On the Classification of Industrial SAT Families

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

(1) Universitat de Lleida (UdL)

(2) Universitat Politècnica de Catalunya (UPC)

(3) **Institut d'Investigació en Intel·ligència Artificial, Consejo Superior de Investigaciones Científicas** (**IIIA-CSIC**)

CCIA 2015
October 22nd, 2015

The Boolean Satisfiability Problem (SAT)

- The **Boolean Satisfiability Problem (SAT)** is the decision problem of determining if there exists an assignment of the Boolean variables of a propositional formulas such that the formula is evaluated as `TRUE`.

- **Notation**:
  - A **literal** is a Boolean variable x, or its negation $\neg x$.
  - A **clause** c is a disjunction of i literals:
    $$c = x_1 \vee x_2 \vee \cdots \vee x_i$$
  - A **CNF** formula Γ is a conjunction of m clauses:
    $$\Gamma = c_1 \wedge c_2 \wedge \cdots \wedge c_m$$
  - A k-**CNF** formula is a CNF with all its clauses having size k.

Applications of SAT

- First known **NP-complete** problem [Cook71].
  - Deeply studied from a theoretical point of view.

- Extensively used to encode many other problems, which are **efficiently solved** by modern SAT solvers.
  - Planning, Scheduling, Formal Verification (hardware and software), cryptography, bioinformatics, ...
  - Therefore, finding good algorithms to solve SAT is of practical use in many areas of Computer Science.

SAT Instances

- **Random** instances:
  - Randomly generated from a well-known model: $F_k(n, m)$.

- **Industrial** (application) instances:
  - Problems encodings from real-world applications.
  - No precise definition/model: crypto, bmc, scheduling, planning, ...
  - Heterogeneity.

  - Usually classified into **families**.

SAT Solvers Performance

- **Random** and **industrial** formulas: **distinct nature**.
  - SAT competitions: different tracks.

- Distinct algorithms have a very **different performance** depending on the type of instance they are solving (i.e., random or industrial).

- **Improvements** on those algorithms make **SAT solvers specialize**.

Performance on Industrial SAT Instances

- **Conflict-Driven Clause Learning** (**CDCL**) are the **dominant** technique **to solve industrial SAT** instances.
  - Backtracking-like depth-first search, Unit Prop. (**DPLL**).
  - **Clause Learning**.
  - Conflict-driven heuristics.
  - Rapid restarts.
  - Lazy data structures, inprocessing variable elimination techniques, clause removal policies, ...

- However, **why** these techniques are **so efficient** solving this type of problems **remains open**.

- The common wisdom is that they exploit some **hidden structure** that actually exists in industrial formulas.

- What is **exactly** this **structure** of **industrial** SAT instances?

The Structure of Industrial SAT Instances (I)

- Inspired by works on **complex networks**.

- We represent SAT instances as **graphs**:
  - **Variable Incidence Graph** (**VIG**):
    - Nodes are variables.
    - Edges between two variables appearing in the same clause.
    - Weights in the edges to represent the length of the clause.
  - **Clause-Variable Incidence Graph** (**CVIG**):
    - Nodes are either variables or clause (bi-partite graph).
    - Edges represent the existence of a variable in a clause.
    - Weights in the edges to represent the length of the clause.

The Structure of Industrial SAT Instances (II)

- The **Scale-free Structure**: [AnsóteguiBonetLevy.CP09].
  - A graph has scale-free structure if the degree of its nodes follows a **power-law distribution** $p(k) \sim k^{-\alpha}$.
  - The number of **variable occurrences** (number of neighbors of variable nodes in the CVIG) follows a power-law distribution in **most industrial SAT instances**: exponent α_v.
- The **Community Structure**: [AnsóteguiGiráldez-CruLevy.SAT12].
- The **Self-Similar Structure**: [AnsóteguiBonetGiráldez-CruLevy.IJCAR14].

The Structure of Industrial SAT Instances (II)

- The **Scale-free Structure**: [AnsóteguiBonetLevy.CP09].
- The **Community Structure**: [AnsóteguiGiráldez-CruLevy.SAT12].
  - A graph has clear community structure if there exists a partition of its nodes into communities such that most edges connect nodes of the **same community**.
  - The **modularity** Q measures how clear this community structure is (value in $[0, 1]$).
  - **Most industrial SAT instances** have a clear community structure: Q (of its VIG).
- The **Self-Similar Structure**: [AnsóteguiBonetGiráldez-CruLevy.IJCAR14].

The Structure of Industrial SAT Instances (II)

- The **Scale-free Structure**: [AnsóteguiBonetLevy.CP09].
- The **Community Structure**: [AnsóteguiGiráldez-CruLevy.SAT12].
- The **Self-Similar Structure**: [AnsóteguiBonetGiráldez-CruLevy.IJCAR14].
  - A graph has self-similar structure if it keeps the same shape after **rescaling** (replacing groups of nodes by a single node).
  - This means that the diameter d grows as $d \sim n^{1/d}$, where n is the number of nodes, and d is its **fractal dimension**.
  - **Most industrial SAT instances** have fractal dimension: d and d^b (of its VIG and CVIG, respectively).

Algorithm Configurations

- Algorithms have **configurations** of their parameters that may vary their performance.

- What is the **best parameter configuration** to solve a particular instance?

- For instance, some industrial families are known to be *easier* when using strategies based on **diversification** (i.e., *high frequency of restarts*). Others *prefer* an **intensified** search (i.e., *low frequency of restarts*).

Portfolio Algorithms

- What is the **best algorithm** to solve a particular instance?
- The **Algorithm Selection Problem**: choose the best algorithm from a **predefined set** to solve a particular instance of a problem, using a **prediction model**.
- **Portfolio SAT solvers**:
  - Building the **predictor**:
    - Set of (core) SAT solvers: vector of *runtimes*.
    - Training set (of instances): vector of *features*.
  - **Solving** a particular instance:
    - Compute the features of such instance.
    - Use the predictor to choose the (expected) best algorithm.
  - How to choose these **features**?

SATzilla (2012)

- A very **well-known** portfolio SAT solver.
- **Winner** in several tracks of several SAT competitions.

- *Prediction model*: **Random forest**.

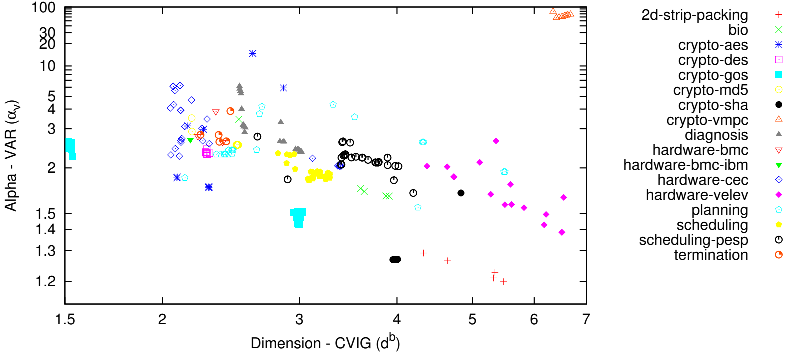
- *Features*: **127**, divided into categories.
  - Problem size.
  - Graph: degrees and clustering coefficients in VIG and CVIG.
  - Hardness: DPLL, LP, SLS, CDCL, SP.
  - Timing.

- The **Scale-free Structure**: α_v.

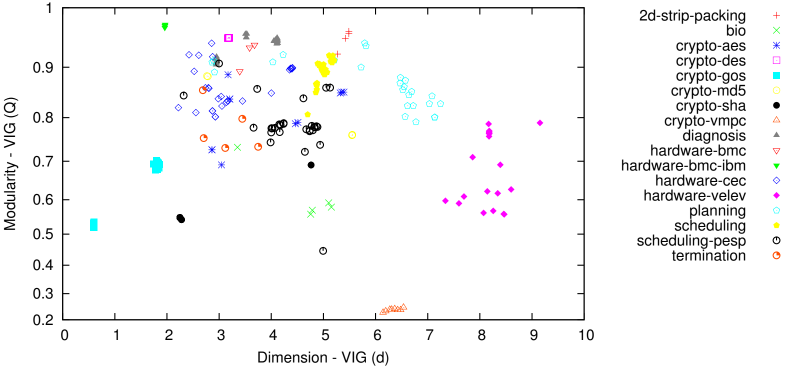
- The **Community Structure**: Q.

- The **Self-Similar Structure**: d and d^b.

Distribution of Industrial SAT Families



Distribution of Industrial SAT Families



Industrial SAT Families Classifiers (I)

- **SAT Competition 2013**:
  - 300 instances.
  - 19 industrial families.
  - SAT solvers submitted to this competition.
- Set of **features**:
  - **Structure** (5): α_v, Q, d, d^b, m/n.
  - **SATzilla** (115): timing features not considered.
- **Evaluation**:
  - **Classifiers**: **C4.5**, Random Forest (**RF**), Naïve Bayes (**NB**), Multi-response Linear Regression (**MLR**), Logistic Regression (**LR**), Sequential Minimal Optimization (**SMO**), **IBk**, **K***, **JRip**.
  - 10-folds cross-validation.

Industrial SAT Families Classifiers (II)

	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%)

In **bold**, classifiers with effectiveness **higher than 90%**.

Discussion

- **SATzilla** characterize the **structure** using 14 features (statistics of node degree and clustering coefficient).
- They are *local* properties of the structure.
- **We** use **4 global features** to characterize the structure.
  - **Statistics** of clustering coefficient **vs Modularity**.
  - **Statistics** of node degrees **vs Power-law exponent**.

- **SATzilla** characterize the **hardness** using 71 features.
- **We** only use m/n as a simple but **weak** metric of such hardness.

Evaluation in a Portfolio (I)

- Portfolio solver: **ISAC**.
  - Prediction: *g-means* (similar to *k-means*).
- SAT instances: **SAT Competition 2013**.
- Core solvers: **SAT Competition 2013**.
- Evaluation: **10-folds cross-validation**.
  - Each instance is randomly assigned to a fold.
  - 9 folds are used to build a classifier (**training set**). Instances of the remaining fold (**test set**) are solved using such classifier.
  - **Cross-validation**: All folds are used as test set once (repeat 10 times).
- **Runtime**: computing its **features** + the **runtime of the solver** selected by the classifier.

Cost of Computing the Set of Features

runtimes	Structure VIG+CVIG	Structure VIG	SATzilla
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

- **VIG**: α_v, Q, d (and m/n).
- **VIG+CVIG**: α_v, Q, d, d^b (and m/n).

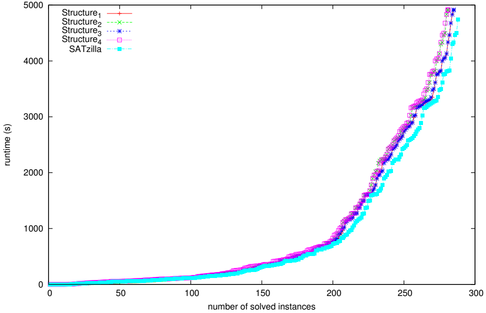
Evaluation in a Portfolio (II)

- **Sets** of features:

Structure$_1$	α_v	Q	d	d^b	m/n	VIG+CVIG+m/n
Structure$_2$	α_v	Q	d		m/n	VIG+m/n
Structure$_3$	α_v	Q	d	d^b		VIG+CVIG
Structure$_4$	α_v	Q	d			VIG
SATzilla	115 features					no timing features

- **Timeout**: 5000 seconds (timeout used in competitions).

Evaluation in a Portfolio (III)



	#solved	runtime max	runtime mean	runtime stdev	
Structure$_1$	**285**	4913.2	874.6	1199.6	VIG+CVIG+m/n
Structure$_2$	281	4913.2	863.3	1197.7	VIG $+m/n$
Structure$_3$	**285**	4913.2	876.8	1200.1	VIG+CVIG
Structure$_4$	281	4913.2	881.2	1207.8	VIG
SATzilla	**288**	**4745.2**	**831.6**	**1143.6**	

Conclusions

- A **good characterization** of SAT instances requires **structure** and **hardness**.

- Computing the **structure** set of features is **faster** than computing the **SATzilla** set.
- However, the clause/variable ratio m/n is a **too weak** metric of the hardness.
- Therefore, the **performance** using **SATzilla** is **slightly better** (**3** instances solved more) and **faster** than using any combination of structure features.

- *Future work*: **combine** the **hardness** features of SATzilla with the **structure** features of our method.

Thank you
for your attention!