Approximate Counting of Minimal Unsatisfiable Subsets*

Jaroslav Bendík[1] and Kuldeep S. Meel[2]

[1] Masaryk University, Brno, Czech Republic
[2] National University of Singapore, Singapore

Given an unsatisfiable Boolean formula F as a set of clauses $\{f_1, f_2, \dots f_n\}$, also known as conjunctive normal form (CNF), a set N of clauses is a Minimal Unsatisfiable Subset (MUS) of F iff $N \subseteq F$, N is unsatisfiable, and for each $f \in N$ the set $N \setminus \{f\}$ is satisfiable. Since MUSes can be viewed as representing the *minimal reasons* for unsatisfiability of a formula, MUSes have found applications in wide variety of domains ranging from diagnosis [16], ontologies debugging [1], spreadsheet debugging [13], formal equivalence checking [10], constrained counting and sampling [12], and the like. As the scalable techniques for identification of MUSes appeared only about decade and half ago, the earliest applications primarily focused on a reduction to the identification of a single MUS or a small set of MUSes. With an improvement in the scalability of MUS identification techniques, researchers have now sought to investigate extensions of MUSes and their corresponding applications. One such extension is MUS counting, i.e., counting the number of MUSes of F. Hunter and Konieczny [11], Mu [16], and Thimm [21] have shown that the number of MUSes can be used to compute different inconsistency metrics for general propositional knowledge bases.

In contrast to the progress in the design of efficient MUS identification techniques, the work on MUS counting is still in its nascent stages. Reminiscent of the early days of model counting, the current approach for MUS counting is to employ a complete MUS enumeration algorithm, e.g., [20, 14, 4, 2], to explicitly identify all MUSes. However, there can be up to exponentially many MUSes of F w.r.t. $|F|$, and thus their complete enumeration can be practically intractable. Indeed, contemporary MUS enumeration algorithms often cannot complete the enumeration within a reasonable time [4, 14, 3, 17]. In this context, one wonders: *whether it is possible to design a scalable MUS counter without performing explicit enumeration of MUSes?*

The answer to the above question is *yes*! In our recent work [5], we have presented a probabilistic counter, called AMUSIC, that takes in a formula F, tolerance parameter ε, confidence parameter δ, and returns an estimate guaranteed to be within $(1 + \varepsilon)$-multiplicative factor of the exact count with confidence at least $1 - \delta$. Crucially, for F defined over n clauses, AMUSIC explicitly identifies only $\mathcal{O}(\log n \cdot \log(1/\delta) \cdot (\varepsilon)^{-2})$ many MUSes even though the number of MUSes can be exponential in n.

The design of AMUSIC is inspired by recent successes in the design of efficient XOR hashing-based techniques [6, 7] for the problem of model counting, i.e., given a Boolean formula G, compute the number of models (also known as solutions) of G. We observe that both the problems are defined over a power-set structure. In MUS counting, the goal is to count MUSes in the power-set of F, whereas in model counting, the goal is to count models in the power-set that represents all valuations of variables of G. Chakraborty et al. [8, 18] proposed an algorithm, called ApproxMC, for approximate model counting that also provides the (ϵ, δ) guarantees. ApproxMC is currently in its third version, ApproxMC3 [18]. The base idea of ApproxMC3 is to partition the power-set into *nCells* small cells, then pick one of the cells, and count the number *inCell* of models in the cell. The total model count is then estimated as

*This work was (will be) originally published [5] at CAV 2020. A preprint version is available at https://www.fi.muni.cz/~xbendik/publications/CAV2020.pdf. The original paper is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/)

nCells $\times$ *inCell*. Our algorithm for MUS counting is based on ApproxMC3. We adopt the high-level idea to partition the power-set of F into small cells and then estimate the total MUS count based on a MUS count in a single cell. The difference between ApproxMC3 and AMUSIC lies in the way of counting the target elements (models vs. MUSes) in a single cell; we propose novel MUS specific techniques to deal with this task. In particular, our contribution is the following:

- We introduce a QBF (quantified Boolean formula) encoding for the problem of counting MUSes in a single cell and use a Σ_3^P oracle to solve it.
- Let $\texttt{UMU}_F$ and $\texttt{IMU}_F$ be the union and the intersection of all MUSes of F, respectively. We observe that every MUS of F (1) contains $\texttt{IMU}_F$ and (2) is contained in $\texttt{UMU}_F$. Consequently, if we determine the sets $\texttt{UMU}_F$ and $\texttt{IMU}_F$, then we can significantly speed up the identification of MUSes in a cell.
- We propose a novel approaches for computing the union $\texttt{UMU}_F$ and the intersection $\texttt{IMU}_F$ of all MUSes of F.
- We implement AMUSIC and conduct an extensive empirical evaluation on a set of *scalable* benchmarks. We observe that AMUSIC is able to compute estimates for problems clearly beyond the reach of existing enumeration-based techniques. We experimentally evaluate the *accuracy* of AMUSIC. In particular, we observe that the estimates computed by AMUSIC are significantly closer to true count than the theoretical guarantees provided by AMUSIC.

Our work opens up several new interesting avenues of research. From a theoretical perspective, we make polynomially many calls to a Σ_3^P oracle while the problem of finding a MUS is known to be in FP^{NP}, i.e. a MUS can be found in polynomial time by executing a polynomial number of calls to an NP-oracle [9, 15]. Contrasting this to model counting techniques, where approximate counter makes polynomially many calls to an NP-oracle when the underlying problem of finding satisfying assignment is NP-complete, a natural question is to close the gap and seek to design a MUS counting algorithm with polynomially many invocations of an FP^{NP} oracle. From a practitioner perspective, our work calls for a design of MUS techniques with native support for XORs; the pursuit of native support for XOR in the context of SAT solvers have led to an exciting line of work over the past decade [19, 18].

References

[1] M. Fareed Arif, Carlos Mencía, Alexey Ignatiev, Norbert Manthey, Rafael Peñaloza, and João Marques-Silva. BEACON: an efficient sat-based tool for debugging *EL*+ ontologies. In *SAT*, volume 9710 of *LNCS*, pages 521–530. Springer, 2016.

[2] Fahiem Bacchus and George Katsirelos. Finding a collection of MUSes incrementally. In *CPAIOR*, volume 9676 of *LNCS*, pages 35–44. Springer, 2016.

[3] Jaroslav Bendík and Ivana Černá. Evaluation of domain agnostic approaches for enumeration of minimal unsatisfiable subsets. In *LPAR*, volume 57 of *EPiC Series in Computing*, pages 131–142. EasyChair, 2018.

[4] Jaroslav Bendík, Ivana Černá, and Nikola Beneš. Recursive online enumeration of all minimal unsatisfiable subsets. In *ATVA*, volume 11138 of *LNCS*, pages 143–159. Springer, 2018.

[5] Jaroslav Bendík and Kuldeep S. Meel. Approximate counting of minimal unsatisfiable subsets. In *CAV*, Lecture Notes in Computer Science. Springer, 2020 (to appear).

[6] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In *Proc. of CP*, pages 200–216, 2013.

[7] 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*, 2016.

[8] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In *IJCAI*, pages 3569–3576. IJCAI/AAAI Press, 2016.

[9] Zhi-Zhong Chen and Seinosuke Toda. The complexity of selecting maximal solutions. *Inf. Comput.*, 119(2):231–239, 1995.

[10] Orly Cohen, Moran Gordon, Michael Lifshits, Alexander Nadel, and Vadim Ryvchin. Designers work less with quality formal equivalence checking. In *DVCon*. Citeseer, 2010.

[11] Anthony Hunter and Sébastien Konieczny. Measuring inconsistency through minimal inconsistent sets. In *KR*, pages 358–366. AAAI Press, 2008.

[12] Alexander Ivrii, Sharad Malik, Kuldeep S. Meel, and Moshe Y. Vardi. On computing minimal independent support and its applications to sampling and counting. *Constraints*, 21(1), 9 2016.

[13] Dietmar Jannach and Thomas Schmitz. Model-based diagnosis of spreadsheet programs: a constraint-based debugging approach. *Autom. Softw. Eng.*, 23(1):105–144, 2016.

[14] Mark H. Liffiton and Ammar Malik. Enumerating infeasibility: Finding multiple MUSes quickly. In *CPAIOR*, volume 7874 of *LNCS*, pages 160–175. Springer, 2013.

[15] João Marques-Silva and Mikolás Janota. On the query complexity of selecting few minimal sets. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:31, 2014.

[16] Kedian Mu. Formulas free from inconsistency: An atom-centric characterization in priest's minimally inconsistent LP. *J. Artif. Intell. Res.*, 66:279–296, 2019.

[17] Nina Narodytska, Nikolaj Bjørner, Maria-Cristina Marinescu, and Mooly Sagiv. Core-guided minimal correction set and core enumeration. In *IJCAI*, pages 1353–1361. ijcai.org, 2018.

[18] Mate Soos and Kuldeep S Meel. Bird: Engineering an efficient CNF-XOR sat solver and its applications to approximate model counting. In *Proc. of the AAAI*, 2019.

[19] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Proc. of SAT*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009.

[20] Roni Tzvi Stern, Meir Kalech, Alexander Feldman, and Gregory M. Provan. Exploring the duality in conflict-directed model-based diagnosis. In *AAAI*. AAAI Press, 2012.

[21] Matthias Thimm. On the evaluation of inconsistency measures. *Measuring Inconsistency in Information*, 73, 2018.