BDDs for Pseudo-Boolean Constraints – Revisited

Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell[⋆]

Abstract. Pseudo-Boolean constraints are omnipresent in practical applications, and therefore a significant effort has been devoted to the development of good SAT encoding techniques for these constraints. Several of these encodings are based on building Binary Decision Diagrams (BDDs) and translating these into CNF. Indeed, as we argue here, BDD-based encodings have important advantages, such as *sharing* the same BDD for representing many constraints.
Here we first prove that, unless NP = Co-NP, there are Pseudo-Boolean constraints that admit no variable ordering giving a polynomial (Reduced, Ordered) BDD. As far as we know, this result is new (in spite of some misleading information in the literature, see below). It gives several interesting insights, also relating proof complexity and BDDs.
But, more interestingly for practice, here we also show how to overcome this theoretical limitation by *coefficient decomposition*. This allows us to give the first polynomial arc-consistent BDD-based encoding for Pseudo-Boolean constraints.

1 Introduction

In this paper we study Pseudo-Boolean constraints (PB constraints for short), that is, constraints of the form $a_1x_1 + \cdots + a_nx_n \# K$, where the a_i and K are integer coefficients, the x_i are Boolean (0/1) variables, and the relation operator # belongs to $\{<, >, \leq, \geq, =\}$. We will assume that # is $\leq$ and the a_i and K are positive since other cases can be easily reduced to this one.

Such a constraint is a Boolean function $C\colon \{0,1\}^n \rightarrow \{0,1\}$ that is (dual) monotonic in the sense that any solution for C remains a solution after flipping inputs from 1 to 0. Therefore these constraints can be expressed by a set of clauses with only negative literals. For example, each clause could simply define a (minimal) subset of variables that cannot be simultaneously true. Note however that not every such a monotonic function is a PB constraint. For example, the function expressed by the two clauses $\overline{x}_1 \vee \overline{x}_2$ and $\overline{x}_3 \vee \overline{x}_4$ has no (single) equivalent PB constraint $a_1x_1 + \cdots + a_nx_n \leq K$ (since wlog. $a_1 \geq a_2$ and $a_3 \geq a_4$, and then also $\overline{x}_1 \vee \overline{x}_3$ is needed). Hence, even among the monotonic Boolean functions, PB constraints are a rather restricted class (see also [J.S07]).

PB constraints are omnipresent in practical SAT applications, not just in typical 0-1 linear integer problems, but also as an ingredient in new SAT approaches to, e.g., cumulative scheduling [SFSW09], so it is not surprising that a

[⋆] Technical University of Catalonia (UPC), Barcelona.

significant number of SAT encodings for these constraints have been proposed in the literature. Here we are interested in encoding a PB constraint C by a clause set S (possibly with auxiliary variables) that is not only equisatisfiable, but also (generalized) *arc-consistent*: given a partial assignment A, if x_i is false in every extension of A satisfying C, then unit propagating A on S sets x_i to false.

To our knowledge, the only polynomial arc-consistent encoding so far was given by Bailleux, Boufkhad and Roussel [BBR09]. Other existing encodings are based on building (forms of) Binary Decision Diagrams (BDDs) and translating these into CNF. Although [BBR09] is not BDD-based, our motivation to revisit BDD-based encodings is still twofold:

Example 1. Consider the constraint $3x_1 + 2x_2 + 4x_3 \leq 5$ and the constraint $30001x_1 + 19999x_2 + 39998x_3 \leq 50007$. Both are clearly equivalent: the Boolean function they represent can be expressed, e.g., by the clauses $\overline{x}_1 \vee \overline{x}_3$ and $\overline{x}_2 \vee \overline{x}_3$. However, encodings like the one of [BBR09] heavily depend on the concrete coefficients of each constraint, and generate a significantly larger SAT encoding for the second one. Since, given a variable ordering, (Reduced, Ordered) BDDs are a canonical representation for Boolean functions [Bry86], i.e., each Boolean function has a unique ROBDD, a ROBDD-based encoding will treat both constraints equivalently. □

The second reason for revisiting BDDs is that in practical problems numerous PB constraints exist that share variables among each other. Representing them all as a single BDD has the potential of generating a much more compact SAT encoding that is moreover likely to have better propagation properties.

Related work. The same authors of [BBR09] proposed an encoding "very close to those using a BDD and translating it into clauses" [BBR06]. It is arc-consistent, but an example of a PB constraint family is given in [BBR06] for which their kind of *non-reduced* BDDs, with *their concrete variable ordering* is exponentially large. However, as we show here, ROBDDs for this family are polynomial. Their method works as follows. Given the PB constraint $a_1x_1 + \cdots + a_nx_n \leq K$ with coefficients ordered from small to large, the root node is labelled with variable $D_{n,K}$, expressing that the sum of the first n terms is no more than K. Its two children are $D_{n-1,K}$ and $D_{n-1,K-a_n}$, which correspond to setting x_n to false and true, respectively, etc. Two binary and two ternary clauses per node express the relationships between the variables.

Example 2. The encoding of [BBR06] on $2x_1 + \cdots + 2x_{10} + 5x_{11} + 6x_{12} \leq 10$ is illustrated in Figure 1. Node $D_{10,5}$ represents $2x_1 + 2x_2 + \cdots 2x_{10} \leq 5$, whereas node $D_{10,4}$ represents $2x_1 + 2x_2 + \cdots 2x_{10} \leq 4$. The method fails to identify that both these PB constraints are equivalent and hence subtrees B and C will not be merged, yielding a much larger representation than with ROBDDs. □

On the other hand, Eén and Sörensson use ROBDDs in MiniSAT+ [ES06]. Their encoding uses six three-literal clauses per BDD node and is arc-consistent, but the proof of arc-consistency relies on a particular variable ordering. Regarding the size of their ROBDDs, they cite [BBR06] to say *"It is proven that*

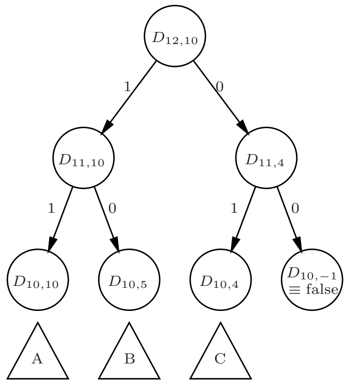

Fig. 1. Tree-like construction of [BBR06] for $2x_1 + \cdots + 2x_{10} + 5x_{11} + 6x_{12} \leq 10$

in general a PB-constraint can generate an exponentially sized BDD [BBR06]" which, as we have seen, cannot be concluded from that paper. Apart from their BDD-based encoding, [ES06] also suggests two alternative methods: one based on adder networks ($\mathcal{O}(n)$ in size but not arc-consistent) and another one based on sorting networks ($\mathcal{O}(n \log n)$ in size and not arc-consistent).

Finally, as we have already mentioned, [BBR09] presents an arc-consistent and polynomial translation (size $\mathcal{O}(n^2 \log n \log a_{max})$, i.e., it depends on the size of the coefficients) based on a network of unary adders.

Main contributions and organization of this paper:

- Subsection 3.2: The first, to our knowledge, PB constraint family for which ROBDDs with small-to-large variable ordering are exponential in size (and also for the large-to-small ordering).
- Subsection 3.3: A proof that, unless NP=co-NP, there are PB constraints that admit no polynomial-size ROBDD, independently of the variable order.
- Subsection 4.1: A proof that PB constraints whose coefficients are powers of two do admit polynomial-size BDDs.
- Subsections 4.2 and 4.3: An arc-consistent and polynomial (size $\mathcal{O}(n^3 \log a_{max})$) BDD-based encoding for PB constraints.
- Section 5: An arc-consistent SAT encoding of BDDS for monotonic functions, a more general class of Boolean functions than PB constraints. This encoding uses only one binary and one ternary clause per node (the standard if-then-else encoding for BDDs used in, e.g., [ES06], requires six ternary clauses per node). Moreover, this translation works for any BDD variable ordering.

2 Preliminaries

We assume the reader is familiar with the basic notions of propositional logic. Otherwise, basic definitions can be found in [BHvMW09]. constraints of the form $a_1x_1 + \cdots + a_nx_n \# K$, where the a_i and K are integer coefficients, the x_i are Boolean (0/1) variables, and the relation operator $\#$ belongs to $\{<, >, \leq, \geq, =\}$. We will assume that $\#$ is $\leq$ and the a_i and K are positive since other cases can be easily reduced to this one [1]: (i) changing into $\leq$ is straightforward if coefficients

[1] An =-constraint can be split into a $\leq$-constraint and a $\geq$-constraint. Here we consider (arc-)consistency for the latter two isolatedly, not for the original =-constraint.

can be negative; (ii) replacing $-ax$ by $a(1-x)-a$; (iii) replacing $(1-x)$ by $\overline{x}$. Negated variables like $\overline{x}$ can be handled as positive ones or, alternatively, replaced by a frexh x' and adding the clauses $x \vee x'$ and $\overline{x} \vee \overline{x}'$.

Our main goal is to find SAT encodings for PB constraints. That is, given a PB-constraint C, construct an equisatisfiable clause set (a CNF) S such that any model for S restricted to the variables of C is a model of C. Two extra properties are sought: (i) *consistency checking by unit propagation* or simply *consistency*: whenever a partial assignment A cannot be extended to a model for C, unit propagation on S and A produces a contradiction (a literal l and its negation $\overline{l}$); and (ii) (generalized) *arc-consistency* (again by unit propagation): given an assignment A that can be extended to a model of C, but such that $A \cup \{x\}$ cannot, unit propagation on S and A produces $\overline{x}$. More concretely, we will use BDDs for finding such encodings, as illustrated by the following example.

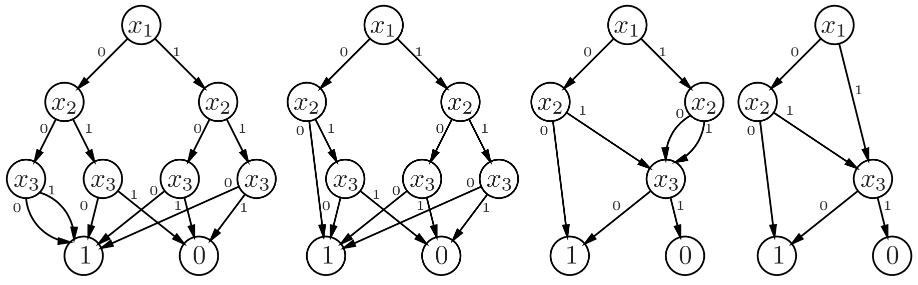

Fig. 2. Construction of a BDD for $2x_1 + 3x_2 + 5x_3 \leq 6$

Example 3. Figure 2 explains (one method for) the construction of a ROBDD for the PB constraint $2x_1 + 3x_2 + 5x_3 \leq 6$ and the ordering $x_1 < x_2 < x_3$. The root node has as *selector variable* x_1. Its *false child* represents the PB constraint assuming $x_1 = 0$ (i.e., $3x_2 + 5x_3 \leq 6$) and its *true child* represents $2 + 3x_2 + 5x_3 \leq 6$, that is, $3x_2 + 5x_3 \leq 4$. The two children have the next variable in the ordering (x_2) as selector, and the process is repeated until we reach the last variable in the sequence. Then, a constraint of the form $0 \leq K$ is the *True* node (1 in the figure) if $K \geq 0$ is positive, and the *False* node (0) if $K < 0$. This construction (leftmost in the figure), is known as an Ordered BDD. For obtaining a Reduced Ordered BDD (BDD for short in the rest of the paper), two reductions are applied until fixpoint: removing nodes with identical children (as done with the leftmost x_3 node in the second BDD of the figure), and merging isomorphic subtrees, as done for x_3 in the third BDD. The fourth final BDD is a fixpoint. For a given ordering, BDDs are a canonical representation of Boolean functions: each Boolean function has a unique BDD. BDDs can be encoded in CNF by introducing an auxiliary variable a for every node. If the selector variable of the node is x and the auxiliary variables for the false and true child are f and t, respectively, add the if-then-else clauses:

$$\begin{array}{lll} \overline{x} \wedge \overline{f} \rightarrow \overline{a} & x \wedge \overline{t} \rightarrow \overline{a} & \overline{f} \wedge \overline{t} \rightarrow \overline{a} \\ \overline{x} \wedge f \rightarrow a & x \wedge t \rightarrow a & f \wedge t \rightarrow a \end{array} \qquad \square$$

In what follows, the *size* of a BDD is its number of nodes. We will say that a BDD *represents* a PB constraint if they are the same Boolean function. Given an assignment A over the variables of a BDD, we will talk about the *path induced by A* as the path that starts at the root of the BDD and at each step, moves to the false (true) child of a node iff its selector variable is false (true) in A.

3 Exponential BDDs for PB Constraints

In this section, we prove that, unless NP=co-NP, there are PB constraints whose BDDs are all exponential, regardless of the variable ordering. We start by defining the notion of the *interval* of a PB constraint. After that, we consider two families of PB constraints and we study the size of their BDDs. Finally, we prove the main result of this section.

3.1 Intervals

Example 4. Consider the constraint $2x_1 + 3x_2 + 5x_3 \leq 6$. Since no combination of its coefficients adds to 6, the constraint is equivalent to $2x_1 + 3x_2 + 5x_3 < 6$, and hence to $2x_1 + 3x_2 + 5x_3 \leq 5$. This process cannot be repeated again since 5 can be obtained with the existing coefficients.

Similarly, we could try to increase the right-hand side of the constraint. However, there is a combination of the coefficients that adds 7, which implies that the constraint is not equivalent to $2x_1 + 3x_2 + 5x_3 \leq 7$. All in all, we can state that the constraint is equivalent to $2x_1 + 3x_2 + 5x_3 \leq K$ for any $K \in [5, 6]$. It is trivial to see that the set of valid K's is always an interval. $\square$

Definition 1 *Let C be a constraint of the form $a_1x_1 + \cdots + a_nx_n \leq K$. The* interval *of C consists of all integers M such that $a_1x_1 + \cdots + a_nx_n \leq M$, seen as a Boolean function, is equivalent to C.*

In the following, given a BDD representing a PB constraint and a node ν, we will refer to *the interval of ν* as the interval of the constraint represented by the BDD rooted at ν. Unless stated otherwise, the ordering used in the BDD will be $x_1 < x_2 < \ldots < x_n$.

Proposition 2 *If $[\beta, \gamma]$ is the interval of a node ν with selector variable x_i then:*

1. *There is an assignment $\{x_j = v_j\}_{j=i}^{n}$ such that $a_iv_i + \cdots + a_nv_n = \beta$.*
2. *There is an assignment $\{x_j = v_j\}_{j=i}^{n}$ such that $a_iv_i + \cdots + a_nv_n = \gamma + 1$.*
3. *There is an assignment $\{x_j = v_j\}_{j=1}^{i-1}$ such that $K - a_1v_1 - a_2v_2 - \cdots - a_{i-1}v_{i-1} \in [\beta, \gamma]$*
4. *Take $h < \beta$. There exists an assignment $\{x_j = v_j\}_{j=i}^{n}$ such that $a_iv_i + \cdots + a_nv_n > h$ and its path goes from ν to True.*
5. *Take $h > \gamma$. There exists an assignment $\{x_j = v_j\}_{j=i}^{n}$ such that $a_iv_i + \cdots + a_nv_n \leq h$ and its path goes from ν to False.*
6. *The interval of the True node is $[0, \infty)$.*
7. *The interval of the False node is $(-\infty, -1]$. Moreover, it is the only interval with negative values.*

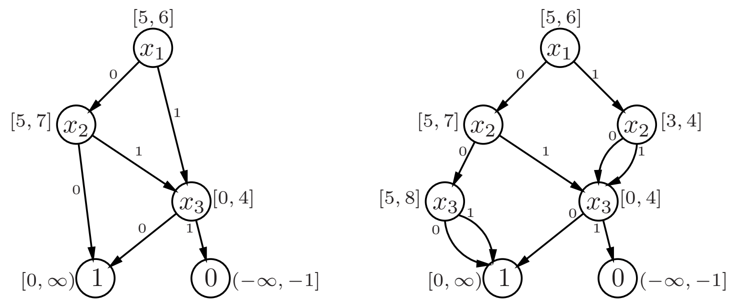

Fig. 3. Intervals of the BDD for $2x_1 + 3x_2 + 5x_3 \leq 6$

We now prove that, given a BDD for a PB constraint, one can easily compute the intervals for every node bottom-up. We first start with an example.

Example 5. Let us consider again the constraint $2x_1 + 3x_2 + 5x_3 \leq 6$. Assume that all variables appear in every path from the root to the leaves (otherwise, add extra nodes as in the rightmost BDD of Figure 3). Assume now that we have computed the intervals for the two children of the root (rightmost BDD in Figure 3). This means that the false child of the root is the BDD for $3x_2 + 5x_3 \leq [5, 7]$ and the true child the BDD for $3x_2 + 5x_3 \leq [3, 4]$. Assuming x_1 to be false, the false child would also represent the constraint $2x_1 + 3x_2 + 5x_3 \leq [5, 7]$, and assuming x_1 to be true, the true child would represent the constraint $2x_1 + 3x_2 + 5x_3 \leq [5, 6]$. Taking the intersection of the two intervals, we can infer that the root node represents $2x_1 + 3x_2 + 5x_3 \leq [5, 6]$. □

More formally, the interval of every node can be computed as follows:

Proposition 3 *Let $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq K$ be a constraint, and let $\mathcal{B}$ be its BDD with the order $x_1 < x_2 < \ldots < x_n$. Consider a node ν with selector variable x_i, false child ν_f (with selector variable x_f and interval $[\beta_f, \gamma_f]$) and true child ν_t (with selector variable x_t and interval $[\beta_t, \gamma_t]$). The interval of ν is $[\beta, \gamma]$, with:*

$$\begin{aligned}
&\beta = \max\{\beta_f + a_{i+1} + \cdots + a_{f-1},\ \ \beta_t + a_i + a_{i+1} + \cdots + a_{t-1}\},\\
&\gamma = \min\{\gamma_f,\ \ \gamma_t + a_i\}.
\end{aligned}$$

If in every path from the root to the leaves of the BDD all variables were present, the definition of β would be much simpler ($\beta = \max\{\beta_f, \beta_t + a_i\}$). The other coefficients are necessary to account for the variables that have been removed due to the BDD reduction process.

3.2 Some families of PB constraints and their BDD size

We start by revisiting the family of PB constraints given in [BBR06], where it is proved that, for their concrete variable ordering, their non-reduced BDDs grow exponentially for this family. Here we prove that ROBDDs are polynomial for this family, and that this is even independent of the variable ordering. The family is defined by considering a, b and n positive integers such that $\sum_{i=1}^{n} b^i < a$. The coefficients are $\omega_i = a+b^i$ and the right-hand side of the constraint is $K = a{\cdot}n/2$. We will first prove that the constraint $C : \omega_1 x_1 + \cdots + \omega_n x_n \leq K$ is equivalent to the cardinality constraint $C' : x_1 + \cdots + x_n \leq n/2 - 1$. For simplicity, we assume that n is even.

– Take an assignment satisfying C'. In this case, there are at most $n/2 - 1$ variables x_i assigned to true, and the assignment also satisfies C since: $\omega_1 x_1 + \cdots + \omega_n x_n \leq \displaystyle\sum_{i=n/2+2}^{n} \omega_n = (n/2-1)a + \sum_{i=n/2+2}^{n} b^n < K - a + \sum_{i=1}^{n} b^i < K.$
– Consider now an assignment not satisfying C'. In this case, there are at least $n/2$ true variables in the assignment and it does not satisfy C either:
$$\omega_1 x_1 + \cdots + \omega_n x_n \geqslant \sum_{i=1}^{n/2} \omega_i = (n/2) \cdot a + \sum_{i=1}^{n/2} b^i > (n/2) \cdot a = K.$$

Since the two constraints are equivalent and BDDs are canonical, the BDD representation of C and C' are the same. But the BDD of C' is known to be of quadratic size because it is a cardinality constraint (see, for instance, [BBR06]).

Theorem 4 *There exists a family of PB constraints parameterized by n, whose ROBDDs grow exponentially in n when ordering the variables according to their coefficients from small to large. The same happens ordering from large to small.*

Proof. We consider constraints of the form $a_1 x_1 + \cdots + a_{4n} x_{4n} \leq K$. It is convenient to describe the coefficients in binary notation:

$$\begin{array}{rcrcl}
 & & & \overbrace{}^{2n} & \\
a_1 & = & 0\,0\,0 & 0\,0\,\cdots\,0\,\mathbf{1} & = 1 \\
a_2 & = & 0\,0\,0 & 0\,0\,\cdots\,\mathbf{1}\,0 & = 2 \\
\cdots & & & \iddots & \\
a_{2n-1} & = & 0\,0\,0 & 0\,\mathbf{1}\,\cdots\,0\,0 & \\
a_{2n} & = & 0\,0\,0 & \mathbf{1}\,0\,\cdots\,0\,0 & = 2^{2n-1} \\
 & & & & \\
a_{2n+1} & = & \mathbf{1}\,0\,0 & 0\,0\,\cdots\,0\,\mathbf{1} & \\
a_{2n+2} & = & \mathbf{1}\,0\,0 & 0\,0\,\cdots\,\mathbf{1}\,0 & \\
\cdots & & & \iddots & \\
a_{4n-1} & = & \mathbf{1}\,0\,0 & 0\,\mathbf{1}\,\cdots\,0\,0 & \\
a_{4n} & = & \mathbf{1}\,0\,0 & \mathbf{1}\,0\,\cdots\,0\,0 & \\
 & & & & \\
K & = & d_m \ldots\ d_0\,0\,0 & \mathbf{1}\,\mathbf{1}\,\cdots\,\mathbf{1}\,\mathbf{1} &
\end{array}$$

where $d_m \ldots d_0$ is the binary representation of n. Note that, to sum to exactly K, one needs exactly n coefficients of the bottom half (between a_{2n+1} and a_{4n}) to obtain the digits $d_m \ldots d_0$, and that, once such a subset is chosen, a unique subset of exactly n coefficients of the top half exists that will complete the $11 \ldots 11$ suffix of K. Reversely, for each subset of size n of the top half, a unique subset of size n of the bottom half exists that complements it to sum exactly K. Now consider a BDD ordered $x_1 < \cdots < x_{4n}$, and any two distinct assignments T and T' for $x_1 \ldots x_{2n}$ that set exactly n variables to true. Then T and T' induce paths that necessarily lead to different nodes of the BDD. To see this, wlog., assume that the sum of coefficients corresponding to true variables in T is smaller than the one of T'. Consider the assignment B to $x_{2n} \ldots x_{4n}$ that sets to true the unique size-n subset of the bottom half coefficients that sums to K for T (and hence exceeds K for T'). Then the PB constraint satisfies $T \cup B$, but not $T' \cup B$; hence B *distinguishes* the nodes for T and T'. Altogether, the BDD must have at least as many nodes as distinct assignments setting exactly n variables of the top half to true, i.e., an exponetial number, $\binom{2n}{n}$. For the large-to-small ordering exactly the same reasoning applies[2]. □

For the PB constraint family of the previous proof, it can be shown that the "interleaved" ordering $x_1 < x_{2n+1} < x_2 < x_{2n+2} < \ \ldots \ < x_{2n} < x_{4n}$ leads to a polynomial-sized BDD (the proof is non-trivial, but we had to omit it here due to space limitations). The next natural step would be to present a concrete family of PB constraints whose BDDs are always exponential regardless of the variable ordering. We have not been unable to find such a family. But in the next section we prove that, unless NP=co-NP, such a family must exist.

3.3 Probably there are no small BDDs for all PB constraints

Our goal is now to prove that, unless NP=co-NP, there are PB constraints all whose BDDs are exponential, independently of the variable ordering. The main ingredient is an algorithm that, given a BDD $\mathcal{B}$ and a PB constraint C : $a_1x_1 + \cdots + a_nx_n \leq K$ over the same set of variables, allows one to decide, in time polynomial in the size of the BDD, whether $\mathcal{B}$ represents C. Again, w.l.o.g., we assume that the BDD ordering is $x_1 < x_2 < \ldots < x_n$.

Given the BDD, the algorithm first computes, in a bottom-up manner, an interval for every node of the BDD, as explained in Proposition 3. Note that the cost of computing a single interval is $\mathcal{O}(n)$ and hence computing all intervals takes $\mathcal{O}(nm)$ time, where m is the BDD's size. After that, we know that $\mathcal{B}$ is a representation of C if and only if K belongs to the interval of the BDD root.

Theorem 5 *$\mathcal{B}$ is a BDD representing a PB constraint $a_1x_1 + \cdots + a_nx_n \leq K$ if, and only if, K belongs to the interval of the root of $\mathcal{B}$ computed by our algorithm.*

Proof. If $\mathcal{B}$ is a BDD representing C, then K belongs to the interval of the root by definition of interval (Def. 1). Moreover, Proposition 3 guarantees that our algorithm correctly computes such an interval.

Let us now assume that $\mathcal{B}$ is not a BDD representing C. Then, there exists an assignment $\{x_1 = v_1, \ldots, x_n = v_n\}$ that either satisfies C but leads to the False node in $\mathcal{B}$ or does not satisfy C but leads to the True node in $\mathcal{B}$.

[2] We thank Guillem Godoy for his help with this example.

Let us assume that the assignment satisfies C. The other case is analog to this one. In this case, we will prove that $\gamma_1 < K$, where $[\beta_1, \gamma_1]$ is the interval computed for the root node.

We define a sequence of nodes $\nu_1, \nu_2, \ldots, \nu_n, \nu_{n+1}$ as follows: ν_1 is the root of $\mathcal{B}$. If the selector variable of ν_1 is not x_1, $\nu_2 = \nu_1$. Otherwise, ν_2 is its false child if $v_1 = 0$ or its true child if $v_1 = 1$, and so on. By definition of the assignment, ν_{n+1} is the False node. If we let $[\beta_i, \gamma_i]$ be the computed interval for the node ν_i, we want to prove that every node ν_i satisfies $\gamma_i < a_{i+1}v_{i+1} + \cdots + a_n v_n$.

Since ν_{n+1} is the False node and its theoretical interval is $(-\infty, -1]$, it holds that $\gamma_{n+1} < 0$. Assume that it is true for every $k > i'$, and let us prove it for i'.

Let us assume that x_i is the selector variable of $\nu_{i'}$ (in this case, $i' \leq i$ by construction). There are two cases:

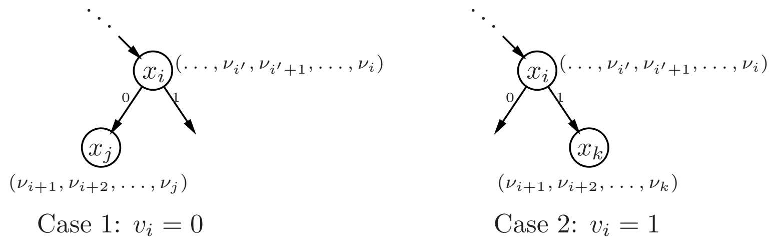
x_i
$(\ldots, \nu_{i'}, \nu_{i'+1}, \ldots, \nu_i)$
0
1
x_j
$(\nu_{i+1}, \nu_{i+2}, \ldots, \nu_j)$
Case 1: $v_i = 0$
x_i
$(\ldots, \nu_{i'}, \nu_{i'+1}, \ldots, \nu_i)$
0
1
x_k
$(\nu_{i+1}, \nu_{i+2}, \ldots, \nu_k)$
Case 2: $v_i = 1$

Fig. 4. Several ν's refer to the same BDD node. ν_j and ν_k are the last in the sequence.

- $v_i = 0$. Let us take j such that ν_j is the false child of $\nu_{i'}$ and the selector variable of ν_j is x_j (see Figure 4). Then, $\gamma_{i'} \leq \gamma_j$ by definition of the algorithm. Using the induction hypothesis:

$$\gamma_{i'} \leq \gamma_j < a_{j+1}v_{j+1} + \cdots + a_n v_n \leq a_{i'+1}v_{i'+1} + \cdots + a_n v_n.$$

- $v_i = 1$. Similarly, let us take k such that ν_k is the true child of $\nu_{i'}$ and the selector variable of ν_k is x_k (see Figure 4). Then, $\gamma_{i'} \leq \gamma_k + a_i$ by definition of the algorithm. Using the induction hypothesis and that $v_i = 1$:

$$\gamma_{i'} \leq a_i + \gamma_k < a_i + a_{k+1}v_{k+1} + \cdots + a_n v_n \leq a_{i'+1}v_{i'+1} + \cdots + a_n v_n.$$

Therefore, it holds that $\gamma_i < a_{i+1}v_{i+1} + \cdots + a_n v_n$ for every i. In particular, it holds for $i = 1$. Since the assignment satisfies the PB constraint by hypothesis, we have

$$\gamma_1 < a_1 v_1 + \cdots + a_n v_n \leq K,$$

and hence K does not belong to the theoretical interval of the root node. $\square$

Notice that if $\mathcal{B}$ is not the BDD of C some of the computed intervals might be empty. However, the algorithm will be able to compute the remaining intervals and, since the interval of the root node will be empty, the algorithm will also be correct. We are now ready for the following result.

Theorem 6 *Unless NP=co-NP, there are PB constraints that do not admit polynomial BDDS.*

Proof. A well-known NP-complete problem is the following (variant of the) *subset sum* problem: given a set integers $\{a_1, \dots, a_n\}$ and an integer K, decide whether there exists a subset of $\{a_1, \dots, a_n\}$ that sums to exactly K. Here we prove that if a polynomial-size BDD existed for every PB constraint then for every unsatisfiable subset sum problem a polynomial-size unsatisfiability certificate would exist, that could moreover be verified in polynomial time, thus collapsing NP and co-NP. Indeed, obviously, a subsetsum problem $(\{a_1, \dots, a_n\}, K)$ is unsatisfiable if, and only if, the PB constraints $a_1x_1 + \cdots + a_nx_n \leq K$ and $a_1x_1 + \cdots + a_nx_n \leq K - 1$ are equivalent, i.e., they are the same Boolean function. So if the subsetsum problem $(\{a_1, \dots, a_n\}, K)$ is unsatisfiable, and a polynomial-size BDD for $a_1x_1 + \cdots + a_nx_n \leq K$ existed, this BDD would also represent $a_1x_1 + \cdots + a_nx_n \leq K - 1$, which, as we proved in the previous theorem, can be checked in polynomial time (for both PB constraints at once). $\square$

We find it quite surprising that, even for the limited kind of monotonic functions that can be represented by a single PB constraint, the existence of polynomial-size BDDs would imply NP=co-NP. As said, to our knowledge it remains unknown whether there exists a family of PB constraints that admit no polynomial-size BDD. This situation is analogous to what happens with extended resolution in Cook's program for propositional proof complexity: it is unknown (again, to our knowledge) whether there exists a family of propositional problems that admit no polynomial-size extended resolution proof. So, finding successively more compact unsatisfiability certificates for subset sum might be an interesting alternative to Cook's program for attacking the NP vs co-NP question.

4 Avoiding Exponential BDDs

In this section we introduce our positive results. We restrict ourselves to a particular class of PB constraints, where all coefficients are powers of two. As we will show below, these constraints admit polynomial BDDs. Moreover, any PB constraint can be reduced to this class.

Example 6. Let us take the PB constraint $9x_1 + 8x_2 + 3x_3 \leq 10$. Considering the binary representation of the coefficients, this constraint can be rewritten into $(2^3x_{3,1} + 2^0x_{0,1}) + (2^3x_{3,2}) + (2^1x_{1,3} + 2^0x_{0,3}) \leq 10$ if we add the binary clauses expressing that $x_{i,r} = x_r$ for appropriate i and r. $\square$

4.1 Power-of-two PB constraints do have polynomial-size BDDs

Let us consider a PB constraints of the form:

$$\begin{array}{rl} C: & 2^0 \cdot \delta_{0,1} \cdot x_{0,1} \;+\; 2^0 \cdot \delta_{0,2} \cdot x_{0,2} \;+ \cdots +\; 2^0 \cdot \delta_{0,n} \cdot x_{0,n} \;+ \\ & 2^1 \cdot \delta_{1,1} \cdot x_{1,1} \;+\; 2^1 \cdot \delta_{1,2} \cdot x_{1,2} \;+ \cdots +\; 2^1 \cdot \delta_{1,n} \cdot x_{1,n} \;+ \\ & \cdots \qquad\qquad\qquad\qquad + \\ & 2^m \cdot \delta_{m,1} \cdot x_{m,1} + 2^m \cdot \delta_{m,2} \cdot x_{m,2} + \cdots + 2^m \cdot \delta_{m,n} \cdot x_{m,n} \leq K, \end{array}$$

where $\delta_{i,r} \in \{0,1\}$ for all i and r. Notice that every PB constraint whose coefficients are powers of 2 can be expressed in this way. Let us consider its BDD representation with the ordering $x_{0,1} < x_{0,2} < \ldots < x_{0,n} < x_{1,1} < \ldots < x_{m,n}$.

Lemma 7 *Let $[\beta, \gamma]$ be the interval of a node with selector variable $x_{i,r}$. Then 2^i divides β and $0 \leq \beta < (n+r-1) \cdot 2^i$.*

Proof. By Proposition 2.1, β can be expressed as a sum of coefficients all of which are multiples of 2^i, and hence β itself is a multiple of 2^i. By Proposition 2.7, the only node whose interval contains negative values is the False node, and hence $\beta \geqslant 0$. Now, using Proposition 2.3, there must be an assignment to the variables $\{x_{0,1}, \ldots, x_{i,r-1}\}$ such that $2^0\delta_{0,1}x_{0,1} + \cdots + 2^i\delta_{i,r-1}x_{i,r-1}$ belongs to the interval. Therefore:

$$\begin{aligned}\beta &\leq 2^0\delta_{0,1}x_{0,1} + \cdots + 2^i\delta_{i,r-1}x_{i,r-1} \leq 2^0 + 2^0 + \cdots + 2^i \\ &= n2^0 + n2^1 + \cdots + n2^{i-1} + (r-1)\cdot 2^i = n(2^i - 1) + 2^i(r-1) \\ &< 2^i(n+r-1)\end{aligned}$$

Corollary 8 *The number of nodes with selector variable $x_{i,r}$ is bounded by $n + r - 1$. In particular, the size of the BDD belongs to $\mathcal{O}(n^2 m)$.*

Proof. Let $\nu_1, \nu_2, \ldots, \nu_t$ be all the nodes with selector variable $x_{i,r}$. Let $[\beta_j, \gamma_j]$ the interval of ν_j. Note that such intervals are pair-wise disjoint since a non-empty intersection would imply that there exists a constraint represented by two different BDDs. Hence we can assume, w.l.o.g., that $\beta_1 < \beta_2 < \cdots < \beta_t$. Due to Lemma 7, we know that $\beta_j - \beta_{j-1} \geqslant 2^i$. Hence $2^i(n+r-1) > \beta_t \geqslant \beta_{t-1} + 2^i \geqslant \cdots \geqslant \beta_1 + 2^i(t-1) \geqslant 2^i(t-1)$ and we can conclude that $t < n + r$. □

4.2 A consistent encoding for PB constraints

Let us now take an arbitrary PB constraint $C : a_1x_1 + \cdots a_nx_n \leq K$ and assume that a_M is the largest coefficient. If $m = \log a_M$, we can rewrite C splitting the coefficients into powers of two as shown in Example 6:

$$\begin{array}{rlllll}\tilde{C}: & 2^0 \cdot \delta_{0,1} \cdot x_{0,1} & + \; 2^0 \cdot \delta_{0,2} \cdot x_{0,2} & + \cdots + & 2^0 \cdot \delta_{0,n} \cdot x_{0,n} & + \\ & 2^1 \cdot \delta_{1,1} \cdot x_{1,1} & + \; 2^1 \cdot \delta_{1,2} \cdot x_{1,2} & + \cdots + & 2^1 \cdot \delta_{1,n} \cdot x_{1,n} & + \\ & & & \cdots & & + \\ & 2^m \cdot \delta_{m,1} \cdot x_{m,1} & + \; 2^m \cdot \delta_{m,2} \cdot x_{m,2} & + \cdots + & 2^m \cdot \delta_{m,n} \cdot x_{m,n} & \leq K,\end{array}$$

where $\delta_{m,r}\ \delta_{m-1,r}\ \cdots\ \delta_{0,r}$ is the binary representation of a_r. Notice that C and $\tilde{C}$ represent the same constraint if we add clauses expressing that $x_{i,r} = x_i$ for appropriate i and r.

The important remark is that, using a consistent SAT encoding of the BDD for $\tilde{C}$ (e.g. the one given in [ES06] or the one presented in the next section) and adding clauses expressing that $x_{i,r} = x_i$ for appropriate i and r, we obtain a consistent encoding for the original constraint C using $\mathcal{O}(n^2 \log a_M)$ auxiliary variables and clauses.

This is not difficult to see. Take an assignment A over the variables of C which cannot be extended to a model of C. This is because the coefficients corresponding to the variables true in A add more than K. Using the clauses for $x_{i,r} = x_i$, unit propagation will produce an assignment to the $x_{i,r}$'s that cannot be extended to a model of $\tilde{C}$. Since the encoding for $\tilde{C}$ is consistent, a false clause will be found. Conversely, if we consider an assignment A over the variables of C than can be extended to a model of C, this assignment can clearly be extended to a model for $\tilde{C}$ and the clauses expressing $x_{i,r} = x_i$. Hence, unit propagation on those clauses and the encoding of $\tilde{C}$ will not detect a false clause.

4.3 An arc-consistent encoding for PB constraints

Unfortunately, the previous approach does not produce an arc-consistency encoding. The intuitive idea can be seen in the following example:

Example 7. Let us consider the constraint $3x_1 + 4x_2 \leq 6$. After splitting the coefficients into powers of two, we obtain $C' : x_{0,1} + 2x_{1,1} + 4x_{2,2} \leq 6$. If we set $x_{2,2}$ to true, C' implies that either $x_{0,1}$ or $x_{1,1}$ have to be false, but the encoding cannot exploit the fact that both variables will receive the same truth value and hence both should be propagated. Adding clauses stating that $x_{0,1} = x_{1,1}$ does not help in this sense. □

In order to overcome this limitation, we follow the method presented in [BKNW09,BBR09]. Let $C : a_1x_1 + \cdots a_nx_n \leq K$ be an arbitrary PB constraint. We denote as C_i the constraint $a_1x_1 + \cdots + a_i \cdot 1 + \cdots + a_nx_n \leq K$, i.e., the constraint assuming x_i to be true. For every i with $1 \leq i \leq n$, we encode C_i as in Section 4.2 and, in addition, we add the binary clause $r_i \vee \neg x_i$, where r_i is the root of the BDD for C_i. This clause helps us to preserve arc-consistency: given an assignment A such that $A \cup \{x_i\}$ cannot be extended to a model of C, literal $\overline{r}_i$ will be propagated using A (because the encoding for C_i is consistent). Hence the added clause will allow us to propagate $\overline{x}_i$.

All in all, the suggested encoding is arc-consistent and uses $\mathcal{O}(n^3 \log(a_M))$ clauses and auxiliary variables, where a_M is the largest coefficient.

5 SAT Encodings of BDDs for Monotonic Functions

In this section we consider a BDD representing a monotonic function F and we want to encode it into SAT. As expected, we want the encoding to be as small as possible and arc-consistent.

As usual, the encoding introduces an auxiliary variable for every node. Let ν be a node with selector variable x and auxiliary variable n. Let f be the variable of its false child and t be the auxiliary variable of its true child. Only two clauses per node are needed:

$$\neg f \rightarrow \neg n \qquad \neg t \wedge x \rightarrow \neg n.$$

Furthermore, we add a unit clause with the variable of the True node and another one with the negation of the variable of the False node.

Theorem 9 *The encoding is consistent in the following sense: a partial assignment A cannot be extended to a model of F if and only if $\neg r$ is propagated by unit propagation, where r is the root of the BDD.*

Proof. We prove the theorem by induction on the number of variables of the BDD. If the BDD has no variables, then the BDD is either the True node or the False node and the result is trivial.

Assume that the result is true for BDDs with less than k variables, and let F be a function whose BDD has k variables. Let r be the root node, x_1 its selector variable and f, t respectively its false and true children (note that we abuse the notation and identify nodes with their auxiliary variable). We denote by F_1 the function $F_{|x_1=1}$ (i.e., F after setting x_1 to true) and by F_0 the function $F_{|x_1=0}$.

– Let A be a partial assignment that cannot be extended to a model of F.
  - Assume $x_1 \in A$. Since A cannot be extended, the assignment $A \setminus \{x_1\}$ cannot be extended to a model of F_1. By definition of the BDD, the function F_1 has t as a BDD. By induction hypothesis, $\neg t$ is propagated, and since $x_1 \in A$, $\neg r$ is also propagated.
  - Assume $x_1 \notin A$. Then, the assignment $A \setminus \{\neg x_1\}$ cannot be extended to a model of F_0. Since F_0 has f as a BDD, by induction hypothesis $\neg f$ is propagated, and hence $\neg r$ also is.

– Let A be a partial assignment, and assume $\neg r$ has been propagated. Then, either $\neg f$ has also been propagated or $\neg t$ has been propagated and $x_1 \in A$ (note that x_1 has not been propagated because it only appears in one clause which is already true).
  - Assume that $\neg f$ has been propagated. Since f is the BDD of F_0, by induction hypothesis the assignment $A \setminus \{x_1, \neg x_1\}$ cannot be extended to a model of F_0. Since the function is monotonic, $A \setminus \{x_1, \neg x_1\}$ neither can be extended to a model of F. Therefore, A cannot be extended to a model of F.
  - Assume that $\neg t$ has been propagated and $x_1 \in A$. Since t is the BDD of F_1, by induction hypothesis $A \setminus \{x_1\}$ cannot be extended to a model of F_1, so neither can A be extended to a model of F.

□

For obtaining an arc-consistent encoding, we only have to add a unit clause.

Theorem 10 *If we add a unit clause forcing the variable of the root node to be true, the previous encoding becomes arc-consistent.*

Proof. We will prove it induction on the variables of the BDD. The case $n = 0$ is trivial, so let us prove the induction case.

As before, let r be the root node, x_1 its selector variable and f, t respectively its false and true children. We denote by F_1 the function $F_{|x_1=1}$ and by F_0 the function $F_{|x_1=0}$.

Let A be a partial assignment than can be extended to a model of F. Assume that $A \cup \{x_i\}$ cannot be extended. We want to prove that $\overline{x}_i$ will be propagated.

- Let us assume that $x_1 \in A$. In this case, t is propagated due to the clause $\neg t \wedge x_1 \rightarrow \neg n$ and the unit clause n. Since $x_1 \in A$ and $A \cup \{x_i\}$ cannot be extended to a model of F, $A \setminus \{x_1\} \cup \{x_i\}$ neither can be extended to an assignment satisfying F_1. By induction hypothesis, since t is the BDD of the function F_1, $\neg x_i$ is propagated.
- Let us assume that $x_1 \notin A$ and $x_i \neq x_1$. Since F is monotonic, $A \cup \{x_i\}$ cannot be extended to a model of F if and only if it cannot be extended to a model of F_0. Notice that f is propagated thanks to the clause $\neg f \rightarrow \overline{n}$ and the unit clause n. By induction hypothesis, the method is arc-consistent for F_0, so $\overline{x}_i$ is propagated.
- Finally, assume that $x_1 \notin A$ and $x_i = x_1$. Since $A \cup \{x_1\}$ cannot be extended to a model of F, A cannot be extended to model of F_1. By Theorem 9, $\neg t$ is propagated and, due to $\neg t \wedge x_1 \rightarrow \neg n$ and n, also is $\neg x_1$. □

6 Conclusions and Future Work

Both theoretical and practical contributions have been made. Regarding the theoretical part, we have proved that, unless NP=co-NP, there are PB constraints that do not admit polynomial BDDs. The existence of a concrete PB constraint family for which no polynomial BDDs exist remains an open problem, with interesting connections to the area of proof complexity. One of our aims is to continue working on this open question in the near future.

At the practical level, we have introduced a BDD-based polynomial and arc-consistent encoding of PB constraints and we have developed a BDD-based arc-consistent encoding of monotonic functions that only uses two clauses per BDD node. Indeed our initial motivation for this work has been practical, and we are currently working on implementation and experimental comparison of our encodings with other existing approaches on realistic problems.

References

[BBR06] O. Bailleux, Y. Boufkhad, and O. Roussel. A Translation of Pseudo Boolean Constraints to SAT. *JSAT*, 2(1-4):191–200, 2006.

[BBR09] O. Bailleux, Y. Boufkhad, and O. Roussel. New Encodings of Pseudo-Boolean Constraints into CNF. In *SAT'09"*, LNCS 5584, pp. 181–194.

[BHvMW09] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, IOS Press, February 2009.

[BKNW09] C. Bessiere, G. Katsirelos, N. Narodytska, and T. Walsh. Circuit Complexity and Decompositions of Global Constraints. In *IJCAI'09*, pp. 412–418, 2009.

[Bry86] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[ES06] N . Eén and N. Sörensson. Translating Pseudo-Boolean Constraints into SAT. *JSAT* 2:1–26, 2006.

[J.S07] J. Smaus. On Boolean Functions Encodable as a Single Linear Pseudo-Boolean Constraint. In *CPAIOR'07*, LNCS 4510, pp. 288–302.

[SFSW09] A. Schutt, T. Feydy, P. J. Stuckey, and M. Wallace. Why Cumulative Decomposition Is Not as Bad as It Sounds. In *CP'09*, LNCS 5732, pp. 746-761.