PAPER *Special Section on Formal Approach*

Construction of an ROBDD for a PB-Constraint in Band Form and Related Techniques for PB-Solvers

Masahiko SAKAI[†a)], ***Fellow and*** **Hidetomo NABESHIMA**[††], ***Member***

SUMMARY Pseudo-Boolean (PB) problems are Integer Linear Problem restricted to 0-1 variables. This paper discusses on acceleration techniques of PB-solvers that employ SAT-solving of combined CNFs each of which is produced from each PB-constraint via a binary decision diagram (BDD). Specifically, we show (i) an efficient construction of a reduced ordered BDD (ROBDD) from a constraint in band form $l \leq \langle\text{Linear term}\rangle \leq h$, (ii) a CNF coding that produces two clauses for some nodes in an ROBDD obtained by (i), and (iii) an incremental SAT-solving of the binary/alternative search for minimizing values of a given goal function. We implemented the proposed constructions and report on experimental results.

key words: *reduced ordered BDD, Pesudo-Boolean constraint, optimization problem*

1. Introduction

A *Pseudo-Boolean (PB)* problem is the problem which answers the satisfiability of a given *instance*, which is a conjunction of linear inequality constraints over Boolean variables. Typical approaches to solve PB-constraints employ *Integer Linear Programming* (restricted to 0-1 variables), DPLL procedures (regarding PB-constraints as generalized clauses [1]), as well as transformations of PB constraints to a CNF (via adders, sorting networks, and BDDs [2], [3]). Abío et al. have shown that a conversion to a reduced ordered BDD (ROBDD) from a given PB-constraint in forms of $\langle\text{Linear term}\rangle \leq k$, and that two-clause coding of a BDD by using monotonic property [4].

This paper extends the ROBDD result [4] for a PB-constraint in band form, i.e., $l \leq \langle\text{Linear term}\rangle \leq h$. An expected benefit to construct a single ROBDD is the reduction of total nodes. The band form is practical, since an equality constraint $\langle\text{Linear term}\rangle = k$ is equivalent to $k \leq \langle\text{Linear term}\rangle \leq k$. We show experimental results of a MiniSat+ based solver, in which we incorporated the proposed ROBDD construction, including the binary search and the alternative search for the optimization problem, which minimize the value of a given function.

Manuscript received August 27, 2014.
Manuscript revised December 19, 2014.
Manuscript publicized February 13, 2015.

[†]The author is with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8603 Japan.
[††]The author is with the Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi, Kofu-shi, 400–0016 Japan.
a) E-mail: sakai@is.nagoya-u.ac.jp

DOI: 10.1587/transinf.2014FOP0007

2. Preliminaries

An *interval* is a set of consecutive integers. An interval $[\beta, \gamma]$ ($\beta, \gamma \in \mathbb{Z} \cup \{-\infty, \infty\}$) represents the set $\{i \in \mathbb{Z} \mid \beta \leq i \leq \gamma\}$, where any integer i satisfies $-\infty \leq i$ and $i \leq \infty$. We use usual notations like $(-\infty, i]$. The summation $I + j$ of an interval I and an integer j is defined as $\{i + j \mid i \in I\}$.

A *valuation* σ is a function that assigns 0 or 1 to variables. An application of σ to a *propositional formula* f, and so on, is naturally extended. A valuation σ *satisfies* a formula f if $\sigma(f) = 1$. A formula f is *satisfiable* if there exists a valuation that satisfies f; otherwise it is *unsatisfiable*. Two propositional formulae f and g, which may have different variables, are *equivalent* if $\sigma(f) = \sigma(g)$ for any valuation σ. They are *equisatisfiable* if f is satisfiable whenever g is, and vice versa.

For a set V of variables, we say σ' is a V-*extension* of σ if $\sigma'(x) = \sigma(x)$ for any $x \notin V$. We write $\sigma_{[x \mapsto b]}$ for a $\{x\}$-extension of σ such that $\sigma_{[x \mapsto b]}(x) = b$. We say a propositional formula f is *monotonically increasing* (resp. *monotonically decreasing*) *with respect to a variable* x if $\sigma_{[x \mapsto 0]}(f)$ implies $\sigma_{[x \mapsto 1]}(f)$ (resp. $\sigma_{[x \mapsto 1]}(f)$ implies $\sigma_{[x \mapsto 0]}(f)$) for any valuation σ. f is *monotonically increasing* if it is so with respect to all variables. A propositional formula with variables $x_1, \dots, x_n$ is regarded as a *Boolean function* $f(x_1, \dots, x_n)$.

A *Binary Decision Diagram* (BDD) is a rooted, directed and acyclic graph, which consists of *decision nodes* and *terminal nodes* N_0 and N_1. Each node N represents a Boolean function, denoted by $\text{Fun}(N)$. For terminal nodes, $\text{Fun}(N_0) = 0$ and $\text{Fun}(N_1) = 1$. A decision node has two children, and is labelled with a *selector variable* x, which is simply called a *selector*. We call the child connected with a solid (resp. dotted) edge *true-child* (resp. *false-child*). A decision node N represents the Boolean function $(x \wedge \text{Fun}(N_t)) \vee (\overline{x} \wedge \text{Fun}(N_f))$ determined by its true-child N_t and false-child N_f. A *variable order* is a total order on variables. This paper assumes that BDDs are ordered (OBDDs), i.e., there exists a variable order $<$ such that for any path the sequence $x_1, \dots, x_n$ of selectors along the path satisfies $x_1 < \cdots < x_n$. An ordered BDD is *reduced* (ROBDD) [5] if Boolean functions represented by the nodes are all different. ROBDDs are a canonical representation for Boolean functions under a given variable order.

A *Pseudo-Boolean constraint* (PB-constraint) is a linear inequality with integer coefficients, where variables have

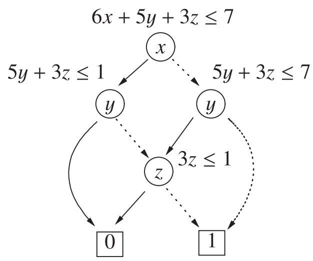


Fig. 1 ROBDD of $6x + 5y + 3z \le 7$.

Boolean domain $\{0, 1\}$. A PB-constraint has a *standard form* $a_n\ell_n + \cdots + a_1\ell_1 \le k$, where the a_i's and k are integers such that $a_i > 0$ and each ℓ_i is a *positive literal* x_i or a *negative literal* $\overline{x_i}$. Note that a negative literal $\overline{x_i}$ is equal to $1 - x_i$. A PB-constraint $a_n\ell_n + \cdots + a_1\ell_1 \le k$ can be seen as a Boolean function $f(x_1, \ldots, x_n)$, and has a BDD representation. Note that a standardized PB-constraint is unsatisfiable if $k < 0$, and is valid if $a_1 + \cdots + a_n \le k$.

Example 1: An ROBDD for $6x+5y+3z \le 7$ with a variable order $x < y < z$ is shown in Fig. 1.

3. ROBDD Construction for Band Form

This section explores a construction of a single ROBDD from a PB-constraint of the form $k_l \le a_n\ell_n + \cdots + a_1\ell_1 \le k_h$, which we call a *band form*, which is an extension of an efficient construction of ROBDDs for PB-constraints in standard form developed in the Ref. [4]. The following lemma suggests a construction of BDD nodes for a PB-constraint.

Lemma 2: Let C be a PB-constraint $k_l \le a\ell+e \le k_h$, where e stands for a linear expression. Let N_t (resp. N_f) be a node of a BDD such that $\mathrm{Fun}(N_t)$ (resp. $\mathrm{Fun}(N_f)$) is equivalent to $k_l - a \le e \le k_h - a$ (resp. $k_l \le e \le k_h$). Then $\mathrm{Fun}(N)$ is equivalent to C, where N has a selector x,

1. N_t as true-child and N_f as false-child if $\ell = x$, and
2. N_f as true-child and N_t as false-child if $\ell = \overline{x}$.

Proof If $\ell = x$, from the definition of BDDs, $\mathrm{Fun}(N)$ is $(x \wedge (k_l - a \le e \le k_h - a)) \vee (\overline{x} \wedge (k_l \le e \le k_h))$, which is equivalent to $k_l \le ax + e \le k_h$. The other case is similar. □

The notion of intervals [4] for an efficient ROBDD construction is extended for PB-constraints in band form.

Definition 3: Let $k_l \le e \le k_h$ be a PB-constraint. We say that a pair $\langle L, H\rangle$ of intervals is *consistent to* e if L and H are maximal intervals (with respect to set inclusion) that satisfy the following condition (*):

> PB-constraints $l \le e \le h$ such that $l \in L$ and $h \in H$ are all equivalent (seen as Boolean functions).

We use $L \le e \le H$ to denote the expected Boolean function. Moreover, for $k_l \in L$ and $k_h \in H$, we say that a pair $\langle L, H\rangle$ of intervals is *consistent to* $k_l \le e \le k_h$.

For example, $\langle [1, 1], [1, 2]\rangle$ is the unique pair of intervals that is consistent to $1 \le x + 3y \le 2$.

Proposition 4: Let $k_l \le e \le k_h$ be a satisfiable PB-constraint. Then its consistent pair $\langle L, H\rangle$ of intervals is unique.

Proof Let $\langle L, H\rangle$ be a pair of intervals that is consistent to $k_l \le e \le k_h$. From the definition, $k_l \in L$ and $k_h \in H$ follow. Let

$$I_l = \{\sigma(e) \mid \sigma(e) < k_l\} \cup \{-\infty\},$$
$$I = \{\sigma(e) \mid k_l \le \sigma(e) \le k_h\},$$
$$I_h = \{\sigma(e) \mid k_h < \sigma(e)\} \cup \{\infty\}.$$

Since $k_l \le e \le k_h$ is satisfiable, I is not an empty set. From the definition of consistency, the interval $\langle L, H\rangle$ is uniquely determined as $\langle [\max(I_l) + 1, \min(I)], [\max(I), \min(I_h) - 1]\rangle$. □

Remark that an unsatisfiable PB-constraint may have two consistent pairs of intervals. For example, $5 \le x + y \le -5$ has two consistent pairs $\langle (-\infty, \infty), (-\infty, -1]\rangle$ and $\langle [3, \infty), (-\infty, \infty)\rangle$. Also consistent pairs of $5 \le 0 \le -5$ are $\langle (-\infty, \infty), (-\infty, -1]\rangle$ and $\langle [1, \infty), (-\infty, \infty)\rangle$.

In the rest of this section, we assume that a given PB-constraint is $C :\ k_l \le a_n\ell_n + \cdots + a_1\ell_1 \le k_h$, and construct an ROBDD, which is equivalent to C, under a fixed variable order $x_n < x_{n-1} < \cdots < x_1$. We use e_i to denote the subexpression $a_i\ell_i + \cdots + a_1\ell_1$.

An important observation in the ROBDD construction is that the pair of intervals for a node is directly calculated from those of its children as shown in the following lemma.

Lemma 5: Let pairs $\langle L_t, H_t\rangle$ and $\langle L_f, H_f\rangle$ intervals be consistent to e_i. If both of intervals $L = (L_t + a_{i+1}) \cap L_f$ and $H = (H_t + a_{i+1}) \cap H_f$ are non-empty, $\langle L, H\rangle$ is consistent to e_{i+1}.

Proof Assume that (*) in Definition 3 does not hold for e_{i+1} and $\langle L, H\rangle$. Then, there exists $l, l' \in L$, $h, h' \in H$, and a valuation σ that satisfies $l \le e_{i+1} \le h$ but does not satisfy $l' \le e_{i+1} \le h'$. In the case that $\sigma(\ell_{i+1}) = 0$, we have $\sigma(e_{i+1}) = \sigma(a_{i+1}\ell_{i+1} + e_i) = \sigma(e_i)$. Thus, σ satisfies $l \le e_i \le h$, but does not satisfy $l' \le e_i \le h'$. Since $l, l' \in L_f$ and $h, h' \in H_f$ from the construction of L and H, this is a contradiction to the consistency of $\langle L_f, H_f\rangle$ to e_i. In the case that $\sigma(\ell_{i+1}) = 1$, we have $\sigma(e_{i+1}) = a_{i+1} + \sigma(e_i)$. Thus σ satisfies $l - a_{i+1} \le e_i \le h - a_{i+1}$, but does not satisfy $l' - a_{i+1} \le e_i \le h' - a_{i+1}$. Since $l - a_{i+1}, l' - a_{i+1} \in L_t$ and $h - a_{i+1}, h' - a_{i+1} \in H_t$, this is a contradiction to the consistency of $\langle L_t, H_t\rangle$ to e_i. Therefore $\langle L, H\rangle$ is consistent to e_{i+1}.

Next we show the maximality of $\langle L, H\rangle$. Since L and H are non-empty, it is enough to show that any extension causes inconsistency. Let $L = [\beta, \gamma]$ and $L' = [\beta, \gamma + 1]$ for $\gamma \ne \infty$. We show only that $\langle L', H\rangle$ is not consistent to e_{i+1} since the other cases are similar. Let $L_t = [\beta_t, \gamma_t]$ and $L_f = [\beta_f, \gamma_f]$. Since $L = (L_t + a_{i+1}) \cap L_f$, we have two cases

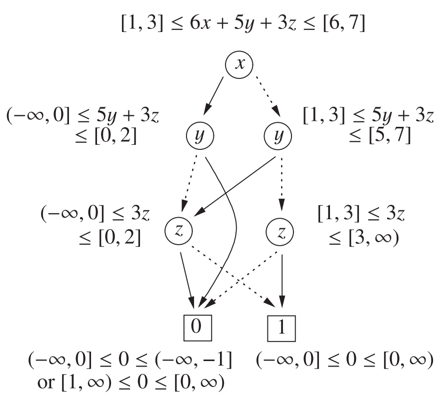


Fig. 2 ROBDD of $3 \le 6x + 5y + 3z \le 7$.

Input: A PB-constraint $k_l \le a_n\ell_n + \cdots + a_1\ell_1 \le k_h$. We assume the variable order $x_n < \cdots < x_1$.
Initialization:

$$D_0 := \{ \langle 0, (-\infty, \infty), (-\infty, -1], N_0 \rangle, \langle 0, [1, \infty), (-\infty, \infty), N_0 \rangle, \langle 0, (-\infty, 0], [0, \infty), N_1 \rangle \},$$

where N_0 and N_1 are the terminal nodes.
Output: The BDD N obtained by

$$\langle L, H, N, D \rangle := \text{CreateBDD}(n, k_l, k_h, D_0).$$

Function CreateBDD(i, l, h, D) =
Step a): Seek $\langle i, L, H, N \rangle$ in D such that $l \in L$ and $h \in H$. If succeeded, return $\langle L, H, N, D \rangle$.
Step b): Let
$\langle L_t, H_t, N_t, D_1 \rangle :=$ CreateBDD$(i - 1, l - a_i, h - a_i, D)$ and
$\langle L_f, H_f, N_f, D_2 \rangle :=$ CreateBDD$(i - 1, l, h, D_1)$.
If $N_t = N_f$ then set $N := N_t$, otherwise:
Create a node N with x_i as a label with N_t as true-child, and N_f as false-child. Swap children if l_i is negative literal $\overline{x_i}$.
Return $\langle L, H, N, D_3 \rangle$, where $L = (L_t + a_i) \cap L_f$, $H = (H_t + a_i) \cap H_f$, and $D_3 = D_2 \cup \{\langle i, L, H, N \rangle\}$.

Fig. 3 Algorithm for the first stage.

that $\gamma = \gamma_f$ and $\gamma = \gamma_t + a_{i+1}$.

Consider the former case. Since $\langle L_f, H_f \rangle$ is consistent to e_i, there exist $l \in L_f$, $h, h' \in H_f$, and a valuation σ that satisfies either $l \le e_i \le h$ or $\gamma_f + 1 \le e_i \le h'$, exclusively. Let σ' satisfies $\sigma'(\ell_{i+1}) = 0$ and $\sigma'(x) = \sigma(x)$ for any x $(\neq x_{i+1})$. Then σ satisfies $l \le e_i \le h$ if and only if σ' satisfies $l \le e_{i+1} \le h$, and also σ satisfies $\gamma_f + 1 \le e_i \le h'$ if and only if σ' satisfies $\gamma_f + 1 \le e_{i+1} \le h'$. Since $l, \gamma_f + 1 \in L'$ and $h, h' \in H$, the pair $\langle L', H \rangle$ is not consistent to e_{i+1}. The latter case is similar. □

Thanks to Lemma 5, consistent intervals $\langle L, H \rangle$ for a node of a BDD for a PB-constraint can be immediately calculated from its children.

Example 6: An ROBDD for $3 \le 6x + 5y + 3z \le 7$ with an order $x < y < z$ is shown in Fig. 2.

The algorithm for the ROBDD construction shown in Fig. 3 works in depth first way with memorizing the interval information, which is a natural extension of the algorithm proposed in Sect. 5 of the Ref. [4] for constraints in standard form. The essential difference from [4] is on memorizing a pair of intervals instead of an interval for each BDD node.

We use a set D each of whose elements is a tuple of a natural number i, intervals L and H, and a node N of BDD in constructing, where i is used to identify the selector variables x_i.

Definition 7: We say D is *consistent*, if $D_0 \subseteq D$ and for every $\langle i, L, H, N \rangle \in D$

(i) $\langle L, H \rangle$ is consistent to e_i, and
(ii) Fun(N) is equivalent to $L \le e_i \le H$.

Lemma 8: D_0 is consistent.

Proof Consider the element $\langle 0, (-\infty, \infty), (-\infty, -1], N_0 \rangle$. Since $e_0 = 0$, $l \le e_0 \le h$ is not satisfied for any $l \in (-\infty, \infty)$ and $h \in (-\infty, -1]$, which is equivalent to Fun$(N_0) = 0$. The maximality is trivial. The other elements are similarly shown. □

The function CreateBDD eventually goes into Step a) if $i = 0$, because the intervals in D_0 cover all pairs of integers. Thus the termination of the algorithm is easily derived.

Lemma 9: Let $\langle L, H, N, D' \rangle :=$ CreateBDD(i, l, h, D) for a consistent D and $0 \le i \le n$,

(1) $l \in L$, $h \in H$, $\langle i, L, H, N \rangle \in D'$, and
(2) D' is consistent,

Proof By induction on i. In the case that $i = 0$, Step a) succeeds to find $\langle 0, L, H, N \rangle$ in $D_0 \subseteq D$. Thus (1) and (2) follow directly from the consistency of D.

Consider the case that $i > 0$. If $\langle i, L, H, N \rangle$ such that $l \in L$ and $h \in H$ is found in D at the Step a), then $D = D'$. Thus (1) is trivial and (2) follows directly from the consistency of D. Otherwise $\langle L_t, H_t, N_t, D_1 \rangle :=$ CreateBDD$(i-1, l-a_i, h-a_i, D)$ and $\langle L_f, H_f, N_f, D_2 \rangle :=$ CreateBDD$(i-1, l, h, D_1)$ are invoked in the algorithm. Here D_1 and D_2 are consistent by induction hypothesis (2), and $l - a_i \in L_t$, $h - a_i \in H_t$, $\langle i - 1, L_t, H_t, N_t \rangle \in D_1$, $l \in L_f$, $h \in H_f$, and $\langle i-1, L_f, H_f, N_f \rangle \in D_2$ by induction hypothesis (1). Since $L = (L_t + a_i) \cap L_f$, we obtain $l \in L$ and hence of L is not empty. Similarly $h \in H$ and H is non-empty.

We show (i) and (ii) in Definition 7 for $\langle i, L, H, N \rangle$. By Lemma 5, (i) follows. Since $l - a_i \le e_{i-1} \le h - a_i$ (resp. $l \le e_{i-1} \le h$) is equivalent to Fun(N_t) (resp. Fun(N_f)) from the consistency of D_1 (resp. D_2). From the construction of N and Lemma 2, we obtain Fun(N) is equivalent to $l \le a_i\ell_i + e_{i-1} \le h$. Combining this with $l \in L$ and $h \in H$, we obtain (ii). Moreover, if $N_t = N_f$ then Fun(N_t), Fun(N_f), and $l \le a_i\ell_i + e_{i,n} \le h$ are equivalent. □

Lemma 10: The algorithm computes an ROBDD, whose root represents the given PB-constraint $k_l \le e_n \le k_h$.

Proof Let $\langle L, H, N, D' \rangle :=$ CreateBDD(n, k_l, k_h, D_0). Since

D_0 is consistent from Lemma 8, we obtain $k_l \in L$ and $k_h \in H$ where $\langle n, L, H, N\rangle$ is in the consistent D'. Therefore, Fun(N) is equivalent to $k_l \leq e_n \leq k_h$.

All nodes in a resulted ordered BDD appears as different elements in the consistent database D' from the following reasons. Nodes N and N' with the same selector variable x_i represent different functions from Proposition 4. Moreover, if $N_t = N_f$ the algorithm do not create a new node. Thus it is reduced. □

This algorithm runs in $O(nm\log(m))$ where m is the size of the ROBDD, which is shown in the same way as [4]; the cost of search and insertion on D is $O(\log m)$, and the number of calls of CreateBDD is bounded by $O(nm)$. Note that n is necessary because Step b) does not always create a BDD node.

4. Monotonic CNF-Coding of BDD

In this section, we present a simple way to detect nodes that represent monotonic functions in an ROBDD produced from PB-constraint in band form.

Definition 11: Given a BDD rooted by R, we define a *CNF-coding*, denoted by Cnf(R), as the conjunction of the following formulae, provided a fresh variable p_N for each node N.

(1) "1" for a terminal node N_1,
(2) "$\overline{p_{N_0}}$" for a terminal node N_0, and
(3) "$p_N \implies$ `if` x `then` p_{N_t} `else` p_{N_f}" for a decision node N labelled with x, where N_t (resp. N_f) is the true-child N_t (resp. false-child N_f) of N.

Note that `if-then-else` has the meaning as expected. We refer the set of newly introduced variables for the coding of a BDD rooted by R by PVar(R).

For coding (3) as clauses, three-clause coding is used in MiniSAT+ [2], which produces

(Three) $x \vee p_{N_f} \vee \overline{p_N},\ \ \overline{x} \vee p_{N_t} \vee \overline{p_N},\ \ p_{N_f} \vee p_{N_t} \vee \overline{p_N}.$

In the Ref. [4], it is proposed two-clause coding, which is more efficient than three-clause one but only applicable to BDDs that represent monotonic functions. Note that PB-constraints in standard form are monotonically decreasing, but not for those in band form. We show that two-clause coding is possible for some nodes of ROBDDs created from PB-constraints in band form.

A node N is *monotonically increasing* (resp. *decreasing*) if Fun(N) is monotonically increasing (resp. decreasing) with respect to its selector variable. Such nodes in an ROBDD constructed in Sect. 3 are easily found from the following proposition.

Proposition 12: Let N be a node with selector variable x_i such that Fun(N) is equivalent to $L \leq a_i\ell_i + \cdots + a_1\ell_1 \leq H$ and

- $a_i + \cdots + a_1 \in H$ and $\ell_i = x_i$ (resp. $\ell_i = \overline{x_i}$), or
- $-1 \in L$ and $\ell_i = \overline{x_i}$ (resp. and $\ell_i = x_i$),

then N is monotonically increasing (resp. decreasing).

Proof We consider only the case that $a_i + \cdots + a_1 \in H$ and $\ell_i = x_i$. The formula is equivalent to $l \leq e_i$ for any $l \in L$, where e_i denotes $a_i\ell_i + \cdots + a_1\ell_1$. Suppose $\sigma_{[x_i \mapsto 0]}$ satisfies $l \leq e_i$ for the selector variable x_i. Since $a_i > 0$, $\sigma_{[x_i \mapsto 1]}$ satisfies $l \leq e_i$. The other cases are shown similarly. □

We encode the formula in (3) as follows.

(Dec) For monotonically decreasing nodes N,

$$\overline{x} \vee p_{N_t} \vee \overline{p_N},\ \ p_{N_f} \vee \overline{p_N}.$$

(Inc) For monotonically increasing nodes N,

$$x \vee p_{N_f} \vee \overline{p_N},\ \ p_{N_t} \vee \overline{p_N}.$$

(Three) For the other nodes N, three-clause coding (Three).

In the rest of the section, we show the correctness of this partial two-clause coding.

Lemma 13: Fun(N) is satisfied by a valuation σ that satisfies $p_N \wedge$ Cnf(N)

Proof By induction on the structure of the BDD.

If N is a terminal node N_1, it is trivial because Fun(N_1) = 1. If N is a terminal node N_0, we have no valuation σ that satisfies $p_{N_0} \wedge \overline{p_{N_0}}$. Otherwise, N is a decision node with a selector variable x, true-child N_t, and false-child N_f. From the assumption, σ satisfies p_N.

- Consider the case that N is encoded by (Three). If $\sigma(x) = 1$, $\sigma(p_{N_t})$ must be 1 from the second clause. Considering the sub-BDD rooted N_t, Cnf(N_t) is included in Cnf(N). Combining these, σ satisfies Fun(N_t) by induction hypothesis. From the definition of Fun, σ satisfies Fun(N).
  If $\sigma(x) = 0$, $\sigma(p_{N_f})$ must be 1 from the first clause. Considering the sub-BDD rooted N_f, Cnf(N_f) is included in Cnf(N). Combining these, σ satisfies Fun(N_f) by induction hypothesis. From the definition of Fun, σ satisfies Fun(N).
- Consider the case that N is encoded by (Dec). If $\sigma(x) = 1$, $\sigma(p_{N_t})$ must be 1 from the first clause. Considering the sub-BDD rooted N_t, Cnf(N_t) is included in Cnf(N). Combining these, σ satisfies Fun(N_t) by induction hypothesis. From the definition of Fun, σ satisfies Fun(N).
  If $\sigma(x) = 0$, $\sigma(p_{N_f})$ must be 1 from the second clause. Considering the sub-BDD rooted N_f, Cnf(N_f) is included in Cnf(N). Combining these, σ satisfies Fun(N_f) by induction hypothesis. From the definition of Fun, σ satisfies Fun(N).
- The case encoded by (Inc) is similar to (Dec). □

For proving the reverse, we introduce a notion of active nodes to construct an extended valuation.

Definition 14: For a given valuation σ and a BDD rooted by R, the set A_σ of *active nodes* is defined as a minimal set satisfying the following conditions:

- $R \in A_\sigma$.
- $N_t \in A_\sigma$ for the true-child N_t of an $N \in A_\sigma$, if $\sigma(x) = 1$ or N is monotonically increasing where x is the selector variable of N.
- $N_f \in A_\sigma$ for the false-child N_f of an $N \in A_\sigma$, if $\sigma(x) = 0$ or N is monotonically decreasing where x is the selector variable of N.

If σ satisfies Fun(R) for a BDD rooted by R, terminal node N_0 are not active from the following lemma.

Lemma 15: For a BDD rooted by R, σ satisfies Fun(N) for any active nodes N if σ satisfies Fun(R).

Proof By induction on the definition of the active nodes. In the case that $N = R \in A_\sigma$, it is trivial.

Consider the case that $N_t \in A_\sigma$, $\sigma(x) = 1$, and the selector variable of N is x. From the definition of Fun(N), σ satisfies Fun(N_t).

Consider the case that $N_t \in A_\sigma$, which is the true-child of a monotonically increasing node $N \in A_\sigma$ with selector variable x. We can assume $\sigma(x) = 0$, otherwise it is subsumed by the previous case. From the definition of Fun(N), σ satisfies Fun(N_f). Since N is monotonically increasing, $\sigma_{[x \mapsto 1]}$ also satisfies Fun(N). Thus $\sigma_{[x \mapsto 1]}$ satisfies Fun(N_t). Since Fun(N_t) does not contain x, σ satisfies Fun(N_t).

We can prove for the last case similarly. □

Lemma 16: For a BDD rooted by R, let σ be a valuation that satisfies Fun(R). Let σ' be a PVar(R)-extension of σ such that $\sigma'(p_N) = 1$ for active nodes N, and $\sigma'(p_N) = 0$ for the other nodes N. Then σ' satisfies $p_R \wedge$ Cnf(R).

Proof Since R is active, $\sigma'(p_R) = 1$.

If N is a terminal node N_1, then the clause is 1. If N is a terminal node N_0, then the clause is $\overline{p_{N_0}}$. Since N_0 is not active by Lemma 15, $\sigma'(p_{N_0}) = 0$.

If N is a decision node with a selector variable x, a true-child N_t, and a false-child N_f.

- Consider the case that N is active and encoded by (Three). In the case that $\sigma(x) = 1$, N_t is also active. Thus, $\sigma'(p_{N_t}) = 1$. In the case that $\sigma(x) = 0$, N_f is also active. Thus $\sigma'(p_{N_f}) = 1$.
- Consider the case that N is active and encoded by (Dec). Since N_f is active, $\sigma'(p_{N_f}) = 1$. If $\sigma(x) = 1$, N_t is also active and hence $\sigma'(p_{N_t}) = 1$. In either of the cases, σ' satisfies the clauses (Dec).
- The case that N is active and encoded by (Inc) is similar to (Dec).
- If N is not active, then $\sigma'(p_N) = 0$. Thus, σ' satisfies all the clauses (Three), (Dec) and (Inc). □

From Lemmas 13 and 16, the following theorem is shown.

Theorem 17: For a BDD rooted by R, $p_R \wedge$ Cnf(R) and Fun(R) is equisatisfiable.

5. Binary/Alternative Searches for Optimization

The PB-optimization answers the minimal value of a given goal expression among valuations that satisfy the PB-constraints, where goal expressions are in forms of $b_1\ell_1 + \cdots + b_n\ell_n$ for integers $b_1, \ldots, b_n$.

Most of PB-solvers such as sat4j and MiniSat+ employ the sequential search, in which the satisfiability is firstly checked without considering the goal expression g. Once a satisfiable valuation is obtained, the best known goal value k is calculated. By augmenting clauses that represents $g < k$, invoke SAT-solver. Repeating this until resulting unsatifiable, the best goal value is fixed as the last k. This can be implemented without restarting SAT-solvers, because recent solvers allow incremental solving; adding clauses incrementally, and continuing solving.

This sequential search is not so bad, because in most cases an improvement of the known goal value is more than one. However, a lot of SAT-solver executions are sometimes necessary. Thus it is natural to consider binary search strategies.

This section describes on binary search strategies for a goal minimization. Avoiding a SAT-solver restart, we use a SAT-solver function that solves under tentatively assuming the given literal instead of adding it. Let k be the best known goal value, and l be the greatest known lower bound, which is initially the sum of negative coefficients b_i's. By introducing a fresh variable p, we add a constraint for $p \implies g < \lfloor (k-l)/2 \rfloor$, and solve them under assumption that $p = 1$. If it is satisfiable, the known best value is renewed from the obtained valuation and proceeds. Otherwise the greatest known lower bound is renewed to $\lfloor (k-l)/2 \rfloor$ and proceeds.

Binary search strategies sometimes take unnecessary executions. For example, we suppose a goal $g = 1000x + 4000y$ and its best goal value is 1000. By the sequential search, after we got 1000 it is enough one more SAT-solver execution with $g < 1000$. By the above binary search strategy, however, considerable times of SAT-solving are necessary with $g < 500$, $g < 750$, and so on. For this, a preprocess of goal expressions is effective; dividing coefficients by their greatest common divisor (GCD). The example is transformed to $x + 4y$.

The simplification by GCD takes no effect for the goal $g' = x + 1000y$ if the best goal value is 1000. For this, we propose an *alternating strategy* that adopts the sequential (resp. binary) search after UNSAT (resp. SAT) answer obtained. Suppose the best known goal value is 1000 for the goal, two-time executions with $g' < 500$ (due to binary one) and $g' < 1000$ (due to sequential one) are enough.

6. Evaluation

We implemented our findings into our tool, named GPW,

Table 1 PB-competition instances.

coding/form/search	DEC-SML		OPT-BIG		OPT-SML		total	constraints		BDD nodes		Sat	UnSat
	Sat	UnSat	Opt	UnSat	Opt	UnSat		std	band	2cl	3cl	calls	calls
2cl/band/seq	170	*213*	48	58	*348*	33	870	379K	5.0K	193M	5M	8.2K	0.5K
2cl/band/bin			70	58	346	33	890	394K	5.0K	282M	3M	2.0K	3.0K
2cl/band/alt			72	58	*348*	33	894	395K	5.0K	395M	3M	2.7K	1.9K
2cl/std/seq	170	*213*	48	58	346	33	868	389K	0	203M	0	8.0K	0.5K
2cl/std/bin			75	*59*	346	33	*896*	422K	0	348M	0	2.0K	3.0K
2cl/std/alt			*76*	58	346	33	*896*	416K	0	387M	0	2.7K	1.9K
3cl/band/seq	*172*	211	57	58	345	33	876	386K	5.0K	0	210M	10.8K	0.5K
3cl/band/bin			58	58	333	33	865	378K	5.0K	0	149M	1.7K	2.7K
3cl/band/alt			59	58	336	33	869	376K	5.0K	0	154M	2.4K	1.7K
3cl/std/seq	171	210	56	58	341	32	868	380K	0	0	209M	10.6K	0.5K
3cl/std/bin			59	58	332	32	862	373K	0	0	154M	1.7K	2.7K
3cl/std/alt			59	58	335	32	865	367K	0	0	159M	2.4K	1.8K
MiniSat+ mode	*172*	210	53	*59*	335	33	862	372K	5.5K	0	219M	10.4K	0.5K

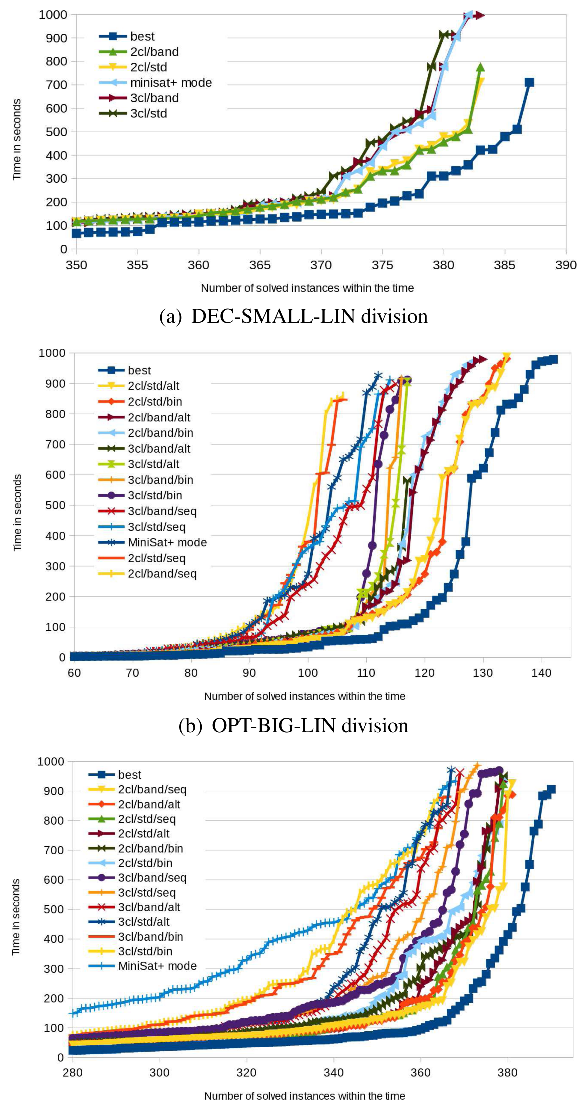
which is constructed based on Minisat+ [2] version 1.0. The major extensions are summarized as follows:

- Minisat+ has a function to generate clauses via BDDs constructed from each PB-constraint in band form. Thus we implemented the construction of ROBDDs.
- Minisat+ employs the three-clause coding. Thus we implemented two-clause coding for nodes having monotonic property.
- Minisat+ has the incremental SAT-solving for the sequential search in the goal minimization. We implemented the incremental SAT-solving for the binary search and the alternative search.

We performed experiments on a machine equipped with dual Xeon W5590 (3.33GHz, 4core 8thread, L2cache4*256KB, and L3cache 8MB) processors and 48GB memory. MiniSat version 1.13 is used as an underlying SAT-solver, which is included in Minisat+, as an underlying solver. The benchmarks are 1683 instances in total; 452, 532, and 699 instances used in DEC-SMALLINT-LIN, OPT-BIGINT-LIN, and OPT-SMALLINT-LIN divisions of Pseudo-Boolean Competition 2010, respectively. The GPW detects at least one band form in 26, 417, and 100 instances for the respective divisions.

Table 1 shows the number of instances that different methods could solve within 1000 seconds timeout. The rows correspond to coding methods in GPW where 2cl/3cl present two-clause/three-clause coding, std/band present BDD construction from standard-form/band-form, seq/bin/alt present sequential/binary/alternative strategies, and MiniSat+ mode adopts three clause, band form, sequential strategy, and (non-RO) BDD construction. The first seven columns correspond to the divisions and total numbers, where Sat/UnSat/Opt present solved instances with satisfiable/unsatisfiable/optimized results. The rest of columns correspond to the total numbers of constraints processed as standard/band form, BDD nodes (coded by 2-clause, and by 3-clause), and Sat/UnSat solver calls for solved instances.

Figure 4 shows cactus plots of the results, which indicate the number of solved instances within the time. In the figures, lines located in lower and more right side show better results. Note that the "best" solver is a virtual one which runs all solvers by the different methods in parallel and takes the best result.



(a) DEC-SMALL-LIN division

(b) OPT-BIG-LIN division

(c) OPT-SMALL-LIN division

Fig. 4 Cactus plots for PB-competition instances.

Generally, processing in standard form with the two-clause coding and with the binary or alternative search cores the best. Let's see more detailed analysis.

The binary search results better than the sequential search, and the alternative search seems the best. This is reasoned from the following facts:

- The sequential search requires more solver-calls than others,
- UNSAT-calls are generally much heavier than SAT-calls, and
- once obtained the minimal goal value, the sequential (resp. binary, alternative) search requires one (resp. many, two) UNSAT-calls.

It is easily shown that the number of solver-calls with the alternative search is not more than double of those with the binary search.

The effect of the band form varies according to coding/search methods and divisions of the instances. Generally, the band form has advantages of smaller memory use and shorter BDD-construction time than the standard form. The band form has an advantage of the number of solved instances in the following cases:

- Instances are processed by the three-clause coding,
- Instances are solved by the sequential search strategy.
- Instances in OPT-SMALL-LIN division are processed.

The band form has a disadvantage for instances in OPT-BIG-LIN division processed by the two-clause coding.

From the other views, ROBDD construction takes effect, because processing 3cl/band/seq with ROBDD scores better than MiniSat+ mode (3cl/band/seq with non-ROBDD). Two-clause coding method takes effect together with standard form, but also with band form, if it is combined with the alternative or binary strategy.

7. Concluding Remarks

This paper proposed the following methods:

- an efficient ROBDD construction algorithm for PB-constraints in band form by modifying the algorithm in the Ref. [4] designed for standard form,
- a partial two-clause coding of BDD, which produces two clauses for a monotonic node and three clauses for a non-monotonic node, and
- a binary search and an alternative search for goal minimization, which allow inremental SAT-solving.

It appears that the best combination of methods is the alternative (or binary search) strategy and the ROBDD construction from the standard form with the two-clause coding. The construction from the band form is effective under the three-clause coding, or the sequential strategy.

We can choose the band or standard form in ROBDD construction for each constraint in an instance. Thus it is interesting to find good choice strategies.

Acknowledgments

We thank Harald Zankl for discussing on band form. Partially supported by the Austrian Science Fund (FWF) project I963 and the Japan Society for the Promotion of Science.

References

[1] C. Herde, "Extending DPLL for Pseudo-Boolean constraints," in Efficient Solving of Large Arithmetic Constraint Systems with Complex Boolean Structure, pp.37–58, Vieweg+Teubner, 2011.
[2] N. Eén and N. Sörensson, "Translating Pseudo-Boolean constraints into SAT," J. Satisfiability, Boolean Modeling and Computation (JSAT), vol.2, no.1-4, pp.1–26, 2006.
[3] O. Bailleux, Y. Boufkhad, and O. Roussel, "A translation of Pseudo Boolean constraints to SAT," J. satisfiability, Boolean modeling and computation (JSAT), vol.2, no.1-4, pp.191–200, 2006.
[4] I. Abío, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and V. Mayer-Eichberger, "A new look at BDDs for Pseudo-Boolean constraints," J. Artificial Intelligence Research (JAIR), vol.45, pp.443–480, 2012.
[5] R.E. Bryant, "Graph-based algorithms for boolean function manipulation," IEEE Trans. Comput., vol.35, no.8, pp.677–691, Aug. 1986.

Masahiko Sakai completed graduate course of Nagoya University in 1989 and became Assistant Professor, where he obtained a D.E. degree in 1992. From April 1993 to March 1997, he was Associate Professor in JAIST. In 1996 he stayed at SUNY at Stony Brook for six months as Visiting Research Professor. From April 1997, he was Associate Professor in Nagoya University. Since December 2002, he has been Professor. He is interested in term rewriting system, verification of specification, software generation and constraint solvers. He received the Best Paper Awards from IEICE in 1992 and 2011. He is a member of JSSST and IPSJ.

Hidetomo Nabeshima recieved a D.E. degree from Kobe University in 2001. From 2001 to 2008, he was Assistant Professor in University of Yamanashi. Since 2009, he has been Associate Professor. He is interested in propositional satisfiability testing (SAT) and automated reasoning in first order logic. He received the Takahashi Award for the Best Presentation and Paper and the Best Software Paper Awards from JSSST in 2011 and 2014, respectively. His SAT solver won the first and second prizes for UNSAT track of application category at SAT competition 2011 and 2013, respectively. He is a member of JSSST and JSAI.