Learn to Relax: Integrating 0–1 Integer Linear Programming with Pseudo-Boolean Conflict-Driven Search

Jo Devriendt[1,3] and Ambros Gleixner[2], and Jakob Nordström[3,1]

[1] Lund University, Lund, Sweden
jo.devriendt@cs.lth.se
[2] Zuse Institute Berlin, Berlin, Germany
gleixner@zib.de
[3] University of Copenhagen, Copenhagen, Denmark
jn@di.ku.dk

Conflict-driven pseudo-Boolean (PB) solvers optimize 0–1 integer linear programs by generalizing the conflict-driven clause learning (CDCL) paradigm [3, 12, 14] from SAT solving. Some PB solvers essentially encode the input back to CNF and run CDCL [8, 13, 15], but another approach, which is our focus in this work, is to extend the solvers from CNF to reason natively with linear constraints [5, 10, 11, 16]. Such solvers have the potential to run exponentially faster than CDCL solvers, since the cutting planes method [6] they use is exponentially stronger than the resolution method underlying CDCL [4]. In practice, however, PB solvers can sometimes get hopelessly stuck even in parts of the search space where the linear programming (LP) relaxation of the residual problem is infeasible [9].

Inspired by mixed integer programming (MIP), we address this problem by interleaving incremental LP solving with the conflict-driven pseudo-Boolean search. Our integration is fully dynamic, with the PB and LP solvers communicating continuously during execution. In order to balance resources and avoid that the LP solver starves the PB solver, LP calls are made with a strict time budget and are terminated as soon as this budget is exceeded. If the LP solver detects infeasibility, we use Farkas' lemma to combine existing constraints into a new linear constraint that can serve as the starting point of pseudo-Boolean conflict analysis. When the LP solver instead finds a rational solution, we generate Gomory cuts that prune away this solution and tighten the search space both on the PB and the LP side. The PB solver can also use information from the rational solution to direct the search, e.g., by determining how to assign variables, and we also explore passing constraints learned during conflict analysis from the PB solver to the LP solver. To the best of our knowledge, this is the first time techniques from MIP solving such as LP relaxations and cut generation have been combined with full-blown pseudo-Boolean conflict analysis, which learns new linear inequalities by operating directly on the linear constraints (rather than applying resolution on clauses derived from such constraints, as has been done previously in MIP and constraint programming solvers in, e.g., [1, 7]).

We report on extensive experiments with a combined solver integrating the LP solver *SoPlex* [17] (part of the MIP solver *SCIP* [2]) with the pseudo-Boolean solver *RoundingSat* [10]. Although we believe that there is ample room for further improvements, this hybrid approach already exhibits significantly improved performance on a wide range of benchmarks, approaching a “best of two worlds'” scenario between SAT-style conflict-driven search and MIP-style branch-and-cut.

References

1. Achterberg, T.: Conflict analysis in mixed integer programming. Discrete Optim. **4**(1), 4–20 (2007)
2. Achterberg, T., Berthold, T., Koch, T., Wolter, K.: Constraint integer programming: a new approach to integrate CP and MIP. In: Proceedings of the 5th International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2008), pp. 6–20 (2008)
3. Bayardo Jr., R.J., Schrag, R.: Using CSP look-back techniques to solve real-world SAT instances. In: Proceedings 14th National Conference on Artificial Intelligence (AAAI 1997), pp. 203–208 (1997)
4. Beame, P., Kautz, H., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. J. Artif. Intell. Res. **22**, 319–351 (2004)
5. Chai, D., Kuehlmann, A.: A fast pseudo-Boolean constraint solver. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **24**(3), 305–317 (2005)
6. Cook, W., Coullard, C.R., Turán, G.: On the complexity of cutting-plane proofs. Discrete Appl. Math. **18**(1), 25–38 (1987)
7. Downing, N.R.: Scheduling and Rostering with Learning Constraint Solvers. Ph.D. thesis, University of Melbourne (2016)
8. Eén, N., Sörensson, N.: Translating pseudo-Boolean constraints into SAT. J. Satisf. Boolean Model. Comput. **2**(1-4), 1–26 (2006)
9. Elffers, J., Giráldez-Cru, J., Nordström, J., Vinyals, M.: Using combinatorial benchmarks to probe the reasoning power of pseudo-Boolean solvers. In: Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT 2018), pp. 75–93 (2018)
10. Elffers, J., Nordström, J.: Divide and conquer: Towards faster pseudo-Boolean solving. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018), pp. 1291–1299 (2018)
11. Le Berre, D., Parrain, A.: The Sat4j library, release 2.2. J. Satisf. Boolean Model. Comput. **7**, 59–64 (2010)
12. Marques-Silva, J.P., Sakallah, K.A.: GRASP: a search algorithm for propositional satisfiability. IEEE Trans. Comput. **48**(5), 506–521 (1999)
13. Martins, R., Manquinho, V., Lynce, I.: Open-WBO: a modular MaxSAT solver,. In: Sinz, C., Egly, U., (eds) SAT 2014. LNCS, vol. 8561, pp. 438–445. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09284-3_33
14. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: Proceedings of the 38th Design Automation Conference (DAC 2001), pp. 530–535 (2001)