SAT Encodings for Pseudo-Boolean Constraints Together With At-Most-One Constraints (Extended Abstract)*

Miquel Bofill[1], **Jordi Coll**[2], **Peter Nightingale**[3], **Josep Suy**[1], **Felix Ulrich-Oltean**[3] and **Mateu Villaret**[1]

[1]Universitat de Girona, Girona, Spain
[2]Institut d'Investigació en Intel·ligència Artificial (IIIA-CSIC), Bellaterra, Spain
[3]University of York, York, United Kingdom
miquel.bofill@udg.edu, jcoll@iiia.csic.es, peter.nightingale@york.ac.uk, josep.suy@udg.edu, felix.ulrich-oltean@york.ac.uk, mateu.villaret@udg.edu

Abstract

When solving a combinatorial problem using propositional satisfiability (SAT), the encoding of the constraints is of vital importance. Pseudo-Boolean (PB) constraints appear frequently in a wide variety of problems. When PB constraints occur together with at-most-one (AMO) constraints over the same variables, they can be combined into PB(AMO) constraints. In this paper we present new encodings for PB(AMO) constraints. Our experiments show that these encodings can be substantially smaller than those of PB constraints and allow many more instances to be solved within a time limit. We also observed that there is no single overall winner among the considered encodings, but efficiency of each encoding may depend on PB(AMO) characteristics such as the magnitude of coefficient values.

1 Introduction

Discrete decision-making problems crop up in many contexts in the modern world. Such problems can be expressed as constraint satisfaction (or optimisation) problems (CSPs or COPs), then solved using a variety of solver types. An increasingly popular and successful approach to solving CSPs and COPs is to encode them into Boolean formulas and then to apply an off-the-shelf SAT solver. This approach is attractive because of the power of modern conflict-directed clause learning (CDCL) SAT solvers, which incorporate conflict learning, powerful search heuristics, and fast propagation of the Boolean constraints.

Linear equations and inequalities are ubiquitous in constraint problems such as scheduling, routing, resource allocation, and many other hard combinatorial problems. Pseudo-Boolean (PB) constraints are a particular type of linear constraint of the following form, where $q_1, \ldots, q_n$ and K are integer constants, and $x_1, \ldots, x_n$ are 0/1 variables:

$$\sum_{i=1}^{n} q_i x_i \ \# \ K \quad \text{where} \quad \# \in \{<, \leq, =, \geq, >\}$$

There has been a great deal of work on encoding PB constraints to SAT, some of which is reviewed by Philipp and Steinke [2015]. State-of-the-art encodings are based on Binary Decision Diagrams [Eén and Sorensson, 2006; Abío *et al.*, 2012], Sequential Weight Counters [Hölldobler *et al.*, 2012], Generalized Totalizers [Joshi *et al.*, 2015; Zha *et al.*, 2019], and Polynomial Watchdog schemes [Bailleux *et al.*, 2009; Manthey *et al.*, 2014]. At-most-one (AMO) constraints (i.e. constraints of the form $\sum_{i=1}^{m} x_i \leq 1$) are also very common, with the most basic being a mutual exclusion between two 0/1 variables.

Combinations of PB and AMO constraints usually appear in settings where one option has to be chosen among a set of incompatible options, and the decision has an associated cost. Bofill, Coll, Suy, and Villaret [Bofill *et al.*, 2017b; Bofill *et al.*, 2020] proposed a SAT encoding based on Multi-valued Decision Diagrams (MDDs) for a conjunction of a PB constraint with a set of AMO constraints over the variables of the PB constraint. Such conjunctions are referred to as PB(AMO) constraints. The AMO constraints (which are encoded separately and may be encoded to SAT using any AMO encoding) allow certain interpretations to be erased from decision diagrams, and to represent the PB constraint as an MDD instead of as a Binary Decision Diagram (BDD). The encoding of the MDD is notably smaller than the encoding of an equivalent BDD, and the solving time is substantially reduced. This technique has been used to provide efficient formulations of particular kinds of scheduling problems [Bofill *et al.*, 2017b; Bofill *et al.*, 2017a]. Also, Ansótegui *et al.* [2019] integrated the MDD-based SAT encoding of PB(AMO) constraints into the automatic reformulation pipeline of Savile Row [Nightingale *et al.*, 2017], showing important size and solving time improvements compared to a BDD-based encoding oblivious to the existence of AMO constraints.

This work is an extended abstract of Bofill *et al.* [2022], whose main contribution is to generalize five state-of-the-art SAT encodings of PB constraints to encode PB(AMO) constraints, as well as the introduction of new optimizations for

*This is an extended abstract of the paper [Bofill *et al.*, 2022], that was published in the Artificial Intelligence journal in 2022.

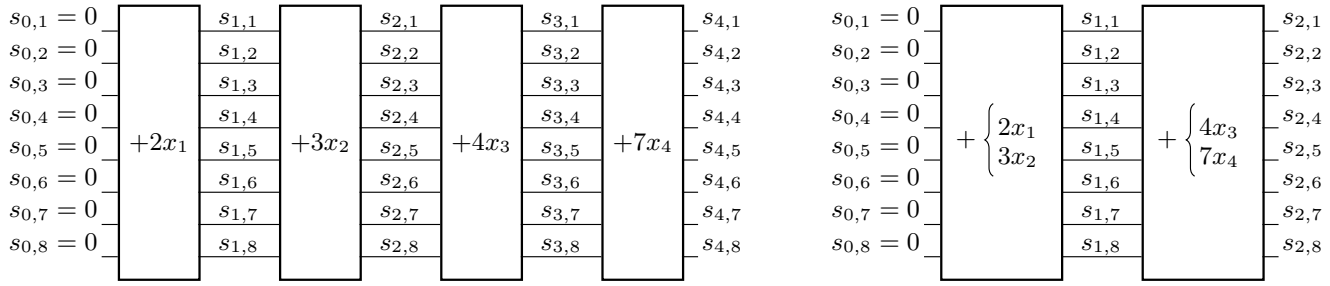

Figure 1: Sequential Weight Counter representation of $2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 8$ (left) and of $(2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 8, \{\{x_1, x_2\}, \{x_3, x_4\}\})$ (right).

already existing encodings of PB constraints. In this extended abstract we provide a summary of the main idea behind the generalizations (Section 2), an overview of the new encodings (Sections 3, 4 and 5), and we briefly comment on the results of our experimental evaluation (Section 6).

In this extended abstract we just provide schematic representations of the developed encodings as well as a short description of the encoding idea, so that the reader can visually appreciate the size reductions achieved thanks to using PB(AMO) encodings. The original work contains many other details, including: normalization and reduction techniques for PB(AMO) constraints; detailed explanation of the existing PB encodings and the introduced ones; formal definitions of the clauses introduced by the different encodings; detailed examples for all encodings; proofs of correctness, sizes and propagation properties of all encodings; new heuristics to build unbalanced totalizers that generate smaller formulas; and extensive experiments with two representative SAT solvers on existing and new benchmarks for different problem classses, namely: the Nurse Scheduling problem, the Combinatorial Auctions problem, the Multimode Resource-Constrained Project Scheduling Problem, the Resource-Constrained Project Scheduling Problem with Time-Dependent Resource Capacities and Requests; and the Multi-choice Multidimensional Knapsack Problem.

2 Encoding Technique

There exist a myriad of techniques to encode AMO and PB constraints to SAT. Therefore, the straightforward approach for a problem containing both kinds of constraints is to encode each of them separately with some existing technique, and join the resulting sets of clauses. However, when the variables of a PB constraint also occur in AMO constraints, we can do much better. As a motivating example, consider the PB constraint:

$$2x_1 + 3x_2 + 4x_3 + 2x_4 + 3x_5 + 4x_6 \leq 7$$

Also, suppose there are two AMO constraints: $x_1{+}x_2{+}x_3{\leq}1$ and $x_4{+}x_5{+}x_6{\leq}1$. Encoding the PB constraint alone would require several clauses and (depending on the chosen encoding) multiple additional variables. For example, the Generalized Totalizer encoding [Joshi *et al.*, 2015] has 23 additional variables and 56 clauses. However, the two AMO constraints rule out most of the values that the sum $(2x_1{+}\cdots{+}4x_6)$ could take, and almost all such values that break the PB constraint. Encoding the PB constraint together with the two AMO constraints requires just one clause to prevent x_3 and x_6 being assigned true together. This simple observation underpins all the PB(AMO) encodings presented in this paper.

In our encodings, we consider PB constraints such that all their variables also belong to some AMO constraint. Notice that this is not a limitation in the sense that any single variable x can constitute itself an AMO constraint $x \leq 1$ if no other AMO constraint contains it. Moreover, we require the AMO constraints to be disjoint. Again, this is not a limitation, since we can always remove variables from AMO constraints to achieve this property, e.g. if we have the constraint $x_1{+}x_2{+}x_3{\leq}1$, implicitly we also have the constraint $x_1{+}x_2{\leq}1$. We refer to this kind of conjunction between one PB constraint and a set of AMO constraints as a PB(AMO) constraint.

The encoding technique we employ for PB(AMO) constraints works as follows. First, we will encode all AMO constraints with some existing technique, thus obtaining a set of clauses $F1$. Then, instead of constructing a full encoding of the PB constraint, we construct a simplified version assuming that the AMO constraints will hold, obtaining a set of clauses $F2$. The resulting encoding of the PB(AMO) constraint is $F_1 \cup F_2$. In order to take into account the AMO constraints in the second step, we consider the PB constraint and a partition of its variables. In particular, the variables of each part corresponds to the variables of a different AMO constraint. For instance, in the motivating example given before, we consider the tuple:

$$(2x_1 + 3x_2 + 4x_3 + 2x_4 + 3x_5 + 4x_6 \leq 7, \\ \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\}\})$$

One encoding of the example would be $F_1 = \{(\overline{x_1} \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_3}), (\overline{x_2} \vee \overline{x_3}), (\overline{x_4} \vee \overline{x_5}), (\overline{x_4} \vee \overline{x_6}), (\overline{x_5} \vee \overline{x_6})\}$, and $F_2 = \{(\overline{x_3} \vee \overline{x_6})\}$.

The generation of F_1 can be easily achieved with any existing encoding technique for AMO constraints. In this work we present many encoding techniques for F_2, taking as input the PB constraint and variable partition.

3 Sequential Weight Counter Encoding

The Sequential Weight Counter encoding for PB constraints was defined in [Hölldobler *et al.*, 2012], and is illustrated

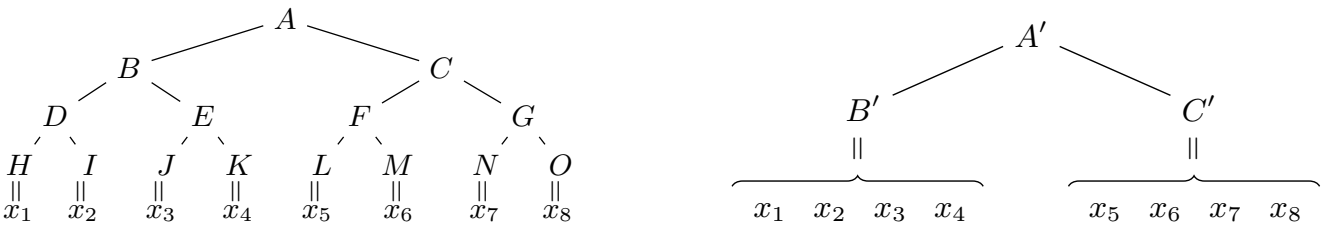

Figure 2: Totalizer representation of $2x_1 + 3x_2 + 4x_3 + 5x_4 + 3x_5 + 4x_6 + 6x_7 + 8x_8 \leq 10$ (left) and, of $(2x_1 + 3x_2 + 4x_3 + 5x_4 + 3x_5 + 4x_6 + 6x_7 + 8x_8 \leq 10, \{\{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7, x_8\}\})$ (right).

by Figure 1 (left). The resulting CNF formula simulates a circuit that sequentially sums the coefficients of the variables assigned 1. For each variable, we include a component (counter) that increases (if needed) the accumulated sum value. Finally, the clauses of the encoding also require that the sum never exceeds the maximum allowed by the PB constraint.

Taking into account that some subsets of variables satisfy an AMO constraint, we can reduce the number of counters. In particular, all the variables of a same part of the partition (AMO constraint) share the same counter. As illustrated by Figure 1 (right), this effectively decreases the number of counters, and therefore the number of output pins (auxiliary variables) and the number of clauses.

4 Totalizer-Based Encodings

A totalizer is a binary tree where the leaf nodes represent items to be summed, and each non-leaf node represents the sum of its two children. An encoding of PB constraints based on totalizers was presented by Joshi *et al.* [2015] and named Generalized Totalizer (GT). In GT, each leaf node is attached to a single PB term $q_i x_i$, and takes the value q_i when x_i is *true* and 0 otherwise. An example of a totalizer representing a PB constraint can be seen in Figure 2 (left). The SAT encoding consists of a CNF formula simulating the totalizer as well as a clause forbidding that the maximum allowed sum is exceeded at the root node. Note that each node of a totalizer represents an integer value. In the GT encoding, these values are (roughly) represented as an order encoding, i.e., for each value $v \in 0, \ldots, k$, that a node O can take, we introduce the variable $o_{\geq v}$. The encoding is optimized by only introducing auxiliary variables $o_{\geq v}$ for values v that are reachable by adding the two children.

Zha *et al.* [2019] presented an alternative totalizer-based encoding for PB constraints, named n-level MTO (Modulo Totalizer). In the MTO, the value of the nodes is represented with a decomposition into multiple digits in a mixed radix base. For instance, in a decimal decomposition, we would introduce variables representing the value of the units, variables for the tens, the hundreds, the thousands, etc. This typically results in much smaller formulas but with weaker propagation properties.

In our work we present generalizations of both GT and MTO to encode PB(AMO) constraints. The overall idea of the generalization is the same in both cases and illustrated in Figure 2 (right). Here, instead of associating only one variable to a leaf node, we associate all the variables of an AMO. However, the way of achieving this association is different for GT and for MTO (more details can be found in the full paper). By grouping the variables of the AMO constraints, we can reduce the number of leaves of the totalizer which in turn will reduce the number of clauses and auxiliary variables of the encoding.

Another contribution of our work is a new totalizer-based encoding, which is novel not only for the PB(AMO) case but also for the classical PB setting. We call this encoding the Reduced Generalized Totalizer (RGT). The starting point of RGT is the tree structure of the GT encoding. Once the tree is constructed, it applies a reduction algorithm in order to obtain a more compact representation that replaces individual numeric values with intervals. In some cases, RGT will detect that terms in the PB constraint are redundant. When this occurs the redundant terms are removed and the entire encoding process is repeated (until a fixpoint is reached). RGT is frequently substantially smaller than GT and this translates to improved solver efficiency.

5 Watchdog-Based Encodings

Two different PB encodings were presented in [Bailleux *et al.*, 2009], namely the Global Polynomial Watchdog (GPW) and the Local Polynomial Watchdog (LPW). Both of them use as a basis a so-called polynomial watchdog circuit, represented in Figure 3 (left). GPW uses just one polynomial watchdog, while LPW introduces one polynomial watchdog for each variable, and the practical difference is that the former results in smaller formulas but with weaker propagation properties. This kind of circuit decomposes the coefficients of the variables in a binary representation. The variables of every bit are added with sorter circuits, and the different bits are later added in merger circuits. Whenever the maximum value of the PB is surpassed, the so-called watchdog pin (variable) w is set to true.

The basic idea of our PB(AMO) generalization, illustrated in Figure 3 (right), is to represent the j-th bit of all variables of the i-th AMO with a single auxiliary variable $y_{i,j}$. This can reduce the number of inputs of the sorters, and by extension reduce the number of variables of clauses not only to encode the sorters but also the mergers, as can be appreciated with the decrease of pins in Figure 3 (right).

Moreover, in the original work we present many formula reduction techniques for watchdog-based encodings, especially for LPW, based on reusing pins and components of the

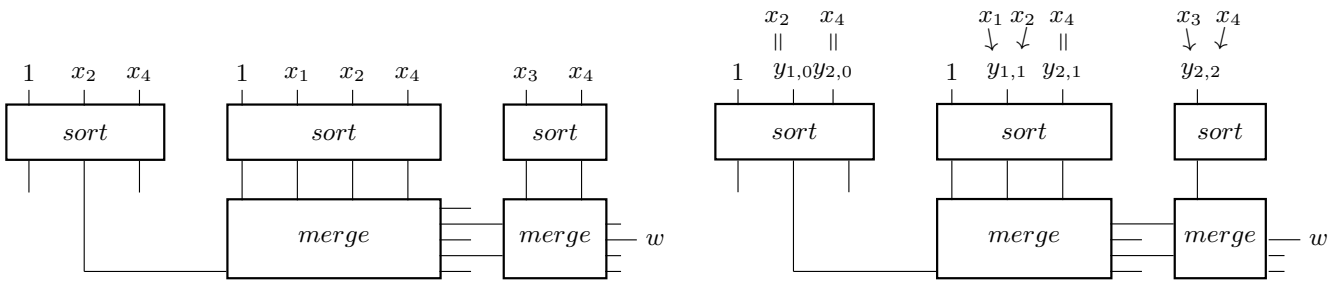

Figure 3: Polynomial Watchdog representation of $2x_1+3x_2+4x_3+7x_4 \leq 8$ (left) and of $(2x_1+3x_2+4x_3+7x_4 \leq 8, \{\{x_1, x_2\}, \{x_3, x_4\}\})$ (right).

circuits. These size reductions are especially suitable in the PB(AMO) case, where the assumption of AMO constraints allows for extra circuit reuses.

6 Results

As proved in the original work, the newly introduced encodings for PB(AMO) constraints maintain the same propagation strength by unit-propagation that their PB counterparts have. We also achieve theoretical size reduction of the encodings, both in terms of number of auxiliary variables and number of clauses. This is certified by our experimental section, where we can identify size reductions of up to three orders of magnitude. This reduction in size has a clear impact on the solving times, which are drastically reduced with the PB(AMO) approach, based on experiments conducted with two recent CDCL SAT solvers. Compared to their PB counterparts, PB(AMO) encodings allow many more instances to be solved within a time limit, and solving time is improved by more than one order of magnitude in some cases. Similarly, the new totalizer-based encoding and the presented totalizer-building heuristic also generate smaller encodings and improve the solving times.

In Table 1 we show a few selected experimental results from the full paper, with five encodings chosen from the eight encodings in the full paper, with only one problem class, and with only one SAT solver. The selected results show very clear benefits from using PB(AMO) encodings on that problem class, with a very substantial reduction in the number of time-out instances.

We also show empirically the usefulness of having distinct encodings with different properties, usually with trade-offs between number of variables, number of clauses and propagation strength, since the best choice depends on the tackled problem class.

PB			PB(AMO)		
Enc.	Med. time	t.o.	Enc.	Med. time	t.o.
BDD	4.87	117	MDD	1.03	41
SWC	8.44	162	GSWC	1.70	41
GTd	3.33	75	GGTd	1.79	35
RGT	4.56	154	RGGT	1.28	38
GPW	7.91	187	GGPW	1.10	26

Table 1: Solving times and number of timeouts using CaDiCaL with the RCPSP/Tj120 benchmarks. Median time (in seconds) and number of time-out instances (over 600 seconds) are reported for five selected PB encodings (left) and their PB(AMO) equivalents (right), showing that many more instances are solved with PB(AMO) encodings and that the solving time is improved.

Acknowledgments

This work has been partially supported by Grants PID2019-111544GB-C21, TED2021-129319B-I00 and PID2021-122274OB-I00 funded by MCIN/AEI /10.13039/501100011033 and by ERDF A way of making Europe, and by Grant EP/R513386/1 from the UK Engineering and Physical Sciences Research Council.

References

[Abío *et al.*, 2012] Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Valentin Mayer-Eichberger. A new look at BDDs for pseudo-Boolean constraints. *Journal of Artificial Intelligence Research*, 45:443–480, 2012.

[Ansótegui *et al.*, 2019] Carlos Ansótegui, Miquel Bofill, Jordi Coll, Nguyen Dang, Juan Luis Esteban, Ian Miguel, Peter Nightingale, András Z. Salamon, Josep Suy, and Mateu Villaret. Automatic detection of at-most-one and exactly-one relations for improved SAT encodings of pseudo-Boolean constraints. In *Proceedings of 25th International Conference Principles and Practice of Constraint Programming - CP*, volume 11802 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2019.

[Bailleux *et al.*, 2009] Olivier Bailleux, Yacine Boufkhad, and Olivier Roussel. New Encodings of Pseudo-Boolean Constraints into CNF. In *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference*, volume 5584 of *LNCS*, pages 181–194. Springer, 2009.

[Bofill *et al.*, 2017a] Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. 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*, volume 10416 of *LNCS*, pages 71–79. Springer, 2017.

[Bofill *et al.*, 2017b] Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. 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*, pages 555–562. ijcai.org, 2017.

[Bofill *et al.*, 2020] Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. An mdd-based sat encoding for pseudo-boolean constraints with at-most-one relations. *Artificial Intelligence Review*, pages 1–32, 2020.

[Bofill *et al.*, 2022] Miquel Bofill, Jordi Coll, Peter Nightingale, Josep Suy, Felix Ulrich-Oltean, and Mateu Villaret. SAT encodings for pseudo-boolean constraints together with at-most-one constraints. *Artif. Intell.*, 302:103604, 2022.

[Eén and Sorensson, 2006] Niklas Eén and Niklas Sorensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.

[Hölldobler *et al.*, 2012] Steffen Hölldobler, Norbert Manthey, and Peter Steinke. A compact encoding of pseudo-Boolean constraints into SAT. In *KI 2012: Advances in Artificial Intelligence - 35th Annual German Conference on Artificial Intelligence*, volume 7526 of *LNCS*, pages 107–118. Springer, 2012.

[Joshi *et al.*, 2015] Saurabh Joshi, Ruben Martins, and Vasco M. Manquinho. Generalized Totalizer Encoding for Pseudo-Boolean Constraints. In *Principles and Practice of Constraint Programming - CP 2015, 21st International Conference*, volume 9255 of *LNCS*, pages 200–209. Springer, 2015.

[Manthey *et al.*, 2014] Norbert Manthey, Tobias Philipp, and Peter Steinke. 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*, volume 8736 of *LNCS*, pages 123–134. Springer, 2014.

[Nightingale *et al.*, 2017] Peter Nightingale, Özgür Akgün, Ian P. Gent, Christopher Jefferson, Ian Miguel, and Patrick Spracklen. Automatically improving constraint models in Savile Row. *Artificial Intelligence*, 251:35–61, 2017.

[Philipp and Steinke, 2015] Tobias Philipp and Peter Steinke. PBLib - A Library for Encoding Pseudo-Boolean Constraints into CNF. In *Theory and Applications of Satisfiability Testing - SAT 2015, 18th International Conference*, volume 9340 of *LNCS*, pages 9–16. Springer, 2015.

[Zha *et al.*, 2019] Aolong Zha, Miyuki Koshimura, and Hiroshi Fujita. N-level modulo-based CNF encodings of pseudo-Boolean constraints for MaxSAT. *Constraints*, 24(2):133–161, 2019.