with the smallest critical exponent $\frac{3}{2}$ as the “winner” here).

Instead of fixing the critical exponent, we can fix the approximation for the threshold $\mathfrak{T}(3)$ and use linear regression in the form

$$\log(\mathfrak{d}_1(0.5, n) - \mathfrak{T}(3)) = \log(\chi_3(0.5)) - \frac{1}{\nu_3} \cdot \log(n)$$

to compute $\log(\chi_3(0.5))$ and $-\frac{1}{\nu_3}$. Using the approximation $x_c = 4.245$ (see model (3)) resp. $x_c = 4.243$ (see Figure 2) we obtain the models

$$\mathfrak{d}_1^{(x_c=4.245)}(0.5, n) = 4.245 + 6.42 \cdot n^{-\frac{1}{0.96}}. \quad (5)$$

resp.

$$\mathfrak{d}_1^{(x_c=4.243)}(0.5, n) = 4.243 + 5.97 \cdot n^{-\frac{1}{0.99}}. \quad (6)$$

These models are also displayed in Figure 3, and they provide a better fit; actually they look quite good, but this is deceiving, since the scale on the density axis is very large.

One can argue that considering “too small” values of n spoils the picture, and thus in Figure 4 we only consider data points for $n \geq 100$. The fit indeed now is better; the resulting approximations are as follows:

$$\mathfrak{d}_1^{(\nu_3=\frac{3}{2})}(0.5, n)' = 4.244 + 0.815 \cdot n^{-\frac{1}{1.5}} \quad (7)$$

$$\mathfrak{d}_1^{(\nu_3=\frac{5}{3})}(0.5, n)' = 4.242 + 0.63 \cdot n^{-\frac{1}{1.6}} \quad (8)$$

$$\mathfrak{d}_1^{(\nu_3=2)}(0.5, n)' = 4.237 + 0.438 \cdot n^{-\frac{1}{2}} \quad (9)$$

$$\mathfrak{d}_1^{(x_c=4.245)}(0.5, n)' = 4.245 + 0.5 \cdot n^{-\frac{1}{1.72}} \quad (10)$$

$$\mathfrak{d}_1^{(x_c=4.243)}(0.5, n)' = 4.243 + 0.42 \cdot n^{-\frac{1}{1.87}}. \quad (11)$$

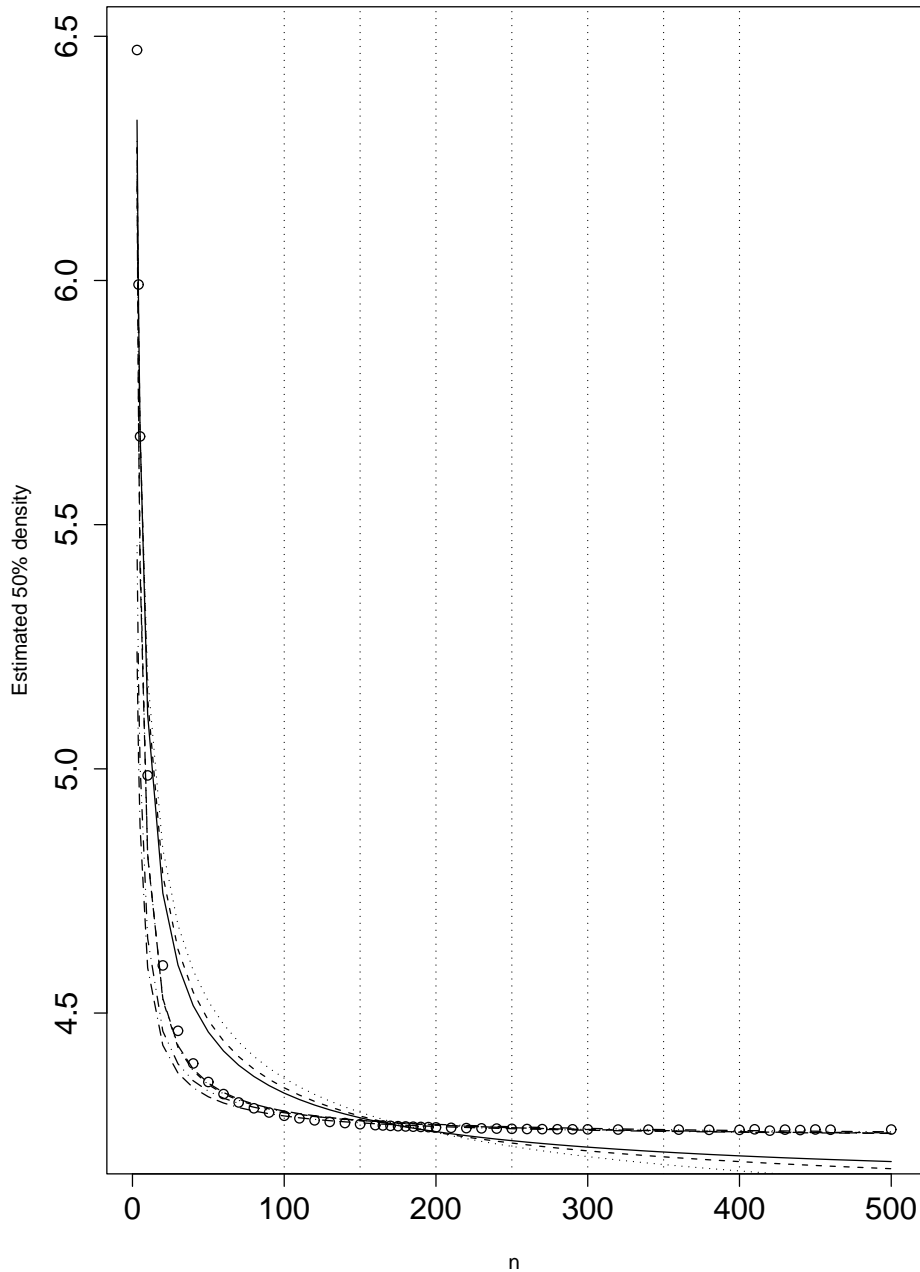


Figure 3. Estimated 50%-satisfiability curve (plotted points calculated from the observed data reported in Table 1) together with seven regression curves. First the three models from the literature (recognisable by their poor fit), shown by a solid line, dashed line, and a dotted lines respectively (with increasing critical exponent). Then the two alternative models (with a better fit; hardly distinguishable): model (5) shown by a dotted-dashed line, and (6) by a long-dashed line. Finally two models derived from global models are shown: Using $p = \frac{1}{2}$ in (3) by a line with two dots followed by a long dash, while the second global model from [10] by a line with two long dashes followed by a dot (these two models offer a reasonable approximation).

Again, model (11) seems to provide the best fit. Finally, taking a purely empirical standpoint and performing non-linear regression (least-squares) for all three parameters $\mathfrak{T}(3), \chi_3(p), \nu_3 \in \mathbb{R}_{>0}$ in model (4); we obtain

$$\mathfrak{d}_1^{(0)}(0.5, n)' = 4.259 + 925.1 \cdot n^{-\frac{1}{0.447}}. \quad (12)$$

This model is shown as the solid line in Figure 4 with the best fit. However also with this method we do not obtain a satisfactory model over the whole range of n -values we considered.

3. The selection of benchmarks

In Table 2 in the appendix the information about the series with *medium size benchmarks* are summarised. For each clause-length $k \in \{3, 5, 7\}$ there are 7 series of benchmarks, each containing 10 benchmarks with a fixed number of (formal) variables and a density near the 50%-density, 5 of them are satisfiable, 5 unsatisfiable (predetermined with `OKsolver`). For clause-size $k = 3$ we have used the data discussed in Subsection 2.2 (with some fluctuations, which we consider to be negligible here).

All instances have been created using `OKgenerator` (see [9, 11]), a strong random formula generator based on AES (the new encryption standard, replacing DES), which allows platform-independent replicable creation of large numbers of large size formulas (also for mixed clause-sizes and for non-boolean variables). In Table 4 for all medium size instances information is given on how to create them, whether they are satisfiable or unsatisfiable, and how successful the solvers were on them. The creation process started with a certain seed, solving all instances until 5 satisfiable and 5 unsatisfiable instances have been obtained, and the first 5 instances have been chosen in each case.⁴

In Table 3 in the appendix the information about the series with *large size benchmarks* are summarised. For each clause-length $k \in \{3, 5, 7\}$ there are 6 series of benchmarks, each containing 10 benchmarks with a fixed number of (formal) variables and a density near but below the 50%-density, so that all of them are likely to be satisfiable (most instances have been solved before the competition, some other instances have been solved during the competition, but there remain for the larger sizes unsolved instances). Again in Table 5 the specifics about the instances are given (how to create them with `OKsolver`, whether they are known to be satisfiable or unsolved, and the success rate of solvers).

4. For the larger instances several keys have been used. This reflects the meaning of the keys (s_0, s_1) , where s_0 in our experimental system (soon to be released as part of the `OKlibrary`) is used to identify a computer, while s_1 is used for the experiment number (both numbers are unsigned 32 bit integers), since to solve the larger instances we run several machines in parallel; this might have introduced some bias towards easier instances, but we believe that the somewhat random fashion in which we run the computers (of different speed) should counteract this tendency (and the large instances where hard enough anyway).

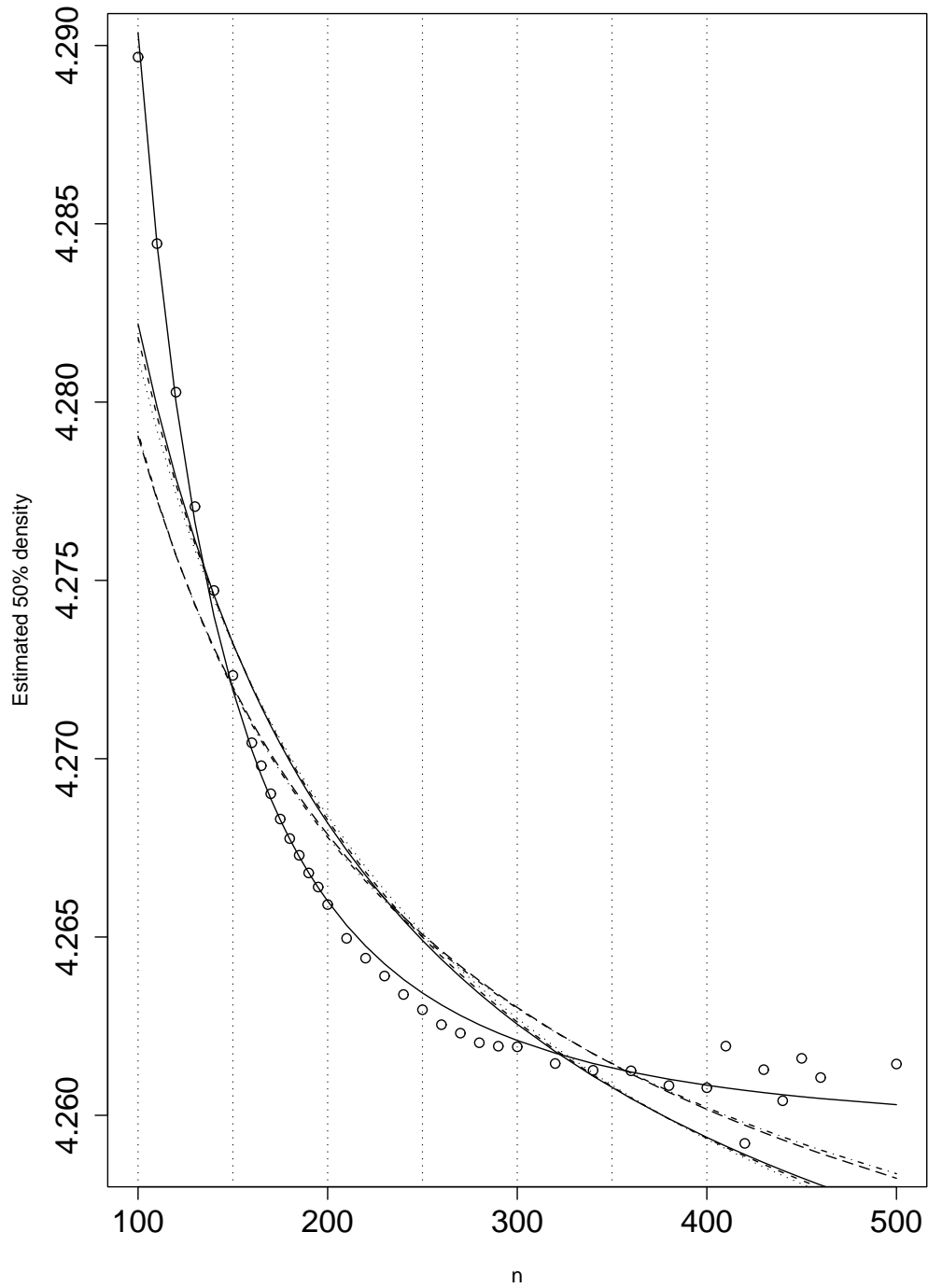


Figure 4. As Figure 3, but evaluated only for $n \geq 100$. Model (7) is shown by a solid line, (8) by a dashed line, and (9) by a dotted line. Then the two alternative models (with a better fit): model (10) shown by a dotted-dashed line, and (11) by a long-dashed line. Finally model (12) shown by a solid line (with the best fit).

4. The evaluation

After giving in Subsection 4.1 a formal definition of the “official” evaluation method (the “scoring method”) of the SAT 2005 competition, in Subsection 4.2 then we discuss the results of the (r-)competition. First the sub-competition on unsatisfiable instances in subsection 4.2.1, then the sub-competition on satisfiable instances in subsection 4.2.2, and finally the sub-competition on all instances in subsection 4.2.3.

We will see that regarding the scoring method in all three sub-competitions we have a clear winner, but when considering the different clause-sizes on its own, then the situation is less clear (for the unsat-sub-competition regarding the second and the third place, for the sat-sub-competition regarding all three places): For each clause-size alone there is always a best solver, but since this solver varies, it is only the “genius” of the scoring method which integrates these partial results into one result. Obviously this is especially the case for the all-instances sub-competition, since it is completely unclear how to compare incomplete and complete solvers.

One can argue that the scoring method makes the competition very unpredictable due to the global nature of the scoring method (how much a solver gains depends also on the other solvers), but it seems to be as close as we can get at this time towards a “fair” method (where the scoring method acts like a kind of Delphian oracle).

4.1 Review of the evaluation method

A **competition** \mathcal{C} for us is a tuple $\mathcal{C} = (S, B, \mathbb{S}, \text{Sb}, \text{time}, \text{timeout}, \text{sat})$, where

- S is a (finite) set; the elements $s \in S$ are called *solvers*.
- B is a (finite) set; the elements $b \in B$ are called *benchmarks*.
- \mathbb{S} is a partition of B (a set of disjoint non-empty subsets of B whose union is B); the elements $S \in \mathbb{S}$ are called *series*'.
- $\text{Sb} \subseteq S \times B$; the elements $(s, b) \in \text{Sb}$ are the pairs of solvers s which *solved* the benchmark b .
- $\text{time} : \text{Sb} \rightarrow \mathbb{R}_{\geq 0}$ with $\text{time} \leq \text{timeout}$; it is $\text{time}(s, b)$ the *running time* of solver s on benchmark b .
- $\text{timeout} : \text{Sb} \rightarrow \mathbb{R}_{> 0}$; $\text{timeout}(s, b)$ is the *time out* of solver s on benchmark b .
- $\text{sat} : B \rightarrow \{0, 1\}$ specifies for each benchmark $b \in B$ whether b is considered to be *satisfiable* (in case of $\text{sat}(b) = 1$) or *unsatisfiable* ($\text{sat}(b) = 0$).

Some remarks:

1. Instead of using real numbers we could use rational numbers.
2. Times are always measured in seconds.

3. Since running times are usually rounded, we allow $\text{time}(s, b) = 0$ for cases where the running time is rounded to zero.
4. $\text{timeout}(s, b)$ is the total time available to solver s when it solved benchmark b ; for the SAT 2005 competition in the sub-competition on random instances we have $\text{timeout}(s, b) = 1200$ (20 minutes) if solver s solved benchmark b in round 1, and $\text{timeout}(s, b) = 6000$ (100 minutes) if s solved b in round 2 (there are no other possibilities).
5. As usual for relations, we use $\text{Sb}(s)$ for the set of benchmarks solver s solved, and $\text{Sb}^{-1}(b)$ for the set of solvers which solved benchmark b .
6. Usually before the competition starts the map $\text{sat} : B \rightarrow \{0, 1\}$ is only partially defined; during the competition some new values for $\text{sat}(b)$ may be added, and also possibly some values are redefined (or un-defined). However, for the evaluation finally every benchmark (or “problem instance”) has to be categorised. For the SAT 2005 competition all medium size random instances have been sat-decided in advance, while all large size random instances are considered to be satisfiable (for most of these instances this was verified in advance, while for the remaining instances the choice of the densities should guarantee that they are satisfiable with very high probability⁵).

A competition \mathcal{C} determines three **sub-competitions**:

- $\mathcal{C}_{0,1} = \mathcal{C}$ is the *all-instances competition*;
- \mathcal{C}_0 is the *only-unsatisfiable-instances competition*, obtained from \mathcal{C} by restricting B to $B_0 := \text{sat}^{-1}(0)$, keeping S , and restricting the other components of \mathcal{C} accordingly.
- \mathcal{C}_1 is the *only-satisfiable-instances competition*, obtained from \mathcal{C} by restricting B to $B_1 := \text{sat}^{-1}(1)$, keeping S , and restricting the other components of \mathcal{C} accordingly.

We emphasise that for the sub-competitions only benchmarks were discarded, not solvers; thus complete solvers participate also at the only-satisfiable-instances competition (though with low chances), and the incomplete solvers participate also at the only-unsatisfiable-instances competition (though at this time, where no incomplete solver was able to handle unsatisfiable instances, they all ended up not solving any instance here). The evaluation process is defined for arbitrary competitions \mathcal{C} ; applied to the sub-competitions $\mathcal{C}_{0,1}, \mathcal{C}_0, \mathcal{C}_1$ one obtains the ranking for the all-instances (sub-)competition, only-unsatisfiable-instances (sub-)competition and only-satisfiable-instances (sub-)competition respectively.

The **evaluation parameters** are given by the tuple $(\text{stpp}, \text{stspf}, \text{stsf}, \text{tss}, \text{fss})$, where

- $\text{stpp} \in \mathbb{R}_{\geq 0}$ is the *standard problem purse*. The value used for the SAT 2005 competition is $\text{stpp} = 1000$. (This value serves only as a standardisation factor, so that one gets “pleasant numbers”.)

5. though it seems, that current knowledge does not allow to make any more precise statements here

- $\text{stspf} \in \mathbb{R}_{\geq 0}$ is the *standard speed factor*; we define the *standard speed purse* as $\text{stsp} := \text{stspf} \cdot \text{stpp}$. The value used for the SAT 2005 competition is $\text{stspf} = 1$, and thus $\text{stsp} = 1000$. (As a result, problem purse and speed purse all in all have equal importance.)
- $\text{stsf} \in \mathbb{R}_{\geq 0}$ is the *standard series factor*; we define the *standard series purse* as $\text{stsp} := \text{stsf} \cdot \text{stpp}$. The value used for the SAT 2005 competition is $\text{stsf} = 3$, and thus $\text{stsp} = 3000$. (We will see in Tables 8 and 7, that due to this choice the series purse is less important than each of problem purse and speed purse.)
- $\text{tss} \in \mathbb{N}$ is the *threshold for series size*. For the SAT 2005 competition we have $\text{tss} = 5$.
- $\text{fss} \in \mathbb{R}_{\geq 0}$ is the *factor for small series'*. For the SAT 2005 competition we have $\text{fss} = \frac{1}{3}$.

Given a competition \mathcal{C} and an evaluation parameter tuple e , we define in the following the scoring function $\text{score}_{\mathcal{C},e} : S \rightarrow \mathbb{R}_{\geq 0}$ which yields a ranking of the solvers (the higher the score, the better the solver). The main equation is

$$\text{score}(s) := \text{pp}(s) + \text{spp}(s) + \text{sp}(s), \quad (13)$$

- $\text{pp} : S \rightarrow \mathbb{R}_{\geq 0}$ gives the *problem purse* of solver s , defined as

$$\text{pp}(s) := \sum_{b \in \text{Sb}(s)} \frac{\text{stpp}}{|\text{Sb}^{-1}(b)|}.$$

- spp gives the *speed purse* of solver s , defined as

$$\text{spp}(s) := \sum_{b \in \text{Sb}(s)} \text{stsp} \cdot \frac{\text{spf}(s, b)}{\sum_{s' \in \text{Sb}^{-1}(b)} \text{spf}(s', b)},$$

where for $(s, b) \in \text{Sb}$ the *speed factor* $\text{spf}(s, b)$ is defined as

$$\text{spf}(s, b) := \frac{\text{timeout}(s, b)}{1 + \text{time}(s, b)}.$$

- Finally let $\text{Ss} := \{(s, S) \in S \times \mathbb{S} : \text{Sb}(s) \cap S \neq \emptyset\}$ be the relation between solver and series' expressing that the solver solved at least one instance of the series (so that $\text{Ss}(s)$ is the set of series' solved by solver s , while $\text{Ss}^{-1}(S)$ is the set of solvers which solved series S). Now

$$\text{sp}(s) := \sum_{S \in \text{Ss}(s)} \frac{\text{sp}(S)}{|\text{Ss}^{-1}(S)|},$$

where for $S \in \mathbb{S}$ we set the series purse

$$\text{sp}(S) := \begin{cases} \text{stsp} & \text{if } |S| \geq \text{tss} \\ \text{fss} \cdot \text{stsp} & \text{if } |S| < \text{tss} \end{cases}.$$

We remark here, that since every series of medium size random instances has exactly 5 satisfiable and 5 unsatisfiable instances, while every large size random series has exactly 10 satisfiable instances (and 0 unsatisfiable), for all three sub-competitions the speed purse of every series is always 3000 (since the second case in the definition of $\text{sp}(S)$ here never applies).

The sum over all problem purses of all solvers equals the sum over all speed purses (due to $\text{stspf} = 1$), namely

$$\sum_{s \in S} \text{pp}(s) = \sum_{s \in S} \text{spp}(s) = |\text{Sb}(S)| \cdot \text{stpp},$$

where $|\text{Sb}(S)|$ is the number of benchmarks solved altogether. The sum over all series purse is given by

$$\sum_{s \in S} \text{sp}(s) = \sum_{S \in \mathbb{S}} \text{sp}(S).$$

A final remark on the choice of the evaluation parameters: As we will see in the sequel, fortunately the winners of the three sub-competitions turn out to be independent of the choice of these parameters, however the second and the third places can be altered for all three sub-competitions by reducing the standard speed factor and/or increasing the standard series factor. So the final outcome contains an element of arbitrariness, but this is hardly surprising, and furthermore quite big changes are needed to change ranks two and below. In the future a sensitivity analysis might be appropriate when deciding on the final result and the final values of the evaluation parameters, but in this article we stick to the given values.

4.2 The results

4.2.1 UNSATISFIABLE INSTANCES ONLY

Table 8 gives the scores of the solvers on unsatisfiable instances. The clear winner (not only with the total score, but also in all three partial scores) is `kcnfs-2004`, the winner of last year's competition. On the second place we have `march_dl`, and on the third place `Dew.Satz_1a`. `march_dl` has a slightly worse problem purse than `Dew.Satz_1a`, a slightly better series purse, and a much better speed purse. The fourth-place solver `wllsatv1` is worse than these three solvers in all partial scores.

The series solved by the first three solvers:

1. `kcnfs-2004` for clause-sizes $k =$

3 : $n = 300, 360, 400, 450, 500$

5 : $n = 70, 80, 90, 100, 110$

7 : $n = 45, 50, 55, 60, 65$

2. `march_dl` for clause-sizes $k =$

3 : $n = 300, 360, 400, 450, 500$

5 : $n = 70, 80, 90$

7 : $n = 45$

3. `Dew_Satz_1a` for clause-sizes $k =$

3 : $n = 300, 360, 400$

5 : $n = 70, 80, 90, 100$

7 : $n = 45, 50, 55$

We see that `march_dl` has problems with higher clause-sizes (likely due to the fact, that all clauses are translated into 3-clauses in the preprocessing phase), but is comparable to `kcnfs-2004` for clause-size 3. A closer look at the performance per clause-length is given in Tables 9, 10 and 11, where we give the details of the behaviour of all solvers on unsatisfiable instances for (fixed) $k = 3, 5, 7$, ranked lexicographically as follows: Each solver gets assigned a 14-tuple of numbers, where the first 7 numbers give the number of instances not solved in the series with $n = 600, \dots, 300$, while the last 7 entries give the average running time of the solver on the instances it solved in the respective series (“ $+\infty$ ” if the solver didn’t solve the instance); now a solver is ranked higher than another solver if its associated vector is lexicographically smaller than the associated vector of the other solver.

We see again, that `kcnfs-2004` is superior for all three clause-lengths; furthermore `kcnfs-2004` and `march_dl` are far superior over all other solvers on clause-length 3, both in speed and solving capability, and the high speed purse `march_dl` got for $k = 3$ over-compensated its poor results for $k = 5, 7$.

4.2.2 SATISFIABLE INSTANCES ONLY

Table 7 gives the scores of the solvers on satisfiable instances. The clear winner (also in all three partial scores) is `ranov`. The second place takes `g2wsat`, and the third place `vw` (`g2wsat` has a slightly better problem purse, a slightly worse series purse, and a much better speed purse). The fourth-place solver `rpaws10` is clearly worse than `vw` regarding problem and series purse (but somewhat better than `vw` regarding the speed purse — it seems `vw` carries its name for good reasons).

The series solved (all three solvers solved all medium size series, so they are not reported here):

1. `ranov` for clause-sizes $k =$

3 : $n = 2000, 4000, 6000, 8000, 10000$

5 : $n = 500, 600, 700, 800, 900, 1000$

7 : $n = 120, 140, 160, 200$

2. `g2wsat` for clause-sizes $k =$

3 : $n = 2000, 4000, 6000, 8000, 10000, 12000$

5 : $n = 500$

7 : $n = 120, 140, 160, 180$

3. **vw** for clause-sizes $k =$

3 : $n = 2000, 4000, 6000, 8000, 10000, 12000$

5 : $n = 500, 600, 700, 800$

7 : $n = 120, 140$

We see that **ranov** misses out for $(k, n) = (3, 12000)$, while **g2wsat** shows disastrous performance for $k = 5$. As before, in Tables 12, 13 and 14 we give the details of the behaviour of all solvers on unsatisfiable instances for (fixed) $k = 3, 5, 7$, ranked lexicographically. We see that **vw**, **g2wsat** and **rpaws10** are somewhat better than **ranov** for $k = 3$, but that **ranov** (and also **vw**, but to a lesser degree) got especially high problem and series purses for $k = 5$, and that **ranov** is also superior for $k = 7$, earning high problem scores.

4.2.3 ALL INSTANCES

In Table 6 we show the scores of the solvers in the all-instances sub-competition. We see that the scores of the leading incomplete solvers (which solved only satisfiable instances) are invariant (are the same as in the only-sat instances sub-competition), while the scores of complete solvers change: to their problem and speed purses the purses earned on satisfiable instances are added, while the series purses get diminished since all the series they solved are solved easily by incomplete solvers. The two leading incomplete solvers come first, and then comes the leading complete solver (who is especially strong regarding the speed-purse).

Now it was decided that only “complete solvers” (meaning that they solved at least one unsatisfiable random instance) are eligible for an official ranking here, and so only the framed solvers in Table 6 actually were officially participating in this sub-competition. The ranking of these solvers is nearly identical to the ranking in the unsatisfiable-instances-only, only places 7 and 8 switched. Please note that the scores on satisfiable instances are computed in the context of the whole competition (not just considering the “complete” solvers, but also the local search solvers).

5. Conclusion

5.1 Resume on this year’s competition

Regarding the set-up of the competition the two main targets have been fulfilled:

- The basic selection of the series’ and benchmarks turned out to be suitable.
- The evaluation method has selected a ranking which seems to be fair.

Regarding the solver, on the unsatisfiability front not much has been gained, but new strong solvers showed up on the satisfiability front.

5.2 The next competition in 2007

Considering the selection of benchmarks and series, it seems reasonable to extend the choice to include also at least the mixed 2,3-CNF case. Whether also other types of random formulas should be considered (for example with variable clause-size, or “planted” satisfiable instances) is not clear to me: In the SAT 2003 r-competition (see [3]) quite different types of random benchmarks were considered, but due to the very different natures of these benchmark types in the following year’s competition (see [4]) it was decided to use only uniform random instances.

How the scoring method will work out next time is hard to say — due to its global nature the method is sensitive against formation of “cliques” among the solvers, but it might be that it is not easy to misuse the system. It seems reasonable that the scoring system for the all-instances sub-competition should consider only eligible solvers (different from this competition, where all solvers have been taken into account for the scores). This year the influence of the diversity of series solved was considerably smaller than the influence of actual instances solved; one might thus consider to increase the standard series factor (which would be in line with previous SAT competitions).

More research must go into the quantitative understanding of the phase transition, extending the results from Section 2 to bigger numbers of variables and larger clause-sizes. It is conceivable that the choice of densities for the large size instances had an effect on the competition, and that with a different setting a different outcome would have resulted (using higher densities, making the instances harder); using the SP-algorithm ([6]) might allow to move closer to the 50%-densities). Regarding the basic set-up the following recommendations seem reasonable to me:

1. $k = 3$:
 - (a) Medium size: One could still use the same parameter value (roughly adding 20 variables doubles the running time on unsatisfiable instances); in case one has special indication about progress perhaps one could shift the set from 300, . . . , 600 to 350, . . . , 650.
 - (b) Large size: I propose $n = 4000, 7000, 10000, 13000, 16000, 19000$.
2. $k = 5$:
 - (a) Medium size: Could also be repeated (the densities could be fine-tuned, but likely this has no big effect).
 - (b) Large size: Perhaps a slight increase, say, $n = 600, 700, 800, 900, 1000, 1100$.
3. $k = 7$:
 - (a) Medium size: As for $k = 5$.
 - (b) Large size: Perhaps a slight increase, say, $n = 130, 150, 170, 190, 210, 230$.

References

- [1] Dimitris Achlioptas, Assaf Naor, and Yuval Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, **435**:759–764, June 2005.
- [2] Dimitris Achlioptas and Yuval Peres. The threshold for random k -SAT is $2^k \log 2 - O(k)$. *Journal of the American Mathematical Society*, **17**(4):947–973, 2004.
- [3] Daniel Le Berre and Laurent Simon. The essentials of the SAT 2003 competition. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing 2003*, volume **2919** of *Lecture Notes in Computer Science*, pages 452–467, Berlin, 2004. Springer. ISBN 3-540-20851-8.
- [4] Daniel Le Berre and Laurent Simon. Fifty-five solvers in Vancouver: The SAT 2004 competition. In Holger H. Hoos and David G. Mitchell, editors, *Theory and Applications of Satisfiability Testing 2004*, volume **3542** of *Lecture Notes in Computer Science*, Berlin, 2005. Springer.
- [5] Béla Bollobás, Christian Borgs, Jennifer T. Chayes, Jeong Han Kim, and David B. Wilson. The scaling window of the 2-SAT transition. *Random Structures and Algorithms*, **18**(3):210–256, 2001.
- [6] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, **27**(2):201–226, March 2005.
- [7] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3SAT. *Artificial Intelligence*, **81**(1-2):31–57, April 1996.
- [8] Scott Kirkpatrick and Bart Selman. Critical behaviour in the satisfiability of random boolean expressions. *Science*, **264**:1297–1301, 1994.
- [9] Oliver Kullmann. First report on an adaptive density based branching rule for DLL-like SAT solvers, using a database for 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, March 2002. Extended version of [12].
- [10] Oliver Kullmann. Investigating the behaviour of a SAT solver on random formulas. Technical Report CSR 23-2002, University of Wales Swansea, Computer Science Report Series, October 2002. 119 pages.
- [11] Oliver Kullmann. The random formula generator OKgenerator. Available at <http://cs-svr1.swan.ac.uk/~csoliver/OKgenerator.html>, February 2002. Implemented in C++; Version 1.3, Release 1.0.
- [12] Oliver Kullmann. Towards an adaptive density based branching rule for SAT solvers, using a database for mixed random conjunctive normal forms built upon the advanced encryption standard (AES). In John Franco, Henry Kautz, Hans Kleine Büning, Hans

van Maaren, Bart Selman, and Ewald Speckenmeyer, editors, *Fifth International Symposium on Theory and Applications of Satisfiability Testing*, pages 190–200, 2002. University of Cincinnati (USA), May 6, 2002 to May 9, 2002.

- [13] Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k -SAT from the cavity method. *Random Structures and Algorithms*, 2006. In press.
- [14] Rémi Monasson, Ricardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. $(2 + p)$ -SAT: Relation of typical-case complexity to the nature of the phase transition. *Random Structures and Algorithms*, **15**(3-4):414–435, 1999.
- [15] R.C. Monson. *The Computer-Aided Verification of Mathematical Reasoning in Education and the Counting of Satisfiable Instances of k -SAT*. PhD thesis, School of Mathematics and Statistics, The University of Western Australia, February 2004.
- [16] David Wilson. On the critical exponents of random k -SAT. *Random Structures and Algorithms*, **21**(2):182–195, August 2002.

Appendix A. Experimental data on the satisfiability threshold

Table 1: Estimated 50% densities $\tilde{\mathfrak{d}}_1(0.5, n)$ together with the probability of the event E , that for the true 50% density $\mathfrak{d}_1(0.5, n)$ we have $\tilde{\rho}_1(0.5, n) < \mathfrak{d}_1(0.5, n) \leq \tilde{\rho}_1(0.5, n)$, using the fixed estimations of $\rho_1(0.5, n)$ and $\rho_r(0.5, n)$ given in the second column (together with the corresponding clause numbers); this probability is also the probability for $\rho_1(p, n) = \tilde{\rho}_1(p, n)$ using the fixed observed “left” density (for all statistical computations the R system has been used, and values “1” just mean that for the given machine precision (on a 64-bit computer) the result is deemed by R to be indistinguishable from 1). In the third column one finds the 95% confidence intervals for $\text{Pr}_1(\rho, n)$ where $\rho \in \{\tilde{\rho}_1(0.5, n), \tilde{\rho}_r(0.5, n)\}$; the observed proportion $\tilde{\text{Pr}}_1(\rho, n)$ for density $\rho_1(0.5, n)$ resp. $\rho_r(0.5, n)$ is the centre of the respective confidence interval, while the sample sizes are given in the fourth column.

n	$\begin{matrix} \tilde{\rho}_1(n) & c_1(n) \\ \tilde{\rho}_r(n) & c_r(n) \end{matrix}$	$\text{CI}_{95\%}(\text{Pr}_1(\tilde{\rho}_1(n), n))$	N	$\tilde{\mathfrak{d}}_1(0.5, n)$	$\text{Pr}(E)$
3	6.333333 19	[0.5215789,0.5218531]	51,000,000	6.471917	1
	6.666667 20	[0.469346,0.46962]	51,000,000		
4	5.75 23	[0.547644,0.5495957]	1,000,000	5.991679	1
	6 24	[0.4973455,0.4993065]	1,000,000		
5	5.6 28	[0.5177162,0.5196757]	1,000,000	5.680624	1
	5.8 29	[0.4713391,0.4732971]	1,000,000		
10	4.9 49	[0.5326916,0.5346482]	1,000,000	4.986589	1
	5 50	[0.4938046,0.4957654]	1,000,000		
20	4.55 91	[0.527624,0.5295818]	1,000,000	4.597499	1
	4.6 92	[0.4975135,0.4994745]	1,000,000		
30	4.433333 133	[0.5222545,0.5242134]	1,000,000	4.463604	1
	4.466667 134	[0.4966685,0.4986295]	1,000,000		
40	4.375 175	[0.5206394,0.5225985]	1,000,000	4.396791	1
	4.4 176	[0.4958355,0.4977965]	1,000,000		
50	4.34 217	[0.5208574,0.5228165]	1,000,000	4.358825	1
	4.36 218	[0.4976565,0.4996175]	1,000,000		
60	4.333333 260	[0.5002668,0.5016532]	2,000,000	4.334077	1
	4.35 261	[0.4784634,0.4804227]	1,000,000		
70	4.314286 302	[0.5025705,0.5045314]	1,000,000	4.316934	1
	4.328571 303	[0.483419,0.4853790]	1,000,000		
80	4.3 344	[0.5072791,0.5078989]	10,000,000	4.30497	1
	4.3125 345	[0.4881931,0.4888129]	10,000,000		
90	4.288889 386	[0.5122839,0.5127221]	20,000,000	4.296335	1
	4.3 387	[0.4936259,0.4940641]	20,000,000		
100	4.28 428	[0.5163577,0.517734]	2,027,000	4.289676	1
	4.29 429	[0.4992249,0.4996331]	23,062,000		
110	4.281818 471	[0.5046035,0.5056605]	3,440,000	4.284444	1
	4.290909 472	[0.4868249,0.4878973]	3,340,000		
120	4.275 513	[0.5100886,0.5110394]	4,250,000	4.280278	1
	4.283333 514	[0.4934154,0.4943546]	4,357,000		
130	4.276923 556	[0.5001457,0.5004744]	35,560,000	4.277069	1
	4.284615 557	[0.4832432,0.4846289]	2,000,000		
140	4.271429 598	[0.5066431,0.5085779]	1,027,000	4.274717	1
	4.278571 599	[0.4901106,0.4920453]	1,027,000		
150	4.266667 640	[0.5126888,0.5146491]	1,000,000	4.272334	1
	4.273333 641	[0.4966204,0.498562]	1,020,000		
160	4.26875 683	[0.5031723,0.5045501]	2,025,000	4.270447	1
	4.275 684	[0.4889535,0.4903327]	2,020,000		
165	4.266667 704	[0.5066583,0.5080377]	2,020,000	4.269803	1
	4.272727 705	[0.4924623,0.49384]	2,025,000		

Continued on next page...

Table 1.– continued from previous page

n	$\frac{\tilde{\rho}_l(n)}{\tilde{\rho}_r(n)} \ c_l(n)$ $c_r(n)$	$CI_{95\%}(\text{Pr}_1(\frac{\tilde{\rho}_l(n)}{\tilde{\rho}_r(n)}, n))$	N	$\tilde{d}_1(0.5, n)$	$\text{Pr}(E)$
170	4.264706 725 4.270588 726	[0.5102851,0.5116643] [0.4953123,0.4966897]	2,020,000 2,026,000	4.269017	1
175	4.262857 746 4.268571 747	[0.5138172,0.5151911] [0.4989044,0.4996978]	2,035,000 6,105,000	4.268309	1
180	4.266667 768 4.272222 769	[0.5021189,0.503485] [0.4878875,0.4892664]	2,060,000 2,021,000	4.267761	1
185	4.264865 789 4.27027 790	[0.5061181,0.5072461] [0.4912335,0.4923606]	3,020,000 3,025,000	4.267291	1
190	4.263158 810 4.268421 811	[0.5088097,0.510751] [0.4946866,0.4965814]	1,020,000 1,071,000	4.266797	1
195	4.261538 831 4.266667 832	[0.5133466,0.5145677] [0.4988026,0.4996775]	2,576,030 5,020,000	4.266402	1
200	4.265 853 4.27 854	[0.5019885,0.503368] [0.4872794,0.4886565]	2,020,000 2,026,000	4.26591	1
210	4.261905 895 4.266667 896	[0.507771,0.5099363] [0.4939819,0.4961474]	820,000 820,000	4.264962	1
220	4.263636 938 4.268182 939	[0.501217,0.5032184] [0.4880582,0.4902155]	960,000 826,000	4.264407	1
230	4.260870 980 4.265217 981	[0.5080306,0.5101841] [0.4950474,0.4970918]	829,000 920,000	4.263907	1
240	4.2625 1023 4.266667 1024	[0.5015623,0.5039406] [0.488656,0.4910165]	680,000 690,000	4.263388	1
250	4.26 1065 4.264 1066	[0.5087945,0.5111533] [0.4953167,0.4976776]	691,000 690,000	4.26296	1
260	4.261538 1108 4.265385 1109	[0.5017549,0.504306] [0.4901354,0.4926884]	591,000 590,000	4.262542	1
270	4.259259 1150 4.262963 1151	[0.5095096,0.5118531] [0.4965166,0.4988606]	700,000 700,000	4.262304	1
280	4.260714 1193 4.264286 1194	[0.5038077,0.5055865] [0.4910922,0.492901]	1,215,000 1,175,000	4.262035	1
290	4.258621 1235 4.262069 1236	[0.5112767,0.5133432] [0.498948,0.50008]	900,000 3,000,000	4.261938	0.9996202
300	4.26 1278 4.263333 1279	[0.5053147,0.5077016] [0.4939959,0.4964382]	675,000 644,788	4.261921	1
320	4.259375 1363 4.2625 1364	[0.5057446,0.5081959] [0.4952544,0.49772]	640,000 632,685	4.261453	1
340	4.258824 1448 4.261765 1449	[0.5072184,0.5098333] [0.4969247,0.4995591]	562,402 554,277	4.261262	1
360	4.261111 1534 4.263889 1535	[0.4989333,0.5023748] [0.4855432,0.4892179]	324,911 284,835	4.261248	0.9320379
380	4.260526 1619 4.263158 1620	[0.4991427,0.5031961] [0.4862026,0.4960579]	234,298 39,739	4.260833	0.9881939
400	4.26 1704 4.2625 1705	[0.499769,0.5094429] [0.4834076,0.4960187]	41,248 24,300	4.260773	0.9999086
410	4.260976 1747 4.263415 1748	[0.498677,0.5131935] [0.4814191,0.500455]	18,362 10,700	4.261941	0.9992649
420	4.257143 1788 4.259524 1789	[0.5054126,0.5261889] [0.4856234,0.5096336]	8,984 6,744	4.259213	0.7820353
430	4.251163 1828 4.262791 1833	[0.537655,0.5794126] [0.4802375,0.5022689]	2,218 8,000	4.26128	0.9991266
440	4.259091 1874 4.261364 1875	[0.492549,0.539765] [0.4668287,0.5097294]	1,761 2,130	4.260408	0.9815843
450	4.26 1917 4.262222 1918	[0.5050388,0.5561743] [0.4624155,0.5136316]	1,500 1,500	4.261597	0.9684218
460	4.260870 1960 4.263043 1961	[0.4667499,0.5369631] [0.4256997,0.5356114]	805 333	4.261059	0.5383751
500	4.26 2130 4.262 2131	[0.4707095,0.5749885] [0.4129634,0.569397]	369 167	4.261439	0.6518139
600	4.261667 2557 6.56 3936	[0.3152781,0.7694221] [0,0.002377092]	20 1,550	4.470606	0.8052274

Appendix B. The instances of the SAT 2005 competition

Table 2. Series with medium size benchmarks; k is clause-length, n the number of variables, c the number of clauses, $\rho = \frac{c}{n}$ the density. In the last two columns the number of solvers which have solved the satisfiable respectively unsatisfiable sub-series (that is, have solved at least one satisfiable resp. unsatisfiable instance) is shown together with the resulting series purse (the total series purse is 3000).

k	n	c	ρ	name	purse sat	purse unsat
3	300	1279	4.26 $\bar{3}$	k3-r4.263-v300	29 103.4	11 272.7
	360	1534	4.26 $\bar{1}$	k3-r4.26-v360	27 111.1	8 375
	400	1704	4.26	k3-r4.26-v400	23 130.4	8 375
	450	1913	4.25 $\bar{1}$	k3-r4.25-v450	13 230.8	2 1500
	500	2130	4.26	k3-r4.26-v500	14 214.3	2 1500
	550	2343	4.26	k3-r4.26-v550	9 333.3	0
	600	2557	4.261 $\bar{6}$	k3-r4.261-v600	12 250	0
5	70	1491	21.3	k5-r21.3-v7	25 120	10 300
	80	1704	21.3	k5-r21.3-v80	20 150	8 375
	90	1917	21.3	k5-r21.3-v90	16 187.5	6 500
	100	2130	21.3	k5-r21.3-v100	13 230.8	2 1500
	110	2343	21.3	k5-r21.3-v110	11 272.7	1 3000
	120	2556	21.3	k5-r21.3-v120	8 375	0
	130	2769	21.3	k5-r21.3-v130	11 272.7	0
7	45	4005	89	k7-r89-v45	24 125	9 333.3
	50	4450	89	k7-r89-v50	18 166.7	7 428.6
	55	4895	89	k7-r89-v55	15 200	3 1000
	60	5340	89	k7-r89-v60	13 230.8	1 3000
	65	5785	89	k7-r89-v65	14 214.3	1 3000
	70	6230	89	k7-r89-v70	8 375	0
	75	6675	89	k7-r89-v75	9 333.3	0

Table 3. Series with large size benchmarks; k is clause-length, n the number of variables, c the number of clauses, $\rho = \frac{c}{n}$ the density. In the last column the number of solvers which have solved the series (that is, have solved at least one instance) is shown together with the resulting series purse (the total series purse is 3000; recall that all instances here are (supposed to be) satisfiable).

k	n	c	ρ	name	purse	sat
3	2000	8400	4.2	k3-r4.2-v2000	6	500
	4000	16800	4.2	k3-r4.2-v4000	5	600
	6000	25200	4.2	k3-r4.2-v6000	5	600
	8000	33600	4.2	k3-r4.2-v8000	4	750
	10000	42000	4.2	k3-r4.2-v10000	3	1000
	12000	50400	4.2	k3-r4.2-v12000	3	1000
5	500	10000	20	k5-r20-v500	5	600
	600	12000	20	k5-r20-v600	2	1500
	700	14000	20	k5-r20-v700	2	1500
	800	16000	20	k5-r20-v800	2	1500
	900	18000	20	k5-r20-v900	1	3000
	1000	20000	20	k5-r20-v1000	1	3000
7	120	10200	85	k7-r85-v120	6	500
	140	11900	85	k7-r85-v140	5	600
	160	13600	85	k7-r85-v160	4	750
	180	15300	85	k7-r85-v180	1	3000
	200	17000	85	k7-r85-v200	1	3000
	220	18700	85	k7-r85-v220	0	

Table 4: The medium size benchmarks of the r-competition in detail. Using the first four columns, the instances can be created from `OKgenerator`; consider for example the very first row: The (satisfiable) instance with the r-competition name “bench1902” has been created by the call “`OK-generator s0=1 s1=21 n=300 l=3 cp=1279 nr0=0 nr1=0 -D -o.cnf`” (looking up the number of clauses in Table 2); s_0 is the computer-identification, s_1 the experiment-identification, while (nr_0, nr_1) is the counter (and “-D” switches to Dimacs-output (default is XML), while “-o” generates the output). In the column “problem-p” the number of solvers which have solved the instance is reported together with the resulting problem purse, while the last column reports the average time used.

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p	μ time
3	300	(1, 21)	(0, 0)	1902	1	25 40.0	50.5
			(0, 1)	1904	0	10 100.0	128.4
			(0, 2)	1906	1	25 40.0	11.7
			(0, 3)	1908	0	10 100.0	90.2
			(0, 4)	1909	0	11 90.9	168.2
			(0, 5)	1910	0	9 111.1	115.3

Continued on next page...

Table 4.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
			(0, 6)	1911	0	10	100.0	145.0
			(0, 7)	1903	1	20	50.0	81.1
			(0, 8)	1905	1	29	34.5	4.1
			(0, 9)	1907	1	25	40.0	81.3
	360	(14, 1)	(0, 0)	1814	1	25	40.0	129.3
			(0, 1)	1816	1	26	38.5	58.4
			(0, 2)	1818	1	19	52.6	88.6
			(0, 3)	1819	0	8	125.0	565.6
			(0, 4)	1820	0	8	125.0	276.1
			(0, 5)	1821	1	17	58.8	179.5
			(0, 6)	1812	1	20	50.0	118.0
			(0, 7)	1815	0	8	125.0	793.8
			(0, 8)	1817	0	8	125.0	469.9
			(0, 10)	1813	0	8	125.0	499.2
	400	(1, 55)	(0, 0)	1824	0	7	142.9	2297.4
			(0, 1)	1826	0	4	250.0	1470.7
			(0, 2)	1829	0	8	125.0	1736.1
			(0, 3)	1831	0	4	250.0	1636.2
			(0, 4)	1822	1	16	62.5	197.3
			(0, 5)	1823	0	8	125.0	1059.9
			(0, 8)	1827	1	17	58.8	107.0
			(0, 9)	1830	1	19	52.6	218.9
			(0, 11)	1825	1	23	43.5	56.0
			(0, 12)	1828	1	19	52.6	301.6
	450	(0, 0)	(0, 0)	1807	1	11	90.9	876.8
			(0, 1)	1809	1	10	100.0	204.3
			(0, 2)	1811	0	2	500.0	555.1
			(0, 3)	1802	0	2	500.0	458.2
			(0, 4)	1804	0	2	500.0	501.0
			(0, 5)	1805	1	11	90.9	461.9
			(0, 6)	1806	1	11	90.9	1022.7
			(0, 7)	1808	1	9	111.1	79.6
			(0, 10)	1810	0	2	500.0	715.8
			(0, 12)	1803	0	2	500.0	318.3
	500	(5, 4)	(0, 0)	1832	0	2	500.0	2117.2
			(0, 1)	1834	0	2	500.0	1969.2
			(0, 2)	1836	1	9	111.1	416.3
			(0, 3)	1838	1	12	83.3	360.3
			(0, 4)	1839	1	10	100.0	437.9
			(0, 5)	1840	0	2	500.0	2728.8

Continued on next page...

Table 4.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
			(0, 6)	1841	0	2	500.0	2201.5
			(0, 7)	1833	0	2	500.0	2221.0
			(0, 8)	1835	1	9	111.1	122.4
			(0, 9)	1837	1	10	100.0	81.0
	550	(18, 6)	(0, 3)	1849	1	9	111.1	93.4
			(0, 4)	1850	0	NaN		
			(0, 5)	1851	0	NaN		
			(0, 6)	1843	1	8	125.0	786.0
			(0, 7)	1845	0	NaN		
		(19, 7)	(0, 0)	1842	1	7	142.9	167.8
			(0, 1)	1844	1	8	125.0	179.7
			(0, 2)	1846	0	NaN		
			(0, 3)	1847	1	9	111.1	932.0
			(0, 4)	1848	0	NaN		
	600	(22, 2)	(0, 3)	1856	0	NaN		
			(0, 5)	1860	0	NaN		
		(23, 2)	(0, 0)	1858	0	NaN		
			(0, 1)	1861	1	10	100.0	256.3
			(0, 2)	1852	1	8	125.0	51.0
			(0, 3)	1853	1	9	111.1	126.0
			(0, 4)	1854	1	9	111.1	619.3
			(0, 5)	1855	1	11	90.9	231.8
			(0, 6)	1857	0	NaN		
			(0, 7)	1859	0	NaN		
5	70	(9, 5)	(0, 0)	1773	1	19	52.6	64.3
			(0, 1)	1774	1	22	45.5	42.6
			(0, 2)	1777	1	20	50.0	98.9
			(0, 3)	1779	1	24	41.7	42.1
			(0, 4)	1781	0	10	100.0	172.4
			(0, 5)	1772	1	19	52.6	89.9
			(0, 8)	1775	0	10	100.0	172.8
			(0, 11)	1776	0	9	111.1	113.3
			(0, 12)	1778	0	9	111.1	107.3
			(0, 13)	1780	0	10	100.0	155.2
	80	(23, 3)	(0, 0)	1785	1	17	58.8	142.1
			(0, 1)	1787	0	8	125.0	419.0
			(0, 2)	1788	1	16	62.5	91.3
			(0, 3)	1790	1	18	55.6	243.6
			(0, 4)	1791	0	8	125.0	590.6
			(0, 5)	1782	1	17	58.8	201.2

Continued on next page...

Table 4.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
			(0, 6)	1783	1	17	58.8	117.3
			(0, 7)	1786	0	8	125.0	438.5
			(0, 9)	1789	0	8	125.0	451.3
			(0, 12)	1784	0	8	125.0	451.5
	90	(29, 1)	(0, 0)	1797	0	6	166.7	1907.4
			(0, 1)	1798	0	5	200.0	1223.6
			(0, 2)	1800	1	16	62.5	913.7
			(0, 3)	1801	1	16	62.5	711.7
			(0, 4)	1792	0	6	166.7	1862.5
			(0, 5)	1794	0	5	200.0	1128.6
			(0, 6)	1796	0	6	166.7	1954.4
			(0, 11)	1795	1	13	76.9	609.7
			(0, 20)	1799	1	13	76.9	206.9
			(0, 23)	1793	1	15	66.7	547.2
	100	(28, 3)	(0, 0)	1867	1	10	100.0	336.6
			(0, 1)	1869	0	2	500.0	2279.1
			(0, 2)	1862	0	2	500.0	2765.1
			(0, 3)	1863	1	10	100.0	472.4
			(0, 4)	1864	0	2	500.0	2521.4
			(0, 5)	1865	1	13	76.9	456.4
			(0, 6)	1866	0	2	500.0	2447.3
			(0, 7)	1868	1	11	90.9	214.4
			(0, 8)	1870	0	2	500.0	1889.9
			(0, 10)	1871	1	12	83.3	179.1
	110	(14, 6)	(0, 0)	1874	0	1	1000.0	3896.0
			(0, 1)	1877	0	1	1000.0	4332.8
			(0, 2)	1878	0	1	1000.0	3794.7
			(0, 3)	1879	0	1	1000.0	4584.7
			(0, 4)	1880	0	1	1000.0	4070.9
			(0, 5)	1881	1	8	125.0	303.5
			(0, 7)	1875	1	8	125.0	206.3
			(0, 10)	1872	1	11	90.9	211.9
			(0, 11)	1873	1	9	111.1	414.1
			(0, 12)	1876	1	8	125.0	390.9
	120	(14, 7)	(0, 0)	1887	1	7	142.9	25.0
			(0, 1)	1889	1	8	125.0	517.3
			(0, 2)	1891	0	NaN		
			(0, 3)	1882	1	7	142.9	11.3
			(0, 4)	1883	0	NaN		
			(0, 5)	1884	0	NaN		

Continued on next page...

Table 4.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
		(27, 3)	(0, 6)	1885	0	NaN		
			(0, 7)	1886	0	NaN		
			(0, 8)	1888	1	7	142.9	18.4
			(0, 9)	1890	1	8	125.0	347.0
	130	(15, 4)	(0, 0)	1892	1	11	90.9	500.8
		(22, 4)	(0, 1)	1894	0	NaN		
			(0, 0)	1898	0	NaN		
			(0, 1)	1900	1	9	111.1	148.6
		(29, 2)	(0, 0)	1901	0	NaN		
			(0, 1)	1893	0	NaN		
			(0, 2)	1896	1	7	142.9	49.0
		(6, 6)	(0, 0)	1895	1	7	142.9	304.0
			(0, 1)	1897	0	NaN		
			(0, 2)	1899	1	7	142.9	12.6
7	45	(22, 3)	(0, 0)	1706	0	9	111.1	577.3
			(0, 1)	1708	0	9	111.1	576.1
			(0, 2)	1710	1	21	47.6	98.1
			(0, 3)	1711	0	9	111.1	583.5
			(0, 4)	1702	1	19	52.6	288.1
			(0, 5)	1704	0	9	111.1	545.7
			(0, 6)	1705	0	9	111.1	560.6
			(0, 7)	1707	1	23	43.5	187.4
			(0, 8)	1709	1	17	58.8	281.9
			(0, 10)	1703	1	18	55.6	216.2
	50	(16, 4)	(0, 0)	1716	0	5	200.0	1657.1
			(0, 1)	1719	1	16	62.5	683.3
			(0, 2)	1721	1	15	66.7	711.5
			(0, 3)	1712	1	17	58.8	359.1
			(0, 4)	1713	0	7	142.9	1554.3
			(0, 5)	1714	1	15	66.7	309.0
			(0, 6)	1715	1	16	62.5	188.6
			(0, 7)	1717	0	5	200.0	1754.7
			(0, 10)	1718	0	7	142.9	1529.0
			(0, 11)	1720	0	5	200.0	1730.6
	55	(15, 3)	(0, 0)	1727	1	12	83.3	878.9
			(0, 1)	1729	1	11	90.9	545.4
			(0, 2)	1730	0	3	333.3	2725.5
			(0, 3)	1722	0	3	333.3	2800.6
			(0, 4)	1723	0	3	333.3	2768.7
		(0, 5)	1724	1	12	83.3	288.0	

Continued on next page...

Table 4.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
			(0, 6)	1725	0	3	333.3	2616.5
			(0, 7)	1728	0	3	333.3	2701.9
			(0, 9)	1731	1	10	100.0	349.9
			(0, 10)	1726	1	14	71.4	884.3
	60	(11, 9)	(0, 0)	1740	1	9	111.1	633.4
			(0, 1)	1732	0	1	1000.0	987.8
			(0, 2)	1734	0	1	1000.0	981.7
			(0, 3)	1736	1	11	90.9	446.7
			(0, 4)	1737	0	1	1000.0	984.0
			(0, 5)	1738	0	1	1000.0	982.2
			(0, 6)	1739	1	10	100.0	593.5
			(0, 7)	1741	1	11	90.9	734.8
			(0, 8)	1733	0	1	1000.0	1002.2
			(0, 9)	1735	1	12	83.3	502.1
	65	(13, 7)	(0, 0)	1744	1	13	76.9	629.6
			(0, 1)	1746	0	1	1000.0	3512.1
			(0, 2)	1747	0	1	1000.0	3430.9
			(0, 3)	1749	0	1	1000.0	3504.7
			(0, 4)	1751	0	1	1000.0	3484.0
			(0, 5)	1742	1	9	111.1	305.6
			(0, 6)	1743	1	10	100.0	1059.6
			(0, 7)	1745	0	1	1000.0	3365.7
			(0, 11)	1748	1	11	90.9	84.0
			(0, 12)	1750	1	10	100.0	661.9
	70	(21, 6)	(0, 0)	1752	1	8	125.0	192.1
			(0, 1)	1755	0	NaN		
			(0, 2)	1756	0	NaN		
			(0, 3)	1758	1	7	142.9	228.7
			(0, 4)	1759	0	NaN		
			(0, 5)	1760	0	NaN		
			(0, 6)	1761	1	8	125.0	383.5
			(0, 7)	1753	0	NaN		
			(0, 9)	1757	1	8	125.0	612.7
			(0, 12)	1754	1	8	125.0	63.1
	75	(23, 4)	(0, 3)	1770	1	6	166.7	894.8
		(25, 4)	(0, 0)	1769	0	NaN		
			(0, 1)	1762	0	NaN		
			(0, 2)	1763	1	8	125.0	1619.0
			(0, 5)	1767	1	5	200.0	169.3
		(26, 8)	(0, 0)	1764	0	NaN		

Continued on next page...

Table 4.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
			(0, 1)	1765	1	8	125.0	323.4
			(0, 2)	1766	0	NaN		
		(9, 6)	(0, 0)	1768	0	NaN		
			(0, 1)	1771	1	7	142.9	130.4

Table 5: The large size benchmarks of the r-competition in detail. Using the first four columns, the instances can be created from `OKgenerator`; consider for example the very first row: The (satisfiable) instance with the r-competition name “bench1663” has been created by the call “`OKgenerator s0=32 s1=2 n=2000 l=3 cp=8400 nr0=0 nr1=0 -D -o.cnf`” (looking up the number of clauses in Table 3); s_0 is the computer-identification, s_1 the experiment-identification, while (nr_0, nr_1) is the counter (and “-D” switches to Dimacs-output (default is XML), while “-o” generates the output). In the column “problem-p” the number of solvers which have solved the instance is reported together with the resulting problem purse, while the last column reports the average time used. An entry “1*” in the sat-column indicates that we have been able to determine the satisfiability of the instance only after the competition (so these instances seem to be quite hard), while a question mark indicates that the instance has not been solved yet at all (despite running several solvers for weeks; two such (very hard) instances remain, one for $k = 3, n = 12000$, and the other for $k = 7, n = 220$).

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
3	2000	(32, 2)	(0, 0)	1663	1	5	200.0	416.6
			(0, 1)	1665	1	4	250.0	1017.3
			(0, 2)	1667	1	2	500.0	665.9
			(0, 3)	1669	1	5	200.0	118.6
			(0, 4)	1670	1	4	250.0	1110.1
			(0, 5)	1671	1	5	200.0	176.1
			(0, 6)	1662	1	2	500.0	639.7
			(0, 7)	1664	1	4	250.0	361.4
			(0, 8)	1666	1	6	166.7	72.4
			(0, 9)	1668	1	6	166.7	185.2
	4000	(32, 40)	(0, 0)	1676	1	3	333.3	2043.6
			(0, 1)	1678	1	4	250.0	90.6
			(0, 2)	1680	1	5	200.0	296.6
			(0, 3)	1672	1	5	200.0	242.9
			(0, 4)	1673	1	5	200.0	477.9
			(0, 5)	1674	1	5	200.0	327.7
			(0, 6)	1675	1	4	250.0	117.9
			(0, 7)	1677	1	4	250.0	1435.8
			(0, 8)	1679	1	5	200.0	603.6
			(0, 9)	1681	1	5	200.0	629.7

Continued on next page...

Table 5.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
6000	(32, 41)	(0, 0)	1682	1	5	200.0	988.4	
		(0, 1)	1684	1	4	250.0	142.1	
		(0, 2)	1686	1	4	250.0	155.3	
		(0, 3)	1688	1	4	250.0	1348.3	
		(0, 4)	1689	1	4	250.0	435.3	
		(0, 5)	1690	1	2	500.0	1533.2	
		(0, 6)	1691	1	2	500.0	1776.6	
		(0, 7)	1683	1	4	250.0	806.0	
		(0, 8)	1685	1	4	250.0	1175.5	
		(0, 9)	1687	1	4	250.0	465.6	
8000	(32, 14)	(0, 0)	1696	1*	NaN			
		(0, 1)	1698	1	4	250.0	411.9	
		(0, 2)	1700	1	3	333.3	347.7	
		(0, 3)	1692	1*	NaN			
		(0, 4)	1693	1	4	250.0	224.1	
		(0, 5)	1694	1	2	500.0	1565.7	
		(0, 6)	1695	1	2	500.0	1136.3	
		(0, 7)	1697	1	4	250.0	1763.9	
		(0, 8)	1699	1	3	333.3	2174.6	
		(0, 9)	1701	1	4	250.0	982.1	
10000	(32, 17)	(0, 0)	1644	1*	NaN			
		(0, 1)	1646	1	3	333.3	1934.8	
		(0, 2)	1648	1	3	333.3	2032.8	
		(0, 3)	1650	1	3	333.3	1896.7	
		(0, 4)	1651	1	1	1000.0	1337.0	
		(0, 5)	1642	1	1	1000.0	2097.4	
		(0, 6)	1643	1*	NaN			
		(0, 7)	1645	1	1	1000.0	5578.6	
		(0, 8)	1647	1*	NaN			
		(0, 9)	1649	1	1	1000.0	4985.6	
12000	(32, 18)	(0, 0)	1656	1	2	500.0	1424.8	
		(0, 1)	1658	1	1	1000.0	1850.1	
		(0, 2)	1660	1*	NaN			
		(0, 3)	1652	?	NaN			
		(0, 4)	1653	1	3	333.3	1083.3	
		(0, 5)	1654	1	1	1000.0	4826.0	
		(0, 6)	1655	1	2	500.0	872.6	
		(0, 7)	1657	1	2	500.0	2540.9	
		(0, 8)	1659	1	1	1000.0	1300.4	
		(0, 9)	1661	1*	NaN			

Continued on next page...

Table 5.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p	μ time	
5	500	(32, 29)	(0, 0)	1540	1	2	500.0	972.6
			(0, 1)	1532	1	2	500.0	261.7
			(0, 2)	1534	1	2	500.0	1009.5
			(0, 3)	1536	1	1	1000.0	384.4
			(0, 4)	1537	1	4	250.0	441.3
			(0, 5)	1538	1	2	500.0	1866.4
			(0, 6)	1539	1	2	500.0	374.8
			(0, 7)	1541	1	5	200.0	315.4
			(0, 8)	1533	1	3	333.3	1337.9
			(0, 9)	1535	1	2	500.0	713.9
	600	(32, 42)	(0, 0)	1542	1	1	1000.0	83.8
			(0, 1)	1544	1	1	1000.0	71.2
			(0, 2)	1546	1	1	1000.0	459.9
			(0, 3)	1548	1	1	1000.0	936.2
			(0, 4)	1549	1	1	1000.0	433.4
			(0, 5)	1550	1	1	1000.0	119.2
			(0, 6)	1551	1	1	1000.0	1233.6
			(0, 7)	1543	1	1	1000.0	192.8
			(0, 8)	1545	1	2	500.0	1580.9
			(0, 9)	1547	1	2	500.0	2157.0
	700	(32, 43)	(0, 0)	1560	1	1	1000.0	1368.9
			(0, 1)	1552	1	1	1000.0	209.6
			(0, 2)	1554	1	1	1000.0	1717.7
			(0, 3)	1556	1	1	1000.0	548.4
			(0, 4)	1557	1	2	500.0	1282.8
			(0, 5)	1558	1	1	1000.0	169.5
			(0, 6)	1559	1	1	1000.0	4698.5
			(0, 7)	1561	1	1	1000.0	167.7
			(0, 8)	1553	1	1	1000.0	2585.8
			(0, 9)	1555	1	2	500.0	970.3
	800	(32, 36)	(0, 0)	1568	1	1	1000.0	10.0
			(0, 1)	1570	1	1	1000.0	949.5
			(0, 2)	1562	1	NaN		
			(0, 3)	1564	1	1	1000.0	888.9
			(0, 4)	1565	1*	NaN		
			(0, 5)	1566	1*	NaN		
			(0, 6)	1567	1	1	1000.0	354.2
			(0, 7)	1569	1	1	1000.0	1081.5
			(0, 8)	1571	1	2	500.0	3228.6
			(0, 9)	1563	1	1	1000.0	3387.5

Continued on next page...

Table 5.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
	900	(32, 44)	(0, 0)	1580	1*	NaN		
			(0, 1)	1572	1	1	1000.0	4010.7
			(0, 2)	1574	1	NaN		
			(0, 3)	1576	1*	NaN		
			(0, 4)	1577	1	NaN		
			(0, 5)	1578	1*	NaN		
			(0, 6)	1579	1	1	1000.0	122.8
			(0, 7)	1581	1	NaN		
			(0, 8)	1573	1	NaN		
			(0, 9)	1575	1	1	1000.0	2932.2
	1000	(32, 28)	(0, 0)	1528	1*	NaN		
			(0, 1)	1530	1	NaN		
			(0, 2)	1522	1*	NaN		
			(0, 3)	1524	1*	NaN		
			(0, 4)	1525	1*	NaN		
			(0, 5)	1526	1	1	1000.0	3503.5
			(0, 6)	1527	1	NaN		
			(0, 7)	1529	1	1	1000.0	5619.1
			(0, 8)	1531	1*	NaN		
			(0, 9)	1523	1	1	1000.0	1153.2
7	120	(32, 24)	(0, 0)	1588	1	4	250.0	541.6
			(0, 1)	1590	1	4	250.0	164.8
			(0, 2)	1582	1	4	250.0	187.0
			(0, 3)	1584	1	3	333.3	194.0
			(0, 4)	1585	1	3	333.3	617.8
			(0, 5)	1586	1	3	333.3	476.6
			(0, 6)	1587	1	5	200.0	1123.3
			(0, 7)	1589	1	4	250.0	257.1
			(0, 8)	1591	1	5	200.0	809.5
			(0, 9)	1583	1	4	250.0	67.1
	140	(32, 30)	(0, 0)	1594	1	1	1000.0	4463.0
			(0, 1)	1596	1	2	500.0	1078.0
			(0, 2)	1598	1	2	500.0	584.9
			(0, 3)	1600	1	2	500.0	3797.7
			(0, 4)	1601	1	1	1000.0	346.7
			(0, 5)	1592	1	5	200.0	1106.0
			(0, 6)	1593	1	2	500.0	2977.0
			(0, 7)	1595	1	2	500.0	1654.4
			(0, 8)	1597	1	2	500.0	1672.0
			(0, 9)	1599	1	3	333.3	845.5

Continued on next page...

Table 5.– continued from previous page

k	n	(s_0, s_1)	(nr_0, nr_1)	code	sat	problem-p		μ time
	160	(32, 31)	(0, 0)	1605	1	1	1000.0	5016.6
			(0, 1)	1606	1	2	500.0	474.2
			(0, 2)	1609	1	1	1000.0	5361.6
			(0, 3)	1611	1	NaN		
			(0, 4)	1602	1	1	1000.0	3251.0
			(0, 5)	1603	1	2	500.0	900.3
			(0, 6)	1604	1	1	1000.0	66.4
			(0, 7)	1606	1	2	500.0	474.2
			(0, 8)	1608	1	2	500.0	3118.9
			(0, 9)	1610	1	2	500.0	3619.2
	180	(32, 37)	(0, 0)	1620	1	NaN		
			(0, 1)	1612	1*	NaN		
			(0, 2)	1614	1	NaN		
			(0, 3)	1616	1	NaN		
			(0, 4)	1617	1	NaN		
			(0, 5)	1618	1	NaN		
			(0, 6)	1619	1	NaN		
			(0, 7)	1621	1	NaN		
			(0, 8)	1613	1	1	1000.0	666.8
			(0, 9)	1615	1	NaN		
	200	(32, 19)	(0, 0)	1626	1*	NaN		
			(0, 1)	1628	1	1	1000.0	4393.4
			(0, 2)	1630	1	NaN		
			(0, 3)	1622	1	NaN		
			(0, 4)	1623	1*	NaN		
			(0, 5)	1624	1	NaN		
			(0, 6)	1625	1*	NaN		
			(0, 7)	1627	1	NaN		
			(0, 8)	1629	1*	NaN		
			(0, 9)	1631	1	NaN		
	220	(32, 20)	(0, 0)	1640	?	NaN		
			(0, 1)	1632	1*	NaN		
			(0, 2)	1634	1*	NaN		
			(0, 3)	1636	1*	NaN		
			(0, 4)	1637	1*	NaN		
			(0, 5)	1638	1*	NaN		
			(0, 6)	1639	1	NaN		
			(0, 7)	1641	1*	NaN		
			(0, 8)	1633	1	NaN		
			(0, 9)	1635	1	NaN		

Appendix C. Scores of the solvers of the SAT 2005 competition

Table 6. Scores of all solvers on all instances (“large size” and “medium size”), showing the total score and the partial scores according to (13); “# p” is the number of benchmarks solved (out of $105 + 285 = 39 \cdot 10 = 390$ benchmarks in total), and “# s” is the number of series solved (out of $3 \cdot 7 + 3 \cdot 6 = 39$ series in total). The framed solvers solved at least one unsatisfiable random instance and thus were eligible to win a price, while the five underlined solvers solved unsatisfiable instances in other sub-competitions but no unsatisfiable random instance (the remaining solvers are local search solvers only capable of solving satisfiable instances).

solver	score	problem-p	speed-p	series-p	# p	# s
ranov	169903.3	66822.0	79054.1	24027.2	209	36
g2wsat	103286.1	35488.7	53270.2	14527.2	178	32
kcnfs-2004	95075.4	37211.4	53236.9	4627.2	167	21
vw	76002.5	33788.7	26936.6	15277.2	170	33
rpaws10	69619.0	20905.4	38186.5	10527.2	151	30
rrsaps	40228.4	12872.0	20379.2	6977.2	116	25
march_dl	27141.6	13120.2	11468.0	2553.3	99	14
Dew_Satz_1a	22940.3	14448.2	5281.6	3210.5	118	17
adaptnovelty	21748.0	12388.7	1932.2	7427.2	119	26
wllsatv1	16145.0	11128.6	2018.0	2998.4	104	16
saps	15603.9	9238.7	1738.1	4627.2	104	21
Dew_Satz_1c	13914.0	7419.4	3995.6	2499.0	85	14
Dew_Satz_1b	13281.2	7276.6	3505.7	2499.0	84	14
SatELiteGTI	10074.1	6806.8	1083.0	2184.3	79	13
minisat_static	10058.2	6714.4	1123.8	2220.0	78	13
sat4j.jar	5289.4	3442.2	522.2	1324.9	50	9
csat	2325.6	1633.6	102.1	590.0	27	5
<u>hsat.5</u>	2162.2	891.9	363.6	906.7	19	7
<u>zchaff</u>	2063.8	738.4	585.4	740.0	16	6
<u>hsat.1</u>	1888.8	783.1	365.7	740.0	17	6
<u>vallst.sh</u>	1454.3	786.1	58.7	609.6	17	5
<u>zchaff_rand</u>	1307.9	533.5	17.8	756.7	12	6
Jerusat1.31.B	1245.0	527.6	247.4	470.0	11	4

Table 7. Scores of all solvers on satisfiable instances (“large size” and “medium size”) which solved at least one instance, showing the total score and the partial scores according to (13); “# p” is the number of benchmarks solved (out of $3 \cdot 7 \cdot 5 + 3 \cdot 6 \cdot 10 = 285$ benchmarks in total), and “# s” is the number of series solved (out of $3 \cdot 7 + 3 \cdot 6 = 39$ series in total). Altogether 231 benchmarks have been solved and 38 series, and thus the sum over problem and speed purse is $231 \cdot 1000 = 231000$, while the sum over the series purse is $38 \cdot 3000 = 114000$ (up to rounding effects).

solver	score	problem-p	speed-p	series-p	# p	# s
ranov	169903.3	66822.0	79054.1	24027.2	209	36
g2wsat	103286.1	35488.7	53270.2	14527.2	178	32
vw	76002.5	33788.7	26936.6	15277.2	170	33
rpaws10	69619.0	20905.4	38186.5	10527.2	151	30
rrsaps	40228.4	12872.0	20379.2	6977.2	116	25
adaptnovelty	21748.0	12388.7	1932.2	7427.2	119	26
saps	15603.9	9238.7	1738.1	4627.2	104	21
kcnfs-2004	14604.9	7536.3	2441.4	4627.2	92	21
Dew_Satz_1a	8943.1	4773.2	959.4	3210.5	68	17
march_dl	7444.6	3897.6	993.7	2553.3	56	14
wllsatv1	7202.6	3953.6	250.6	2998.4	59	16
Dew_Satz_1c	6590.6	3011.1	1080.5	2499.0	50	14
Dew_Satz_1b	6187.3	3011.1	677.3	2499.0	50	14
SatELiteGTI	5198.0	2698.5	315.3	2184.3	46	13
minisat_static	5147.9	2606.1	321.8	2220.0	45	13
sat4j.jar	3400.7	1862.4	213.4	1324.9	35	9
hsat.5	2162.2	891.9	363.6	906.7	19	7
zchaff	2063.8	738.4	585.4	740.0	16	6
hsat.1	1888.8	783.1	365.7	740.0	17	6
csat	1617.3	942.7	84.7	590.0	20	5
vallst.sh	1454.3	786.1	58.7	609.6	17	5
zchaff_rand	1307.9	533.5	17.8	756.7	12	6
Jerusat1.31_B	1152.8	436.7	246.1	470.0	10	4
hsattrr	881.7	196.4	340.3	345.0	5	3
Jerusat1.31_A	673.8	276.4	57.8	339.6	7	3
HaifaSatw	547.4	194.6	18.2	334.6	5	3
lsatv1.1	432.1	137.1	71.5	223.4	3	2
HaifaSat	357.3	116.1	17.7	223.4	3	2
compsat	349.6	112.9	22.1	214.6	3	2

Table 8. Scores of all solvers on unsatisfiable instances (only “medium size”) which solved at least one instance, showing the total score and the partial scores according to (13); “# p” is the number of benchmarks solved (out of $3 \cdot 7 \cdot 5 = 105$ benchmarks in total), and “# s” is the number of series solved (out of $3 \cdot 7 = 21$ series in total). Altogether 75 benchmarks have been solved and 15 series (all by *kcnfs-2004*), and thus the sum over problem and speed purse is $75 \cdot 1000 = 75000$, while the sum over the series purse is $15 \cdot 3000 = 45000$ (up to rounding effects).

solver	score	problem-p	speed-p	series-p	# p	# s
<i>kcnfs-2004</i>	97930.2	29675.0	50795.5	17459.6	75	15
<i>march_dl</i>	25228.1	9222.7	10474.4	5531.1	43	9
<i>Dew_Satz_1a</i>	19456.8	9675.0	4322.2	5459.6	50	10
<i>wllsatv1</i>	12902.0	7175.0	1767.4	3959.6	45	9
<i>Dew_Satz_1c</i>	10283.0	4408.4	2915.0	2959.6	35	8
<i>Dew_Satz_1b</i>	10053.5	4265.5	2828.4	2959.6	34	8
<i>minisat_static</i>	7370.0	4108.4	802.0	2459.6	33	7
<i>SatELiteGTI</i>	7335.7	4108.4	767.7	2459.6	33	7
<i>sat4j.jar</i>	2794.7	1579.8	308.8	906.1	15	3
<i>csat</i>	1281.0	690.9	17.4	572.7	7	2
<i>Jerusat1.31_B</i>	364.9	90.9	1.3	272.7	1	1

Appendix D. A more detailed analysis

D.1 Unsatisfiable instances

Table 9. Lexicographical order of solvers for clause-length $k = 3$ on unsatisfiable instances (all solvers are shown which solved at least one instance); the entry “5 : 1997” at the left top for example means that 5 instances of the series with 500 variables have been solved, and this using an average running time of 1997s. All series contain 5 (unsatisfiable) instances, and no solver solved any (unsatisfiable) instance with $n \in \{550, 600\}$.

solver	$n = 500$	$n = 450$	$n = 400$	$n = 360$	$n = 300$
<i>kcnfs-2004</i>	5 : 1997	5 : 465	5 : 71	5 : 22	5 : 3.0
<i>march_dl</i>	5 : 2498	5 : 554	5 : 84	5 : 26	5 : 3.4
<i>Dew_Satz_1a</i>	0	0	5 : 1052	5 : 228	5 : 19
<i>wllsatv1</i>	0	0	5 : 3657	5 : 1060	5 : 78
<i>Dew_Satz_1c</i>	0	0	3 : 879	5 : 241	5 : 19
<i>minisat_static</i>	0	0	3 : 3176	5 : 1147	5 : 51
<i>SatELiteGTI</i>	0	0	3 : 4298	5 : 1184	5 : 61
<i>Dew_Satz_1b</i>	0	0	2 : 748	5 : 259	5 : 20
<i>sat4j.jar</i>	0	0	0	0	5 : 302
<i>csat</i>	0	0	0	0	4 : 699
<i>Jerusat1.31_B</i>	0	0	0	0	1 : 952

Table 10. Lexicographical order of solvers for clause-length $k = 5$ on unsatisfiable instances (all solvers are shown which solved at least one instance); the entry “5 : 4136” at the left top for example means that 5 instances of the series with 110 variables have been solved, and this using an average running time of 4136s. All series contain 5 (unsatisfiable) instances, and no solver solved any (unsatisfiable) instance with $n \in \{120, 130\}$.

solver	110	100	90	80	70
kcnfs-2004	5 : 4136	5 : 838	5 : 171	5 : 33	5 : 6.5
Dew_Satz_1a	0	5 : 3923	5 : 805	5 : 151	5 : 27
Dew_Satz_1c	0	0	5 : 796	5 : 150	5 : 27
Dew_Satz_1b	0	0	5 : 796	5 : 151	5 : 27
wllsatv1	0	0	5 : 3181	5 : 533	5 : 86
march_dl	0	0	3 : 5787	5 : 992	5 : 164
minisat_static	0	0	0	5 : 870	5 : 96
SatELiteGTI	0	0	0	5 : 882	5 : 96
sat4j.jar	0	0	0	0	5 : 277
csat	0	0	0	0	3 : 986

Table 11. Lexicographical order of solvers for clause-length $k = 7$ on unsatisfiable instances (all solvers are shown which solved at least one instance); the entry “5 : 3459” at the left top for example means that 5 instances of the series with 65 variables have been solved, and this using an average running time of 3459s. All series contain 5 (unsatisfiable) instances, and no solver solved any (unsatisfiable) instance with $n \in \{70, 75\}$.

solver	65	60	55	50	45
kcnfs-2004	5 : 3459	5 : 988	5 : 282	5 : 82	5 : 23
wllsatv1	0	0	5 : 3650	5 : 928	5 : 232
Dew_Satz_1a	0	0	5 : 4236	5 : 1210	5 : 351
SatELiteGTI	0	0	0	5 : 3152	5 : 522
minisat_static	0	0	0	5 : 3155	5 : 520
Dew_Satz_1c	0	0	0	2 : 1161	5 : 352
Dew_Satz_1b	0	0	0	2 : 1169	5 : 351
sat4j.jar	0	0	0	0	5 : 381
march_dl	0	0	0	0	5 : 2387

D.2 Satisfiable instances

Table 12. Lexicographical order of solvers for clause-length $k = 3$ on satisfiable instances (all solvers which solved at least one instance)

solver	12000	10000	8000	6000	4000	2000	600	550	500	450	400	360	300
vw	7 : 1769	5 : 1642	8 : 847	10 : 1243	9 : 61	10 : 700	5 : 1.6	5 : 2.3	5 : 0.6	5 : 2.3	5 : 0.2	5 : 0.3	5 : 0.0
g2wsat	4 : 2050	5 : 1506	8 : 1367	10 : 198	10 : 134	10 : 139	5 : 0.2	5 : 0.6	5 : 0.1	5 : 0.4	5 : 0.0	5 : 0.3	5 : 0.0
rpaws10	1 : 322	0	6 : 60	8 : 181	10 : 88	8 : 85	5 : 0.2	5 : 0.4	5 : 0.1	5 : 0.8	5 : 0.1	5 : 0.1	5 : 0.1
ranov	0	3 : 5283	4 : 2109	8 : 1139	10 : 1084	8 : 458	5 : 0.5	5 : 1.1	5 : 0.2	5 : 0.3	5 : 0.1	5 : 0.2	5 : 0.1
adaptnovelty	0	0	0	1 : 4713	6 : 1998	5 : 725	5 : 63	5 : 86	5 : 12	5 : 87	5 : 2.7	5 : 12	5 : 2.1
rrsaps	0	0	0	0	0	2 : 657	5 : 9.7	5 : 3.4	5 : 0.8	5 : 3.9	5 : 0.1	5 : 0.2	5 : 0.1
saps	0	0	0	0	0	0	5 : 376	5 : 549	5 : 54	5 : 217	5 : 5.7	5 : 15	5 : 2.2
kcnfs-2004	0	0	0	0	0	0	5 : 726	3 : 3972	5 : 575	5 : 108	5 : 6.8	5 : 3.3	5 : 0.6
march_dl	0	0	0	0	0	0	4 : 666	3 : 999	5 : 707	5 : 231	5 : 15	5 : 2.5	5 : 0.5
Dew_Satz_1a	0	0	0	0	0	0	1 : 1342	0	1 : 332	3 : 3768	5 : 139	5 : 39	5 : 2.6
wllsatv1	0	0	0	0	0	0	1 : 1413	0	1 : 4059	1 : 5745	5 : 688	5 : 237	5 : 31
minisat_static	0	0	0	0	0	0	1 : 922	0	0	1 : 2434	5 : 175	5 : 71	5 : 9.7
SatELiteGTI	0	0	0	0	0	0	0	0	1 : 1658	2 : 2999	5 : 234	5 : 190	5 : 5.8
Dew_Satz_1c	0	0	0	0	0	0	0	0	1 : 773	0	5 : 269	5 : 37	5 : 7.9
Dew_Satz_1b	0	0	0	0	0	0	0	0	1 : 791	0	5 : 282	5 : 40	5 : 8.4
sat4j.jar	0	0	0	0	0	0	0	0	0	0	4 : 580	4 : 267	5 : 31
csat	0	0	0	0	0	0	0	0	0	0	4 : 325	3 : 480	5 : 130
zchaff	0	0	0	0	0	0	0	0	0	0	3 : 213	3 : 12	4 : 82
hsat.5	0	0	0	0	0	0	0	0	0	0	2 : 529	5 : 286	5 : 29
hsat.1	0	0	0	0	0	0	0	0	0	0	2 : 454	4 : 163	5 : 43
Jerusat1.31_B	0	0	0	0	0	0	0	0	0	0	2 : 224	1 : 0.2	5 : 330
zchaff_rand	0	0	0	0	0	0	0	0	0	0	1 : 393	3 : 352	2 : 87
hsatrr	0	0	0	0	0	0	0	0	0	0	1 : 0.1	1 : 0.0	3 : 0.0
vallst.sh	0	0	0	0	0	0	0	0	0	0	0	3 : 158	4 : 78
Jerusat1.31_A	0	0	0	0	0	0	0	0	0	0	0	2 : 308	4 : 92
HaifaSatw	0	0	0	0	0	0	0	0	0	0	0	2 : 758	2 : 15
compsat	0	0	0	0	0	0	0	0	0	0	0	1 : 320	2 : 451
HaifaSat	0	0	0	0	0	0	0	0	0	0	0	0	2 : 15
lsatv1.1	0	0	0	0	0	0	0	0	0	0	0	0	1 : 3.2

Table 13. Lexicographical order of solvers for clause-length $k = 5$ on satisfiable instances (all solvers which solved at least one instance)

solver	1000	900	800	700	600	500	130	120	110	100	90	80	70
ranov	3 : 3425	3 : 2355	7 : 1153	10 : 1230	10 : 404	10 : 119	5 : 2.5	5 : 0.7	5 : 0.3	5 : 0.2	5 : 0.3	5 : 0.5	5 : 0.1
vw	0	0	1 : 5060	2 : 1836	2 : 3483	7 : 653	5 : 3.8	5 : 2.6	5 : 7.5	5 : 1.2	5 : 0.6	5 : 0.9	5 : 0.9
g2wsat	0	0	0	0	0	4 : 1673	5 : 1.4	5 : 0.7	5 : 1.0	5 : 0.1	5 : 0.1	5 : 0.2	5 : 0.1
adaptnovelty	0	0	0	0	0	3 : 1793	5 : 303	5 : 60	5 : 81	5 : 17	5 : 12	5 : 20	5 : 11
rpaws10	0	0	0	0	0	1 : 302	5 : 2.3	5 : 0.6	5 : 0.9	5 : 0.4	5 : 0.3	5 : 0.3	5 : 0.1
rrsaps	0	0	0	0	0	0	5 : 2.2	5 : 1.5	5 : 0.8	5 : 0.3	5 : 0.3	5 : 0.3	5 : 0.2
saps	0	0	0	0	0	0	5 : 337	5 : 77	5 : 76	5 : 19	5 : 13	5 : 13	5 : 6.9
kcnfs-2004	0	0	0	0	0	0	2 : 753	2 : 3289	5 : 1384	5 : 218	5 : 14	5 : 13	5 : 3.3
Dew_Satz_1a	0	0	0	0	0	0	2 : 1141	0	2 : 1796	5 : 754	5 : 253	5 : 62	5 : 7.3
Dew_Satz_1b	0	0	0	0	0	0	1 : 1169	0	1 : 958	3 : 241	5 : 221	5 : 64	5 : 7.1
Dew_Satz_1c	0	0	0	0	0	0	1 : 1188	0	1 : 954	3 : 238	5 : 220	5 : 64	5 : 7.0
wllsatv1	0	0	0	0	0	0	0	0	0	4 : 2115	5 : 1488	5 : 317	5 : 15
march_dl	0	0	0	0	0	0	0	0	0	1 : 3585	5 : 2151	5 : 348	5 : 5.1
minisat_static	0	0	0	0	0	0	0	0	0	0	3 : 3557	5 : 609	5 : 3.4
SatELiteGTI	0	0	0	0	0	0	0	0	0	0	3 : 3573	5 : 613	5 : 3.4
sat4j.jar	0	0	0	0	0	0	0	0	0	0	2 : 786	4 : 374	5 : 4.8
hsat.5	0	0	0	0	0	0	0	0	0	0	0	2 : 372	2 : 7.2
hsat.1	0	0	0	0	0	0	0	0	0	0	0	2 : 388	2 : 7.3
vallst.sh	0	0	0	0	0	0	0	0	0	0	0	1 : 27	5 : 15.6
zchaff	0	0	0	0	0	0	0	0	0	0	0	1 : 6.5	3 : 8.4
csat	0	0	0	0	0	0	0	0	0	0	0	0	5 : 5.74
zchaff_rand	0	0	0	0	0	0	0	0	0	0	0	0	3 : 3.75
lsatv1.1	0	0	0	0	0	0	0	0	0	0	0	0	2 : 4.8
HaifaSatw	0	0	0	0	0	0	0	0	0	0	0	0	1 : 1.96
HaifaSat	0	0	0	0	0	0	0	0	0	0	0	0	1 : 1.99

Table 14. Lexicographical order of solvers for clause-length $k = 7$ on satisfiable instances (all solvers which solved at least one instance)

solver	200	180	160	140	120	75	70	65	60	55	50	45
ranov	1 : 4393	0	8 : 2228	9 : 1179	10 : 127	5 : 21	5 : 11	5 : 5.8	5 : 5.8	5 : 6.0	5 : 4.1	5 : 3.0
g2wsat	0	1 : 667	2 : 5478	9 : 2253	10 : 375	5 : 9.8	5 : 5.8	5 : 2.0	5 : 4.3	5 : 3.9	5 : 1.6	5 : 1.1
rrsaps	0	0	2 : 210	1 : 384	6 : 123	5 : 19	5 : 8.1	5 : 7.3	5 : 5.1	5 : 6.2	5 : 3.9	5 : 2.7
rpaws10	0	0	1 : 799	2 : 221	9 : 373	5 : 25	5 : 8.0	5 : 6.8	5 : 5.3	5 : 5.7	5 : 3.3	5 : 2.7
vw	0	0	0	1 : 4688	3 : 1414	5 : 515	5 : 64	5 : 12	5 : 92	5 : 97	5 : 16	5 : 4.9
adaptnovelty	0	0	0	0	1 : 5048	3 : 1541	5 : 586	5 : 249	5 : 534	5 : 774	5 : 169	5 : 52
saps	0	0	0	0	0	4 : 2138	5 : 451	5 : 138	5 : 243	5 : 241	5 : 87	5 : 23
kcnfs-2004	0	0	0	0	0	1 : 5322	4 : 1485	5 : 433	5 : 329	5 : 150	5 : 43	5 : 12
wllsatv1	0	0	0	0	0	1 : 1223	0	2 : 3996	4 : 2248	5 : 1166	5 : 437	5 : 71
Dew_Satz_1a	0	0	0	0	0	0	0	5 : 2020	4 : 2993	5 : 1640	5 : 637	5 : 173
Dew_Satz_1c	0	0	0	0	0	0	0	2 : 437	0	2 : 808	5 : 643	5 : 172
Dew_Satz_1b	0	0	0	0	0	0	0	2 : 441	0	2 : 793	5 : 639	5 : 173
minisat_static	0	0	0	0	0	0	0	1 : 2473	2 : 864	2 : 2373	5 : 1273	5 : 187
SatELiteGTI	0	0	0	0	0	0	0	1 : 2479	2 : 881	2 : 2382	5 : 1253	5 : 188
sat4j.jar	0	0	0	0	0	0	0	0	1 : 113	0	5 : 603	5 : 179
march_dl	0	0	0	0	0	0	0	0	0	1 : 2722	2 : 2397	5 : 930
zchaff_rand	0	0	0	0	0	0	0	0	0	0	1 : 334	2 : 565
hsat.5	0	0	0	0	0	0	0	0	0	0	1 : 1178	2 : 200
vallst.sh	0	0	0	0	0	0	0	0	0	0	0	4 : 612
csat	0	0	0	0	0	0	0	0	0	0	0	3 : 726
hsat.1	0	0	0	0	0	0	0	0	0	0	0	2 : 244
zchaff	0	0	0	0	0	0	0	0	0	0	0	2 : 634
Jerusat1.31.B	0	0	0	0	0	0	0	0	0	0	0	2 : 759
Jerusat1.31.A	0	0	0	0	0	0	0	0	0	0	0	1 : 224