# CPMpy dataset submissions to PB26

Tias Guns
*Department of Computer Science*
*KU Leuven*
Leuven, Belgium
tias.guns@kuleuven.be

Thomas Sergeys
*Department of Computer Science*
*KU Leuven*
Leuven, Belgium
thomas.sergeys@kuleuven.be

Ignace Bleukx
*Department of Computer Science*
*KU Leuven*
Leuven, Belgium
ignace.bleukx@kuleuven.be

Hendrik Bierlee
*Department of Computer Science*
*KU Leuven*
Leuven, Belgium
henk.bierlee@kuleuven.be

***Abstract*—CPMpy [5] is an open-source Python library for modeling combinatorial (optimisation) problems. It provides a high-level modeling interface and connects to solver backends from various CO paradigms. Solvers don't all support the same types of decision variables and constraints. Through a series of equivalence-preserving constraint reformulations, CPMpy can transform a constraint model into the target format, enabling systematic comparisons of solvers across paradigms. Thanks to a collection of file format readers and loaders, well-known CO benchmarking and application datasets can be brought into CPMpy and solved with one of its backends. Combined with our recent work on format writers, CPMpy can serve as a translation tool, converting datasets to different formats and making them accessible to other communities [11]. For the PB26 competition we translated CP models for the JSPLib, PSPLib RCPSP and MIPLib datasets to *.opb* files.**

**License** : CPMpy is licensed under the Apache-2.0 license
**Code** : CPMpy's code is available on GitHub: https://github.com/CPMpy/cpmpy
**Acknowledgement** : This research received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant No. 101002802, CHAT-Opt) and Proof of Concept (POC) grant (Grant No.101158376, PO-CPMpy).

## I. Format loaders

In the recent work of [11], we introduced a collection of file format readers and loaders to CPMpy. These tools can convert well-known benchmarking instances into CPMpy models, giving access to our constraint reformulation stack; an extensive set of rewrite operations such as constraint decomposition, linearization, and Boolean encodings, that allow transforming between different constraint formalisms.

Application datasets like JSPLib and PSPLib only contain problem data. Since they do not explicitly define any constraints, there are many design choices to make regarding the model. The provided modelling contains one set of choices, and can be exchanged for alternatives.

## II. Constraint reformulations

Solvers from different CO paradigms support different types of decision variables and constraints. Constraint Programming (CP) has the most expressive language we consider, supporting global constraints [12], potentially non-linear, over both integer and Boolean decision variables. Next are Integer Linear Programming (ILP) solvers supporting only linear inequality constraints, Pseudo-Boolean (PB) solvers supporting linear inequality constraints over Boolean variables, and (Max)SAT-solvers which support Boolean formulas in Conjunctive Normal Form (CNF).

Solvers from the more expressive paradigms can take less expressive languages as input, but the reverse direction requires rewriting a set of high-level expressions into a set of lower level expressions, often requiring the use of auxiliary variables. This is achieved through a series of equivalence-preserving constraint reformulations, transforming a constraint model into the target format. We identify five main reformulation stages: decomposition, flattening, linearization, encoding integer variables to Booleans, and encoding pseudo-Boolean constraints into CNF. Each of the transformations is implemented in CPMpy as a separate function.

## III. File writing

To convert any CPMpy model back to file, we provide a collection of file format writers. The currently supported formats are MPS, OPB and DIMACS; an overview is given in Table I. Through the SCIP Python interface we additionally support LP, CIP, GMS, PIP, and MiniZinc files. The file writers make use of the same set of transformations to translate any generic CPMpy model into expressions matching the required output language.

During reformulation of the input model, some variables are encoded and auxiliary variables are introduced. For example, integer decision variables are encoded to Boolean literals when the target format is DIMACS or OPB. To allow for checking validity of the solution found by the solver, we augment the output files with the *semantics* of an auxiliary variable. More specifically, we adopt the VeriPB-style annotations to indicate a literal is part of the encoding of an integer variable. Such annotations allow to reconstruct the value of the original

TABLE I: Initial set of supported datasets and I/O capabilities

| Format | Dataset | Read | Write |
|---|---|---|---|
| jsplib | JSPLib [7] | ✓ | |
| psplib | PSPLib RCPSP [8] | ✓ | |
| nurserostering | Nurse rostering (1–24) [10] | ✓ | |
| XCSP3 | XCSP3 competition [1] | ✓ | |
| OPB | PB Competition [9] | ✓ | ✓ |
| MSE (wcnf) | MaxSAT Evaluation [2] | ✓ | ✓ |
| MPS | MIPLib [4] | ✓ | ✓ |
| DIMACS (cnf) | SAT competition [6] | ✓ | ✓ |

integer variables after solving, and check the solution with the specification of the high-level model.

## IV. Datasets

Our recent work also added dataset classes to CPMpy, with a PyTorch compatible access pattern, for a collection of well-known CO benchmarks [11]. Such a dataset object can be used to download, transform, and iterate over problem instances. Combined with CPMpy's constraint reformulations, expressions can be translated to the Python API of many different solvers spanning CP, MIP, SMT, pseudo-Boolean and SAT technologies, enabling cross-paradigm benchmarking. When combined with the format writers, we can translate datasets to other file formats from potentially different CO solving paradigms.

We submitted instances from the following translated application datasets to MSE26:

- `JSPLib` [7] Benchmark of job-shop scheduling problems
- `PSPLib RCPSP` [8] Benchmark of Resource-Constrained Project Scheduling Problems
- `MIPLib` [4] The Mixed Integer Programming Library

### A. JSPLib

JSPLib [7] is a collection of benchmark instances for the classical Job Shop Scheduling Problem (JSSP), in which a set of jobs must be processed on a set of machines subject to precedence and machine-capacity constraints. Each job consists of an ordered sequence of operations, where every operation requires exclusive use of a specified machine for a fixed processing time. The standard objective for these instances is the minimisation of the makespan. The collection aggregates several well-known benchmark families from the scheduling literature. Instance names typically encode the originating benchmark family and an instance identifier, for example `ft06`, `la21`, `orb01`, `swv01`, or `ta80`.

### B. PSPLib RCPSP

PSPLib [8] is a benchmark library for project scheduling problems, including the single-mode Resource-Constrained Project Scheduling Problem (RCPSP). In the RCPSP, project activities must be scheduled subject to precedence constraints and limited renewable resources, with the objective of minimising the total project duration. The benchmark instances were generated systematically using the ProGen project generator [3] and are characterized by input parameters controlling structural and resource-related properties of the projects, such as resource factor, resource strength, and network complexity. The standard single-mode RCPSP benchmark sets are commonly denoted `j30`, `j60`, `j90`, and `j120`, where the number indicates the number of non-dummy project activities.

### C. MIPLib

MIPLib [4] is a benchmark library of both pure and mixed-integer programs. It contains problems arising from both academic and real-world applications, and is widely used to evaluate and compare the performance of MIP solvers. The library includes instances of varying size and difficulty.

Due to the large number of instances included in these datasets, we restricted our submission to a subset of the problem families and instances. We solved each instance using CPMpy with the default OR-Tools CP-SAT as solver, and selected a subset with varying solve-time requirements; both easy and hard instances to evaluate the scaling rules of different solvers.

## References


[1] Gilles Audemard, Christophe Lecoutre, and Emmanuel Lonca. Proceedings of the 2024 xcsp3 competition, 2024. URL: https://arxiv.org/abs/2412.00117, arXiv:2412.00117.

[2] Jeremias Berg, Matti Järvisalo, Ruben Martins, Andreas Niskanen, and Tobias Paxian. Maxsat evaluation 2024: Solver and benchmark descriptions, 2024.

[3] Andreas Drexl, Rüdiger Nissen, James H. Patterson, and Frank Salewski. Progen/πx – an instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research*, 125(1):59–72, 2000. URL: https://www.sciencedirect.com/science/article/pii/S0377221799002052, doi:10.1016/S0377-2217(99)00205-2.

[4] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, 2021. doi:10.1007/s12532-020-00194-3.

[5] Tias Guns. Increasing modeling language convenience with a universal n-dimensional array, cppy as python-embedded example. In *Proceedings of the 18th workshop on Constraint Modelling and Reformulation at CP (Modref 2019)*, volume 19, 2019.

[6] Marijn JH Heule, Markus Iser, Matti Järvisalo, and Martin Suda. Proceedings of sat competition 2024: Solver, benchmark and proof checker descriptions, 2024.

[7] Taemyung Kim. JSPLIB: Job shop scheduling problem library. https://github.com/tamy0612/JSPLIB, 2016.

[8] Rainer Kolisch and Arno Sprecher. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European journal of operational research*, 96(1):205–216, 1997.

[9] Pseudo-Boolean Competition 2024 (PB24). https://www.cril.univ-artois.fr/PB24/, 2024.

[10] Shift Scheduling Benchmark Instances — schedulingbenchmarks.org. https://schedulingbenchmarks.org/. [Accessed 14-05-2026].

[11] Thomas Sergeys, Ignace Bleukx, and Tias Guns. Unified programmatic access to co benchmarks, to connect constraint solving communities. In *Proceedings of the 29th International Conference on Theory and Applications of Satisfiability Testing (SAT 2026)*, 2026. Accepted for publication.

[12] Willem-Jan van Hoeve and Irit Katriel. Global constraints. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 169–208. Elsevier, 2006. doi:10.1016/S1574-6526(06)80010-6.