Predicting Satisfiability at the Phase Transition

Lin Xu, Holger H. Hoos, Kevin Leyton-Brown
University of British Columbia,
201-2366 Main Mall, Vancouver, BC, Canada
{xulin730, hoos, kevinlb}@cs.ubc.ca

Abstract

Uniform random 3-SAT at the solubility phase transition is one of the most widely studied and empirically hardest distributions of SAT instances. For 20 years, this distribution has been used extensively for evaluating and comparing algorithms. In this work, we demonstrate that simple rules can predict the solubility of these instances with surprisingly high accuracy. Specifically, we show how classification accuracies of about 70% can be obtained based on cheaply (polynomial-time) computable features on a wide range of instance sizes. We argue in two ways that classification accuracy does not decrease with instance size: first, we show that our models' predictive accuracy remains roughly constant across a wide range of problem sizes; second, we show that a classifier trained on small instances is sufficient to achieve very accurate predictions across the entire range of instance sizes currently solvable by complete methods. Finally, we demonstrate that a simple decision tree based on only two features, and again trained only on the smallest instances, achieves predictive accuracies close to those of our most complex model. We conjecture that this two-feature model outperforms random guessing asymptotically; due to the model's extreme simplicity, we believe that this conjecture is a worthwhile direction for future theoretical work.

Introduction

The propositional satisfiability problem (SAT) is arguably the most widely studied NP-complete problem. Given a Boolean formula F, it asks whether there exists an assignment of truth values to the variables in F under which F evaluates to *true*. A prominent family of SAT instances is uniform random 3-SAT, which consists of formulae in conjunctive normal form, parameterized by a number of variables v, and of clauses c. Each clause consists of three literals (*i.e.*, variables or their negations); it is generated by selecting these literals uniformly at random, without replacement, from a set of v variables, and by negating each variable thus selected with probability 0.5. Each instances is created by independently generating c clauses. Because random 3-SAT instances are easy to generate and often hard to solve, they have frequently been used as a test bed for the design and evaluation of heuristic algorithms (see, *e.g.*, Le Berre, Roussel, and Simon, 2011).

In the early 1990s, it was observed that the probability that a random 3-SAT instance is satisfiable exhibits sharp threshold behavior when the control parameter $\alpha = c/v$ passes a critical value (Cheeseman, Kanefsky, and Taylor 1991; Mitchell, Selman, and Levesque 1992). The width of the window in which this solubility phase transition takes place becomes narrower as the instance size grows. Most interestingly, a wide range of state-of-the-art SAT solvers exhibit dramatically longer runtimes for instances in this critical region. Intuitively, note that instances are underconstrained when α is small (there are few constraints, and therefore many solutions), and overconstrained when α is large (there are many constraints, making it relatively easy to derive a contradiction). The so-called phase transition point occurs between these extremes, when the probability of generating a satisfiable instance is 0.5.

Crawford and Auton (1996) confirmed these findings in an extensive empirical study and proposed a more accurate formula for identifying the phase transition point: $c = 4.258 \cdot v + 58.26 \cdot v^{-2/3}$. Kirkpatrick and Selman (1994) used finite-size scaling, a method from statistical physics, to characterize size-dependent effects near the transition point, with the width of this transition narrowing as the number of variables increases. Yokoo (1997) studied the behavior of simple local search algorithms on uniform random 3-SAT instances, observing a peak in the hardness for solving satisfiable instances at the phase transition point. He attributed this hardness peak to a relatively larger number of local minima present in critically constrained instances, as compared to overconstrained satisfiable instances. Hoos and Stützle (1999) further investigated the behavior of stochastic local search algorithms on random 3-SAT instances at the phase transition, demonstrating substantial runtime variability across sets of instances with the same number of variables and clauses, and showing that the runtime over independent runs on the same instance tends to be exponentially distributed (for near-optimal parameter settings of the algorithm).

There is a useful analogy between uniform random 3-SAT problems and what physicists call "disordered materials": conflicting interactions in the latter are similar to the randomly negated variables in the former. Exploiting this connection, uniform random 3-SAT has been studied using meth-

ods from statistical physics. Monasson and Zecchina (1996; 1997) applied replica methods to determine the characteristics of uniform random 3-SAT and showed that the ground state entropy is finite at the phase transition. They concluded that the transition itself is due to the abrupt appearance of logical contradictions in all solutions and not to a progressive decrease in the number of models. Based on a conceptual link between uniform random 3-SAT and spin glass models, Mézard and Zecchina (2002) developed the survey propagation algorithm and demonstrated that it can solve random 3-SAT instances just below the phase transition point with up to 100 000 variables and 420 000 clauses. However, the phase transition becomes dramatically sharper with problem size; thus, these instances are almost always satisfiable and are much easier than those at the phase transition.

More recent work, much of it by our own group, has studied the use of machine learning methods to make instance-specific predictions about solver runtimes. Leyton-Brown, Nudelman, and Shoham (2002; 2009) introduced the use of such models for predicting the runtimes of solvers for NP-hard problems, and Nudelman et al. (2004) showed that using this approach, surprisingly accurate runtime predictions can be obtained for uniform random 3-SAT at the phase transition. That work also noted that satisfiable and unsatisfiable instances exhibited very different performance characteristics, and hence that training models on only SAT or UNSAT instances allowed much simpler—albeit, very dissimilar—models to achieve high accuracies. We subsequently reasoned that because unconditional models are able to predict runtimes accurately, despite the qualitative differences between the SAT and UNSAT regimes, the models must implicitly predict satisfiability status (Xu, Hoos, and Leyton-Brown 2007). Thus, we tried predicting satisfiability status directly, and achieved classification accuracies more than 70% on four well studied instance sets (86% on random 3-SAT at the phase transition with 400 variables). However, that finding was not the focus of our earlier work; instead, our main goal was to show how to leverage such predictions to obtain more accurate runtime predictions. Thus, critically, we included features that were capable of solving relatively small instances (e.g., so-called *probing features*). (Observe that such features have predictive value beyond their ability to solve an instance. Nevertheless, when a feature does solve an instance, it is quite easy for the model to accurately "predict" satisfiability status, boosting the reported classification accuracy.) Also, we only considered phase transition instances at a single size ($v = 400$). Taken together, these facts raise the concern that our previous finding may have been a small-size effect: that satisfiability status might be predictable only for small 3-SAT instances (which are in any case relatively easy to solve with modern methods), but that as instance size grows and runtimes increase exponentially, predictive accuracy could decrease to that of random guessing.

Main Contributions

Our work presented here is the first to thoroughly investigate the prediction of satisfiability status. We consider instances at the 3-SAT solubility phase transition, varying from 100 variables (for which median runtime of a high-performance SAT algorithm was too fast to measure accurately) to 600 variables (median runtime: 10 hours). We build classification models that achieve accuracies of about 70%, despite restricting ourselves to features that are not able to solve instances. We offer two arguments that these model accuracies are not a small-size effect. First, we show that our models' predictive accuracy remains roughly constant—and thus far better than that of random guessing—across the entire range of problem sizes. Second, we show that we can achieve very similar accuracy (across instances of all sizes) using a classifier trained only on very easy instances ($v = 100$).

We also conducted a detailed investigation into the minimal set of features sufficient for such accurate predictions. We found that two features sufficed to achieve good performance: one feature based on variation in the slack vectors of an LP relaxation of the SAT instance, and another based on the ratio of positive to negative literals in the formula. Finally, we present a three-leaf decision tree based on these two features, and trained only on the smallest instances, which achieved predictive accuracies across the entire range of instance sizes close to those of our most complex models. We conjecture that our findings hold asymptotically, and believe our findings could open a new direction for theoretical analysis of unform random 3-SAT. Essentially, the classifier can be viewed as a simple approximate (and polynomial-time) solver. In our experiments this solver achieved robust performance (accuracy above 65%) on a class of SAT instances that are extremely hard for current state-of-the-art solvers.

Experimental Setup

We considered uniform random 3-SAT instances generated at the solubility phase transition with v ranging from 100 to 600 variables in steps of 25. Following Crawford and Auton (1996), we estimated the location of the phase transition as $c = 4.258 \cdot v + 58.26 \cdot v^{-2/3}$ clauses. For each value of v, we generated 1000 instances with different random seeds, using the same instance generator as SAT competitions since 2002 (Simon 2002). In total, we thus obtained 21 instance sets jointly comprising 21 000 3-SAT instances. For $v = 100$, we generated an additional 25 000 instances; we refer to this instance set as $v100(large)$.[1]

We solved all of our instances using `kcnfs07` (Dubois and Dequen 2001) with a budget of 36 000 CPU seconds per instance, with the exception of 2 instances for $v = 575$ and 117 instances for $v = 600$. For these, we performed an additional 5 runs of `adaptg2wsat09++` (Li and Wei 2009) with a cutoff time of 36 000 CPU seconds, which also failed to solve them. Because this cutoff is more than 100 times larger than the longest runtime of `adaptg2wsat09++` on any of the solvable 600-variable instances, and the largest increase in running time needed for solving an additional instance for $v > 475$ was lower than a factor of 6.5, we believe that these instances

[1] To verify that we were indeed generating instances at the phase transition point, we examined the fraction of satisfiable and unsatisfiable instances in each set. The majority contained between 49 and 51% satisfiable instances (with mean 50.2% and standard deviation 1.6%), and there was no indication that deviations from the 50% mark correlated with instance size. Our large set of instances with $v = 100$ contained 49.5% satisfiable instances.

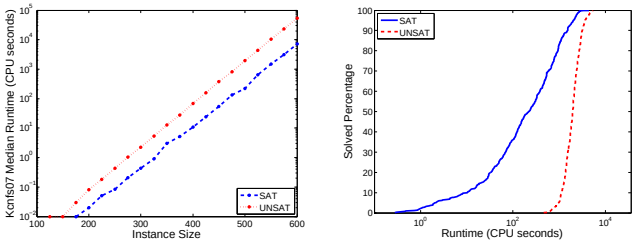

Figure 1: *Left: Median runtime of* `kcnfs07` *for each instance set. The solutions of some instances in* $v \geq 575$ *were estimated by running* `adaptg2wsat09++` *for 36 000 CPU seconds. Right: Cumulative distribution function of* `kcnfs07`*'s runtime for* $v = 500$.

are unsatisfiable and treated them as such for the remainder of our study. (Readers who feel uncomfortable with this approach should feel free to disregard our results for $v \geq 575$; none of our qualitative conclusions is affected.)

Figure 1 (left) shows the median runtime of `kcnfs07` on both satisfiable and unsatisfiable instances across our 21 instance sets. Median `kcnfs07` runtime increased exponentially with the number of variables, growing by a factor of about 2.3 with every increase of 25 variables beyond $v = 200$ (before this point, many instances were solved more quickly than the smallest CPU time we could measure).

We observed large variation in runtime on satisfiable instances, as illustrated in Figure 1 (right) for $v = 500$. Overall, unsatisfiable instances tended to be harder to solve, and to give rise to less runtime variation. Intuitively, to prove unsatisfiability, a complete solver like `kncfs07` needs to reason about the entire space of candidate assignments, while satisfiability may be proven by producing a single model. Depending on the number of solutions of a given instance, which is known to vary at the phase transition point, the search cost of finding the first one can vary significantly.

We opted to rely on a different classification method than used in our previous work (Xu, Hoos, and Leyton-Brown 2007); specifically, we used decision forests rather than SMLR. This change had two advantages: we obtained robust uncertainty estimates, were able to visualize models directly (as, indeed, we do later in this work) while achieving very good predictive accuracies. We also experimented with SMLR on the 21 instance sets used in this work; it yielded similar classification accuracies to those reported here (the result of a Mann-Whitney U test showed no significant difference). We constructed decision forests as collections of T decision trees (Ting 2002), with $T = 99$. Following Breiman (2001), given n training data points with k features each, for each tree we drew a bootstrap sample of n training data points sampled uniformly at random with repetitions; during tree construction, we sampled a random subset of $\log_2(k) + 1$ features at each internal node to be considered for splitting the data at that node. Predictions were based on majority voting across all T trees. In our case, the class labels were SAT and UNSAT. Since we used 99 trees, an instance i was classified as SAT if more than 44 trees predicted that it was satisfiable. We measured the decision forest's confidence as the fraction of trees that predicted i to be satisfiable; by choosing T as an odd number, we avoided the possibility of ties. We validated our decision forests on the same 3-SAT instance set at the phase transition used in our previous work (Xu, Hoos, and Leyton-Brown 2007). It produced predictive accuracy very close to that of SMLR (0.2% lower).

We used 61 cheaply computable instance features, of which 7 are related to problem size, 29 to graph-based representations of the CNF formula, 13 to balance properties, 6 to proximity to a Horn formula, and 6 to LP relaxations. These features were obtained from the feature computation code used in Xu et al. (2009); The feature computation time depends on the size of the instance under consideration (e.g., about 41.1 CPU seconds on average for all features on a single instance with $v = 550$, of which about 41.0 CPU seconds were spent on computing the 6 LP-based features). We normalized each feature so that it had a mean of 0 and standard deviation of 1 across the training set.

For each size v, we first partitioned the respective instance set into two subsets based on satisfiability status. Then, we randomly split each subset 60:40 into training and test sets. Finally, we combined the training sets for SAT and UNSAT to form the final training set, and the SAT and UNSAT test sets to form the final test set. We trained our decision forests on the training sets only, and used only the test sets to measure model accuracy. In order to reduce variance in these accuracy measurements we repeated this whole process 25 times (with different random training/test splits); the results reported in this paper are medians across these 25 runs.

We collected all runtime and feature data on a computer cluster with 840 nodes, each equipped with two 3.06 GHz Intel Xeon 32-bit processors and 2GB of RAM per processor. The decision forest classifier was implemented in Matlab, version R2010a, which we also used for data analysis.

Experimental Results

Predictive quality. At the solubility phase transition, uniform random 3-SAT instances are equally likely to be satisfiable or unsatisfiable. Thus, random (and, indeed, deterministic) guessing can achieve predictive accuracy of only 50%. Our first goal was to investigate the extent to which our models were able to make more accurate predictions. We found that they did; specifically, they achieved accuracies of between about 70% and 75%, as shown in Figure 2 and Table 1. The result of a Mann-Whitney U test showed no significant difference in the frequency of the two possible predictive errors (predicting SAT as UNSAT and vice versa).

Classifier confidence. Figure 3 shows two sample distributions ($v = 200$ and $v = 500$) of classifier confidence. The plots for other instance sets (not shown here) were qualitatively similar. Recall that we measured the confidence of the classifier by the fraction of 'SAT' predictions among the 99 trees. Therefore, the classifier had complete confidence if all 99 predictions were consistent, and had the least confidence if the numbers of 'SAT' predictions and 'UNSAT' predictions were the same. As illustrated in Figure 3, the classifier had low levels of confidence more often than high levels of confidence; however, these low confidence levels occurred on somewhat fewer instances as instance size grew.

As one might hope, we found that confidence was positively correlated with classification accuracy. This can be seen by comparing the height of the bars for correct and wrong

Variables	Median Accuracy	Incorrect "SAT"	Incorrect "UNSAT"
100	0.694	0.138	0.168
125	0.709	0.125	0.166
150	0.702	0.148	0.150
175	0.702	0.155	0.144
200	0.682	0.153	0.164
225	0.703	0.148	0.153
250	0.697	0.158	0.148
275	0.740	0.140	0.120
300	0.714	0.143	0.143
325	0.749	0.122	0.130
350	0.704	0.151	0.143
375	0.697	0.148	0.155
400	0.724	0.143	0.135
425	0.727	0.138	0.135
450	0.740	0.128	0.132
475	0.744	0.118	0.138
500	0.737	0.130	0.133
525	0.733	0.143	0.125
550	0.747	0.120	0.133
575	0.762	0.113	0.125
600	0.732	0.129	0.139

Table 1: *The performance of decision forests with 61 features on our 21 primary instance sets. We report median classification accuracy over 25 replicates with different random splits of training and test data, as well as the fraction of false positive and false negative predictions.*

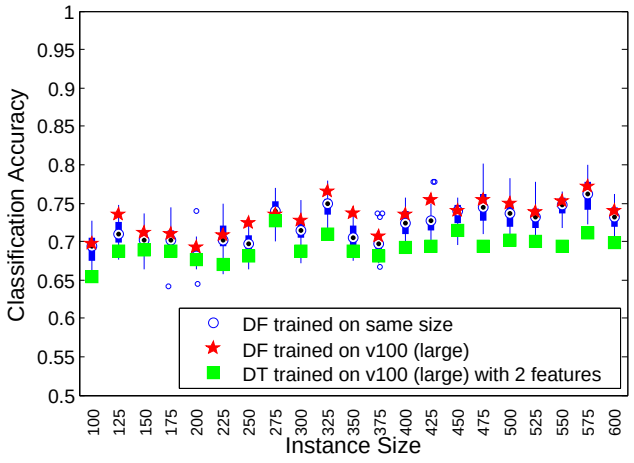

Figure 2: *Classification accuracies achieved on our 21 primary instance sets. The box plots are based on 25 replicates of decision forest models, trained and evaluated on different random splits of training and test data. The median predictive accuracies of using the decision forest trained on $v100(large)$ are shown as stars. The median predictive accuracies of using a single decision tree trained on $v100(large)$ based on two features are shown as squares.*

predictions at each predicted probability of SAT. When predicted probability of SAT was close to 0 or 1, the classifier was almost always correct, and when the predicted probability of SAT was close to 0.5, accuracy dropped towards 0.5 (i.e., that of random guessing).

The decision forest's confidence was also correlated with `kcnfs07`'s runtime. As shown in Figure 4, instances tended to be easier to solve when the predicted probabilities of SAT were close to either 0 or 1. Recall that variation in runtime was more pronounced on satisfiable instances, as previously illustrated in Figure 1 (right).

Problem size: Pairwise significance tests. We now examine the hypothesis that our models' predictive accuracy decreases as problem size grows. We offer two arguments that this hypothesis should be rejected. First, we describe the results of a pairwise comparison of the classification accuracies obtained from the full decision forest models trained for each instance size. For each pair of data sets with instance sizes i and j ($i > j$), Figure 5 shows a dark dot when classification accuracy on size i was significantly higher than on size j, and a light dot when classification accuracy on size i was significantly lower than on size j, according to a Mann-Whitney U test. Among the 210 paired comparisons with significance level 0.05, there are 133 dark dots (63.3%), and 21 light dots (10.0%). Thus, we found little evidence that predictive accuracy decreases as instance size grows; indeed, our data appears to be more consistent with the hypothesis that predictive accuracy *increases* with instance size.

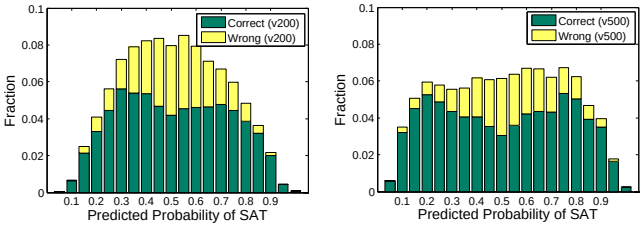

Figure 3: *Classifier confidence* vs *fraction of instances. Left: $v = 200$; Right: $v = 500$.*

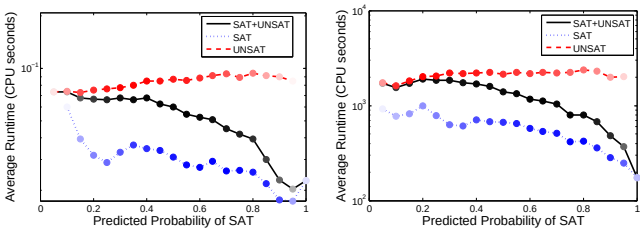

Figure 4: *Classifier confidence* vs *instance hardness. Each marker ($[x, y]$) shows the average runtime of `kcnf07` over a bin of instances with classifier confidence (predicted probability of SAT) between $x - 0.05$ and x. Each marker's intensity corresponds to the amount of data inside the bin. Left: $v = 200$; Right: $v = 500$.*

Problem size: Generalizing from small to large problems. We now give a second argument against the hypothesis that predictive accuracy decreases with problem size: models trained only on the smallest problems achieved high levels of predictive accuracy across the whole range of problem sizes. The stars in Figure 2 indicate the performance of the decision forest trained on $v100(large)$ evaluated on problems of other sizes. This single model performed about as well—indeed, in many cases better—than the models specialized to different problem sizes.

We note that, in order to evaluate a model trained on instances of size x on (test set) instances of size $y \neq x$, feature values were normalized to obtain mean 0 and standard deviation 1 across the *training set* instances of size y. This normalization was then applied to the features computed for test set instances of size y, and the original model was evaluated on these normalized feature values. This additional step

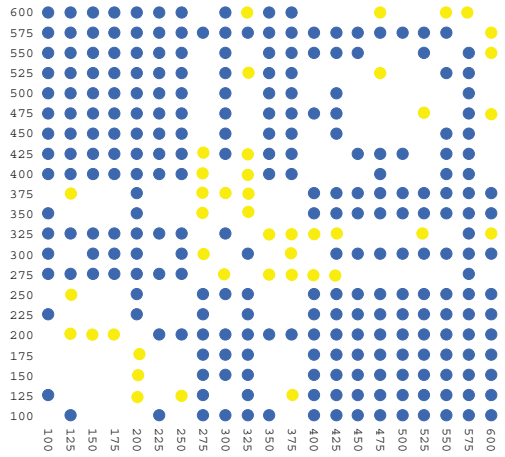

Figure 5: *Statistical significance of pairwise differences in classification accuracy for our 21 primary instance sets. Light dots: accuracy on the smaller instance size is significantly higher than on the larger size. Dark dots: accuracy on the smaller instance size is significantly lower than on the larger size. No dot: the difference is insignificant.*

was easy in practice, as there is no obstacle to generating an arbitrary number of uniform-random 3-SAT formulae of a given size, and as feature computation scales polynomially in problem size.

Although we do not report the results here, we also trained decision forests on each of the other 20 instance sets; in each case, we found that the models generalized across the entire range of problem sizes in qualitatively the same way.

Most predictive features. Next, we tried to identify the smallest set of features that could be used to build accurate models. We imagined that most predictive features might be different for large instances and for small instances, and therefore divided the 21 instance sets into two groups, *small* (10 instance sets, $v = 100$ to 325) and *large* (10 instance sets, $v = 375$ to 600). We did not use the 350-variable set in this analysis, in order to keep the two groups balanced. We considered every subset of our 61 features having cardinality 1, 2 or 3, and in each case, measured the accuracy of the corresponding model. For both groups, we found that the best 1- and 2-feature sets were subsets of the best 3-feature subset. Next, we performed forward selection to build subsets of up to 10 features, as exhaustive enumeration was infeasible above 3 features. Specifically, starting with the best 3-feature set, and for every feature not in the set, we determined the mean of median classification accuracy across all of the instance sets in the group. We then added the feature with the best such accuracy to the set and repeated the process, again considering adding another feature. We repeated these steps until we had 10 features in our set.

We list the sets of features that we obtained, as well as the improvement in classification accuracy achieved at each step, in Table 2. In both cases, we were be able to achieve good classification accuracy with a small number of features; for each number of features, classification accuracy on large instances was better than on small instances. The two most informative features were identical for both groups, and adding additional features beyond this point offered little marginal benefit. (We note that, since we performed an exhaustive analysis of feature subsets up to size three, this is not an artifact of our subset selection technique: there really was no 3-feature subset that performed substantially better than the best 2-feature subset.) It is therefore worth understanding the meaning of these two features. In words, `LPSLACK_coeff_variation` is the coefficient of variation in the integer slack vector of the linear programming relaxation of the integer programming formulation of SAT; `POSNEG_ratio_var_mean` is the average imbalance in the number of positive and negated occurrences of each variable.

Definition 1 (`LPSLACK_coeff_variation`) *Let C denote the set of clauses, let L denote the set of positive literals, and let $\overline{L}$ denote the set of negative literals. Then let v^* denote a solution to the linear program*

$$\text{maximize} \sum_{c \in C} \left(\sum_{i \in L \cap c} v_i + \sum_{j \in \overline{L} \cap c} (1 - v_j) \right)$$

$$\text{subject to} \sum_{i \in L \cap c} v_i + \sum_{j \in \overline{L} \cap c} (1 - v_j) \geq 1 \qquad \forall c \in C.$$

Let $\texttt{LPSLACK}_i = \min\{1 - v_i^*, v_i^*\}$, i.e., v_i^*'s proximity to integrality, μ^* the mean of $\texttt{LPSLACK}_i$ and σ^* the standard deviation of $\texttt{LPSLACK}_i$ over all i. Then `LPSLACK_coeff_variation` $= \sigma^*/\mu^*$, i.e., the coefficient of variation of `LPSLACK`.

Definition 2 (`POSNEG_ratio_var_mean`) *Let P_i and N_i denote the number of positive and negated occurrences of variable i, and let n denote the number of variables. Then*

$$\texttt{POSNEG\_ratio\_var\_mean} = \frac{2}{n} \cdot \sum_i \left| 0.5 - \frac{P_i}{P_i + N_i} \right|.$$

Simple classifier. So far, we have presented three main findings: (1) that our models achieved high accuracies; (2) that models trained on small instances were effective for large instances; (3) that a model consisting of only two features was nearly as accurate as the full decision forest models. We now show that all of these findings also held simultaneously: that we were able to achieve high accuracies using a two-feature model trained only on small instances. Specifically, we constructed a single decision tree (rather than a random forest) using only the `LPSLACK_coeff_variation` and `POSNEG_ratio_var_mean` features, and trained it using only our easiest instances, $v100(large)$. We further simplified this model by setting the parameter `minparent` of the tree building procedure to 10 000. The `minparent` parameter defines the smallest number of observations that impure nodes may contain before they are allowed to further split; setting it to such a large value forced the decision tree to be extremely simple. This tree's performance is indicated by the squares in Figure 2. Overall, it achieved remarkably good predictive accuracies, always exceeding 65%.

Figure 8 shows the decision tree. First, it classifies instances as satisfiable if `LPSLACK_coeff_variation` takes a value above its mean: that is, if `LPSLACK` exhibits large variance across the variables in the given formulae (region `A`).

Small instance sets: between 100 and 325 variables			Large instance sets: between 375 and 600 variables		
Features ordered by FW	Classification Accuracy	Stepwise Improvement	Features ordered by FW	Classification Accuracy	Stepwise Improvement
LPSLACK_coeff_variation	**0.614**	–	**LPSLACK_coeff_variation**	**0.646**	–
POSNEG_ratio_var_mean	**0.670**	**0.056**	**POSNEG_ratio_var_mean**	**0.696**	**0.050**
LP_OBJ	0.681	0.011	LPSLACK_mean	0.706	0.010
VG_mean	0.688	0.007	LP_int_ratio	0.714	0.008
LPSLACK_max	0.690	0.002	VCG_clause_max	0.720	0.006
VG_max	0.692	0.002	CG_mean	0.721	0.001
VCG_var_max	0.694	0.002	TRINARYp	0.725	0.004
POSNEG_ratio_var_max	0.694	0.000	HORNY_var_coeff_variation	0.727	0.002
LPSLACK_mean	0.695	0.001	DIAMETER_entropy	0.728	0.001
LP_int_ratio	0.697	0.002	POSNEG_ratio_clause_entropy	0.728	0.000

Table 2: *The mean of median classification accuracy with up to 10 features selected by forward selection. The stepwise improvement for a feature f_i at forward selection step k is the improvement when we add f_i to the existing $k-1$ features. Left: mean of median over* small *instance sets. Right: mean of median over* large *instance sets. Each median classification accuracy is based on the results of 25 runs of classification with different random splits of training and test data.*

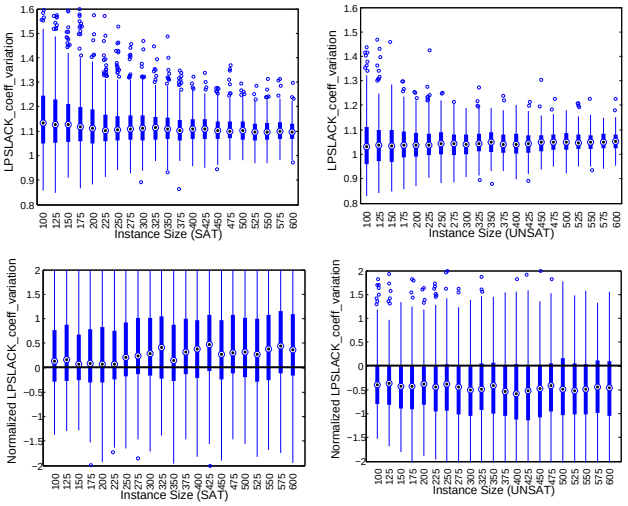

Figure 6: *Distribution of* `LPSLACK_coeff_variation` *over instances in each of our 21 sets. Left: SAT; Right: UNSAT. Top: original value; Bottom: value after normalization. The line at $y = 0.0047$ indicates the decision threshold used in the tree from Figure 8.*

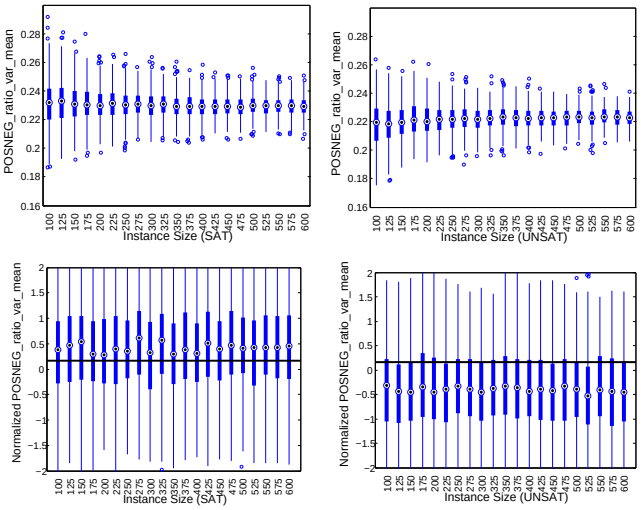

Figure 7: *Distribution of* `POSNEG_ratio_var_mean` *over instances in each of our 21 instance sets. Left: SAT; Right: UNSAT. Top: original value; Bottom: value after normalization. The line at $y = 0.1650$ indicates the decision threshold used in the tree from Figure 8.*

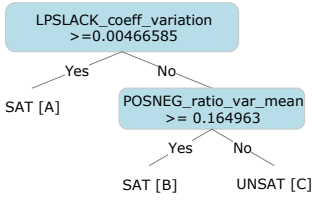

Figure 8: *The decision tree trained on $v100(large)$ with only the (normalized) features* `LPSLACK_coeff_variation` *and* `POSNEG_ratio_var_mean`*, and with* `minparent` *set to 10 000.*

(Recall that the feature was normalized to have mean 0.) When `LPSLACK_coeff_variation` takes a value below its mean, the model considers the balance between positive and negative literals in the formula (`POSNEG_ratio_var_mean`). If the literals' signs are relatively balanced, the model predicts unsatisfiability (region `C`). Otherwise, it predicts satisfiability (region `B`). To gain further understanding about the effectiveness of this model, we partitioned each of our 21 data sets into the three regions and observed the fraction of each partition that was correctly labeled by the tree. These fractions were between 60 and 70% (region `A`), between 70 and 80% (region `C`), and about 50% (region `B`).

Finally, Figures 6 and 7 show the distribution of the `LPSLACK_coeff_variation` and `POSNEG_ratio_var_mean` features over each of our 21 instance sets, before and after normalization, and considering satisfiable and unsatisfiable instances separately. We observe that both features' pre-normalization variation decreased with instance size, while their median values remained relatively constant. After normalization, both features' distributions remained very similar as instance size increased. The decision thresholds used by our simple decision tree are plotted as solid horizontal lines in these figures. Both thresholds were located near the 25th or 75th percentiles of the respective distributions, regardless of instance size.

We end by stating the following conjecture, which we hope will serve as a basis for future theoretical investigation.

Conjecture 1 *The decision tree in Figure 8 can determine the satisfiability status of uniform-random 3-SAT instances at the phase transition with accuracy bounded strictly above 0.5 as instance size grows.*

Indeed, we note that a simpler version of this conjecture might be easier to analyze, and is implied by Conjecture 1. Specifically, we can make the same claim about either of two one-node decision trees: (1) the subtree rooted at the POSNEG_ratio_var_mean node; (2) the root node, with the "no" edge leading to an "UNSAT" leaf. We also note that, while our model is dependent on normalized feature values, it is likely possible to obtain an analytic solution for the normalization factors.

Conclusions

Uniform random 3-SAT instances from the solubility phase transition are challenging to solve considering their size. Nevertheless, we have shown that the satisfiability of such instances can be predicted efficiently and with surprisingly high accuracy. We have demonstrated that high predictive accuracies (19% – 26% better than random guessing) can be achieved across a wide range of instance sizes, and that there is little support for the hypothesis that this accuracy decreases as instance sizes grow. The predictive confidence of our classifiers correlates with the predictive accuracy obtained and with the runtime of a state-of-the-art complete SAT solver. A classifier trained on small, very easy instances also performed well on large, extremely challenging instances. Furthermore, the features most important to models trained on different problem sizes were substantially the same. Finally, we showed that using only two features, LPSLACK_coeff_variation and POSNEG_ratio_var_mean, we could build a trivial, three-leaf decision tree that achieved classification accuracies only slightly below those for our most complex decision forest classifier. Examining the operation of this model, we observe the surprisingly simple rules that instances with large variation in LPSLACK (distance of LP solutions to integer values) across variables are likely to be satisfiable, and that instances with small variation of LPSLACK and roughly balanced numbers of positive and negated occurrences of variables are likely to be unsatisfiable. We hope that these rules will lead to novel heuristics for SAT solvers targeting random instances, and will serve as a starting point for new theoretical analysis of uniform random 3-SAT at the phase transition.

Acknowledgements. We thank Uri Feige for an inspirational discussion that led us to undertake this project. This research was supported by computational resources from Compute Canada / Westgrid and funding by the Mprime NCE.

References

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. M. 1991. Where the really hard problems are. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 331–337.

Crawford, J. M.; and Auton, L. D. 1996. Experimental results on the crossover point in random 3SAT. *Artificial Intelligence* 81:31–35.

Dubois, O.; and Dequen, G. 2001. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 248–253.

Hoos, H. H.; and Stützle, T. 1999. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence Journal* 112:213–232.

Kirkpatrick, S.; and Selman, B. 1994. Critical behavior in the satisfiability of random boolean formulae. *Science* 264:1297–1301.

Le Berre, D.; Roussel, O.; and Simon, L. 2011. The international SAT Competitions. www.satcompetition.org.

Leyton-Brown, K.; Nudelman, E.; and Shoham, Y. 2002. Learning the empirical hardness of optimization problems: the case of combinatorial auctions. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 556–572.

Leyton-Brown, K.; Nudelman, E.; and Shoham, Y. 2009. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM* 56(4):1–52.

Li, C. M.; and Wei, W. 2009. Combining adaptive noise and promising decreasing variables in local search for SAT. Solver description, SAT competition 2009.

Mézard, M.; and Zecchina, R. 2002. Random k-satisfiability: from an analytic solution to a new efficient algorithm. *Phys. Rev. E* 66:1357–1370.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *Proceedings of the American Association for Artificial Intelligence*, 459–465.

Monasson., R.; and Zecchina, R. 1996. Entropy of the k-satisfiability problem. *Phys. Rev. Lett.* 76:3881–3885.

Monasson., R.; and Zecchina, R. 1997. The statistical mechanics of the random k-satisfibility model. *Phys. Rev. E* 56:1357–1370.

Nudelman, E.; Leyton-Brown, K.; Devkar, A.; Shoham, Y.; and Hoos, H. 2004. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 438–452.

Simon, L. 2002. SAT competition random 3CNF generator. www.satcompetition.org/2003/TOOLBOX/genAlea.c.

Ting, K. M. 2002. An instance-weighting method to induce cost-sensitive trees. *IEEE Trans. Knowl. Data Eng.* 14(3):659–665.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2009. SATzilla2009: An automatic algorithm portfolio for SAT. Solver description, 2009 SAT Competition.

Xu, L.; Hoos, H. H.; and Leyton-Brown, K. 2007. Hierarchical hardness models for SAT. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 696–711.

Yokoo, M. 1997. Why adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 356–370.