Lifting Symmetry Breaking Constraints with Inductive Logic Programming (Extended Abstract)

Alice Tarzariol[1] , **Martin Gebser**[1,2] , **Konstantin Schekotihin**[1]
[1]Univeristy of Klagenfurt
[2]Graz University of Technology
{alice.tarzariol, martin.gebser, konstantin.schekotihin}@aau.at

Abstract

Our work addresses the generation of first-order constraints to reduce symmetries and improve the solving performance for classes of instances of a given combinatorial problem. To this end, we devise a model-oriented approach obtaining positive and negative examples for an Inductive Logic Programming task by analyzing instance-specific symmetries for a training set of instances. The learned first-order constraints are interpretable and can be used to augment a general problem encoding in Answer Set Programming. This extented abstract introduces the context of our work, contributions and results.

1 Introduction

Modern declarative programming paradigms allow for relatively simple modeling of various combinatorial problems. However, solving such problems might become infeasible when the size of input instances and, correspondingly, the number of possible solution candidates start to grow (Dodaro et al. 2016). Often many of these candidates are symmetric, namely, one candidate can be obtained from another by renaming constants. Therefore, the ability to incorporate *Symmetry Breaking Constraints* (SBCs) in a problem encoding becomes an essential skill for programmers. Nevertheless, identifying symmetric solutions and formulating constraints that remove (only) them might be a time-consuming and challenging task. Consequently, various tools emerged for avoiding the computation of symmetric solution candidates. A popular technique consists of the automatic detection and introduction of SBCs using properties of permutation groups, while dynamic approaches apply specific search methods that detect and discard symmetric states; see (Sakallah 2009; Walsh 2012) for overviews.

In general, two families of automatic symmetry breaking techniques can be distinguished: *instance-specific* and *model-oriented* approaches. The former identify symmetries for a particular instance at hand and augment a ground problem representation with specific SBCs to improve the subsequent solving process. Unfortunately, computational advantages do often not carry forward to large-scale instances or advanced encodings, where *instance-specific* symmetry breaking can require as much computational effort as it takes to solve the original problem. Moreover, ground SBCs generated by *instance-specific* approaches are *(i)* not transferable, since the knowledge obtained is limited to a single instance; *(ii)* usually hard to interpret and comprehend; *(iii)* derived from permutation group generators, whose computation is itself a combinatorial problem; and *(iv)* often redundant and might result in a degradation of the solving performance. In contrast, *model-oriented* approaches aim to find general SBCs that depend less on a particular instance.

For problem encodings in Answer Set Programming (ASP) (Brewka, Eiter, and Truszczyński 2011; Gebser et al. 2012; Lifschitz 2019), the tool SBASS (Drescher 2015) implements an *instance-specific* symmetry breaking approach for ground programs. However, to the best of our knowledge, there was no *model-oriented* system to lift ground SBCs for ASP programs. Therefore, in (Tarzariol, Gebser, and Schekotihin 2021), we have introduced a novel *model-oriented* method, using Inductive Logic Programming (ILP) (Cropper, Dumančić, and Muggleton 2020) to generalize the process of discarding redundant solution candidates for instances of a target domain. ILP is a form of machine learning whose goal is to learn a logic program that explains given observations in the context of some pre-existing knowledge. The currently most expressive ILP system for ASP is ILASP (Law, Russo, and Broda 2020; Law, Russo, and Broda 2021), which can be used to solve a variety of ILP tasks.

2 Paper Contributions

The goal of our work is the identification and lifting of SBCs obtained for small instances of a combinatorial problem encoded in ASP, in order to derive interpretable first-order constraints. Such constraints cut the search space, while preserving the satisfiability of a problem for the considered instance distribution, and improve the solving performance, especially in the case of unsatisfiability. More precisely, in our original paper, we *(i)* propose an approach to generate a training set including positive and negative examples, allowing an ILP method to learn first-order SBCs for the problem at hand; *(ii)* define the components of ILP learning tasks enabling the generation of effective constraints for ASP solving; *(iii)* show how to apply our method iteratively, to revise first-order constraints when new permutation group generators or more training instances become available; and *(iv)* conduct experiments on variants of the pigeon-hole problem as well as the application-oriented house configuration problem (Friedrich et al. 2011).

3 Discussion of Results

The results of experiments in our paper show that the introduced learning framework manages to significantly outperform a state-of-the-art *instance-specific* method (SBASS) as well as the ASP system CLINGO (Gebser et al. 2012) without SBCs for a family of combinatorial problems. The solving speed-up is not the only improvement, as the learned first-order constraints are also better interpretable than ground SBCs, which are even not represented symbolically but in the SMODELS internal format (Syrjänen 2001) resembling DIMACS for propositional formulas. We believe that our paper can be of interest to a broad KR community, as it combines machine learning with KRR methods to gain insights and performance improvements for classes of problem instances. Although our method is designed and implemented for ASP, the main concepts can be applied to other KR formalisms as well, like constraint or logic programming. This may benefit KR applications in practice and especially in industrial settings, where the prevalent process and customer demands tend to remain stable over significant time periods.

Acknowledgments

This work was partially funded by KWF project 28472, cms electronics GmbH, FunderMax GmbH, Hirsch Armbänder GmbH, incubed IT GmbH, Infineon Technologies Austria AG, Isovolta AG, Kostwein Holding GmbH, and Privatstiftung Kärntner Sparkasse.

References

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Cropper, A.; Dumančić, S.; and Muggleton, S. 2020. Turning 30: New ideas in inductive logic programming. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI'20)*, 4833–4839. ijcai.org.

Dodaro, C.; Gasteiger, P.; Leone, N.; Musitsch, B.; Ricca, F.; and Schekotihin, K. 2016. Combining answer set programming and domain heuristics for solving hard industrial problems. *Theory and Practice of Logic Programming* 16(5-6):653–669.

Drescher, C. 2015. *Conflict-driven constraint answer set solving*. Ph.D. Dissertation, Computer Science and Engineering, Faculty of Engineering, UNSW.

Friedrich, G.; Ryabokon, A.; Falkner, A.; Haselböck, A.; Schenner, G.; and Schreiner, H. 2011. (Re)configuration using answer set programming. In *IJCAI 2011 Workshop on Configuration*, 17–24. CEUR-WS.org.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers.

Law, M.; Russo, A.; and Broda, K. 2020. The ILASP system for inductive learning of answer set programs. The Association for Logic Programming Newsletter.

Law, M.; Russo, A.; and Broda, K. 2021. Ilasp. www.ilasp.com.

Lifschitz, V. 2019. *Answer Set Programming*. Springer-Verlag.

Sakallah, K. 2009. Symmetry and satisfiability. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. chapter 10, 289–338.

Syrjänen, T. 2001. Lparse 1.0 user's manual.

Tarzariol, A.; Gebser, M.; and Schekotihin, K. 2021. Lifting symmetry breaking constraints with inductive logic programming. In Zhi-Hua, Z., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI'21)*, 2062–2068. ijcai.org.

Walsh, T. 2012. Symmetry breaking constraints: Recent results. In Hoffmann, J., and Selman, B., eds., *Proceedings of the Twenty-Sixth National Conference on Artificial Intelligence (AAAI'12)*, 2192–2198. AAAI Press.