PB-OLL-RS and MIXED-BAG in Pseudo-Boolean Competition 2024

Christoph Jabs, Jeremias Berg, Matti Järvisalo
Department of Computer Science, University of Helsinki, Finland
{christoph.jabs, jeremias.berg, matti.jarvisalo}@helsinki.fi

I. INTRODUCTION

We give a brief overview of the pseudo-Boolean optimization solvers PB-OLL-RS and MIXED-BAG as they competed in Pseudo-Boolean Competition 2024. Both solvers support linear constraints and objectives with arbitrarily large coefficients.

The source code for both solvers is available at https://bitbucket.org/coreo-group/pbcomp24-cg.

II. PB-OLL-RS

PB-OLL-RS (for "pseudo-Boolean OLL on top of RoundingSat") is a core-guided pseudo-Boolean optimization solver. The implementation employs RoundingSat [1][1] as a decision oracle for unsatisfiable core extraction. However, while RoundingSat also includes an implementation of core-guided algorithms PB optimization [2], the implementation of OLL in PB-OLL-RS is mostly new code (with a rough estimate of 80% of the code written from scratch).

Algorithmically, compared to the core-guided approach implemented originally in RoundingSat, PB-OLL-RS includes (i) more techniques that have proven effective in core-guided MaxSAT solving, as well as (ii) a generalization of the cardinality core reformulations known from MaxSAT to pseudo-Boolean reformulations. For point (i), PB-OLL-RS employs weight-aware core extraction [3], heuristic core minimization (i.e., core trimming) [4], and detection and reformulation of intrinsic at-most-one constraints [4]. For point (ii), after obtaining a core from RoundingSat and reducing it to a cardinality constraint, instead of performing so-called clause cloning [3], where new relaxation variables with the smallest weight of the literals in the core are introduced, and some core literals might remain in the reformulated objective, we build a *pseudo-Boolean* reformulation assigning each core literal its weight in the objective. This way all literals appearing the core are immediately removed from the objective and clause cloning is not required. Since such a reformulation might introduce exponentially many new variables and computing the weight of each new variable requires computing the subset sums of the coefficients, we heuristically decide when to perform build a pseudo-Boolean reformulation and when to perform clause cloning. At the end of each weight-aware core extraction phase we estimate how many new variables actually need to be lazily added and only choose a pseudo-Boolean reformulation if we expect to add only few new variables.

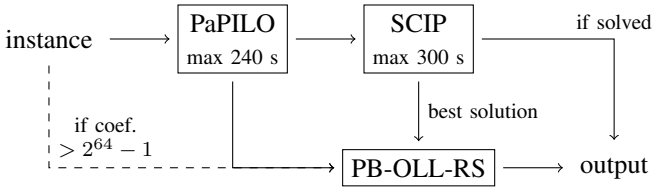

Fig. 1. Illustration of the building blocks and their interaction in the MIXED-BAG solver.

III. MIXED-BAG

The MIXED-BAG solver, as the name suggests, combines PB-OLL-RS with the preprocessor/presolver PaPILO [5] (version 2.2.1) and the integer programming solver SCIP [6] (version 9.0.1). PaPILO and SCIP are executed in their default configurations with the only change being that the presolving techniques for fixing continuous variables, dual inference, substitution, and sparsification are turned off. The interaction of these components is illustrated in Figure 1. We first run PaPILO for at most 240 seconds, followed by SCIP on the instance simplified by PaPILO for at most 300 seconds. If neither PaPILO nor SCIP solve the instance to optimum, we run PB-OLL-RS on the instance simplified by PaPILO for the remaining time. In addition executing PB-OLL-RS on the instance simplified by PaPILO, we also make use of the best solution found by SCIP within its time limit as an upper bound (introducing an upper-bounding constraint on the objective and performing hardening) within PB-OLL-RS. As a special case, if the original input instance contains any coefficients larger than $2^{64} - 1$, PB-OLL-RS is directly executed to avoid numerical errors. In addition to the instance simplified by PaPILO, PB-OLL-RS the best solution found by SCIP within its time limit as an upper bound.

Work financially supported by Research Council of Finland under grants 362987 and 356046.

[1]https://gitlab.com/MIAOresearch/software/roundingsat, commit c548e109 from 15.11.2023

REFERENCES

[1] J. Elffers and J. Nordström, "Divide and conquer: Towards faster pseudo-boolean solving," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden* (J. Lang, ed.), pp. 1291–1299, ijcai.org, 2018.

[2] J. Devriendt, S. Gocht, E. Demirovic, J. Nordström, and P. J. Stuckey, "Cutting to the core of pseudo-boolean optimization: Combining core-guided search with cutting planes reasoning," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021*, pp. 3750–3758, AAAI Press, 2021.

[3] J. Berg and M. Järvisalo, "Weight-aware core extraction in SAT-based MaxSAT solving," in *Principles and Practice of Constraint Programming—23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 – September 1, 2017, Proceedings* (J. C. Beck, ed.), vol. 10416 of *Lecture Notes in Computer Science*, pp. 652–670, Springer, 2017.

[4] A. Ignatiev, A. Morgado, and J. Marques-Silva, "RC2: an efficient MaxSAT solver," *J. Satisf. Boolean Model. Comput.*, vol. 11, no. 1, pp. 53–64, 2019.

[5] A. M. Gleixner, L. Gottwald, and A. Hoen, "PaPILO: A parallel presolving library for integer and linear optimization with multiprecision support," *INFORMS J. Comput.*, vol. 35, no. 6, pp. 1329–1341, 2023.

[6] T. Achterberg, "SCIP: solving constraint integer programs," *Math. Program. Comput.*, vol. 1, no. 1, pp. 1–41, 2009.