Learning constraint models from data

Dimosthenis C. Tsouros [1], **Tias Guns** [1], **Kostas Stergiou** [2]

[1]Department of Computer Science,
KU Leuven, Leuven, Belgium
dimos.tsouros@kuleuven.be, *tias.guns@kuleuven.be*
[2]Department of Electrical & Computer Engineering,
University of Western Macedonia, Kozani, Greece
kstergiou@uowm.gr

Abstract

We propose an overview of constraint acquisition research, in which learning techniques are used to learn constraint models from data. We discuss passive and (inter)active learning, the connections to the machine learning field, as well as challenges that arise in constraint expressivity, search space size, convergence and noisy data.

Introduction

Constraint programming (CP) is widely used for solving real-world problems. The basic assumption in CP is that the user models the problem and a solver is then used to solve it. Despite the many successful applications of CP on combinatorial problems from various domains, there are still challenges to be faced in order to make CP technology even more widely used. A major bottleneck in the use of CP is modeling. Expressing a combinatorial problem as a set of constraints over decision variables requires substantial expertise, and this non-trivial task is often a major bottleneck for the widespread adoption of CP.

To overcome this obstacle, several techniques have been proposed for modeling a constraint problem (semi-)automatically, and nowadays assisting the user in modeling is regarded as one of the important aspects of CP research. An area of research that has started to attract a lot of attention is that of *constraint acquisition*, which is an area where CP meets Machine Learning (ML). In constraint acquisition, the model of a constraint problem is acquired (i.e. learned) using a set of examples of solutions, and possibly non-solutions.

Passive and active constraint learning

Constraint acquisition can come in various flavours depending on the assumptions on the input: do we assume that a set of solutions is given upfront? Do they data given contain also non-solutions? Or is there no data given apriori, but a user can be queried about whether an assignment is a solution or not? and what kinds of queries can be asked?

- In *passive* acquisition, examples of solutions and optionally non-solutions are provided by the user. It can be seen as an ML dataset, often assumed to be noise-free. The goal is to reconstruct the model of the problem, extracting a set of constraints from the date, accepting the solutions and rejecting the non-solutions given.
- In contrast, in *active* or *interactive* acquisition, the learner interacts with a user dynamically while acquiring the target set of constraints. It can be seen as a form of active learning in ML, where labels are only revealed after asking a user to classify the examples.

Techniques for learning constraints from data

In most state-f-the-art constraint acquisition systems, the user provides the system with the decision variables of the problem and a set of abstract relations. Applying these relations to each subset of variables, the system obtains the set of candidate constraints, usually referred to as *bias*. The goal is to find a subset of constraints that accepts the positive examples and rejects the negative ones. The size of the bias can be very large, so efficient techniques have to be used to search over the candidates for the target constraint set.

Many methods, both for passive and active constraint acquisition, are based on ideas from the candidate elimination and version space paradigms (Beldiceanu and Simonis 2012; Bessiere et al. 2017, 2013; Tsouros and Stergiou 2020) proposed in the early days of ML. In such systems, the examples are taken into account one by one, with each example either shrinking the set of candidate constraints by removing the candidates that are inconsistent or learning constraints that must hold for the assignments to be classified as solutions. This is also called the ”generate-and-test” approach, where each candidate constraint from the bias is generated and tested whether it violates any of the examples.

For passive constraint acquisition, a recent promising alternative to ”generate-and-test” is the ”generate-and-aggregate” approach: using a simple grammar of aggregation operators, different aggregation expressions are generated and applied to various slices of matrices and, in general, tensors of decision variables. By finding the lower and upper bounds of these expressions, relevant parameters can be identified from the data (Kumar, Kolb, and Guns 2022). In addition, there are methods that aim to link the constraint acquisition field with data science more closely, by training classifiers that are then transformed into constraint models (Prestwich et al. 2021).

Challenges

Despite recent advances in constraint learning, there are significant obstacles to surmount:

- Although active constraint acquisition systems have a good theoretical bound in terms of the number of queries, the number of queries is still quite high for a real interaction with human users, which can be a preventive factor. This could be alleviated by exploiting the structure of the learned network, focusing the queries on specific parts of the problem that are more promising. Apart from asking the user to just classify generated examples, more expressive queries can also be used to cut down the number of interactions (Daoudi et al. 2015; Bessiere et al. 2014).
- Most methods in constraint learning use a candidate set of constraints to search for the target constraint set. However, as there is a lack of methods to handle high arity constraints, or even global constraints, efficiently, the size of the set of candidates can become too large. Methods that are able to handle such cases are required for real-world applications.
- Besides the space management problem, the luck of specific methods to learn specific classes of constraints, such as linear constraints with unknown constants, also exacerbates the issue with the number of queries in interactive methods. Thus, specific methods to handle classes of constraints are needed to improve the performance of interactive constraint acquisition systems.
- In both passive and active constraint learning, the classification of all the examples is assumed to be correct. However, in practical applications we can have noisy data. Techniques to deal with incorrect classifications have to be adapted.
- Despite the recent progress in the field of constraint acquisition, the focus has been mainly on learning satisfaction problems and hard constraints. Not a lot of studies deal with learning constraint optimization problems via soft constraints, in the context of constraint programming (Tsouros and Stergiou 2021). However, soft constraints are widely used to represent preferences, which are of paramount importance in Artificial Intelligence. This is an important drawback of constraint acquisition, presenting many limitations to its application in a variety of real applications.

For constraint learning to become more broadly applicable and to better assist the user during the modeling process, it is important to tackle the aforementioned shortcomings.

Relevant Publications by the authors:

- Dimosthenis C. Tsouros: (Tsouros, Stergiou, and Sarigiannidis 2018; Tsouros, Stergiou, and Bessiere 2019; Tsouros and Stergiou 2020; Tsouros, Stergiou, and Bessiere 2020; Tsouros and Stergiou 2021)
- Tias Guns: (Berden et al. 2022; Kumar, Kolb, and Guns 2022)
- Kostas Stergiou: (Tsouros, Stergiou, and Sarigiannidis 2018; Tsouros, Stergiou, and Bessiere 2019; Tsouros and Stergiou 2020; Tsouros, Stergiou, and Bessiere 2020; Tsouros and Stergiou 2021)

References

Beldiceanu, N.; and Simonis, H. 2012. A model seeker: Extracting global constraint models from positive examples. In *Principles and practice of constraint programming*, 141–157. Springer.

Berden, S.; Kumar, M.; Kolb, S.; and Guns, T. 2022. Learning MAX-SAT Models from Examples using Genetic Algorithms and Knowledge Compilation. In *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*.

Bessiere, C.; Coletta, R.; Daoudi, A.; Lazaar, N.; Mechqrane, Y.; and Bouyakhf, E.-H. 2014. Boosting Constraint Acquisition via Generalization Queries. In *ECAI*, 99–104.

Bessiere, C.; Coletta, R.; Hebrard, E.; Katsirelos, G.; Lazaar, N.; Narodytska, N.; Quimper, C.-G.; Walsh, T.; et al. 2013. Constraint Acquisition via Partial Queries. In *IJCAI*, volume 13, 475–481.

Bessiere, C.; Koriche, F.; Lazaar, N.; and O'Sullivan, B. 2017. Constraint acquisition. *Artificial Intelligence*, 244: 315–342.

Daoudi, A.; Lazaar, N.; Mechqrane, Y.; Bessiere, C.; and Bouyakhf, E. H. 2015. Detecting types of variables for generalization in constraint acquisition. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 413–420. IEEE.

Kumar, M.; Kolb, S.; and Guns, T. 2022. Learning Constraint Programming Models from Data Using Generate-And-Aggregate. In *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*.

Prestwich, S. D.; Freuder, E. C.; O'Sullivan, B.; and Browne, D. 2021. Classifier-based constraint acquisition. *Annals of Mathematics and Artificial Intelligence*, 1–20.

Tsouros, D. C.; and Stergiou, K. 2020. Efficient multiple constraint acquisition. *Constraints*, 25(3): 180–225.

Tsouros, D. C.; and Stergiou, K. 2021. Learning Max-CSPs via Active Constraint Acqusition. In *27th International Conference on Principles and Practice of Constraint Programming*.

Tsouros, D. C.; Stergiou, K.; and Bessiere, C. 2019. Structure-Driven Multiple Constraint Acquisition. In *International Conference on Principles and Practice of Constraint Programming*, 709–725. Springer.

Tsouros, D. C.; Stergiou, K.; and Bessiere, C. 2020. Omissions in Constraint Acquisition. In *International Conference on Principles and Practice of Constraint Programming*, 935–951. Springer.

Tsouros, D. C.; Stergiou, K.; and Sarigiannidis, P. G. 2018. Efficient Methods for Constraint Acquisition. In *24th International Conference on Principles and Practice of Constraint Programming*.