NaPS: Nagoya pseudo-Boolean solver

Masahiko Sakai† and Hidetomo Nabeshima‡

† Nagoya University
‡ University of Yamanashi

1 Introduction

NaPS (Nagoya pseudo-Boolean solver) [7] is a pseudo-Boolean solver branched from MiniSAT+ version 1.0 [5]. NaPS accepts the DIMACS PB-instances for PB-competitions that have no non-linear constraints. The extended features of the NaPS input language are listed as follows:

- Negated Boolean variables: negations `~x` of variables `x` are allowed in constraints.
- Each constraint may include an implication: constraints are in the following form:

  `d` *literal implication-operator linear-constraint* `;`

  where, `=>`, `<=`, and `<=>` are allowed as an *implication-operator*.

NaPS optionally accepts the DIMACS format for CNF/WCNF and features a MaxSat mode. It has a (possibly partial) model-count facility.

2 Techniques

2.1 NaPS

NaPS is the one with version 1.03a6, which fixes some bugs of 1.03a2 submitted to the PB'24 competition. The solver transforms PB-instances to ordinary Boolean SAT-instances mainly via reduced ordered BDDs with two-clause coding [1]. It sometimes preprocesses coefficients by a multi-based decomposition [4]. It constructs a single BDD for each PB constraint using equation [7]. In optimization, it adopts an alternative strategy [7], which combines sequential and binary strategies, where the latter has a ratio of $1 : 2$.

It uses the following underlying SAT solvers:

- Kissat [2] version 4.0.2 for decision problems, and
- CaDiCaL [2] version 2.1.3 for others by default.

2.2 Scip-NaPS

Scip-NaPS version 1.00 is a combination solver that first attempts to solve using SCIP [3] and then NaPS. For optimization problems, NaPS receives a possible objective interval and a variable assignment whose values were fixed by SCIP, enabling NaPS to operate on a simplified instance.

References

1. I. Abío, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and V. Mayer-Eichberger. A new look at BDDs for Pseudo-Boolean constraints. *Journal of Artificial Intelligence Research* (JAIR), vol. 45, pp. 443–480, 2012.
2. Armin Biere, Mathias Fleury and Florian Pollitt:CaDiCaL vivinst, IsaSAT, Gimsatul, Kissat, and TabularaSAT Entering the SAT Competition 2023. In *Proc. of SAT Competition 2023 – Solver, Benchmark and Proof Checker Descriptions*, vol. B-2023-1 of Department of Computer Science Publicatoins B, pp. 14–15, University of Helsinki, 2023.
3. Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, Lixing Xu:The SCIP Optimization Suite 9.0. Available at *Optimization Online* and as *ZIB-Report* 24-02-29, February 2024.
4. M. Codish, Y. Fekete, C. Fuhs, and P. Schneider-Kamp. Optimal base encodings for Pseudo-Boolean constraints. In *TACAS*, volume 6605 of *LNCS*, pages 189–204, 2011.
5. N. Eén and N. Sörensson. Translating Pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation* (JSAT), vol. 2, no. 1–4, pp. 1–26, 2006.
6. H. Nabeshima, K. Iwanuma, and K. Inoue. On-the-fly lazy clause simplification based on binary resolvents. In *ICTAI*, IEEE, pp. 987–995, 2013.
7. M. Sakai and H. Nabeshima. Construction of an ROBDD for a PB-constraint in band form and related techniques for PB-solvers. *IEICE Transaction on Information and Systems*, vol. E98-D, no. 6, pp. 1121–1127, 2015.