Exploring the Computational Complexity of SAT Counting and Uniform Sampling with Phase Transitions

Olivier Zeyen
University of Luxembourg, SnT
Luxembourg

Maxime Cordy
University of Luxembourg, SnT
Luxembourg

Gilles Perrouin
NaDI, University of Namur
Belgium

Mathieu Acher
Univ Rennes, Inria, CNRS, IRISA
France

ACM Reference Format:
Olivier Zeyen, Maxime Cordy, Gilles Perrouin, and Mathieu Acher. 2024. Exploring the Computational Complexity of SAT Counting and Uniform Sampling with Phase Transitions. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3639478.3643097

1 PROBLEM

Uniform Random Sampling (URS) is the problem of selecting solutions (models) from a Boolean formula such that each solution gets the same probability of being selected. URS has many applications. In large configurable software systems, one wants an unbiased sample of configurations to look for bugs at an affordable cost [12, 13]. Other applications of URS include deep learning verification (to sample inputs from unknown distributions) [2] and evolutionary algorithms (to initialize the input population) [4].

Model counting (#SAT) – the problem of counting the number of solutions of a Boolean formula – is closely related to URS. These two problems generally rely on the same principles and heuristics to be solved; URS can also be reduced to #SAT [11]. Beyond URS, #SAT has many applications in (configurable) software engineering, such as variability reduction [15], variability evolution and analysis [7, 16], feature prioritization [15] and bug fix prioritization [8]. URS and #SAT are both challenging to solve efficiently: existing solutions hardly scale to real-world formulas [13]. Unlike the traditional problem of satisfiability solving (SAT), the reasons behind the complexity of URS and #SAT have been underexplored.

Problem: This poster addresses the understanding of the computational complexity of URS and #SAT to determine whether or not a formula is computationally prohibitive. Similarly to research efforts made for the conventional SAT problem in the past 30+ years [9, 10], our objective is to unveil these factors by conducting systematic analyses.

2 METHODS

Phase transitions are sudden changes in the properties of a subject observed after a small variation of a parameter. For SAT problems, Mitchell *et al.* [9] have shown that phase transitions occur when the clause-to-variable ratio of the formula reaches around 4.25. In other words, they observed that the time required for a SAT solver to process a formula suddenly changes when approaching this ratio. This discovery has later guided research in the development of new algorithms [3, 5, 14] specifically focused on instances lying at the phase transition. The analysis of phase transitions for SAT thus has had considerable theoretical and practical importance [6].

We contribute to a principled understanding of URS and #SAT complexity by studying whether phase transitions also occur in these problems. Our investigations require both experimental analysis (based on controlled experiments and artifacts) and empirical analysis (uncontrolled observations made on existing practices). Indeed, while empirical observations on real-world formulas are needed to validate conclusions in practice, the low availability and high heterogeneity of these formulas entail an insufficient coverage of all possible structural variations to draw general conclusions. Therefore, we also conduct experiments on synthetic formulas created through systematic and controlled procedures to identify general trends and limit cases. Doing so allows us to explore the role of specific characteristics in the complexity of URS and #SAT.

Thus, we first study experimentally the complexity of model counters and uniform random samplers by generating k-CNF formulas, i.e., random formulas in CNF form where every clause has exactly k literals. These formulas are generated by randomly choosing k variables, which are then negated with probability $\frac{1}{2}$. The randomly chosen variables form a clause. We repeat the process to generate the desired number of clauses. We control the generation to cover different values of clause-to-variable ratio – the structural characteristics previously used to reveal phase transitions in SAT problems [9].

Second, we explore the reasons behind the phase transitions. The phase transition for SAT solving has been explained by the fact that there is a sudden change in the probability of a generated formula being satisfiable. Because #SAT and URS techniques do not rely on the same principles as classical SAT (e.g. SAT aims to find one solution to the formula, whereas #SAT/URS have to explore all solutions), we look for another explanation for phase transitions in #SAT and URS. Instead of exploring the clause-to-variable ratio, we explore the ratio $r = log_2(|R_F|)/|Var(F)|$ which

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0502-1/24/04.
https://doi.org/10.1145/3639478.3643097

is the logarithm base 2 of the number of solutions divided by the number of variables in the studied formula F. Thus, r represents the proportion of variables that are theoretically necessary to encode the solutions of the formula.

Finally, we empirically analyze real-world formulas to check whether the general trends observed on synthetic data are confirmed. A key difference is that real-world formulas have a heterogeneous structure, e.g., they mix clauses of different sizes. This heterogeneity impedes drawing general conclusions from the sample we use. To limit the influence of this heterogeneity, we first preprocess the formulas by doing a boolean constraint propagation. We continue by eliminating subsumed clauses (i.e. clauses that are supersets of other clauses in the CNF formula). Finally, we remove variables that are either unconstrained (i.e. do not appear in any clause) or constants (i.e., variables appearing in clauses containing only one literal). We then compute the clause-to-variable ratio.

Methods. We use *phase transitions* [9, 10] as tool to understand URS and #SAT complexity. Because of URS, #SAT and SAT theoretical differences, we offer the r ratio as a clause-to-variable replacement. Our methods mix *experimental analysis* (based on controlled experiments and artifacts) on curated synthetic data and *empirical analysis* (uncontrolled observations made on existing practices) on real-world formulas.

3 RESULTS

By setting $k = 3$, we observe that phase transitions indeed occur for both URS and #SAT, though at a different ratio than for SAT (2.00 versus 4.25). This observation persists regardless of the modularity of the formulas, where modularity is a measure of the extent to which formula clauses are independent [1]. This is interesting as Ansótegui *et al.* have shown that real-world formulas have a high modularity. However, we find that modularity does affect the amplitude of the phase transitions – a lower modularity means a greater increase in computation time.

We generate random k-CNF formulas with $k = 3$ and $k = 4$. We find that the phase transition for $k = 4$ moves regarding the clause-to-variable ratio but is more stable for r. We thus uncover that the complexity is related to the number of models of the formula relative to its number of variables.

Finally, our observations on 503 formulas indicate that phase transitions also occur in real-world formulas as the hard instances all have a clause-to-variable ratio higher than 2.

Results. We observe phase transitions on 3-CNF synthetic formulas at a ratio of 2.00 (instead of 4.25 for SAT problems) independently of modularity. Modularity affects phase transition amplitude. Hard real-world formulas have a higher clause-to-variable ratio.

ACKNOWLEDGEMENT

Maxime Cordy and Olivier Zeyen are supported by FNR Luxembourg (grants C19/IS/13566661/BEEHIVE/Cordy and AFR Grant 17047437). Gilles Perrouin is an FNRS Research Associate.

REFERENCES

[1] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. 2012. The Community Structure of SAT Formulas. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 410–423.

[2] Teodora Baluta, Zheng Leong Chua, Kuldeep S. Meel, and Prateek Saxena. 2021. Scalable Quantitative Verification For Deep Neural Networks. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 312–323. https://doi.org/10.1109/ICSE43902.2021.00039

[3] Shaowei Cai, Chuan Luo, and Kaile Su. 2014. Scoring Functions Based on Second Level Score for k-SAT with Long Clauses. *J. Artif. Intell. Res.* 51 (2014), 413–441. https://api.semanticscholar.org/CorpusID:16385185

[4] Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. 2015. Money for Nothing: Speeding Up Evolutionary Algorithms Through Better Initialization. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO '15)*. ACM, New York, NY, USA, 815–822. https://doi.org/10.1145/2739480.2754760

[5] Jian Gao, Ruizhi Li, and Minghao Yin. 2017. A randomized diversification strategy for solving satisfiability problem with long clauses. *Science China Information Sciences* 60 (2017), 1–11. https://api.semanticscholar.org/CorpusID:13219850

[6] Ian P. Gent and Toby Walsh. 1994. The SAT Phase Transition. In *European Conference on Artificial Intelligence*, Vol. 94. PITMAN, 105–109.

[7] Ruben Heradio, Hector Perez-Morago, David Fernández-Amorós, Roberto Bean, Francisco Javier Cabrerizo, Carlos Cerrada, and Enrique Enrique Herrera-Viedma. 2016. Binary Decision Diagram Algorithms to Perform Hard Analysis Operations on Variability Models. In *New Trends in Software Methodologies, Tools and Techniques.*

[8] Andreas Kübler, Christoph Zengler, and Wolfgang Küchlin. 2010. Model Counting in Product Configuration. In *LoCoCo.*

[9] David G. Mitchell, Bart Selman, and Hector J. Levesque. 1992. Hard and Easy Distributions of SAT Problems. In *AAAI Conference on Artificial Intelligence*, Vol. 92. 459–465.

[10] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. 1999. Determining computational complexity from characteristic 'phase transitions'. *Nature* 400, 6740 (1999), 133–137.

[11] Jeho Oh, Paul Gazzillo, Don Batory, Marijn Heule, and Margaret Myers. 2020. *Scalable Uniform Sampling for Real-World Software Product Lines.* Technical Report TR-20-01.

[12] Jeho Oh, Paul Gazzillo, and Don S. Batory. 2019. *t*-wise coverage by uniform sampling. In *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume A, Paris, France, September 9-13, 2019*, Thorsten Berger, Philippe Collet, Laurence Duchien, Thomas Fogdal, Patrick Heymans, Timo Kehrer, Jabier Martinez, Raúl Mazo, Leticia Montalvillo, Camille Salinesi, Xhevahire Tërnava, Thomas Thüm, and Tewfik Ziadi (Eds.). ACM, 15:1–15:4.

[13] Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. 2019. Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet?. In *12th IEEE Conference on Software Testing, Validation and Verification, ICST 2019, Xi'an, China, April 22-27, 2019*. 240–251.

[14] Yash Pote, Saurabh Joshi, and Kuldeep S. Meel. 2019. Phase Transition Behavior of Cardinality and XOR Constraints. *ArXiv* abs/1910.09755 (2019). https://api.semanticscholar.org/CorpusID:199465708

[15] Chico Sundermann, Tobias Heß, Michael Nieke, Paul Maximilian Bittner, Jeffrey M. Young, Thomas Thüm, and Ina Schaefer. 2023. Evaluating state-of-the-art # SAT solvers on industrial configuration spaces. *Empirical Software Engineering* 28, 2 (2023), 29.

[16] Chico Sundermann, Michael Nieke, Paul Maximilian Bittner, Tobias Heß, Thomas Thüm, and Ina Schaefer. 2021. Applications of #SAT Solvers on Feature Models. *Proceedings of the 15th International Working Conference on Variability Modelling of Software-Intensive Systems* (2021), 1–10.