JaCoP - Java Constraint Programming Solver

Kuchcinski, Krzysztof; Szymanek, Radoslaw

2013

JaCoP - Java Constraint Programming Solver

Krzysztof Kuchcinski[1] and Radosław Szymanek[2]

[1] Dept. of Computer Science, Lund University, Sweden
krzysztof.kuchcinski@cs.lth.se
[2] Crossing-Tech, Switzerland/Vistula University, Poland
radoslaw.szymanek@gmail.com

Abstract. This paper introduces JaCoP solver implemented entirely in Java. The solver offers a rich set of primitive, logical, conditional and global constraints as well as configurable search methods. The solver can be used directly from Java or through one of its front-end, Minizinc front-end or Scala DSL. The solver got three silver medals on Minizinc Challenge in the period 2010-2012.

Keywords: constraint solver, Java, Scala, Minizinc

1 Introduction

JaCoP solver [6] is Java-based open source solver developed and maintained mainly by the authors of this paper. Moreover, a number of students have contributed to the solver by programming versions of different global constraints and set constraints. The solver is being used in academia for research and teaching. The most successful use of the solver is within Electronic Design Automation community [1, 5, 9, 13, 18, 21–25], since the authors come from that community.

JaCoP provides a significant number of constraints to facilitate modeling as well as modular design of search. This allows tailoring model and search to characteristics of the problem being addressed. It has currently more than 90,000 lines of code, not including examples and testing code. The examples, which are the preferred way to document the abilities of JaCoP have more than 20.000 lines of code. The core developers have been working on JaCoP for the past ten years during their free time. It has been refactored, transformed, and improved many times. First versions of JaCoP were even three orders of magnitude slower than the current version. JaCoP implementation has been influenced heavily by more than 20 research articles. Moreover, JaCoP was used as a tool to conduct experiments for CP publications [4, 7, 8, 16, 19]. JaCoP supports finite domains of integers and sets of integers.

2 Constraints

The major focus of JaCoP are its constraints. These constraints include rich set of primitive, logical, and conditional constraints as well as many global constraints. The most important global constraints are as follows.

- diff2,

- cumulative,
- alldifferent (with bounds consistency and complete method based on Régin's algorithm),
- gcc,
- extensional support (with three different state-of-the-art approaches) and extensional conflict [7],
- among,
- element,
- circuit,
- bin packing,
- knapsack [8],
- regular,
- network flow [16], and
- geost [2].

The rich set of JaCoP global constraints includes also unique constraint, not available in other solvers, such as knapsack and network flow. Knapsack constraint defines an efficient filtering method for the standard problem that can be expressed as inequalities in integer programming but it execution time is expected to be sub-linear. Network flow constraint, on the other hand, offers a specialized constraint for a minimum-cost network flow problem. The problem is defined by a set of inequalities and the filtering method is based on specialized network simplex algorithm. In this way JaCoP offers a connection to MILP methods but makes them specialized and more efficient.

JaCoP has also graph constraint implementing graph and sub-graph isomorphism, clique, simple path and connected component global constraints. It is an experimental implementation but has been used in several projects [9, 21–23, 25]. This part is not provided under the open source licence.

3 Search

JaCoP offers classical depth first search method. It is, however, augmented with various ways of extending it. First, it offers a rich set of variable selection and value selection methods. these methods can be easily reprogrammed. Moreover, the value selection methods are not limited to selection of a single value. One can return a primitive constraint that will be used to build a search tree by selecting a branch constraint by the returned constrained and the other branch constrained by a negated constraint.

A unique feature of JaCoP is a search plug-in that can be defined by a user and, if registered, are called automatically during search. In this way, a user can influence the search in a number of ways. Based on this method we define search heuristics, such as credit search, and search visualization (generation of traces for CP-VIZ [15]).

The search functionality within JaCoP offers also methods for defining sequential and hierarchical search methods.

4 Front-ends

JaCoP solver contains also front-end for FlatZinc language that makes it possible to execute MiniZinc models. It allows us to perform extensive testing with the help of other solvers as we can compare results from different solvers.

The most recent addition is Scala based DSL (Domain Specific Language) and AMPL front-end [26]. Scala DSL makes it easier to create constraint programs in more intuitive manner. AMPL front-end opens JaCoP to new applications and encourages work on extending AMPL with CP extensions, such as global constraints.

5 Applications

JaCoP solver has been used in different areas. The distribution contains many examples of problems that can be solved with JaCoP, such as Golomb ruler, social golfer problem, car sequencing, perfect square placement, Langford's number problem, Steiner problem, and nonograms. More problems can be run in JaCoP using available models in Minizinc.

The authors used JaCoP mainly in the area of design automation of embedded systems where many optimization problems exist. The most advanced application of JaCoP, presented in [9, 22, 23], was related to the identification of computational patterns in programs and selection of special purpose instructions for ASIP's (Application Specific Instruction Processors). This work divides the problem into two sub problems. The first one focuses on identifying the most commonly occurring computational patterns with the help of sub-graph isomorphism constraints and connected component constraints to find the candidates for additional instructions. The second sub problem uses already mentioned global constraints for instruction selection and scheduling.

Recently, the solver was used in application mapping and communication routing for MPSoC's (Multi-Processor System on Chip) [10]. A parallel application has been mapped into an array of processors connected by a mesh interconnection net. In this work, minimum-cost network flow constraint has been used to model communications of an application. Using this method it is possible to consider different mappings alternatives, influence the decision process and select the best alternative.

JaCoP has also been used by other people in different context for solving problems encountered, for example, in control systems [11, 12], software engineering [3, 14], and electronic design automation [20].

6 Conclusions and Future Plans

JaCoP is an ongoing activity. There are several ongoing JaCoP projects. We have developed the first version of stochastic variable and related constraints in JaCoP. They can be used for handling uncertain information when modeling a problem. The first version of a SAT solver is also implemented and the future work includes its integration with JaCoP. The goal is to provide possibility to generate SAT clauses from JaCoP in a similar way to lazy clause generation [17]. Currently, our focus is on providing a simple Java API for JaCoP as well as integrate it well with industrial quality technologies like OSGi and Spring.

References

1. Arslan, M.A., Kuchcinski, K.: Instruction Selection and Scheduling for DSP Kernels on Custom Architectures. In: 16th EUROMICRO Conference on Digital System Design (Sep 2013)
2. Beldiceanu, N., Carlsson, M., Poder, E., Sadek, R., Truchet, C.: A generic geometrical constraint kernel in space and time for handling polymorphic k-dimensional objects. In: Proceedings of the 13th international conference on Principles and practice of constraint programming. pp. 180–194. CP'07, Springer-Verlag, Berlin, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1771668.1771686
3. Benavides, D., Segura, S., Trinidad, P., Ruiz-Cortés, A.: FAMA: Tooling a framework for the automated analysis of feature models. In: Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS). pp. 129–134 (2007)
4. Buddha, S.K.: Reasoning under uncertainty: Stochastic constraint programming using JaCoP. Tech. rep., Ecole polytechnique fédérale de Lausanne, Laboratory of Articial Intelligence (LIA) (2011)
5. Kuchcinski, K.: Constraints-driven scheduling and resource assignment. ACM Transactions on Design Automation of Electronic Systems (TODAES) 8(3), 355–383 (Jul 2003)
6. Kuchcinski, K., Szymanek, R.: JaCoP Library. User's Guide. http://www.jacop.eu (2013)
7. Lecoutre, C., Szymanek, R.: Generalized arc consistency for positive table constraints. In: In Proceedings of CP'06. pp. 284–298 (2006)
8. Malitsky, Y., Sellmann, M., Szymanek, R.: Filtering bounded knapsack constraints in expected sublinear time. In: 24th AAAI Conference on Artificial Intelligence (2010)
9. Martin, K., Wolinski, C., Kuchcinski, K., Floch, A., Charot, F.: Constraint programming approach to reconfigurable processor extension generation and application compilation. ACM Trans. on Reconfigurable Technology and Systems (TRETS) 5(2), 10:1–10:38 (Jun 2012), http://doi.acm.org/10.1145/2209285.2209289
10. Mirza, U.M., Gruian, F., Kuchcinski, K.: Design space exploration for streaming applications on multiprocessors with guaranteed service noc. In: submitted for publication (2013)
11. Nica, M., Peischl, B., Wotawa, F.: A constraint model for automated deployment of automotive control software. In: SEKE. pp. 899–904. Knowledge Systems Institute Graduate School (2008)
12. Oriol, M., Wahler, M., Steiger, R., Stoeter, S., Vardar, E., Koziolek, H., Kumar, A.: FASA: a scalable software framework for distributed control systems. In: Proceedings of the 3rd international ACM SIGSOFT symposium on Architecting Critical Systems. pp. 51–60. ISARCS '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2304656.2304664
13. Raffin, E., Wolinski, C., Charot, F., Kuchcinski, K., Guyetant, S., Chevobbe, S., Casseau, E.: Scheduling, binding and routing system for a run-time reconfigurable operator based multimedia architecture. In: Conference on Design and Architectures for Signal and Image Processing (DASIP). Edinburgh, UK (Oct 26-28, 2010 (Best Paper Award))
14. Regnell, B., Kuchcinski, K.: Exploring software product management decision problems with constraint solving - opportunities for prioritization and release planning. In: Software Product Management (IWSPM), 2011 Fifth International Workshop on. pp. 47–56 (2011)
15. Simonis, H., Davern, P., Feldman, J., Mehta, D., Quesada, L., Carlsson, M.: A generic visualization platform for CP. In: Proceedings of the 16th international conference on Principles and practice of constraint programming. pp. 460–474. CP'10, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1886008.1886048
16. Steiger, R., van Hoeve, W.J., Szymanek, R.: An efficient generic network flow constraint. In: SAC. pp. 893–900 (2011)

17. Stuckey, P.: Lazy clause generation: Combining the power of SAT and CP (and MIP?) solving. In: Lodi, A., Milano, M., Toth, P. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Lecture Notes in Computer Science, vol. 6140, pp. 5–9. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-13520-0_3
18. Szymanek, R., Catthoor, F., Kuchcinski, K.: Time-energy design space exploration for multi-layer memory architectures. In: Proc. of 7th ACM/IEEE Design, Automation and Test in Europe Conference. Paris, France (Feb 16–20, 2004)
19. Szymanek, R., Lecoutre, C.: Constraint-level advice for shaving. In: ICLP (2008)
20. Tseng, I.L., Postula, A.: Partitioning parameterized 45-degree polygons with constraint programming. ACM Trans. Des. Autom. Electron. Syst. 13(3), 1–29 (2008)
21. Wolinski, C., Kuchcinski, K.: Identification of application specific instructions based on subgraph isomorphism constraints. In: IEEE 18th Intl. Conference on Application-specific Systems, Architectures and Processors. Montréal, Canada (Jul 8-11, 2007)
22. Wolinski, C., Kuchcinski, K.: Automatic selection of application-specific reconfigurable processor extensions. In: Proc. Design Automation and Test in Europe. Munich, Germany (Mar 10-14, 2008)
23. Wolinski, C., Kuchcinski, K., Raffin, E.: Automatic design of application-specific reconfigurable processor extensions with UPaK synthesis kernel. ACM Trans. on Design Automation of Electronic Systems (TODAES) 15(1), 1–36 (2009)
24. Wolinski, C., Kuchcinski, K., Teich, J., Hannig, F.: Area and reconfiguration time minimization of the communication network in regular 2D reconfigurable architectures. In: Proc. of the International Conference on Field Programmable Logic and Applications (FPL). Heidelberg, Germany (Sep 8-10, 2008)
25. Wolinski, C., Kuchcinski, K., Martin, K., Floch, A., Raffin, E., Charot, F.: Graph constraints in embedded system design. In: Proc. Workshop on Combinatorial Optimization for Embedded System Design, collocated to CPAIOR conference. Bologna, Italy (June 15, 2010)
26. Zverovich, V.: AMPL interface to JaCoP (2013), https://github.com/vitaut/ampl/tree/master/solvers/jacop