Artificial Intelligence Review (2020) 53:5157–5188
https://doi.org/10.1007/s10462-020-09817-6

An MDD-based SAT encoding for pseudo-Boolean constraints with at-most-one relations

Miquel Bofill[1] · **Jordi Coll**[1] · **Josep Suy**[1] · **Mateu Villaret**[1]

Published online: 18 February 2020

Abstract
Pseudo-Boolean (PB) constraints are ubiquitous in Constraint Satisfaction Problems. Encoding such constraints to SAT has proved to be an efficient solving approach. A commonly used technique for encoding PB constraints consists in representing the constraint as a Binary Decision Diagram (BDD), and then encoding this BDD to SAT. A key point in this technique is to obtain small BDD representations to generate small SAT formulas. In many problems, some subsets of the Boolean variables occurring in a PB constraint may also have other constraints imposed on them. In this work we introduce a way to take advantage of those constraints in order to obtain smaller SAT encodings. The main idea is that decision diagrams may be smaller if they avoid to represent truth assignments that are already forbidden by some other constraints. In particular, we present encodings for monotonic decreasing PB constraints, in conjunction with other constraints such as at-most-one, exactly-one and implication chains on subsets of their variables. We provide empirical evidence of the usefulness of this technique to reduce the size of the encodings as well as the solving time.

Keywords Pseudo-Boolean · Decision diagram · At-most-one · Exactly-one · Implication chain · SAT · Encodings

1 Introduction

Pseudo-Boolean (PB) constraints are linear expressions of the form $\sum_{i=1}^{n} q_i x_i \# K$, where $\# \in \{<, \leq, =, \geq, >\}$, $q_1, \ldots, q_n$ and K are integer constants, and $x_1, \ldots, x_n$ are 0/1 variables. A PB constraint represents a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$. A well-known

✉ Jordi Coll
jordi.coll@imae.udg.edu

Miquel Bofill
miquel.bofill@imae.udg.edu

Josep Suy
josep.suy@imae.udg.edu

Mateu Villaret
mateu.villaret@imae.udg.edu

[1] Universitat de Girona, Girona, Spain

approach to solve PB constraints is to represent them as Binary Decision Diagrams (BDD), which are then encoded into SAT (Eén and Sorensson 2006; Abío et al. 2012).

PB constraints appear frequently in formulations of Constraint Satisfaction Problems (CSP). Sometimes, there are also other constraints imposed on the Boolean variables of a PB constraint. A very frequent case is that of the at-most-one (AMO) constraint, which states that at most one of the Boolean variables in a set can be assigned true. For example, in routing problems (Miller et al. 1960; Dantzig and Ramser 1959; Laporte 1992), the length of paths can be represented using PB constraints, where each variable encodes if the path is joining two particular points. Since it is usually required that Hamiltonian paths are followed, only one variable among the ones which represent going from a particular point to any other can be set to true. Also, in combinatorial auctions the objective function is usually a PB constraint (De Vries and Vohra 2003; Bofill et al. 2013), where each Boolean variable represents whether a certain bid has been selected. The PB constraint contains all the possible bids, but many bids can contain a same product, and therefore at most one among them can be selected. In the general context of scheduling, PB constraints are a natural way to express constraints over the use of shared resources (Pritsker et al. 1996; Bofill et al. 2016, 2017). In these constraints, Boolean variables usually state the execution of a certain activity at a certain time point. Many notions of incompatibility between activities arise: there can be precedences between activities, or it may have to be decided a single running configuration for each activity. It is also usual to find exactly-one (EO) constraints in this kind of problems, in which case assigning all variables to false is neither allowed.

It is well known that the search for small SAT encodings is a worthwhile technique to achieve better solving times (Bacchus and Winter 2003; Eén and Biere 2005). In this work, we show how to reduce the size of the decision diagram representations of PB constraints, taking into account the AMO constraints that are also present (either explicitly or implicitly) in the problem. The overall idea is to remove from the decision diagrams the paths whose corresponding variable assignments are already forbidden by the AMO constraints. This way, the decision diagrams do not represent such inconsistent assignments, i.e., they only cover the subset of the assignments that are consistent with the AMO constraints, and hence they can be much smaller.

In summary, we propose to obtain small SAT encodings of PB constraints by taking into account that some assignments are forbidden due to other constraints. We introduce the general notion of PB($\mathscr{C}$) constraint, which is defined as the conjunction of a PB constraint with a set of constraints $\mathscr{C}$. In particular, we focus on PB(AMO) constraints, defined as the conjunction of a PB constraint and a set of AMO constraints. We use a compact Multi-valued Decision Diagram (MDD) representation for the PB constraints, taking into account the AMO constraints imposed. Finally, by encoding such compact diagrams to SAT, together with the rest of constraints, we are able to obtain very small SAT encodings of PB(AMO) constraints. As we show in the experimental section, the small size of the encodings obtained with our treatment has a dramatic impact on the solving time of some well-known problems.

The main contributions and organization of the paper are:

- In Sect. 2 we review some basic notions and notation.
- In Sect. 3 we introduce PB($\mathscr{C}$) constraints and the particular case of PB(AMO) constraints.
- In Sect. 4 we present AMO-MDDs, a Multi-valued Decision Diagram representation for PB constraints under the assumption of AMO constraints, and provide an algorithm to construct reduced ordered AMO-MDDs.

- In Sect. 5 we provide an AMO-MDD based SAT encoding of monotonic decreasing PB(AMO) constraints. We prove its correctness and that it UP-maintains GAC.
- In Sect. 6 we present some reformulation techniques that allow us to reduce other PB($\mathscr{C}$) constraints to monotonic decreasing PB(AMO) constraints, namely (i) PB(EO) constraints; (ii) not monotonic decreasing PB(AMO) constraints; (iii) PB constraints with implication chains.
- In Sect. 7 we experimentally evaluate our contributions.
- In Sect. 8 we discuss related work.
- In Sect. 9 we conclude and discuss further work.

This work is an extended version of Bofill et al. (2017), where it is shown how PB and AMO constraints can be used in an SMT based approach for solving two well-known scheduling problems, namely the MRCPSP and the RCPSP/t. In these problems, there exist PB constraints on resources with several AMO constraints among their variables. The obtained encodings are so compact that this approach turns to be the state-of-the-art among exact approaches.

In this paper we provide the general framework of PB($\mathscr{C}$) constraints to model conjunctions of PB constraints and other constraints over the variables of the PB constraint. We also present additional reformulations, correctness proofs and a detailed experimental section with experiments on two additional problem classes: Multiple-choice Multidimensional Knapsack Problem (MMKP), and the Combinatorial Auction problem (CA).

2 Preliminaries

In this section we review some notions on SAT formulas, pseudo-Boolean constraints, at-most-one constraints, and decision diagrams, and introduce the notation that we use in the rest of the paper.

A *Boolean variable* is a variable than can take truth values 0 (false) and 1 (true). A *literal* is a Boolean variable x or its negation $\overline{x}$. A *clause* is a disjunction of literals. A *propositional formula in conjunctive normal form* (CNF) is a conjunction of clauses. Clauses are usually seen as sets of literals, and formulas as sets of clauses. A *Boolean function* is a function of the form $f : \{0,1\}^n \to \{0,1\}$. In this paper we will only consider constraints which are defined on a finite set of Boolean variables, i.e., Boolean functions. The *scope* of a constraint is the set of variables which appear in the constraint. An *assignment* is a mapping of Boolean variables to truth values; it can also be seen as a set of literals (e.g., $\{x = 1, y = 0, z = 0\}$ is usually denoted $\{x, \overline{y}, \overline{z}\}$). A *satisfying assignment* of a Boolean function f is an assignment that makes it evaluate to 1. In particular, an assignment A satisfies a formula F in CNF if at least one literal of each clause in F belongs to A. Such an assignment is called a *model* of the formula. W.l.o.g., we will assume that all propositional formulas are in CNF. SAT is the problem of determining if there exists a satisfying assignment for a given Boolean formula. Given two Boolean functions (or constraints) f and g, we say that g is a logical consequence of f, written $f \vDash g$, iff every model of f is also a model of g. We say that two Boolean functions f and g are logically equivalent, denoted $f \equiv g$, if $f \vDash g$ and $g \vDash f$.

We say that a formula $E(C)$ is an *encoding* of a constraint C if the following holds: given an assignment A over the variables of C, A satisfies C iff A can be extended to a satisfying assignment of $E(C)$.

Unit propagation (UP) is the core deduction mechanism in modern SAT solvers: whenever each literal of a clause but one is false, the remaining literal must be set to true in order to satisfy the clause. An encoding $E(C)$ of a constraint C is said to maintain Generalized Arc Consistency by Unit Propagation (*UP-maintain GAC*) if it satisfies the following property: given a partial assignment A, if a variable x of C is true (respectively false) in every extension of A satisfying C, then unit propagating A on $E(C)$ will extend A to $A \cup \{x\}$ (respectively $A \cup \{\overline{x}\}$) (Bailleux et al. 2009).

Example 1 Given the constraint *at most one of* x_1, x_2, x_3 *is true*, where x_1, x_2, x_3 are Boolean variables, the following set of clauses is a UP-maintaining GAC encoding of the constraint:

$$(\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3})$$

Given an assignment with one variable assigned to true, w.l.o.g. x_1, the other two variables must be assigned to false to satisfy the constraint. Unit propagation would assign x_2 and x_3 to false due to the first and the third clauses.

Definition 1 An *at-most-one* (AMO) constraint is a Boolean function of the form $\sum_{i=1}^{n} x_i \leq 1$, where all x_i are 0/1 variables.

Definition 2 An *at-least-one* (ALO) constraint is a Boolean function of the form $\sum_{i=1}^{n} x_i \geq 1$, where all x_i are 0/1 variables.

Definition 3 An *exactly-one* (EO) constraint is a Boolean function of the form $\sum_{i=1}^{n} x_i = 1$, where all x_i are 0/1 variables. It can be defined the conjunction of an AMO constraint and an ALO constraint.

Definition 4 A *pseudo-Boolean* (PB) constraint is a Boolean function of the form $\sum_{i=1}^{n} q_i x_i \# K$ where K and all q_i are integer constants, all x_i are 0/1 variables, and $\# \in \{<, \leq, =, \geq, >\}$.

As usual in the literature and w.l.o.g. we will assume that # is $\leq$, and that the q_i and K are positive, since the other cases can be easily reduced to this one (Eén and Sorensson 2006). Such a constraint ($\leq$ with positive coefficients) is monotonic decreasing in the sense that any model remains a model after flipping inputs from 1 to 0 (Abío et al. 2012).

Definition 5 A *Binary Decision Diagram* (BDD) is a rooted, directed, acyclic graph which represents a Boolean function. BDDs have two terminal nodes, namely $\bot$-terminal and $\top$-terminal. Each nonterminal node has associated a Boolean variable (*selector*), and has two outgoing edges, representing the *true* and the *false* assignment of the selector. Every truth assignment of the variables follows a path from the root to the $\top$-terminal when it satisfies the formula, or to the $\bot$-terminal otherwise.

A BDD is called *ordered* if different variables appear in the same order on all paths from the root. A BDD is said to be *reduced* if it satisfies the following two conditions:

- It contains no isomorphic sub-BDDs.
- There is no node whose true and false child are the same.

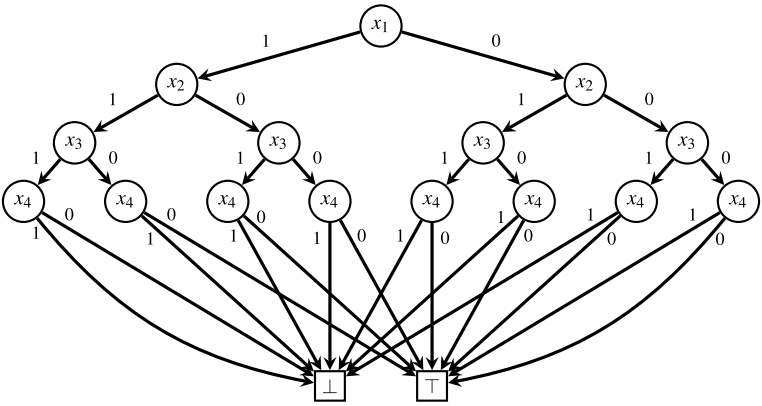
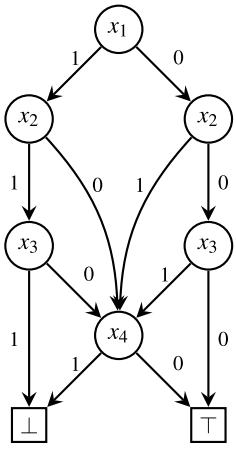


Fig. 1 Non-reduced Ordered BDD for the PB constraint $2x_1 + 3x_2 + 4x_3 + 6x_4 \leq 7$ with variable ordering $x_1 \prec x_2 \prec x_3 \prec x_4$

Fig. 2 ROBDD for the PB constraint $2x_1 + 3x_2 + 4x_3 + 6x_4 \leq 7$ with variable ordering $x_1 \prec x_2 \prec x_3 \prec x_4$



A *Reduced Ordered Binary Decision Diagram* (ROBDD) is canonical (unique) for a particular function and given variable order.

(RO)BDDs can in particular represent PB constraints (Eén and Sorensson 2006). Figure 1 contains a non-reduced ordered BDD representation of the PB constraint $2x_1 + 3x_2 + 4x_3 + 6x_4 \leq 7$ with the variable ordering $x_1 \prec x_2 \prec x_3 \prec x_4$. Figure 2 shows the ROBDD representation of the same PB constraint and variable order. As seen in the pictures, an ordered BDD can be organized in different *layers*, where at each layer it is considered a different selector variable. For instance, in all the nodes of the second layer we choose whether to set x_2 to 1 or to 0.

The ROBDD for a given PB constraint and variable order can be constructed in polynomial time w.r.t. the size of the final ROBDD (Abío et al. 2012).

2.1 Paper notation

In order to ease the readability of the paper, we provide an overview of the nomenclature conventions that we use to identify instances of the elements occurring the most throughout this paper:

- We denote Boolean functions with letters f and g.
- We denote constraints (over Boolean variables) with letter C, and with other letters in particular cases: P for PB constraints, AM for AMO constraints, PA for PB(AMO) constraints, PC for PB($\mathscr{C}$) constraints, EO for EO constraints and IC for implication chains.
- We denote Boolean formulas with letter F, or with $E(C)$ when the formulas are encodings of a constraint C.
- We denote variable assignments with letters A and B.
- We denote Boolean variables with letter x, and sets of Boolean variables with letter X.
- We denote coefficients in a PB constraint with letter q, and the right-hand side of a PB constraint with letter K.
- We denote AMO-MDDs with letter M.

Most of times the previous terms are sub-scripted to identify different occurrences of a same kind of element.

3 Conjunctions of pseudo-Boolean constraints with other constraints

Given a constraint PC of the form $P \wedge C_1 \wedge \cdots \wedge C_m$, where P is a pseudo-Boolean constraint and $C_1, \ldots, C_m$ are any other constraints, a straightforward approach to encode it is to generate a formula $E(P) \wedge E(C_1 \wedge \cdots \wedge C_m)$, where $E(P)$ is an encoding of P, and $E(C_1 \wedge \cdots \wedge C_m)$ is an encoding of $C_1 \wedge \cdots \wedge C_m$. We propose to relax the encoding of P by only considering assignments that satisfy $C_1 \wedge \cdots \wedge C_m$, since the remaining assignments falsify PC.

Let us develop this idea with a motivating example.

Example 2 Consider a constraint $PC : P \wedge C_1 \wedge C_2$ over Boolean variables x_1, x_2, x_3, which has the following truth table, with assignments labelled $A_0, \ldots, A_7$:

	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	P	C_1	C_2	$P \wedge C_1 \wedge C_2$
A_0	0	0	0	0	1	0	0
A_1	0	0	1	1	1	1	1
A_2	0	1	0	1	1	0	0
A_3	0	1	1	0	0	1	0
A_4	1	0	0	1	1	1	1
A_5	1	0	1	0	0	0	0
A_6	1	1	0	0	1	1	0
A_7	1	1	1	1	0	1	0

The straightforward approach to encode the constraint is to generate two formulas $E(P)$, $E(C_1 \wedge C_2)$ which are encodings of P and $C_1 \wedge C_2$ respectively, so that $E(P) \wedge E(C_1 \wedge C_2)$ is an encoding of PC. Note that the assignments A_0, A_2, A_3, A_5 and A_7 are not extendable to a model of $E(C_1 \wedge C_2)$, regardless of the satisfiability of $E(P)$ under these assignments. Therefore, $E(P)$ could be replaced by a formula F such that A_1 and A_4 are extendable to a model of F, and A_6 is not. For such formula F, independently of its evaluation on assignments A_0, A_2, A_3, A_5 and A_7, we have that $F \wedge E(C_1 \wedge C_2)$ is an encoding of PC. The potential of this idea is that F can be substantially smaller than $E(P)$, and this can have a positive impact in the solving time.

Definition 6 Let P be a PB constraint and $\mathscr{C} = \{C_1, \ldots, C_m\}$ be a set of constraints over the variables of P. We will refer to the formula $P \wedge C_1 \wedge \cdots \wedge C_m$ as a $PB(\mathscr{C})$ constraint, and call it a *PB modulo* $\mathscr{C}$ constraint.

Following the idea of Example 2, we propose to encode PB($\mathscr{C}$) constraints in a combined way. On the one hand we will encode the conjunction of constraints in $\mathscr{C}$ in the usual way, i.e., by encoding each of them separately and using the conjunction of all the resulting clauses. On the other hand, we will translate the PB constraint P into a set of clauses that may not be equisatisfiable to P. However, these clauses are equisatisfiable to P for assignments that satisfy the accompanying constraints $\mathscr{C}$. This way, the PB($\mathscr{C}$) constraint is correctly encoded to SAT, while the set of clauses for P can be significantly smaller than an encoding of the PB constraint alone.

The following lemma trivially follows from the definition of encoding (see Sect. 2):

Lemma 1 *Let $P \wedge C_1 \wedge \cdots \wedge C_m$ be a $PB(\mathscr{C})$ constraint, and $E(C_1 \wedge \cdots \wedge C_m)$ an encoding of $C_1 \wedge \cdots \wedge C_m$. Let F be a formula such that any assignment A satisfying $C_1 \wedge \cdots \wedge C_m$ can be extended to a model of F iff $A \models P$. Then, $F \wedge E(C_1 \wedge \cdots \wedge C_m)$ is an encoding of $P \wedge C_1 \wedge \cdots \wedge C_m$.*

We can go a bit further in the idea of encoding a relaxation of a pseudo-Boolean constraint. In the context of a bigger formula, some constraints $\mathscr{C}$ can be logically implied. We can take into account those implied constraints $\mathscr{C}$ to relax the encoding of a PB constraint. Moreover, there is no need to encode the implied constraints.

Lemma 2 *Let P be a PB constraint, C any constraint and $E(C)$ an encoding of C. Let $\mathscr{C} = \{C_1, \ldots, C_m\}$ be a set of constraints such that $C \models C_1 \wedge \cdots \wedge C_m$. Let F be a formula such that any assignment A satisfying $C_1 \wedge \cdots \wedge C_m$ can be extended to a model of F iff $A \models P$. Then, $F \wedge E(C)$ is an encoding of $P \wedge C$.*

In this work we focus in the encoding PB(AMO) constraints, a particular case of PB($\mathscr{C}$) constraints defined as the conjunction of a PB constraint and a set of AMO constraints.

Definition 7 By *PB(AMO) constraint* we refer to a constraint of the form $P \wedge AM_1 \wedge \cdots \wedge AM_m$, where P is a PB constraint, and $AM_1, \ldots, AM_m$ are AMO constraints such that$\{scope(AM_1), \ldots, scope(AM_m)\}$ is a partition of $scope(P)$.

Note that from a set of AMO constraints with non-disjoint scopes, we can always extract an implied set of AMO constraints with disjoint scopes, and therefore compose a

PB(AMO) constraint. Note also that a variable x can always be included in a single-variable AMO constraint of the form $x \leq 1$. Therefore, PB constraints are a particular case of PB(AMO) constraints.

4 MDD based representation of PB constraints with AMO relations

In this section we show how to represent PB constraints under the assumption of AMO constraints using *Multi-valued Decision Diagrams* (MDDs). In their classical definition, MDDs can be seen as a generalization of BDDs which have a multi-valued selector variable in each node instead of a Boolean variable (Srinivasan et al. 1990), and each possible value corresponds to a different decision. However, and especially in the context of SAT encodings of MDDs, a set of Boolean variables can be used as selectors, each variable representing a different decision. We introduce the following variant of MDD.

Definition 8 An *At-Most-One Multi-Decision Diagram* (AMO-MDD) is a generalization of a BDD. It is a rooted, directed, acyclic multigraph which has two terminal nodes, namely $\bot$-terminal and $\top$-terminal. Each nonterminal node has associated a set of Boolean selector variables $x_1, \ldots, x_s$, and has an outgoing edge for each variable. Moreover, there is an additional outgoing edge, which we denote as the *else* edge. Each one of the $s+1$ outgoing edges corresponds to a different decision, namely assigning exactly one of the x_i to *true*, for $i \in 1..s$, or assigning all of them to *false*, hence choosing the *else* edge. The definitions of ordered and reduced can be also applied to AMO-MDDs, giving place to AMO-ROMDDs.

Given a PB(AMO) constraint of the form $P \wedge AM_1 \wedge \cdots \wedge AM_m$, such that $X_i = scope(AM_i)$ for $i \in 1..m$ and $\{X_1, \ldots, X_m\}$ is a partition of $scope(P)$, we will represent its PB constraint P by an AMO-MDD where X_i will be the set of selector variables of the nodes in layer i. This way, all paths from the root to a terminal node will choose (assign to 1) at most one of the variables in the scope of each AM_i and, therefore, the AMO-MDD will cover all the assignments that satisfy $AM_1 \wedge \cdots \wedge AM_m$. In this representation, every one of those truth assignments will follow a path from the root to the $\top$-terminal if it satisfies P, or to the $\bot$-terminal otherwise. Note that assignments which do not satisfy $AM_1 \wedge \cdots \wedge AM_m$ (i.e., assigning more than one variable to 1 in the scope of some AM_i) are not covered by this AMO-MDD.

As said, unlike ROBDDs representing PB constraints where the i-th layer deals with a single variable x_i, the i-th layer of an AMO-ROMDD deals with a set of variables X_i. Hence, the order of the AMO-ROMDD is defined on sets of selector variables, i.e., the AMO-ROMDD representation is subject to an ordered partition of the variables of the PB constraint. Figure 3 shows the AMO-ROMDD representation of $P : 2x_1 + 3x_2 + 4x_3 + 6x_4 \leq 7$ with the ordered partition $\{x_1, x_2\} \prec \{x_3, x_4\}$. Notice the significant reduction in size with respect to the ROBDD representation of P in Fig. 2. In particular, the AMO-ROMDD has only 2 nonterminal nodes and 6 edges instead of the 6 nonterminal nodes and 12 edges of the ROBDD, and the number of layers is reduced from 5 to 3.

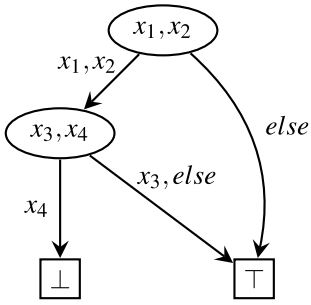


Fig. 3 AMO-ROMDD for $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$, with ordered partition $\{x_1, x_2\} \prec \{x_3, x_4\}$. Multiple edges between two nodes are represented as a single edge with multiple labels

4.1 AMO-ROMDD construction

Abío et al. (2012) introduces an algorithm to construct a ROBDD representing a PB constraint with a given variable ordering, whose running time is polynomial w.r.t. the size of the resulting ROBDD. Here we present a generalization of that algorithm to construct an AMO-ROMDD for a PB constraint with a given ordered variable partition $X_1 \prec \cdots \prec X_m$ of its variables. Our algorithm is very similar to the one presented in Abío and Stuckey (2014) to construct MDDs representing Linear Integer Arithmetic expressions. The main differences are that here we provide the full algorithm to construct reduced diagrams, and that we do not deal with integer variables but a with vector of coefficients of pseudo-Boolean variables. Notice that our algorithm supports gaps in the values of the coefficients as well as repetitions.

For the sake of self-containment, this section carefully describes the proposed algorithm. First of all we introduce the idea of interval for a PB constraint with an associated partition.

Definition 9 Let $P : \sum_{i=1}^{n} q_i x_i \leq K$ be a PB constraint and let $\mathscr{X} = \{X_1, \ldots, X_m\}$ be a partition of its variables. The *interval* of P with the partition $\mathscr{X}$ are all the integers K' such that $\sum_{i=1}^{n} q_i x_i \leq K'$, interpreted as a Boolean function, has the same evaluation than P for any assignment A such that $A \vDash \sum_{x_j \in X_i} x_j \leq 1$, for all $X_i \in \mathscr{X}$. The set of such K' is always an interval that we denote by $[\beta, \gamma]$.

The idea of interval for a PB constraint is already defined in Abío et al. (2012). The difference is that in our case, where we are given a partition together with a PB constraint, intervals only consider the subset of assignments that fulfill the AMO constraints defined by the given partition.

Example 3 The interval of the PB constraint $x_1 + 5x_2 + 4x_3 + 4x_4 \leq 6$ with the partition $\{\{x_1, x_2, x_3\}, \{x_4\}\}$ of its variables is [5, 7], because its truth table, limited to the assignments satisfying $x_1 + x_2 + x_3 \leq 1$ and $x_4 \leq 1$, is the same for $K' \in [5, 7]$, and different for $K' = 4$ and $K' = 8$.

The AMO-ROMDD representation for a given PB constraint and ordered partition of its variables is the same for any K' in its interval. Note also that every node of an AMO-ROMDD is the root of an AMO-ROMDD, and hence it also has its corresponding interval. In particular, every node at layer i is the root of an AMO-ROMDD representing the

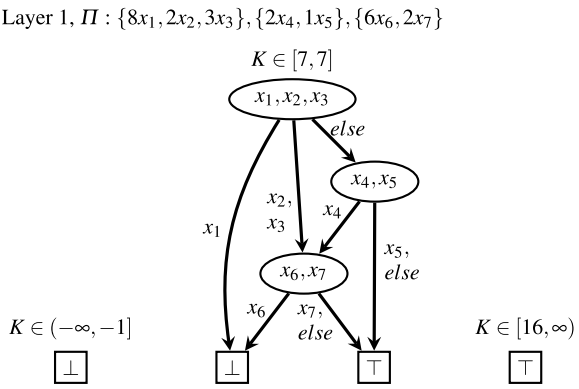
PB constraint $\sum_{j=i}^{m} \sum_{x_k \in X_j} q_k x_k \leq K'$ with ordered partition $X_i \prec \cdots \prec X_m$, with a different K' at each node. The algorithm presented below maintains a set L_i of tuples of the form $([\beta, \gamma], M)$ for each layer i of the AMO-ROMDD, where M is an AMO-ROMDD and $[\beta, \gamma]$ is its corresponding interval. By means of dynamic programming, the AMO-ROMDD representation of a particular PB and partition is constructed only once, and it is inserted in L_i together with its interval. This way, given a partition and a PB constraint whose K belongs to an interval already stored in the corresponding list L_i, the stored AMO-ROMDD representation can be reused and therefore its re-computation is avoided. On the contrary, if there is no already computed AMO-ROMDD for that PB constraint and partition, a new node must be created. The interval of this new node will be computed from the intervals of the layer immediately below.

The main procedure is described in Algorithm 1. In order to simplify the notation of recursive calls, the algorithm receives as input a PB constraint $P : \sum_{i=1}^{n} q_i x_i \leq K$ and an ordered partition $X_1 \prec \cdots \prec X_m$ of its variables, represented by a pair (Π, K), where $\Pi = p_1, \ldots, p_m$ is a list of sets of monomials, such that every set p_i contains the monomials $q_j x_j$ for each variable $x_j \in X_i$, $i \in 1..m$. The algorithm starts by inserting the $\top$-terminal and the $\bot$-terminal AMO-ROMDDs to every layer, with the corresponding interval in that layer (line 2). Namely, the interval ot the $\bot$-terminal node is $(-\infty, -1]$ at all layers, and the interval of the $\top$-terminal node for layer j is $[\sum_{j \in i..m} \max_{q_k x_k \in p_j}(q_k), \infty)$. Then, it calls the recursive procedure *MDDBuild* (Algorithm 2).

MDDBuild searches and eventually inserts AMO-ROMDDs in the lists L_i while recursively constructing the global AMO-ROMDD. More precisely, each call checks whether the AMO-ROMDD representing the given PB constraint and partition is already contained in L_i (lines 1–3), and if it is not, it is obtained and inserted into L_i (lines 4–20). To obtain this AMO-ROMDD, the child for each possible decision is recursively obtained, taking into account that choosing x_j contributes with q_j to the left-hand-side sum the PB constraint (line 7), and that choosing *else* has a contribution of 0 (line 9). If all the children are the same AMO-ROMDD, i.e., all the children have the same interval, no new node is created but this unique child is already the AMO-ROMDD that must be returned (line 12). Otherwise, the root node for the AMO-ROMDD is created, and its children are properly linked (line 15). In both cases, the interval of the resulting AMO-ROMDD at layer i is computed as the intersection of the intervals of the children, shifted accordingly. Namely, the interval is computed as shown in line 16, or in a simpler way when all the children are the same, as in line 13. Finally, the obtained AMO-ROMDD and its interval are inserted in L_i and returned. Figure 4 shows an example of which is the content of each L_i after the construction of an AMO-ROMDD.

Algorithm 2 uses the following functions:

`search`(K, L_i):	If there is a tuple (I, M) in L_i, such that $K \in I$, it is returned. Otherwise, an empty interval is returned in the first component of the tuple.
`insert`$((I, M), L_i)$:	Inserts (I, M) into the set L_i.
`indexOfMax`$(q_1, \ldots, q_s)$:	Returns the index of the maximum coefficient in the list $q_1, \ldots, q_s$.
`mdd`$(\langle x_1, \ldots, x_s \rangle, \langle M_1, \ldots, M_s \rangle, M_{else})$:	Constructs an AMO-ROMDD with a new node as root, M_j as child for each selector variable x_j, and M_{else} as the *else* child.



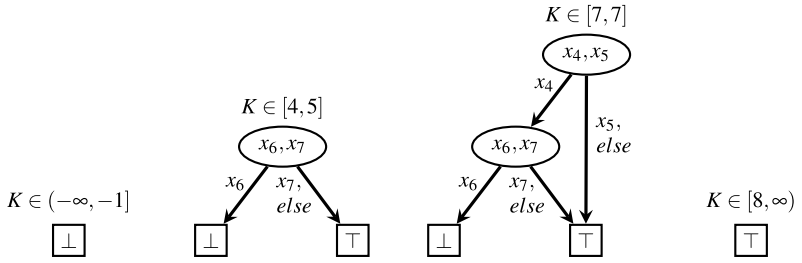
Minimal Encoding:

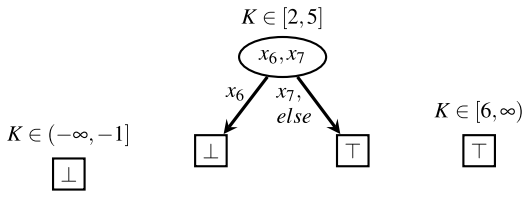
$v_\perp \vee \overline{x_1} \vee \overline{v_r}$ $v_b \vee \overline{x_3} \vee \overline{v_r}$
$v_b \vee \overline{x_2} \vee \overline{v_r}$ $v_a \vee \overline{v_r}$

$v_b \vee \overline{x_4} \vee \overline{v_a}$ $v_\top \vee \overline{v_a}$

$v_\perp \vee \overline{x_6} \vee \overline{v_b}$ $v_\top \vee \overline{v_b}$

v_r $v_\top$ $\overline{v_\perp}$





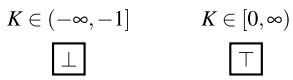


Fig. 4 Content of *LL* at the end of the construction of the AMO-ROMDD for $8x_1 + 2x_2 + 3x_3 + 2x_4 + x_5 + 6x_6 + 2x_7 \leq 7$ with the ordered partition $\{x_1, x_2, x_3\} \prec \{x_4, x_5\} \prec \{x_6, x_7\}$. Multiple edges between two nodes are represented as a single edge with multiple labels. At top right, resulting set of clauses of the Minimal Encoding of this AMO-ROMDD

Algorithm 1 Construction of AMO-ROMDD algorithm

Require: list $\Pi : p_1, \ldots, p_m$, integer K
Ensure: returns M the AMO-ROMDD of (Π, K)
1: **for all** $i \in 1..m+1$ **do**
2: $\quad L_i \leftarrow \left\{ ((-\infty,-1],\bot),([\sum_{j \in 1..m} (\max_{q_k x_k \in p_j} q_k),\infty),\top) \right\}$
3: **end for**
4: $LL \leftarrow \langle L_1, \ldots, L_{m+1} \rangle$
5: $([\beta,\gamma],M) \leftarrow$ **MDDBuild**$(1,\Pi,K,LL)$
6: **return** M

Algorithm 2 Procedure MDDBuild

Require: integer $i \in [1, m+1]$, list $\Pi : p_i, \ldots, p_m$, integer K, list LL
Ensure: returns $[\beta,\gamma]$ interval of (Π, K), and M its AMO-ROMDD
1: $([\beta,\gamma],M) \leftarrow$ **search**(K, L_i)
2: **if** $[\beta,\gamma] \neq \emptyset$ **then**
3: $\quad$ **return** $([\beta,\gamma],M)$
4: **else**
5: $\quad$ **let** $p_i = q_1 x_1, \ldots, q_s x_s$
6: $\quad$ **for all** $j \in 1..s$ **do**
7: $\quad\quad$ $([\beta_j,\gamma_j],M_j) \leftarrow$ **MDDBuild**$(i+1, \langle p_{i+1}, \ldots, p_m \rangle, K - q_j, LL)$
8: $\quad$ **end for**
9: $\quad$ $([\beta_{else},\gamma_{else}],M_{else}) \leftarrow$ **MDDBuild**$(i+1, \langle p_{i+1}, \ldots, p_m \rangle, K, LL)$
10: $\quad$ $\alpha \leftarrow$ **indexOfMax**$(q_1, \ldots, q_s)$
11: $\quad$ **if** $[\beta_\alpha,\gamma_\alpha] = [\beta_{else},\gamma_{else}]$ **then**
12: $\quad\quad$ $M \leftarrow M_\alpha$
13: $\quad\quad$ $[\beta,\gamma] \leftarrow [\beta_\alpha + q_\alpha, \gamma_\alpha]$
14: $\quad$ **else**
15: $\quad\quad$ $M \leftarrow$ **mdd**$(\langle x_1, \ldots, x_s \rangle, \langle M_1, \ldots, M_s \rangle, M_{else})$
16: $\quad\quad$ $[\beta,\gamma] \leftarrow [\beta_{else},\gamma_{else}] \cap \bigcap_{j \in 1..s} [\beta_j + q_j, \gamma_j + q_j]$
17: $\quad$ **end if**
18: $\quad$ **insert**$(([\beta,\gamma],M), L_i)$
19: $\quad$ **return** $([\beta,\gamma],M)$
20: **end if**

The algorithm in Abío et al. (2012) runs in polynomial time with respect to the size of the generated ROBDD. We argue that Algorithm 1, which is a generalization for the case of AMO-ROMDDs, preserves the polynomial running time. All the searches and insertions in LL in Algorithm 2 can be done in logarithmic time. Algorithm 2 is called once for the root node of the AMO-ROMDD, and $\mathcal{O}(h \cdot S)$ times for each edge of the AMO-ROMDD, being h the length of the edge, and $S = \max_{p_i \in \Pi} |p_i|$.

5 An AMO-MDD based SAT encoding of monotonic decreasing PB(AMO) constraints

In this section we present an adaptation of the encoding for monotonic decreasing functions from Abío et al. (2012), which is the smallest known BDD-based encoding, and is the one which has shown the best performance in preliminary experiments. Our adaptation deals

with PB(AMO) constraints $PA : P \wedge AM_1 \wedge \cdots \wedge AM_m$, where P is a monotonic decreasing PB constraint, i.e., P is of the form $\sum_{i=1}^{n} q_i x_i \leq K$, with positive coefficients. We propose a method for obtaining a set of clauses F for P such that in conjunction with an encoding of $AM_1 \wedge \cdots \wedge AM_m$ gives us an encoding of PA.

Though the generated set of clauses F is not exactly an encoding of P, we refer to it as the *Minimal Encoding*, due to its resemblance to the MDD encoding for Linear Integer expressions presented in Abío et al. (2016), also named Minimal. Corollary 1 states which is the constraint encoded by the Minimal Encoding.

Minimal Encoding The input of the Minimal Encoding is a monotonic decreasing PB constraint P, and a partition of its variables $\{X_1, \ldots, X_m\}$. This input is obtained from a PB(AMO) of the form $P \wedge AM_1 \wedge \cdots \wedge AM_m$, where $X_i = scope(AM_i)$ for $i \in 1..m$. First of all, P and $\{X_1, \ldots, X_m\}$ are represented as a (RO)AMO-MDD M, and then M is encoded as a set of clauses, that we refer to as the Minimal Encoding. We want to remark that this resulting set of clauses is not encoding the AMO constraints $AM_1, \ldots, AM_m$ which, as explained in Sect. 3, are assumed to be encoded or implied separately by other clauses. The actual semantics of the set of clauses generated by the Minimal Encoding is that when more than one variable of some X_i is assigned to true, only the one with maximum coefficient in P is taken into account to evaluate P. This is stated in Corollary 1. Corollary 2 states that the Minimal Encoding in conjunction with a separate encoding of $AM_1 \wedge \cdots \wedge AM_m$ is an encoding of the whole PB(AMO) constraint.

For each node of the AMO-MDD M at hand, the encoding adds a fresh auxiliary Boolean variable v. The encoding will enforce v to be false if the given assignment over the variables of the PB constraint follow a path from the current node to the $\bot$-terminal node. If the root of M is the $\top$-terminal node (i.e., it represents a tautology) the encoding only adds the clause v_r, being v_r the auxiliary variable of the root node. On the other hand, if the root of M is the $\bot$-terminal node (i.e., it represents a contradiction) the encoding adds the clauses v_r and $\overline{v_r}$. Finally, if the root of M is not a terminal node, the clauses v_r, $v_\top$ and $\overline{v_\bot}$ are included in the formula, where $v_r, v_\top, v_\bot$, are the auxiliary variables of the root, the $\top$-terminal and the $\bot$-terminal nodes respectively. Also, the following clauses are added for each non-terminal node with set of selector variables X_i, that is representing the PB $\sum_{k=i}^{m} \sum_{x_l \in X_k} q_l x_l \leq K$ with partition $\{X_i, \ldots, X_m\}$:

$$v_j \vee \overline{x_j} \vee \overline{v} \quad \forall x_j \in X_i \text{ s.t. } v_j \neq v_0 \tag{1}$$

$$v_0 \vee \overline{v} \tag{2}$$

Here v is the auxiliary variable of the node, v_j is the auxiliary variable of the child node pointed by the selector variable x_j, and v_0 is the auxiliary variable of the *else* child. Intuitively, clause (1) for a selector x_j states that if the child with variable v_j is false (i.e. $\sum_{k=i+1}^{m} \sum_{x_l \in X_k} q_l x_l > K - q_j$) and x_j is true, then the current node is also false (i.e. $q_j + \sum_{k=i+1}^{m} \sum_{x_l \in X_k} q_l x_l > K$). Similarly, clause (2) states that in any case, if the else child is false (i.e. $\sum_{k=i+1}^{m} \sum_{x_l \in X_k} q_l x_l > K$), then the current node is also false (i.e. $\sum_{k=i}^{m} \sum_{x_l \in X_k} q_l x_l > K$). Notice that a node can have more than one selector variable pointing to the same child. In particular, if the edge of a selector variable x_j points to the *else* child, there is no need to add clause (1) for that selector variable, because $v_0 \vee \overline{v}$ implies $v_0 \vee \overline{x_j} \vee \overline{v}$. Figure 4 contains the resulting Minimal Encoding of an example AMO-ROMDD.

The following theorems and corollaries are devoted to explain the semantics of the Minimal Encoding (Theorem 1 and Corollary 1), to show how the Minimal

Encoding can be used to encode PB(AMO) constraints (Corollary 2), and to prove that the obtained PB(AMO) encoding UP-maintains GAC (Theorem 2).

Theorem 1 *Let $P : q_1x_1 + \cdots + q_nx_n \leq K$ be a monotonic decreasing PB constraint and $\mathscr{X} = \{X_1, \ldots, X_m\}$ be a partition of its variables. Let F be the Minimal Encoding of an AMO-MDD representation for P and $\mathscr{X}$, and v_r the auxiliary variable of its root node. Let A be any total assignment over the variables $\{x_1, \ldots, x_n\}$, and let C denote the constraint $\sum_{i=1}^{m} \max_{x_j \in X_i} (q_j x_j) \leq K$. Then the following holds*:

- *If A is a model of C, then A can be extended to a model of F.*
- *If A is not a model of C, then there exists an extension of A that satisfies $F \setminus \{v_r\}$, and any such extension assigns v_r false.*

Proof We proceed by induction on the number of summands of C. Note that C has one summand for each set X_i.

Base case: In the base case, C is of the form $0 \leq K$, and so is P.

- Assume that we have an assignment A over the variables of P (in fact, an empty assignment) satisfying C, i.e., $K \geq 0$. In this case, the AMO-MDD representation for P and $\mathscr{X}$ is simply the $\top$-terminal node. Hence, its Minimal Encoding F contains only the unit clause v_r, and $A \cup \{v_r\}$ is trivially a model of F.
- Assume the contrary, i.e., that we have an assignment A over the variables of P not satisfying C, i.e., A is empty and $K < 0$. In this case, the AMO-MDD representation of P and $\mathscr{X}$ is simply the $\bot$-terminal node. Hence, its Minimal Encoding F consists of the two unit clauses $v_r, \overline{v_r}$. In this case, $A \cup \{\overline{v_r}\}$ is the only extension of A satisfying $F \setminus \{v_r\}$.

Inductive step: In this case C is of the form $\sum_{i=1}^{m} \max_{x_j \in X_i} (q_j \cdot x_j) \leq K$, with $m \geq 1$, and P can be written as $\sum_{i=1}^{m} \sum_{x_j \in X_i} q_j x_j \leq K$.
Let M be an AMO-MDD representation for P and $\mathscr{X}$, and F its Minimal Encoding. Let X_1 be of the form $\{x_1, \ldots, x_s\}$ and assume, w.l.o.g., that $q_i \leq q_{i+1}$ for all $i \in 1..s-1$.
In order to ease the proof, we define $q_0 = 0$, a neutral coefficient for the *else* case, and introduce the Boolean constant $x_0 = true$, which we assume belongs to any assignment, and allows us to replace the clause $v_0 \vee \overline{v_r}$ in F by $v_0 \vee \overline{x_0} \vee \overline{v_r}$.
Since all coefficients in P are positive, we know that $q_0 \leq q_1$. For all $k \in \{0, \ldots, s\}$, we define the following terms:

- P_k is $\sum_{i=2}^{m} \sum_{x_j \in X_i} q_j x_j \leq K - q_k$, i.e., the constraint resulting of assigning $x_k = true$ in P.
- C_k is $\sum_{i=2}^{m} \max_{x_j \in X_i} (q_j x_j) \leq K - q_k$, i.e., the constraint resulting of assigning $x_k = true$ in C.
- F_k is the Minimal Encoding for the subgraph of M which corresponds to P_k and the partition $\{X_2, \ldots, X_m\}$.
- $F'_k = F_k \setminus \{v_k\}$, which fulfills $F'_k \subset F$.

If the root node of M does not have $\langle x_1, \dots, x_s \rangle$ as selector variables, this means that M is also the AMO-MDD representation of all $P_0, \dots, P_s$, and hence $F = F_0 = \cdots = F_s$. In this case the theorem trivially holds by induction hypothesis, taking $P = P_0$, $F = F_0$ and $C = C_0$.

From now on, we assume that the root node of M has $\langle x_1, \dots, x_s \rangle$ as selector variables. By the definition of Minimal Encoding, we have $F = F'_0 \cup \cdots \cup F'_s \cup \{v_0 \vee \overline{x_0} \vee \overline{v_r}, \dots, v_s \vee \overline{x_s} \vee \overline{v_r}, v_r\}$.

- Assume that we have an assignment A over the variables of P satisfying C. Then, there is an index $max \in 0..s$ such that $max = 0$ or $x_{max} \in A$, $x_j \notin A \ \forall_{j \in max+1..s}$, and A satisfies $C_0, \dots, C_{max}$. We can construct an assignment $B \supset A$ satisfying F as follows: $B = A \cup B_0 \cup \cdots \cup B_s \cup \{v_r\}$, where each $B_k \supset A|_{vars(C_k)}$ is an assignment satisfying F'_k as we show below. Note that several B_k can share auxiliary node variables, because the corresponding F'_k may share such variables (i.e., the child AMO-MDDs of the root of M may not be disjoint). However, in this proof we show a way to deterministically construct the assignment B, and the same procedure can be applied in the construction of all B_k. This means that B can be consistently constructed, in the sense that a same auxiliary node variable does not have two different values in two different B_k. First of all, we have that v_r must be in B to satisfy the clause v_r. By definition of max, A is a model of $C_0, \dots, C_{max}$, and therefore by induction hypothesis there exist assignments $B_0, \dots, B_{max}$ satisfying $F'_0 \cup \{v_0\}, \dots, F'_{max} \cup \{v_{max}\}$, respectively. Then, B also satisfies the formulas $F'_0, \dots, F'_{max}$ and the clauses $v_0 \vee \overline{x_0} \vee \overline{v_r}, \dots, v_{max} \vee \overline{x_{max}} \vee \overline{v_r}$. We also know by definition of x_{max} that $\overline{x_j} \in A \subset B$ for all $j \in max + 1..s$, and therefore B satisfies the clauses $v_{max+1} \vee \overline{x_{max+1}} \vee \overline{v_r}, \dots, v_s \vee \overline{x_s} \vee \overline{v_r}$. Finally, we have to consider the remaining formulas $F'_{max+1}, \dots, F'_s$. For i in $max + 1..s$ we distinguish the following cases:

  $A \vDash C_k$: By induction hypothesis, let B_k be an assignment satisfying $F'_k \cup \{v_k\}$.
  $A \nvDash C_k$: By induction hypothesis, let B_k be an assignment satisfying F'_k such that $\overline{v_k} \in B_k$.
  Assignments $B_{max+1}, \dots, B_s$ satisfy $F'_{max+1}, \dots, F'_s$ respectively, and therefore also does B.

- Assume that we have an assignment A over the variables of P not satisfying C. Then there is an index $max \in 0..s$ such that $max = 0$ or $x_{max} \in A$, $x_j \notin A \ \forall_{j \in max+1..s}$, and A does not satisfy $C_{max}, \dots, C_s$.

We will show that there exists an assignment $B \supset A$ that satisfies $F \setminus \{v_r\}$, and any such assignment must assign v_r to *false*. This assignment is $B = A \cup B_0 \cup \cdots \cup B_s \cup \{\overline{v_r}\}$, where each $B_k \supset A|_{vars(C_k)}$ is an assignment satisfying F'_k as we show below. Again, the coherence of the shared variables in different B_k is guaranteed.

Since A is not a model of $C_{max}, \dots, C_s$, by induction hypothesis let $B_{max}, \dots, B_n$ be assignments respectively satisfying $F'_{max}, \dots, F'_s$, and therefore respectively assigning $v_{max}, \dots, v_s$ to false. We have that $x_{max} \in B$ because either $x_{max} \in A$, or $max = 0$ and $x_0 = true$ by assumption, and we also have that $\overline{v_{max}} \in B$ since $\overline{v_{max}} \in B_{max}$. Therefore, v_r must be false in B in order to satisfy the clause $v_{max} \vee \overline{x_{max}} \vee \overline{v_r}$. In consequence all clauses $v_0 \vee \overline{x_0} \vee \overline{v_r}, \dots, v_s \vee \overline{x_s} \vee \overline{v_r}$ are also satisfied. Finally, we have two possible cases for each one of the remaining formulas $F'_0, \dots, F'_{max-1}$:

$A \vDash C_k$: By induction hypothesis, let B_k be an assignment satisfying $F'_k \cup \{v_k\}$.
$A \nvDash C_k$: By induction hypothesis, let B_k be an assignment satisfying F'_k such that $\overline{v_k} \in B_k$.

Such assignments $B_0, \ldots, B_{max-1}$ satisfy $F'_0, \ldots, F'_{max-1}$ respectively and therefore also does B. □

Corollary 1 *Let F be a Minimal Encoding for a PB constraint $\sum_{i=1}^{n} q_i x_i \leq K$ with positive coefficients and partition $\{X_1, \ldots, X_m\}$ of its variables. Then F is also an encoding of $\sum_{i=1}^{m} \max_{x_j \in X_i}(q_j x_j) \leq K$.*

The following corollary states how to use the Minimal Encoding to encode a PB(AMO) constraint.

Corollary 2 *Let PA be a PB(AMO) constraint of the form $P \wedge AM_1 \wedge \cdots \wedge AM_m$ with positive coefficients in P. Let F be a Minimal Encoding for P and partition $\{X_1, \ldots, X_m\}$, where $X_i = scope(AM_i)$. Let $E(AM_1 \wedge \cdots \wedge AM_m)$ be an encoding of $AM_1 \wedge \cdots \wedge AM_m$. Then, $F \wedge E(AM_1 \wedge \cdots \wedge AM_m)$ is an encoding of PA.*

Proof The corollary follows from Lemma 1 and Corollary 1 of Theorem 1. □

The following theorem states that we can generate an UP-maintaining GAC encoding of a PB(AMO) constraint using the Minimal Encoding.

Theorem 2 *Let PA be a PB(AMO) constraint of the form $P \wedge AM_1 \wedge \cdots \wedge AM_m$ with positive coefficients in P. Let F be a Minimal Encoding for P and partition $\{X_1, \ldots, X_m\}$, where $X_i = scope(AM_i)$. Let $E(AM_1 \wedge \cdots \wedge AM_m)$ be a UP-maintaining GAC encoding of $AM_1 \wedge \cdots \wedge AM_m$. Then, $F \wedge E(AM_1 \wedge \cdots \wedge AM_m)$ is an UP-maintaining GAC encoding of PA.*

Proof Consider any partial assignment A over the variables of PA that can be extended to a model of PA. We need to show that for every variable x of PA such that x is not assigned in A, if $A \cup \{x\}$ cannot be extended to a satisfying assignment of PA, then x will be set to false by unit propagating A on $F \wedge E(AM_1 \wedge \cdots \wedge AM_m)$. Note that, due to the monotonicity of P and of $AM_1, \ldots, AM_m$, $A \cup \{\overline{x}\}$ can always be extended to a satisfying assignment, so we do not need to consider this case.

First note that, if $A \cup \{x\}$ could be extended to a model of $AM_1, \ldots, AM_m$, and could also be extended to a model of P then, due to monotonicity, $A \cup \{x\}$ could be extended to a model of PA by setting the remaining variables of PA to false. Therefore, since we assume that $A \cup \{x\}$ cannot be extended to a model of PA, we only need to analyse the following two reasons why $A \cup \{x\}$ cannot be extended:

- $A \cup \{x\}$ falsifies some AMO constraint. In such case, the assumption that $E(AM_1 \wedge \cdots \wedge AM_m)$ UP-maintains GAC will unit-propagate $\overline{x}$.
- $A \cup \{x\}$ can be extended to satisfy $AM_1 \wedge \cdots \wedge AM_m$ but $A \cup \{x\}$ falsifies P. In this case, the proof is a trivial generalization of Theorem 23 in Abío et al. (2012).

□

6 Other PB($\mathscr{C}$) constraints

In this section we present other instances of PB modulo $\mathscr{C}$ constraints, and propose how to encode them.

6.1 PB(EO) constraints

In many applications there appear conjunctions of PB constraints with exactly-one (EO) constraints over their variables. A particular case where this happens is when encoding Linear Integer Arithmetic constraints as pseudo-Boolean constraints with a *direct encoding* of the integer variables.

Definition 10 By *PB(EO) constraint* we refer to a constraint of the form $P \wedge EO_1 \wedge \cdots \wedge EO_m$, where P is a PB constraint, $EO_1, \ldots, EO_m$ are EO constraints, and $\{scope(EO_1), \ldots, scope(EO_m)\}$ is a partition of $scope(P)$.

Since an EO constraint implies an AMO constraint, by Lemma 2 we can use the presented AMO-MDD based encoding to encode a PB(EO), which by itself is a noticeable improvement with respect to a naive encoding of a PB(EO) constraint.

However, we can do better, by reducing the number of variables of the PB constraint. By subtracting a same integer to all the coefficients of a set of variables holding an EO constraint, as well as to the right-hand side of the inequality, we can make some coefficients become zero, and then remove those zero coefficient variables. For example, let P be the PB constraint $2x_1 + 3x_2 + 4x_3 + 3x_4 + 4x_5 + 5x_6 \leq 7$, and suppose we want to encode $P \wedge x_1 + x_2 + x_3 = 1 \wedge x_4 + x_5 + x_6 = 1$. Then we can replace P by $(2-2)x_1 + (3-2)x_2 + (4-2)x_3 + 3x_4 + 4x_5 + 5x_6 \leq 7 - 2$, because exactly one of x_1, x_2 and x_3 will be set to 1 in any assignment satisfying $x_1 + x_2 + x_3 = 1$, and therefore 2 will be subtracted exactly once in the left-hand side of the inequality. Similarly, we can subtract 3 to the coefficients of x_4, x_5 and x_6, obtaining $P' : x_2 + 2x_3 + x_5 + 2x_6 \leq 2$. We have that $P \wedge x_1 + x_2 + x_3 = 1 \wedge x_4 + x_5 + x_6 = 1$ is equivalent to $P' \wedge x_1 + x_2 + x_3 = 1 \wedge x_4 + x_5 + x_6 = 1$, with the advantage that P' has a smaller ROBDD representation. Moreover, by Lemma 2 we can still use the AMO-MDD based encoding of P', since $x_1 + x_2 + x_3 = 1 \models x_2 + x_3 \leq 1$ and $x_4 + x_5 + x_6 = 1 \models x_5 + x_6 \leq 1$.

6.2 PB(EO) and PB(AMO) constraints with negative coefficients

As stated before, the existing encodings of PB constraints to SAT are usually designed for constraints of the form $\sum_{i=i}^{n} q_i x_i \leq K$, with non-negative coefficients q_i, since other cases can be easily transformed to this one. The usual way of getting rid of negative coefficients (see Eén and Sorensson 2006) is by using the equality $x = 1 - \overline{x}$. For example, $-2x_1 + 6x_2 \leq 5 \equiv -2(1 - \overline{x_1}) + 6x_2 \leq 5 \equiv 2\overline{x_1} + 6x_2 \leq 7$. Then, if we want to encode a constraint of the form $\sum_{i=i}^{n} q_i x_i \geq K$ with positive coefficients, we can simply replace it by $-\sum_{i=i}^{n} q_i x_i \leq -K$ and get rid of the negative coefficients.

However, this rewrite method might not be applicable to PB(AMO) constraints. We will illustrate the situation with an example. Consider the PB(AMO) constraint $P \wedge x_1 + x_2 + x_3 \leq 1 \wedge x_4 + x_5 + x_6 \leq 1$, with $P : x_1 + 3x_2 + 4x_3 + 2x_4 + 3x_5 + 5x_6 \geq 6$. If

we transform P to the standard form, we obtain $P' : \overline{x_1} + 3\overline{x_2} + 4\overline{x_3} + 2\overline{x_4} + 3\overline{x_5} + 5\overline{x_6} \leq 12$. The problem is that now $x_1 + x_2 + x_3 \leq 1$ (similarly $x_4 + x_5 + x_6 \leq 1$) no longer impose AMO constraints over the literals of P', and therefore we cannot set x_1, x_2, x_3 as selector variables of a node of an AMO-MDD. We could still use a BDD-based encoding of P', but we would not be using the simplification potential of PB(AMO) constraints.

To overcome this weakness, we present another rewrite method to get rid of negative coefficients, which does not require to negate the literals of the original PB constraint, and hence still allows us to take into account the original AMO constraints. We begin with the case of PB(EO) constraints, which is the most straightforward, and for simplicity we assume that there is only one EO constraint (in case of more than one EO constraint, the procedure would be repeated for each one).

Given a constraint of the form $q_1x_1 + \cdots + q_sx_s + \cdots + q_nx_n \leq K \wedge x_1 + \cdots + x_s = 1$ with some negative coefficient in q_1,...,q_s, choose any integer Z such that $Z \geq -q_i$, for all $1 \leq i \leq s$. Then, the new constraint is $(Z + q_1)x_1 + \cdots + (Z + q_s)x_s + q_{s+1}x_{s+1} + \cdots + q_nx_n \leq K + Z$. Note that Z will be added exactly once to both sides of the inequality in any assignment satisfying $x_1 + \cdots + x_s = 1$. The best Z to pick is $-\min_{i=1}^{s}(q_i)$, since it will cancel at least one coefficient and therefore will reduce the number of variables of the constraint.

For the case of a PB(AMO) constraint of the form $q_1x_1 + \cdots + q_sx_s + \cdots + q_nx_n \leq K \wedge x_1 + \cdots + x_s \leq 1$ with some negative coefficient in q_1 ,...,q_s, we can first transform it into a PB(EO) constraint

$$0x_0 + q_1x_1 + \cdots + q_sx_s + \cdots + q_nx_n \leq K \wedge x_1 + \cdots + x_s \leq 1 \wedge (x_0 \leftrightarrow \overline{x_1} \wedge \cdots \wedge \overline{x_s})$$

where x_0 is a fresh variable, and it holds $x_1 + \cdots + x_s \leq 1 \wedge (x_0 \leftrightarrow \overline{x_1} \wedge \cdots \wedge \overline{x_s}) \models x_0 + x_1 + \cdots + x_s = 1$. Then we can apply the proposed PB(EO) transformation, rewriting over $0x_0 + q_1x_1 + \cdots + q_nx_n \leq K$ to get rid of the negative coefficients. Though this transformation introduces an auxiliary variable for each AMO and few reification clauses, the number of variables appearing in the resulting normalized PB constraint will not increase if we choose $Z = -\min_{i=1}^{s}(q_i)$.

Note that the procedure to rewrite PB(AMO) constraints is a generalization of the usual procedure to rewrite PB constraints, since in the case of single-variable AMO constraints of the form $x \leq 1$, instead of defining a fresh variable $x_0 \leftrightarrow \bar{x}$ we can directly use $\bar{x}$ for the transformation and we obtain the same result.

6.3 PB(IC) constraints

Abío et al. (2015) introduces an MDD-based encoding for PB constraints with implication chains over their variables. An implication chain (IC) is a constraint of the form $x_1 \leftarrow x_2 \wedge x_2 \leftarrow x_3 \wedge \cdots \wedge x_{s-1} \leftarrow x_s$, denoted $x_1 \Leftarrow x_2 \Leftarrow x_3 \Leftarrow \cdots \Leftarrow x_s$. A PB constraint with a set of implication chains can be seen as another kind of PB($\mathscr{C}$) constraint, that we will denote by PB(IC).

Definition 11 By *PB(IC) constraint* we refer to a constraint of the form $P \wedge IC_1 \wedge \cdots \wedge IC_m$, where P is a PB constraint, and $IC_1, \ldots, IC_m$ are implication chains. We assume that every variable in P occurs exactly in one IC_i (a variable x is by itself a single-variable chain).

As shown in Abío et al. (2015), by adding a set of channelling clauses, a PB(IC) encoding can be used to encode a PB(AMO). Essentially, the AMO constraints have to be

encoded with the *regular* encoding (also known as *ladder* encoding). The auxiliary variables introduced by this encoding are involved in an implication chain, and the PB constraint can be reformulated in terms of the new auxiliary variables.

Here we show how to do the translation the other way round, i.e., by adding a set of channelling clauses, a PB(AMO) encoding can be used to encode a PB(IC).

Let the $PB(IC)$ at hand be of the form $q_1x_1 + \cdots + q_nx_n \leq K \wedge IC_1 \wedge \cdots \wedge IC_m$. For each IC_i of the form $x^i_1 \Leftarrow \cdots \Leftarrow x^i_{s_i}$, we add a set of fresh variables $y^i_1, \ldots, y^i_{s_i}$, and the following clauses:

$$y^i_j \leftrightarrow x^i_j \wedge \overline{x^i_{j+1}} \qquad 1 \leq j < s_i \tag{3}$$

$$y^i_{s_i} \leftrightarrow x^i_{s_i} \tag{4}$$

Finally, we have to encode the PB(AMO) constraint $\sum_{i=1}^{m} \sum_{j=1}^{s_i} {q'}^i_j y^i_j \leq K$, where ${q'}^i_1 = q^i_1$, and ${q'}^i_j = q^i_j + {q'}^i_{j-1}$, for $1 < j \leq s_i$.

For instance, the PB(IC) constraint $2x_1 + 3x_2 + 4x_3 + 6x_4 + 3x_5 \leq 7 \wedge x_1 \Leftarrow x_2 \wedge x_3 \Leftarrow x_4 \Leftarrow x_5$ can be encoded as:

$$\begin{aligned} x_1 \Leftarrow x_2 \wedge x_3 \Leftarrow x_4 \Leftarrow x_5 \\ y_1 \leftrightarrow x_1 \wedge \overline{x_2} \\ y_2 \leftrightarrow x_2 \\ y_3 \leftrightarrow x_3 \wedge \overline{x_4} \\ y_4 \leftrightarrow x_4 \wedge \overline{x_5} \\ y_5 \leftrightarrow x_5 \\ F \end{aligned}$$

where F is the Minimal Encoding for $2y_1 + 5y_2 + 4y_3 + 10y_4 + 13y_5 \leq 7$ with partition $\{\{y_1, y_2\}, \{y_3, y_4, y_5\}\}$. Note that

$$(y_1 \leftrightarrow x_1 \wedge \overline{x_2}) \wedge (y_2 \leftrightarrow x_2) \wedge x_1 \Leftarrow x_2 \vDash y_1 + y_2 \leq 1$$

and

$$(y_3 \leftrightarrow x_3 \wedge \overline{x_4}) \wedge (y_4 \leftrightarrow x_4 \wedge \overline{x_5}) \wedge (y_5 \leftrightarrow x_5) \wedge x_3 \Leftarrow x_4 \Leftarrow x_5 \vDash y_3 + y_4 + y_5 \leq 1$$

and, hence, we can use the Minimal Encoding.

7 Experiments

In this section we analyse which is the gain in using the presented MDD-based encoding for PB(AMO) constraints. On one hand, we study the impact on the size of the decision diagrams, and hence on the size of the encodings both in number of variables and clauses. On the other hand, we study the impact on the solving time.

We deepen in the analysis started in Bofill et al. (2017) on the Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP) (Brucker et al. 1999) and the Resource-Constrained Project Scheduling Problem with Time-Dependent Resource Capacities and Requests (RCPSP/t) (Hartmann 2013), and consider two additional problem classes, namely the Multiple-choice Multidimensional Knapsack Problem

(MMKP) (Kellerer et al. 2004), and the Combinatorial Auction problem (CA) (Leyton-Brown and Shoham 2006). Classical models of these problems contain PB constraints in conjunction with AMO constraints. The four studied problems are optimization problems. In order to obtain results independent of any optimization process, we have encoded the decision version of these problems and analyzed the solving times of satisfiablity checks. These decision problems have been encoded into SAT/SMT formulas.

All the studied problems already contain either implicit or explicit AMO constraints in the formulations. Moreover, all problems contain a number of PB(AMO) and PB(EO) constraints, that have been encoded in different settings which we compare:

PB	All PB constraints are encoded to SAT using the Minimal Encoding of a ROBDD representation, i.e., not taking into account the AMO constraints to simplify the PB encoding. No reduction is done in PB(EO) constraints.
PB(AMO)	All PB constraints are encoded to SAT using the Minimal Encoding of an AMO-ROMDD representation. No reduction is done in PB(EO) constraints.
PB-redEO and *PB(EO)*	The same as *PB* and *PB(AMO)*, respectively, but applying the reduction of Sect. 6.1 to all PB(EO) constraints. Only used in MRCPSP and MMKP instances, that are the only ones with PB(EO) constraints.

The details of the formulations and the considered datasets for each problem follows. Table 1 summarizes the formulas involved in the experiments, showing the number of instances of each benchmark dataset, and the number of PB(AMO) and PB(EO) constraints that are encoded to SAT.

MRCPSP and RCPSP/t These two problems consist of determining a schedule for a set of activities of a project, i.e., a start time for each one of them, such that it minimizes the completion time (a.k.a. makespan). In both problems there are a number of resource constraints, stating that the capacities of the resources must not be exceeded by the demands of the scheduled activities at any time. We consider time-indexed formulations: for each resource and for each discretized time instant from 0 to an upper bound on the makespan, there is a PB constraint stating that the capacity of a particular resource must not be exceeded at that particular time instant. On the other hand there is a number of precedence constraints which state that, for each given predecessor-successor pair of activities, the successor activity cannot start until the predecessor has finished. These precedence relations are encoded as Integer Difference Logic (ILD) expressions. Both in MRCPSP and RCPSP/t, the precedence relations introduce incompatibilities between activities, and these incompatibilities imply AMO constraints over the variables of the PB resource constraints. Moreover, these problem have particular specifications that introduce further AMO constraints: in MRCPSP each activity has multiple execution modes, and one of them must be selected; in RCPSP/t, the resource requirements of the activities vary over their execution. We refer the reader to Bofill et al. (2017) for full formulations of the problems, and a detailed description of how to identify AMO constraints and formulate resource constraints as PB(AMO) constraints. We have generated a set of SMT formulas encoding the decision version of the MRCPSP and the RCPSP/t. In particular, we have considered benchmark instances of the MRCPSP and the RCPSP/t problems from the PSPLIB (Kolisch and Sprecher 1997) and the MMLIB (Van Peteghem and Vanhoucke 2014) libraries. We have only

Table 1 Generated sets of formulas

Problem	Dataset	Size	PB(AMO)	PB(EO)
MRCPSP	j30			
	sat	545	36,132	1090
	unsat	545	35,037	1090
	MMLIB50			
	sat	456	32,229	912
	unsat	456	31,314	912
	MMLIB100			
	sat	309	23,660	618
	unsat	309	23,042	618
	Total	2620	181,414	5240
RCPSP/t	j30			
	sat	2826	854,900	0
	unsat	2826	842,328	0
	j120			
	sat	2210	1,273,847	0
	unsat	2210	1,263,954	0
	Total	10,072	4,235,029	0
MMKP	Set1	500	0	5000
	Set2	500	0	5000
	Set3	400	0	20,000
	Total	1400	0	30,000
CA	CATS			
	sat	162	162	0
	unsat	162	162	0
	Total	324	324	0

The columns contain, in this order: the name of the encoded problem; the name of the instance dataset; *sat*/*unsat* for the problems where two decision versions per instance have been generated (MMKP instances are already decision instances with both satisfiable and unsatisfiable instances); the number of generated formulas for the dataset; the total number of PB(AMO) constraints and PB(EO) constraints in the dataset (considering all the formulas)

used instances for which the optimal makespan is known, hence being able to generate two tight decision cases for every instance, one satisfiable (with an upper bound equal to the optimal makespan) and one unsatisfiable (with an upper bound equal to the optimal makespan minus one). The average ratio of IDL/PB constraints in MRCPSP is 423/69, 1195/72 and 4115/78 for j30, MMLIB50 and MMLIB100 respectively, and in RCPSP/t the ratios are 257/310 and 2220/585 in j30 and j120 respectively. However, we remark that each IDL constraint corresponds to just one clause in the SMT formula, whereas a PB constraint is encoded with hundreds or thousands of clauses, as will be seen in Table 3.

MMKP In the Multiple-choice Multidimensional Knapsack Problem there is a number of classes of items, and exactly one item for each class must be chosen. For each item we know its profit, and the weigh that occupies for each dimension of a multidimensional knapsack. The knapsack has a particular capacity for each dimension. The problem consists in choosing one item of each class, such that the items fill in all the dimensions of the

knapsack and the maximum profit is obtained. We consider the three sets of instances generated in Bofill et al. (2019), which are decision instances where no constraint is imposed on the obtained profit. In these datasets there is a balanced number of satisfiable and unsatisfiable instances, as shown in Table 1. PB(EO) constraints appear when stating that the capacity of each dimension cannot be exceeded: Boolean variables model whether an item is chosen, and EO constraints state that exactly one item of each class must be chosen.

CA The Combinatorial Auction problem consist in deciding which sets of items, or packages, will be sold so that the maximum profit is obtained. For each package there is a bid, that is the profit obtained if that package is sold. However the items contained in one package can also be present in other packages, and therefore at most one of the packages containing a particular item can be sold. In the decision version of this problem a minimum profit is required, and this can be naturally modeled as a PB constraint, where Boolean variables state whether a package is sold. AMOs over these variables appear from packages sharing a same item. We consider the instances from Bofill et al. (2014) generated with the Combinatorial Auction Test Suit (CATS) instance generator (Leyton-Brown and Shoham 2006). Again we obtain a satisfiable and an unsatisfiable instance from optimization instance with known objective value, this time by using a lower bound equal to the optimum profit (satisfiable case) and the optimum plus one (unsatisfiable case). There are no other constraints present in the formulation apart from the PB objective function and the AMO constraints.

The formulas have been generated and solved using the C++ API of Yices 2.4.2 (Dutertre and de Moura 2006). All experiments have been run on a 8GB Intel® Xeon® E3-1220v2 machine at 3.10 GHz, with a timeout of 600 seconds in each execution.

For each studied problem, we use scatter plots to make pair-wise comparisons of the different PB(AMO) encoding settings. In the plots each point corresponds to the value of a same metric in the two compared settings, one in the x axis and the other in the y axis. More precisely, the plots in the left column of the figures compare solving time, where each point corresponds to a different instance, time is in seconds, and we use logarithmic axes. The plots in the middle and rigth columns compare the encoding size in number of variables and clauses respectively. In these size plots, each point corresponds to a different PB constraint of a different instance, and the values of the axes are in thousands. We also provide a summarized numeric comparison of time and sizes in Tables 2 and 3.

Figure 5 compares the *PB* and *PB(AMO)* settings for all the problems. In all problems there is a significant reduction in the solving time when using the *PB(AMO)* approach, which is up to one order of magnitude in the RCPSP/t and CA, and up to three orders of magnitude in the MRCPSP and MMKP. For instance, there are formulas which are solved in less than one second with *PB(AMO)*, whereas with *PB* they time out getting lost in the search tree, which contains an order of magnitude more variables. It can be seen that in all problems there is a significant reduction in size, which in a large number of constraints is of one order of magnitude. Due to the properties of the encoded problems, there are different clusters of points in the size plots. This is especially noticeable in the plots of the MRCPSP. Each cluster correspond to PB constraints modelling a particular kind of constraint in a family of instances. There is a subset of constraints in the MRCPSP which has a dramatic size reduction of more than one order of magnitude. This subset is in fact the set of PB(EO) constraints, which are the largest PB constraints. In particular, in each instance of the MRCPSP there are two PB(EO) constraints, and for this reason we do not observe any impact in the first quartile and median in Table 3 when applying the EO reduction. In MMKP we observe some patterns in the time and sizes plots, which are due to the

Table 2 Solving times for each problem

Setting	Q1	Med	Q3	Avg	To
MRCPSP					
PB	0.15	0.70	26.24	119.86	473
PB-redEO	0.10	0.25	0.97	22.35	44
PB(AMO)	0.06	0.20	0.79	17.71	37
PB(EO)	0.05	0.14	0.49	10.27	18
RCPSP/t					
PB	0.24	0.88	3.39	11.92	57
PB(AMO)	0.07	0.33	1.53	4.29	2
MMKP					
PB	9.81	389.29	600.00	319.91	685
PB-redEO	2.33	38.68	218.84	159.17	272
PB(AMO)	0.70	9.11	440.66	181.16	333
PB(EO)	0.40	4.96	46.71	99.79	164
CA					
PB	0.02	0.03	3.64	27.59	17
PB(AMO)	0.02	0.03	1.29	17.74	2

Columns contain first quartile (Q1), median (med) and third quartile (Q3) of the solving times, average solving time (avg) counting time-outs as 600 seconds, and number of time outs (to)

Table 3 Number of variables and clauses, in thousands, of the encodings of PB constraints

Setting	Variables					Clauses				
	Q1	Med	Q3	Max	Avg	Q1	Med	Q3	Max	Avg
MRCPSP										
PB	0.13	0.39	0.92	102.5	1.37	0.26	0.78	1.84	204.9	2.74
PB-redEO	0.13	0.39	0.90	23.4	0.90	0.26	0.78	1.80	46.8	1.79
PB(AMO)	0.02	0.07	0.15	13.1	0.21	0.06	0.29	0.79	52.2	0.89
PB(EO)	0.02	0.07	0.15	10.9	0.18	0.06	0.29	0.78	27.0	0.76
RCPSP/t										
PB	0.09	0.49	1.62	44.0	1.68	0.18	0.98	3.23	87.9	3.36
PB(AMO)	0.00	0.04	0.18	2.3	0.16	0.03	0.33	1.56	40.7	1.80
MMKP										
PB	3.18	3.46	41.39	1158.5	108.38	6.36	6.93	82.78	2316.9	216.76
PB-redEO	1.45	1.66	24.44	901.2	76.97	2.89	3.31	48.88	1802.4	153.95
PB(AMO)	0.44	0.47	2.17	39.9	4.88	2.51	2.70	22.05	428.1	49.91
PB(EO)	0.35	0.38	1.64	36.5	3.91	1.58	1.72	14.62	355.1	36.23
CA										
PB	0.03	0.46	192.50	834.1	127.05	0.22	1.07	385.08	1668.2	254.22
PB(AMO)	0.00	0.02	59.99	179.0	35.35	0.21	0.41	250.38	955.8	162.35

Columns contain first quartile (Q1), median (med), third quartile (Q3), maximum (max) and average (avg) of the sizes of all PB(AMO) and PB(EO) constraints of a problem

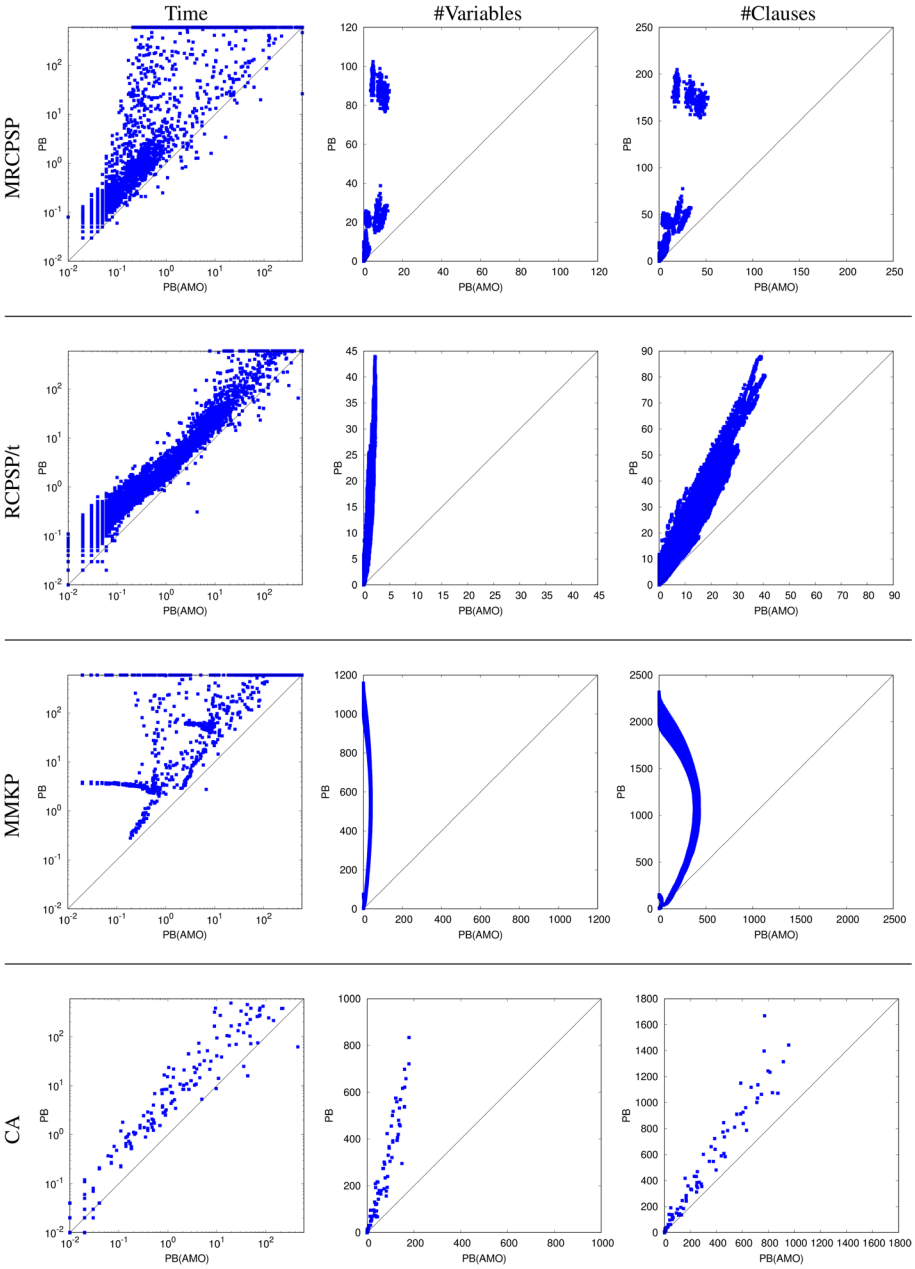


Fig. 5 Time in seconds (left), number of variables in thousands (middle) and number of clauses in thousands (right) comparison of the *PB(AMO)* and *PB* settings to solve, from top to bottom, MRCPSP, RCPSP/t, MMKP and CA

fact that the benchmark datasets were built modifying the value of some parameter over the instances of each dataset.

Figure 6 depicts the comparison between *PB(AMO)* and *PB(EO)* in MRCPSP and MMKP formulas, which are the only ones containing PB(EO) constraints. We observe a further reduction in size, with encodings that are between twice and ten times smaller in most of the cases, and a reduction of the solving time of one order of magnitude. Figure 7 shows the improvement achieved by combining the use of the AMO-MDD based encoding and the reductions by EO constraints (*PB(EO)*), with respect to the standard BDD-based encoding (*PB*). In fact, the EO reduction by itself gives a huge improvement, as can be seen in Fig. 8, which compares the *PB* and the *PB-redEO* settings.

When comparing the *PB* and the *PB(AMO)* approaches, the reduction in the number of variables is always strictly higher than the reduction in the number of clauses. This happens because the Minimal Encoding introduces one clause per edge and one variable per node, and the reduction in the number of nodes is higher than the reduction in the number of edges: while the nodes of BDDs have two outgoing edges, the nodes of AMO-MDDs can have a larger number of edges. Despite this fact, the results show that the number of clauses in the *PB(AMO)* approach is noticeably smaller than those in the *PB* approach.

The reduction in size is directly attributable to the fact that the AMO-MDDs only cover a subset of the possible truth assignments for the variables of the original PB constraints, while the BDDs cover all of them. It is especially noticeable that there are decision diagrams with size 1 in the *PB(AMO)* approach, and with size pf thousands in the *PB* approach. The decision diagrams of size 1 correspond to the ⊤-terminal node. This means that there are PB constraints which are trivially true under the assignments that satisfy an AMO constraint on the selector variables of the AMO-MDD nodes.

The depth of the AMO-MDDs is also significantly smaller than the depth of the BDDs. In the considered instances, in most of the cases we are able to set at least three selectors in each AMO-MDD node, which means that the depth of the AMO-MDDs is at most the

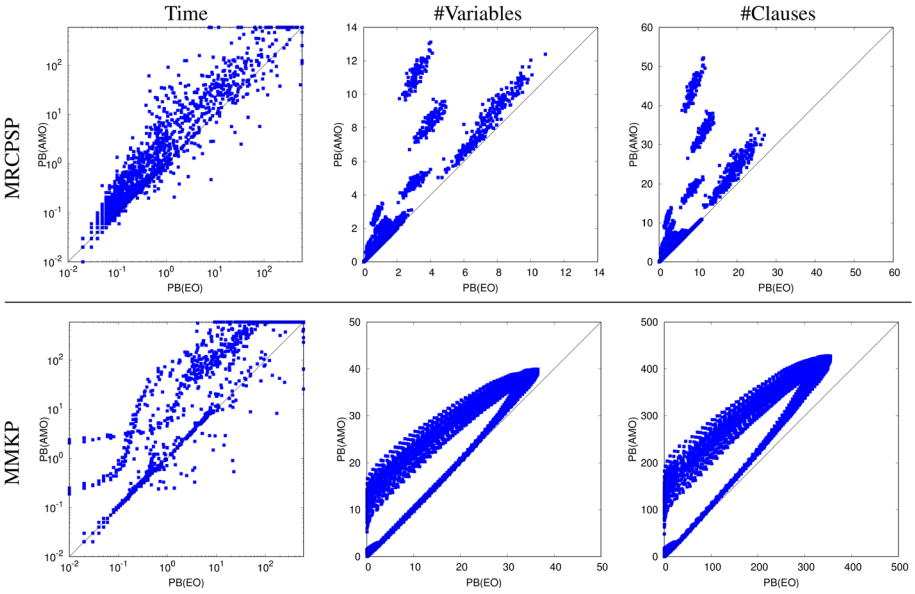


Fig. 6 Time in seconds (left), number of variables in thousands (middle) and number of clauses in thousands (right) comparison of the *PB(EO)* and *PB(AMO)* settings to solve, from top to bottom, MRCPSP, and MMKP

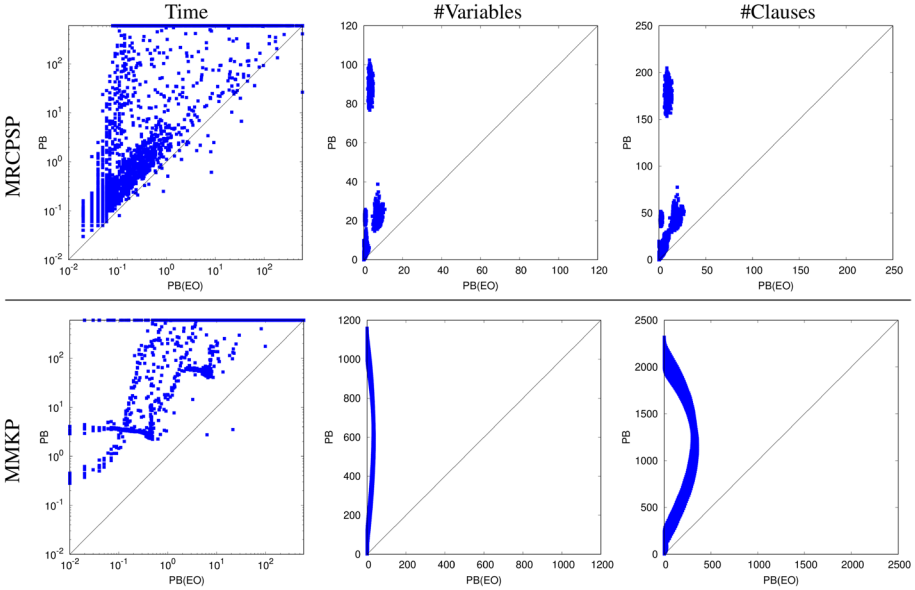


Fig. 7 Time in seconds (left), number of variables in thousands (middle) and number of clauses in thousands (right) comparison of the *PB(EO)* and *PB* settings to solve, from top to bottom, MRCPSP, and MMKP

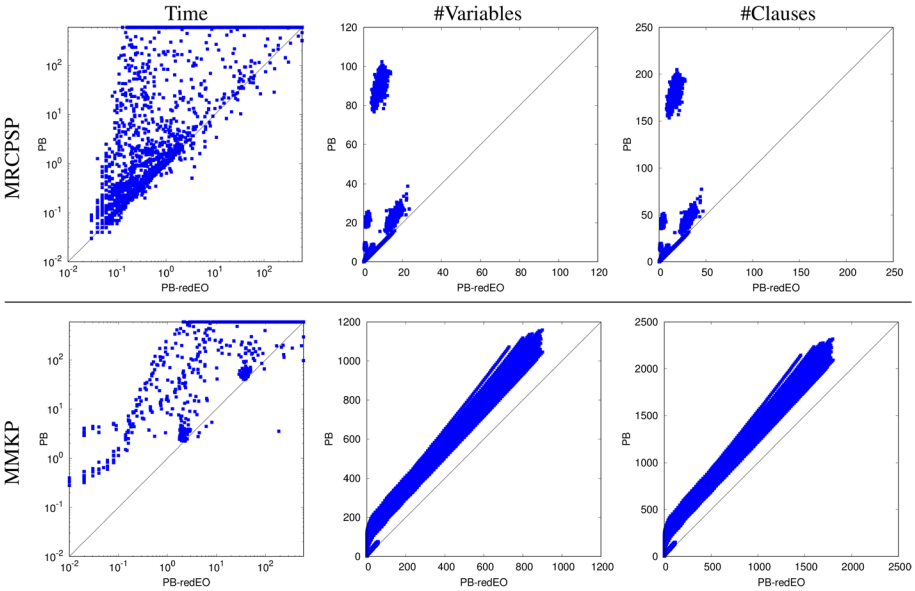


Fig. 8 Time in seconds (left), number of variables in thousands (middle) and number of clauses in thousands (right) comparison of the *PB-redEO* and *PB* settings to solve, from top to bottom, MRCPSP, and MMKP

Table 4 Solving times for each problem

Setting	Q1	Med	Q3	Avg	To
MRCPSP					
IC	0.05	0.14	0.52	9.83	17
PB(EO)	0.05	0.14	0.49	10.27	18
RCPSP/t					
IC	0.09	0.40	1.78	6.24	11
PB(AMO)	0.07	0.33	1.53	4.29	2
MMKP					
IC	0.41	5.49	79.75	120.90	212
PB(EO)	0.40	4.96	46.71	99.79	164
CA					
IC	0.02	0.03	1.57	14.48	0
PB(AMO)	0.02	0.03	1.29	17.74	2

Columns contain first quartile (Q1), median (med) and third quartile (Q3) of the solving times, average solving time (avg) counting time-outs as 600 seconds, and number of time outs (to)

Table 5 Number of variables and clauses, in thousands, of the PB constraint encodings

Setting	Variables					Clauses				
	Q1	Med	Q3	Max	Avg	Q1	Med	Q3	Max	Avg
MRCPSP										
IC	0.04	0.10	0.20	11.01	0.22	0.10	0.36	0.91	27.19	0.85
PB(EO)	0.02	0.07	0.15	10.87	0.18	0.06	0.29	0.78	26.96	0.76
RCPSP/t										
IC	0.01	0.10	0.31	2.93	0.25	0.08	0.46	1.91	42.51	2.06
PB(AMO)	0.00	0.04	0.18	2.34	0.16	0.03	0.33	1.56	40.74	1.80
MMKP										
IC	0.40	0.43	1.77	36.66	3.99	1.71	1.86	15.00	355.50	36.45
PB(EO)	0.35	0.38	1.64	36.53	3.91	1.58	1.72	14.62	355.12	36.23
CA										
IC	0.00	0.10	60.06	179.15	35.40	0.21	0.65	250.58	956.12	162.47
PB(AMO)	0.00	0.02	59.99	179.00	35.35	0.21	0.41	250.38	955.80	162.35

Columns contain first quartile (Q1), median (med), third quartile (Q3), maximum (max) and average (avg) of the sizes of all PB(AMO) and PB(EO) constraints of a problem

depth of the BDDs divided by three. In a large number of cases we are able to get depth reductions of up to seven times, and in few cases the reduction is more than ten times.

Finally, in Fig. 9 we compare our contributions with the implication chain based technique to encode PB(AMO) constraints proposed in Abío et al. (2015), which we denote by *IC*. We apply EO reduction whenever possible, that is, for RCPSP/t and CA we compare *PB(AMO)* and *IC*, and for MRCPSP and MMKP we compare *PB(EO)* and *IC* (also with EO reduction in the PB constraints). Tables 4 and 5 contain a numeric comparison. The results suggest that in MRCPSP and CA both approaches behave similarly, but in RCPSP/t

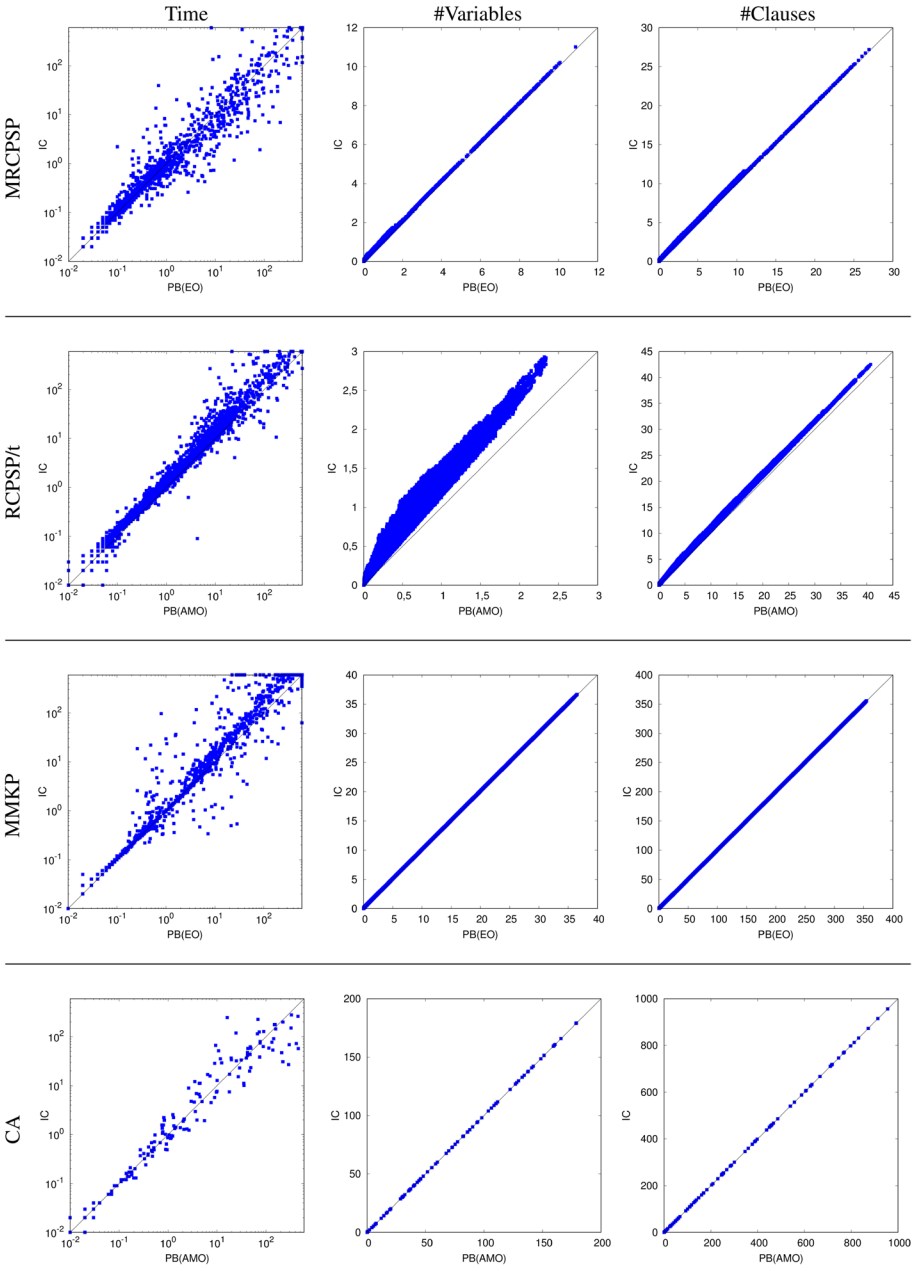


Fig. 9 Time in seconds (left), number of variables in thousands (middle) and number of clauses in thousands (right) comparison of the *PB(AMO)* (for RCPSP/t and CA) and *PB(EO)* (for MRCPSP and MMKP) against *IC* (with EO reduction applied in MRCPSP and MMKP)

and MMKP the AMO-MDD based encoding performs better (e.g. the difference in timeouts between both approaches is 9 and 48). We have run a paired t-test to determine if there is a significant difference in the average run time between the *IC* and *PB(AMO)*/*PB(EO)*

approaches, which is not clear apparently in the cases of the MRCPSP and CA. We get a p-value$= 0.4665$ in the MRCPSP and p-value$= 0.0591$ in CA. Therefore there is no evidence that one approach is better than the other in MRCPSP, and in the case of CA there is no clear statistical evidence but we obtain slightly better results with *IC*. Regarding the RCPSP/t and MMKP, *PB(AMO)*/*PB(EO)* is statistically significantly better than *IC*, with a p-value smaller than $2.2e-16$ in both cases. The reason may be due to the fact that the RCPSP/t and MMKP contains big AMO constraints, and the *IC* approach needs to explicitly encode the AMOs using a particular encoding that introduces additional Boolean variables and clauses. Size results show this increment, which is only remarkable in the RCPSP/t.

8 Related work

There exist many works in the literature on the encoding of PB constraints into SAT, not only based on BDDs, but also on adder networks, sorting networks and other approaches (Eén and Sorensson 2006; Bailleux et al. 2006, 2009; Abío et al. 2012; Hölldobler et al. 2012; Tamura et al. 2013; Joshi et al. 2015). Different SAT encodings of PB constraints have been compared in Philipp and Steinke (2015). This paper introduces the *PBLIB*, a library to translate PB constraints into CNF formulas which includes fifteen different encodings of PB constraints from the literature. In the experiments performed in that paper, the BDD-based approach clearly outperforms the other encodings in terms of solving time.

The use of BDDs to encode PB constraints to SAT was firstly considered in Eén and Sorensson (2006). The approach consists in, first of all, representing the PB constraint as a BDD, then treating the BDD as a circuit of if-then-else gates, and finally translating this circuit to clauses by the Tseitin transformation. The result is a GAC encoding which introduces one fresh variable and six ternary clauses per node, defining that the fresh variable is equivalent to the PB constraint represented by that node. By the same time, in Bailleux et al. (2006) it was introduced an encoding which, although not derived from an explicit BDD, in essence gives the same as the BDD encoding proposed in Eén and Sorensson (2006), with two main differences. The first difference is that, by assuming monotonicity of PB constraints, only two ternary clauses and two binary clauses per node are required. The second difference is that the encoding procedure cannot guarantee that the implicitly generated BDD is reduced.

Those encodings introduce a number of clauses and fresh variables linear in the number of nodes of the encoded BDD. Since there are PB constraints which only have ROBDD representations of exponential size in the number of variables (Hosaka et al. 1994), the encodings have exponential size in the worst case. Nevertheless, there exist polynomial size encodings of PB constraints that maintain GAC by unit propagation (Bailleux et al. 2009). In Abío et al. (2012) it is proved that all PB constraints whose coefficients are powers of two have a polynomial size ROBDD representation. Using the fact that any PB constraint can be reduced to a set of equalities plus a PB constraint whose coefficients are all powers of two, they provide a BDD-based GAC polynomial encoding of PB constraints. Also in Abío et al. (2012), it is presented a simplification of the encoding in Eén and Sorensson (2006) which, assuming that the encoded PB constraint is monotonic, it only introduces one binary clause, one ternary clause, and one fresh variable per node. The main difference with respect to Eén and Sorensson (2006) is that the selector variable of a node is not

defined to have the truth value of the PB constraint represented by that node, but it is unit-propagated to false whenever a partial assignment does not satisfy the PB constraint. This encoding is also GAC. Another contribution in Abío et al. (2012) is an algorithm to construct ROBDDs representing PB constraints, which runs in polynomial time with respect to the size of the final ROBDD.

In Abío and Stuckey (2014) it is presented a generalization of the two-clause encoding from Abío et al. (2012), for the case of MDDs representing Linear Integer Arithmetic (LIA) constraints, and also a generalization of the algorithm in Abío et al. (2012) to construct such MDDs. This technique is closely related to our work. In fact, a LIA expression like $q_1 z_1 + \cdots + q_m z_m \leq K$ can be easily converted to a PB(EO) constraint by introducing a direct encoding for each integer variables z_i with domain $[0, d_i]$, which introduces Boolean variables $x^i_0, \ldots, x^i_{d_i}$ and the corresponding EO constraint. Then the LIA constraint must be replaced by a PB constraint of the form, $\sum_{i=1}^{m} \sum_{j=0}^{d_i} (q_i \cdot d_j) \cdot x^i_j \leq K$. Nevertheless, as explained in the introduction, PB models where AMO constraints occur are ubiquitous in the literature. Our approach allows for a direct efficient SAT encoding of these models without the need of previously reformulating those models to LIA expressions. This reformulation might be involved in some cases, where AMO constraints over Boolean variables have no natural semantic relation with an integer variable in the problem at hand. For instance, in the case of RCPSP-like problems, AMOs arise implicitly from precedence relations between activities (Bofill et al. 2017).

The MDD encoding from Abío and Stuckey (2014) is revisited in Abío et al. (2016), where also some other encodings of MDDs representing LIA constraints are introduced. These are based on encodings of paths, modelling whether an assignment follows a particular path in the MDD. The encodings introduce fresh variables and clauses both for the nodes and the edges of the MDD, and all of them are of linear size with respect to the size of the MDD. In Abío et al. (2015) are presented the PB constraints with implication chains that we describe in Sect. 6.3.

9 Conclusions

With the goal of finding efficient SAT encodings for PB constraints we have defined PB($\mathscr{C}$) constraints, which are a generalization of PB constraints that assume some relations $\mathscr{C}$ between subsets of the variables in the PB constraint. We have shown how to take advantage of this feature by means of a specialized and compact type of decision diagrams (AMO-MDD) to represent PB constraints in the presence of AMO/EO relations. Finally, we have provided a UP-maintaining GAC encoding for PB(AMO) constraints based on AMO-MDDs.

We have reported a huge impact in the size of the encodings as well as in the solving time when using this approach. The presented techniques conform a new and efficient way of handling PB constraints with SAT. Encoding PB($\mathscr{C}$) constraints in a combined way may let to obtain reasonably sized SAT translations that otherwise could be too big for a SAT solver.

From our point of view, PB($\mathscr{C}$) can be seen as a new global constraint applicable to many domains, which is interesting by itself and it can be handled with different solving approaches. In particular, other SAT encodings of PB($\mathscr{C}$) could be studied, not necessarily based on decision diagrams nor limited to AMO constraints.

In the experimental section of this paper we have tackled problems of four different domains and we have obtained significant improvements in the solving times. In the studied problems, AMO/EO constraints either appear explicitly or are easily identifiable. Identifying the AMO constraints implied by the formulation of the problem at hand is a key factor of this approach, since reduction in size strongly depends on it. It is a matter of future work to use methods to automatically detect AMO constraints in any given formula containing PB constraints.

Acknowledgements This work was supported by MINECO/FEDER, UE (grant number TIN2015-66293-R), Ayudas para Contratos Predoctorales 2016, funded by MINECO and co-funded by FSE (grant number BES-2016-076867), and MICINN/FEDER, UE (grant number RTI2018-095609-B-I00).

References

Abío I, Stuckey PJ (2014) Encoding linear constraints into SAT. In: International conference on principles and practice of constraint programming (CP), lecture notes in computer science, vol 8656, pp 75–91

Abío I, Nieuwenhuis R, Oliveras A, Rodríguez-Carbonell E, Mayer-Eichberger V (2012) A new look at BDDs for pseudo-Boolean constraints. J Artif Intell Res 45:443–480

Abío I, Mayer-Eichberger V, Stuckey PJ (2015) Encoding linear constraints with implication chains to CNF. In: International conference on principles and practice of constraint programming (CP), lecture notes in computer science, vol 9255, Springer, pp 3–11

Abío I, Gange G, Mayer-Eichberger V, Stuckey PJ (2016) On CNF encodings of decision diagrams. In: 13th International conference on integration of AI and OR techniques in constraint programming (CPAIOR), lecture notes in computer science, vol 9676, pp 1–17

Bacchus F, Winter J (2003) Effective preprocessing with hyper-resolution and equality reduction. In: International conference on theory and applications of satisfiability testing (SAT), lecture notes in computer science, vol 2919, pp 341–355

Bailleux O, Boufkhad Y, Roussel O (2006) A translation of pseudo-boolean constraints to SAT. J Satisf Boolean Model Comput 2:191–200

Bailleux O, Boufkhad Y, Roussel O (2009) New encodings of pseudo-boolean constraints into CNF. In: International conference on theory and applications of satisfiability testing (SAT), lecture notes in computer science, vol 5584, pp 181–194

Bofill M, Busquets D, Muñoz V, Villaret M (2013) Reformulation based MaxSAT robustness. Constraints 18(2):202–235

Bofill M, Palahí M, Suy J, Villaret M (2014) Solving intensional weighted CSPs by incremental optimization with BDDs. In: Prodeedings of the 20th international conference on principles and practice of constraint programming, CP2014, pp 207–223. https://doi.org/10.1007/978-3-319-10428-7_17

Bofill M, Coll J, Suy J, Villaret M (2016) Solving the multi-mode resource-constrained project scheduling problem with SMT. In: 28th IEEE international conference on tools with artificial intelligence (ICTAI), pp 239–246

Bofill M, Coll J, Suy J, Villaret M (2017) Compact MDDs for pseudo-Boolean constraints with at-most-one relations in resource-constrained scheduling problems. In: Proceedings of the twenty-sixth international joint conference on artificial intelligence (IJCAI), pp 555–562

Bofill M, Coll J, Suy J, Villaret M (2019) SAT encodings of pseudo-Boolean constraints with at-most-one relations. In: Proceedings of the 16th international conference of integration of constraint programming, artificial intelligence, and operations reserarch (CPAIOR), pp 112–128. https://doi.org/10.1007/978-3-030-19212-9_8

Brucker P, Drexl A, Möhring R, Neumann K, Pesch E (1999) Resource-constrained project scheduling: notation, classification, models, and methods. Eur J Oper Res 112(1):3–41

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manag Sci 6(1):80–91

De Vries S, Vohra RV (2003) Combinatorial auctions: a survey. INFORMS J Comput 15(3):284–309

Dutertre B, de Moura L (2006) The Yices SMT solver. Tech. rep., Computer Science Laboratory, SRI International, available at http://yices.csl.sri.com

Eén N, Biere A (2005) Effective preprocessing in SAT through variable and clause elimination. In: International conference on theory and applications of satisfiability testing (SAT), lecture notes in computer science, vol 3569, pp 61–75

Eén N, Sorensson N (2006) Translating pseudo-boolean constraints into SAT. J Satisf Boolean Model Comput 2:1–26

Hartmann S (2013) Project scheduling with resource capacities and requests varying with time: a case study. Flex Serv Manuf J 25(1–2):74–93

Hölldobler S, Manthey N, Steinke P (2012) A compact encoding of pseudo-Boolean constraints into SAT. In: KI 2012: advances in artificial intelligence 35th annual german conference on artificial intelligence, lecture notes in computer science, vol 7526, pp 107–118

Hosaka K, Takenaga Y, Yajima S (1994) On the size of ordered binary decision diagrams representing threshold functions. In: 5th International symposium on algorithms and computation (ISAAC), lecture notes in computer science, vol 834, pp 584–592

Joshi S, Martins R, Manquinho VM (2015) Generalized totalizer encoding for pseudo-boolean constraints. In: Principles and practice of constraint programming—CP 2015, 21st international conference, LNCS, vol 9255, Springer, pp 200–209

Kellerer H, Pferschy U, Pisinger D (2004) Multidimensional knapsack problems. In: Knapsack problems, Springer, Berlin, pp 235–283. https://doi.org/10.1007/978-3-540-24777-7_9

Kolisch R, Sprecher A (1997) PSPLIB: a project scheduling problem library. Eur J Oper Res 96(1):205–216

Laporte G (1992) The vehicle routing problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(3):345–358

Leyton-Brown K, Shoham Y (2006) A test suite for combinatorial auctions. In: Combinatorial auctions, The MIT Press, chap 18, pp 451–478

Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. J ACM 7(4):326–329

Philipp T, Steinke P (2015) PBLib–a library for encoding pseudo-boolean constraints into CNF. In: International conference on theory and applications of satisfiability testing (SAT), lecture notes in computer science, vol 9340, pp 9–16

Pritsker AAB, Waiters LJ, Wlofe PM (1996) Multiproject scheduling with limited resources: a zero-one programming approach. Manag Sci 16:93–108

Srinivasan A, Ham T, Malik S, Brayton RK (1990) Algorithms for discrete function manipulation. In: IEEE international conference on computer-aided design (ICCAD), IEEE, pp 92–95

Tamura N, Banbara M, Soh T (2013) Compiling pseudo-boolean constraints to SAT with order encoding. In: 25th IEEE international conference on tools with artificial intelligence (ICTAI), pp 1020–1027

Van Peteghem V, Vanhoucke M (2014) An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. Eur J Oper Res 235(1):62–72