GANAK: A Scalable Probabilistic Exact Model Counter*†

Shubham Sharma[1], Subhajit Roy[1], Mate Soos[2], and Kuldeep S. Meel[2]

[1] Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur, India
{smsharma, subhajit}@iitk.ac.in,
[2] School of Computing, National University of Singapore
soos.mate@gmail.com, meel@comp.nus.edu.sg

1 Abstract

Given a Boolean formula F, the problem of propositional model counting, also referred to as #SAT, seeks to compute the number of solutions of F. Model counting is a fundamental problem with a wide variety of applications ranging from quantified information flow, network reliability, planning, probabilistic reasoning, and the like [19, 2, 9, 12, 17, 10, 3]. For example, given a graph G such that each of its edges fails with some probability and two nodes, s and t, the problem of computing probability of existence of a path from s to t can be reduced to that of propositional model counting [10].

In his seminal paper, Valiant showed that #SAT is #P-complete, where #P is the set of counting problems associated with NP decision problems [26]. Theoretical investigations of #P have led to the discovery of deep connections in complexity theory, and there is strong evidence for its hardness. In particular, Toda proved that every problem in the polynomial hierarchy could be solved by just one call to a #P oracle; more formally, $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$ [25].

The earliest efforts to #SAT focused on extending the Davis-Putnam-Loveland-Longemann (DPLL) procedure [8] by incrementally computing the number of solutions and adding appropriate multiplicative factors after a partial solution was found [4]. Subsequently, Relsat focused on partitioning the formula into components with a disjoint set of variables. In a significant breakthrough, Sang et al. pioneered the idea of component caching combined with Conflict Driven Clause Learning (CDCL) architecture in their exact counter Cachet [20, 21]. Thurley [24] improved upon Cachet's component caching scheme along with tighter engineering integration and developed sharpSAT. Several knowledge compilation-based counters, often a hybrid of static and dynamic decomposition, have been proposed over the past few years along with novel techniques for preprocessing [14, 13, 15].

Despite significant progress in model counting over the years, the core components of the architecture of dynamic decomposition based techniques have remained constant. Furthermore, SAT solving have witnessed significant improvements over the past decade owing to the development of new heuristics [16, 18, 11]. Moreover, recent years have witnessed the rise of approximate model counters owing to the combination of hashing-based frameworks and use of independent support [23, 12, 5, 6, 22]. In this context, we revisit the architecture of the state-of-the-art exact model counter, sharpSAT, and seek to redesign the architecture and augment the existing techniques with new heuristics.

The primary contribution of this work is a novel architecture, called GANAK[1], that deviates significantly from sharpSAT as follows:

*This paper is published in International Joint Conference on Artificial Intelligence (IJCAI), 2019.

†The open source tool along with benchmarks is available at https://github.com/meelgroup/ganak

[1]GANAK (गणक in Sanskrit) refers to a device that counts.

1. We investigate the usage of universal hash functions for exact model counting. To this end, we design, to the best of our knowledge, the first probabilistic component cache scheme and the first probabilistic exact model counter. In particular, Ganak takes in a formula F and a confidence parameter δ as input and returns `count` such that `count` is the number of solutions of F with confidence at least $1 - \delta$. Note that probabilistic exact model counting is almost as hard as exact model counting and significantly hard compared to probabilistic approximate model counting [7].
2. We propose new branching heuristic that seek to achieve the best of both worlds: perform branching on variables so as to maximize cache hits and perform branching on variables that lead to conflict as soon as possible.

Moreover, we propose new heuristics: *phase selection heuristic*, *independent support*, *exponentially decaying randomness* and *learn and start over*, and perform extensive experiments to study the effect of these heuristics, in isolation and in combination. Finally, we use our experience from the above study to build Ganak that inherits current advancements in SAT solving and model counting, improves upon them and contributes new ideas, thereby outperforming state-of-the-art model counters. In particular, Ganak outperforms state-of-the-art exact and approximate model counters sharpSAT and ApproxMC3 respectively, both in terms of PAR-2[2] score and the number of instances solved. Moreover, in our experiments, the model count returned by Ganak was equal to the exact model count for all the benchmarks.

References

[1] Proc. of SAT Competition 2017: Solver and Benchmark Descriptions. University of Helsinki, Department of Computer Science, 2017.

[2] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *Proc. of FOCS*, pages 340–351, 2003.

[3] Fabrizio Biondi, Michael Enescu, Annelie Heuser, Axel Legay, Kuldeep S. Meel, and Jean Quilbeuf. Scalable approximation of quantitative information flow in programs. In *Proc. of VMCAI*, pages 71–93, 2018.

[4] Elazar Birnbaum and Eliezer L. Lozinskii. The Good Old Davis-Putnam Procedure Helps Counting Models. *J. Artif. Int. Res.*, pages 457–477, 1999.

[5] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A Scalable Approximate Model Counter. In *Proc. of CP*, pages 200–216, 2013.

[6] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Algorithmic Improvements in Approximate Counting for Probabilistic Inference: From Linear to Logarithmic SAT Calls. In *Proc. of IJCAI*, pages 3569–3576, 2016.

[7] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. On the Hardness of Probabilistic Inference Relaxations. In *Proc. of AAAI*, 2019.

[8] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, pages 201–215, 1960.

[9] Carmel Domshlak and Jörg Hoffmann. Probabilistic planning via heuristic forward search and weighted model counting. *JAIR*, pages 565–620, 2007.

[10] Leonardo Dueñas-Osorio, Kuldeep S. Meel, Roger Paredes, and Moshe Y. Vardi. Counting-based reliability estimation for power-transmission grids. In *Proc. of AAAI*, 2017.

[11] Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In *Proc. of SAT*, pages 502–518, 2004.

[2] PAR-2 scheme, that is, penalized average runtime, used in SAT-2017 Competition [1], assigns a runtime of two times the time limit (instead of a “not solved” status) for each benchmark not solved by a solver.

[12] Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Near-Uniform sampling of combinatorial spaces using XOR constraints. In *Proc. of NIPS*, pages 481–488, 2007.

[13] Jean-Marie Lagniez, Emmanue Lonca, and Pierre Marquis. Improving Model Counting by Leveraging Definability. In *Proc. of IJCAI*, pages 751–757, 2016.

[14] Jean-Marie Lagniez and Pierre Marquis. Preprocessing for propositional model counting. In *Proc of AAAI*, pages 2688–2694, 2014.

[15] Jean-Marie Lagniez and Pierre Marquis. An Improved Decision-DNNF Compiler. In *Proc. of IJCAI*, pages 667–673, 2017.

[16] J. P. Marques-Silva and K. A. Sakallah. GRASP: a search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, pages 506–521, 1999.

[17] Kuldeep S. Meel, Moshe Y. Vardi, Supratik Chakraborty, Daniel J Fremont, Sanjit A Seshia, Dror Fried, Alexander Ivrii, and Sharad Malik. Constrained Sampling and Counting: Universal Hashing Meets SAT Solving. In *Proc. of Beyond NP Workshop*, 2016.

[18] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proc. of DAC*, pages 530–535, 2001.

[19] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, pages 273–302, 1996.

[20] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. In *Proc. of SAT*, 2004.

[21] Tian Sang, Paul Beame, and Henry Kautz. Performing Bayesian inference by weighted model counting. In *Prof. of AAAI*, pages 475–481, 2005.

[22] Mate Soos and Kuldeep S. Meel. BIRD: Engineering an Efficient CNF-XOR SAT Solver and its Applications to Approximate Model Counting. In *Proc of AAAI*, 2019.

[23] Larry Stockmeyer. The complexity of approximate counting. In *Proc. of STOC*, pages 118–126, 1983.

[24] Marc Thurley. SharpSAT: counting models with advanced component caching and implicit BCP. In *Proc. of SAT*, pages 424–429, 2006.

[25] Seinosuke Toda. On the computational power of PP and (+)P. In *Proc. of FOCS*, pages 514–519, 1989.

[26] Leslie G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, pages 410–421, 1979.