# G12 - Towards the Separation of Problem Modelling and Problem Solving

Mark Wallace and the G12 team

Monash University, Faculty of Information Technology,
Building H, Caulfield, Vic. 3800, Australia
mark.wallace@infotech.monash.edu.au
www.infotech.monash.edu.au/~wallace

**Abstract.** This paper presents the G12 large scale optimisation software platform, and discusses aspects of its architecture.

**Keywords:** constraint programming, modeling, optimization, software platform, search.

G12 is a software platform for solving combinatorial optimisation problems [SGM+05]. It was originally called a "constraint programming" platform, but we rather see it as an agnostic system which equally supports linear and mixed integer programming, constraint propagation and inference and a variety of other search and inference-based approaches for solving complex problems.

Problem modelling is separated in G12 as much as possible from problem solving. It is not (of course) our goal to automatically compile user-oriented problem models to highly efficient algorithms. Our target is to provide a software environment in which a user can initially write down a precise problem specification of his or her problem, without considering issues of computational efficiency. G12 supports the powerful 'Zinc' specification language [MNR+08], and the freely available 'MiniZinc' subset [NSB+07]. Subsequently a possibly different user can then add control information so as to guide the G12 system to exploit particular problem decompositions, inference techniques and search methods [BBB+08].

In recent years three different kinds of language have emerged for specifying and solving combinatorial optimisation problems. The first, most generic, languages are high-level modelling languages whose implementations include a number of solving techniques. These languages include mathematical programming languages, such as AMPL, constraint programming languages, such as CHIP and hybrid languages such as OPL. Such languages can be thought of as "80/20" languages which are designed to be able to handle many problems efficiently, but do not seek to cover all classes of problem.

Many problems require specialised solving techniques, which are not supported by the previous high-level languages. For these problems it is necessary to express specialised constraints and constraint behaviours, as well as specialised problem decompositions, and solver hybrids. Languages for expressing constraint behaviour have been particularly researched in the CP community. In particular

W.-J. van Hoeve and J.N. Hooker (Eds.): CPAIOR 2009, LNCS 5547, pp. 8–10, 2009.

we think of languages for writing propagators in Oz; attributed variables, suspensions, delayed goals, demons and action rules in languages such as SICStus Prolog, ECLiPSe and B-Prolog; and novel languages for encoding new global constraints by Beldiceanu, Pesant and others.

The third class of languages for solving complex problems are search languages supporting sophisticated combinations of branching, iteration, and improvement. These languages include Salsa, ToOLS and Comet.

A few years ago it when I was in the ECLiPSE team, we planned that the ECLiPSE language be broken down into three language subsets along these lines: a language for problem specification, a language for specifying constraint behaviour and a language for controlling search.

The G12 approach has deliberately taken quite a different path. Firstly G12 offers no language for specifying constraint behaviour. It supports a range of libraries for constraint propagation and solving, and interfaces which make it relatively straightforward to introduce new libraries at will [BGM+06]. New "global" constraints can also be easily interfaced to the system. However the G12 user cannot easily build new constraints with specialised constraint behaviours.

The G12 view is that efficiency is most effectively and easily achieved by mapping a problem down to an appropriate combination of solvers and search [PSWB08], rather than by adding new constraint implementations. Instead of a language for specifying constraint behaviours, therefore, G12 has a language for mapping problems to a form which can be evaluated efficiently [DDS08]. The problem modelling language is Zinc, and the language for mapping Zinc to a more efficient form is Cadmium. (The language used to run the resulting efficient algorithm is Mercury. The name G12 comes from the group in the periodic table which contains Mercury, Cadmium and Zinc.)

There is no additional language for expressing search in G12. Instead it supports two quite different ways of specifying the search method. The first way is as an extension to the Zinc search language [RMG+08]. A few generic search functions are available in Zinc, parameterised by Zinc functions. The philosophy behind this is that a Zinc problem model has a default behaviour, and the search function naturally belongs to Zinc both syntactically and semantically as a way of overruling its default behaviour. The second way that G12 supports search is that it can be built into certain solvers. G12 solvers have different capabilities: some can simply check for consistency of the current set of constraints, some can infer ("propagate") new constraints, some can optimise and some can even return candidate solutions. Search may be used in support of this last capability.

The G12 experiment is now gradually coming to fruition. G12 suports finite domain constraints, interval constraints over float variables, linear and mixed integer constraint solvers, propositional satisfiability solvers and search methods, BDD, set solvers and more. The Zinc and MiniZinc modelling languages are supported. The Cadmium mapping language is implemented - now in a fully general form so it can be applied to Zinc or any other modelling language for which a syntax and semantics are defined.

The first public release of G12 is due in early 2010, but already there are a number of major application projects in Australia for which G12 is the chosen software platform. We are excited to see G12 emerging at last from the laboratory.

## References


[BBB+08] Becket, R., Brand, S., Brown, M., Duck, G., Feydy, T., Fischer, J., Huang, J., Marriott, K., Nethercote, N., Puchinger, J., Rafeh, R., Stuckey, P., Wallace, M.: The many roads leading to Rome: Solving Zinc models by various solvers. In: Proc. ModRef: 7th International Workshop on Constraint Modelling and Reformulation (2008)

[BGM+06] Becket, R., de la Banda, M.G., Marriott, K., Somogyi, Z., Stuckey, P.J., Wallace, M.: Adding constraint solving to Mercury. In: Van Hentenryck, P. (ed.) PADL 2006. LNCS, vol. 3819, pp. 118–133. Springer, Heidelberg (2005)

[DDS08] Duck, G., De Koninck, L., Stuckey, P.: Cadmium: An implementation of ACD term rewriting. In: de la Banda, M.G., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 531–545. Springer, Heidelberg (2008)

[MNR+08] Marriott, K., Nethercote, N., Rafeh, R., Stuckey, P., Garcia de la Banda, M., Wallace, M.: The design of the Zinc modelling language. Constraints 13(3), 229–267 (2008)

[NSB+07] Nethercote, N., Stuckey, P., Becket, R., Brand, S., Duck, G., Tack, G.: MiniZinc: Towards a standard CP modelling language. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007)

[PSWB08] Puchinger, J., Stuckey, P., Wallace, M., Brand, S.: From high-level model to branch-and-price solution in G12. In: Perron, L., Trick, M.A. (eds.) CPAIOR 2008. LNCS, vol. 5015, pp. 218–232. Springer, Heidelberg (2008)

[RMG+08] Rafeh, R., Marriott, K., de la Banda, M.G., Nethercote, N., Wallace, M.: Adding search to Zinc. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202, pp. 624–629. Springer, Heidelberg (2008)

[SGM+05] Stuckey, P., de la Banda, M.G., Maher, M., Marriott, K., Slaney, J., Somogyi, Z., Wallace, M., Walsh, T.: The G12 project: Mapping solver independent models to efficient solutions. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 13–16. Springer, Heidelberg (2005)