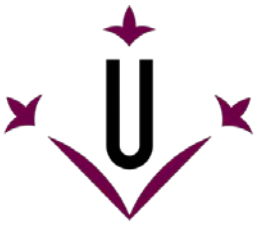

The Fractal Dimension of SAT Formulas

Carlos Ansótegui[1], Maria Luisa Bonet[2], Jesús Giráldez-Cru[3], and Jordi Levy[3]

[1] DIEI, Univ. de Lleida, carlos@diei.udl.cat
[2] LSI, UPC, bonet@lsi.upc.edu
[3] IIIA-CSIC, jgiraldez,levy@iiia.csic.es

Abstract. Modern SAT solvers have experienced a remarkable progress on solving industrial instances. Most of the techniques have been developed after an intensive experimental process. It is believed that these techniques exploit the underlying structure of industrial instances. However, there is not a precise definition of the notion of structure.
Recently, there have been some attempts to analyze this structure in terms of complex networks, with the long-term aim of explaining the success of SAT solving techniques, and possibly improving them.
We study the fractal dimension of SAT instances with the aim of complementing the model that describes the structure of industrial instances. We show that many industrial families of formulas are self-similar, with a small fractal dimension. We also show how this dimension is affected by the addition of learnt clauses during the execution of SAT solvers.

1 Introduction

The SAT community has been able to come up with successful SAT solvers for industrial applications. However, nowadays we can hardly explain why these solvers are so efficient working on industrial SAT instances with hundreds of thousands of variables and not on random instances with hundreds of variables. The common wisdom is that the success of modern SAT/CSP solvers is correlated to their ability to exploit the hidden structure of real-world instances [13]. Unfortunately, there is no precise definition of the notion of structure.

Parallelly, the community of complex networks has produced tools for describing and analyzing the structure of social, biological and communication networks [1] which can explain some interactions in the real-world. *Preferential attachment* (where the probability that a new edge is attached to a node is proportional to its degree) has been proposed as the responsible of scalefree structure in real-world graphs [5]. Thus, in the web, the probability of a web page to get new connections is proportional to its popularity (the number of connections it already has). In cite [9], it is proposed *similarity* (where nodes tend to get connected to similar nodes, according to some topological distance) as a mechanism that, together with preferential attachment or *popularity*, explains the structure of some real-world graphs. This explains the self-similarity property observed in many real-world graphs [11].

Representing SAT instances as graphs, we can use some of the techniques from complex networks to characterize the structure of SAT instances. Recently,

some progress has been made in this direction. It is known that many industrial instances have the *small-world* property [12], exhibit high *modularity* [4], and have a *scale-free structure* [2]. In this later work, it is shown that in many formulas the number of occurrences of a variable (i.e. the degree of graph nodes) follows a powerlaw distribution with *hub* variables having a huge number of occurrences. A method to generate scale-free random instances is proposed in [3]. They show that SAT solvers specialized on industrial formulas perform better than random-specialized solvers on these scale-free random instances. In [7], the *eigenvector centrality* of variables in industrial instances is analyzed. They show that it is correlated with some aspects of SAT solvers. For instance, decision variables selected by the SAT solvers are usually the most central variables in the formula. However, how these analyses may help to improve the performance of SAT solvers is not known at this stage.

The contribution of this paper is to analyze the existence of self-similarity in industrial SAT instances. The existence of a self-similar structure would mean that after *rescaling* (replacing groups of nodes by a single node, for example), we would observe the same kind of structure. It would also mean that the diameter d^{max} of the graph 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 or small-world graphs. Therefore, actions in some part of the graph (like variable instantiation) may not *propagate* to other parts as fast as in random graphs. Our analysis shows that many industrial formulas are self-similar. We think that the self-similarity, as well as the scale-free structure, is already present in many of the problems encoded as SAT instances. Thus, for instance, hardware-verification instances may have this structure because the circuits they encode already have this structure.

Studying graph properties of formulas has several direct applications. One of them, is the generation of industrial-like random SAT instances. Understanding the structure of industrial instances is a first step towards the development of random instance generators, reproducing the features of industrial instances. This would allow us to generate industrial-like random instances of a predefined size and structure to support the testing of industrial SAT solvers under development. Related work in this direction can be found in [3].

Another potential application is to improve portfolio approaches [14,6] which are solutions to the algorithm selection problem. State-of-the-art SAT Portfolios compute a set of features of SAT instances in order to select the best solver from a predefined set to be run on a particular SAT instance. It is reasonable to think that more informative structural features of SAT instances can help to improve portfolios.

Our experimental investigation shows that most industrial instances are self-similar, and their dimension ranges between 2 and 4. In the case of crafted instances, they also exhibit a clear self-similar behaviour, but their fractal dimensions are bigger in some cases. On the other hand, random instances are clearly not self-similar. We also show that using a very reduced set of complex networks properties we are able to classify industrial instances into families quite accurately.

Finally, we have investigated how the addition of learnt clauses during the execution of a SAT solver affects the dimension of the working instance. The addition of learnt clauses increases the fractal dimension, as expected. However, we show that modern SAT solvers produce a smooth increase, that suggests that SAT solvers tend to work locally. In contrast, the substitution of the learnt clauses by random clauses of the same size, produces a much bigger increase in the dimension.

The paper proceeds as follows. We introduce the fractal dimension of graphs in Section 2. In Section 3, we define the notion of fractal dimension of a SAT formula and compare it with the notion of diameter of a SAT formula. Then, we analyze whether SAT instances represented as graphs have a fractal dimension in Section 4. In Section 5, we study the effect of learnt clauses on the fractal dimension. Section 6 contains the conclusions.

2 Fractal Dimension of a Graph

We can define a notion of fractal dimension of a graph following the principle of self-similarity. We will use the definition of box covering by Hausdorff [8].

Definition 1. *The* **distance** *between two nodes is the minimum number of edges we need to follow to go from one node to the other.*
The **diameter** d^{max} *of a graph is the maximal distance between any two nodes of the graph.*
Given a graph G, a **box** B *of size l is a subset of nodes such that the distance between any pair of them is strictly smaller than l.*
We say that a set of boxes covers a graph, if every node of the graph is in some box. Let $N(l)$ be the minimum number of boxes of size l required to cover *the graph.*
We say that a graph has the **self-similarity** *property if the function $N(l)$ decreases polynomially, i.e. $N(l) \sim l^{-d}$, for some value d. In this case, we call d the* **dimension** *of the graph.*

Notice that $N(1)$ is equal to the number of nodes of G, and $N(d^{max}+1)$ is the number of connected components of the graph.

Lemma 1. *Computing the function $N(l)$ is NP-hard.* [4]

Proof: We prove that computing $N(2)$ is already NP-hard by reducing the graph coloring problem to the computation of $N(2)$. Given a graph G, let $\overline{G}$, the complement of G, be a graph with the same nodes, and where any pair of distinct nodes are connected in $\overline{G}$ iff they are not connected in G. Boxes of size 2 in $\overline{G}$ are cliques, thus they are sets of nodes of G without an edge between them. Therefore, the minimal number of colors needed to color G is equal to the minimal number of cliques needed to cover $\overline{G}$, i.e. $N(2)$. ■

[4] In [10] the same result is stated, but there, they prove the wrong reduction. They reduce the computation of $N(2)$ to the graph coloring problem.

There are several efficient algorithms that approximate $N(l)$. They compute upper bounds of $N(l)$. They are called *burning* algorithms (see [10]). Following a greedy strategy, at every step they try to select the box that covers (burns) the maximal number of uncovered (unburned) nodes. Although they are polynomial algorithms, we still need to do some further approximations to make the algorithms of practical use in very large graphs.

First, instead of boxes, we will use *circles*.

Definition 2. *A* **circle** *of* radius r *and* center c *is a subset of nodes of* G *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.*

Notice that any circle of radius r is inside of a box of size $2\,r-1$ (the opposite is in general false) and any box of size l is inside a circle of radius l (it does not matter what node of the box we use as center). Notice also that every radius r and center c characterizes a unique circle.

According to Hausdorff's dimension definition, $N(r) \sim r^{-d}$ also characterizes self-similar graphs of dimension d. We can approximate this fractal dimension using the *Maximum-Excluded-Mass-Burning (MEMB)* algorithm [10], which works as follows: Consider a graph G and a radius r. We compute an upper bound of the number of circles with radius r necessary to cover the graph $N(r)$. We start with all nodes set to unburned. At every step, for every possible node c, we compute the number of unburned nodes covered by the circle of center c and radius r, then select the node c that maximizes this number, and burn the new covered nodes.

The MEMB algorithm is still too costly for our purposes. We apply the following strategy to make the algorithm more efficient. We order the nodes according to their degree: $\langle c_1, \ldots, c_n \rangle$ such that $degree(c_i) \geq degree(c_j)$, when $i > j$. Now, for $i = 1$ to n, if c_i is not burned, then select the circle of center c_i and radius r (even if it does not maximizes the number of unburned covered nodes), and burn all its unburned nodes. We call this algorithm *Burning by Node Degree (BND)*, and describe it in Alg. 1. After we give the definition of fractal dimension of a SAT instance, we will compare the accuracy and efficiency of algorithms MEMB and BND in subsection 4.1 to justify the use of algorithm BND in our experimentation.

3 The Fractal Dimension of SAT Instances

Given a SAT instance, we can build a graph from it. Here, we propose two models.

Definition 3. *Given a SAT formula, the* **Clause-Variable Incidence Graph (CVIG)** *associated to it is a* bipartite *graph whose nodes are the set of variables and the set of clauses, and its edges connect a variable and a clause whenever that variable occurs in the clause.*

Algorithm 1: Burning by Node Degree (**BND**)

```
   Input: Graph G = (V, E)
   Output: vector[int] N
 1 N[1] := |V|;
 2 int i := 2;
 3 while N[i − 1] > connectedComponents(G) do
 4 |   vector[bool] burned(|V|);
 5 |   N[i] := 0;
 6 |   burned := {false, ..., false};
 7 |   while existsUnburnedNode(burned) do
 8 |   |   c := highestDegreeUnburnedNode(G, burned);
 9 |   |   S := circle(c, i); // circle with center c and radius i;
10 |   |   foreach x ∈ S do
11 |   |   |   burned[x] := true;
12 |   |   N[i] + +;
13 |   i := i+1;
```

The **Variable Incidence Graph (VIG)** *associated to a formula is a graph whose nodes represent the set of variables, and an edge between two nodes indicates the existence of a clause containing both variables.*

In this paper we analyze the function $N(r)$ for the graphs obtained from a SAT instance following the VIG and CVIG models. These two functions are denoted $N(r)$ and $N^b(r)$, respectively, and they relate to each other as follows.

Lemma 2. *If* $N(r) \sim r^{-d}$ *then* $N^b(r) \sim r^{-d}$.
If $N(r) \sim e^{-\beta r}$ *then* $N^b(r) \sim e^{-\frac{\beta}{2} r}$.

Proof: Notice that, for any formula, given a circle of radius r in the VIG model, using the same center and radius $2\,r-1$ we can cover the same variable nodes in the CVIG model. With radius $2\,r$ we can also cover all clauses adjacent to some covered variable. Hence $N^b(2\,r) \leq N(r)$.

Conversely, given a circle of radius $2\,r$ in the CVIG model, we consider two possibilities. If the center is a variable node, we cover the same variables in the VIG model using a circle of radius r and the same center. If the center is a clause c, to cover the same variables in the VIG model, we need a circle of radius $r+1$ centered in a variable node adjacent to c. Hence $N(r+1) \leq N^b(2\,r)$.

Therefore $N(r+1) \leq N^b(2\,r) \leq N(r)$, and $N(r) \sim N^b(2\,r)$. From this asymptotic relation, we can derive the two implications stated in the lemma. ■

Previous lemma states that if a SAT formula is (fully) self-similar, then in both models, VIG and CVIG, the fractal dimension is the same. In such case, if we plot $N(r)$ as a function of r in double-logarithmic axes, we obtain a line with slope $-d$. If $N(r)$ decays exponentially (as in random SAT formulas), then the decay factor in the CVIG model is half of the decay factor in the VIG model.

In such case, if we plot $N(r)$ in semi-logarithmic axes, we obtain a line with slope $-\beta$. We will always plot $N(r)$ in double-logarithmic axes. Thus, when $N(r)$ decays exponentially, we will observe a concave curve.

3.1 Fractal Dimension versus Diameter

The function $N(r)$ determines the *maximal radius* r^{max} of a connected graph, defined as the minimum radius of a circle covering the whole graph minus one: $N(r^{max}+1) = 1$. The maximal radius and the *diameter* d^{max} of a graph are also related, because $r^{max} \leq d^{max} \leq 2\,r^{max}$. From these relations we can conclude the following.

Lemma 3. *For self-similar graphs or SAT formulas (where $N(r) \sim r^{-d}$), the diameter is $d^{max} \approx n^{1/d}$, where d is the fractal dimension.*
In graphs or SAT formulas where $N(r) \sim e^{-\beta\,r}$, the diameter is $d^{max} \approx \frac{\log n}{\beta}$.

Proof: The diameter of a graph and the maximal radius are related as $r^{max} \leq d^{max} \leq 2\,r^{max}$. Notice that, by definition of the function $N(r)$, we have $N(1) = n$, where n is the number of nodes, and $N(r^{max}+1) = 1$.

Assuming $N(r) = C\,r^{-d}$ and replacing r by 1 we get $C = n$. Then, replacing r by $r^{max}+1$, we get $1 = N(r^{max}+1) = n\,(r^{max}+1)^{-d}$. Hence, $r^{max} = n^{1/d} - 1$.

Assuming $N(r) = C\,e^{-\beta\,r}$ and replacing r by 1 we get $C = n\,e^{\beta}$. Then, replacing r by $r^{max}+1$, we get $1 = N(r^{max}+1) = n\,e^{-\beta\,(r^{max})}$. Hence, $r^{max} = \frac{\log n}{\beta}$. ■

The diameter, as well as the *typical distance*[5] L of a graph, have been widely used in the characterization of graphs. For instance, *small world graphs* [12] are characterized as those graphs with a small typical distance $L \sim \log n$ and a large clustering coefficient. This definition works well for *families* of graphs because then we can quantify the typical distance as a function on the number of nodes. But it is quite imprecise in the case of individual graphs, because it is difficult to decide what is a "small" distance and a "large" clustering coefficient, for a concrete graph. Moreover, the diameter and the typical distance of a graph are measures quite expensive to compute in practice (for huge graphs, as the ones representing many industrial SAT formulas), even though there is a quadratic algorithm. In fact, our approximation to the fractal dimension can be computed more efficiently than the diameter.

Since we are interested in characterizing the structure of formulas, the fractal dimension is a better measure because it is independent of the size. Thus, formulas of the same family (and similar structure), but very distinct size, will have similar dimension and $N(r)$ function shape.

[5] The typical distance of a graph is the average of the distances between any two nodes.

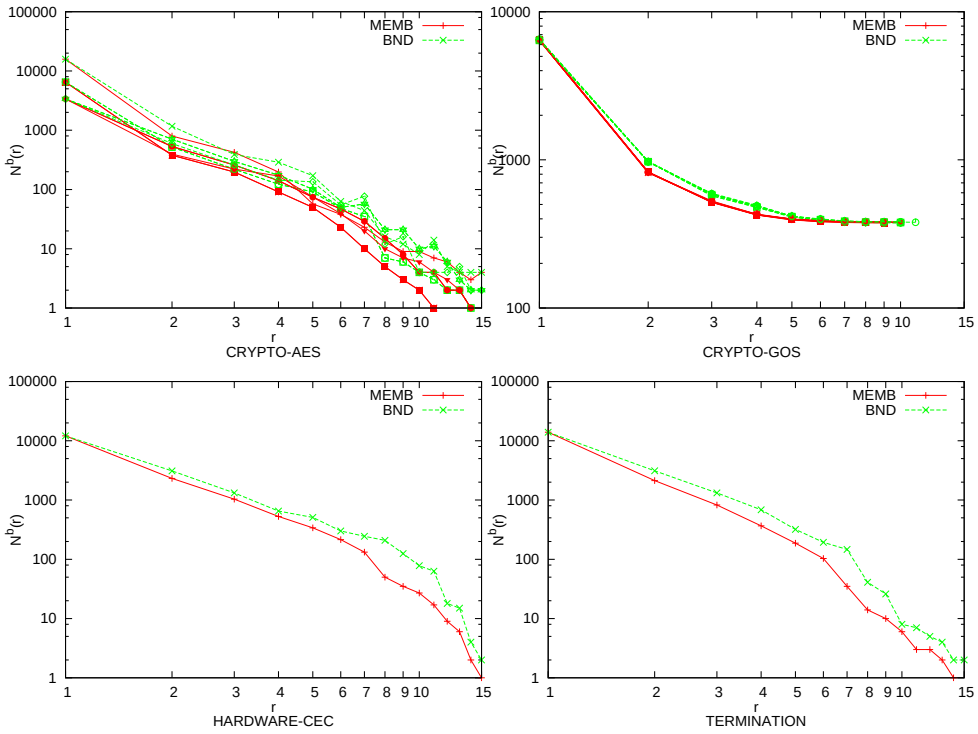

Fig. 1. Upper bounds for $N^b(r)$ obtained with MEMB and BND algorithms, for the 17 industrial instances that MEMB is able to compute in 30 minutes, grouped by families.

4 Experimental Evaluation

We have conducted an exhaustive analysis of the 300 industrial SAT instances and the 300 crafted instances of the SAT Competition 2013[6], and 90 random 3CNF formulas of 10^5 variables at different clause/variable ratios. We will see that most industrial and crafted instances are self-similar and have a small fractal dimension, i.e. $N(r) \sim r^{-d}$, for small d. In random instances $N(r)$ decays exponentially, i.e. $N(r) \sim e^{-\beta\, r}$.

Before presenting the results of this evaluation, let us justify the use of the BND algorithm to calculate the fractal dimension, instead of the MEMB algorithm.

4.1 The Accuracy of the BND Algorithm

In order to evaluate how accurate the algorithm BND is, we compare it to the MEMB algorithm presented in [10].

[6] `http://satcompetition.org/2013/`

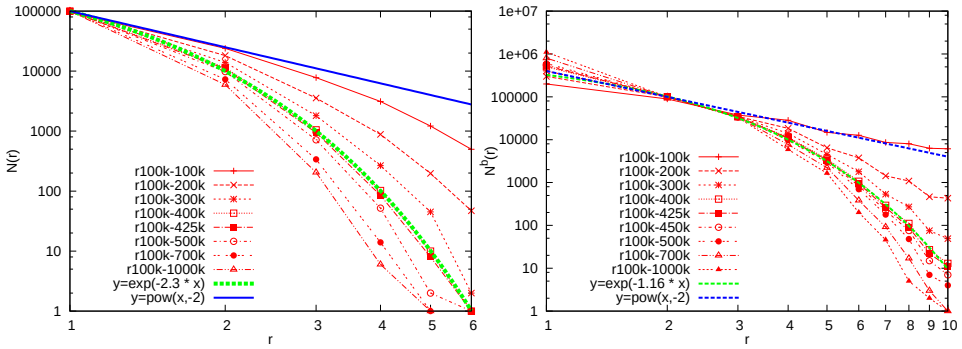

Fig. 2. Functions $N(r)$ for VIG (left), and $N^b(r)$ for CVIG (right), for some 3CNF random formulas with distinct values of the m/n fraction. Axes are both logarithmic, and "r100k-425k" indicates $n = 10^5$ variables and $m = 4.25 \cdot 10^5$ clauses.

We run both algorithms for the set of 300 industrial instances of the SAT Competition 2013 with a timeout of 30 minutes. While the BND algorithm finishes for all the 300 instances, MEMB is only able to approximate $N^b(r)$ in 17 instances. Moreover, while the average run-time of BND for these instances is 0.11 seconds, MEMB takes an average of 10 minutes and 7.2 seconds to compute them. On the other hand, the approximations of $N^b(r)$ computed by MEMB and BND are very similar (see Fig. 1).

Since the MEMB algorithm is more accurate than the BND algorithm, the upper bounds of $N^b(r)$ that MEMB calculates are below the ones calculated by BND. The real values of $N^b(r)$ are probably even lower in the final points (where the approximation is less accurate).

4.2 Random Formulas

Random 2SAT formulas in the VIG model correspond to Endös-Renyi graphs. It is known that these formulas have a phase transition point at $m/n = 1$ where formulas pass from satisfiable to unsatisfiable with probability one. It is also known that at $m/n = 0.5$ there is a percolation threshold. Formulas below this point have an non-connected associated VIG graph, and above this threshold there is a major connected component. In the percolation point the formula is self-similar with a fractal dimension $d = 2$. Above this point $N(r)$ decays exponentially. To the best of our knowledge, a result of this kind is not known for random 3CNF formulas.

Experimentally, we observe that the function $N(r)$ only depends on the clause/variable ratio m/n, and not on the number of variables (this is not shown in figures). In the phase transition point $m/n = 4.25$, the function has the form $N(r) \sim e^{-2.3\,r}$, i.e. it decays exponentially with $\beta = 2.3$ (see Fig. 2). Hence, $r^{max} = \frac{\log n}{2.3} + 1$. For instance, for $n = 10^5$ variables, random formulas have a radius $r^{max} \approx 6$. For bigger values of m/n, the decay β is bigger. In the CVIG

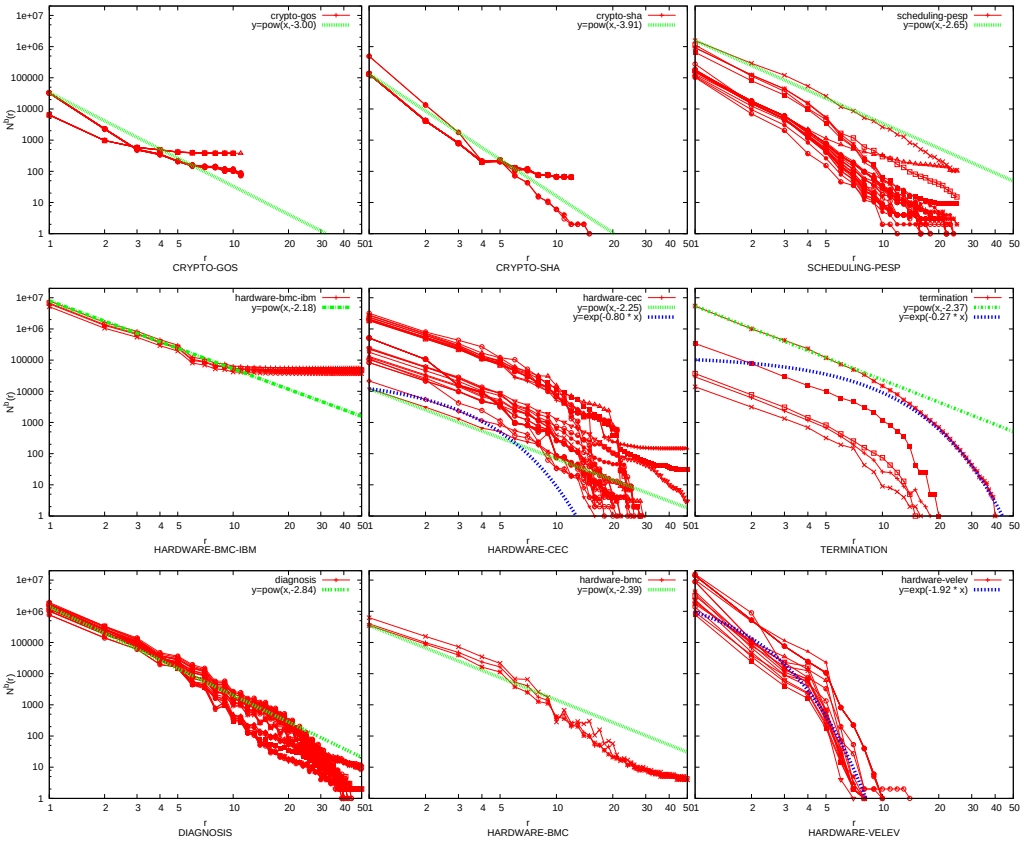

Fig. 3. Function $N^b(r)$ for some industrial SAT formulas grouped by families.

model, we observe the same behavior. However, in this case, in the phase transition point, $N(r)$ decays exponentially with $\beta = 1.16 \approx 2.3/2$. Hence, the decay is just half of the decay of the VIG model, as we expected by Lemma 2.

For random 3CNF formulas, we have experimentally found a percolation threshold at $m/n \approx 0.17$. At this point the principal connected component also exhibits a fractal dimension $d = 2$.

4.3 Industrial Instances

Analyzing industrial instances, we observe that most of them are self-similar, and most dimensions ranges between 2 and 4. In the SAT Competition 2013, instances are grouped into families. In many of these families, all instances have the same fractal dimension, being this dimension a characteristic of the family. See, for instance, families *crypto-sha* or *diagnosis* in Fig. 3. Notice that the size of the formulas does not affect the value of the dimension (in the representation the function can be higher or lower, but with the same slope).

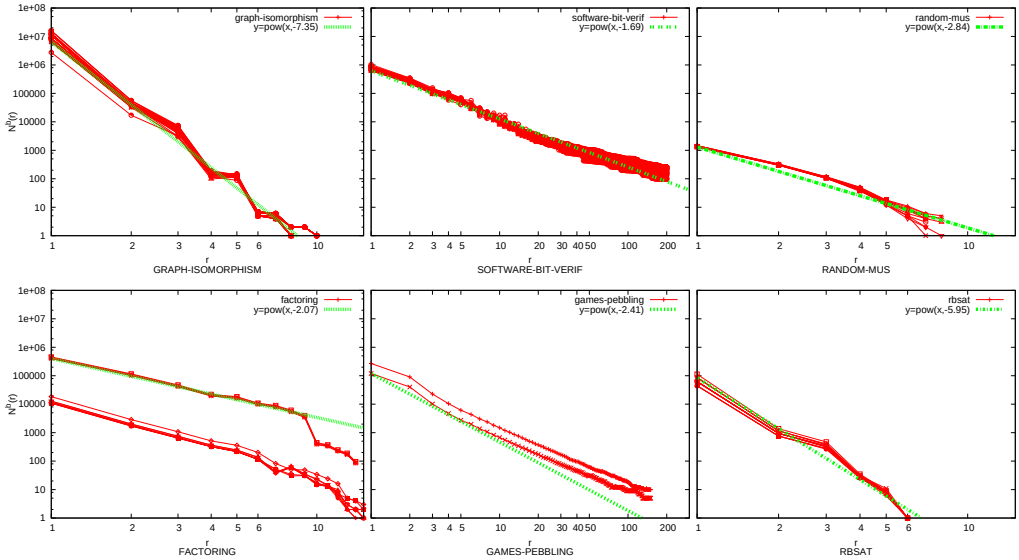

Fig. 4. Function $N^b(r)$ for some crafted SAT formulas grouped by families.

In general, the polynomial decay is clearer for small values of r. Moreover, in this area, the slope is the same for all instances of the same family of formulas.

For big values of r, we must make some considerations. First, the upper bound on $N^b(r)$ that we calculate can be a bad approximation. Second, there are two phenomena that we can identify. In some cases there is an abrupt decay, but the whole function can not be approximated by an exponential function (see some *hardware-cec* or *termination* instances, for instance). This decay in the number of required tiles can be due to a small number of edges connecting distant areas of the graph. These edges have no effect for small values of r, but may drop down the number of tiles for big values of r. In some other cases (see *hardware-bmc-ibm*, for instance), there is a long tail. In this case, it is due to the existence of (small) unconnected components in the graph. If we compute $N(r)$ only for the major component, this tail disappears.[7]

Finally, all instances of the *hardware-velev* family have a $N(r)$ function with exponential decay, i.e. are not self-similar.

4.4 Crafted Instances

Studying crafted instances, we see that most of them are self-similar. However, their fractal dimension have bigger values than the ones of the industrial formulas (some values are even bigger than 7).

[7] In the figures, we can subtract from $N(r)$ the number of unconnected components, as an approximation, since most are covered with a few tiles.

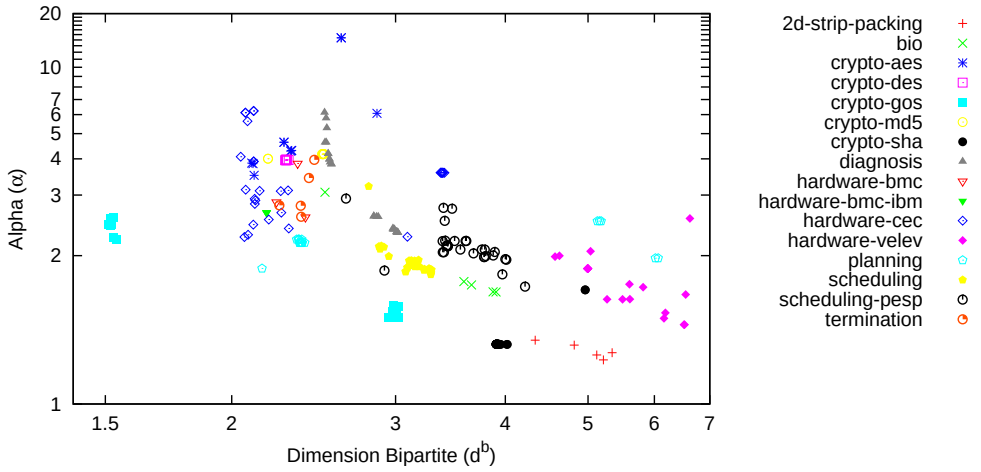

Fig. 5. Distribution of families according to the exponent α of the powerlaw distribution of node degrees, and fractal dimensions d^b at a fine-grained scale. Heterogeneous families (*software-bit-verif* and *software-bmc*) are not plotted.

The crafted instances of the SAT Competition 2013, as well as the industrial instances of this competition, are grouped into families. In general, we find that many families exhibit an homogeneus curve of $N(r)$ in all their instances. Moreover, in many of these families, $N(r)$ has a polynomial decay (i.e., all instances have the same fractal dimension). The fractal dimension of crafted formulas ranges from 1.5 to 7.5. In Fig. 4 we represent some crafted families.

4.5 Fractal Dimension at Fine-Grained Scale

If a graph is self-similar, then it has the same structure at all scales. We could replace groups of nodes tiled by a box by a single node, obtaining another graph with the same structure. In our experiments, we observe that this is the case for small values of r (for small values of r, function $N(r) \approx C\, r^{-d}$). However, this is more arguable for big values of r. Perhaps this is because the graph is not self-similar at large scale (coarse-grained), or because our approximation of $N(r)$ is not precise enough. If the formula has a small refutation, this will be visualized in our VIG or CVIG graphs as a small cycle. This means that what is really relevant is the fractal dimension looking at the graph at small scale (fine-grained dimension). In other words, we think that, more than whether there exists a self-similar structure, what is important, is the value of the fractal dimension at fine-grained, i.e. the slope of the function $N(r)$ for small values of r.

In our next experiment, we try to classify industrial instances according to their fractal dimension at fine-grained, and the exponent α of the powerlaw

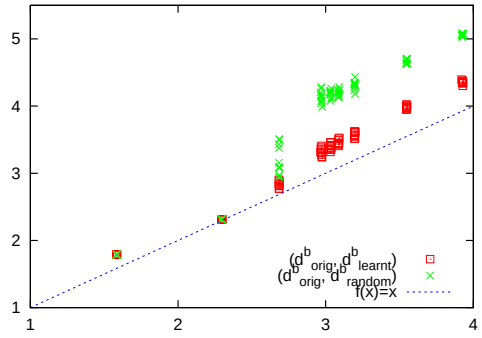

Fig. 6. Relation between the original fractal dimension d^b_{orig}, and the dimension d^b_{learnt} after adding learned clauses, or after adding random clauses d^b_{rand}, in random 3CNF formulas. Learnt clauses are computed after 10^3 conflicts.

distribution of node degrees (see [2] for a description of how to compute exponent α). We will also note these fine-grained dimensions as d and d^b for the VIG and CVIG, respectively. We compute them as the interpolation, by linear regression, of $\log N(r)$ vs. $\log r$. We use the values of $N(r)$ and $N^b(r)$, for $r = 1, \ldots, 6$. Experimentally, we see that these approximations are accurate enough. As we can see in Fig. 5, just with the fractal dimension d^b and the powerlaw exponent α, we are able to determine which family an instance belongs to.

5 The Effect of Learning

State-of-the-art SAT solvers add learnt clauses from conflicts during their execution. When a learnt clause is unitary, it can be propagated simplifying the original formula. Given a unitary clause x, clauses with literal x are completely removed, and literals $\neg x$ are removed from the formula. Learnt clauses of bigger length create new relations between variables, i.e., new edges in the VIG model.

Both, the addition of learnt clauses, and the simplification of formulas, due to unitary learnt clauses, may affect the dimension of the formula. The addition of edges in a graph (preserving the nodes) always increases its dimension, because tiles may cover more nodes, and the number $N(r)$ of tiles required to cover the graph decreases, whereas $N(1)$ is preserved. This contributes to increase the slope of function $N(r)$, hence the dimension. The effect of simplifications due to unitary learnt clauses is more difficult to predict, since we remove satisfied clauses (edges in the VIG model), but also nodes (decreasing $N(1)$).

We have conducted some experiments to analyze how the fractal dimension evolves during the execution of the SAT solver. First we show the effect of introducing learnt clauses in random 3CNF instances with 10^5 variables and distinct clause/variable ratios. In these instances almost all learnt clauses are not unitary, hence we do not remove variable nodes. In the VIG model, the addition

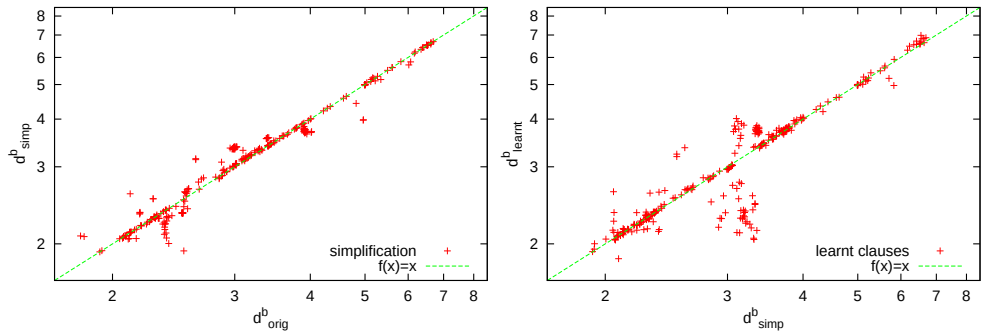

Fig. 7. Relation between the original fractal dimension d^b_{orig} and the fractal dimension d^b_{simp} after simplifying the formula with the unitary learnt clauses (left), and relation between the fractal dimension d^b_{simp} and the fractal dimension d^b_{learnt} after simplification and adding learnt clauses (right), for all industrial formulas. Learnt clauses are the result of 10^3 conflicts.

of these learnt clauses introduces edges, and increases the dimension. In the CVIG model dimension also increases due to the same reason. In Fig. 6, we plot the dimension d^b_{learnt} after adding learnt clauses w.r.t. the original dimension d^b_{orig}. We observe that the addition of learnt clauses increases the dimension of the formula. This increase is bigger for formulas with higher clause/variable ratio. In order to *quantify* the increase in the dimension, we repeat the same experiment replacing learnt clauses by random clauses of the same size, and computing the new dimension d^b_{random} (results are also shown in Fig. 6). We observe that in this second experiment the increase in the dimension is bigger than adding learnt clauses: $d^b_{random} \geq d^b_{learnt} \geq d^b_{orig}$. This means that learnt clauses, even in these random formulas, tend to connect variables that were already *close* in the graph. Therefore, their effect in the dimension is not as important as adding random clauses. In industrial instances some of the learnt clauses are unitary. We have analyzed separately the effect of simplifying the formula using these unitary clauses, and the effect of adding non-unitary learnt clauses. In the first case, when we learn x, and remove satisfied clauses containing x, we may remove edges connecting pairs of variables of those clauses. This contributes to decrease the dimension. However, we also remove the variable node x and the clauses nodes satisfied by x ($N(1)$ decreases). The effect of this second transformation on the graph cannot be predicted. Experimentally, we observe that simplifying the formula using unitary learnt clauses tends to decrease the dimension of the VIG and CVIG graph (see Fig. 7). The only exceptions are the *crypto-sha* and the *crypto-gos* families where a great number of variable nodes are removed.

In Fig. 8 we show the change in the dimension after 10^3, 10^4 and 10^5 conflicts. We observe that at the beginning dimensions may increase or decrease slightly. However, after 10^5 conflicts, the dimension clearly increases in most of the cases. Finally, in Fig. 9 we *quantify* the variation of the dimension due to the addition of

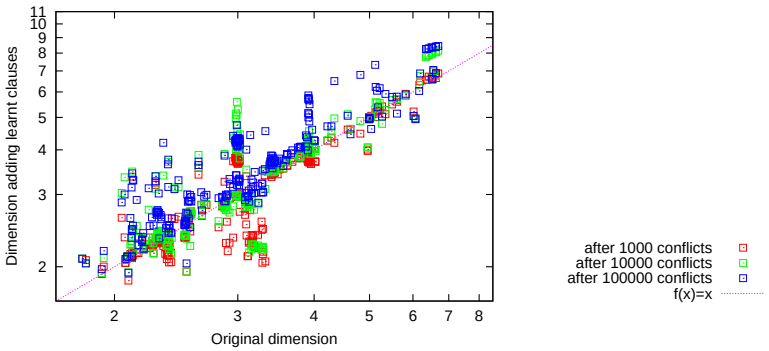

Fig. 8. Relation between the original fractal dimension and the fractal dimensions after learning clauses, in industrial formulas.

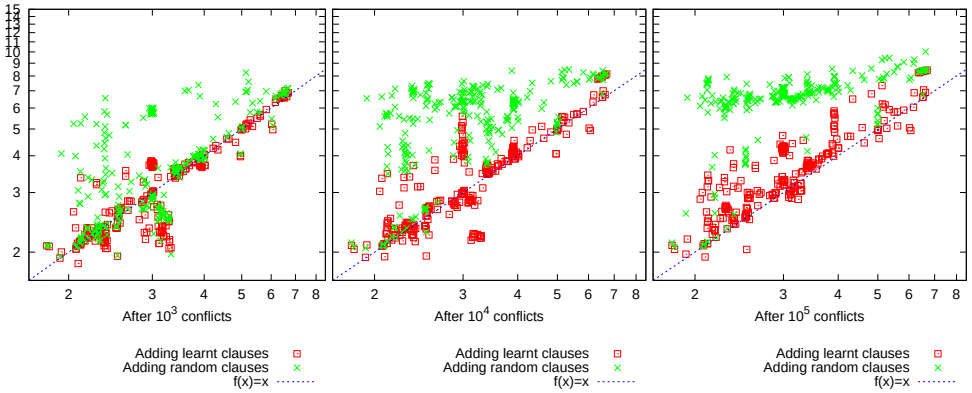

Fig. 9. Relation between the original fractal dimension and the fractal dimensions after adding learnt clauses, or after adding random clauses, in industrial formulas.

learnt clauses, compared with the addition of the same number of random clauses with the same sizes. The effect of random clauses is much more significant, i.e., most of learnt clauses do not contribute to make tiles bigger (i.e. to reduce the number of needed tiles). They mainly connect nodes inside the tiles, i.e. nodes that where already close. Therefore, learning acts quite locally in the formula.

6 Conclusions

We conclude that many industrial instances are self-similar, with most fractal dimensions ranging between 2 and 4. Fractal dimension, typical distances and graph diameter are related (small dimension implies big distance and diameter). Hence, industrial SAT instances have a big diameter (intuitively, we need long chains of implications to propagate a variable instantiation to others). We ob-

serve the same behaviour in crafted instances, although the fractal dimension is bigger in some cases. On the other hand, random instances are not self-similar.

We have also observed that fractal dimension increases due to learnt clauses. Moreover, the increase is specially abrupt in instances that show exponential decays (for instance, in the family *hardware-velev* or random formulas). This increase is bigger, if we substitute learnt clauses by random clauses of the same size. Therefore, learning *does not* contribute very much to connect distant parts of the formula, as one could think.

We have proved that we can determine the family an industrial instance belongs according to their fractal dimension at fine-grained, and the exponent α of the powerlaw distribution of node degrees. This is of interest for the development of portfolio solvers.

As future work, we plan to investigate how to develop industrial-like random instance generators to produce instances whose structural graph features such as the fractal dimension, the α exponent or the modularity are similar to the ones of industrial instances.

References

1. R. Albert, H. Jeong, and A.-L. Barabási. The diameter of the WWW. *Nature*, 401:130–131, 1999.
2. C. Ansótegui, M. L. Bonet, and J. Levy. On the structure of industrial SAT instances. In *CP'09*, volume 5732 of *LNCS*, pages 127–141. Springer, 2009.
3. C. Ansótegui, M. L. Bonet, and J. Levy. Towards industrial-like random SAT instances. In *IJCAI'09*, pages 387–392, 2009.
4. C. Ansótegui, J. Giráldez-Cru, and J. Levy. The community structure of SAT formulas. In *SAT'12*, volume 7317 of *LNCS*, pages 410–423. Springer, 2012.
5. A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
6. S. Kadioglu, Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann. Algorithm selection and scheduling. In *CP'11*, pages 454–469, 2011.
7. G. Katsirelos and L. Simon. Eigenvector centrality in industrial SAT instances. In *CP'12*, volume 7514 of *LNCS*, pages 348–356. Springer, 2012.
8. B. B. Mandelbrot. *The fractal geometry of nature*. Macmillan, 1983.
9. F. Papadopoulos, M. Kitsak, M. Serrano, M. Bogu, and D. Krioukov. Popularity versus similarity in growing networks. *Nature*, 489:537–540, 2012.
10. C. Song, L. K. Gallos, S. Havlin, and H. A. Makse. How to calculate the fractal dimension of a complex network: the box covering algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(03):P03006, 2007.
11. C. Song, S. Havlin, and H. A. Makse. Self-similarity of complex networks. *Nature*, 433:392–395, 2005.
12. T. Walsh. Search in a small world. In *IJCAI'99*, pages 1172–1177, 1999.
13. R. Williams, C. P. Gomes, and B. Selman. Backdoors to typical case complexity. In *IJCAI'03*, pages 1173–1178, 2003.
14. L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *J. Artif. Int. Res.*, 32(1):565–606, 2008.