SAT Encodings of Pseudo-Boolean Constraints with At-Most-One Relations

Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret

Universitat de Girona, Girona, Spain
{miquel.bofill,jordi.coll,josep.suy,mateu.villaret}@imae.udg.edu

Abstract. Pseudo-Boolean (PB) constraints appear often in a large variety of constraint satisfaction problems. Encoding such constraints to SAT has proved to be an efficient approach in many applications. However, most of the existing encodings in the literature do not take profit from side constraints that often occur together with the PB constraints. In this work we introduce specialized encodings for PB constraints occurring together with at-most-one (AMO) constraints over subsets of their variables. We show that many state-of-the-art SAT encodings of PB constraints from the literature can be dramatically reduced in size thanks to the presence of AMO constraints. Moreover, the new encodings preserve the propagation properties of the original ones. Our experiments show a significant reduction in solving time thanks to the new encodings.

Keywords: SAT · Pseudo-Boolean · Encoding

1 Introduction

Linear equations are ubiquitous in Constraint Satisfaction Problems (CSP). A particular case are Pseudo-Boolean (PB) constraints, which 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 successful approach to solve CSPs is to encode them to propositional Boolean formulas, which are then solved using off-the-shelf SAT solvers. Therefore, there exist many works on encoding PB constraints to SAT [20]. State-of-the-art encodings are based on Binary Decision Diagrams [12, 1], Sequential Weight Counters [15], Generalized Totalizers [16], and Polynomial Watchdog schemes [7, 18].

In [10] it is proposed a Multi-Decision Diagram (MDD) based SAT encoding of PB constraints, under the assumption that there exist some at-most-one (AMO) relations on disjoint subsets of variables. The AMO relations let to erase certain interpretations from decision diagrams, and to represent the PB as an MDD instead of as a Binary Decision Diagram (BDD). This way, the resulting encoding is notably smaller than the one of an equivalent BDD, and solving time is significantly reduced. This technique has been used in [9, 10] to formulate particular kinds of scheduling problems. Similar techniques are also applied in [2] in the field of Mixed Integer Linear Programming, where AMO relations between a set of 0/1 variables are used to substitute these variables by an integer variable.

Efficient encodings of conjunctions of PB and AMO constraints can have a high impact in a wider range of CSPs. Such combination of constraints appear in settings where it has to be chosen one option among a set of incompatible options, and this decision has an associated cost. A non exhaustive list of applications where this happens are logistics [8], resource allocation [17], capital budgeting [21], telecommunications [22], combinatorial auctions [11] and routing [19]. In short, any problem which is essentially a multi-choice knapsack problem is likely to contain both PB and AMO constraints. Therefore, it is of big interest to provide new and better encodings for this combination constraints.

In this paper we address the question of whether other SAT encodings of PB constraints different than decision diagram-based ones can be improved in the presence of AMOs. We revisit many state-of-the-art SAT encodings of PB constraints and propose improved versions of those encodings for conjunctions of PB and AMO constraints. More precisely, we provide modifications of the Sequential Weight Counter, Generalized Totalizer, and Global Polynomial Watchdog encodings. We also show that the new encodings preserve the propagation properties of the original ones. Our experimental results show that the size of the SAT encodings of PB constraints can be dramatically reduced thanks to taking the AMO constraints into account, and that there can be a huge time performance improvement when using the new encodings. We provide datasets which contain AMO constraints and PB constraints with different configurations, and we show that some encodings are better than others for particular kinds of PB.

2 Preliminaries

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. 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 l of each clause in F belongs to A. Such an assignment is called a *model* of the formula. In this paper we will assume that all propositional formulas are in CNF. Given two Boolean functions F and G, we say that G is a logical consequence of F, written $F \models G$, iff every model of F is also a model of G.

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. *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* $\# \in \{<, \leq, =, \geq, >\}$.

It is usually assumed that # is $\leq$, and that K and all q_i are non-negative, since the other cases can be easily reduced to this one [12].

Definition 3. *By* PB(AMO) constraint *we refer to a constraint of the form* $P \wedge M_1 \wedge \cdots \wedge M_m$*, where* P *is a PB constraint, and* $M_1, \ldots, M_m$ *are AMO constraints.*

Unit propagation (UP) is a propagation mechanism used in modern SAT solvers. It is based on the principle that if a clause contains a single literal (i.e., under a given assignment, all literals but one are false), then every model must make that literal true. Hence, the assignment can be extended with this literal.

We say that a formula G is an *encoding* of a Boolean function F if the following holds: given an assignment A over the variables of F, A satisfies F iff A can be extended to a satisfying assignment of G.

An encoding E of a constraint C is said to *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 will extend A to $A \cup \{x\}$ (respectively $A \cup \{\overline{x}\}$) [7].

3 New Encodings of monotonic decreasing PB(AMO) Constraints

In this section we present three different SAT encodings for PB(AMO) constraints. Given a PB(AMO) of the form $P \wedge M_1 \wedge \cdots \wedge M_m$, a straightforward approach to encode it is to generate a formula F of the form $G \wedge H_1 \wedge \cdots \wedge H_m$, where G is an encoding of P, and each H_i is an encoding of M_i. Instead, similarly to the MDD-based approach of [10], we propose to encode PB(AMO)s in a combined way. On the one hand we encode the conjunction of AMO constraints in the usual way, i.e., we encode each AMO separately and use the conjunction of all the resulting clauses. On the other hand we encode the PB constraint assuming that the accompanying AMO constraints are already enforced in some way. This is precisely what will let us reduce the size of the encoding of the PB constraint. We do not restrict to a particular encoding for the AMO constraints. Even more, in the context of a bigger formula, if the AMO constraints are logically implied by the formula at hand, then the encoding of the PB constraint will suffice to obtain a correct encoding of the PB(AMO) constraint.

We start each subsection giving an intuitive explanation of an already existing encoding of PB constraints. Then, we propose a generalized version of it in order to encode PB(AMO) constraints. Since PB(AMO) constraints generalize PB constraints,[1] we follow the convention of naming the new encodings after the original encoding, prefixing them with the word *Generalized*, e.g., from the Sequential Weight Counter (SWC) encoding we provide the Generalized Sequential Weight Counter (GSWC) encoding.

[1] A PB constraint of the form $\sum_{i=1}^{n} q_i x_i \leq K$ corresponds to a PB(AMO) constraint of the form $\sum_{i=1}^{n} q_i x_i \leq K \wedge x_1 \leq 1 \wedge \cdots \wedge x_n \leq 1$.

We make the following assumptions on the PB(AMO) constraint $P \wedge M_1 \wedge \cdots \wedge M_m$ at hand: (i) every variable in P occurs at least in one M_i (a variable x can always be included in a single-variable AMO constraint of the form $x \leq 1$); (ii) unless otherwise stated, we assume that P is of the form $\sum_{i=1}^{n} q_i x_i \leq K$, with $q_i \geq 0$, i.e. it is monotonic decreasing; (iii) P is neither trivially true nor false (i.e., we assume that $0 \leq K < \sum_{i=1}^{n} q_i$); (iv) no variable is trivially removable (i.e., $0 < q_i \leq K$).

Input. Given a PB(AMO) constraint $\mathcal{P}$ of the form $P \wedge M_1 \wedge \cdots \wedge M_m$, the encodings of the following subsections receive as input the PB constraint P and a partition $\mathcal{X} = \{X_1, \ldots, X_N\}$ of the variables of P, such that any assignment satisfying $M_1 \wedge \cdots \wedge M_m$ also satisfies the AMO constraints $\sum_{x_{i_j} \in X_i} x_{i_j} \leq 1$, for all $X_i \in \mathcal{X}$. Note that there may be more than one possible partition, and that for every partition we will have $n = \sum_{i=1}^{N} |X_i|$, where n is the number of variables in the scope of P. A simple way to obtain such a partition is to start with a list of sets $X_1, \ldots, X_m$, where each set X_i contains the variables in the scope of M_i. Then, remove every variable from all X_i but one, and finally remove all the empty sets.

3.1 Sequential Weight Counter Encoding

The *Sequential Weight Counter* (SWC) encoding for PB constraints was introduced in [15]. The idea is to encode the PB constraint by a circuit that sequentially sums from left to right the coefficients (a.k.a. weights) q_i whose variable x_i is set to true. Specifically, given a PB constraint $\sum_{i=1}^{n} q_i x_i \leq K$, there is a sequence of n counters of K inputs and K outputs, where the i-th counter is associated to the variable x_i. Each counter receives as input a vector of Boolean variables, which is the unary representation of an integer value, and adds the weight q_i to the output if the associated variable x_i is set to true. Therefore, the i-th counter receives as input $\sum_{j=1}^{i-1} q_j x_j$ and outputs $\sum_{j=1}^{i} q_j x_j$. Note that the output of the counter number $i-1$ is the input of the i-th counter.

An example of a sequence of counters is shown in Figure 1. The encoding introduces $n \cdot K$ variables, denoted $s_{i,j}$, with $1 \leq i \leq n$, $1 \leq j \leq K$, where $s_{i,j}$ is the j-th output of the i-th counter and also the j-th input of the $(i+1)$-th counter. The encoding introduces the clauses

$$\overline{s_{i-1,j}} \vee s_{i,j} \qquad 2 \leq i < n, 1 \leq j \leq K \tag{1}$$

$$\overline{x_i} \vee s_{i,j} \qquad 1 \leq i < n, 1 \leq j \leq q_i \tag{2}$$

$$\overline{s_{i-1,j}} \vee \overline{x_i} \vee s_{i,j+q_i} \qquad 2 \leq i < n, 1 \leq j \leq K - q_i \tag{3}$$

$$\overline{s_{i-1,K+1-q_i}} \vee \overline{x_i} \qquad 2 \leq i \leq n \tag{4}$$

where $s_{0,j}$ is the constant 0 for all j, to represent the input of the first counter which is the empty sum. Clauses (1) state that $\sum_{j=1}^{i} q_j x_j \geq \sum_{j=1}^{i-1} q_j x_j$. Clauses (2) and (3) enforce that if a variable x_i is true then its coefficient is added to the input of the next counter. Finally, Clauses (4) enforce that the sum never exceeds K.

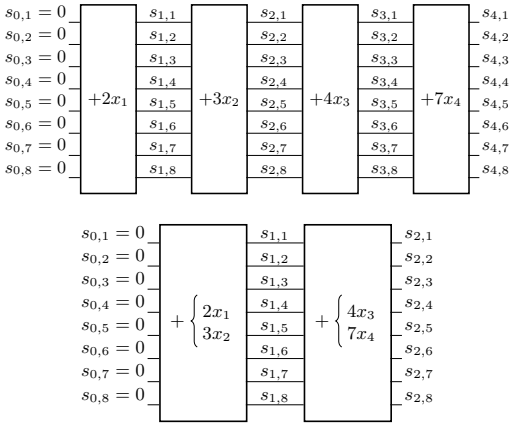

Fig. 1. At the top: high level circuit representation of $SWC(2x_1+3x_2+4x_3+7x_4 \leq 8)$. At the bottom: high level circuit representation of $GSWC(2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 8, \{\{x_1, x_2\}, \{x_3, x_4\}\})$.

Generalized Sequential Weight Counter (GSWC) We define the GSWC encoding by, instead of associating a single product $q_i x_i$ from the PB constraint to each counter, associating a set of products to each of them. In our generalization, given a partition $\mathcal{X} = \{X_1, \ldots, X_N\}$ of the variables of the PB constraint, the resulting formulation will have just N counters, where the i-th counter will handle all the products $q_l x_l$ for the variables x_l in X_i. If the variables in each set X_i are subject to an AMO constraint then, given an assignment satisfying those constraints, at most one coefficient q_l will be added by each counter, and the output of the whole circuit will correspond to the value of the left hand side sum of the PB constraint, i.e., $\sum_{i=1}^{n} q_i x_i$. Analogously as in the original encoding, we will enforce that it is not reached a sum that exceeds K. The GSWC encoding introduces the following clauses:

$$\overline{s_{i-1,j}} \vee s_{i,j} \qquad 2 \leq i < N,\ 1 \leq j \leq K \tag{5}$$

$$\overline{x_l} \vee s_{i,j} \qquad 1 \leq i < N,\ x_l \in X_i,\ 1 \leq j \leq q_l \tag{6}$$

$$\overline{s_{i-1,j}} \vee \overline{x_l} \vee s_{i,j+q_l} \qquad 2 \leq i < N,\ x_l \in X_i,\ 1 \leq j \leq K - q_l \tag{7}$$

$$\overline{s_{i-1,K+1-q_l}} \vee \overline{x_l} \qquad 2 \leq i \leq N,\ x_l \in X_i \tag{8}$$

Clauses (5) propagate the accumulated sum in the same way as Clauses (1). Clauses (6) and (7) enforce $S_i \geq S_{i-1} + q_l x_l$, for all $x_l \in X_i$, where S_{i-1} and S_i are respectively the input and output value of the i-th counter. Clauses (8) enforce that the sum never exceeds K. A high level circuit representation of a GSWC encoding is shown in Figure 1.

The main difference between the SWC and GSWC encodings is that the latter has only N counters, instead of n, and therefore introduces less fresh variables (assuming $N < n$). Also, the number of Clauses (5) in the GSWC encoding is smaller than the number of Clauses (1) in the SWC encoding. The

SWC encoding requires $O(nK)$ auxiliary variables and $O(nK)$ clauses, while the GSWC encoding requires $O(NK)$ auxiliary variables and $O(nK)$ clauses.

By $GSWC(P, \mathcal{X})$ we denote the set of clauses derived from a PB constraint P and a partition $\mathcal{X}$, as described above.

Lemma 1. *Let $\mathcal{P}$ be a PB(AMO) of the form $P \wedge M_1 \wedge \cdots \wedge M_m$ and $\mathcal{X}$ be a partition of the variables of P such that $M_1 \wedge \cdots \wedge M_m \models \sum_{x_{i_j} \in X_i} x_{i_j} \leq 1$, for all $X_i \in \mathcal{X}$. The conjunction of $GSWC(P, \mathcal{X})$ with an encoding of $M_1 \wedge \cdots \wedge M_m$ is an encoding of $\mathcal{P}$.*

In [15] it is proved that the SWC encoding UP-maintains GAC. The GSWC encoding preserves this property.

Theorem 1. *Let $\mathcal{P}$ be a PB(AMO) of the form $P \wedge M_1 \wedge \cdots \wedge M_m$ and $\mathcal{X}$ be a partition of the variables of P such that $M_1 \wedge \cdots \wedge M_m \models \sum_{x_{i_j} \in X_i} x_{i_j} \leq 1$, for all $X_i \in \mathcal{X}$. The conjunction of $GSWC(P, \mathcal{X})$ with an UP-maintaining GAC encoding of $M_1 \wedge \cdots \wedge M_m$ is UP-maintaining GAC.*

Proof. Let S denote the conjunction of $GSWC(P, \mathcal{X})$ with an UP-maintaining GAC encoding of $M_1 \wedge \cdots \wedge M_m$. Let A be a partial assignment to the variables of S, which is extendible to a satisfying assignment of $\mathcal{P}$. Therefore, no AMO constraint M_i is violated under A. We need to show that for every variable x of $\mathcal{P}$ such that x is not assigned in A, if $A \cup \{x\}$ cannot be extended to a satisfying assignment of $\mathcal{P}$, then x is set to false by unit propagating A on S (note that $A \cup \{\overline{x}\}$ can always be extended to a satisfying assignment, so we don't need to consider this case). W.l.o.g., assume that $x_1 \in X_1$ is such variable. If $A \cup \{x_1\}$ cannot be extended to a satisfying assignment of $M_1 \wedge \cdots \wedge M_m$ then, by the assumption that S contains an UP-maintaining GAC encoding of $M_1 \wedge \cdots \wedge M_m$, we have that x_1 is set to false by unit propagation. Assume now the contrary, i.e., that $A \cup \{x_1\}$ can be extended to an assignment satisfying the AMOs. In this case, the reason why UP should set x_1 to false is that $A \cup \{x_1\}$ cannot be extended to satisfy P. Since $A \cup \{x_1\}$ does not violate $M_1 \wedge \cdots \wedge M_m$, at most one variable in X_i is true in A, for $2 \leq i \leq N$, and no variable in X_1 is true in A. Let us construct a PB constraint P' from P by picking one variable x_{j_i} from each set X_i, $2 \leq i \leq N$, as follows: if X_i contains a variable which is true in A, then this is the variable to be picked up from X_i, otherwise pick up any variable. We define $P' : q_1 x_1 + \sum_{i=2}^{N} q_{j_i} x_{j_i} \leq K$. Since P' contains all the variables which are true in A, and due to the monotonicity of P, we have that $q_1 x_1 + \sum_{i=2}^{N} q_{j_i} x_{j_i}$ is equisatisfiable to $\sum_{i=1}^{n} q_i x_i$ under the assignment $A \cup \{x_1\}$, and therefore $A \cup \{x_1\}$ can neither be extended to a model of P'. It is not hard to see that $GSWC(P, \mathcal{X})$ contains all the clauses of $SWC(P')$, and it is already proved that the SWC encoding UP-maintains GAC. Therefore, all the clauses required to set x_1 to false by UP are contained in S.

3.2 Generalized Totalizer Encoding

The *Generalized Totalizer* (GT) encoding was presented in [16] as a generalization of the Totalizer encoding for cardinality constraints [6]. The overall idea of

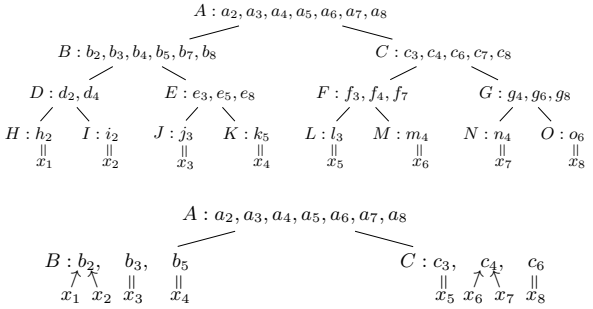

Fig. 2. At the top: binary tree of $GT(2x_1+2x_2+3x_3+5x_4+3x_5+4x_6+4x_7+6x_8 \leq 7)$. At the bottom: binary tree of $GGT(2x_1 + 2x_2 + 3x_3 + 5x_4 + 3x_5 + 4x_6 + 4x_7 + 6x_8 \leq 7, \{\{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7, x_8\}\})$.

GT is to represent a PB constraint $\sum_{i=1}^{n} q_i x_i \leq K$ as a binary tree. Every node of the tree has associated a distinct label and an attribute *vars* which consists of a set of Boolean variables. Each variable x_i of the PB constraint is placed into the attribute *vars* of a different leaf node, and is renamed after the label of the node and its associated coefficient q_i (e.g., given the product $3x_1$, if the variable x_1 is inserted into a leaf node labelled by letter O, then the variable is named o_3). The attribute *vars* of any non-leaf node labelled O contains a variable o_w for every subset of the underlying leaves which sums exactly w, for values of w in the range $[1, K]$, taking i for the value of each leaf node L with variable l_i. Also, *vars* contains a variable o_{K+1} iff any of the sums is larger than K. Figure 2 illustrates an example binary tree.

The clauses of the encoding enforce that each non-leaf variable o_w is set to true if the underlying variables which sum w (or more than K for $w = K + 1$) are set to true. Moreover it is enforced, at the root node, that the variable representing a sum larger than K is false. The GT encoding introduces the following clauses for each non-leaf node O with children L and R:

$$\overline{l_{w_1}} \vee \overline{r_{w_2}} \vee o_{w_3} \quad l_{w_1} \in L.vars,\ r_{w_2} \in R.vars,\ w_3 = \min(w_1 + w_2, K + 1) \tag{9}$$

$$\overline{t_w} \vee o_w \quad t_w \in L.vars \cup R.vars \tag{10}$$

It also introduces the unary clause

$$\overline{a_{k+1}} \tag{11}$$

where A is the root node of the tree and $a_{k+1} \in A.vars$. Note that variable a_{k+1} will exist, since otherwise the constraint would be trivially satisfied. Clauses (9) enforce that the variable o_{w_3} will be set to true by UP if there exists a pair of variables l_{w_1}, r_{w_2} from the children nodes that are set to true and such that $w_3 = \min(w_1 + w_2, K + 1)$. Clauses (10) enforce that the variable o_w will be set to true by UP if some child has a variable t_w set to true. Finally, Clause (11) states that the sum of the tree (i.e., the value of the left hand side expression of the PB constraint) cannot be greater than K.

Generalized Generalized Totalizer (GGT) In our generalization of the GT encoding, we will use the same definition of the binary tree, but the leafs will be instantiated differently. Instead of introducing a leaf node for each variable of the PB constraint, what we do is to introduce a leaf node for each of the sets in the partition $\mathcal{X}$. The leaf node O associated to set X_i will contain a variable o_{q_l} in its *vars* attribute for each different coefficient q_l such that $x_l \in X_i$. If there is a single coefficient q_l, then x_l is renamed as o_{q_l} and placed in $O.vars$, as in the GT encoding. If there are multiple occurrences of a coefficient q_l, we introduce a fresh variable o_{q_l}. The following clauses relate the fresh leaf variables with the variables of the PB constraint:

$$\overline{x_l} \vee o_{q_l} \qquad X_i \in \mathcal{X},\, x_l \in X_i,\, |\{x_{l'} \in X_i \mid q_{l'} = q_l\}| \geq 2 \tag{12}$$

The GGT encoding introduces Clauses (9), (10) and (11) as in the GT encoding, and Clauses (12). Figure 2 depicts the binary tree of a GGT encoding. Note that assuming that an AMO constraint over each set X_i is satisfied, at most one of the variables in each leaf node will be true, and therefore the encoding correctly evaluates $\sum_{i=1}^{n} q_i x_i \leq K$.

The GT encoding requires $O(nK)$ auxiliary variables and $O(nK^2)$ clauses, while GGT encoding requires $O(NK)$ auxiliary variables and $O(NK^2)$ clauses.

By $GGT(P, \mathcal{X})$ we denote the set of clauses derived from a PB constraint P and a partition $\mathcal{X}$, as described above.

Lemma 2. *Let $\mathcal{P}$ be a PB(AMO) of the form $P \wedge M_1 \wedge \cdots \wedge M_m$ and $\mathcal{X}$ be a partition of the variables of P such that $M_1 \wedge \cdots \wedge M_m \models \sum_{x_{i_j} \in X_i} x_{i_j} \leq 1$, for all $X_i \in \mathcal{X}$. The conjunction of $GGT(P, \mathcal{X})$ with an encoding of $M_1 \wedge \cdots \wedge M_m$ is an encoding of $\mathcal{P}$.*

In [16] it is proved that the GT encoding UP-maintains GAC. The GGT encoding preserves this property.

Theorem 2. *Let $\mathcal{P}$ be a PB(AMO) of the form $P \wedge M_1 \wedge \cdots \wedge M_m$ and $\mathcal{X}$ be a partition of the variables of P such that $M_1 \wedge \cdots \wedge M_m \models \sum_{x_{i_j} \in X_i} x_{i_j} \leq 1$, for all $X_i \in \mathcal{X}$. The conjunction of $GGT(P, \mathcal{X})$ with an UP-maintaining GAC encoding of $M_1 \wedge \cdots \wedge M_m$ is UP-maintaining GAC.*

The proof is analogous to the proof of Theorem 1.

3.3 Global Polynomial Watchdog Encoding

The *Global Polynomial Watchdog* (GPW) encoding was presented in [7]. It uses as basis a *polynomial watchdog* formula, denoted by $PW(P)$, which is associated with a PB constraint P. The formula $PW(P)$ has a variable named the *output* variable, denoted w, which is set to 1 by UP as soon as P is falsified.

The GPW encoding is defined for PB constraints of the form $\sum_{i=1}^{n} q_i x_i < K$, i.e., with a strict inequality instead of a non-strict one. The first step is to normalize the constraint to the form $T + \sum_{i=1}^{n} q_i x_i < m2^p$, where p, T and

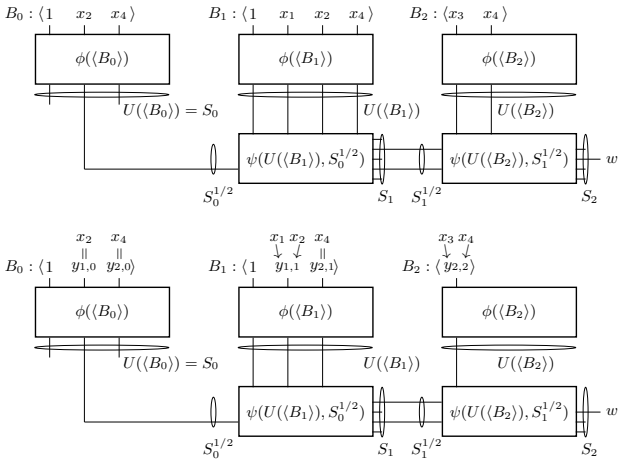

Fig. 3. At the top: circuit representation of $PW(2x_1 + 3x_2 + 4x_3 + 7x_4 < 9)$. At the bottom: circuit representation of $PW(2x_1 + 3x_2 + 4x_3 + 7x_4 < 9, \{\{x_1, x_2\}, \{x_3, x_4\}\})$.

m are defined as follows: $p = \lfloor \log_2(\max_{i=1..n}(q_i)) \rfloor$ is the index of the most significant bit in the binary representation of the largest coefficient q_i, being 0 the index of the least significant bit. In other words, $p+1$ is the number of bits needed to represent q_i in binary notation; T is the smallest non-negative integer such that $K + T$ is a multiple of 2^p; $m = (K + T)/2^p$.

Once the constraint is expressed in this form, it is computed a set B_r of variables of P (called *bucket*) for each bit $0 \leq r \leq p$. We denote by $b_r(q_i)$ the r-th bit of the binary representation of the integer q_i. Bucket B_r contains all the variables x_i such that $b_r(q_i) = 1$. Bucket B_r also contains a 1 constant if $b_r(T) = 1$.

Example 1. The following is the transformation to apply to the PB constraint $2x_1 + 3x_2 + 4x_3 + 7x_4 < 9$. We have that $p = 2$, and $T = 3$ is the smallest integer such that $T + K = 12$ is a multiple of 2^p, with $m = 3$. Therefore, the constraint is expressed as $3 + 2x_1 + 3x_2 + 4x_3 + 7x_4 < 12$. The content of buckets B_0, B_1 and B_2 is illustrated in Figure 3.

The idea is to decompose each coefficient in its binary representation an sum each bit having the same weight.

The formula $PW(P)$ can be represented as a circuit, as can be seen in Figure 3 corresponding to Example 1. We denote by $\langle B_r \rangle$ a vector with an arbitrary order containing the elements of bucket B_r. The formula $PW(P)$ uses two main components: the formulas $\phi(V)$ and $\psi(V_1, V_2)$. The formula $\phi(V)$ has as input a vector of Boolean variables V, and has as output a vector of $|V|$ variables named $U(V)$. The formula $\phi(V)$ enforces that $U(V)$ is the unary representation of the sum of the input variables. The formula $\psi(V_1, V_2)$, has as input two vectors of variables V_1 and V_2, which are the unary representation of two integers, and has

as output a vector of $|V_1|+|V_2|$ variables named S. The formula $\psi(V_1, V_2)$ enforces that S is the unary representation of $V_1 + V_2$. In the definition of $PW(P)$, we denote by S_r the output of the ψ formula related with bucket B_r, for $1 \leq r \leq p$, and we define $S_0 = U(\langle B_0 \rangle)$. Half of the value of S_k, for a weight 2^k, is computed with operator $\frac{1}{2}$ and integrated in the sum for weight 2^{k+1}. Then, the formula $PW(P)$ is defined as the conjunction of these two formulas:

$$\phi(\langle B_r \rangle) \qquad 0 \leq r \leq p \tag{13}$$

$$\psi(U(\langle B_r \rangle), S_{r-1}^{1/2}) \qquad 1 \leq r \leq p \tag{14}$$

The GPW encoding is then defined as

$$PW(P) \wedge \overline{w} \tag{15}$$

where $PW(P)$ encodes ϕ with a totalizer, and ψ with an adder of unary numbers. Due space restrictions we cannot elaborate more on the correctness of this encoding, so we refer the reader to [7]. The basic idea is that the m-th bit of S_p, represented with variable w, is set to 1 by UP if the sum of the constraint is greater or equal than $m2^p = T + K$. If w is set to 1 the encoding is not satisfied.

Generalized Global Polynomial Watchdog (GGPW) We define the GGPW encoding by using a *generalized polynomial watchdog formula* $PW(P, \mathcal{X})$ instead of the original polynomial watchdog formula. Again, P has to be normalized to the form $T + \sum_{i=1}^{n} q_i x_i < m2^p$ in the same way as in $PW(P)$. For each set X_i, $PW(P, \mathcal{X})$ will contain a vector of variables $Y_i = \langle y_{i,p}, y_{i,p-1}, \ldots, y_{i,0} \rangle$.

Y_i is interpreted as binary number, where (at least) the bits corresponding to the binary representation of q_l, for all $x_l \in X_i$ such that x_l is true, are set to one. Therefore, when exactly one x_l is true, Y_i will be greater than or equal to q_l. The following clauses define the variables Y_i:

$$\overline{x_l} \vee y_{i,r} \qquad 0 \leq r \leq p,\ 1 \leq i \leq N,\ x_l \in X_i,\ b_r(q_l) = 1 \tag{16}$$

In this case the bucket B_r, for each bit $0 \leq r \leq p$, will contain the variables $y_{1,r}, y_{2,r}, \ldots, y_{N,r}$. Bucket B_r will also contain a 1 constant if $b_r(T) = 1$.

The formula $PW(P, \mathcal{X})$ is defined as the conjunction of (13), (14) and (16). Some considerations can be taken into account on Clauses (16) in order to optimize the encoding:

- If there is no $x_l \in X_i$ such that $b_r(q_l) = 1$, and therefore the variable $y_{i,r}$ does not appear in any clause of (16), then this variable is not created nor included in any bucket.
- If there is only one variable $x_l \in X_i$ such that $b_r(q_l) = 1$, then the variable $y_{i,r}$ is the variable x_l itself, and Clause (16) is not added for $y_{i,r}$.
- Otherwise, $y_{i,r}$ is indeed a fresh variable and Clause (16) is added.

Figure 3 contains a circuit representation of $PW(P, \mathcal{X})$.

The GGPW encoding is defined as

$$PW(P, \mathcal{X}) \wedge \overline{w} \tag{17}$$

where $PW(P, \mathcal{X})$ encodes ϕ with a totalizer, and ψ with an adder of unary numbers. Similarly as in the other newly introduced encodings, given an assignment that satisfies an AMO constraint over each $X_i \in \mathcal{X}$, this encoding represents the PB constraint $\sum_{i=1}^{n} q_i x_i < K$ in a more compact way.

The GPW encoding introduces $O(n \log(n) \log(q_{max}))$ fresh variables and $O(n^2 \log(n) \log(q_{max}))$ clauses, while the GGPW encoding introduces $O(N \log(N) \log(q_{max}))$ fresh variables and $O(N^2 \log(N) \log(q_{max}))$ clauses, where $q_{max} = \max_{i=1}^{n} q_i$. This follows from the fact that a totalizer ϕ with n input variables requires $O(n \log(n))$ auxiliary variables and $O(n^2 \log(n))$ clauses, and an adder ψ of unary numbers with n input variables requires $O(n)$ auxiliary variables and $O(n^2)$ clauses; see [7].

Lemma 3. *Let $\mathcal{P}$ be a PB(AMO) of the form $P \wedge M_1 \wedge \cdots \wedge M_m$, with P of the form $\sum_{i=1}^{n} q_i x_i < K$, and $\mathcal{X}$ be a partition of the variables of P such that $M_1 \wedge \cdots \wedge M_m \models \sum_{x_{i_j} \in X_i} x_{i_j} \leq 1$, for all $X_i \in \mathcal{X}$. The conjunction of $GGPW(P, \mathcal{X})$ with an encoding of $M_1 \wedge \cdots \wedge M_m$ is an encoding of $\mathcal{P}$.*

In [7] it is shown that the GPW encoding does not UP-maintain GAC. As stated earlier, a PB(AMO) constraint with only AMO constraints of one variable is indeed a PB constraint. In this case the GGPW and GPW encodings would be identical. Therefore, the GGPW encoding does neither UP-maintain GAC.

Binary Merger (BM) The *Binary Merger* (BM) encoding was introduced in [18]. This encoding is essentially another implementation of the GPW encoding, in which the formulas ϕ and ψ are respectively implemented using sorters and odd-even mergers [4]. This way, the BM encoding is asymptotically smaller in the number of clauses and slightly bigger in the number of variables than the GPW encoding. The BM encoding can be generalized to deal with PB(AMO) constraints in the same way as GPW encoding. However, in our experiments we have not observed significant differences between the BM and GPW based encodings, and therefore we do not provide detailed results for BM encoding.

4 Normalization of PB(AMO) constraints

Most existing encodings of PB constraints to SAT are designed for constraints of the form $\sum_{i=i}^{n} q_i x_i \leq K$, with non-negative q_i, since other cases can be easily transformed to this one. Also the encoding that we have presented in Section 3 requires this normalization. The usual way of getting rid of negative coefficients [12] is by using the equality $x = 1 - \overline{x}$, e.g. $-2x_1 + 6x_2 \leq 5 \equiv 2\overline{x_1} + 6x_2 \leq 7$. Then, if we want to encode a constraint of the form $\sum_{i=i}^{n} q_i x_i \geq K$, 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 rewriting might not be applicable to PB(AMO) constraints. 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 $P : -x_1 - 3x_2 - 4x_3 - 2x_4 - 3x_5 - 5x_6 \leq -6$. If we remove the negative coefficients we obtain $P' : \overline{x_1} + 3\overline{x_2} + 4\overline{x_3} + 2\overline{x_4} + 3\overline{x_5} + 5\overline{x_6} \leq 12$. Notice that $x_1 + x_2 + x_3 \leq 1$ (similarly $x_4 + x_5 + x_6 \leq 1$) no longer impose AMO constraints over the literals of P', and therefore $P' \wedge x_1 + x_2 + x_3 \leq 1 \wedge x_4 + x_5 + x_6 \leq 1$ is not a PB(AMO) constraint. We could still use any existing PB encoding to encode P' without taking the AMO constraints into account, but we would not be using the simplification potential of PB(AMO) constraints.

We present another rewriting procedure 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 AMO constraints. Moreover, this procedure let choose the polarity in the PB of any variable by using $x = 1 - \overline{x}$, even if the substitution introduces a negative coefficient, because it will be dealt with in the rewriting. Hence it is possible to make use not only of AMOs between the variables of the PB, but also of AMOs between literals with any polarity.

The first step is, for each AMO constraint of the form $x_{i_1} + \cdots + x_{i_l} \leq 1$, define a fresh variable y_i as $y_i \leftrightarrow \overline{x_{i_1}} \wedge \cdots \wedge \overline{x_{i_l}}$. Then, all the auxiliary y_i variables can be included in the PB constraint with coefficient 0. In previous example, from:

$$x_1 - 3x_2 - 4x_3 - 2x_4 - 3x_5 - 5x_6 \leq -6 \wedge x_1 + x_2 + x_3 \leq 1 \wedge x_4 + x_5 + x_6 \leq 1$$

$$\begin{aligned} \text{we get :} \qquad & 0y_1 - x_1 - 3x_2 - 4x_3 + 0y_2 - 2x_4 - 3x_5 - 5x_6 \leq -6 \\ & \wedge x_1 + x_2 + x_3 \leq 1 \ \wedge \ x_4 + x_5 + x_6 \leq 1 \\ & \wedge (y_1 \leftrightarrow \overline{x_1} \ \wedge \ \overline{x_2} \wedge \overline{x_3}) \ \wedge \ (y_2 \leftrightarrow \overline{x_4} \wedge \overline{x_5} \wedge \overline{x_6}) \end{aligned}$$

After that, for each AMO constraint $x_{i_1} + \cdots + x_{i_l} \leq 1$, choose any integer I such that $I_i \geq -q_{i_j}$, for all $1 \leq j \leq l$. We have that $x_{i_1} + \cdots + x_{i_l} \leq 1 \wedge y_i \leftrightarrow \overline{x_{i_1}} \wedge \cdots \wedge \overline{x_{i_l}} \models y_i + x_{i_1} + \cdots + x_{i_l} = 1$, and therefore $I_i y_i + I_i x_{i_1} + \cdots + I_i x_{i_l} = I_i$. By adding these equalities to the PB constraint, all the negative coefficients become non-negative, due to the values of I_i that we have chosen. The size of the constraint will not increase if we choose $I = -\min_{i=1}^{n}(q_i)$, since we will be cancelling at least one coefficient of each AMO. In our example, we get:

$$\begin{aligned} & (4+0)y_1 + (4-1)x_1 + (4-3)x_2 + (4-4)x_3 \\ +(5+0)y_2 + (5-2)x_4 + (5-3)x_5 + (5-5)x_6 & \leq -6 + 4 + 5 \\ \wedge x_1 + x_2 + x_3 \leq 1 \ \wedge \ x_4 + x_5 + x_6 & \leq 1 \\ \wedge (y_1 \leftrightarrow \overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}) \ \wedge \ (y_2 \leftrightarrow \overline{x_4} \wedge \overline{x_5} \wedge \overline{x_6}) & \end{aligned}$$

$$\begin{aligned} \text{that is:} \qquad & 4y_1 + 3x_1 + x_2 + 5y_2 + 3x_4 + 2x_5 \leq 3 \\ & \wedge x_1 + x_2 + x_3 \leq 1 \ \wedge \ x_4 + x_5 + x_6 \leq 1 \\ & \wedge (y_1 \leftrightarrow \overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}) \ \wedge \ (y_2 \leftrightarrow \overline{x_4} \wedge \overline{x_5} \wedge \overline{x_6}) \end{aligned}$$

Finally, since $y_i + x_{i_1} + \cdots + x_{i_l} = 1 \models y_i + x_{i_1} + \cdots + x_{i_l} \leq 1$, we can use any of the presented PB(AMO) encodings, for instance GSWC:

$$GSWC(4y_1 + 3x_1 + x_2 + 5y_2 + 3x_4 + 2x_5 \leq 3, \{\{y_1, x_1, x_2\}, \{y_2, x_4, x_5\}\})$$
$$\wedge\, x_1 + x_2 + x_3 \leq 1 \;\wedge\; x_4 + x_5 + x_6 \leq 1$$
$$\wedge\, (y_1 \leftrightarrow \overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}) \;\wedge\; (y_2 \leftrightarrow \overline{x_4} \wedge \overline{x_5} \wedge \overline{x_6})$$

5 Experiments

In this section we report a clean comparison between the different encodings for PB(AMO) constraints, and also between those and the classical encodings for PB constraints. For this purpose, we use benchmark sets of problems consisting on conjunctions of AMO constraints and PB constraints. Each instance is defined by four parameters: L is the number of PB constraints, N is the number of AMO constraints, M is the number of Boolean variables in each AMO constraint, and Q is the maximum coefficient of a variable in a PB constraint. The variables of the AMO constraints will be disjoint, so there is a total of $n = N \cdot M$ Boolean variables in each instance. The PB constraints contain all n variables. The j-th variable in the i-it AMO constraint is named $x_{i,j}$. The coefficients in the PB constraints are generated uniformly and independently at random in the range $[1, Q]$. The resulting instance has the following constraints:

$$\sum_{i=1}^{N}\sum_{j=1}^{M} q_{i,j,k} \cdot x_{i,j} \leq K_k \qquad 1 \leq k \leq L \tag{18}$$

$$\sum_{j=1}^{M} x_{i,j} \leq 1 \qquad 1 \leq i \leq N \tag{19}$$

$$\sum_{j=1}^{M} x_{i,j} \geq 1 \qquad 1 \leq i \leq N \tag{20}$$

The conjunction of PB and AMO constraints (18) and (19) is not a hard problem, since a trivial solution is to set all the variables $x_{i,j}$ to 0. For this reason we add *at-least-one* Constraints (20), which require that at least one variable in each AMO group is set to true. Essentially, this set of constraints is a decision version of the Multi-Choice Multidimensional Knapsack Problem (MMKP), which is NP-complete. We have generated the benchmarks using the MMKP instance generator from [14].

We provide three different datasets with different parameters, with the aim of showing which kind of PB constraints are better suited for the different encodings. The instances in a dataset are distributed in families, and every family has values of K_k randomly distributed around a different mean in the range $[1, M \cdot Q]$. The values of K_k are proportional to the values of the coefficients, because otherwise the PB constraints would be trivially satisfied or unsatisfied. We choose different values of K_k to ensure that in the datasets there are instances of different hardness, and that approximately half the instances are satisfiable:

Set1 100 families of 5 instances, with $L = 10$, $N = 15$, $M = 10$, $Q = 1000$. The families have increasing K_k values from family 1 (capacities of about 1000) to family 100 (capacities of about 14000).

Set2 100 families of 5 instances, with $L = 10$, $N = 15$, $M = 10$, $Q = 60$. The families have increasing K_k values from family 1 (capacities of about 100) to family 100 (capacities of about 800).

Set3 20 families of 20 instances, with $L = 50$, $N = 15$, $M = 5$, $Q = 10$. The values of K_k increase in each family, ranging between 65 and 100.

All the instances have been encoded to SAT using the presented encodings for PB(AMO) constraints. The AMO Constraints (19) have been encoded with the well-known UP-maintaining GAC encoding referred as *regular* in [3] and *ladder* in [13], which only introduces a number of clauses and variables linear in the number of variables of the AMO constraint. Constraints (20) have been encoded with clauses $x_{i,1} \vee \cdots \vee x_{i,M}$, for all $1 \leq i \leq N$. We have used the Glucose 4.1 SAT solver [5] to solve the instances, on a 8GB, 3.10GHz Intel® Xeon® E3-1220v2.

The results are contained in Table 1. The evaluated encodings are the ones introduced in this paper, and their counterpart original ones. For completeness we also report results on the MDD-based encoding from [10] (MDD), and its version without taking AMOs into account (BDD), from [1]. Both MDD and BDD UP-maintain GAC. For each encoding we report solving times on each dataset and the average size required to encode a PB(AMO) constraint. The size results do not include the number of variables and clauses introduced by AMO constraints (19), which is the same for all encodings and negligible.

In summary, it can be observed a dramatic decrease in size, and hence in generation time, as well a significant decrease in solving time, in all the generalized encodings for PB(AMO)s w.r.t. the original encodings for PBs. In most of cases the size and solving time reduction is of one order of magnitude. Even in Set3, which is the one with smallest AMO constraints (only 5 variables per AMO) the reduction is notable. The GPW encoding is the smallest, and the one which is less reduced when using the AMO constraints.

In Set1, which contains instances with large coefficients, the best approach is GGPW. Although this encoding do not UP-maintain GAC, the number of clauses and variables is remarkably small compared to the other encodings, whose size is proportional to K. In particular this dataset is prohibitive for GT and GGT encodings, which require a number of clauses quadratic in the value of K.

Set2 is similar to Set1 but it contains instances with small coefficients. In this case, the best approaches are MDD and GSWC, whose sizes are reasonably smaller than the ones in Set1. The GGT encoding introduces the largest number of clauses in this dataset, and has the worst time performance.

Instances in Set3 contain more PB constraints than the other datasets, and the values of K are distributed around the transition value from unsatisfiable instances to satisfiable instances. We have observed empirically that it is in this transition where the instances become harder. In this case, GGPW has the worst time performance although it still has a smaller size than the other encodings. This may be because GGPW is the only one which does not UP-maintain GAC.

Table 1. From left to right: first quartile (Q1), median (med) and third quartile (Q3) of solving times (in seconds); average solving time in seconds (avg) counting time outs as 600 seconds; number of instances that timed out before being solved (t.o.); in thousands, average number of auxiliary variables (v.) and clauses (cl.) needed to encode one of the Constraints (18); in seconds, average time required to generate the CNF formula of an instance (g.t.). Solving time out (t.o.) is set to 600 seconds. Long dash (—) means that the resulting formulas are too large and their generation either run out of memory or did not finish in less than 600 seconds (in these instances we have been able to identify constraints requiring 33,000,000 clauses).

	enc.	Q1	med	Q3	avg	t.o.	v.	cl.	g.t.	enc.	Q1	med	Q3	avg	t.o.	v.	cl.	g.t.
Set1	BDD	14.00	17.59	t.o.	219	158	857	1714	35.6	MDD	3.89	14.78	73	131	87	25	266	3.71
	SWC	10.51	14.12	t.o.	199	144	1100	2177	17.2	GSWC	4.50	5.92	277	158	112	105	1076	10.01
	GT	—	—	—	—	—	—	—	—	GGT	—	—	—	—	—	—	—	—
	GPW	0.93	0.97	23	114	85	5.9	77	0.8	GGPW	0.04	0.04	5.54	93	67	1.0	4.4	0.05
Set2	BDD	4.29	5.65	133	141	96	57	115	2.0	MDD	0.21	0.41	1.42	74	53	2.1	21	0.28
	SWC	4.10	5.41	138	140	95	68	135	1.3	GSWC	0.58	0.62	1.09	71	52	6.4	66	0.62
	GT	5.33	6.94	182	154	110	10	1640	18.0	GGT	2.42	8.83	53	132	95	1.9	120	1.53
	GPW	0.46	0.48	11	108	77	3.5	42	0.4	GGPW	0.02	0.03	3.36	89	65	0.6	2.5	0.03
Set3	BDD	215	t.o.	t.o.	423	218	4.8	9.6	0.7	MDD	16.2	78.6	525	221	97	0.4	2.6	0.17
	SWC	247	t.o.	t.o.	429	227	6.0	12	0.6	GSWC	17.5	87.3	597	225	100	1.1	6.5	0.32
	GT	240	t.o.	t.o.	427	223	1.3	31	1.6	GGT	70.8	281	t.o.	322	152	0.4	4.6	0.25
	GPW	172	t.o.	t.o.	415	229	0.8	5.1	0.3	GGPW	133	t.o.	t.o.	407	226	0.3	1.2	0.07

6 Conclusions and Further Work

In this work we have provided different SAT encodings of PB(AMO)s, i.e., conjunctions of PB and AMO constraints. These new encodings have been defined by generalizing existing state-of-the-art SAT encodings of PB constraints, in a way that the size is highly reduced thanks to assuming that the AMO constraints are already enforced. Moreover, the propagation properties of the original encodings are preserved in the new ones. Our results show that all the new encodings are dramatically smaller and more efficient than their counterpart PB encodings. We have observed size reductions of an order of magnitude and solving time improvements of 1 or 2 orders of magnitude in many cases.

We have also shown that there is no best encoding for PB(AMO)s but it depends on the characteristics of the instances at hand. The datasets that we provide expose some strengths and weaknesses of the different encodings.

The PB(AMO) constraints have application in a large number of problems, and our new encodings should be taken into account as an alternative to tackle them. It is matter of a future work to see how this SAT-based approach compares with other solving approaches in different specific domains. Finally, other new encodings for PB(AMO)s might be studied, either based on existing ones or brand new ones.

Acknowledgements

Work supported by grants TIN2015-66293-R (MINECO/ FEDER, UE), and *Ayudas para Contratos Predoctorales 2016* (grant number BES-2016-076867, funded by MINECO and co-funded by FSE).

References

1. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E., Mayer-Eichberger, V.: A new look at BDDs for pseudo-Boolean constraints. Journal of Artificial Intelligence Research **45**, 443–480 (2012)
2. Achterberg, T., Bixby, R.E., Gu, Z., Rothberg, E., Weninger, D.: Presolve reductions in mixed integer programming. Zuse Institute Berlin (2016)
3. Ansótegui, C., Manyà, F.: Mapping Problems with Finite-Domain Variables to Problems with Boolean Variables. In: Theory and Applications of Satisfiability Testing - SAT 2004, 7th International Conference. LNCS, vol. 3542, pp. 1–15. Springer (2005)
4. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality Networks: a theoretical and empirical study. Constraints **16**(2), 195–221 (2011). https://doi.org/10.1007/s10601-010-9105-0
5. Audemard, G., Simon, L.: On the glucose SAT solver. International Journal on Artificial Intelligence Tools **27**(1), 1–25 (2018)
6. Bailleux, O., Boufkhad, Y.: Efficient CNF Encoding of Boolean Cardinality Constraints. In: Principles and Practice of Constraint Programming - CP 2003, 9th International Conference. LNCS, vol. 2833, pp. 108–122. Springer (2003)
7. Bailleux, O., Boufkhad, Y., Roussel, O.: New Encodings of Pseudo-Boolean Constraints into CNF. In: Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference. LNCS, vol. 5584, pp. 181–194. Springer (2009)
8. Basnet, C., Wilson, J.: Heuristics for determining the number of warehouses for storing non-compatible products. International transactions in operational research **12**(5), 527–538 (2005)
9. Bofill, M., Coll, J., Suy, J., Villaret, M.: An Efficient SMT Approach to Solve MRCPSP/max Instances with Tight Constraints on Resources. In: Principles and Practice of Constraint Programming - CP 2017, 23rd International Conference. LNCS, vol. 10416, pp. 71–79. Springer (2017)
10. Bofill, M., Coll, J., Suy, J., Villaret, M.: 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 2017. pp. 555–562. ijcai.org (2017)
11. De Vries, S., Vohra, R.V.: Combinatorial auctions: A survey. INFORMS Journal on computing **15**(3), 284–309 (2003)
12. Eén, N., Sorensson, N.: Translating pseudo-Boolean constraints into SAT. Journal on Satisfiability, Boolean Modeling and Computation **2**, 1–26 (2006)
13. Gent, I.P., Nightingale, P.: A new encoding of alldifferent into sat. In: 3rd International Workshop on Modelling and Reformulating Constraint Satisfaction - CP 2004. pp. 95–110 (2004)
14. Han, B., Leblet, J., Simon, G.: Hard multidimensional multiple choice knapsack problems, an empirical study. Computers & Operations Research **37**(1), 172 – 181 (2010)
15. Hölldobler, S., Manthey, N., Steinke, P.: A compact encoding of pseudo-Boolean constraints into SAT. In: KI 2012: Advances in Artificial Intelligence - 35th Annual German Conference on Artificial Intelligence. LNCS, vol. 7526, pp. 107–118. Springer (2012)
16. Joshi, S., Martins, R., Manquinho, V.M.: Generalized Totalizer Encoding for Pseudo-Boolean Constraints. In: Principles and Practice of Constraint Programming - CP 2015, 21st International Conference. LNCS, vol. 9255, pp. 200–209. Springer (2015)

17. Ma, P.R., Lee, E.Y.S., Tsuchiya, M.: A Task Allocation Model for Distributed Computing Systems. IEEE Trans. Computers **31**(1), 41–47 (1982)
18. Manthey, N., Philipp, T., Steinke, P.: A More Compact Translation of Pseudo-Boolean Constraints into CNF Such That Generalized Arc Consistency Is Maintained. In: KI 2014: Advances in Artificial Intelligence - 37th Annual German Conference on AI. LNCS, vol. 8736, pp. 123–134. Springer (2014)
19. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. Journal of the ACM **7**(4), 326–329 (1960)
20. Philipp, T., Steinke, P.: PBLib - A Library for Encoding Pseudo-Boolean Constraints into CNF. In: Theory and Applications of Satisfiability Testing - SAT 2015, 18th International Conference. LNCS, vol. 9340, pp. 9–16. Springer (2015)
21. Pisinger, D.: Budgeting with bounded multiple-choice constraints. European Journal of Operational Research **129**(3), 471–480 (2001)
22. Watson, R.: Packet networks and optimal admission and upgrade of service level agreements: applying the utility model. MA Sc. Ph.D. thesis, Department of ECE, University of Victoria (2001)