An Approximate Model Counter for ASP – Extended Abstract –*

Flavio Everardo[1,2], Markus Hecher[1,3], and Ankit Shukla[4]

[1] University of Potsdam, Germany
[2] Tecnológico de Monterrey Puebla Campus, Mexico
[3] TU Wien, Vienna, Austria
[4] JKU, Linz, Austria

1 Introduction

Answer Set Programming (ASP; [18]) is a declarative framework that is well-suited for problems in KR, AI, and other areas as well as plenty of practical applications, in both academia and industry. [5] While modern ASP solvers not only compute one solution (answer set), but actually support different reasoning modes, the problem of counting answer sets has not been subject to intense studies yet. This is in contrast to the neighboring area of propositional satisfiability (SAT), where several applications and problems related to quantitative reasoning trace back to model counting. However, due to high computational complexity and depending on the actual application, approximate counting might be sufficient. Indeed, there are plenty of applications, where approximate counting for SAT is well-suited. This extended abstract presents the work published in [6] by introducing the first approximate counter for ASP. This is done by lifting ideas from SAT to ASP by adapting the algorithms and approaches from [2] into an extension of the *clingo*-based system *xorro* [7].

Recently, there has been growing interest in counting solutions to problems. Indeed, counting solutions is a well-known task not only in mathematics and computer science, but also in other areas [4,13]. Examples of these cover also applications in machine learning and probabilistic inference [3]. In terms of computational complexity, counting has been well-studied since the late 70s [15,21]. There are also results for counting involving projection, where one wants to count only with respect to a given set of projected atoms, which has been established for logic [8,17,14,19], reliability estimation [5] as well as in ASP [11,9].

Given that in general counting answer sets is hard, namely $\# \cdot$ coNP-complete [10], which further increases to $\# \cdot \Sigma_2^P$-completeness [9] if counting with respect to a projection, a different approach than exact counting seems to be required in practice. Indeed, such approaches were successful for propositional logic (SAT), which is $\# \cdot$ P-complete and where reasoning modes like sampling (near-uniform generation) or (approximate) model counting [12,1,2,19] were studied. For this purpose, so-called parity (XOR) constraints are used specifically to partition the search space in parts that preferably are of roughly the same size.

* Accepted at the NMR'20 workshop [6].

[5] An incomplete list of ASP applications:
https://www.dropbox.com/s/pe261e4qi6bcyyh/aspAppTable.pdf

Parity constraints have been recently accommodated in ASP as the fundamental part of the *clingo*-based system *xorro* [7]. With different solving approaches over parity constraints in *xorro*, these constraints amount to the classical XOR operator following the aggregates-like syntax using theory atoms. These constraints are interpreted as directives, solved on top of an ASP program acting as answer set filters that do not satisfy the parity constraints in question.

2 Approximate Counting by Partitioning the Search Space

The algorithm from our approximate count takes the notions from [2], where the main idea is to use parity constraints to partition the search space as the ones used in *xorro*. On each iteration, a new parity constraint is added to reduce the search space roughly by half. We use 3-wise independent hashing (XOR) functions to partition the set of models of an input formula into "small" cells. These small cells with high probability will carry the same number of answer sets after each partition. The algorithm uses two input values, a tolerance $\epsilon(0 < \epsilon \leq 1)$ and a confidence δ $(0 < \delta \leq 1)$ value to calculate how small a cell must be, and the number of iterations of the algorithm. So, for each iteration calculated from δ, the algoritm will perform the following:

- Add a new parity constraint and solve.
- If the number of remaining answer sets is less or equal the *pivot* value, then we count all the remaining answer sets and store them in a list C.
- Otherwise, we repeat the first two steps.
- This is done until one of two cases happen, either we have a small cell to count, or the cell is empty (UNSAT) and this iteration is discarded.

Lastly, the approximate count is the mean of C.

To test our ASP approximate counter, we performed a series of benchmarks with different ASP problem classes with random-generated instances. Our setup involves the use of different tolerance and confidence values under a modest resources laptop. For now, we focus only on the quality of the counting, leaving the scalability and performance for further work. To track the quality of our counting, we only used "easy to solve" instances for *clingo*, meaning that *clingo* must be able to enumerate all answer sets within 600 seconds timeout (without printing).

While our preliminary results are promising and show that indeed approximate counting works for ASP, there is still potential for future works. On the one hand, we highly recommend studying and showing proper guarantees that prove our results are guaranteed to be accurate with high probability and do not deviate far from the actual result. Further, we highly encourage additional tuning and improvements to our preliminary implementation concerning several aspects such as the XOR constraints solving, scalability, and the abolition of linear combinations. Talking about scalability, we need to perform more experiments and algorithm revisions in order to perform better than *clingo*'s (*clasp*'s) enumeration. We hope that this work fosters applications and further research on quantitative reasoning like e.g., [16,20], for ASP.

References

1. Chakraborty, S., Meel, K., Vardi, M.: A scalable and nearly uniform generator of SAT witnesses. In: Sharygina, N., Veith, H. (eds.) Proceedings of the Twenty-fifth International Conference on Computer Aided Verification (CAV'13). LNCS, vol. 8044, pp. 608–623. Springer (2013)
2. Chakraborty, S., Meel, K.S., Vardi, M.Y.: A scalable approximate model counter. In: Schulte, C. (ed.) Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8124, pp. 200–216. Springer (2013), https://doi.org/10.1007/978-3-642-40627-0_18
3. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artificial Intelligence 172(6–7), 772—799 (2008)
4. Domshlak, C., Hoffmann, J.: Probabilistic planning via heuristic forward search and weighted model counting. J. Artif. Intell. Res. 30 (2007)
5. Dueñas-Osorio, L., Meel, K.S., Paredes, R., Vardi, M.Y.: Counting-based reliability estimation for power-transmission grids. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17). pp. 4488–4494. The AAAI Press, San Francisco, CA, USA (Feb 2017)
6. Everardo, F., Hecher, M., Shukla, A.: An Approximate Model Counter for ASP. In: NMR@KR (2020)
7. Everardo, F., Janhunen, T., Kaminski, R., Schaub, T.: The return of xorro. In: Balduccini, M., Lierler, Y., Woltran, S. (eds.) Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3-7, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11481, pp. 284–297. Springer (2019), https://doi.org/10.1007/978-3-030-20528-7_21
8. Fichte, J.K., Hecher, M., Morak, M., Woltran, S.: Exploiting treewidth for projected model counting and its limits. In: SAT'18. LNCS, vol. 10929, pp. 165–184. Springer (Jul 2018)
9. Fichte, J.K., Hecher, M.: Treewidth and counting projected answer sets. In: LPNMR'19. LNCS, vol. 11481, pp. 105–119. Springer (2019)
10. Fichte, J.K., Hecher, M., Morak, M., Woltran, S.: Answer set solving with bounded treewidth revisited. In: LPNMR. Lecture Notes in Computer Science, vol. 10377, pp. 132–145. Springer (2017)
11. Gebser, M., Kaufmann, B., Schaub, T.: Solution enumeration for projected boolean search problems. In: van Hoeve, W.J., Hooker, J.N. (eds.) Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'09). LNCS, vol. 5547, pp. 71–86. Springer, Berlin (2009)
12. Gomes, C., Sabharwal, A., Selman, B.: Near-uniform sampling of combinatorial spaces using XOR constraints. In: Schölkopf, B., Platt, J., Hofmann, T. (eds.) Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems (NIPS'06). pp. 481–488. MIT Press (2007)
13. Gomes, C.P., Sabharwal, A., Selman, B.: Chapter 20: Model counting. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 633–654. IOS Press, Amsterdam, Netherlands (Feb 2009)
14. Gupta, R., Sharma, S., Roy, S., Meel, K.S.: Waps: Weighted and projected sampling. In: Vojnar, T., Zhang, L. (eds.) Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19). pp. 59–76. Springer, Prague, Czech Republic (Apr 2019), held as Part of the European Joint Conferences on Theory and Practice of Software
15. Hemaspaandra, L.A., Vollmer, H.: The satanic notations: Counting classes beyond #P and other definitional adventures. SIGACT News 26(1), 2–13 (Mar 1995)

16. Kimmig, A., Demoen, B., Raedt, L.D., Costa, V.S., Rocha, R.: On the implementation of the probabilistic logic programming language problog. Theory Pract. Log. Program. 11(2-3), 235–262 (2011), https://doi.org/10.1017/S1471068410000566
17. Lagniez, J.M., Marquis, P.: A recursive algorithm for projected model counting. In: Hentenryck, P.V., Zhou, Z.H. (eds.) Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19). Honolulu, Hawaii, USA (2019)
18. Lifschitz, V.: Answer set planning. In: ICLP. pp. 23–37. MIT Press (1999)
19. Sharma, S., Roy, S., Soos, M., Meel, K.S.: Ganak: A scalable probabilistic exact model counter. In: Kraus, S. (ed.) Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 1169–1176. IJCAI (7 2019)
20. Tsamoura, E., Gutiérrez-Basulto, V., Kimmig, A.: Beyond the grounding bottleneck: Datalog techniques for inference in probabilistic logic programs. In: AAAI. pp. 10284–10291. AAAI Press (2020)
21. Valiant, L.: The complexity of enumeration and reliability problems. SIAM J. Comput. 8(3), 410–421 (1979)