Ontologies for Continuous Global Optimization

Laurent Granvilliers and Mina Ouabiba
IRIN – University of Nantes
B.P. 92208 – 44322 Nantes cedex 3 – France
e-mail: {granvilliers, ouabiba}@irin.univ-nantes.fr

Global Optimization tackles the problem of finding all feasible points of a set of constraints that optimize an objective function. In the following, we restrict our attention to Continuous Global Optimization, and more specifically to Global Optimization over interval domains [6, 9].

We believe that Software Design for Continuous Global Optimization has things in common with Astronomy of Sixteenth Century. Many powerful systems implementing various solving techniques have already been developed. However, there exists no general cooperation architecture [4] describing knowledge sharing among solvers. More precisely, a cooperation model would define representations and specifications of shared knowledge, and protocols of communication.

In Knowledge Engineering, the problem of representing shared and reusable knowledge among software agents has heavily been studied in the recent past. This problem can be addressed by ontologies. An *ontology* [5] is a specification of concepts of a given domain and relationships among them. One of the objectives is to lay foundations for libraries of reusable components and knowledge sharing functionalities.

Knowledge based systems (KBS) are often modeled by means of three concepts: task, PSM (Problem Solving Method) and domain [8]:

- Domain describes the knowledge of a particular domain, e.g., global optimization.
- Tasks define problems that should be solved, e.g., constraint solving.
- PSMs define resolution processes of problems, e.g., LP techniques.

The relation between them can be specified by means of semantic links:

- Intra-concept link for the relationships between two identical concepts task/task, PSM/PSM and domain/domain.
- Inter-concept link for two different concepts. Such a link can be used to transfer information between them.

Figure 1: Objectives of Ontologies Design for Solver Cooperation.

A real-world application can be seen as a composition of specific components described by means of tasks, PSMs, domain, intra-concept and inter-concept links. In the optimization domain, task ontologies describe classes of optimization problems and relationships between them.

For example, Glopt [3] is a PSM of some specific problems (tasks) defined by a block-separable objective function subject to bound constraints or block-separable constraints. In that case, task/PSM inter-concept links transfer the specific format of the problem structure called NOP[1]. Glopt implements branch-and-bound technique (PSM) which is often used by many other problems. As a consequence, there is a need for describing a generic branch-and-bound algorithm which can be specialized by means of PSM/PSM intra-concept links.

In this work, we are designing ontologies for the domain of Continuous Global Optimization. This research is part of the COCONUT project from the European Community. We have noticed that most of existing systems and platforms for Continuous Global Optimization implement specific and intra-solver cooperations (dotted box of Figure 1): no heterogeneous encoding of data structures, one cooperation concept, *e.g.*, sequential or concurrent, brick solvers and routing of data fixed *a priori*. We have then identified two directions for future research [2]:

- Modeling of extra-solver cooperations, at two-levels: knowledge sharing and strategies of application of solvers.
- Extraction and definition of generic and reusable components.

Part of this research was concerned by the definition of an ontology for a specific intra-solver cooperation: constraint solving using Branch-and-Prune

[1]a compact format for specifying general constrained nonlinear optimization problems.

Algorithms based on Interval Constraint Satisfaction Techniques [10]. This work has led to the implementation of a `C++` library of reusable components and generic strategies. In this extended abstract, we briefly describe design decisions supported by the ontology.

Branch-and-Prune Algorithms are generic in essence: they alternate domain pruning by enforcing local consistency techniques and interval computations [7], and branching to traverse the search space and exhibit all the solutions. Four levels of genericity have been identified and represented in our library:

- **Genericity w.r.t. Interval Arithmetic** (IA). IA is used to compute reliable approximations of ranges of real functions. The evaluation of functions needs to be independent of the IA library in order to plug in extern IA libraries such as `Bias`, `Jail` and `Sun's Forte`. Another motivation is to easily interchange libraries for solving a problem, *e.g.*, for the use of a multi-precision library such as `Mpfi` for handling instable numerical computations. The interval data type is implemented by the *traits* `C++` technique defining all services required by Branch-and-Prune Algorithms.

- **Genericity w.r.t. interval extensions**. The truth value of interval constraints is computed in two consecutive steps: evaluation of function expressions using interval extensions, and interpretation of relation symbols. Interval constraints are parameterized by data types for interval extensions and interpretation of relation symbols. Note that for standard constraints, the interpretation of relation symbols is given by the IA library. This mechanism allows one to define constraints for non standard IA, *e.g.*, modal IA, with no additionnal cost.

- **Genericity w.r.t. domain pruning methods**. This is a main task in itself since there exist various kinds of algorithms. The ontology is based on chaotic iteration framework for constraint propagation [1] which has been adapted for event-based programming. Doma in pruning is modeled as an iteration of reduction functions over interval do mains. Areduction function models either a box consistency operator, or an interval Newton operator, *etc.* Scheduling of reduction functions depends on priorities and properties of functions, *etc.* Strong consistencies are described by strategies combining local splitting and domain pruning.

- **Genericity w.r.t. branching strategies**. Branching strategies are represented by three components: a strategy for selecting the next variable domain to be bisected, a method for splitting domains, and a mechanism for managing memorization of domains (copying or trailing).

Such a generic library has many advantages in terms of flexibility, maintenance, reuse of code, prototyping of strategies, cooperation, *etc.* We believe that this research on Branch-and-Prune Algorithms can be extended for the wider domain of Continuous Global Optimization.

Acknowledgments. This work has been partly supported by the COCONUT project IST-2000-26063 from the European Community.

References

[1] K. R. Apt, “The Essence of Constraint Propagation”, *Theoretical Computer Science*, 1999, Vol. 221, No. 1/2, pp. 179–210.

[2] The COCONUT Consortium, *Algorithms for Solving Nonlinear Constrained and Optimization Problems: the State of The Art*, 2001. Deliverable 2 of the COCONUT project IST-2000-26063 from the European Community.

[3] S. Dallwig, A. Neumaier, and H. Schichl, “GLOPT – a program for constrained global optimization”, In: I. Bomze et al. (eds.), *Developments in Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1997, pp. 19–37.

[4] L. Granvilliers, E. Monfroy, and F. Benhamou, “Symbolic-Interval Cooperation in Constraint Programming”, In: *Proceedings of ISSAC'2001, 26th International Symposium on Symbolic and Algebraic Computation*, University of Western Ontario, Canada, 2001, pp. 150–166.

[5] T. R. Gruber, “Toward Principles for the Design of Ontologies Used for Knowledge Sharing”, In: N. Guarino and R. Poli (eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Kluwer Academic Publishers, 1993.

[6] E. R. Hansen, *Global Optimization Using Interval Analysis*, Marcel Dekker Inc., 1992.

[7] R. E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[8] E. Motta, *Reusable Components for Knowledge Modelling*, IOS Press, Amsterdam, 1999.

[9] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990.

[10] P. Van Hentenryck, L. Michel, and Y. Deville, *Numerica: a Modeling Language for Global Optimization*, MIT Press, 1997.