Semi-Automatic Modeling by Constraint Acquisition ⋆ ⋆⋆

Remi Coletta[1], Christian Bessiere[1], Barry O'Sullivan[2], Eugene C. Freuder[2], Sarah O'Connell[2], and Joel Quinqueton[1]

[1] LIRMM-CNRS (UMR 5506), 161 rue Ada 34392 Montpellier Cedex 5, France
{coletta,bessiere,jq}@lirmm.fr
[2] Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{b.osullivan,e.freuder,s.oconnell}@4c.ucc.ie

Abstract. Constraint programming is a technology which is now widely used to solve combinatorial problems in industrial applications. However, using it requires considerable knowledge and expertise in the field of constraint reasoning. This paper introduces a framework for automatically learning constraint networks from sets of instances that are either acceptable solutions or non-desirable assignments of the problem we would like to express. Such an approach has the potential to be of assistance to a novice who is trying to articulate her constraints. By restricting the language of constraints used to build the network, this could also assist an expert to develop an efficient model of a given problem.

1 Introduction

Over the last 30 years, considerable progress has been made in the field of Constraint Programming (CP). However, the use of CP still remains limited to specialists in the field. Modelling a problem in the constraint formalism requires significant expertise in constraint programming. Indeed, humans usually find it difficult to articulate their constraints. While the human user can recognize examples of where their constraints should be satisfied or violated, they cannot articulate the constraints themselves. However, by presenting examples of what is acceptable, the human user can be assisted in developing a model of the set of constraints she is trying to articulate. This can be regarded as an instance of constraint acquisition. One of the goals of our work is to assist the, possibly novice, human user by providing semi-automatic methods for acquiring the user's constraints.

Furthermore, even if the user has sufficient experience in CP to encode her problem, a poor model can negate the utility of a good solver based on state-of-the-art filtering techniques. For example, in order to provide support for modelling, some solvers

⋆ [1] contains a long version of this paper.

⋆⋆ The collaboration between LIRMM and the Cork Constraint Computation Centre is supported by a Ulysses Travel Grant from Enterprise Ireland, the Royal Irish Academy and CNRS (Grant Number FR/2003/022). This work has also received support from Science Foundation Ireland under Grant 00/PI.1/C075.

provide facilities for defining constraints extensionally (i.e., by enumerating the set of allowed tuples). Such facilities considerably extend the expressiveness and ease-of-use of the constraints language, thus facilitating the definition of complex relationships between variables. However, a disadvantage of modelling constraints extensionally is that the constraints lose any useful semantics they may have which can have a negative impact on the inference and propagation capabilities of a solver. Therefore, another goal of our work is to facilitate the expert user who wishes to reformulate her problem (or a part of it that is suspected of slowing down the resolution). Given sets of accepted/forbidden instantiations of the (sub)problem (that can be generated automatically on the initial formulation), the expert will be able, for instance, to test whether an optimised constraint library associated with her solver is able to model the (sub)problem in a way which lends itself to being efficiently solved.

However, constraint acquisition is not only important in an interactive situation involving a human user. Often we may wish to acquire a constraint model from a large set of data. For example, given a large database of tuples defining buyer behaviour in a variety of markets, for a variety of buyer profiles, for a variety of products, we may wish to acquire a constraint network which describes the data in this database. While the nature of the interaction with the source of training data is different, the constraint acquisition problem is fundamentally the same.

Our contribution is an algorithm (named CONACQ), that extends version space machine learning techniques [3] to deal with the specificity of learning constraints. It takes solutions (positive instances) and non solutions (negative instances), called a training set, as input, and generates a (set of) constraint network(s) consistent with the training set. Using version spaces we can maintain the whole set of possible 'target' networks during the learning process. This set is represented by the more tighter (specific bound) and looser (general bound) networks consistent with the training data. We adapted the classical version space technique to maintain a reasonably low space complexity. In the following, we just give an overview of the learning framework, and discuss preliminary experiments and the issues they raise. A comprehensive description of the CONACQ algorithm can be found in [1].

2 The Fundamental Problem

As a starting point, we assume that the user knows the set of variables of her problem and their domains of possible values. She is also assumed to be able to provide or classify both positive (a solution) and negative (non-solution) examples. Therefore, the available data are the set $\mathcal{X}$ of the variables of the problem, their domains $\mathcal{D}$, a subset E^+ of the solutions of the problem, and a set E^- of non-solutions.

In addition, from the 'assisting the expert' perspective, the aim is to encode the problem efficiently, using only efficient constraint relations between these variables; i.e. a library of constraints with efficient propagation features is assumed to be known. Indications can also be given revealing the possible location of the constraints, by defining variables between which constraints must be found (learned), or by restricting ourselves to binary constraints only. These semantical and structural limitations define the inductive bias:

Definition 1 (Bias). *Given a set $\mathcal{X}$ of variables and the set $\mathcal{D}$ of their domains, a* bias $\mathcal{B}$ *on $(\mathcal{X}, \mathcal{D})$ is a sequence $(B_1, \ldots, B_m)$ of local biases, where a local bias B_i is defined by a sequence $var(B_i) \subseteq \mathcal{X}$ of variables, and a set $L(B_i)$ of possible relations on $var(B_i)$.*

The set $L(B_i)$ of relations allowed on a set of variables $var(B_i)$ can be any library of constraints of arity $|var(B_i)|$.

Definition 2 (Membership of a Bias). *Given a set $\mathcal{X}$ of variables and the set $\mathcal{D}$ of their domains, a sequence of constraints $\mathcal{C} = (C_1, \ldots, C_m)$* belongs *to the bias $\mathcal{B} = (B_1, \ldots, B_m)$ on $(\mathcal{X}, \mathcal{D})$ if $\forall C_i \in \mathcal{C}, var(C_i) = var(B_i)$ and $rel(C_i) \in L(B_i)$. We note $\mathcal{C} \in \mathcal{B}$.*

The problem consists in looking for a sequence of constraints $\mathcal{C}$ belonging to a given bias $\mathcal{B}$, and whose solution set is a superset of E^+ containing no element of E^-.

Definition 3 (Constraint Acquisition Problem). *Given a set of variables $\mathcal{X}$, their domains $\mathcal{D}$, two sets E^+ and E^- of instances on $\mathcal{X}$, and a bias $\mathcal{B}$ on $(\mathcal{X}, \mathcal{D})$, the* constraint acquisition problem *consists in finding a sequence of constraints $\mathcal{C}$ such that:*

$\mathcal{C} \in \mathcal{B}$,
$\forall e^- \in E^-$, e^- is a non solution of $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, and,
$\forall e^+ \in E^+$, e^+ is a solution of $(\mathcal{X}, \mathcal{D}, \mathcal{C})$.

If the sets E^+ and E^-, called the *training data*, are provided by an interaction with the user, then the acquisition problem can be regarded as the modelling phase for the user's problem. Otherwise, it can be regarded as an assistance to the expert for an automatic reformulation of her problem.

As stated in the introduction, a version space does not only provide one consistent hypothesis, but all constraint sequences belonging to a bias that are consistent with the training data:

Definition 4 (Version Space). *Given $(\mathcal{X}, \mathcal{D})$ a set of variables and their domains, E^+ and E^- two training data sets , and $\mathcal{B}$ a bias on $(\mathcal{X}, \mathcal{D})$, the* version space *is the set:*

$$V = \{\mathcal{C} \in H / E^+ \subseteq Sol(\mathcal{X}, \mathcal{D}, \mathcal{C}), E^- \cap Sol(\mathcal{X}, \mathcal{D}, \mathcal{C}) = \emptyset\}$$

3 Experiments and Observations

We report here on some preliminary experiments to evaluate the learning capabilities of our approach. Rather than focusing on techniques for minimising the number of interactions, our focus here is on studying a number of properties of the CONACQ algorithm which provide motivation for our research agenda.

We performed experiments with a simulated teacher, which plays the role of the user, and a simulated learner. The teacher has the knowledge of a randomly generated (target) network, represented by the triple $< 50, 8, C >$, defining a problem involving 50 variables with domains $\{1, ..8\}$, and a number C of constraints. Each constraint is randomly chosen from the bias $\{<, =, >, \leq, \neq, \geq\}$. The teacher provides the learner with solutions and non solutions. The learner acquires a version space for the problem using the CONACQ algorithm [1].

3.1 Experiment 1: Effect of the order of the instances

In this following experiment, we assess aspects of the runtime characteristics of the CONACQ algorithm. In particular, we study computing time and the size of the version space while varying the order in which examples are presented. Instances from a set E of size 100 are given by the teacher to the learner based on a $< 50, 8, 50 >$ network. The set E contains 10 positive and 90 negative instances.

Table 1 presents the time needed by the learner to acquire the version space, V, for the example set while varying the arrival time of the 10 positive instances. The positive instances were presented at the beginning (a), middle (b), and end (c) of the interaction between teacher and learner.

Table 1. Effect of the timing of the introduction of positive instances

Introduction date for positives	0 (a)	50 (b)	90 (c)
Computing time (in sec.)	3.3	5.1	8.6
$log(\|V\|)$	2,234	2,234	2,234

We observe that *"the sooner, the better"* seems to be the good strategy for the introduction of positive instances. Indeed, the specific bound rises quickly with positive instances, reducing the size of the version space. Because of that, the CPU time needed is also reduced when positive instances arrive at the beginning. But we can see that the final size of the version space is not affected by the order of the instances. This is due to the commutativity property of version spaces.

3.2 Experiment 2: Partial instances

In some cases, the user can reject an instance while justifying it by a negative sub-instance. For example, in a real-estate setting the customer (teacher) might reject an apartment citing the reason that *"this living-room is too small for me"*. The estate agent (learner) knows that the violation is due to the variables defining the living room, which can being very helpful for handling negative examples. The usefulness of such justified rejections can be measured by providing our learner with partial instances. In the following experiment (Table 2), the teacher provides the learner with 90 partial negative instances (after 10 complete positive ones) in the training data. We consider partial instances involving 2, 5, 10 variables, and report the size of the version space and of the set of clauses (effective space used to represent the general bound) after 100 instances have been given.

Table 2. Effect of the partial instances

Nb of variables involved in instances of E^-	50	10	5	2
$log(\|V\|)$	2,234	2,233	2,225	2,144
$\|K\|$ (10^4 meta-variables)	7.6	6.1	3.2	0

We observe that partial instances speed up the process of convergence of the version space. The smaller are these partial instances, the more helpful they are. This opens a

promising way of helping the learning process: asking the user to justify why she rejects some instances can assist in reducing the length of the dialog with the teacher. This is a critical issue if we are learning in an interactive setting from a human user.

4 Aspects of our Research Agenda

In this paper we have presented an approach to acquiring models of constraint satisfaction problems from examples. There is signifi cant scope for research in this area. Here we give some insights into some aspects of our research in this area.

Standard version space learning algorithms are senstive to noise in the training data and, as a consequence, are brittle to false positives and negatives provided to the algorithm. However, some recent work from the machine learning community gives us a basis for making our approach more robust to such errors [2].

Another issue for which we did not show experiments because of space limitations, is that of implicit constraints and redundancy. An implicit constraint is one that does not belong to a network but that could be detected by inference. For example, if we have $X_1 = X_2$ and $X_2 = X_3$ in a network $\mathcal{N}$, the constraint $X_1 = X_3$ is an implicit constraint for $\mathcal{N}$. The general phenomenon of constraints that can be inferred by other constraints can prevent the version space from converging to the smallest possible one. Applying some levels of local consistency seems to be a promising approach for improving the reduction of the version space, by adding implicit constraints to the learned network. When we will deal with partial instances, this will have some interesting implications, such as the effect that the order in which examples are provided has on the representability of a particular problem in the given constraint language.

Finally, we considering the effect that various models of interaction can have on the speed with which we can learn the target problem, particularly from the perspective of minimising the number of interactions with the user. Some preliminary results have already been reported on this issue [4, 5].

References

1. R. Coletta, C. Bessiere, Barry O'Sullivan, E.C. Freuder, Sarah O'Connell, and Joel Quinqueton. Semi-automatic modeling by constraint acquisition. Technical Report 03050, LIRMM – University of Montpellier II, Montpellier, France, June 2003. (available at http://www.lirmm.fr/~bessiere/).
2. F.A. Marginean. Soft learning: A bridge between data mining and machine learning. In *Proceedings of the 4th International Conference on Recent Advances in Soft Computing (RASC-2002)*, pages 108–115, December 2002.
3. Tom Mitchell. Concept learning and the general-to-specific ordering. In *Machine Learning*, chapter 2, pages 20–51. McGraw Hill, 1997.
4. S. O'Connell, B. O'Sullivan, and E.C. Freuder. Strategies for interactive constraint acquisition. In *Proceedings of the CP-2002 Workshop on User-Interaction in Constraint Satisfaction*, pages 62–76, September 2002.
5. B. O'Sullivan, S. O'Connell, and E.C. Freuder. Interactive constraint acquisition for concurrent engineering. In *Proceedings of the 7th International Conference on Concurrent Enterprising – ICE-2003*, 2003.