

MultiObjective Optimization and Constraint Programming

Narendra Jussien*

Vincent Barichard†

*École des Mines de Nantes, LINA CNRS FRE 2729
4, rue Alfred Kastler – BP 20722 — F-44307 Nantes Cedex 3, France
Narendra.Jussien@emn.fr

†LERIA, University of Angers
2, Boulevard Lavoisier — F-49045 Angers Cedex 01, France
Vincent.Barichard@univ-angers.fr

Fast increasing computing power in the sixties led to a wealth of works around problem solving, at the root of Operational Research, Numerical Analysis, Symbolic Computing, Scientific Computing, and a large part of Artificial Intelligence and programming languages. Constraint Programming is a discipline that gathers, interbreeds, and unifies ideas shared by all these domains to tackle decision support problems. This technology has reached in fifteen years a central role in modelling and automatically solving various, both academic and industrial, problems. For example, constraint programming has been successful in production planning, airport traffic control, frequency allocation, hardware validation, molecular biology, robotics, and conception.

1 Principles of Constraint programming

Constraint programming [9] is the study of computational systems based on constraints. The idea of constraint programming is to solve problems by stating constraints (conditions, properties) which must be satisfied by the solution. Work in this area can be tracked back to research in Artificial Intelligence and Computer Graphics in the sixties and seventies. Only in the last decade, however, has there emerged a growing realization that these ideas provide the basis for a powerful approach to programming, modelling and problem solving and that different efforts to exploit these ideas can be unified under a common conceptual and practical framework, constraint programming.

A constraint is a logical relation among several variables, each taking a value in a given domain. A constraint thus restricts the possible values that variables can take, it represents some partial information about the variables of interest. For instance, *the circle is inside the square* relates two objects without precisely specifying their positions, *i.e.*, their coordinates. Now, one may move the square or the circle and he or she is still able to maintain the relation between these two objects. Also, one may want to add another object, say a triangle, and

Loire Valley (City of Tours), France, June 12–14, 2006

introduce another constraint, say *the square is to the left of the triangle*. From the user (human) point of view, everything remains absolutely transparent. Constraints naturally meet the following properties: they can specify partial information, they are non-directional, declarative, additive. However, they are rarely independent: constraints do share variables. Constraints are a natural medium for people to express problems in many fields.

In constraint programming, constraints are considered locally. Each constraint is responsible for ensuring both satisfiability (there exists a solution in the current search space – defined by the current domains of the variable) and pruning (irrelevant parts of the problem are discarded regarding the local view provided by the constraint). They therefore correspond to specialized subproblems. Communication between constraints is done through propagation: shared variables transmit information about domain reduction (pruning). This separation of concerns is at the heart of constraint programming as it allows genericity: constraint programming techniques can be used to solve any kind of combinatorial problem.

Search in constraint programming is usually based on a depth-first exploration of the search space. Various variable and value ordering heuristics have been designed to efficiently explore the search space. But, new techniques arose recently: branching on other meta considerations (precedence constraints in the context of scheduling, etc.); designing new exploration techniques that do not use a tree as search metaphor but try to learn from failures to improve the exploration (using for example SAT related techniques, back-jumping techniques [6], etc.); designing powerful partial exploration that scatters through the search space (LDS-based techniques [4], etc.).

2 Current trends in constraint programming

Current trends in Constraint Programming are multiple:

- extending the scope of constraint programming: handling preferences, optimization, distribution, dynamic and uncertain environments [10], hybrid (numerical/finite) problems, etc.
- extending the power of constraints themselves: designing new consistency techniques, designing global constraints (a known efficient algorithm for solving often encountered subproblems: **alldifferent** for matching problems [7], **cumulative** for scheduling problems [1], etc.), design new search techniques, etc.
- making constraint programming accessible to any engineers: providing modelling tools, learning constraint models, providing debugging and explanation tools [5], providing auto-adaptive constraint solvers, etc.

3 Constraint programming and (multiobjective) optimization

Many bridges have been led between OR and CP: using OR techniques for designing global constraints, cross-fertilizing for designing hybrid search techniques, etc. But, Constraint Programming is essentially a satisfaction based technique. Optimization is handled as a sequence

Loire Valley (City of Tours), France, June 12–14, 2006

of satisfaction problems. CP needs to address optimization problems as such and should take high benefits from cross-fertilization with (multiobjective) optimization.

Constraint Programming can be also extremely useful to multiobjective optimization techniques: solving hard satisfaction sub-problems or computing lower bounds by providing constraints handling tools (*eg.* [2] deals with repropagation tools to solve multiobjective continuous constrained problems and [8] produces a set of lower bounds to the exact solutions for multiobjective constrained problems where all objective functions are additive). Constraint programming is suited for tackling constraints efficiently (*i.e.* pruning the search space) instead of aggregating them in objective functions or during selection process (like evolutionary algorithms do [3]).

References

- [1] Aggoun, A. and Beldiceanu, N. (1993). Extending CHIP in order to solve complex scheduling and placement problems. *Mathl. Comput. Modelling*, 17(7):57–73, 1993.
- [2] Barichard, Vincent and Hao, Jin-Kao (2003). A population and interval constraint propagation algorithm, *Lecture Notes in Computer Science*, 2632: 88-101, Springer.
- [3] Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley.
- [4] Harvey, W. D., and Ginsberg, M. L. (1995): Limited Discrepancy Search. In Proceedings of the *Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 607-615.
- [5] Jussien, Narendra (2003). *The versatility of using explanations within constraint programming*, Habilitation thesis of Nantes University (also available as RR-03-04 research report at École des Mines de Nantes).
- [6] Prosser, Patrick (1993). Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3):268–299.
- [7] Régis, Jean-Charles (1994). A filtering algorithm for constraints of difference in CSPs. In proceedings *AAAI 94, Twelfth National Conference on Artificial Intelligence*, pages 362–367, Seattle, Washington.
- [8] Rollon, E. and Larrosa, J. (in press). *Bucket Elimination for Multiobjective Optimization Problems*. Journal of Heuristics.
- [9] Tsang, Edward (1993). *Foundations of Constraint Satisfaction*. Academic Press.
- [10] Verfaillie, Gérard and Jussien, Narendra (2005). Constraint solving in uncertain and dynamic environments – a survey, *Constraints*, (10)3: 253–281.