The Journal of Open Source Software

RealPaver 1.1: A C++ Library for Constraint Programming over Numeric or Mixed Discrete-Continuous Domains

Raphaël Chenouard [1*¶] **and Laurent Granvilliers** [1*]

1 Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, Nantes, France ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.09331

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @cvanaret
- @dannys4
- @victorapm

Submitted: 20 May 2025
Published: 16 February 2026

Summary

Constraint Programming (CP) is a paradigm for solving constraint satisfaction and optimization problems (Rossi et al., 2006). Although CP mainly addresses combinatorial problems, it can also handle continuous problems by approximating real numbers with intervals (Benhamou & Granvilliers, 2006).

RealPaver is a C++ library for CP over numeric or mixed discrete-continuous domains. Constraint Satisfaction Problems (CSPs) can be described either in C++ with the API or in a text file using the syntax of RealPaver specific language. Then, they can be solved using the C++ API or using the CSP solver from the command line. The CSP solver is pre-configured, but various parameters can be modified in another text file.

With respect to the first version of the software developed twenty years ago (Granvilliers & Benhamou, 2006), this new library incorporates new types of variables and constraints, new algorithms, a clean object-oriented architecture, the management of parameters, Meson Build as build engine (Pakkanen, 2025), an interface with third-party softwares, and a C++ API.

Statement of need

CP associates a rich modeling language with powerful solving techniques. The main algorithm behind the RealPaver solver is branch-and-prune (B&P), which implements a complete search to find all the solutions of a given problem (Chabert & Jaulin, 2009; Van Hentenryck et al., 1997). The branching component separates a problem into easier-to-solve sub-problems. The pruning or contracting component aims at reducing the region delimited by a sub-problem. RealPaver relies on the GAOL interval library (Goualard, 2020) to ensure rigorous computations.

This technology has been applied with success in many fields of engineering like automatic control (Jaulin et al., 2001), preliminary design (Yvars & Zimmer, 2021), and robotics (Merlet, 2004).

The set of solutions of a numerical CSP or a mixed discrete-continuous problem can be rigorously bounded, even in the presence of roundoff errors. All the solutions of continuous nonlinear equations or inequations can be automatically separated and rigorously bounded. The B&P algorithm with interval analysis (Moore et al., 2009) can provide proof of infeasibility or proof of existence of solutions.

State of the field

Various solvers on discrete domains exist in CP, like OR-tools and Chuffed, but they are not able to handle continuous variables. The Ibex ("Ibex-Lib," 2025) library is based on a similar branch-and-contract framework to that of RealPaver, but RealPaver proposes some new types of constraints such as table constraints, conditional constraints, and piecewise constraints, and the native definition of discrete variables (integer, enumeration of values). For pure continuous problems, RealPaver achieves performances equivalent to Ibex. Regarding other mixed integer non-linear solvers like Couenne or Baron, they only solve optimization problems. These solvers are not able to compute several solutions for CSPs, even by adding a constant objective.

Brief overview

Modeling language

There are three types of variables:

- A boolean variable has domain $\{0, 1\}$
- An integer variable can take values from a set of integers
- A real variable lies in a union of intervals

The language defines several types of constraints:

- An arithmetic constraint involves the usual operations over the reals and relations from the set $\{=, \leq, \geq\}$
- $G \to B$ is a conditional constraint, where a constraint B (body) is activated when a constraint G (guard) holds true
- $table(X, S)$ is a table constraint, where X is a vector of variables and S is a set of valid assignments for X
- $piecewise(x, \{I_k \to C_k\}_k)$ is a piecewise constraint where a constraint C_k is activated when the variable x lies in the interval I_k

Solving strategies

The B&P algorithm creates a search tree by recursively dividing the initial region, i.e., the Cartesian product of variable domains. Each solving step applies a pruning of domains based on a propagation of contractors:

- the HC4 or BC4 operators (Benhamou et al., 1999)
- the ACID algorithm (Neveu et al., 2015)
- the interval Newton operator for nonlinear systems of equations (Moore et al., 2009)
- linear methods applied to affine or Taylor relaxations of nonlinear problems (Ninin et al., 2015; Trombettoni et al., 2011)
- specific algorithms for the non-arithmetic or global constraints

It uses a search strategy responsible for the selection of the next node to explore (depth-first search, breadth-first search, and distant-most-depth-first search (Chenouard et al., 2009)) and the selection of a variable in this node defining the domain to be split (e.g., the largest domains or the greatest impacts on the constraints (Trombettoni et al., 2011)), hence generating sub-nodes.

Parameters and RealPaver customization

RealPaver integrates classes to handle three types of parameters: double-valued, integer-valued, and string-valued parameters.

All existing parameters, with their default value, are defined in the class `Params`. This class organizes them using 10 categories that cover all the aspects of the library.

Moreover, the section about the parameters in the documentation (processed by MkDocs) is generated from the default parameter file, using the Python script `doc/gen_params_doc.py`.

Building system and requirements

The meson build system is used to orchestrate the configuration, the building of the library, and the generation of `rp_solver` (the CSP solver executable). The user can select one of the supported linear solving libraries (Coin-or CLP, HiGHS, SoPlex, and Gurobi) and can activate assertions, logging, or the generation of the documentation, directly as meson command line options.

The current building system does not install dependencies or third party softwares. The user has to install, on their own, the GAOL interval library (Goualard, 2020) and one of the supported linear solving libraries, as well as MkDocs if the building of the documentation is activated.

Running the solver and getting the solutions

Using the C++ API, one can use the `CSPSolver` class and call the `solve` method after having created a problem instance. Using the command-line executable `rp_solver`, one can solve problems written in a text file following the RealPaver language syntax:

```
rp_solver my_problem.rp -p params.txt
```

The `-p` is optional and allows customizing the parameters using a text file (here `params.txt`). By default, the summary of the solving process and all computed solutions will be stored in a text file, automatically named from the base file name, `my_problem.sol` in this example. A brief report is also displayed in the console, with the processed files, pre-processing summary, and solving summary (time, number of solutions, solving status, number of nodes in the search tree, and number of pending nodes when ending with a partial solving).

References

Benhamou, F., Goualard, F., Granvilliers, L., & Puget, J.-F. (1999). Revising hull and box consistency. *ICLP 1999, Proceedings of the 16th International Conference on Logic Programming, Cambridge, MA, USA*, 230–244. https://doi.org/10.7551/mitpress/4304.003.0024

Benhamou, F., & Granvilliers, L. (2006). *Continuous and interval constraints* (F. Rossi, P. van Beek, & T. Walsh, Eds.; Vol. 2, pp. 571–603). Elsevier. https://doi.org/10.1016/S1574-6526(06)80020-9

Chabert, G., & Jaulin, L. (2009). Contractor programming. *Artificial Intelligence*, *173*, 1079–1100. https://doi.org/10.1016/j.artint.2009.03.002

Chenouard, R., Goldsztejn, A., & Jermann, C. (2009). Search strategies for an anytime usage of the branch and prune algorithm. In C. Boutilier (Ed.), *IJCAI 2009, proceedings of the 21st international joint conference on artificial intelligence, pasadena, california, USA, 2009* (pp. 468–473). http://ijcai.org/Proceedings/09/Papers/085.pdf

Goualard, F. (2020). GAOL (not just another interval library). In *GitHub repository*. https://github.com/goualard-f/GAOL; GitHub.

Granvilliers, L., & Benhamou, F. (2006). Algorithm 852: RealPaver: An interval solver using constraint satisfaction techniques. *ACM Trans. Math. Softw.*, *32*(1), 138–156. https://doi.org/10.1145/1132973.1132980

Ibex-lib. (2025). In *GitHub repository*. https://github.com/ibex-team/ibex-lib; GitHub.

Jaulin, L., Kieffer, M., Didrit, O., & Walter, É. (2001). *Applied interval analysis with examples in parameter and state estimation, robust control and robotics*. Springer. https://doi.org/10.1007/978-1-4471-0249-6

Merlet, J.-P. (2004). Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis. *The International Journal of Robotics Research*, *23*(3), 221–235. https://doi.org/10.1177/0278364904039806

Moore, R. E., Kearfott, R. B., & Cloud, M. J. (2009). *Introduction to interval analysis*. SIAM. https://doi.org/10.1137/1.9780898717716

Neveu, B., Trombettoni, G., & Araya, I. (2015). Adaptive constructive interval disjunction: Algorithms and experiments. *Constraints*, *20*(4), 452–467. https://doi.org/10.1007/s10601-015-9180-3

Ninin, J., Messine, F., & Hansen, P. (2015). A reliable affine relaxation method for global optimization. *OR: A Quarterly Journal of Operations Research*, *13*(3), 247–277. https://doi.org/10.1007/s10288-014-0269-0

Pakkanen, J. (2025). The Meson build system. In *GitHub repository*. https://github.com/mesonbuild/meson; GitHub.

Rossi, F., van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of constraint programming* (Vol. 2). Elsevier. https://doi.org/10.1016/S1574-6526(12)70003-2

Trombettoni, G., Araya, I., Neveu, B., & Chabert, G. (2011). Inner regions and interval linearizations for global optimization. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 99–104. https://doi.org/10.1609/aaai.v25i1.7817

Van Hentenryck, P., McAllester, D., & Kapur, D. (1997). Solving polynomial systems using a branch and prune approach. *SIAM Journal on Numerical Analysis*, *34*(2), 797–827. https://doi.org/10.1137/S0036142995281504

Yvars, P.-A., & Zimmer, L. (2021). Integration of constraint programming and model-based approach for system synthesis. *IEEE International Systems Conference, SysCon 2021, Vancouver, BC, Canada, April 15 - May 15, 2021*, 1–8. https://doi.org/10.1109/SYSCON48628.2021.9447096