Requirements for Practical Constraint Acquisition

Helmut Simonis
Insight SFI Research Centre for Data Analytics
School for Computer Science and Information Technology
University College Cork
Cork, Ireland
helmut.simonis@insight-centre.org

November 19, 2022

Constraint Acquisition [1] is one of the on-going research areas which combines Constraint Programming (CP) and Machine Learning ideas. We argue that the currently defined use case for Constraint Acquisition does not allow a practical use of this technology, and a more general use case should be considered.

Constraint Acquisition (CA) is the process of defining a constraint model from a set of example solutions and non-solutions. In most of the existing literature, a model of a single problem instance is learned, which can then only be used to generate new solutions of the same problem, or perhaps the same problem with different domain bounds. In most practical CP applications, the actual constraint model heavily depends on input data. Consider a scheduling problem for a factory. Every day the set of orders that need to be scheduled will be different, and from time to time new products or processes will be introduced. A CP model for this factory needs to be able to create the appropriate constraint model for a given dataset, and change the specific constraints of the instance from one day to the next, due to different process steps or resources used. If we want to acquire a generic constraint model in this scenario, we have to understand how the number of variables, their domains, and the constraints depend on the input data. Having acquired the model, we then should be able to apply the model to a new, previously unseen set of input data, and solve the resulting model. This is a more challenging problem than the use case previously discussed in constraint acquisition.

We propose that Constraint Acquisition should consider a more general use case: We are given a set of input data, solutions and non-solutions for multiple problem instances of different sizes. A constraint acquisition tool produces a generic model based on those problem instances. This model can then be fed, together with new, unseen input data, into an existing constraint solver to produce a solution for this new instance. The user can interact with the Constraint Acquisition to accept or reject certain constraints, in addition the users should be able to understand and extend the generated model if they so wish.

This leads to the following set of properties that a generic constraint acquisition tool should have:

diverse structure We must be able to handle not just a single set of unstructured variables, but a collection of variables, vectors and matrices, and possibly more complex structures. We can exploit this structural information to find likely constraints linking multiple structures, which would be difficult to identify in an unstructured list.

hidden variables The variables described in the solution may not describe all variables used in the constraint model. Some derived variables, especially cost variables, may be missing from the samples, with only aggregate cost values presented. The CA tool should reconstruct such missing, hidden variables in order to find the complete constraint model.

generic The model should not be specific to one given instance only, but apply to a series of instances which are specified by different input data and parameter values.

consistent The generated model should be consistent, so that all given positive samples are accepted, and all negative samples are rejected. A more challenging problem arises when is is possible that some instances are labelled incorrectly.

minimal The model should not contain any constraints that are not required, in particular any subsumed constraint should be filtered out.

transferable We must be able to use the generated model on new, previously unseen input data, as long as the structure of the problem remains the same.

executable It is not enough to generate an abstract model, we should be able to execute the model with some existing constraint solver.

explainable We should be able to explain the generated model to a user not familiar with specific constraint programming systems and languages. A natural language description of the model would provide a good initial explanation.

efficient As we are solving combinatorially hard problems, we will not be able to solve every new problem instance. We consider the generated model efficient if its runtime is comparable to a handwritten program for the same problem written in a high-level modelling language like MiniZinc.

For the bridge event, we propose a discussion on the aims of Constraint Acquisition, and how we can make this research more relevant to practical constraint problems. This could take the form of a presentation of the ideas presented here, followed by a discussion in the audience (or a panel) on this topic. Minimum time would be 20 minutes for a presentation, followed by a discussion phase of up to one hour.

References

[1] Christian Bessiere, Frédéric Koriche, Nadjib Lazaar, and Barry O'Sullivan. Constraint acquisition. *Artif. Intell.*, 244:315–342, 2017.