# Sat4j-CSP

**XCSP3 Competition – 2018**

Daniel Le Berre and Emmanuel Lonca

CRIL – Univ. Artois and CNRS
lastname@cril.fr

## 1 Solver Description

Sat4j[1] is a constraint satisfaction and optimization library developed in Java. It is a mature project (Sat4j was born in 2004) included since June 2008 in the Eclipse IDE as part of its plugin dependency management engine. As indicated by its name, the original library design was built on top of a SAT solver. Taking advantage of the this initial solver, several new kinds of solvers were added to the library (pseudo-Boolean solver, MAXSAT solver, ...) including a CSP solver.

Sat4J-CSP3, the version submitted to the 2018 XCSP3 Competition[1], is far more than an improvement to Sat4j-CSP2[4] giving it the ability to read XCSP3 input files. While Sat4j-CSP2 was intended to handle the most common constraints (extension, intension, allDifferent), the current version was built with the idea to handle the whole set of constraints proposed by the specifications of the XCSP3-core instance format[2]. Since, for most constraints, we had to start from scratch, we just translated most of the constraints into intension ones, taking advantage of the instance parser provided by Christophe Lecoutre[2].

The intension constraints encoder developed for Sat4j-CSP2 was simply evaluating each constraint using javascript (Rhino library) and generated the whole set of nogoods for each of them. Although it was sufficient for instances with intension constraints considering few variables with small domains, the encoding of global constraints into intension constraints made the encoder absolutely unefficient. We thus developed a new intension constraint encoder, based on the principle of the Tseitin encoding:

1. from a constraint, we build a tree where the nodes are labeled with operators, and leaves with variables or constants;
2. each node is associated with a new integer variable giving its value (0 or 1 for nodes labeled with Boolean operators);
3. considering the nodes from the leaves to the root, the mapping between the values of its children and its own value is encoded using Boolean constraints;
4. the value of the root is enforced to a non-zero value. Concerning the optimization of objectives in functional forms, the same algorithm is used except for the last step: instead of enforcing the tree root value, the variable it is associated with is set as the objective function.

[1] http://xcsp.org/competition

## 2 Future Work

Sat4j-CSP3 was developed with the aim to handle the whole set of constraints allowed in XCSP3-core. However, at this step, there is room for improvements. First, the encoding of integer variables uses one binary variable for each value in its range. For arrays of integer variables taking their values in large domains, this results in the declaration of a huge amount of Boolean variables, making the solver running out of memory. A slight adaptation of some encodings used for At-Most-1 constraints (like the binary encoding or the Two-Product encoding[3]) may help in terms of efficiency to keep the memory consumption under the system limits.

At this time, we did not develop any specific constraint encodings. We translated most of the global constraints into intension ones, so we do not take advantage of known encodings of these constraints. In addition, new integer variable encodings (as described in the beginning of this section) may bring some efficiency tracks to intension constraints encoding.

Finally, we plan to support several multicriteria optimization processes. This is possible because Sat4j has such capabilities for some families of Boolean functions.

## 3 Acknowledgments

This work has been supported by the project CPER DATA from the "Hauts-de-France" Region.

## References

1. Berre, D.L., Parrain, A.: The sat4j library, release 2.2. JSAT **7**(2-3), 59–6 (2010), https://satassociation.org/jsat/index.php/jsat/article/view/82
2. Boussemart, F., Lecoutre, C., Piette, C.: XCSP3: an integrated format for benchmarking combinatorial constrained problems. CoRR **abs/1611.03398** (2016), http://arxiv.org/abs/1611.03398
3. Chen, J.: A new sat encoding of the at-most-one constraint. Proc. Constraint Modelling and Reformulation (2010)
4. Le Berre, D., Lynce, I.: Csp2sat4j: a simple csp to sat translator. Proceedings of the 2nd International CSP Solver Competition pp. 43–54 (2008)