A constraint optimization framework for real-time applications

Simon de Givry, Philippe Gérard, Laurent Jeannin,
Juliette Mattioli, Nicolas Museux, Pierre Savéant

THALES Research & Technology
Domaine de Corbeville 91404 Orsay cedex France
simon.degivry@thalesgroup.com

This position paper presents the constraint technology that has been developed[1] at THALES since 1997 for introducing Constraint Programming (CP) in THALES operational systems (see [Givry.et.al01a] for a longer presentation). These systems involve combinatorial optimization problems such as planning and scheduling problems that can be expressed with finite-domain variables and constraints. Typical examples of THALES systems concern supervision, for weapon allocation, radar configuration, weapon deployment and aircraft sequencing. All these systems are subject to specific requirements coming from the operational constraints of embedded real-time systems and from the strategic context of Defense applications:

- The system involves several functions/tasks such as situation assessment, resource management, visualization, etc.; each task is periodical and the period can be much shorter than a second;
- There is a memory space limit (a few megabytes);
- The system has to be supported for a long time, typically over 20 years for Defense applications, including several retrofitting (functional and platform evolutions);
- The system can be reused and modified for building a specific system for a new client (product line);
- The development of the system must be made and mastered in house for reasons of confidentiality and market protection.

The CP paradigm partially meets these requirements. A constraint model has modularity properties, i.e. adding/removing a constraint is easy, which enables an incremental development process, reducing the development time and effort. CP solvers provide efficient algorithms through the use of global constraints. The declarative nature of CP enables the programmer to focus on the application requirements rather than on debugging low-level programming errors. Validated CP models can be reused in a product line approach.
Unfortunately, off-the-shelf CP solvers do not provide any guarantee on time and space usage. The classical backtracking search algorithm used in CP does not take into account any time contract. Recently an effort was made to provide better search algorithms in CP solvers, for instance in [Beldiceanu.et.al98,Laburthe98,Perron99], but without any explicit time contract. Our aim is to extend CP solver with new search features that would keep the same nice software engineering properties as for modeling. This led to develop a high-level language for designing search algorithms. This approach allows to propose a set of search primitives on top of the real-time finite-domain constraint solver Eclair© [Laburthe.et.al98,Platon01]. The resulting search algorithms are based on partial search methods and take into account the time contract explicitly. Such algorithms can take advantage better of platform evolutions.

Eclair offers time and space guarantees. Deadlines are guaranteed by the operating system alarm and Eclair is able to restore a coherent state after an interruption in order to deliver a valid solution, or just a partial solution (when not all variables are instantiated). The memory allocation for the constraints is static: a global constraint model is built once and only parts of the model are made active and used at a given cyclical call. The memory consumed during the search is limited by using only restricted depth-first search or restricted best-first search.

Partial search methods are anytime algorithms [Zilberstein96] based on tree search methods having better quality profiles than the classical backtracking search algorithm. The main idea is to apply some arbitrary limits on the nodes visited in the tree search[2], depending on the behavior of the heuristics and on the remaining computation time. We distinguish four approaches: the iterative weakening methods (e.g. [Harvey&Ginsberg95]), the real-time search methods (e.g. [Korf90]), the iterative sampling methods (e.g. [Gomes.et.al98]) and the interleaving methods (e.g. [Meseguer97]). These methods use one or several *search schemes*[3]. The practical complexity of the search can be increasing, self-adjusting, or stable. In [Givry.et.al99], we propose the notion of *parameterized search* applied to one search scheme. The

[1] This work is partially funded by the EOLE project [Eole01].

[2] This description of partial search is compatible with the depth-first search principle. In [Perron99], partial search methods are based on the order of node exploration, which is memory consuming.

[3] A search scheme is a procedure which describes a search tree. For example, a combination of choice points.

parameters of the search limits are given explicitly. We can tune the degree of incompleteness of the search by varying the values of the parameters. A *tuning policy* indicates the relevant values of the parameters for different time contracts. In [Givry.et.al01b], we integrate the parameterized search approach into a *hybridization scheme* to express partial search based on several search schemes. The hybridization scheme is a sequence or an interleaving of parameterized searches. The searches can cooperate by exchanging solutions. A *time-sharing policy* specifies how to distribute the time contract to the searches.

Our constraint optimization framework is called ToOLS© (Templates Of On-Line Search). A search algorithm is expressed in ToOLS as the conjunction of four distinct components:

- A set of heuristics to rank every choice;
- A set of primitives to express a search scheme independent of any time limit; it is composed by predefined choice points and combinations of choice points as in the OPL language [Hentenryck99];
- A set of primitives to express the search limits that depend on the current node, the current path or the current sub-tree; the resulting parameterized search algorithm controls the size of the explored search tree defined by one search scheme;
- A temporal strategy defined by a hybridization scheme, i.e. a cooperation of several parameterized searches, dealing with time allocation and selecting the tuning strategy of the parameters (static tuning, iterative tuning or adaptive tuning).

A *template of search* defines an abstract component of a search algorithm that can be reused to speed up the development process of customized partial search algorithms. This framework makes it easier to try new combinations of search limits and new temporal strategies.

Experiments on the weapon allocation problem show that partial search algorithms significantly improve the solution quality compared to a traditional approach [Givry.et.al99] and also demonstrates the gain in development time of new customized search algorithms. The code is clearer and more concise when using the search primitives. Another application in the Telecom domain is currently tested in our framework [Eole01].

The hybridization scheme is a way to define specific local search methods, such as large neighborhood search based on a sequence of partial searches in different neighborhoods. Pure local search methods could also be introduced in our framework as a black-box used by the hybridization scheme. The temporal control could be enhanced by an on-line learning mechanism, using the fact that similar problems are repeatedly solved in a real-time system. [Crawford.et.al01] gives the base for this mechanism.

References

[Beldiceanu.et.al98] Beldiceanu, N., E. Bourreau, H. Simonis, and D. Rivreau (1998). Introduction de métaheuristiques dans CHIP. In Proc. of MIC-98.

[Crawford.et.al01] Lara S. Crawford, Markus P.J. Fromherz, Christophe Guettier, Yi Shang. A Framework for On-line Adaptive Control of Problem Solving. In Proc. of CP-2001 workshop on On-Line combinatorial problem solving and Constraint Programming, Paphos, Cyprus, December 2001.

[Givry.et.al99] Simon de Givry, Pierre Savéant, Jean Jourdan. Optimization combinatoire en temps limité : Depth first branch and bound adaptatif. In Proc. of JFPLC-99, pages 161-178, Lyon, France, 1999.

[Givry.et.al01a] S. de Givry, P. Gérard, J. Jourdan, J. Mattioli, N. Museux, P. Savéant. How does constraint technology meet industrial constraints ? In Proc. of ESA workshop on On-Board Autonomy, pages 189-200, Noordwijk, The Netherlands, 17-19 October 2001, 12p.

[Givry.et.al01b] S. de Givry, Y. Hamadi, J. Mattioli, M. Lemaître, G. Verfaillie, A. Aggoun, I. Gouachi, T. Benoist, E. Bourreau, F. Laburthe, P. David, S. Loudni, S. Bourgault. Towards an on-line optimization framework. In Proc. of CP-2001 workshop on On-Line combinatorial problem solving and Constraint Programming, Paphos, Cyprus, December 2001.
http://www.lcr.thomson-csf.com/projects/www_eole/workshop/olcp01-eole.ps

[Gomes.et.al98] C. Gomes, B. Selman, H. Kautz. Boosting Combinatorial Search Through Randomization. In Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98), pages 431--437, Madison, WI, USA, 1998.

[Eole01] RNRT EOLE project, http://www.lcr.thomson-csf.com/projects/www_eole.

[Harvey&Ginsberg95] William D. Harvey, Matthew L. Ginsberg. Limited discrepancy search. In Proc. of IJCAI-95, pages 607-613, Montréal, Canada, 1995.

[Hentenryck99] P. Van Hentenryck. OPL: The Optimization Programming Language. The MIT Press, Cambridge, Mass., 1999.

[Korf90] Richard E. Korf. Real-time heuristic search. Artificial Intelligence, 42:189-211, 1990.

[Laburthe98] François Laburthe. SaLSA: a language for search algorithms. In Proc. of CP-98, pages 310-324, Pisa, Italy, October 26-30 1998.

[Laburthe.et.al98] Laburthe, F., P. Savéant, S. de Givry, and J. Jourdan. Eclair: A library of constraints over finite domains. Technical Report ATS 98-2, Thomson-CSF LCR, Orsay, France, 1998.

[Meseguer97] Meseguer, P. (1997). Interleaved depth-first search. In Proc. of IJCAI-97, Nagoya, Japan, pp. 1382-1387.

[Platon01] PLATON Team (2001). Eclair reference manual, Version 6.0. Technical Report Platon-01.16, THALES Research and Technology, Orsay, France.

[Perron99] L. Perron. Search Procedures and Parallelism in Constraint Programming. In Proc. of CP-99, pages 346-360, Alexandria, Virginia, 1999.

[Zilberstein96] Shlomo Zilberstein. Using Anytime Algorithms in Intelligent Systems. AI Magazine, 17(3):73-83, 1996.