Introduction to the special issue on constraint modelling and reformulation

Andrea Rendl · J. Christopher Beck

Published online: 22 February 2013
© Springer Science+Business Media New York 2013

Over the past 20 years, Constraint Programming (CP) has developed into a successful approach for solving a wide variety of combinatorial optimization and decision problems. The general approach in CP is two-fold. First, as with any mathematical tools, the problem of interest must be represented or *modeled* in a form that enables the tools to be applied. Second, the techniques that have been developed and embodied in a constraint solver are applied to *solve* the problem by acting on the representation. In the case of CP, the primary solution techniques are a combination of search and inference in the form of constraint propagation. A particular strength of Constraint Programming is its expressiveness: many models are possible for a given problem and the representation can be extended through the definition and implementation of new global constraints that become both modelling objects and locii of solving via the accompanying inference algorithms.

However, as is well-known, the model has a strong impact on the solving performance: a poorly formulated model will not be solvable within reasonable time, independent of the capabilities of the constraint solver. In contrast, a different, but logically equivalent, model may solve quickly. Thus, modelling is a key component of the successful use of Constraint Programming techniques. The development of an understanding of modelling, such as a characterization of good and bad models, is a crucial component for the continued success and applicability of CP.

A. Rendl (✉)
Austrian Institute of Technology,
Vienna, Austria
e-mail: andrea.rendl@ait.ac.at

J. C. Beck
University of Toronto,
Toronto, ON, Canada
e-mail: jcb@mie.utoronto.ca

The last ten years have witnessed significant research devoted to modelling problems with constraints. Some of the main trends have been:¹

- to simplify modelling for novices
- to develop automated reformulation techniques both for model enhancement and to allow specification of constraint programs in higher level languages
- to adapt remodelling techniques from related fields, such as SAT solving and mixed-integer programming
- to hybridize constraint models by adding both representation and solving techniques from related domains
- to develop advanced declarative languages for constraint modelling
- to decouple declarative languages from solvers
- to develop automated symmetry breaking techniques to enhance models and solving approaches
- to formulate new problems in CP

The *Workshop on Constraint Modelling and Reformulation* brings together researchers who are interested in various facets of constraint modelling and reformulation, including but not limited to the themes noted above. In honour of the tenth edition of the Workshop on September 17, 2011 in Perugia, Italy, the *Constraints* journal has solicited submissions for a special issue. Twelve papers were submitted and went through the standard rigorous reviewing process of *Constraints*. The result is the five accepted papers in this special issue.

In *Formulating the Template ILP Consistency Problem as a Constraint Satisfaction Problem*, Barták et al. present an example of using CP to model and solve a problem from another area: the template matching problem in inductive logic programming. Through careful use of “out of the box” constraints, a CP model is built that is competitive with and, in some cases, much better than the specialized solvers that represent the current state of the art.

Heinz et al., in *Using Dual Presolving Reductions to Reformulate Cumulative Constraints*, show how the sharing of information among global constraints can allow a global constraint to perform dual reasoning: fixing or removing variables and decomposing constraints while guaranteeing that an optimal solution remains in the feasible region. Given the substantial exploitation of such approaches in mixed-integer solving, this is a promising cross-fertilization that bears on both reformulation and solving of constraint programs.

The papers by Bofill et al. and Ansótegui et al. are both examples of reformulation of existing problems to exploit solution techniques for related areas. In the former, *Reformulation based MaxSAT Robustness*, the authors examine finding robust solutions to MaxSAT problems via a reformulation to a pseudo-Boolean problem. The approach provides robust solutions, if they exist, as well as possible repairs to restore feasibility in the case of unexpected changes to the problem definition. This work builds upon recent contributions in the SAT and CP communities and shows that reformulation as a means to gain robustness can be very effective, in contrast to

¹A number of these items in this list come from the invited talk entitled “A Decade of Research on Constraint Modelling and Reformulation: The Quest for Abstraction and Automation” given by Alan M. Frisch at the *Tenth Workshop on Constraint Modelling and Reformulation* (www-users.cs.york.ac.uk/~frisch/Research/decade.pdf, accessed January 24, 2013).

the common approach of applying dedicated search techniques. In *Solving Weighted CSPs with Meta-Constraints by Reformulation into Satisfiability Modulo Theories*, Ansótegui et al. introduce a declarative language for modelling weighted constraint satisfaction problems and a reformulation process to automatically derive a MaxSMT representation in order to solve the problem with the SAT-based SMT technology. This paper is an example, therefore, of both the development of declarative languages and the integration of approaches from related areas of problem solving into constraint programming.

In the final paper in this issue, *Search Combinators*, Schrijvers et al. propose a modelling approach not for constraint programs but for heuristic search algorithms that can be applied to constraint programs. While declarative languages for search techniques are not listed in the themes above, the literature contains a number of examples, as indicated by the authors. They introduce a domain-specific modular search language based on the concept of combinators as well as a modular, extendable architecture. It is demonstrated that search heuristics declared in the language can lead to exactly the same search trees as heuristics coded in lower level programming languages. Furthermore, the overhead for the declarative language is insignificant compared to the other components of the search, such as inference and backtracking.

We gratefully acknowledge the help of all reviewers who contributed their time and expertise to this special issue. Moreover, we are particularly grateful to Gilles Pesant, Editor-in-chief of Constraints, for inviting us to edit this special issue and for offering expert guidance during the process.

Andrea Rendl & Chris Beck, January 2013.