On the Classification of Industrial SAT Families

Carlos ANSÓTEGUI [a], Maria Luisa BONET [b], Jesús GIRÁLDEZ-CRU [c] and Jordi LEVY [c]
[a] *Universitat de Lleida (DIEI, UdL)*
[b] *Universitat Politècnica de Catalunya (LSI, UPC)*
[c] *Artificial Intelligence Research Institute (IIIA-CSIC)*

Abstract. The success of portfolio approaches in SAT solving relies on the observation that different SAT solving techniques perform better on different SAT instances. The Algorithm Selection Problem faces the problem of choosing, using a prediction model, the best algorithm from a predefined set, to solve a particular instance of a problem. Using Machine Learning techniques, this prediction is performed by analyzing some features of the instance and using an empirical hardness model, previously built, to select the expected fastest algorithm to solve such instance.

Recently, there have been some attempts to characterize the structure of industrial SAT instances. In this paper, we use some structural 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. We measure the effectiveness of these classifiers by comparing them to other sets of SAT instances features commonly used in portfolio SAT solving approaches. Finally, we evaluate the performance of this set of structural features when used in a real portfolio SAT solver.

Keywords. 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. However, besides its complexity, it is extensively used in many real-world domains to efficiently solve many real-world problems, as planning, bounded model checking, software and hardware verification, scheduling or cryptography, among others.

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 so known *portfolio*. This approach faces the Algorithm Selection Problem [22], 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. This predictor is built using Machine Learning techniques. This way, portfolio SAT solvers proceed as follows. In a first step, an offline process is performed: for a set of training SAT instances, a vector of features is computed, and they are solved by a predefined set of solvers, which generates a vector of runtimes.

Then, a classifier is built using these two vectors. Finally, in a online step, for a given instance the portfolio computes its features, and it solves it using the (expected) best solver using the empirical hardness model previously built.

The success of portfolio 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 dominants solving industrial SAT instances; Look-Ahead SAT solvers are specially efficient solving random SAT instances; Stochastic Local Search (SLS) SAT solvers have exhibit a very good performance on satisfiable instances, specially in random k-CNF.

In the case of industrial SAT instances, the remarkable success of CDCL SAT solving techniques 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 [4], community structure [5], and self-similar structure [3].

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 structural 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 structural features even improves the performance of the classifier. Finally, we evaluate the performance of a real portfolio SAT solver using these structural features, observing that its performance is almost unaffected .

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 structural 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 structural features w.r.t. other set of SAT features commonly used in portfolio approaches. Finally, we conclude in Section 6.

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 $w : V_1 \times V_2 \to \mathbb{R}^+$.

The Variable Incidence Graph (VIG) of a SAT instance Γ is the graph whose nodes represent the variables of Γ, and there exists an edge between two variables if they both appear in a clause c. A clause with l literals results into $\binom{l}{2}$ edges. Thus, to give the same relevance to all clauses, edges have a weight $w(x,y) = \sum_{\substack{c\in\Gamma\\ x,y\in c}} 1/\binom{|c|}{2}$, where $|c| = l$ is the length of the clause c. Notice that the sum of the weights of the edges generated by a single clause is 1, independently of its length.

The Clause-Variable Incidence Graph (CVIG) of a SAT instance Γ is the bi-partite graph whose nodes represent either the variables of Γ or the clauses of Γ, and there exists an edge between a variable v and a clause c if v appears in c. Again, we give the same relevance to all clauses giving edges a weight $w(v,c) = \sum_{\substack{c\in\Gamma\\ v\in c}} 1/|c|$.

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 [4], the community structure [5], and the self-similar structure [3]. 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* [12], 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 [2] 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 α_v of the power-law distribution that best fits this collection of points. This estimation is computed by the method of maximum likelihood [9].

3.2. The Community Structure

The *community structure* of a graph is usually measured using the notion of *modularity* Q [19]. 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. This avoids that the best partition is the one made up by an only community containing all vertexes. The modularity of a graph is the maximal modularity for any possible partition: $Q(G) = \max\{Q(G,C) \mid C\}$.

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

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 [7], instead that computing the (exact) value of the modularity, most methods in the literature approximate a lower-bound in the value of Q. In [5], it is used the *Louvain method* [6], one of the most accurate algorithms.

3.3. The Self-Similar Structure

A self-similar graph keeps its structure after *rescaling* it. This means replacing groups of nodes by a single node, for example. 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 $d^{max} \sim \log n$, as 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 [18].

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., $N(r) \approx r^{-d}$), 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.

4. Classifying Industrial SAT Families

Most industrial SAT instances exhibit a power-law distribution in the number of variable occurrences [4], a clear community structure with high modularity [5], and a fractal dimension that characterizes their self-similarity [3]. Therefore, for each industrial SAT instance we compute the exponent α_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 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

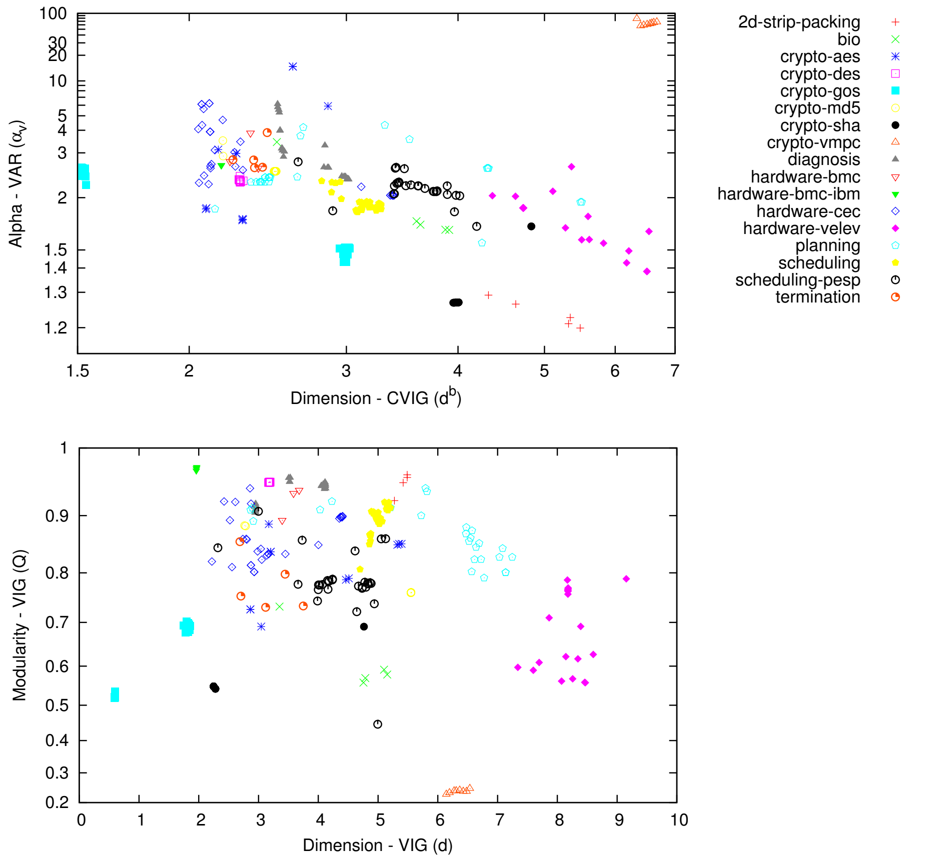

Figure 1. Distribution of families according to the exponent of the power-law distribution of variable occurrences (α_v), the fractal dimensions of the VIG and CVIG (d and d^b, respectively), and the modularity (Q). Some heterogeneous families (*software-bit-verif* and *software-bmc*) are not plotted.

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 structural features. In Figure 1, we represent the relation between the scale-free structure (α_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

	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 (α_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%.

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 structural features. For instance, the industrial family *hardware-velev* is characterized by an exponent α_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 structural features plus the clause/variable ratio m/n (in the experiments, this set of 5 features is named as ***Structure***) 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 [23]. 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), and timing[1] features. See [23] for a detailed description of these features. In this analysis, we consider the set of 115 resulting from removing the 12 timing features (in the experiments, this set of 115 is named as ***SATzilla***).

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 learning techniques. In order to evaluate each classifier, we perform a 10-folds cross-validation. Let us introduce the classifiers used in this experiment:

- **C4.5**. This algorithm [21] 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 [8] 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 [15] 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., problem size).

- **Multi-response Linear Regression (MLR)**. This classifier [13] transforms the classification problem into a problem of function approximation, and this approximation is performed using regression methods.
- **Logistic Regression (LR)**. This algorithm [17] builds and uses a multinomial logistic regression model with a ridge estimator.
- **Sequential Minimal Optimization (SMO)**. This classifier [20] trains a Support Vector Machine (SVM), reducing this training problem into a series of smallest possible quadratic programming problems.
- **IBk**. This method [1] implements the instance-based learning k-nearest neighbors algorithm, with a fixed value of k.
- **K***. This model [10] uses the notion of entropy as a distance measure to determine the similarity between two instances.
- **JRip**. This algorithm [11] 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 [14]. 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, 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 degrees of nodes is analyzed in SATzilla via 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) can explain a common behavior in the whole graph. On the other hand, in our metrics we use the exponent α_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 α_v. Similarly, the links between the neighbors of a particular node of the graph are analyzed in SATzilla via the clustering coefficient, computing the maximum, the average, the median, the standard deviation and the minimum. Again, these metrics only characterize the community structure very *locally*. However, the modularity Q is a *global* metric of the graph, and therefore it gives a richer information about this structure. In conclusion, we think that SATzilla needs to include many (*local*) features to analyze the structure of the formula, while we simply characterize it with 4 (*global*) graph features, and even so, SATzilla is still missing some important information about the structure of the graph. 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 for-

	Structure		SATzilla
runtimes	VIG+CVIG	VIG	
minimum	0.07	**0.04**	11.71
median	4.9	**3.31**	49.24
average	21.65	**17.70**	170.43
std	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.

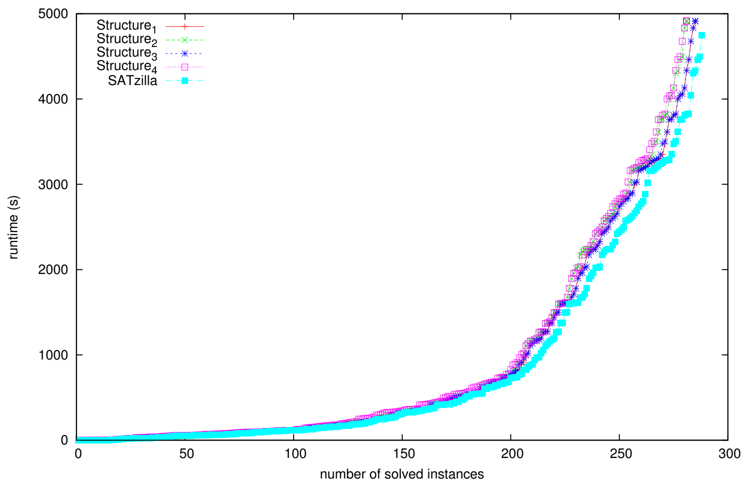

Figure 2. Cactus plot for the solver ISAC trained with the set of features *SATzilla* and some subsets of features from the set **Structure**. Specifically, *Structure*$_1$ uses α_v, Q, d, d^b and m/n; *Structure*$_2$ uses α_v, Q, d and m/n; *Structure*$_3$ uses α_v, Q, d and d^b; and *Structure*$_4$ uses α_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).

mula[2], while SATzilla analyzes it in a more exhaustive way using a total of 71 hardness features. Therefore, even when we are able to characterize the structure of SAT instances properly, our characterization of their hardness is still weak.

In conclusion, we observe that the proposed set of structural 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 Structural 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 structural features presented in the previous section. In particular, we

[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.

evaluate the performance of the solver ISAC [16], 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.

Let us analyze first the cost of computing the set of structural 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 structural 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 structural 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 structural features, named as *Structure*$_x$, uses VIG+CVIG features if $x = \{1, 3\}$, or only VIG features if $x = \{2, 4\}$; and considers the clause/variable ratio m/n if $x = \{1, 2\}$, or they do not if $x = \{3, 4\}$. While using *SATzilla* features solves a total of 288 instances, using *Structure*$_x$ features solves 285, 281, 285 and 281 instances for $x = \{1, 2, 3, 4\}$, respectively. 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 Figure 2). Moreover, we do not observe significant differences between using VIG+CVIG features or using only VIG features (*Structure*$_{1,2}$ solves exactly the same as *Structure*$_{3,4}$). Finally, using the clause/variable ratio m/n is useful (*Structure*$_{1,3}$ solves more instances than *Structure*$_{2,4}$).

In conclusion, we show that computing structural 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) structural features, and the small differences between *SATzilla* and *Structure* are probably due to the richer study of the hardness performed by SATzilla.

6. Conclusions

In this paper, we have presented a set of structural features that (partially) defines the structure of industrial SAT instances. In particular, we use 4 structural features: the ex-

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

ponent α_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; plus the clause/variable ratio m/n. We have shown that using this reduced set of 5 features to classify industrial SAT families results into an effective classification. Its effectiveness is comparable to the one 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 structural features is, in general, more than one order of magnitude faster than computing SATzilla features. Finally, 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.

References

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

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

[3] C. Ansótegui, M. L. Bonet, J. Giráldez-Cru, and J. Levy. The fractal dimension of SAT formulas. In *Proc. of IJCAR'14*, pages 107–121, 2014.

[4] C. Ansótegui, M. L. Bonet, and J. Levy. On the structure of industrial SAT instances. In *Proc. of CP'09*, pages 127–141, 2009.

[5] C. Ansótegui, J. Giráldez-Cru, and J. Levy. The community structure of SAT formulas. In *Proc. of SAT'12*, pages 410–423, 2012.

[6] 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.

[7] 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.

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

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

[10] J. G. Cleary and L. E. Trigg. K*: An instance-based learner using an entropic distance measure. In *Proc. of ML'95*, pages 108–114, 1995.

[11] W. W. Cohen. Fast effective rule induction. In *Proc. of ML'95*, pages 115–123. Morgan Kaufmann, 1995.

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

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

[14] 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.

[15] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proc. of UAI'15*, pages 338–345, 1995.

[16] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC - instance-specific algorithm configuration. In *Proc. of ECAI'10*, pages 751–756, 2010.

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

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

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

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

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

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

[23] 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.