# Structure Features for SAT Instances Classification☆

Carlos Ansótegui[a], Maria Luisa Bonet[b], Jesús Giráldez-Cru[c], Jordi Levy[c]

[a] *Universitat de Lleida (DIEI, UdL)*
[b] *Universitat Politècnica de Catalunya (LSI, UPC)*
[c] *Artificial Intelligence Research Institute, Spanish National Research Council (IIIA-CSIC)*

## Abstract

The success of portfolio approaches in SAT solving relies on the observation that different SAT solvers may dramatically change their performance depending on the *class* of SAT instances they are trying to solve. In these approaches, a set of features of the problem is used to build a prediction model, which classifies instances into classes, and computes the fastest algorithm to solve each of them. Therefore, the set of features used to build these classifiers plays a crucial role. Traditionally, portfolio SAT solvers include features about the *structure* of the problem and its *hardness*.

Recently, there have been some attempts to better characterize the structure of industrial SAT instances. In this paper, we use some structure features of industrial SAT instances to build some classifiers of industrial SAT families of instances. Namely, they are the scale-free structure, the community structure and the self-similar structure. First, we measure the effectiveness of these classifiers by comparing them to other sets of SAT features commonly used in portfolio SAT solving approaches. Then, we evaluate the performance of this set of structure features when used in a real portfolio SAT solver. Finally, we analyze the relevance of these features on the analyzed classifiers.

*Key words:* SAT solving, Complex networks, Classifiers, Portfolio

## 1. Introduction

The Boolean Satisfiability problem (SAT) is one of the most studied problems in Computer Science. It is the first known NP-complete problem [20, 31]. However, the irruption of certain SAT solving technologies and their sophisticated implementations allows us to efficiently solve problems from many real-world domains, as planning, software and hardware verification, scheduling or cryptography, among others [13, 43, 29].

☆This work is partially supported by the CICYT research project TASSAT2 (TIN2013-48031) and the CSIC project 201450E045.

*Email addresses:* carlos@diei.udl.cat (Carlos Ansótegui), bonet@lsi.upc.edu (Maria Luisa Bonet), jgiraldez@iiia.csic.es (Jesús Giráldez-Cru), levy@iiia.csic.es (Jordi Levy)

In the last decades, there have been many works dedicated to improve the efficiency of SAT solving algorithms. One of the most promising approaches is the *portfolio* paradigm. This approach faces the Algorithm Selection Problem [42], which is the problem of choosing, using a prediction model, the best algorithm, from a predefined set, to solve a particular instance of a problem. Such prediction is usually performed using machine learning techniques. Portfolio SAT solvers proceed as follows. First, the prediction model is built in an offline process. To this purpose, a representative set of instances is selected. A vector of (predefined) features and a vector of runtimes (of a predefined set of algorithms) is computed for each of these instances. Then, instances are grouped into *classes* based on their features, and the best algorithm is calculated for each class. Finally, given an input instance, its features are computed and it is assigned to a class (using the prediction model previously built), and it is solved by the correspondent solver assigned to that class. Some examples of portfolio approaches to SAT solving are [47, 46, 28, 32, 37, 24, 38, 45].

A common classification used in SAT Competitions [26] divides the set of instances into 3 major categories: (i) *random*, (ii) *industrial* (or *application*) and (iii) *crafted* (or *hard combinatorial*). Random formulas are $k$-SAT instances randomly generated with an uniform distribution [1]. The clear model to generate them and its connections to statistical physics motivate this category. Industrial benchmarks encode problems of real-world domains, and its motivation is to analyze SAT solving technologies in real applications. Finally, crafted instances are designed to represent hard or challenging problems to SAT solvers, in order to reveal the limits of the current SAT solving techniques. Usually, industrial and crafted instances are grouped into families, according to their application domain (e.g., *cryptography* and *planning* are industrial families).

The success of portfolio algorithms is due to the observation that different SAT solving techniques perform better on different SAT instances. This has resulted into a *specialization* of SAT solvers. Some examples of this specialization are: Conflict-Driven Clause Learning (CDCL) SAT solvers are the dominant technique for solving industrial SAT instances; Look-Ahead SAT solvers are specially efficient solving random SAT problems; Stochastic Local Search (SLS) SAT solvers exhibit a very good performance on satisfiable random $k$-SAT.

In the case of industrial SAT instances, the remarkable success of CDCL SAT solvers has been reached after an extensive test-and-error process. However, understanding why these techniques exhibit this extremely good performance in this kind of instances remains open. The common wisdom in the SAT community is that CDCL SAT solvers exploit the *hidden* structure of industrial SAT instances. In the last years, there have been some attempts to characterize this structure. For instance, it has been shown that industrial SAT instances exhibit scale-free structure [8], community structure [11], and self-similar structure [6]. In [35, 36], the community structure of SAT instances has been related to the hardness of solving them by CDCL techniques. In [9], the ratio between the runtimes needed by a CDCL solver and by a random-specialized solver is related to the scale-free structure of the formula. Also, in [23], it is proved that this ratio is also correlated to the modularity of the formula. Finally, there are some works where a more precise notion of hardness is defined [10, 12].

In this paper, we show that these three notions of structure can be used to effectively classify industrial SAT families. This classification can be useful for further SAT solvers specializations. In particular, there may exist different techniques exploiting the singularities of different industrial families. We show that using these structure features can result into a classification with similar effectiveness than using other sets of SAT features commonly used in portfolio approaches, independently of the classifier used. Interestingly, for some classifiers, using structure features even improves the performance of the classifier. We evaluate the performance of a portfolio SAT solver using these structure features, observing that its performance is almost unaffected. Finally, we analyze the relevance of these structure features in the classifiers previously analyzed. This paper is a revisited version of [7].

The rest of the paper proceeds as follows. After some preliminaries presented in Section 2, we review some notion of structure in Section 3. In Section 4, we study how the structure features can be used to classify industrial SAT families. In Section 5, we analyze the performance of a portfolio SAT solver trained with this set of structure features w.r.t. other set of SAT features commonly used in portfolio approaches. In Section 6, we analyze the relevance of these structure features presented in this paper. Finally, we conclude in Section 7.

## 2. Preliminaries

SAT is the problem of determining if there exists an assignment of the Boolean variables of a propositional formula such that the formula is evaluated as `true`. A *literal* is either a variable or its negation, a *clause* is a disjunction of literals, and a formula in conjunctive normal form (*CNF*) is a conjunction of clauses.

An undirected weighted graph is a pair $(V, w)$ where $V$ is a set of vertexes and $w : V \times V \to \mathbb{R}^+$ satisfies $w(x, y) = w(y, x)$. This definition generalizes the classical notion of graph $(V, E)$, where $E \subseteq V \times V$, by taking $w(x, y) = 1$ if $(x, y) \in E$ and $w(x, y) = 0$ otherwise. The degree of a vertex $x$ is defined as $\deg(x) = \sum_{y \in V} w(x, y)$. A bi-partite graph is a tuple $(V_1, V_2, w)$ where $w : V_1 \times V_2 \to \mathbb{R}^+$.

**Definition 1 (Variable Incidence Graph (VIG)).** *Given a SAT instance $\Gamma$ over the set of variables $X$, its variable incidence graph is a graph $(X, w)$ with set of vertexes the set of Boolean variables, and weight function:*

$$w(x, y) = \sum_{\substack{c \in \Gamma \\ x, y \in c}} \frac{1}{\binom{|c|}{2}}$$

**Definition 2 (Clause-Variable Incidence Graph (CVIG)).** *Given a SAT instance $\Gamma$ over the set of variables $X$, its clause-variable incidence graph is a bipartite graph $(X, \{c \mid c \in \Gamma\}, w)$, with vertexes the set of variables and the set of clauses, and weight function:*

$$w(x, c) = \begin{cases} 1/|c| & \textit{if } x \in c \\ 0 & \textit{otherwise} \end{cases}$$

In the previous definitions, edges are weighted in order to give the same relevance to all clauses, independently of their length. Notice that, in both models, the sum of the weights of all the edges generated by one clause is equal to 1. Notice also that the sign of literals is not considered in these models, as done in [8, 11, 6].

## 3. The Structure of Industrial SAT Instances

In this section, we review some notions of structure that have been previously analyzed in industrial SAT instances. Namely, they are the scale-free structure [8], the community structure [11], and the self-similar structure [6]. For more detailed and technical reports, we address the reader to the previous references.

### *3.1. The Scale-free Structure*

In the classical *Erdös-Rényi random graph model* [21], the degree of nodes follows a binomial distribution. Therefore, the variability of this distribution is very small. In general, the variability of exponentially decreasing tail distributions, as normal, binomial or Poisson, is very small.

In contrast, the *scale-free* model is characterized by a big variability. This model was introduced in [3] to describe the structure of the World Wide Web, viewed as a graph, which cannot be described by the classical random graph model. In the scale-free model, the degree of nodes follows a power-law distribution $p(k) \sim k^{-\alpha}$, and this distribution is scale-free. *Power-law (zeta and Pareto) distributions* are characterized by a big variability, consequence of a polynomially decreasing tail. These distributions are also called *heavy-tailed*, and they are very frequent in nature. The popular rule known as *80:20 rule* explain this behavior: a small fraction of the individuals is responsible for most of the average (i.e. 80% of the land is owned by the 20% of the population).

In the case of industrial SAT instances, the number of variable occurrences has been analyzed. More precisely, we can compute the function $f_v(k)$, which is the number of variables that have a number of occurrences equal to $k$, divided by the number of variables $n$. Assuming that this function follows a power-law distribution (i.e., $f_v(k) \approx ck^{-\alpha_v}$), we can estimate the exponent $\alpha_v$ of the power-law distribution that best fits this collection of points. This estimation is computed by the method of maximum likelihood [17].

### *3.2. The Community Structure*

The *community structure* of a graph is usually measured using the notion of *modularity $Q$* [34]. Having high modularity (or clear community structure) means that nodes can be grouped into communities, such that most edges connect nodes of the same community. Defined for a graph $G$ and a partition $C$ of its vertexes into communities, the modularity $Q$ (see Eq. 1) measures the fraction of internal edges (edges connecting vertexes of the same community) w.r.t. a random graph with same number of vertexes and same degree distribution. The second term avoids that the best partition is the one made up by an single community containing all vertexes.

$$Q(G,C) = \sum_{C_i \in C} \frac{\displaystyle\sum_{x,y \in C_i} w(x,y)}{\displaystyle\sum_{x,y \in V} w(x,y)} - \left( \frac{\displaystyle\sum_{x \in C_i} deg(x))}{\displaystyle\sum_{x \in V} deg(x)} \right)^2 \tag{1}$$

The modularity of a graph is the maximal modularity for any possible partition: $Q(G) = \max\{Q(G,C) \mid C\}$.

In the case of industrial SAT instances, the community structure has been studied computing the modularity $Q$ of the VIG. As computing the modularity of a graph is NP-hard [15], instead that computing the (exact) value of the modularity, most methods in the literature approximate a lower-bound in the value of $Q$. In [11], it is used the *Louvain method* [14], one of the most accurate algorithms in large graphs.

### *3.3. The Self-Similar Structure*

A self-similar graph keeps its structure after *rescaling* it. Rescaling means replacing groups of nodes by a single node. In these graphs, the diameter grows as $d^{max} \sim n^{1/d}$, where $d$ is the fractal dimension of the graph, and not as $d^{max} \sim \log n$, which is the case in random graphs. Computing the fractal dimension of a graph is computing the number of *tiles* required to cover the graph. A *tile* of *radius* $r$ and *center* $c$ is a subset of nodes of the graph such that the distance between any of them and the node $c$ is strictly smaller that $r$. Let $N(r)$ be the minimum number of circles of radius $r$ required to cover a graph. A graph has the *self-similarity* property if the function $N(r)$ decreases polynomially, i.e. $N(r) \sim r^{-d}$, for some value $d$. In this case, we call $d$ the *fractal dimension* of the graph [33].

In the case of industrial SAT instances, the fractal dimension has been analyzed computing the function $N(r)$ for the VIG and for the CVIG. Assuming that this function decays polynomially (i.e., $N(r) \sim r^{-d}$), we can compute the degree $d$ (i.e., the fractal dimension) that best fits the function $N(r)$. This value is estimated by linear regression interpolating the points $\log N(r)$ vs. $\log r$. In our experimentation, we name $d$ and $d^b$ the values estimated for the fractal dimension of the VIG and the CVIG, respectively. See [6] for more details about the computation of $N(r)$ and the estimation of $d$.

## 4. Classifying Industrial SAT Families

Most industrial SAT instances exhibit a power-law distribution in the number of variable occurrences [8], a clear community structure with high modularity [11], and a fractal dimension that characterizes their self-similarity [6]. Therefore, for each industrial SAT instance we compute the exponent $\alpha_v$ of a power-law distribution that best fits the number of variable occurrences, the modularity $Q$ of its VIG, and the fractal dimensions $d$ and $d^b$ of its VIG and CVIG, respectively. Notice that the number of variable occurrences is exactly the degree of variable-nodes in the CVIG. In the case

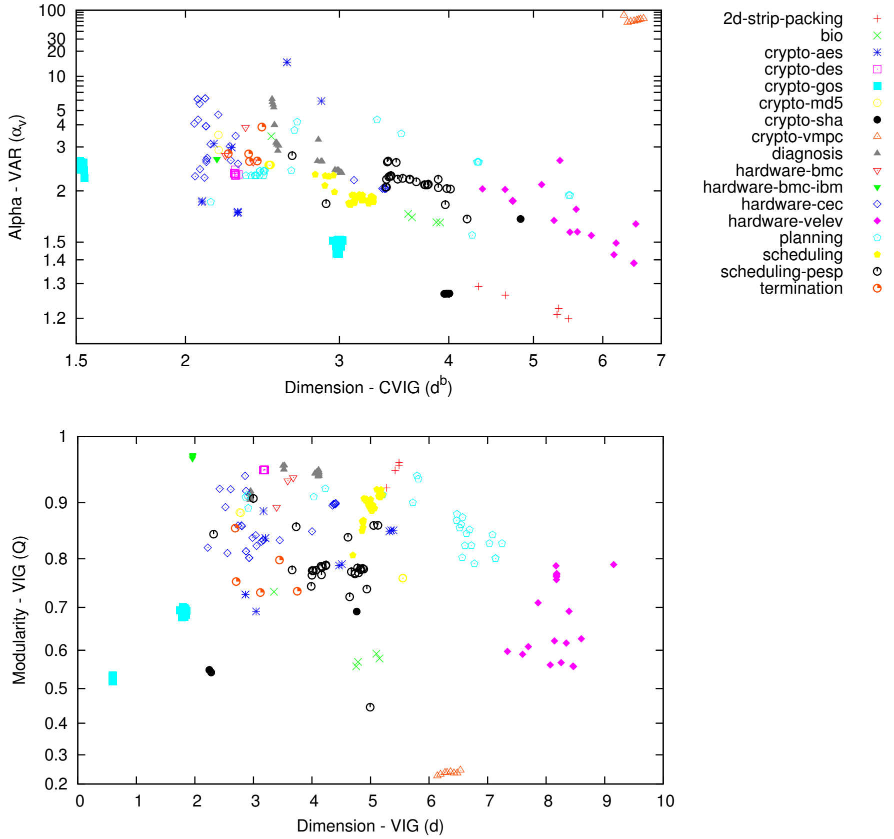

Figure 1: Distribution of families according to the exponent of the power-law distribution of variable occurrences ($\alpha_v$), the fractal dimensions of the VIG and CVIG ($d$ and $d^b$, respectively), and the modularity ($Q$). The heterogeneous families *software-bit-verif* (14 instances) and *software-bmc* (3 instances) are not plotted.

of the clause length, which corresponds to the degree of clause-nodes in the CVIG, it is not clear if the distribution that best fits these data is, in most of cases, a power-law. Also, the modularity $Q^b$ of the CVIG could be computed, but most methods in the literature are not adapted to be used in bi-partite graphs (they are either not accurate or not fast enough for these graphs).

In this paper, we use the set of 300 industrial SAT instances of the SAT Competition 2013. These instances are grouped into 19 industrial families, according to their application domain: *2d-strip-packing*, *bio*, *crypto-aes*, *crypto-des*, *crypto-gos*, *crypto-md5*, *crypto-sha*, *crytpo-vmpc*, *diagnosis*, *hardware-bmc*, *hardware-bmc-ibm*, *hardware-cec*, *hardware-velev*, *planning*, *scheduling*, *scheduling-pesp*, *software-bit-verif*, *software-bmc* and *termination*. All instances are *industrial*, in the sense that they come from a real-world problem.

In a first experiment, we analyze if the classification of industrial SAT instances into families according to their domain corresponds to a classification of families by structure features. In Figure 1, we represent the relation between the scale-free structure ($\alpha_v$), the community structure ($Q$) and the self-similar structure ($d$ and $d^b$) for each industrial family. Each industrial SAT instance is represented by a different point, and each industrial family is characterized by a different symbol. In order to facilitate the visualization of this plot, we have omitted the industrial families *software-bit-verif* (14 instances) and *software-bmc* (3 instances), due to their heterogeneity. We have observed that most industrial SAT families are homogeneous, and many of them are clearly characterized by these structure features. For instance, the industrial family *hardware-velev* is characterized by an exponent $\alpha_v$ in the interval $[1.4, 3]$, high fractal dimensions, with $d > 4$ and $d^b > 7$, and a modularity $Q$ in the interval $[0.5, 0.8]$.

Next, we want to determine if this reduced set of $4$ structure features plus the clause/variable ratio $m/n$ has similar results classifying industrial SAT families than other sets of SAT features commonly used in portfolio approaches. In particular, we use the set of SAT features used in the portfolio SAT solver SATzilla [47]. The version of SATzilla submitted to the SAT Competition 2012 uses 127 features grouped in several categories: problem size, graphs (including statistics about the degree or clustering coefficient of nodes in the VIG and CVIG, among others), hardness (including DPLL, LP-based, SLS, clause learning and survey propagation statistics), balance, and timing[1] features. See [47] for a detailed description of these features. In this analysis, we consider the set of $115$ resulting from removing the $12$ timing features. We name the set of 5 structure features as *Structure*, and the set of 115 SATzilla features as *SATzilla* in our analysis.

In a second experiment, we build several classifiers using both the *Structure* and the *SATzilla* sets of features, in order to classify the industrial SAT family of a given instance. For each classifier, we measure the number of correctly classified instances (i.e., the number of SAT instances whose industrial family was correctly predicted). Notice that we know *a priori* the family each industrial SAT instance belongs to. Therefore, we use supervised machine learning techniques. In order to evaluate each classifier, we perform a $k$-folds cross-validation (i.e., dividing the set of instances into $k$ folds such that each fold is evaluated using the classifier built with the other $k - 1$ folds), with $k = 10$. Let us introduce the classifiers used in this experiment:

- **C4.5**. This algorithm [41] generates a decision tree to determine the category of each element. It is an improved extension of the earlier ID3 algorithm.

- **Random Forest (RF)**. This model [16] builds a combination, or forest, of random decision trees, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

- **Naïve Bayes (NB)**. This algorithms [27] models a probability distribution with a Bayesian network, and handles continuous variables using statistical methods for non-parametric density estimations.

[1] In SATzilla, some features represent the runtime needed to compute some categories of features (e.g., the runtime of computing graph features).

| | *Structure* | *SATzilla* |
|---|---|---|
| C4.5 | 259 (86.33%) | 263 (87.67%) |
| RF | **274 (91.33%)** | **288 (96.00%)** |
| NB | 254 (84.67%) | 256 (85.33%) |
| MLR | 247 (82.33%) | 262 (87.33%) |
| LR | 251 (83.67%) | **280 (93.33%)** |
| SMO | 153 (51.00%) | 241 (80.33%) |
| IBk | **275 (91.67%)** | 264 (88.00%) |
| K* | **273 (91.00%)** | 199 (66.33%) |
| JRip | 246 (82.00%) | 251 (83.67%) |

Table 1: Number of correctly classified instances (and its percentage over the total set of instances in brackets), using the *Structure* features ($\alpha_v$, $Q$, $d$ and $d^b$ plus the clause/variable ratio $m/n$) or the 115 *SATzilla* features, for some classifiers. In bold, we remark those classifiers whose effectiveness is higher than $90\%$.

- **Multi-response Linear Regression (MLR)**. This classifier [22] transforms the classification problem into a problem of function approximation, and this approximation is performed using regression methods.

- **Logistic Regression (LR)**. This algorithm [30] builds and uses a multinomial logistic regression model with a ridge estimator.

- **Sequential Minimal Optimization (SMO)**. This classifier [40] trains a Support Vector Machine (SVM), reducing this training problem into a series of smallest possible quadratic programming problems.

- **IBk**. This method [2] implements the instance-based learning $k$-nearest neighbors algorithm, with a fixed value of $k$.

- **K***. This model [18] uses the notion of entropy as a distance measure to determine the similarity between two instances.

- **JRip**. This algorithm [19] implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER).

In Table 1, we represent the number of correctly classified instances by these classifiers, using the features sets *Structure* and *SATzilla*. We run each classifier with their default parameters values used in Weka [25]. In bold, we remark those classifiers whose effectiveness is higher than $90\%$, i.e., they correctly classify the industrial family of more than $90\%$ of the 300 industrial SAT instances. As we observe, the effectiveness of the classifier is, in some cases, very low. This is the case of SMO and K* for the classifications of *Structure* and *SATzilla* sets, respectively, for which the reduced/large number of features has a negative effect on the classifier. However, most of these classifiers have a very high effectiveness. In general, using the set of features *SATzilla* slightly outperforms the results of the set *Structure*. However, the differences between these two sets are very small. It is worth noting that while the set *SATzilla* contains a total of 115 features, the proposed set *Structure* only contains 5 features, and even so, the obtained classification and its effectiveness is similar. Interestingly, the classifiers K* and IBk improve their performance when using the set *Structure*.

Let us conjecture why this is the case. SATzilla characterizes the structure of SAT instances using a total of $14$ graph features. However, these features represent *local* properties of its structure. For instance, the distribution of node degree is analyzed in SATzilla using some statistical features: maximum, average, median, standard deviation and minimum. Even so, these $5$ features only characterize some *local* properties of the graph. We say they are *local* in the sense that none of them (used separately) speaks about a common behavior in the whole graph. On the other hand, in our metrics we use the exponent $\alpha_v$, which characterizes the distribution of degrees, and thus it is a *global* property of the graph. Therefore, the previously mentioned $5$ graph features used by SATzilla may be *implied* using the exponent $\alpha_v$. Similarly, the clustering of variables is analyzed in SATzilla using the clustering coefficient, computing the values maximum, average, median, standard deviation and minimum for each node in the VIG. Again, these metrics only characterize a very *local* community structure. However, the modularity $Q$ is a *global* metric of the graph, and therefore it gives a stronger information about this clustering.

In summary, SATzilla uses many (*local*) features to determine the structure of the formula, but even so, some global characteristics of such structure are not represented. On the other hand, we simply characterize it with $4$ (*global*) graph features. Remark that we include the clause/variable ratio $m/n$ in our set of features as a very simple metric about the hardness of the formula[2], while SATzilla analyzes it in a more exhaustive way using a total of $71$ hardness features. Therefore, our characterization of their hardness is still weak.

In conclusion, we observe that the proposed set of structure features can be useful for classifying industrial SAT families of instances. This can beneficial for the specialization of SAT solvers, which may exploit the particularities of each industrial SAT family, when used in portfolio SAT solving approaches.

## 5. Evaluation of Structure Features in a Portfolio SAT Solver

In this section, we analyze the performance of a portfolio SAT solver when it is trained with the set of structure features presented in the previous section. In particular, we evaluate the performance of the solver ISAC [28], and we compare it when it is trained with the set of SATzilla features. For each set of features, the performance is evaluated with a 10-fold cross validation. This means that the set of 300 SAT instances is randomly divided into 10 disjoints subsets, or folds. For each fold, ISAC is trained using the remaining 9 folds (training set) to build a prediction model, and this is used it to predict the best (core) solver to solve each instance of this fold (testing set). We use all solvers submitted to the application track of the SAT Competition 2013 as core solvers. Finally, the reported runtime of solving a SAT instance is the runtime of computing its features plus the runtime of the (core) solver selected by ISAC.

[2] While the hardness of random $k$-CNF can be characterized using the clause/variable ratio $m/n$, this is not the case in industrial SAT instances. However, bigger industrial SAT formulas may be intuitively harder.

| | Structure | | SATzilla |
|---|---:|---:|---:|
| runtime | VIG+CVIG | VIG | |
| minimum | 0.07 | **0.04** | 11.71 |
| median | 4.9 | **3.31** | 49.24 |
| average | 21.65 | **17.70** | 170.43 |
| stdev | 36.87 | **34.11** | 362.27 |
| maximum | 287.12 | **275.11** | 3675.28 |

Table 2: Statistics results of the runtime (in seconds) of computing the set of SAT features over the set of 300 industrial SAT instances of the SAT Competition 2013, for the set *Structure*, using only graph features of the VIG, and using graph features of both VIG and CVIG; and for the set *SATzilla*. We remark in bold the fastest method.

| runtime | Structure$_{VCr}$ | Structure$_{Vr}$ | Structure$_{VC}$ | Structure$_{V}$ | SATzilla |
|---|---:|---:|---:|---:|---:|
| minimum | 0.26 | 0.26 | 0.26 | 0.26 | 0.26 |
| median | 272.62 | 268.10 | 262.02 | 310.27 | 261.84 |
| average | 874.59 | 863.32 | 876.84 | 881.15 | 831.61 |
| stdev | 1199.59 | 1197.70 | 1200.11 | 1207.83 | 1143.55 |
| maximum | 4913.23 | 4913.23 | 4913.23 | 4913.23 | 4745.21 |
| #solved | 285 | 281 | 285 | 281 | **288** |

Table 3: Statistics about the runtime required to solve the set of 300 instances of the SAT Competition 2013 by ISAC trained with different set of features.

Let us analyze first the cost of computing the set of structure features presented in this paper. Notice that this set contains features of both the VIG and the CVIG. Therefore, it is plausible to consider only computing the features of one of these graphs. In Table 2, we present some statistics of the runtime needed to compute these sets of features. For the case of the set *Structure*, we consider two cases: the features of the VIG, and the features of both VIG and CVIG. We observe that computing the structure features is, in general, more than one order of magnitude faster than computing the SATzilla features[3], even when we use our two graphs (VIG and CVIG). As expected, computing the structure features in only one graph (VIG) is faster than using both of them (VIG and CVIG).

In Figure 2, we represent the cactus plot of solving the 300 industrial SAT instances of the SAT Competition 2013 by ISAC when it is trained with the set of features *SATzilla* and some combinations of features from the set *Structure*. This plot represents the runtime (in seconds) needed to solve a certain number of instances, i.e., each point $(x, y)$ represents that $x$ instances were solved in at most $y$ seconds (each of them). The combinations of structure features, named as *Structure*$_X$, uses VIG features (i.e., $\alpha_v$, $Q$ and $d$) when ’$V$’ $\in X$, CVIG features (i.e., $d^b$) when ’$C$’ $\in X$, and the clause/variable ratio when ’$r$’ $\in X$. In Table 3, we also report some statistics about the results.

We observe that using the set of 115 *SATzilla* features is the best strategy, solv-

[3] We use the tool provided in the solver SATzilla.

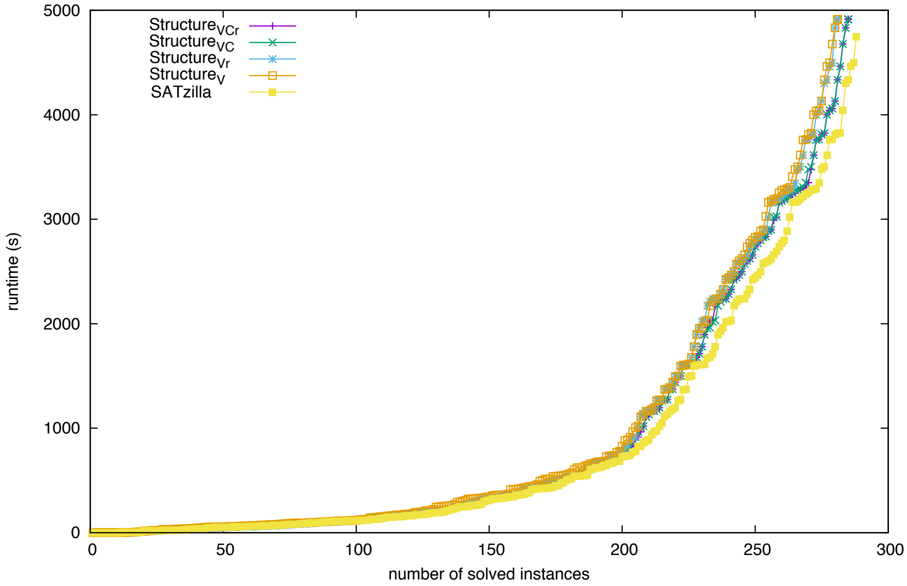

Figure 2: Cactus plot for the solver ISAC trained with the set of features *SATzilla* and some subsets of features from the set *Structure*. Specifically, *Structure*$_{VCr}$ uses $\alpha_v$, $Q$, $d$, $d^b$ and $m/n$; *Structure*$_{VC}$ uses $\alpha_v$, $Q$, $d$ and $d^b$; *Structure*$_{Vr}$ uses $\alpha_v$, $Q$, $d$ and $m/n$; and *Structure*$_V$ uses $\alpha_v$, $Q$ and $d$. Each point $(x, y)$ in the plot represents that $x$ instances were solved in at most $y$ seconds (each of them).

ing a total of 288 instances. Surprisingly, using the *Structure* set of features has very similar. In particular, using the features of the VIG and CVIG (i.e., *Structure*$_{VC}$ and *Structure*$_{VCr}$) solves 285 instances in both cases, while using only VIG features (i.e., *Structure*$_V$ and *Structure*$_{Vr}$) solves 281 instances in both cases. Moreover, using the clause/variable ratio $m/n$ does not affect the performance. Therefore, the number of instances solved by these methods is very similar in all cases. Also, the runtime required for solving them is also very similar in all cases (see Table 3).

In conclusion, we show that computing structure SAT features is, in general, much faster than computing other sets of SAT features commonly used in portfolio approaches, as the set used by SATzilla. Also, we observe that the performance of a portfolio SAT solver is almost unaffected when, instead that training it with the 115 SATzilla features, we train it using just only 5 (or less) structure features, and the small differences between *SATzilla* and *Structure* are probably due to the richer study of the hardness performed by SATzilla.

## 6. Relevance of Structure SAT Features

In this section, we analyze the *relevance* of all features used in the previous experiments. They are the 115 features of SATzilla plus the 4 structure features used in this paper. We want to evaluate their relevance independently of the classification method used (e.g, in a portfolio SAT solver). For this purpose we use a filtering method for feature selection, a classical method of machine learning.

In the minimal-Redundancy-Maximum-Relevance method (mRMR) [39], we try to select a subset of features mutually as dissimilar to each other as possible (minimal *redundancy*), but marginally as similar to the classification variable as possible (maximal *relevance*). This is achieved finding the subset $S$ of features that maximizes:

$$\max_S \left( \sum_{j \in S} I(x_j, c) - \frac{1}{M-1} \sum_{\substack{i,j \in S \\ i<j}} I(x_i, x_j) \right) \tag{2}$$

where $c$ is the classification variable, $M$ is the number of features, and $I(x_i, x_j)$ measures the mutual information between features $x_i$ and $x_j$. Formally, the mutual information between two random variables $x_i$ and $x_j$ is defined as:

$$I(x_i, x_j) = \int \int p(x_i, x_j) \log \frac{p(x_i, x_j)}{p(x_i) p(x_j)} dx_i dx_j$$

Computing mutual information is based on estimating the probability distributions $p(x_i)$, $p(x_j)$ and $p(x_i, x_j)$. These distributions can be either discretized or estimated by density functions methods [44]. The first term of Eq. 2 computes the relevance and the second term the redundancy. For real applications, this objective function is difficult to compute exactly. In [39], they propose to use a greedy or gradient algorithm that, starting with $S = \emptyset$, proceeds adding the feature that most increases the objective function at each step.

In [44], they write the previous objective function as a pseudoBoolean quadratic function, and use a parameter $\alpha \in [0, 1]$ to regulate the relative weight between relevance and redundancy:

$$\min_{X \in \{0,1\}^M} \left\{ \frac{1}{2}(1-\alpha) X^T Q X - \alpha F^T X \right\} \tag{3}$$

where $q_{ij} = I(x_i, x_j)$ and $f_j = I(x_j, c)$. In general, values of $\alpha$ closer to 1 result into smaller sets of selected features optimizing the objective function. They use the Nyström method to approximate the optimum of this function.

In our experiments, we compute the relevance $I(x_j, c)$ of all the $119$ features $x_j$ (i.e., the $115$ SATzilla features plus the $4$ structure features) as proposed in [44]. Recall that the classification variable $c$ corresponds in our case to the industrial SAT family each instance belongs to. The relevance of a set of features is the matrix $F$ according to Eq. 3. In Table 4, we report the most relevant features, as well as the ranking of the structure features, with their correspondent relevance value (higher is better). We observe that $2$ of our $4$ structure features are between the $5$ most relevant features. In particular, they are the fractal dimension for the VIG $d$ (ranked in the second position), and the fractal dimension for the CVIG (ranked in the fifth position). The exponent $\alpha_v$ appears in the position $22$, and the modularity $Q$ in the position $41$. Remark that we use $\log \alpha_v$ (instead of using directly $\alpha_v$) because some industrial families are characterized by an exponent $\alpha_v$ some orders of magnitude higher than the rest. Recall that we are

| rank | feature (and description) | category | relevance |
|---|---|---|---|
| 1 | *SP-bias-mean*:<br>mean of confidence of survey propagation (the higher of $P(true)/P(false)$ or $P(false)/P(true)$) for each variable | Survey Prop | 0.6359 |
| 2 | $d$: fractal dimension for VIG | ***Structure*** | 0.5541 |
| 3 | *POSNEG-RATIO-VAR-max*:<br>max of ratio of positive to negative occurrences of each variable | Balance | 0.5231 |
| 4 | *POSNEG-RATIO-CLAUSE-coeff-variation*:<br>variation coefficient of ratio of positive to negative literals in each clause | Balance | 0.5171 |
| 5 | $d^b$: fractal dimension for CVIG | ***Structure*** | 0.4769 |
| 6 | *SP-unconstraint-coeff-variation*:<br>variation coefficient of probability that a variable is unconstrained in survey propagation | Survey Prop | 0.4250 |
| 7 | *POSNEG-RATIO-VAR-mean*:<br>mean of ratio of positive to negative occurrences of each variable | Balance | 0.4168 |
|  | . . . | . . . |  |
| 22 | $\log \alpha_v$: powerlaw exponent | ***Structure*** | 0.2558 |
|  | . . . | . . . |  |
| 41 | $Q$: modularity (for VIG) | ***Structure*** | 0.1844 |
|  | . . . | . . . |  |

Table 4: Relevance of SAT features to classify industrial SAT families.

considering a total of 119 SAT features. Therefore, structure features are very relevant features.

It is also interesting to remark that the 5 most relevant SATzilla features belong to the categories *survey propagation* and *balance*. Survey propagation is a technique, based on works of spin glasses and statistical physics, to estimate the probability that a Boolean variable has a certain value in all satisfying assignments. The category *Balance* refers to the ratio of positive and negative polarities of literals appearing in the formula. For instance, a totally unbalanced formula only contains variables appearing with the same polarity. Notice that this kind of formulas are trivially satisfiable by the pure literal rule. On the contrary, in [4, 5] it is shown that balanced formulas produce hard SAT instances. Therefore, the most relevant SATzilla features are related to the *hardness* of the instance.

Finally, we analyze the redundancy between pairs of features. This is the matrix $Q$ according to Eq. 3. We represent the results in Fig. 3 as a heat map, i.e., the redundancy between features $x_i$ and $x_j$ is represented in the point $(i, j)$ (and $(j, i)$) according to the

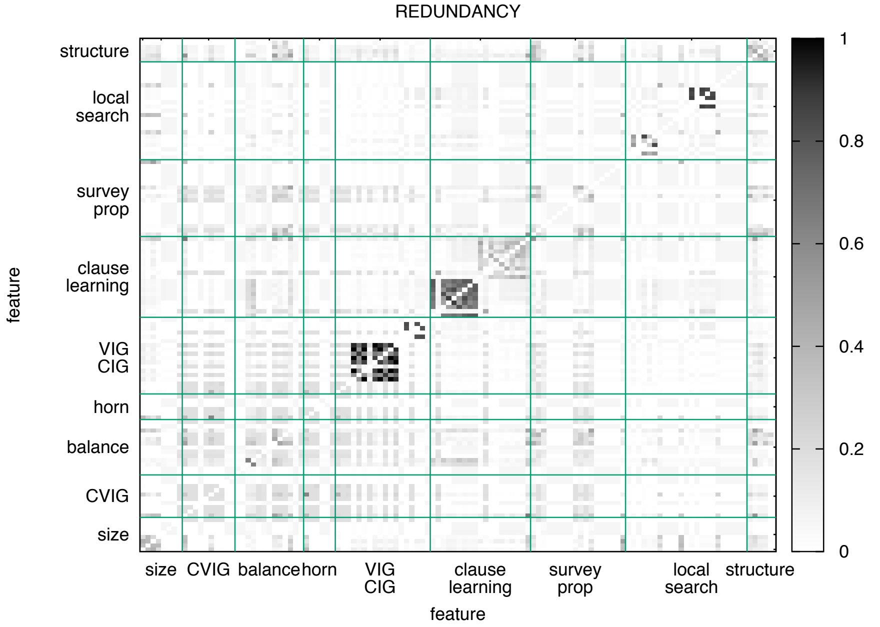

Figure 3: Redundancy between pairs of features. Each point $(x, y)$ represents the redundancy between features $x$ and $y$, according to the grey scale. Features are grouped by SATzilla categories, and *Structure* category contains $\alpha_v$, $d$, $d^b$ and $Q$ (in this order).

grey scale indicated in the figure[4]. The values of this matrix are normalized between $0$ and $1$. Features are grouped into SATzilla categories: problem *size*, *CVIG* node degree statistics, *balance*, proximity to *horn* formula, *VIG* node degree and diameter statistics and *CVIG* node degree and clustering coefficient statistics, *clause learning* statistics (based on 2 seconds of running Zchaff_rand), *survey propagation*, and *local search* statistics (based on 2 seconds of running each of SAPS and GSAT). Finally, our set of *structure* features includes $\alpha_v$, $d$, $d^b$ and $Q$.

We observe that the most redundant pairs of features are found within each category. For instances, many VIG/CIG features are very redundant with each other. In the case of our proposed $4$ *structure* features, we observe the same behavior. In particular, the pairs $(\alpha_v, Q)$ and $(d, d^b)$ are the most redundant in this category. However, these redundancies have a normalized value around $0.3$, i.e., they are not very redundant. Interestingly, the most redundant categories w.r.t. *structure* features are the categories *balance* and *survey propagation*, which are also the most relevant features.

[4] It has no sense to compute the redundancy for points $(i, i)$.

## 7. Conclusions

In this paper, we have presented a set of structure features that (partially) defines the structure of industrial SAT instances. In particular, they are the exponent $\alpha_v$ of the power-law distribution that best fits the number of variable occurrences, the modularity $Q$ of its VIG, and the fractal dimension $d$ and $d^b$ of its VIG and CVIG, respectively.

We have shown that using this reduced set of $4$ features (plus the clause/variable ratio $m/n$) to classify industrial SAT families results into an effective classification. Its effectiveness is comparable to the effectiveness of the classification obtained with other sets of SAT features commonly used in portfolio approaches, as the set used by SATzilla.

Also, we have observed that computing this set of structure features is, in general, more than one order of magnitude faster than computing SATzilla features. We have evaluated the performance of the portfolio SAT solver ISAC after being trained with these two sets of SAT features (*Structure* and *SATzilla*). We observe that the performance of this solver is very similar in both cases.

Finally, we have analyzed the relevance of both set of features, and we show that structure features are very relevant, as other hardness features (as *Survey Propagation* or *balance* statistics) computed by SATzilla.

## References

## References

[1] D. Achlioptas. Random satisfiability. In *Handbook of Satisfiability*, pages 245–270. IOS Press, 2009.

[2] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] R. Albert, H. Jeong, and A.-L. Barabási. The diameter of the WWW. *Nature*, 401:130–131, 1999.

[4] C. Ansótegui, R. Béjar, C. Fernández, C. P. Gomes, and C. Mateu. The impact of balancing on problem hardness in a highly structured domain. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI'06)*, pages 10–15, 2006.

[5] C. Ansótegui, R. Béjar, C. Fernández, and C. Mateu. On balanced csps with high treewidth. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07*, pages 161–166, 2007.

[6] C. Ansótegui, M. L. Bonet, J. Giráldez-Cru, and J. Levy. The fractal dimension of SAT formulas. In *Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR'14)*, pages 107–121, 2014.

[7] C. Ansótegui, M. L. Bonet, J. Giráldez-Cru, and J. Levy. On the classification of industrial SAT families. In *Proceedings of the 18th International Conference of the Catalan Association for Artificial Intelligence (CCIA'15)*, pages 163–172, 2015.

[8] C. Ansótegui, M. L. Bonet, and J. Levy. On the structure of industrial SAT instances. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming (CP'09)*, pages 127–141, 2009.

[9] C. Ansótegui, M. L. Bonet, and J. Levy. Towards industrial-like random SAT instances. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 387–392, 2009.

[10] C. Ansótegui, M. L. Bonet, J. Levy, and F. Manyà. Measuring the hardness of SAT instances. In *Proceedings of the 23th AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 222–228, 2008.

[11] C. Ansótegui, J. Giráldez-Cru, and J. Levy. The community structure of SAT formulas. In *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT'12)*, pages 410–423, 2012.

[12] O. Beyersdorff and O. Kullmann. Unified characterisations of resolution hardness measures. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT'14)*, pages 170–187, 2014.

[13] A. Biere. Bounded model checking. In *Handbook of Satisfiability*, pages 457–481. IOS Press, 2009.

[14] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[15] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Trans. on Knowledge and Data Engineering*, 20(2):172–188, 2008.

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

[17] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *Arxiv*, 0706.1062, 2007.

[18] J. G. Cleary and L. E. Trigg. K*: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning (ML'95)*, pages 108–114, 1995.

[19] W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ML'95)*, pages 115–123, 1995.

[20] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC'71)*, pages 151–158, 1971.

[21] P. Erdós and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[22] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.

[23] J. Giráldez-Cru and J. Levy. A modularity-based random SAT instances generator. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 1952–1958, 2015.

[24] C. P. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43–62, 2001.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[26] M. Järvisalo, D. L. Berre, O. Roussel, and L. Simon. The international SAT solver competitions. *AI Magazine*, 33(1), 2012.

[27] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pages 338–345, 1995.

[28] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC - instance-specific algorithm configuration. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*, pages 751–756, 2010.

[29] D. Kroening. Software verification. In *Handbook of Satisfiability*, pages 505–532. IOS Press, 2009.

[30] S. le Cessie and J. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[31] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[32] Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann. Non-model-based algorithm portfolios for SAT. In *Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing (SAT'11)*, pages 369–370, 2011.

[33] B. B. Mandelbrot. *The fractal geometry of nature*. Macmillan, 1983.

[34] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, 2004.

[35] Z. Newsham, V. Ganesh, S. Fischmeister, G. Audemard, and L. Simon. Impact of community structure on SAT solver performance. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT'14)*, pages 252–268, 2014.

[36] Z. Newsham, W. Lindsay, V. Ganesh, J. H. Liang, S. Fischmeister, and K. Czarnecki. SATGraf: Visualizing the evolution of SAT formula structure in solvers. In *Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing (SAT'15)*, pages 62–70, 2015.

[37] M. Nikolic, F. Maric, and P. Janicic. Instance-based selection of policies for SAT solvers. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, pages 326–340, 2009.

[38] M. Nikolic, F. Maric, and P. Janicic. Simple algorithm portfolio for SAT. *Artif. Intell. Rev.*, 40(4):457–465, 2013.

[39] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, 2005.

[40] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1998.

[41] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[42] J. R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[43] J. Rintanen. Planning and SAT. In *Handbook of Satisfiability*, pages 483–504. IOS Press, 2009.

[44] I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. S. Cruz. Quadratic programming feature selection. *Journal of Machine Learning Research*, 11:1491–1516, 2010.

[45] B. Silverthorn and R. Miikkulainen. Latent class models for algorithm portfolio methods. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI'10)*, 2010.

[46] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. The design and analysis of an algorithm portfolio for SAT. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP'07)*, pages 712–727, 2007.

[47] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32(1):565–606, 2008.