# Building Bridges Between Machine Learning and Constraint Solving

Alexander Felfernig[1], Viet-Man Le[1], Mathias Uta[2], Denis Helic[3], Seda Polat-Erdeniz[1], Sebastian Lubos[1], Thi Ngoc Trang Tran[1], Damian Garber[1], and Tamim Burgstaller[1]

[1]Graz University of Technology, Graz, Austria

[2]Siemens Energy AG, Erlangen, Germany

[3]Modul University Vienna, Austria

Constraint-based reasoning is applied in various scenarios such as the optimization of production schedules [1], product configuration [2], and recommendation [3]. Taking knowledge-based configuration as an example, typical tasks supported by constraint solvers (acting as configurators) are (1) to *check the consistency* of a set of variable assignments, (2) to *check the feasibility of a given set of user requirements* (regarding a configuration), and (3) to *complete given partial configurations* taking into account additional constraints such as to *minimize the total price* or the *included set of components* (only needed components should be included in a final configuration).

The mentioned tasks can be easily supported by existing constraint solving techniques which can be regarded as typical representatives of knowledge-based Artificial Intelligence [4,7]. A more detailed analysis of constraint-based configuration scenarios (and constraint solving scenarios in general) shows a need to support further related tasks – for a related overview, we refer to Popescu et al. [6]. When activating a constraint solver to solve a given configuration task, it would be good to know *which heuristics* (e.g., in terms of variable and value orderings) should be applied to solve a specific problem. In this context, we could also be interested in a *prediction of satisfiability*, i.e., to predict if a solution exists for a specific CSP (without the need of activating the solver). Furthermore, specifically in interactive scenarios, it is highly relevant to *predict relevant user preferences*, i.e., being able to recommend variable value settings to a user (in situations where users are not able and willing to specify all of a given set of parameters). Finally, it would also be of interest to predict which of a potentially large set of *explanations* will be the most relevant ones for a user [8]. Different machine learning (ML) approaches can help to support these tasks [6].

On an abstract level, the following can be regarded as representative integration scenarios of constraint solving and machine learning [5,6,10]. (1) *solver selection* is the task of identifying an optimal solver (or solver parametrization) to solve a given CSP as fast as possible. In this context, the input for a machine learning component is meta information about previously solved CSPs (e.g., in terms of structural properties of the CSP, used algorithm/parametrization, and corresponding runtime needed to solve the problem). The output of a machine learning model is a recommended parametrization (solver selection) for the current CSP. (2) Similar to solver recommendation, *search heuristics could be recommended* on the basis of information about structural properties of the current (to be solved) CSP. (3) in some scenarios, it can also be of interest to apply a machine learning model to *predict the satisfiability of a CSP* (without activating a constraint solver). In this context, the input for a machine learning component would again be meta-information about previously solved (or unsolved) CSPs. (4) Specifically in interactive scenarios, it is highly relevant to *predict the relevance of specific variable settings for a user*. To learn user preferences, the input for a

corresponding machine learning component is a set of existing CSP solutions, the output is a machine learning model that is able to predict user preferences on the basis of a given set of already specified user preferences – a simplified example is given in Table 1.

| session | x1 | x2 | x3 | x4 | x5 |
|---|---|---|---|---|---|
| **1** | **1** | **2** | **1** | **3** | **6** |
| 2 | 2 | 1 | 3 | 2 | 2 |
| 3 | 3 | 3 | 2 | 3 | 1 |
| **current** | **1** | **2** | **1** | **2** | **?-> 6** |

*Table 1: A simple example: the user in the current session has already defined his/her preferences regarding the variables x1-x4 and would like to have a recommended setting for the variable x5. The nearest neighbour user session (in terms of similar preferences) is session 1. Consequently, a recommendation for the current user regarding the setting of variable x5 could be x5=6. This is a simplified example of a memory-based recommendation approach which does not take into account domain-specific constraints, i.e., such a recommendation could result in an inconsistency. Alternative approaches do not directly recommend variable settings but focus on recommending, for example, variable value orderings. For more related information, we refer to the overview of Popescu et al. [6].*

Basically, machine learning (ML) models can be (1) *memory-based*, where predictions/classifications are determined, for example, on the basis of parametrizations chosen for problems with similar meta-properties (e.g., *k-nearest neighbour based ML models*) and (2) *model-based*, where predictions/classifications are determined with ML models following the idea of dimensionality reduction, i.e., the set of features used for predicting/classifying is a reduced one compared to the number of "input" parameter settings (e.g., *matrix factorization* and *neural network based ML models*). Typical *metrics* for evaluating the performance of a machine learning model take into account *prediction errors* (e.g., mean absolute error - MAE) and *classification errors* (e.g., accuracy) [9]. The major research challenge in this interdisciplinary field is to find ways to *exploit synergy effects*, for example, on the levels of (1) *integrating machine learning models into constraint learning and constraint reasoning* as well as (2) *extending existing machine learning approaches* in such a way that domain knowledge defined in terms of constraints can be easily integrated / taken into account in ML model building processes.

**References**:


[1] R. Barták and M. Salido. Constraint Satisfaction for Planning and Scheduling Problems. Constraints, 16(3):223-227, 2011.

[2] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. Knowledge-based Configuration: From Research to Business Cases. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[3] A. Felfernig and R. Burke. Constraint-Based Recommender Systems: Technologies and Research Issues. In 10th Intl. Conference on Electronic Commerce, ICEC '08, 3:1-10, 2008.

[4] E. Freuder. In Pursuit of the Holy Grail. Constraints, 2:57–61, 1997.

[5] E. O'Mahony, E. Hebrard, A. Holland, C. Nugent, and B. O'Sullivan. Using Case-based Reasoning in an Algorithm Portfolio for Constraint Solving? Irish Conference on Artificial Intelligence and Cognitive Science, pp. 210-216, 2013.

[6] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V. Le, K. Pilsl, M. Enzelsberger, and T. Tran. An Overview of Machine Learning Techniques in Constraint Solving. Journal of Intelligent Information Systems, 58(1): 91–118, 2022.

[7] F. Rossi, P. van Beek, and T. Walsh. Handbook of Constraint Programming. ISSN. Elsevier Science, 2006.

[8] B. O'Sullivan, A. Papadopoulos, B. Faltings, and Pearl Pu: Representative Explanations for Over-Constrained Problems. AAAI 2007, pp. 323-328, 2007.

[9] M. Uta, A. Felfernig, V. Le, A. Popescu, T. Tran, and D. Helic. Evaluating Recommender Systems in Feature Model Configuration. ACM SPLC'21, pp. 58-63, 2021.

[10] L. Xu, H. Hoos, and K. Leyton-Brown. Predicting Satisfiability at the Phase Transition. AAAI 2012, pp. 584-590, 2012.