International Journal of Computational Intelligence Systems (2022) 15:69
https://doi.org/10.1007/s44196-022-00122-4

RESEARCH ARTICLE

Characterizing the Temperature of SAT Formulas

Pedro Almagro-Blanco[1] · **Jesús Giráldez-Cru**[2]

Received: 26 February 2022 / Accepted: 25 July 2022

Abstract
The remarkable advances in SAT solving achieved in the last years have allowed to use this technology to solve many real-world applications, such as planning, formal verification and cryptography, 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 of these application problems to better understand the success of those SAT solving techniques on 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 random model for SAT, which has been recently presented to reproduce two crucial features of application SAT benchmarks: scale-free and community structures. This model is able to control the hardness of the generated formula by introducing some randomizations in the expected structure. Using our regression model, we observe that the estimated temperature of the applications benchmarks used in the last SAT Competitions correlates to their hardness in most of the cases.

Keywords SAT · Hardness · Temperature · Popularity–similarity · Entropy

Abbreviations

SAT	Boolean satisfiability problem
PS	Popularity–similarity random model
ML	Machine learning
RANSAC	Random sample consensus
FNN	Feed-forward neural networks
CDCL	Conflict-driven clause learning
DPLL	Davis–Putnam–Logemann–Loveland

1 Introduction

The Boolean satisfiability problem (SAT) is the problem of deciding whether the Boolean variables of a propositional formula can be assigned in such a way that the formula is evaluated as `true`. SAT is the first known NP-complete problem [14], which means that existing solvers can run during exponentially long executions in the worst case. Interestingly, in the last 2 decades, we have witnessed a remarkable progress in SAT solving techniques, which has allowed us to solve huge SAT instances in a reasonable amount of time. These advances are integrated in the so-known CDCL algorithm [43], and they have been especially relevant to solve real-world benchmarks, i.e., SAT instances encoding problems from industrial applications, including domains of Artificial Intelligence and Computer Science as diverse as hardware and software verification, security analysis, planning, formal methods, bioinformatics, and cryptography or compilers, among others.

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 in many cases. An alternative solution would be to *accurately estimate* its *hardness*, i.e., the solving time required to solve it.

✉ Jesús Giráldez-Cru
jgiraldez@ugr.es

Pedro Almagro-Blanco
palmagro@us.es

[1] CCIA, Universidad de Sevilla, Seville, Spain

[2] DaSCI, DECSAI, Universidad de Granada, Granada, Spain

Published online: 24 August 2022

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 [37], where a 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.[1] The empirical hardness of this model has been extensively studied [13, 36, 37, 46]. 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 towards 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 [8, 28, 45], there have been some recent attempts to study common features on these industrial problems [6] with the aim of explaining the good performance of these solvers on this benchmark. In this context and due to the heterogeneity of application SAT benchmarks, realistic pseudo-industrial random SAT instances generators have emerged, stated as one of the most important challenges in propositional search [42]. The cornerstone of these models is to produce random formulas with computational properties similar to real-world instances. The popularity–similarity (PS) random model [27] has been proposed as one of these realistic random SAT generators.

The *entropy* of a physical system measures its macroscopic energy given the configurations of its microscopic particles. The zero entropy state occurs when all particles are in the configuration with the highest probability. On the contrary, the entropy of the system grows as the likelihood of the configuration decreases, which is commonly achieved by increasing its *temperature*. Inspired by this, the PS model defines an expected structure composed of scale-free structure [2] (the number of variables occurrences follows a power-law distribution, i.e., a few variables occurs a lot while most of them occur very little) and community structure [4] 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 observed in most real-world SAT benchmarks [2, 4]. In order to control the entropy of the resulting formula, the PS model defines a parameter, called *temperature* T, to control the degree of distortion into this structure. This is, at $T = 0$ the model produces a formula with clear scale-free and community structures 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 exploit both the scale-free and the community structure of industrial SAT formulas. In particular, they focus on frequent variables and on variables of the same community [3, 5, 7–9, 26]. 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 [27]. Based on that, 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, in this work we assume a real-world SAT problem as an instantiation of the PS model [27], 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). However, most of real-world SAT instances exhibit the structure of the PS model and, therefore, we consider our assumption plausible for most of application formulas.

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 [40]. 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 with PS formulas generated at distinct temperatures, and measure their accuracy in the estimation. As we will see, training the model requires a careful design of the formula generation in, e.g., their temperature values. 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. Specifically, we analyze changes in the scale-free structure, the temperature range and the set of features used to train the models. We obtain interesting results about model robustness, as well as the relations between different regression techniques and the feature sets. Empirically, we show that ML techniques based on ensembles (e.g., random forest) are the most accurate approaches, and they remain robust to perturbations.

Our second contribution uses the previous *estimators* to analyze the hardness of real-world SAT instances. In particular, we analyze the hardness of the application benchmarks

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

from the SAT Competition 2017 to 2021. Interestingly we observe that formulas with high (estimated) temperature seem to be harder than those with low temperature, measured as the percentage of SAT solvers submitted to the competition that were able to solve such a formula. Therefore, we consider the present work is a first step towards a simple hardness characterization of real-world SAT instances based on the notion of temperature, which may be useful to estimate their solving cost without solving them.

The rest of this work is organized as follows. Section 2 describes some preliminaries on the PS model, whereas in Sect. 3, we summarize the main related works. Section 4 provides a brief description of the ML-based regression techniques used in our analysis. Section 5 is devoted to the analysis of the temperature estimation, whereas Sect. 6 analyzes the temperature of real-world SAT benchmarks. Finally, we conclude in Sect. 7.

2 Preliminaries

In this section, we provide some preliminaries on the PS random SAT model [27]. 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 randomly assigned radial and angular coordinates $r_i \in [0, 1]$ and $\theta_i \in [0, 2\pi]$, 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

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

where r_i and r_j represent the radii 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. Notice that the values of r_i, r_j, and θ_{ij} affect the probability $P(i \leftrightarrow j)$. Let us assume that β is large enough.[2] When T is small enough, the smaller the values r_i, r_j, and θ_{ij}, the bigger their product and hence the probability $P(i \leftrightarrow j)$. As (one of) these values increase, this probability, which depends on the radii r_i and r_j and their angle difference θ_{ij}, decreases. On the other hand, when T is large, the value $(r_i r_j \theta_{ij})^{1/T}$ tends to 1, and hence, the probability $P(i \leftrightarrow j)$ does not depend on r_i, r_j, θ_{ij}. 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.

In order to illustrate the effects of the temperature on the generated PS formulas, in Fig. 1 we depict two PS formulas with $n = 100$ variables, $m = 425$ clauses, and $\beta = 0.8$, only differing in their temperatures. In particular, one formula is generated with $T = 0.1$ (low temperature), while the other has $T = 10$ (high temperature). It can be observed that, when the temperature is low, variables tend to connect to *close* clauses (i.e., clauses with small radius or clauses with a similar angle), whereas a high temperature may produce occurrences of any variable in any clause, as in the classical random model. Moreover, the temperature has a dramatic effect in the scale-free structure. In Fig. 2, we represent the distribution of variables occurrences of these formulas with low and high temperature. When the temperature is low, this distribution clearly fits a power-law distribution. In fact, the estimated value of β is 0.89, very close to the original value 0.8. In contrast, when the temperature is high, the data do not exhibit the heavy-tail behavior of power-law distributions. In particular the tail of these data decreases exponentially, hence this formula dos not show scale-free structure. See [2] for more details in the estimation of β.

In summary, this model has the following parameters. The number of variables and clauses is, respectively, n and m. The (minimum) clause length is K. The scale-free structure of variables occurrences is set by β, which is the exponent of the corresponding power-law distribution. In addition, clause lengths may follow another power-law distribution with exponent β', where the average clause length is $(K + k)$. Finally, the temperature is T.

3 Related Work

There are in the literature other realistic SAT generators, such as the scale-free SAT model [3], which generates purely scale-free SAT instances, and the community attachment model [25], which is able to produce formulas with clear community structure. We recall that both features can be observed in PS formulas.

[2] We do not consider cases where β is small, since they are not realistic.

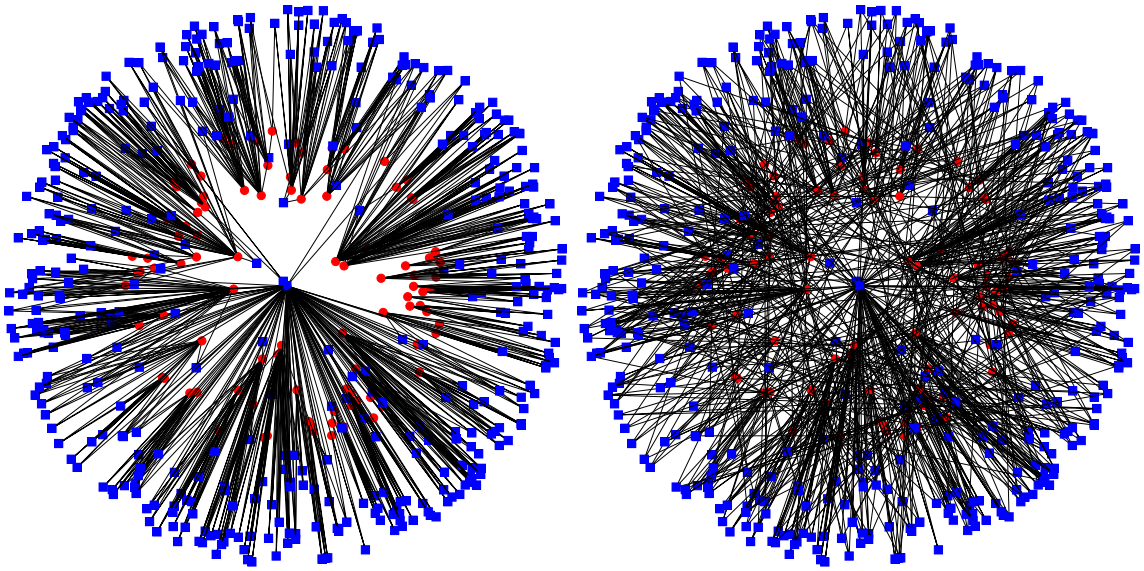

Fig. 1 Graphical representation of PS formulas with low $T = 0.1$ (left) and high $T = 10$ (right), generated with $n = 100$, $m = 425$, and $\beta = 0.8$. Blue and red nodes represent variables and clauses, respectively. The (x, y) coordinates of each node represent their radius and their angle, respectively

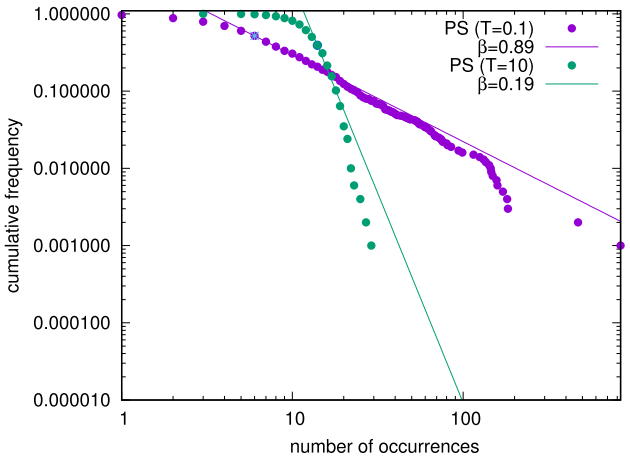

Fig. 2 Scale-free structure of PS formulas with low $T = 0.1$ and high $T = 10$, generated with $n = 1000$, $m = 4250$, and $\beta = 0.8$, and their estimated values of β

Our work is based on the conjecture that the hardness of real-world SAT instances is based on the entropy and other simple formula features. This is the case in (purely) scale-free SAT formulas [3], for which the hardness, besides k, n, m, depends on the exponent β of the power-law distribution that characterizes their scale-free structure [1, 15, 20–23, 39]. We recall that the PS formulas used in our experiments also exhibit this scale-free structure.

A seminal contribution on ML applied to SAT solving is SATzilla [31, 47]. 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 [34]. 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 [32, 35].

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 $\mathbf{x}_\phi = [x^1_\phi, \ldots, x^n_\phi]$ of n features. Being $x^* \notin \mathbf{x}$ the target feature to estimate, the problem consists of finding the function f s.t. $f(\mathbf{x}) = x^* \pm \epsilon$ 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 *learn* 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* [29] 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: linear regression, ensemble methods, and other techniques (including neural networks). For all

the techniques except neuronal networks, we use the implementations available at *scikit-learn* [41] with their default hyperparameter settings. In the case of neural networks, we use the implementation of *Keras* [12]. In addition, we evaluate all these models using the best hyperparameter settings found after a Bayesian optimization. In what follows, we briefly describe the ML algorithms used in our experiments, and the description of the Bayesian optimization performed.

4.1 Linear Regression Methods

Linear regression is an ordinary least squares linear regression using singular value decomposition. Stochastic gradient descent is a linear model minimizing a regularized empirical loss with stochastic gradient descent. The regularization is a penalty added to the loss function that shrinks the model parameters towards the zero vector. Passive aggressive [16] is a margin-based linear algorithm with no learning rate and a regularization parameter (maximum step size). Random sample consensus (RANSAC) [17] calculates linear solutions minimizing least squares on subsets of fixed size from the training samples, selecting the solution that best fits to the subset of the data. Theil–Sen [17] also calculates linear solutions minimizing the sum of squared residuals on subsets of fixed size from the training samples. In this case, the L1 median is obtained for all the computed solutions. Huber [30] is a linear model that optimizes the squared loss for the samples where $|(y - y'w)\sigma| < \epsilon$ and the absolute loss in other case, where y and y' represent the real and predicted target value, respectively, and w and σ are parameters to be optimized in order to be less sensitive to outliers. Bayesian ridge [44] implements a Bayesian linear regression, an approach to linear regression in which a particular form of prior distribution is assumed (a normal distribution) for the model parameters.

4.2 Ensemble Methods

Random Forest [10] fits a set of decision trees on various subsamples of the dataset. To make a prediction, it uses the average of all decision trees predictions. Extra Trees [24] fits a set of randomized decision trees on various subsamples of the dataset and also uses the average of all decision trees predictions. When looking for the best split in a randomized decision tree, a subset of random splits are drawn and the best split among those is chosen. AdaBoost [19] fits a sequence of estimators (regression trees) on the same dataset weighting instances according to the error of predictions, such that subsequent trees focus more on difficult cases. XGBoost [11] is an implementation of gradient boosting, a method that fits a sequences of estimators (regression trees) based on the negative gradient of a loss function.

4.3 Other Methods

Feed-forward neural networks (FNN) with one hidden layer, using its default configuration, where the number of hidden neurons is set to 64, the batch size to 32, and *Adam* [33] as the optimizer. The hyperbolic tangent is the activation function. In Kneighbors, the target is predicted by local interpolation of the targets associated to the nearest neighbors in the training set.

4.4 Bayesian Optimization

Bayesian optimization [38] is a sequential design strategy for global optimization, which does not require derivatives. This strategy treats the objective function as a random function and places a prior over it. In our case, the objective function is the determination coefficient over a validation set, and the prior is a Gaussian process. In order to evaluate each algorithm, we perform a tenfold cross-validation, i.e., the learning procedure is performed a total of ten times, each with 90% of the instances as training set and the remaining 10% used for validation.

For linear regression methods, we will optimize the specific hyperparameters of each model. In the case of RANSAC and Theil–Sen, only the size of the training samples subsets will be tuned. In the case of techniques based on ensembles, the number of estimators and various constraints on their structure will be considered. In the specific case of AdaBoost and XGBoost, also the learning rate and the loss function will be tuned. For neural networks, the hyperparameters considered are the number of hidden neurons, the optimization strategy and the *batch size*. Finally, for Kneighbors, it is only optimized the number of neighbors considered to make a prediction.

5 Analysis of the Temperature Estimation

In this section, we present an exhaustive experimental evaluation of our method to estimate the entropy of SAT instances, with the aim of showing the robustness of our approach. First, we introduce our experimental setup, describing the generation of instances, and the evaluation process. Then, we present the results on the estimation of the temperature of SAT instances, calculated by a number of state-of-the-art regression methods, using both default and optimized hyperparameter settings.

Next, we analyze the performance of our method exposing the generation of SAT formulas to several perturbations in the training stage. Namely, we increase the parameter values used in the generation, and study how this affects the accuracy of the regression models. Finally, we examine the impact of the set of SAT features used in the training phase

on the accuracy of the models. In the following subsections, we, respectively, present these investigations.

5.1 Experimental Setup

5.1.1 Generation of SAT Formulas

The training set is composed of a 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.

In order to adequately train the regression models, we need to generate a heterogeneous set of PS random SAT formulas. Since the temperature is the parameter to estimate in our approach, it is the most critical choice in the generation of the benchmark. However, it is also important to consider an ample range of formulas differing in the remaining parameters in order to make the regression methods *learn* the behavior of the PS model at different settings. Before discussing the generation of the benchmark at distinct temperatures, we first describe the general batch of PS formulas used in our experiments.

The main difference among the formulas in the benchmark, besides their temperature, is the formula size. In particular, we generate random PS formulas with the following number of variables n and clause/variable ratios m/n:

- $n = \{100, 200, 500, 1000, 2000, 5000\}$
- $m/n = \{2.0, 3.0, 3.5, 4.0, 4.1, 4.2, 4.3, 4.4, 4.5, 5.0, 6.0, 8.0\}$

Notice that we generate both satisfiable and unsatisfiable instances, with a greater density of formulas in the SAT-UNSAT phase transition region. In addition, note that these formulas are much smaller than actual real-world SAT instances existing in the SAT Competitions. However, these sizes are enough to generalize the structure of the resulting formulas and, hence, to test our hypothesis.

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 Sect. 5.3.1). In particular, we train the regression techniques with the same set of formulas generated with $\beta = \{5/6, 4/6, 3/6\}$, as well as the union of these four benchmarks.

As in [26], we restrict our analysis to 3-CNF SAT formulas, hence we generate the instances with $K = 3$ and $k = 0$. Since there is no variability in the clause lengths, there is no need to define the parameter β'.

5.1.2 Values of Temperature

The values of the temperature of the generated formulas are a key choice of our solution. It is desirable to generate formulas in an ample interval of temperatures, and these values must be uniformly distributed in such an interval in order to adequately train the regression model. However, a small difference of T at low temperatures may result in major differences in the resulting structure of the generated formula, whereas the same small difference at high temperatures results in a formula whose structure is almost unaltered. For instance, the PS model behaves quite distinctly with temperatures $T = 0.1$ and $T = 0.6$, whereas there is no remarkable difference between the model at temperatures $T = 10$ and $T = 10.5$. Notice that this is a direct consequence of Eq. 1. This distinct behavior makes inadequate the uniform selection of random values in a certain interval. In order to solve this drawback, we apply a logarithmic transformation to the range of temperature values. This is, instead of using the interval $[a, b]$, we use the interval $[\log(a), \log(b)]$.

To generate our batch of PS formulas, we sample 100 uniformly distributed random values in the interval $[-1, 1]$ for each formula size, i.e., for each combination of n and m/n. These random values correspond to the logarithm of the temperature of the generated formulas, and they are the values used to train the regression models. In other words, instead of estimating the temperature T of a given formula, the regression model estimates its logarithm $\log(T)$. Therefore, the temperature ranges in the interval $[1/e, e]$. Although this interval contains a reasonable range of values of temperature, we also evaluate more ample intervals in Sect. 5.3.2.

It is important to mention that the regression models are trained with no PS formulas at temperature $T = 0$. There is a twofold explanation to this choice. First, the PS model at very low temperatures behaves similarly than at the absolute zero limit. Second, since we consider that real-world SAT instances always have a certain degree of entropy, there is no need to train our model with *unrealistic* instances at $T = 0$.

5.1.3 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 [47]. In particular, we use a total of 101 features,[3] including formula size, graph characteristics, and solver statistics. In Sect. 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.

5.1.4 Filtering Out Trivial Instances

Random PS SAT instances at low temperatures may be very easy [27]. 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, which cannot be computed if the formula is already solved. Moreover, we observed that the resulting unbalance after filtering out these trivial instances does not affect the performance of the regression models with the best accuracy, due to the already large number of formulas in the benchmark.

5.1.5 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. Let X and Y be, respectively, a sample of observed data and their predicted values. The coefficient of determination R^2 between them is defined as

$$R^2(X, Y) = 1 - \frac{\sum_i (x_i - y_i)^2}{\sum_i (x_i - \bar{X})^2},$$

where $\bar{X}$ is the mean of X. Therefore, $R^2 \in [-\infty, 1]$ 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 is the prediction).

For each regression technique, a tenfold cross-validation is performed. For each fold, we compute the R^2 of the model trained with the remaining ninefold. The global performance of each method is expressed as the average R^2 and its corresponding standard deviation. This procedure allows us to reduce the variance in the results and obtain a confidence criteria.

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 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).

[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.

Table 1 Average coefficient of determination R^2 (and its standard deviation) for different regression methods, with default and optimized hyperparameters

Regression method	Default	Optimized
Linear regression	0.789 ± 0.35	0.789 ± 0.35
Stochastic gradient descent	-1.230 ± 2.24	$-2.5e8 \pm 3e8$
Passive aggressive	-181.7 ± 271	-1.072 ± 0.13
RANSAC	-0.982 ± 1.53	0.511 ± 1.20
Theil–Sen	-8.272 ± 25.3	$\mathbf{0.898 \pm 0.03}$
Huber	-0.085 ± 0.14	-0.076 ± 0.15
Bayesian ridge	0.760 ± 0.46	0.694 ± 0.66
Random forest	$\mathbf{0.921 \pm 0.01}$	$\mathbf{0.931 \pm 0.01}$
Extra trees	$\mathbf{0.922 \pm 0.01}$	$\mathbf{0.935 \pm 0.01}$
AdaBoost	$\mathbf{0.878 \pm 0.02}$	$\mathbf{0.896 \pm 0.01}$
XGBoost	$\mathbf{0.924 \pm 0.01}$	$\mathbf{0.935 \pm 0.00}$
FNN	0.687 ± 0.07	$\mathbf{0.912 \pm 0.01}$
Kneighbors	0.795 ± 0.03	0.796 ± 0.02

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. To answer this question, we first perform the regression using the methods presented in the previous section.

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. We recall that this coefficient measures the differences between the actual temperatures and the predicted ones; its values range between $-\infty$ and 1, with (greater) positive values indicating (stronger) correlation between both samples.

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 our approach. Those accurate methods are based on ensembles, which are commonly more robust to hyperparameter tuning. This can be due to the low sensitivity of this kind of methods to modifications of their hyperparameter values. In addition, FNN and Kneighbors present an acceptable accuracy with a low standard deviation. Although models linear regression and Bayesian ridge

also obtain an acceptable R^2 average, we cannot draw definitive conclusions due to their high deviation. For the Bayesian optimization to tune the hyperparameters settings, we assume the objective function to be unknown (determination coefficient over the validation set), treating it as a random function and placing a prior over it (a Gaussian process). The prior captures beliefs about the behavior of the function and after each evaluation, the prior is updated to form the posterior distribution over the objective function. The posterior distribution, in turn, is used to construct an acquisition function that determines the next evaluation point. In order to find the best parameter settings of each regression method, we evaluate the objective function using 100 evaluation points.

After optimizing hyperparemeters, we observe noticeable improvements in most of the methods, especially in FNN and Theil–Sen. 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 Theil–Sen, 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.

Besides the techniques that already show a good performance with default parameters, we find a very good accuracy (with R^2 above 0.8) in the following regression models: Theil–Sen, RANSAC, and Bayesian ridge. However, the improvements in the accuracy of these techniques suggest that they are not robust to variations in the benchmark, since the good performance is only achieved after an optimization process of their parameters settings.

In Fig. 3, we depict the predicted temperature versus the actual temperature of random PS formulas for the six regression techniques with a good performance after optimizing their hyperparameters. Notice that when R^2 is close to 1, the points on the figure must be close to the diagonal. For the methods based on ensembles of decision trees (Random Forest, AdaBoost and XGBoost) and due to their low sensitivity to hyperparameter tuning, there is no remarkable difference between their performance with default and optimized settings. In fact, their values of R^2 are close to 1 in both settings, suggesting that they are robust to estimate the temperature of SAT formulas. Interestingly, for Theil–Sen and FNN, we observe considerable improvements, resulting in a good performance after optimizing their settings. Recall that Theil–Sen performs a linear regression. In the case of FNN, it can be clearly observed its poor performance with its default setting, caused by its default learning rate. For Kneighbors, it can be observed that its performance is worse than the one of the other 5 models, in both settings, due to the number of points far from the diagonal.

5.3 Robustness to Benchmark Perturbations

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. The following subsections describe these experiments.

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 hyperparameters computed for $\beta = 1$. As before, we measure the accuracy of the models using the determination coefficient R^2, but we restrict our analysis to the regression methods that already showed a good performance in the previous experiment, and adding linear regression as baseline. In Table 2, we summarize the results of this experiment.

We observe that, in all the 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: Random Forest, Extra Trees 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 linear regression method is able to obtain a reasonably 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 Theil–Sen, 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

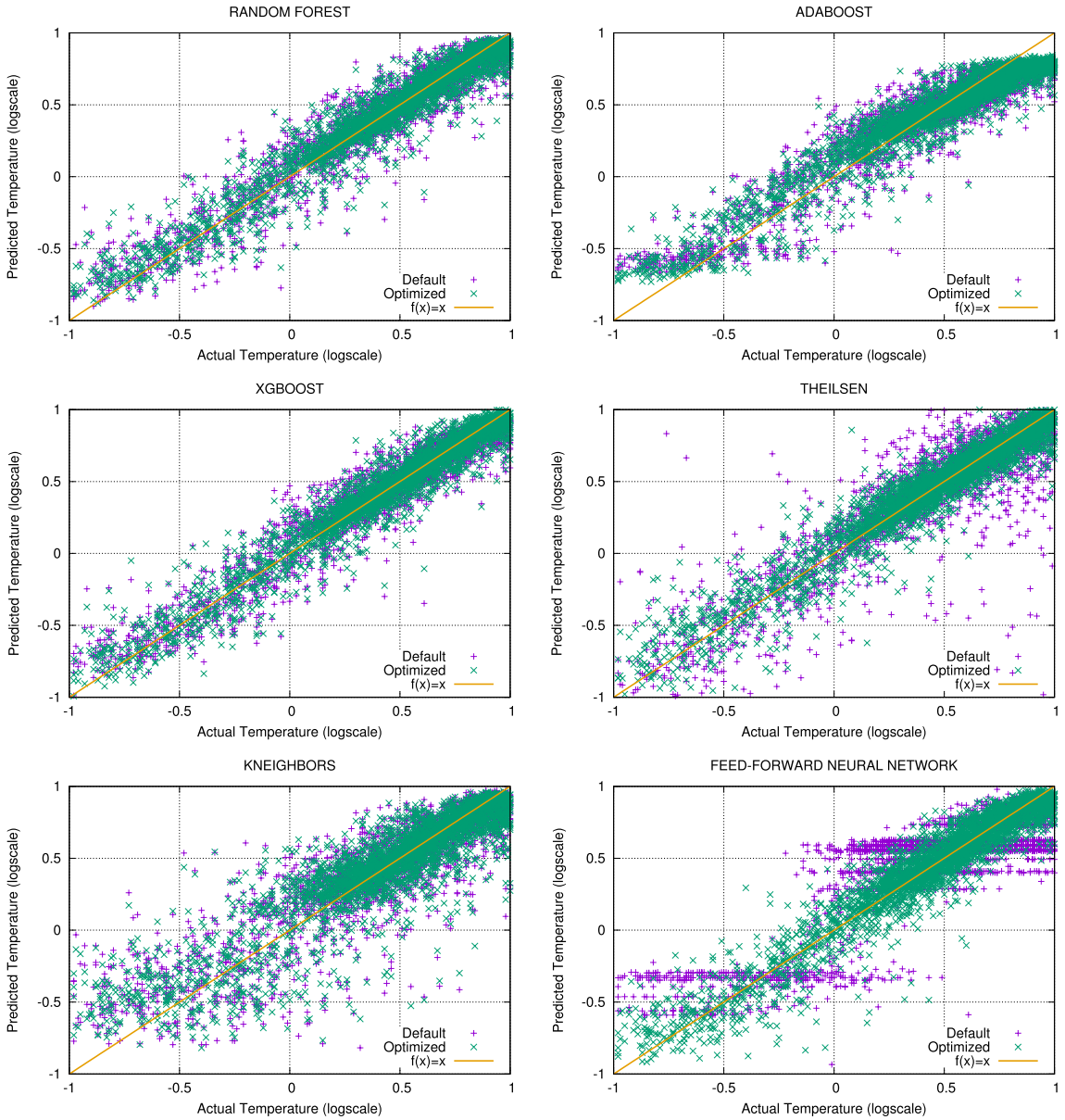

Fig. 3 Predicted temperature versus actual temperature, for some regression techniques with a small standard deviation

Table 2 Average of coefficient of determination R^2 (and its standard deviation) for different regression methods (with optimized hyperparameter values), varying the value of β in the benchmark

Regressor	$\beta = 1$ (baseline)	$\beta = 5/6$	$\beta = 4/6$	$\beta = 3/6$	Union
Linear regression	0.789 ± 0.35	-3.199 ± 12.4	-0.568 ± 4.45	$\mathbf{0.856 \pm 0.13}$	0.785 ± 0.02
Theil–Sen	$\mathbf{0.898 \pm 0.03}$	-2.223 ± 9.40	-6.625 ± 22.3	-197.1 ± 241	-213.6 ± 270
Random Forest	$\mathbf{0.931 \pm 0.01}$	$\mathbf{0.932 \pm 0.01}$	$\mathbf{0.942 \pm 0.01}$	$\mathbf{0.958 \pm 0.00}$	$\mathbf{0.858 \pm 0.01}$
Extra Trees	$\mathbf{0.935 \pm 0.01}$	$\mathbf{0.935 \pm 0.01}$	$\mathbf{0.946 \pm 0.01}$	$\mathbf{0.960 \pm 0.01}$	$\mathbf{0.861 \pm 0.01}$
AdaBoost	$\mathbf{0.896 \pm 0.01}$	$\mathbf{0.882 \pm 0.01}$	$\mathbf{0.883 \pm 0.01}$	$\mathbf{0.904 \pm 0.01}$	0.678 ± 0.01
XGBoost	$\mathbf{0.935 \pm 0.00}$	$\mathbf{0.937 \pm 0.01}$	$\mathbf{0.948 \pm 0.01}$	$\mathbf{0.963 \pm 0.01}$	$\mathbf{0.872 \pm 0.01}$
FNN	$\mathbf{0.912 \pm 0.01}$	$\mathbf{0.904 \pm 0.02}$	$\mathbf{0.915 \pm 0.01}$	$\mathbf{0.927 \pm 0.01}$	0.780 ± 0.02

Union stands for the benchmark composed of the union of the other four

Table 3 Average of coefficient of determination R^2 (and its standard deviation) for different regression methods (with optimized hyperparameter values), varying the interval of the temperature used to generate the benchmark

Regressor	$[-1, 1]$ (baseline)	$[-2, 2]$	$[-3, 3]$
Linear regression	0.789 ± 0.35	$\mathbf{0.936 \pm 0.02}$	$\mathbf{0.907 \pm 0.01}$
Theil–Sen	$\mathbf{0.898 \pm 0.03}$	$\mathbf{0.939 \pm 0.01}$	$\mathbf{0.904 \pm 0.02}$
Random Forest	$\mathbf{0.931 \pm 0.01}$	$\mathbf{0.955 \pm 0.01}$	$\mathbf{0.945 \pm 0.01}$
Extra Trees	$\mathbf{0.935 \pm 0.01}$	$\mathbf{0.956 \pm 0.01}$	$\mathbf{0.947 \pm 0.01}$
AdaBoost	$\mathbf{0.896 \pm 0.01}$	$\mathbf{0.928 \pm 0.00}$	$\mathbf{0.904 \pm 0.01}$
XGBoost	$\mathbf{0.935 \pm 0.00}$	$\mathbf{0.956 \pm 0.01}$	$\mathbf{0.943 \pm 0.01}$
FNN	$\mathbf{0.912 \pm 0.01}$	$\mathbf{0.917 \pm 0.01}$	$\mathbf{0.874 \pm 0.02}$

the temperature of the random PS formulas in the benchmark. We recall that 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 $[-3, 3]$; the rest of the generation remains unaltered.

In Table 3, we also summarize the results of this perturbation. We recall that we are evaluating regression methods whose hyperparameters were optimized for the benchmark with temperatures in the interval $[-1, 1]$.

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

Related to the set of features used in the experiments, we can consider the following questions. What is the relationship between the set of features used in the training step and the predictive capacity of the different regression methods and their configurations? Can this relationship give some information about which sets of features have a linear relationship with temperature and which do not? Is the features set used in our experiments introducing some bias in the results? In this subsection, we carry out some experiments to address these questions.

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. Basic features concern problem size information (e.g., number of variables and clauses), proximity to the Horn formula, and the so-called balance features (e.g., ratio between positive and negative literals). Graph features rely on three graph representations of the formula, namely variable graph, clause graph, and variable-clause graph, and the features represent some statistics about these graphs (e.g., node degree). CDCL features contain information in very short runs (i.e., it does not solve the formula, it just runs the solver during a few seconds) about number of unit propagations, estimation of the search space, and number and size of learned clauses. The remaining subset contains the rest of features described in the documentation of SATzilla, including linear programming, local search, and survey propagation 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 [47].

In Table 4, 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 their 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 (linear regression and Theil–Sen) have very good performance when they are trained using graph features only, while 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. Notice that linear estimators are less sensitive to overfitting than others.

Table 4 Average of coefficient of determination R^2 for different regression methods (with optimized hyperparameter), for different features sets

Regressor	All (baseline)	Basic (A)	Graph (B)	CDCL (C)	Union $A \cup B \cup C$
Linear regression	0.789 ± 0.35	0.118 ± 2.14	$\mathbf{0.808 \pm 0.08}$	0.330 ± 0.07	0.749 ± 0.41
Theil–Sen	$\mathbf{0.898 \pm 0.03}$	-0.058 ± 2.62	$\mathbf{0.832 \pm 0.03}$	0.336 ± 0.06	-0.143 ± 3.11
Random Forest	$\mathbf{0.931 \pm 0.01}$	$\mathbf{0.911 \pm 0.01}$	$\mathbf{0.917 \pm 0.01}$	0.579 ± 0.04	$\mathbf{0.927 \pm 0.01}$
Extra Trees	$\mathbf{0.935 \pm 0.01}$	$\mathbf{0.918 \pm 0.01}$	$\mathbf{0.924 \pm 0.01}$	0.565 ± 0.05	$\mathbf{0.933 \pm 0.01}$
AdaBoost	$\mathbf{0.896 \pm 0.01}$	$\mathbf{0.863 \pm 0.02}$	$\mathbf{0.864 \pm 0.01}$	0.331 ± 0.06	$\mathbf{0.876 \pm 0.02}$
XGBoost	$\mathbf{0.935 \pm 0.00}$	$\mathbf{0.916 \pm 0.01}$	$\mathbf{0.923 \pm 0.02}$	0.554 ± 0.02	$\mathbf{0.932 \pm 0.01}$
FNN	$\mathbf{0.912 \pm 0.01}$	$\mathbf{0.889 \pm 0.01}$	$\mathbf{0.932 \pm 0.02}$	0.535 ± 0.06	$\mathbf{0.911 \pm 0.01}$

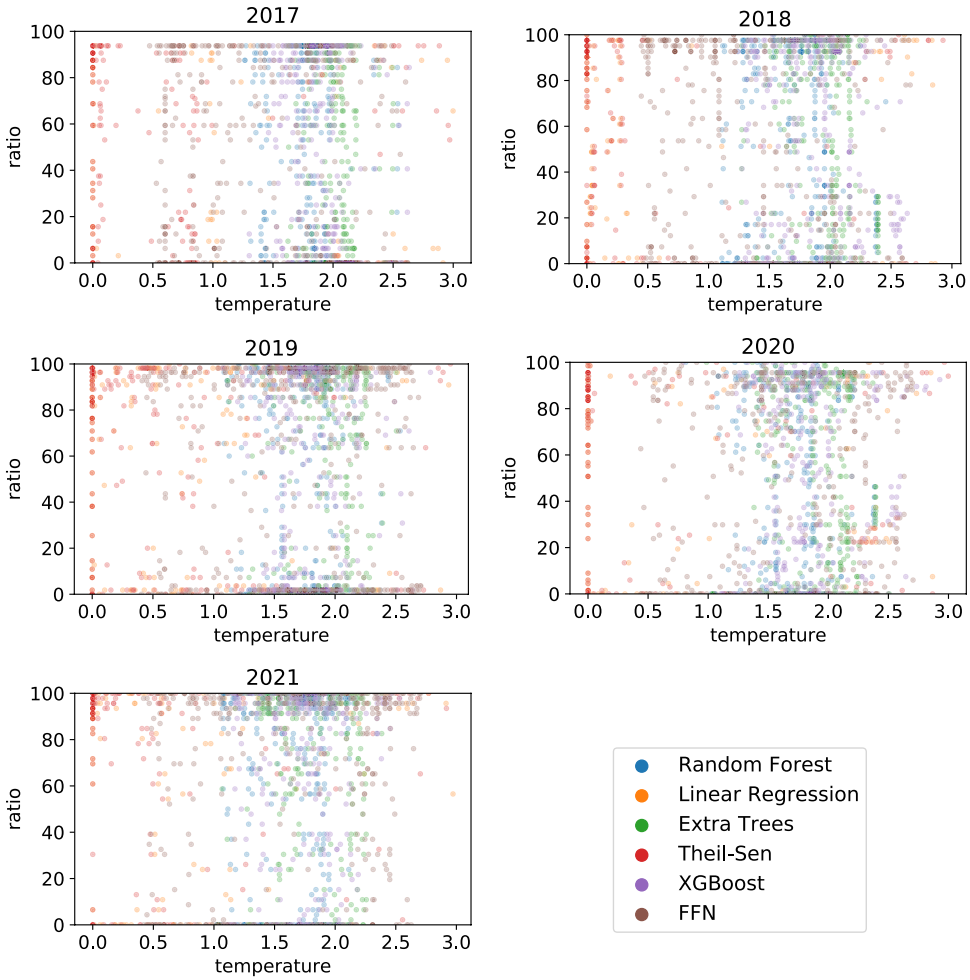

Fig. 4 Estimated temperature versus ratio of solvers submitted to the SAT Competitions from 2017 to 2021 solving the instances in those competitions, for several regression techniques trained baseline dataset with optimized hyperparameters

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 On the Hardness of Real-World SAT Instances

In this section, we analyze the relation between the hardness of real-world SAT instances and their (estimated) temperature. To estimate such a temperature, we simply use the regression methods trained with PS formulas which were analyzed in the previous section.

If the temperature of real-world SAT instances is related to their hardness, we may observe that the ones with high (estimated) temperature behave similarly to random SAT formulas, i.e., they are hard to solve. In order to analyze it, we carry out the following experiment. We select a set of real-world SAT instances and estimate their temperature. In particular, we use the set of benchmarks from the SAT Competitions from 2017 to 2021.[4] In addition, for each formula, we measure the percentage of solvers submitted to each competition that were able to solve it. This percentage represents an indirect indicator of hardness: easy formulas are solved by (almost) all solvers, whereas hard formulas are solved by (almost) no solver. Finally, we compare these two metrics.

In Fig. 4, we represent the results of this experiment, where the estimated temperatures are computed with the six regression methods that showed a good performance in the previous experiments, i.e., Random Forest, Extra Trees, Theil–Sen, XGBoost, FNN, and linear regression as baseline.

[4] http://www.satcompetition.org/.

Table 5 Percentage of correctly classified real-world SAT benchmarks of the SAT Competitions from 2017 to 2021 according to their estimated temperature for different regression methods trained

	LR (%)	RF (%)	ET (%)	TS (%)	XGB (%)	FNN (%)
SAT Comp. 2017	**64.43**	50.58	49.42	57.20	49.71	54.07
SAT Comp. 2018	**59.09**	43.81	45.10	57.67	44.33	63.4
SAT Race. 2019	**63.98**	48.97	44.87	59.75	45.64	47.18
SAT Comp. 2020	59.21	51.18	49.12	**59.17**	49.12	50.29
SAT Comp. 2021	**65.27**	40.26	38.95	65.13	38.42	38.68

The regressor with the best accuracy for each competition is marked in bold. LR stands for linear regression, RF for Random Forest, LR for Extra Trees, TS for Theil–Sen, and XGB for XGBoost

In the five competitions, the empirical results show that (i) many formulas with high temperature are only solved by a small fraction of solvers (i.e., they seem to be hard), and (ii) many formulas with low temperatures are solved by a high percentage of solvers (in particular, a large fraction is solved by 100% of the solvers, hence they are easy). However, there is a set of instances that does not exactly follow this pattern. This can be explained by several reasons. First, we conjecture that the hardness of real-world SAT formulas depends on the temperature, but there may be other parameters affecting such a hardness, e.g., formula size.[5] Second, there may exist a very hard combinatorial subproblem embedded into the formula, as in cryptography problems, representing a very challenging case for our estimators. Nevertheless, these formulas represent a small fraction of the competition.

In order to compare the accuracy of the analyzed classifiers, we measure the percentage of correctly classified instances for these regression techniques. In particular, we consider that an instance is correctly classified either if it is easy with low (estimated) temperature (top left area of the plots), or if it is hard with high (estimated) temperature (bottom right area of the plots). In our analysis, an instance is easy (resp. hard) if it is solved by at least (resp. at most) 50% of the solvers, and a temperature is low (resp. high) if it is greater (resp. less or equal) than $T = 1.5$.

In Table 5, we report these results. As it can be observed, linear regression and Theil–Sen are the ones with the best accuracy, with a performance around 65% in some cases. This confirms that, in many cases, the (estimated) temperature of the formula is related to their hardness. However, there is still a large number of formulas incorrectly classified. In general, they are easy formulas with high (estimated) temperature. We conjecture that they are formulas without scale-free and community structures and, thus, the PS model is unable to instantiate them. Interestingly enough, these methods based on linear regression outperform those based on ensembles. It is worth noticing that the simple linear regression shows the best performance in two of the five competitions analyzed. This supports the hypothesis that the characterization of the temperature of SAT instances may be *linearized* with certain features of the formulas.

7 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. Nevertheless, simple methods based on linear regression also show a very good performance in many cases. In addition, we have showed that a successful application like SATzilla is able to indirectly infer the temperature of the formulas using only a very small subset of (graph) features. Using these estimators, we observe that real-world SAT instances with a high (estimated) temperature seem to be harder than those with low temperature. These empirical evidences suggest that the hardness of real-world SAT instances is related to their temperature. We consider that the present work is a first step towards a simple hardness characterization of real-world SAT instances based on their temperature, which may be used to estimate their solving cost without solving them. As future work, we plan to extend this analysis with more sophisticated neural networks, including graph neural networks, and other automated ML techniques [18]. Moreover, we plan to extend our analysis to characterize the temperature of MaxSAT formulas, the optimization version of the SAT problem. This characterization may be crucial in order to select the most adequate algorithm to solve them (e.g., either MaxSAT solvers more specialized in real-world instances, or MaxSAT solvers more specialized in random formulas).

[5] A formula with hundreds of variables may be solvable in milliseconds, whereas solving a formula with millions of variables may require several hours or even days.

Author Contributions All the authors have equally contributed to this work.

Funding Jesús Giráldez-Cru is supported through the Juan de la Cierva program, fellowship IJC2019-040489-I, funded by MCIN and AEI.

Data Availability Statement All the authors give their consent for the publication of this work.

Declarations

Conflict of Interest The authors declare that they have on conflict of interest.

Consent for Publication Not applicable.

Ethics Approval and Consent to Participate Not applicable.

References

1. Ansótegui, C., Bonet, M., Levy, J.: Scale-free random SAT instances. CoRR, abs/1708.06805, (2017)
2. Ansótegui, C., Bonet, M., Levy, J.: On the structure of industrial SAT instances. Proc. CP **2009**, 127–141 (2009)
3. Ansótegui, C., Bonet, M., Levy, J.: Towards industrial-like random SAT instances. Proc. IJCAI **2009**, 387–392 (2009)
4. Ansótegui, C., Giráldez-Cru, J., Levy, J.: The community structure of SAT formulas. Proc. SAT **2012**, 410–423 (2012)
5. Ansótegui, C., Giráldez-Cru, J., Levy, J., Simon, L.: Using community structure to detect relevant learnt clauses. Proc. SAT **2015**, 238–254 (2015)
6. Ansótegui, C., Bonet, M., Giráldez-Cru, J., Levy, J.: Structure features for SAT instances classification. J. Appl. Log. **23**, 27–39 (2017)
7. Ansótegui, C., Bonet, M., Giráldez-Cru, J., Levy, J., Simon, L.: Community structure in industrial SAT instances. J. Artif. Intell. Res. **66**, 443–472 (2019)
8. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. Proc. IJCAI **2009**, 399–404 (2009)
9. Baud-Berthier, G., Giráldez-Cru, J., Simon, L.: On the community structure of bounded model checking SAT problems. Proc. SAT **2017**, 65–82 (2017)
10. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
11. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. Proc. KDD **2016**, 785–794 (2016)
12. Chollet, F., et al., Keras. https://keras.io (2015)
13. Chvátal, V., Reed, B.: Mick gets some (the odds are on his side). Proc. FOCS **1992**, 620–627 (1992)
14. Cook, S.: The complexity of theorem-proving procedures. Proc. STOC **1971**, 151–158 (1971)
15. Cooper, C., Frieze, A., Sorkin, G.: Random 2-SAT with prescribed literal degrees. Algorithmica **48**(3), 249–265 (2007)
16. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**(Mar), 551–585 (2006)
17. Dang, X., Peng, H., Wang, X., Zhang, H.: Theil-sen estimators in a multiple linear regression model. Olemiss Educ. (2008)
18. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning—Methods, Systems, Challenges*. Springer, pp 113–134 (2019)
19. Freund, Y., Schapire, R.: A decision-theoretic generalization of online learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
20. Friedrich, T., Krohmer, A., Rothenberger, R., Sauerwald, T., Sutton, A.: Bounds on the satisfiability threshold for power law distributed random SAT. In *Proceedings of ESA 2017*, vol. **87**, pp. 37:1–37:15 (2017)
21. Friedrich, T., Rothenberger, R.: The satisfiability threshold for non-uniform random 2-SAT. In *Proceedings of ICALP 2019*, pp. 61:1–61:14 (2019)
22. Friedrich, T., Rothenberger, R.: Sharpness of the satisfiability threshold for non-uniform random k-SAT. Proc. SAT **2018**, 273–291 (2018)
23. Friedrich, T., Krohmer, A., Rothenberger, R., Sutton, A.: Phase transition for sclae-free SAT formulas. Proc. AAAI **2017**, 3893–3899 (2017)
24. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006)
25. Giráldez-Cru, J., Levy, J.: Generating SAT instances with community structure. Artif. Intell. **238**, 119–134 (2016)
26. Giráldez-Cru, J., Levy, J.: Locality in random SAT instances. Proc. IJCAI **2017**, 638–644 (2017)
27. Giráldez-Cru, J., Levy, J.: Popularity-similarity random SAT formulas. Artif. Intell. **299**, 103537 (2021)
28. Gomes, C., Selman, B.: Problem structure in the presence of perturbations. Proc. AAAI **1997**, 221–226 (1997)
29. Ho, Y., Pepyne, D.: Simple explanation of the no-free-lunch theorem and its implications. J. Optim. Theory Appl. **115**(3), 549–570 (2002)
30. Huber, P.: Robust Statistics. Springer, Berlin (2011)
31. Hutter, F., Xu, L., Hoos, H., Leyton-Brown, K.: Algorithm runtime prediction: methods and evaluation. Artif. Intell. **206**, 79–111 (2014)
32. Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K.: ISAC—instance-specific algorithm configuration. Proc. ECAI **2010**, 751–756 (2010)
33. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv (2014)
34. Leyton-Brown, K., Hoos, H., Hutter, F., Xu, L.: Understanding the empirical hardness of NP-complete problems. Commun. ACM **57**(5), 98–107 (2014)
35. Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Non-model-based algorithm portfolios for SAT. Proc. SAT **2011**, 369–370 (2011)
36. Mitchell, D., Levesque, H.: Some pitfalls for experimenters with random SAT. Artif. Intell. **81**(1–2), 111–125 (1996)
37. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of SAT problems. Proc. AAAI **1992**, 459–465 (1992)
38. Mockus, J.: Bayesian Approach to Global Optimization: Theory and Applications, vol. 37. Springer Science & Business Media, Berlin (2012)

39. Omelchenko, O., Bulatov, A.A.: Satisfiability threshold for power law random 2-SAT in configuration model. Theoret. Comput. Sci. **888**, 70–94 (2021)
40. Papadopoulos, F., Kitsak, M., Serrano, M., Boguñá, M., Krioukov, D.: Popularity versus similarity in growing networks. Nature **489**, 537–540 (2012)
41. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
42. Selman, B., Kautz, H., McAllester, D.: Ten challenges in propositional reasoning and search. Proc. IJCAI **1997**, 50–54 (1997)
43. Silva, J.M., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Volume 185 of Frontiers in Artificial Intelligence and Applications, pp. 131–153. IOS Press (2009)
44. Tipping, M.: Sparse bayesian learning and the relevance vector machine. J. Mach. Learn. Res. **1**(Jun), 211–244 (2001)
45. Williams, R., Gomes, C., Selman, B.: Backdoors to typical case complexity. Proc. IJCAI **2003**, 1173–1178 (2003)
46. Xu, L., Hoos, H., Leyton-Brown, K.: Predicting satisfiability at the phase transition. In Proceedings of AAAI 2012, (2012)
47. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Satzilla: portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. **32**, 565–606 (2008)