Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

TIMO BERTHOLD* STEFAN HEINZ* MARC E. PFETSCH**

Nonlinear pseudo-Boolean optimization: relaxation or propagation?

* Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.
** Technische Universität Braunschweig, Institut für Mathematische Optimierung, Pockelsstraße 14, 38106 Braunschweig, Germany

ZIB-Report 09-11 (June 2009)

Nonlinear pseudo-Boolean optimization: relaxation or propagation?

Timo Berthold[1,⋆], Stefan Heinz[1,⋆], and Marc E. Pfetsch[2]

[1] Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
{berthold,heinz}@zib.de
[2] Technische Universität Braunschweig, Institut für Mathematische Optimierung, Pockelsstraße 14, 38106 Braunschweig, Germany
m.pfetsch@tu-bs.de

Abstract. Pseudo-Boolean problems lie on the border between satisfiability problems, constraint programming, and integer programming. In particular, nonlinear constraints in pseudo-Boolean optimization can be handled by methods arising in these different fields: One can either linearize them and work on a linear programming relaxation or one can treat them directly by propagation. In this paper, we investigate the individual strengths of these approaches and compare their computational performance. Furthermore, we integrate these techniques into a branch-and-cut-and-propagate framework, resulting in an efficient nonlinear pseudo-Boolean solver.

1 Introduction

Pseudo-Boolean (PB) optimization extends the satisfiability (SAT) problem by allowing integer coefficients in the constraints, multiplication of variables, and an objective function. As in SAT, variables take 0/1 (false/true) values.

There are several, fundamentally different, ways to attack the solution of PB-problems. One way is to apply a transformation to a SAT problem. This approach is used, for instance, in the solver MiniSat+ [10]. Another way is to handle PB-constraints directly in the solver, see, e.g., SAT4JPseudo [8], PBS [6], and pueblo [18]. Other solvers use a constraint programming approach, e.g., absconPseudo [13]. Pseudo-Boolean problems can also be formulated as an 0/1 integer program (IP), in which the nonlinear constraints are linearized. For instance glpPB uses this idea and applies the IP-solver glpk [12]. The solver bsolo [14] combines SAT-solving techniques with IP-methodologies to solve linear pseudo-Boolean problems, if the bounds from the linear programming (LP) relaxation are promising. The performance of the IP-solver Cplex for linear pseudo-Boolean problems was investigated in [5]. A variety of PB-solvers have been compared during the Pseudo-Boolean Evaluations [15, 16].

⋆ Supported by the DFG Research Center Matheon *Mathematics for key technologies* in Berlin.

In this paper, we approach nonlinear PB-problems via *constraint integer programming* (CIP). CIP is a combination of IP, SAT, and constraint programming (CP) methodologies. CIP was introduced by Achterberg [1, 2] and implemented in the framework SCIP. The basic idea is to apply a branch-and-cut-and-propagate method. Hence, one performs a branch-and-bound algorithm to decompose the problem into subproblems (as in SAT, CP, and IP-solvers). One solves a linear relaxation, which is strengthened by additional inequalities/cutting planes if possible (as in IP-solvers). One uses propagation techniques (similar to CP-solvers) in the nodes of the search tree. Moreover, one applies conflict analysis and restarts (similar to SAT-solvers). Detailed descriptions of the CIP-paradigm and the algorithmic design of SCIP can be found in [1–3].

The main goal when applying CIP to PB problems is to use the IP-machinery with LP-relaxations, cutting planes, elaborated branching rules, etc., and directly propagating the (nonlinear) multiplications. As far as we know, all of the PB-solvers discussed above *either* handle nonlinearities directly *or* add a complete linearization to the problem formulation. We compare both ideas and introduce a hybrid approach which only partially linearizes the nonlinear part of the problem. It turns out that the combination of both, propagation and (partial) linear relaxation, performs better than applying only one of these.

2 Problem Definition

For a Boolean variable $x \in \{0,1\}$, a *literal* ℓ is either the original variable x or its negation $\overline{x} := 1 - x$. A *(nonlinear) pseudo-Boolean problem* with Boolean variables $x_1, \ldots, x_n$ ($n \in \mathbb{N}$) is an optimization problem of the following form:

$$\begin{aligned} \min \quad & \sum_{j=1}^{t_0} c_j \cdot \prod_{\ell \in I_{0j}} \ell && (1)\\ & \sum_{j=1}^{t_i} a_{ij} \cdot \prod_{\ell \in I_{ij}} \ell \ \geq b_i && \text{for } i = 1, \ldots, m \\ & \boldsymbol{x} \in \{0,1\}^n. \end{aligned}$$

Here, $m \in \mathbb{N}$ defines the number of constraints, I_{ij} is a subset of literals for $i = 0, \ldots, m$ and $j = 1, \ldots, t_i$, where $t_i \in \mathbb{N}$ is the number of summands in constraint i. All coefficients a_{ij}, b_i, c_j are required to be integral. Let

$$\mathcal{I} := \{(i,j) \,:\, i \in \{0, \ldots, m\},\ j \in \{1, \ldots, t_i\}\}.$$

The above formulation is quite general: one can easily incorporate maximization, "$\leq$" constraints, equations, and pure satisfiability problems. If $|I_{ij}| \leq 1$ for all $(i,j) \in \mathcal{I}$, the objective function and the constraints are linear expressions in the variables. We call such instances *linear pseudo-Boolean problems*. If the objective function equals zero (or any other constant), we have *satisfiability* problems, otherwise *optimization* problems.

SAT problems are special cases of PB problems with $b_i = 1$, for $i = 1, \ldots, m$, $c_j = 0$, for $j = 1, \ldots, t_0$, and $a_{ij} \in \{0,1\}$ for $i = 1, \ldots, m$, $j = 1, \ldots, t_i$.

3 Handling of nonlinearities

Linear Relaxations. To deal with the nonlinear constraints, we transform Problem (1) as follows. For each $(i,j) \in \mathcal{I}$ with $|I_{ij}| > 1$, we introduce a new Boolean variable $z_{ij} = \prod_{\ell \in I_{ij}} \ell$. The product can also be seen as an AND-expression $z_{ij} = \bigwedge_{\ell \in I_{ij}} \ell$, for which we apply the following linearization:

$$z_{ij} - \ell \leq 0 \qquad \text{for } \ell \in I_{ij} \tag{2}$$

$$\sum_{\ell \in I_{ij}} \ell - z_{ij} \leq |I_{ij}| - 1. \tag{3}$$

After replacing the AND-expressions by z_{ij}, the resulting constraint is linear in z_{ij}. This linearization has the following nice feature.

Lemma. *The polyhedron defined by* (2), (3), $z_{ij} \geq 0$, *and* $\ell \leq 1$ *for all* $\ell \in I_{ij}$ *is integral, i.e., has only integral vertices.*

Note that $\ell \geq 0$, $\ell \in I_{ij}$, is implied by $z_{ij} \geq 0$ and inequalities (2). Similarly, $z_{ij} \leq 1$ is implied by (2) and $\ell \leq 1$ for $\ell \in I_{ij}$. Hence, P_{AND} is bounded, i.e., a polytope.

Proof. For ease of exhibition, denote $I_{ij} = \{x_1, \ldots, x_n\}$ and $y = z_{ij}$. Consider an arbitrary vertex solution $(x^\star, y^\star)$ to (2)–(3), $-y \leq 0$, $x_i \leq 1$, $i = 1, \ldots, n$.

We first assume that $y^\star \in (0,1)$. Since $(x^\star, y^\star)$ is a vertex solution, there are $n+1$ inequalities that are satisfied with equality. Since $-y \leq 0$ is not satisfied with equality, and at most one of $y - x_i \leq 0$ and $x_i \leq 1$ can be satisfied with equality, we have $\sum_{i=1}^n x_i^\star - y^\star = n - 1$. Let $S = \{i : x_i^\star = 1\}$ and $k = |S|$. It follows that

$$n - 1 = \sum_{i=1}^{n} x_i^\star - y^\star = k + \sum_{i \notin S} y^\star - y^\star = k + (n - k - 1)y^\star.$$

This is a contradiction since $y^\star < 1$.

Hence, $y^\star \in \{0,1\}$. If $y^\star = 1$, then $x_i^\star = 1$ for all $i = 1, \ldots, n$ by (3), and $(x^\star, y^\star)$ is integral.

Now assume that $y^\star = 0$. After removing y from the inequalities and replacing (3) by an equation, the remaining ones define a so-called hypersimplex $\Delta_{n-1}(n-1)$, which is integral, see Ziegler [19]. If we again relax the equation to an inequality, the polytope remains integral, since the hypersimplex is the intersection of the unit cube $[0,1]^n$ with one hyperplane. □

Note that the above linearization is different from

$$\sum_{\ell \in I_{ij}} \ell - |I_{ij}|\, z_{ij} \geq 0, \qquad \sum_{\ell \in I_{ij}} \ell - z_{ij} \leq |I_{ij}| - 1, \tag{4}$$

which is used in the Pseudo-Boolean Evaluation [16]. Both linearizations have the property that 0/1-solutions of the corresponding systems are solutions of

their AND-expressions and conversely. However, while the above Lemma holds for the first linearization, the corresponding polyhedron of (4) is not integral, since $z_{ij} = \frac{1}{n}$, $\hat{\ell} = 1$ for some $\hat{\ell} \in I_{ij}$, $\ell = 0$ for all $\ell \in I_{ij} \setminus \hat{\ell}$ is a fractional vertex.

Linearization (4) has the advantage that it contains only two constraints, compared to $|I_{ij}|+1$ in (2) and (3). The larger size of the first linearization can be handled by a so-called *separation mechanism*, i.e., the necessary inequalities are generated on the fly, if they are violated. More precisely, one solves an LP which initially only consists of the objective function and the linear constraints in z_{ij} and neglect the AND-expressions (and integrality constraints). Inequalities (2) and (3) are added, only if they violate the optimal solution of this LP-relaxation. Then the resulting LP is solved and the process is iterated.

The advantages and disadvantages of the above linearizations have been widely discussed in the literature, see, for instance, Glover and Woolsey [11], Balas and Mazzola [7], and Adams and Sherali [4].

Constraint Programming. The CP approach applies a domain propagation algorithm at each subproblem of the branch-and-bound process in order to fix further variables. The propagation rules are as follows. If one of the operand variables $\ell \in I_{ij}$ is fixed to zero the resultant variable z_{ij} has to be zero, too. On the other hand, if all operand variables are set to one, the resultant variables must also be fixed to one, and vice versa. Finally, if the resultant variable z_{ij} is zero and all but one of the operand variables are one, the remaining operand variable can be fixed to zero.

The main advantage of this approach is that all these propagation rules can be applied very efficiently and therefore the computation time per node is very small. The disadvantage is that one looses the global view of the LP-relaxation and its strong capability of pruning suboptimal parts of the tree.

Constraint Integer Programming. The hope of an integrated approach is that on the one hand the fixings derived by domain propagation reduce the size of the LP and therefore potentially the computational overhead. On the other hand, these fixings may even yield a stronger LP-bound which vice versa can lead to further variable fixings which can be propagated and so forth.

We study three different variants of integration. First, we apply the suggested separation mechanism simultaneously with the propagation algorithm. Second, we add the complete linearization and apply propagation. Third, we change the strategy dynamically depending on the problem's degree of nonlinearity.

4 Computational results

In this section, we analyze how each of the approaches performs for the nonlinear test sets of the Pseudo-Boolean Evaluation 2007 [16].

All computations reported in the following were obtained using version 1.1.0.6 of SCIP [17] on Intel Xeon Core 2.66 GHz computers (in 64 bit mode) with 4 MB cache, running Linux, and 6 GB of main memory. We integrated CLP release

Table 1. Results for the 405 OPT-SMALLINT-NLC instances

Setting	opt	sat	unkn	Nodes total(k)	Nodes geom[1]	Time in [s] total(k)	Time in [s] geom[1]
only propagation	269	321	84	152027	13477	235.3	62.6
only separation	225	276	129	67313	5583	359.3	202.5
only relaxation	236	326	79	90639	4184	340.5	194.0
separation/propagation	288	341	64	12219	1267	225.2	61.5
relaxation/propagation	284	372	33	5105	846	226.3	59.6
dynamic	291	342	63	11009	1219	223.5	64.3
MINISAT+	279	397	8	–	–	234.0	46.2

version 1.9.0 as underlying LP-solver [9]. Thus, we only used noncommercial software, which is available in source code.

As in the PB evaluation, we set a time limit of 1800 seconds. We compared the performance of SCIP for six different settings, which only differ in the way they handle the AND-expressions. The setting "only relaxation" only applies the complete linearization of the AND-expressions before starting the search, "only separation" only uses separation, i.e., adding inequalities (2) and (3) when they are violated, and "only propagation" only performs propagation without using the linearization. The settings "relaxation/propagation" and "separation/propagation" linearize the AND-expressions in advance and on the fly, respectively. Additionally, they apply the described propagation algorithms, thereby combining CP and IP techniques. The setting "dynamic" incorporates the latter two: If the linearization of the AND-expressions consists of less than 10 000 linear constraints, "relaxation/propagation" will be used, otherwise, "separation/propagation" will be used. The motivation was to work on the complete linear description only if it is small and not likely to produce a huge computational overhead. We use the inequalities (2) and (3) as linearization. All remaining parameters of SCIP were set to their default values, hence we use primal heuristics such as the feasibility pump, general purpose cutting planes such as Gomory cuts, preprocessing strategies, and we use conflict analysis and restarts.

According to the PB evaluation, the instances are split into the following two groups, both with "small" integers, i.e., all coefficients are representable as 32 bit integers: OPT-SMALLINT-NLC (nonlinear PB optimization), SATUNSAT-SMALLINT-NLC (nonlinear PB satisfiability). For details we refer to [16].

We compare for how many instances optimality ("opt") or at least satisfiability ("sat") could be proven, the number of instances for which no result was obtained ("unkn"), total time and number of branch-and-bound nodes over all instances in the test set and the shifted geometric means[1] ("geom") over these two performance measures. For the satisfiability test set, we compare the number of instances for which an answer could be found ("solved"), which we subdivide into feasible ("sat") and infeasible ("unsat") instances, the time and the number of branch-and-bound nodes in total and in shifted geometric mean as before.

[1] The shifted geometric mean of values $t_1, \ldots, t_n$ is defined as $\left(\prod(t_i + s)\right)^{1/n} - s$ with shift s. We use a shift $s = 10$ for time and $s = 100$ for nodes in order to decrease the strong influence of the very easy instances in the mean values.

Table 2. Results for the 100 SatUnsat-Smallint-Nlc instances

Setting	solved	sat	unsat	unkn	Nodes total(k)	Nodes geom[1]	Time in [s] total(k)	Time in [s] geom[1]
only propagation	60	50	10	40	41085	6883	72.5	86.5
only separation	70	50	20	30	671	281	55.0	57.3
only relaxation	71	51	20	29	446	161	58.4	69.6
separation/propagation	72	52	20	28	883	284	51.7	53.3
relaxation/propagation	73	53	20	27	489	154	55.7	66.4
dynamic	72	52	20	28	835	279	51.8	55.6
MiniSat+	65	50	15	35	–	–	63.1	56.4

The results of Tables 1 and 2 show that the combined approaches are superior to the ones which use only one algorithm. For the optimization instances, each of them solves more instances to optimality and finds more feasible solutions than each of the "only" settings. The same holds for the number of solved instances in the satisfiability test set. Furthermore, the combined approaches usually need less branch-and-bound nodes and less overall running time.

As one would expect, the setting "relaxation/propagation", the method with the highest computational effort, needs the fewest branch-and-bound nodes for both test sets, but spends the most time per node. In contrast, "only propagation" requires little time per node, but needs the most branch-and-bound nodes.

The "dynamic" setting enables to solve most of the optimization problems and only one instance less than the best setting for the satisfiability instances. The setting "relaxation/propagation" is the best performing for the satisfiability instances and, moreover, the best in finding feasible solutions for the optimization problems. This can be explained by the fact that the primal heuristics work best, if there is a full linear description present.

We conclude that combining LP-relaxation and domain propagation techniques help to solve nonlinear pseudo-Boolean problems. Furthermore, for proving optimality, it is recommendable to only use a partial linearization for instances with a large nonlinear part. We also performed all experiments using the linearization (4). The results are similar, but slightly worse.

For comparison, we ran MiniSat+, the best solver for nonlinear PB problems in the PB evaluation 2007, on the same computational environment. For the results in Tables 1 and 2 we used linearization (2) and (3), while with linearization (4) MiniSat+ solved two instances less of the optimization test set and performed slightly worse in both cases.

References

1. T. Achterberg. *Constraint Integer Programming.* PhD thesis, TU Berlin, 2007.
2. T. Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.
3. T. Achterberg, T. Berthold, T. Koch, and K. Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In L. Perron and M. A. Trick, editors, *Proc. CPAIOR 2008*, volume 5015 of *LNCS*, pages 6–20. Springer, 2008.
4. W. P. Adams and H. D. Sherali. Linearization strategies for a class of zero-one mixed integer programming problems. *Oper. Res.*, 38(2):217–226, 1990.

5. F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah. Generic ILP versus specialized 0-1 ILP: an update. In L. T. Pileggi and A. Kuehlmann, editors, *Proc. of the 2002 IEEE/ACM International Conference on Computer-aided Design*, pages 450–457. ACM, 2002.
6. F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah. PBS: A backtrack-search pseudo-boolean solver and optimizer. In *Proc. Fifth International Symposium on Theory and Applications of Satisfiability Testing (SAT 2002)*, pages 346–353, 2002.
7. E. Balas and J. B. Mazzola. Nonlinear 0-1 programming: I. Linearization techniques. *Math. Prog.*, 30(1):1–21, 1984.
8. D. L. Berre. Sat4j. `http://www.sat4j.org/`.
9. Clp. COIN-OR LP-solver. `http://www.coin-or.org/projects/Clp.xml`.
10. N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *J. Satisf. Boolean Model. Comput.*, 2:1–26, 2006.
11. F. Glover and E. Woolsey. Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Oper. Res.*, 22(1):180–182, 1974.
12. GLPK. GNU linear programming kit. `http://www.gnu.org/software/glpk/`.
13. F. Hemery and C. Lecoutre. AbsconPseudo 2006. `http://www.cril.univ-artois.fr/PB06/papers/abscon2006V2.pdf`, 2006.
14. V. M. Manquinho and J. Marques-Silva. On using cutting planes in pseudo-Boolean optimization. *J. Satisf. Boolean Model. Comput.*, 2:209–219, 2006.
15. V. M. Manquinho and O. Roussel. The first evaluation of pseudo-Boolean solvers (PB'05). *J. Satisf. Boolean Model. Comput.*, 2:103–143, 2006.
16. V. M. Manquinho and O. Roussel. Pseudo-Boolean evaluation 2007. `http://www.cril.univ-artois.fr/PB07/`, 2007.
17. SCIP. Solving Constraint Integer Programs. `http://scip.zib.de/`.
18. H. M. Sheini and K. A. Sakallah. Pueblo: A hybrid pseudo-boolean SAT solver. *J. Satisf. Boolean Model. Comput.*, 2:165–189, 2006.
19. G. M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, New York, 1995. Revised edition 1998.