Artificial Intelligence Research and Development
M. Villaret et al. (Eds.)
© 2021 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA210115

On the Temperature of SAT Formulas

Jesús GIRÁLDEZ-CRU [a,1], Pedro ALMAGRO-BLANCO [b]
[a] *DaSCI, DECSAI, Universidad de Granada (jgiraldez@ugr.es)*
[b] *Dpto. Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla (palmagro@us.es)*

Abstract.
The remarkable advances in SAT solving achieved in the last years have allowed to use this technology in many real-world applications of Artificial Intelligence, such as planning, formal verification, and scheduling, among others. Interestingly, these industrial SAT problems are commonly believed to be easier than classical random SAT formulas, but estimating their actual hardness is still a very challenging question, which in some cases even requires to solve them. In this context, realistic pseudo-industrial random SAT generators have emerged with the aim of reproducing the main features shared by the majority of these application problems. The study of these models may help to better understand the success of those SAT solving techniques and possibly improve them.

In this work, we present a model to estimate the *temperature* of real-world SAT instances. This temperature represents the degree of distortion into the expected structure of the formula, from highly structured benchmarks (more similar to real-world SAT instances) to the complete absence of structure (observed in the classical random SAT model). Our solution is based on the Popularity-Similarity (PS) random model for SAT, which has been recently presented to reproduce two crucial features of application SAT benchmarks: scale-free and community structures. The PS model is able to control the hardness of the generated formula by introducing some randomizations in the expected structure. Our solution is a first step towards a hardness oracle based on the temperature of SAT formulas, which may be able to estimate the cost of solving real-world SAT instances without solving them.

Keywords. SAT, hardness, temperature, Popularity-Similarity, entropy

1. Introduction

Despite the remarkable progress in SAT solving techniques in the last years, determining the time required to solve a given SAT instance by a certain algorithm is still today one of the most interesting and challenging questions in the SAT community. The simplest solution is to run that algorithm until termination, but unfortunately this task may be extremely costly, and hence infeasible. An alternative solution would be to *accurately estimate* its solving time.

Most of the traditional approaches on the study of the hardness of SAT instances have focused on the so-known classical random model of SAT formulas [28], where a

[1]Corresponding author. Supported through the Juan de la Cierva program (FJCI-2017-32420, IJC2019-040489-I).

random formula $F_k(n,m)$ is a set of m clauses over n variables, and clauses are chosen uniformly and independently among all the $2^k\binom{n}{k}$ non-trivial clauses of length k.[2] The empirical hardness of this model has been extensively studied [28,35]. In particular, for any fixed n and $k > 2$, there exists an easy-hard-easy pattern depending on the clause/variable ratio m/n, which is also related to the satisfiability of the formula. Therefore, the hardness of random SAT formulas simply depends on k, n and m. The natural question is whether a *simple* hardness characterization also exists for real-world SAT instances, which is the question that motivates our work. Far from providing such a characterization, in this work we analyze the relation between the hardness of real-world SAT instances and a simple parameter of them, as a first step toward facing this challenge.

Although the reasons of the success of CDCL SAT solvers on the heterogeneous set of application SAT instances are not completely understood yet [34,6], there have been some recent attempts to study common features on these industrial problems [1] with the aim of explaining the good performance of these solvers on this benchmark. Because of this heterogeneity, realistic pseudo-industrial random SAT instances generators have emerged, stated as one of the most important challenges in propositional search [32]. The cornerstone of these models is to produce random formulas with computational properties similar to real-world instances. The Popularity-Similarity (PS) random model [20] has been proposed as one of these realistic random SAT generators.

The PS model defines an expected structure composed of scale-free structure [3] (the number of variables occurrences follows a power-law distribution, i.e., a few variables occurs a lot whilst most of them occur very little) and community structure [5] as a result of high clustering (the set of variables can be split into disjoint communities such that variables mostly occur in clauses with other variables of the same community). They are two common features in most real-world SAT benchmarks. Inspired by the *entropy* of physical systems, the PS model uses the *temperature* T as a parameter to control the degree of distortion into this structure. This is, at $T = 0$ the model produces that structure with high probability (hence the generated formula is more similar to real-world SAT instances), whereas at high temperature the model behaves like the classical random SAT model (hence the generated formula does not exhibit any structure at all).

In practice, it has been observed that CDCL solvers focus on frequent variables and on variables of the same community [6,4,19,7,2]. Using the synthetic PS model, it has been also observed that CDCL solvers perform better on PS formulas with low temperature. On the contrary, SAT solvers specialized in classical random SAT formulas perform better on PS formulas with high temperature [20]. We conjecture that the hardness of real-world SAT formulas depends on a notion of temperature, which characterizes the distortion into the structure of a particular formula from the structure exhibited in most real-world SAT benchmarks. To this end, we consider a real-world SAT problem as an instantiation of the PS model [20], and its temperature corresponds to the value of T in this instantiation. We emphasize that this does not require the real-world instance to have any structure (e.g., high T).

In order to test our hypothesis, we need first to compute the temperature of a given SAT formula. Unfortunately, there is no known analytical method to this purpose [31]. In our work we present an extensive study of Machine Learning (ML) regression methods to estimate it. In particular, we analyze the performance of different ML techniques trained

[2] A non-trivial clause of length k contains k distinct, non-complementary literals.

with PS formulas generated at distinct temperatures, and measure their accuracy in the estimation. We also evaluate the robustness of each ML technique when the training set is altered with perturbations in the generation step. Empirically, we show that ML techniques based on ensembles (e.g., random forest) are the most accurate approaches, and they remain robust to perturbations.

2. Related Work

There are in the literature other realistic SAT generators, such as the scale-free SAT model [4], which generates purely scale-free SAT instances, and the community attachment model [18], which is able to produce formulas with clear community structure. We recall that both features can be observed in PS formulas. The hardness of scale-free SAT formulas [4] also depends on the exponent β of the power-law distribution that characterizes them [15,16,30].

A seminal contribution on ML applied to SAT solving is SATzilla [36,23]. A SAT solver unlikely dominates all others on unrestricted SAT instances, but it may show a particularly good performance on a certain class of problems [26]. On this idea, SATzilla proposes a per-instance algorithm portfolio that estimates the best solver to solve a given formula from a predefined set. This portfolio approach has also been successfully applied in other works [24,27].

3. The PS Random SAT Model

In this section we provide a brief description of the Popularity-Similarity (PS) random SAT model [20]. For further details, we address the reader to the original reference.

The PS model is able to generate random SAT formulas with both scale-free and community structure as the result of two orthogonal forces: popularity and similarity. To model them, every variable i is assigned radial and angular coordinates r_i and θ_i, representing respectively its popularity and its similarity to other variables. Popular variables have a small radius and similar variables have close angles. These two coordinates are also assigned to every clause j. In this model, the probability $P(i \leftrightarrow j)$ of a variable i occurring in a clause j (with any sign) is:

$$P(i \leftrightarrow j) = \left(1 + \left(\frac{r_i^{\beta} \cdot r'^{\beta'}_j \cdot \theta_{ij}}{R}\right)^{1/T}\right)^{-1}$$

where r_i and r_j represent the radiuses of i and j respectively, θ_{ij} is the angular distance between them, β and β' are respectively the exponents of the power-law distributions for variables occurrences and clauses length, R is a normalization constant ensuring the expected formula size, and T is the temperature of the model. Therefore, the temperature T precisely controls the entropy of the system, i.e., the degree of distortion into the expected probabilities. The aforementioned structures are the result of this probability distribution, which is clearly non-uniform at low T: it is more likely that a clause j contains a popular variable (low r_i) or a variable similar to it (low θ_{ij}). In contrast, the probability distribution becomes (close to) uniform for high values of T, as in the classical random SAT model.

4. ML-based Regression Techniques

In this section, we provide a general overview of the regression problem to solve, and the techniques we use for that task.

Let us consider an instance ϕ, which is characterized by a vector $\mathbf{x}_\phi = [x_\phi^1, \dots, x_\phi^n]$ of n features. Being $x^* \notin \mathbf{x}$ the target feature to estimate, the problem consists of finding the function f s.t. $f(\mathbf{x}) = x^* \pm \varepsilon$ that minimizes ε. In our case, ϕ represents a SAT instance, $\mathbf{x}$ its features, and x^* its temperature T. Since the temperatures of the PS instances used in the training step are known *a priori*, we use supervised ML techniques to estimate f. In the following section, it is discussed the set of features $\mathbf{x}$ used in our experiments.

In our problem, the target estimation x^* is a continuous value. Although there exist many different ML algorithms to predict these values, the *no-free-lunch theorem* [21] states that it cannot be known *a priori* which techniques show a good performance in a particular problem, and finding them usually requires a trial and error process.

In our experimental analysis, we evaluate a total of 13 distinct regression methods. They can be grouped into the following categories. (1) *Linear regression*: LR (ordinary least squares linear regression), SGD (Stochastic Gradient Descent), PassiveAgressive [10], RANSAC (Random Sample Consensus) [11], Theil-Sen (TS) [11], Huber [22], and Bayesian Ridge [33]. (2) *Ensemble methods*: RandomForest (RF) [8], ExtraTrees (ET) [17], AdaBoost (AB) [14], and XGBoost (XGB) [9]. (3) *Neural networks*: FNN (Feed-forward Neural Network) with one hidden layer, 64 hidden neurons, and *Adam* optimizer [25]. (4) *Other methods*: K-neighbors. For all these methods, we use the implementation in sklearn [12].

For space constraints, we only give a general overview of these techniques. Linear regression methods will help us to evaluate possible linear relations between the set of features used to represent a formula and its temperature. Ensemble methods construct a set of *weak regressors* and aggregate their predictions reducing the overfitting that would be obtained when applying the regressors individually. It is well known that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions. Due to the already good performance achieved by the FNN used in our experiments, we have not considered more complex FNN, or even other graph neural network architectures. However, it would be interesting to study them and analyze their performance in future works.

In our experiments, we also use Bayesian optimization [29] to tune the hyperparameters of the ML techniques in order to increase their accuracy. In order to evaluate each regression model, we perform a 10-folds cross-validation.

5. Analysis of the Temperature Estimation

In this section, we present an exhaustive experimental analysis of regression techniques to estimate the temperature of SAT instances.

5.1. Experimental Set-Up

Generation of SAT formulas. The training set is composed of an heterogeneous set of PS random SAT formulas, differing in their number of variables $n \in [100 \dots 5000]$, and their clause/variables densities $m/n \in [2 \dots 8]$. For each value of n and m/n, we generate 100 random PS formulas with distinct temperatures. Our main benchmark results into a total of 7200 SAT instances, containing both satisfiable and unsatisfiable formulas. All formulas are 3-CNF and much smaller than real-world SAT instances. However, we found experimentally that the formula size has no impact on the performance of ML methods.

The popularity and similarity of the generated formulas is controlled by the parameter β. In the main batch of experiments, we use $\beta = 1$, i.e., a very clear scale-free structure. We also evaluate the robustness of the regression techniques exposing the training set to some perturbations on β (see Section 5.3.1)

Values of Temperature. In the PS model, a small difference of T at low temperatures may result in major differences in the resulting structure of the generated formula, whereas at high temperatures this structure is almost unaltered. For this reason, instead of working directly with the temperature, we sample and estimate its logarithm. In particular, for each formula size we sample 100 uniformly distributed random values in the interval $[-1, 1]$. These values correspond to the logarithm of the temperature of the generated formulas, and they are the values used to train the regression models and to be estimated. Therefore, the actual temperature ranges in the interval $[1/e, e]$. Although this generates a reasonable range of temperatures, we also evaluate some perturbations on this interval in Section 5.3.2.

Set of features. Every SAT instance is characterized by a vector of features. Ideally, this vector contains a set of uncorrelated, fast-to-compute features of the formula. In our experiments, we use the extended and well-known set of features used in the SATzilla toolkit [36]. In particular, we use a total of 101 features,[3] including formula size, graph characteristics, and solver statistics. In Section 5.4 we evaluate the impact of reducing the number of features used to train the regression models, in order to analyze their feature importance.

Filtering out trivial instances. Random PS SAT instances at low temperatures may be very easy [20]. This is especially relevant in small formulas (e.g., $n = 100$), which might be even solved by simple preprocessing techniques. For this reason, we filter out those trivial instances from the benchmark because some SATzilla features include solver statistics. We observed that the resulting unbalance does not affect the performance of the regression models with the best accuracy, due to the already large number of formulas in the benchmark.

Accuracy of the model. In order to evaluate each regression technique, we use the well-known coefficient of determination R^2 between the actual temperature and its prediction. Recall that $R^2 \in [-\infty, 1]$ with positive values indicating the existence of a certain correlation between the observed data and their predictions (the higher the value of R^2, the better the prediction).

In our experiments we use the value $R^2 \geq 0.8$ to distinguish those regression methods achieving a strong correlation (i.e., a good accuracy in the prediction), although any other

[3]We skip the computation of LP-based and SLS-based features due to their long execution time for some formulas. We also skip diameter features, as done in the last SATzilla version.

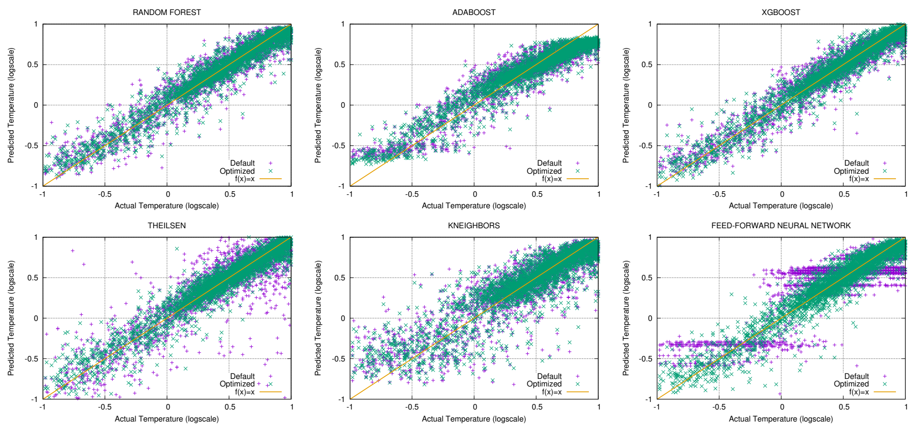

Figure 1. Predicted temperature versus actual temperature, for some regression techniques with a small standard deviation.

reasonable high value of R^2 could have been used instead. It is worth noticing that all methods with such a strong correlation also show a very small standard deviation of R^2, always lower than 0.15 (and usually much lower).

5.2. *Performance of Regression Methods*

The first natural question in our analysis is whether a regression method is able to estimate the temperature of a given PS SAT formula given the vector of SATzilla features for this formula.

In Table 1, we summarize the performance of each regression method on this problem, with both default and optimized hyperparameter settings, measuring the average and standard deviation of the coefficient of determination R^2.

Regression Method	Default	Optimized
LR	0.789 ± 0.35	0.789 ± 0.35
SGD	-1.230 ± 2.24	-2.5e8 ± 3e8
PassiveAggressive	-181.7 ± 271	-1.072 ± 0.13
RANSAC	-0.982 ± 1.53	0.511 ± 1.20
TS	-8.272 ± 25.3	**0.898** ± **0.03**
Huber	-0.085 ± 0.14	-0.076 ± 0.15
BayesianRidge	0.760 ± 0.46	0.694 ± 0.66
RF	**0.921** ± **0.01**	**0.931** ± **0.01**
ET	**0.922** ± **0.01**	**0.935** ± **0.01**
AB	**0.878** ± **0.02**	**0.896** ± **0.01**
XGB	**0.924** ± **0.01**	**0.935** ± **0.00**
FNN	0.687 ± 0.07	**0.912** ± **0.01**
Kneighbors	0.795 ± 0.03	0.796 ± 0.02

Table 1. Avg. coefficient of determination R^2 for different regression methods, with default and optimized hyperparameters.

We can observe that many of the methods with default settings are able of predicting the temperature of the formulas with a high accuracy, hence showing the robustness of

	Baseline	Perturbations on β				Perturbations on T interval	
β	$\beta = 1$	$\beta = 5/6$	$\beta = 4/6$	$\beta = 3/6$	Union(β)		
$log T$	$[-1,1]$					$[-2,2]$	$[-3,3]$
LR	0.789 ± 0.35	-3.199 ± 12.4	-0.568 ± 4.45	**0.856 ± 0.13**	0.785 ± 0.02	**0.936 ± 0.02**	**0.907 ± 0.01**
TS	**0.898 ± 0.03**	-2.223 ± 9.40	-6.625 ± 22.3	-197.1 ± 241	-213.6 ± 270	**0.939 ± 0.01**	**0.904 ± 0.02**
RF	**0.931 ± 0.01**	**0.932 ± 0.01**	**0.942 ± 0.01**	**0.958 ± 0.00**	**0.858 ± 0.01**	**0.955 ± 0.01**	**0.945 ± 0.01**
ET	**0.935 ± 0.01**	**0.935 ± 0.01**	**0.946 ± 0.01**	**0.960 ± 0.01**	**0.861 ± 0.01**	**0.956 ± 0.01**	**0.947 ± 0.01**
AB	**0.896 ± 0.01**	**0.882 ± 0.01**	**0.883 ± 0.01**	**0.904 ± 0.01**	0.678 ± 0.01	**0.928 ± 0.00**	**0.904 ± 0.01**
XGB	**0.935 ± 0.00**	**0.937 ± 0.01**	**0.948 ± 0.01**	**0.963 ± 0.01**	**0.872 ± 0.01**	**0.956 ± 0.01**	**0.943 ± 0.01**
FNN	**0.912 ± 0.01**	**0.904 ± 0.02**	**0.915 ± 0.01**	**0.927 ± 0.01**	0.780 ± 0.02	**0.917 ± 0.01**	**0.874 ± 0.02**

Table 2. Avg. coefficient of determination R^2 for different regression methods (with optimized hyperparameter), varying the training data in: (i) the value of β, and (ii) the interval of the temperatures.

our approach. Those accurate methods are based on ensembles, which are commonly more robust to hyperparameter tuning. FNN and Kneighbors present an acceptable accuracy with a low standard deviation. Although models LinReg and BayesianRidge also obtain an acceptable R^2 average, we cannot draw definitive conclusions due to their high deviation.

After optimizing hyperparemeters, we observe noticeable improvements in most of the methods, especially in FNN and TheilSen. In the case of FNN, the method shows a considerable improvement after adjusting the number of neurons in the hidden layer, the optimizer and the batch size. In the case of TheilSen, this linear method is able to learn from different subsets of the training data, and with an optimal configuration acquires resistance against outliers. This fact suggests a certain linear relation between SAT features and the temperature T. It is worth noticing that, in general, linear methods do not outperform (non-linear) methods based on neural networks and ensembles.

In Figure 1, we depict the predicted temperature versus the actual temperature of random PS formulas for six regression techniques. Notice that when R^2 is close to 1, the points on the figure must be close to the diagonal.

5.3. Robustness to Benchmark Perturbations

Regression Met.	All (baseline)	Basic (A)	Graph (B)	CDCL (C)	Union $A \cup B \cup C$
LinReg	0.789 ± 0.35	0.118 ± 2.14	**0.808 ± 0.08**	0.330± 0.07	0.749± 0.41
TheilSen	**0.898 ± 0.03**	-0.058 ± 2.62	**0.832 ± 0.03**	0.336± 0.06	-0.143± 3.11
RandomForest	**0.931 ± 0.01**	**0.911 ± 0.01**	**0.917 ± 0.01**	0.579± 0.04	**0.927± 0.01**
ExtraTrees	**0.935 ± 0.01**	**0.918 ± 0.01**	**0.924 ± 0.01**	0.565± 0.05	**0.933± 0.01**
AdaBoost	**0.896 ± 0.01**	**0.863 ± 0.02**	**0.864 ± 0.01**	0.331± 0.06	**0.876± 0.02**
XGBoost	**0.935 ± 0.00**	**0.916 ± 0.01**	**0.923 ± 0.02**	0.554± 0.02	**0.932± 0.01**
FNN	**0.912 ± 0.01**	**0.889 ± 0.01**	**0.932 ± 0.02**	0.535± 0.06	**0.911± 0.01**

Table 3. Avg. coefficient of determination R^2 for different regression methods (with optimized hyperparameter), for different features sets.

The next natural question is whether the accuracy of our approach is robust to perturbations. In particular, we consider perturbations in the training step modifying the parameters values of the generated PS random SAT formulas varying: (i) the scale-free structure of the benchmark, and (ii) the interval used to sample the values of the temperature.

5.3.1. Varying the Scale-Free Structure

In our main benchmark, all PS random formulas are generated with $\beta = 1$, i.e., with a clear scale-free structure. Now we analyze the performance robustness of the regression models in benchmarks just differing in the value of β used in the generation of the SAT formulas. In particular, we evaluate the cases with $\beta = \{5/6, 2/3, 1/2\}$, and the union of these four. In this experiment, we use the regression models with optimized parameters computed for $\beta = 1$. We restrict our analysis to the regression methods that already showed a good performance in the previous experiment, and adding LinReg as baseline. In Table 2, we summarize the results of this experiment.

We observe that, in all cases, the regression methods showing the best performance in all benchmarks (with any value of β) are techniques that already showed a very good performance with default parameters in the benchmark with $\beta = 1$, i.e., (non-linear) methods based on ensembles of decision trees: RandomForest, ExtraTrees and XGBoost. Therefore, these techniques seem to be robust to this perturbation, and hence they are good candidates to build a promising temperature estimator. Surprisingly, the LinReg method is able to obtain a good result in the case of the union of the different sets. This can be due to the fact that this set of instances is larger than the others, allowing the linear method to learn more effectively. In the case of TheilSen, the optimization made for $\beta = 1$ does not generalize correctly, thus this model is not robust.

5.3.2. Varying the Interval of Temperatures

Another perturbation to study the robustness of the regression methods is the interval used to sample the values of the temperature of the random PS formulas in the benchmark. We recall these values represent the logarithm of the temperature of the generated formulas. In the main benchmark, we use the interval $[-1, 1]$ (i.e., the temperature ranges in $[1/e, e]$). In this experiment, we generate two similar benchmarks only differing in this interval: $[-2, 2]$ and $[-3, 3]$. In Table 2 we also summarize the results of this perturbation.

We observe that all regression methods show a very good accuracy, suggesting that they all are robust to this kind of perturbation in the temperature. Interestingly, there seems to be a peak of performance in the intermediate interval $[-2, 2]$, i.e., using a relatively ample interval is beneficial, but using a too ample one is not.

5.4. Features Set and Feature Importance

In this section we analyze the relation between the set of features used by the regression techniques and their predictive capacity.

The set of features provided by the SATzilla toolkit can be divided into the following categories: (a) Basic features, (b) Graph features, (c) CDCL (and DPLL) features, and (d) other solving and timing statistics. In our analysis, we focus on the first three subsets, which respectively have 26, 25, and 24 features. We use the original SATzilla tool to produce all these features [36].

In Table 3, we summarize the coefficient of determination R^2 of different regression methods using a different set of features to train the models and compute the regression. Again, we use the models with optimized hyperparameter settings from the main experiment (see Table 1).

We observe that the set of features used to compute the regression may have dramatic consequences in its performance. In particular, we observe a very poor performance when only CDCL features are used. On the contrary, the performance is generally good using the set of Graph features. Surprisingly, the linear methods (LinReg and TheilSen) have very good performance when they are trained using Graphs features only, whilst their performance gets worse with the rest of the feature sets. This shows that these graph characteristics are able to *linearize* the relation between the SAT formula and its temperature. Linear estimators are less sensitive to overfitting than others. Therefore, the combination between this small set of features and these linear methods must be emphasized.

In the case of CDCL features, the determination coefficient R^2 of linear methods is similar to the one of the remaining non-linear regression techniques. This suggests a second linear relation between the set of features and the temperature. Nevertheless, it is much weaker. Basic features only produces good results with non-linear regression methods based on neural networks and ensembles, and this may explain the worse performance of linear methods when using them.

6. Conclusions

In this work, we have presented an extensive analysis of ML regression techniques in order to estimate the temperature of real-world SAT instances. Our experimental results show that ML methods based on ensembles (e.g., random forest) show the best performance, remaining robust to perturbations in the training step. Additionally, we have showed that a successful application like SATzilla is able to indirectly infer the temperature of the formulas using only a subset of (graph) features. As future work, we plan to extend this analysis with more sophisticated neural networks, including graph neural networks, and other automated ML techniques [13], and use these regression methods to estimate the temperature of real-world SAT instances.

References

[1] C. Ansótegui, M.L. Bonet, J. Giráldez-Cru, and J. Levy. Structure features for SAT instances classification. *Journal of Applied Logic*, 23:27–39, 2017.

[2] C. Ansótegui, M.L. Bonet, J. Giráldez-Cru, J. Levy, and L. Simon. Community structure in industrial SAT instances. *Journal of Artificial Intelligence Research*, 66:443–472, 2019.

[3] C. Ansótegui, M.L. Bonet, and J. Levy. On the structure of industrial SAT instances. In *Proc. of CP 2009*, pages 127–141, 2009.

[4] C. Ansótegui, M.L. Bonet, and J. Levy. Towards industrial-like random SAT instances. In *Proc. of IJCAI 2009*, pages 387–392, 2009.

[5] C. Ansótegui, J. Giráldez-Cru, and J. Levy. The community structure of SAT formulas. In *Proc. of SAT 2012*, pages 410–423, 2012.

[6] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proc. of IJCAI 2009*, pages 399–404, 2009.

[7] G. Baud-Berthier, J. Giráldez-Cru, and L. Simon. On the community structure of bounded model checking SAT problems. In *Proc. of SAT 2017*, pages 65–82, 2017.

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

[9] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proc. of KDD 2016*, pages 785–794, 2016.

[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[11] X. Dang, H. Peng, X. Wang, and H. Zhang. Theil-sen estimators in a multiple linear regression model. *Olemiss Edu*, 2008.

[12] L. Buitinck et al. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[13] M. Feurer, A. Klein, K. Eggensperger, J.T. Springenberg, M. Blum, and F. Hutter. Auto-sklearn: Efficient and robust automated machine learning. In *Automated Machine Learning - Methods, Systems, Challenges*, pages 113–134. Springer, 2019.

[14] Y. Freund and R.E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[15] T. Friedrich, A. Krohmer, R. Rothenberger, and A.M. Sutton. Phase transition for sclae-free SAT formulas. In *Proc. of AAAI 2017*, pages 3893–3899, 2017.

[16] T. Friedrich and R. Rothenberger. Sharpness of the satisfiability threshold for non-uniform random k-SAT. In *Proc. of SAT 2018*, pages 273–291, 2018.

[17] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[18] J. Giráldez-Cru and J. Levy. Generating SAT instances with community structure. *Artificial Intelligence*, 238:119–134, 2016.

[19] J. Giráldez-Cru and J. Levy. Locality in random SAT instances. In *Proc. of IJCAI 2017*, pages 638–644, 2017.

[20] J. Giráldez-Cru and J. Levy. Popularity-similarity random SAT formulas. *Artificial Intelligence*, 299:103537, 2021.

[21] Y. Ho and D.L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, 115(3):549–570, 2002.

[22] P.J. Huber. *Robust statistics*. Springer, 2011.

[23] F. Hutter, L. Xu, H.H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

[24] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC - instance-specific algorithm configuration. In *Proc. of ECAI 2010*, pages 751–756, 2010.

[25] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[26] K. Leyton-Brown, H.H. Hoos, F. Hutter, and L. Xu. Understanding the empirical hardness of *NP*-complete problems. *Communications of the ACM*, 57(5):98–107, 2014.

[27] Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann. Non-model-based algorithm portfolios for SAT. In *Proc. of SAT 2011*, pages 369–370, 2011.

[28] D.G. Mitchell, B. Selman, and H.J. Levesque. Hard and easy distributions of SAT problems. In *Proc. of AAAI 1992*, pages 459–465, 1992.

[29] J. Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

[30] O. Omelchenko and A. Bulatov. Satisfiability threshold for power law random 2-SAT in configuration model. *CoRR*, abs/1905.04827, 2019.

[31] F. Papadopoulos, M. Kitsak, M.A. Serrano, M. Boguñá, and D. Krioukov. Popularity versus similarity in growing networks. *Nature*, 489:537–540, 2012.

[32] B. Selman, H.A. Kautz, and D.A. McAllester. Ten challenges in propositional reasoning and search. In *Proc. of IJCAI 1997*, pages 50–54, 1997.

[33] M.E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(Jun):211–244, 2001.

[34] R. Williams, C.P. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proc. of IJCAI 2003*, pages 1173–1178, 2003.

[35] L. Xu, H.H. Hoos, and K. Leyton-Brown. Predicting satisfiability at the phase transition. In *Proc. of AAAI 2012*, 2012.

[36] L. Xu, F. Hutter, H.H. Hoos, and K. Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008.