Enhancing Pigeon-Hole based Encoding of Boolean Cardinality Constraints

Soukaina Hattad, Said Jabbour, Lakhdar Sais and Yakoub Salhi
CRIL - CNRS UMR 8188, University of Artois, Lens, France

Keywords: Satisfiability, Linear inequalities, Cardinality Constraints.

Abstract: In this paper, we propose to deal with the encoding of cardinality constraints $\sum_{i=1}^{n} x_i \geqslant b$ into conjunctive normal form. We consider the one proposed recently (Jabbour et al., 2014) based on pigeon-hole problem. Then, we show that even if the number of clauses of the CNF based encoding is in $O(b \times (n-b))$, the number of literals of resulting formula can be much more higher: $O(b(n-b)^2)$. To decrease the complexity in terms of number of literals, we propose a compact representation of some clauses of the encoding. Our approach allows to have a quadratic encoding in terms of literals while maintaining the same complexity in terms of clauses and additional variables. An experimental evaluation is performed to show the competitiveness of the new encoding.

1 INTRODUCTION

Today, Boolean satisfiability (SAT) has gained a considerable audience with the advent of a new generation of solvers able to solve large instances encoding real-world problems. In addition to the traditional applications of SAT to hardware and software formal verification, this impressive progress led to increasing use of SAT technology to solve new real-world applications such as planning, bioinformatics, cryptography, and data mining. Encoding applications as formulas in CNF became now a usual practice. One of the most important flaws of CNF or Boolean representation in general rises in the difficulty to deal with counting constraints, among them the cardinality constraint and its more general form the pseudo Boolean constraint. Indeed, several applications involve counting arguments expressed as cardinality or pseudo Boolean constraint. This kind of constraints arises frequently out of the encoding of real-world problems such as radio frequency assignment, time tabling and product configuration problems to cite a few. For the above reasons, several authors have addressed the issue of finding an efficient encoding of cardinality (e.g. (Warners, 1996), (Bailleux and Boufkhad, 2003), (Sinz, 2005), (Silva and Lynce, 2007), (Asín et al., 2009)) and pseudo Boolean constraints (e.g. (Eén and Sörensson, 2006; Bailleux et al., 2009)) as a CNF formula. Efficiency refers to both the compactness of the representation (size of the CNF formula) and to the ability to achieve the same level of constraint propagation (generalized arc consistency) on the CNF formula.

In this paper, we present an enhancement of the pigeon-hole based encoding of the cardinality constraint into CNF. We provide a new encoding allowing a compact representation leading to a reduction in terms of the number of literals of the original one.

The rest of this paper is organized as follows. After some preliminary definitions and technical background, we recall the Pigeon-Hole based CNF encoding of the cardinality constraint. Then, we present our approach to enhance this encoding by providing a more compact representation in terms of number of literals. An experimental evaluation on Partial MaxSAT instances is performed to demonstrate the competitiveness of our proposal. We conclude with some interesting and general perspectives.

2 TECHNICAL BACKGROUND AND PRELIMINARY DEFINITIONS

2.1 Preliminary Definitions and Notations

A Boolean formula $\mathcal{F}$ in *Conjunctive Normal Form (CNF)* is a conjunction of *clauses*, where a clause is

Hattad S., Jabbour S., Sais L. and Salhi Y.
Enhancing Pigeon-Hole based Encoding of Boolean Cardinality Constraints.
DOI: 10.5220/0006252502990307
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 299-307
ISBN: 978-989-758-220-2

a disjunction of *literals*. A literal is a positive (x) or negated ($\neg x$) propositional variable. The two literals x and $\neg x$ are called *complementary*. We denote by $\tilde{l}$ the complementary literal of l. More precisely, if $l = x$ then $\tilde{l} = \neg x$, otherwise $\tilde{l} = x$. The variable associated to a literal l is denoted by $|l|$. Let us recall that any Boolean formula can be translated to CNF using linear Tseitin's encoding (Tseitin, 1968). The size of the CNF $\mathcal{F}$ is defined as $\sum_{c\in\mathcal{F}} |c|$, where $|c|$ is the number of literals in c. A *unit* clause is a clause containing only one literal (called *unit literal*), while a binary clause contains exactly two literals. A Horn (resp. reverse Horn) clause is a clause with at-most one positive (resp. negative) literal. A positive (resp. negative) clause is a clause whose literals are all positive (resp. negative). An *empty clause*, denoted $\bot$, is interpreted as false (unsatisfiable), whereas an *empty CNF formula*, denoted $\top$, is interpreted as true (satisfiable).

The set of variables occurring in $\mathcal{F}$ is denoted $V_\mathcal{F}$ and its associated set of literals $\mathcal{L}_\mathcal{F} = \cup_{x\in\mathcal{V}_\mathcal{F}} \{x, \neg x\}$. A set of literals is *complete* if it contains one literal for each variable in $V_\mathcal{F}$, and *fundamental* if it does not contain complementary literals. A literal l is called *monotone or pure* if $\tilde{l}$ does not appear in $\mathcal{F}$. An *interpretation* ρ of a Boolean formula $\mathcal{F}$ is a function which associates a truth value $\rho(x) \in \{0,1\}$ (0 for false and 1 for *true*) to some of the variables $x \in V_\mathcal{F}$. ρ is *complete* if it assigns a value to every $x \in V_\mathcal{F}$, and *partial* otherwise. An interpretation is alternatively represented by a complete and fundamental set of literals. A *model* of a formula $\mathcal{F}$ is an interpretation ρ that satisfies the formula, denoted $\rho \models \mathcal{F}$. A formula $\mathcal{G}$ is a logical consequence of a formula $\mathcal{F}$, denoted $\mathcal{F} \models \mathcal{G}$, iff every model of $\mathcal{F}$ is a model of $\mathcal{G}$.

Let c_i and c_j be two clauses such that $c_i = (x \vee \alpha)$ and $c_j = (\neg x \vee \beta)$, $\eta[x, c_i, c_j] = (\alpha \vee \beta)$ denotes the *resolvent* on x between c_i and c_j. A resolvent is called *tautological* when it contains complementary literals.

$\mathcal{F}|_x$ denotes the formula obtained from $\mathcal{F}$ by assigning x the truth-value *true*. Formally $\mathcal{F}|_x = \{c \mid c \in \mathcal{F}, \{x, \neg x\} \cap c = \emptyset\} \cup \{c \backslash \{\neg x\} \mid c \in \mathcal{F}, \neg x \in c\}$. This notation is extended to interpretations: given an interpretation $\rho = \{x_1, \ldots, x_n\}$, we define $\mathcal{F}|_\rho = (\ldots((\mathcal{F}|_{x_1})|_{x_2})\ldots|_{x_n})$.

$\mathcal{F}^*$ denotes the formula $\mathcal{F}$ closed under unit propagation, defined recursively as follows: (1) $\mathcal{F}^* = \mathcal{F}$ if $\mathcal{F}$ does not contain any unit clause, (2) $\mathcal{F}^* = \bot$ if $\mathcal{F}$ contains two unit-clauses $\{x\}$ and $\{\neg x\}$, (3) otherwise, $\mathcal{F}^* = (\mathcal{F}|_x)^*$ where x is the literal appearing in a unit clause of $\mathcal{F}$.

Let c_1 and c_2 be two clauses of a formula $\mathcal{F}$. We say that c_1 (respectively c_2) subsume (resp. is subsumed) c_2 (resp. by c_1) iff $c_1 \subseteq c_2$. If c_1 subsume c_2, then $c_1 \models c_2$ (the converse is not true).

Let $c \in \mathcal{F}$ such that $x \in c$, the literal x of c is called *blocked* if $\forall c' \in \mathcal{F}$ such that $\neg x \in c'$ and $c \neq c'$, $\eta[x, c, c']$ is a tautology. A clause $c \in \mathcal{F}$ is a *blocked clause* if it contains a blocked literal (Kullmann, 1997). A blocked clause $c \in \mathcal{F}$ can be deleted from $\mathcal{F}$ while preserving satisfiability.

2.2 Pigeon-Hole Principle

The pigeon-hole based encoding is based on the Pigeon-Hole principle widely used in proof complexity. It asserts that there is no injective mapping from b pigeons to n holes as long as $b > n$. Stephen A. Cook proved that the propositional formula encoding the Pigeon-Hole problem have polynomial size proof in extended resolution proof system (Cook, 1976). A polynomial proof is also obtained by Krishnamurthy (Krishnamurthy, 1985) using resolution with symmetry. The Pigeon-Hole principle PHP_n^b can be expressed as a propositional formula in conjunctive normal form. The variables of PHP_n^b are p_{ij} with $1 \leqslant i \leqslant b$, $1 \leqslant j \leqslant n$; the variable p_{ij} is intended to denote the condition that pigeon i is sitting in hole j. The CNF formula encoding PHP_n^b can be stated as follows:

$$\bigvee_{j=1}^{n} p_{ij}, \quad 1 \leqslant i \leqslant b \tag{1}$$

$$\bigwedge_{1\leqslant i<k\leqslant b} (\neg p_{ij} \vee \neg p_{kj}), \quad 1 \leqslant j \leqslant n \tag{2}$$

The first equation (1) expresses that any pigeon must be put in at least one hole, while the equation (2) constrains each hole to contain at most one pigeon.

2.3 Symmetries in SAT

As the Pigeon-Hole based encoding heavily exploit symmetries (Krishnamurthy, 1985), we briefly recall the symmetry breaking framework in SAT. For more details on symmetry, we refer the reader to some but not exhaustive list of works in SAT (Benhamou and Sais, 1992), (Benhamou and Sais, 1994), (Crawford et al., 1996; Aloul et al., 2003) and CSP (Puget, 1993; Gent et al., 2006).

First, let us introduce some definitions on group theory. A group $(\mathcal{G}, \circ)$ is a finite set $\mathcal{G}$ with an associative binary operation $\circ : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ admitting a neutral and an inverse element. The set of all permutation $\mathcal{P}$ over a finite set E associated to the composition operator $\circ$, denoted $(\mathcal{P}, \circ)$, forms a group. Furthermore, each permutation $\sigma \in \mathcal{P}$ can be represented by a set of cycles $\{c_1 \ldots c_n\}$ where each cycle c_i is a list of

elements of $E\ (l_{i_1} \dots l_{i_{n_i}})$ s.t. $\forall 1 \leqslant k < n_i, \sigma(l_{i_k}) = l_{i_{k+1}}$ and $\sigma(l_{i_{n_i}}) = l_{i_1}$.

Let $\mathcal{F}$ be a CNF formula, and σ a permutation over $\mathcal{L}(\mathcal{F})$. We can extend the definition of the permutation σ to $\mathcal{F}$ as follows: $\sigma(\mathcal{F}) = \{\sigma(c) | c \in \mathcal{F}\}$ and $\sigma(c) = \{\sigma(l) | l \in c\}$.

Definition 1. *Let $\mathcal{F}$ be a CNF formula and σ a permutation over the literals of $\mathcal{F}$, σ is a symmetry of $\mathcal{F}$ if it satisfies the following conditions:*

- $\sigma(\neg x) = \neg\sigma(x), \forall x \in \mathcal{L}_{\mathcal{F}}$
- $\sigma(\mathcal{F}) = \mathcal{F}$

From the definition above, a symmetry σ defines an equivalence relation over the set of possible assignments. We need to consider only one assignment from each equivalence class. Breaking symmetries consist in eliminating all symmetric assignments except one in each equivalence class. The most used approach to break symmetries consists in adding new clauses - called symmetry breaking predicates (SBP) or lex leader constraints - to the original formula (Crawford, 1992; Crawford et al., 1996; Aloul et al., 2003; Walsh, 2006).

Before introducing the general definition of SBP, let us illustrate the main idea behind this technique using a simple example. Let $\sigma = (x_1, y_1)$ be a symmetry of a CNF formula $\mathcal{F}$ with only one cycle. Suppose that $\mathcal{F}$ admits $m = \{x_1, \neg y_1 \dots\}$ as a model, then $\sigma(m) = \{\neg x_1, y_1, \dots\}$ is also a model of $\mathcal{F}$. To break this symmetry, it is sufficient to lay down an ordering on the values of x_1 and y_1. For example, adding conjunctively the constraint $x_1 \leqslant y_1$, which can be expressed by the clause $c = (\neg x_1 \vee y_1)$, to the formula $\mathcal{F}$, leads to a new formula $\Phi = \mathcal{F} \cup \{c\}$ while preserving satisfiability. The model m of $\mathcal{F}$ is eliminated as it is not a model of Φ. All other models of $\mathcal{F}$ not satisfying the added binary clause are also eliminated. This idea is generalized in definition 3 to a symmetry containing arbitrary number of cycles.

Definition 2. *Let $\sigma = (x_1, y_1), \dots, (x_n, y_n)$ be a symmetry of $\mathcal{F}$. σ is called lexicographically ordered iff $\forall i (1 \leqslant i \leqslant n-1)\ |x_i| < |x_{i+1}|$ and $\forall i (1 \leqslant i \leqslant n)$ $|x_i| < |y_i|$ holds.*

Definition 3 (SBP (Crawford et al., 1996))**.** *Let $\mathcal{F}$ be a CNF and $\sigma = (x_1, y_1)(x_2, y_2) \dots (x_n, y_n)$ a symmetry of $\mathcal{F}$. Then the symmetry breaking predicates, called sbp_σ, associated to a lexicographically ordered symmetry σ is defined as the conjunction of the following constraints:*

- $(x_1 \leqslant y_1) \wedge$
- $(x_1 = y_1) \rightarrow (x_2 \leqslant y_2) \wedge$
- $\dots \wedge$
- $(x_1 = y_1) \wedge (x_2 = y_2) \dots (x_{n-1} = y_{n-1}) \rightarrow (x_n \leqslant y_n)$

Similarly, in order to break a set of symmetries one need to add conjunctively symmetry breaking predicates associated to each individual symmetry.

The following property shows that symmetry breaking predicates approach preserves the satisfiability between the original formula and the generated one.

Proposition 1 ((Crawford et al., 1996))**.** *Let $\mathcal{F}$ be a CNF formula and σ a symmetry of $\mathcal{F}$. Then $\mathcal{F}$ and $(\mathcal{F} \wedge sbp_\sigma)$ are equivalent w.r.t. satisfiability.*

In order to limit the combinatorial explosion of the clausal transformation of these predicates, one has to add one variable α_i per cycle (x_i, y_i) to express the equality between x_i and y_i. However, one of the major drawbacks of this approach is that the size of the symmetry breaking predicates is exponential in the worst case. Recently, interesting reductions in the size of the SBP has been obtained in (Aloul et al., 2006) using non redundant generators concept.

3 PIEGON-HOLE BASED ENCODING OF CARDINALITY CONSTRAINTS

$\sum_{i=1}^{n} x_i \geqslant b$ such that x_i is propositional variable ($x_i \in \{0,1\}$), for $1 \leqslant i \leqslant n$, is a well known cardinality constraint. As mentioned by Joot P. Warners in (Warners, 1996), this kind of constraints and its generalized form $\sum_{i=1}^{n} a_i x_i \geqslant b$ (where a_i are positive integers) can be polynomially encoded as a propositional formula in CNF. The first polynomial CNF expansion of cardinality constraint is proposed by Hooker in an unpublished note (see also (Warners, 1996)). The authors start from the encoding formulated of the constraint $\sum_{i=1}^{n} x_i \geqslant b$ as it is was described in (Warners, 1996) (page 12):

$$(\neg z_{ik} \vee x_i), \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant k \leqslant b \tag{3}$$

$$\bigvee_{i=1}^{n} z_{ik}, \quad 1 \leqslant k \leqslant b \tag{4}$$

$$(\neg z_{ik} \vee \neg z_{jk}), \quad 1 \leqslant i < j \leqslant n, \quad 1 \leqslant k \leqslant b \tag{5}$$

In (Warners, 1996) the author mentions that the formula (3) says that x_i is true if some z_{ik} is true, while formula (4) combined with formula (5) say that for each k exactly one z_{ik} must be true.

However this formulation is clearly wrong. Let us give a counter example. Suppose that $x_i = 0$ for $1 \leqslant i \leqslant n-(b-1)$. In such a case, the cardinality constraint $\sum_{i=1}^{n} x_i \geqslant b$ is unsatisfiable as one needs to set

b variables to *true* among the set of remaining unassigned variables $R = \{x_{n-(b-2)}, x_{n-(b-3)}, \ldots, x_n\}$. Indeed, this is clearly impossible as the number of unassigned variables is $n-(n-(b-2))+1 = b-1$. On the contrary, the CNF formula made of (3), (4) and (5) is satisfiable. One can set the remaining variables of R to *true* and for each k ($1 \leqslant k \leqslant b$) set exactly one z_{ik} to *true* for $(n-(b-2) \leqslant i \leqslant n)$.

Despite of the importance of the Warners' paper and its precursory nature on the subject, to our knowledge, this error in the formulation of the first translation of the cardinality constraint to CNF reported by Warners was never raised.

Based on the description above, Jabbour et al. proposed in (Jabbour et al., 2014) the correct reformulation of the CNF representation of the cardinality constraint $\sum_{j=1}^{n} x_j \geqslant b$, denoted $\mathcal{P}_n^b$ in the sequel:

$$\bigwedge_{k=1}^{b} (\neg p_{ki} \vee x_i), \quad 1 \leqslant i \leqslant n \tag{6}$$

$$\bigvee_{i=1}^{n} p_{ki}, \quad 1 \leqslant k \leqslant b \tag{7}$$

$$\bigwedge_{1 \leqslant k < k' \leqslant b} (\neg p_{ki} \vee \neg p_{k'i}), \quad 1 \leqslant i \leqslant n \tag{8}$$

Let us mention that the two equations (7) and (8) encode the well-known pigeon hole problem PHP_n^b, where b is the number of pigeons and n is the number of holes (p_{ki} expresses that pigeon k is in hole i). The mapping between the models of PHP_n^b and those of $\sum_{i=1}^{n} x_i \geqslant b$ are obtained thanks to the equation (6). Indeed, the propositional variable x_i is true if the hole i contains one of the pigeons k for $1 \leqslant k \leqslant b$. If we take again the previous counter example, the CNF formula $\mathcal{P}_n^b$ becomes unsatisfiable as it encodes an unsatisfiable Pigeon-Hole problem PHP_{b-1}^b.

In this original polynomial transformation, the number of variables is $n + b \times n$ and the number of clauses required is $n \times b + b + n \times \frac{b \times (b-1)}{2}$. The overall complexity is in $O(b \times n)$ variables and $O(n \times b^2)$ clauses.

Unfortunately, checking the satisfiability of a Pigeon-Hole formula is computationally hard except if we use resolution with symmetry or extended resolution proof systems. In the following, we show how to improve the efficiency of this Pigeon-Hole based encoding of the cardinality constraint. By efficiency, we mean enhancing the propagation capabilities (unit propagation) of the obtained CNF. To this end, we show in the next section, how symmetries of the this Pigeon-Hole formulation can be used to enhance this first version of our encoding.

3.1 Symmetry Breaking on the Pigeon-Hole based Encoding

An enhancement of Pigeon-Hole Based Encoding is proposed using symmetry breaking predicates and used to reduce the size of Pigeon-Hole based encoding of the cardinality constraint while ensuring unit propagation.

For clarity reason, and to better visualize the reductions on the previous encoding $\mathcal{P}_n^b$, we use the following matrix representation for the CNF formula (7). Each row represents a positive clause of (7).

$$\begin{pmatrix} p_{11} & \cdots & & [p_{1b} & \cdots & p_{1n}] \\ p_{21} & \cdots & [p_{2(b-1)} & \cdots & p_{2(n-1)}] & p_{2n} \\ \vdots & & \iddots & & \iddots & \vdots \\ [p_{b1} & \cdots & p_{b(n-(b-1))}] & & \cdots & p_{bn} \end{pmatrix}$$

Efficient Encoding. The enhanced CNF Pigeon-Hole based encoding, called $ph\mathcal{P}_n^b$, of a cardinality constraint is defined as:

$$\neg p_{(b-k+1)(i+k-1)} \vee x_{(i+k-1)}, \quad 1 \leqslant i \leqslant n-b+1, \; 1 \leqslant k \leqslant b \tag{9}$$

$$\bigvee_{i=1}^{n-b+1} p_{(b-k+1)(i+k-1)}, \quad 1 \leqslant k \leqslant b \tag{10}$$

$$p_{(b-k+1)k} \vee \cdots \vee p_{(b-k+1)(i+k)} \vee \neg p_{(b-k)(i+k+1)}, \; 0 \leqslant i \leqslant n-b-1, 1 \leqslant k \leqslant b-1 \tag{11}$$

This efficient $ph\mathcal{P}_n^b$ encoding is obtained from $\mathcal{P}_n^b$ encoding using sophisticated reductions. Before illustrating how such reductions are performed, let us describe briefly this encoding. The formula (10) corresponds to the reduction of (7) to only the subclauses represented in brackets (see the previous matrix). These sub-clauses are obtained by deducing that the literals belonging to the upper-left corner triangle and to the lower-right corner triangle of the previous matrix must be assigned to false. For instance, the clause $p_{b1} \vee \cdots \vee p_{b(n-(b-1))} \in (11)$ is obtained for $k = 1$, corresponding to the last clause in brackets of the previous matrix. Moreover, the formula (9) corresponds to the restriction of (6) to the variables appearing in (10). Finally, the formula (11), called stair-implications, link successive rows in the matrix from the bottom to the top. With these implications the set of negative binary clauses (8) are made redundant and

then can be dropped. One can see that the number of clauses of (11) is smaller than that of (8).
From $ph\mathcal{P}_n^b$, one can deduce that the overall complexity of our encoding is in $O(b \times (n-b))$ variables and clauses.

3.2 $ph\mathcal{P}_n^b$ Encoding: Algorithm

In this section, we provide an algorithm to help the user to generate CNF Pigeon-Hole Based encoding in a simple way. Let us first consider the following example.

Example 1. *Let $x_1+x_2+x_3+x_4+x_5+x_6+x_7 \geq 5$ a cardinality constraint. The following matrix, is given in order to better visualise how the CNF Pigeon-Hole Based encoding is derived.*

$$\begin{pmatrix} & & & & p_{51} & p_{52} & p_{53} \\ & & & p_{41} & p_{42} & p_{43} & \\ & & p_{31} & p_{32} & p_{33} & & \\ & p_{21} & p_{22} & p_{23} & & & \\ p_{11} & p_{12} & p_{13} & & & & \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{pmatrix}$$

The rows of the matrix allow us to derive the positive clauses of (10):

$p_{51} \vee p_{52} \vee p_{53}$
$p_{41} \vee p_{42} \vee p_{43}$
$p_{31} \vee p_{32} \vee p_{33}$
$p_{21} \vee p_{22} \vee p_{23}$
$p_{11} \vee p_{12} \vee p_{13}$

The binary clauses of (9) are obtained as follows: for each column j of the matrix, we generate the binary clauses connecting the literals of the column j with the variables x_j

$\neg p_{11} \vee x_1$
$\neg p_{12} \vee x_2$ $\quad \neg p_{21} \vee x_2$
$\neg p_{13} \vee x_3$ $\quad \neg p_{22} \vee x_3$ $\quad \neg p_{31} \vee x_3$
$\neg p_{23} \vee x_4$ $\quad \neg p_{32} \vee x_4$ $\quad \neg p_{41} \vee x_4$
$\neg p_{33} \vee x_5$ $\quad \neg p_{42} \vee x_5$ $\quad \neg p_{51} \vee x_5$
$\neg p_{43} \vee x_6$ $\quad \neg p_{52} \vee x_6$
$\neg p_{53} \vee x_7$

The last category of clauses corresponds to (11). The clauses express a relation between two successives lines in the matrix representation. Such a relation can be easily derived by a simple observation on the above matrix.

$p_{11} \vee \neg p_{21}$ $\quad p_{11} \vee p_{12} \vee \neg p_{22}$
$p_{21} \vee \neg p_{31}$ $\quad p_{21} \vee p_{22} \vee \neg p_{32}$
$p_{31} \vee \neg p_{41}$ $\quad p_{31} \vee p_{32} \vee \neg p_{42}$
$p_{41} \vee \neg p_{51}$ $\quad p_{41} \vee p_{42} \vee \neg p_{52}$

In the sequel we present a transformation procedure (Algorithm 1) allowing us to derive $ph\mathcal{P}_n^b$ from a given cardinality constraint. Algorithm 1 starts by creating the matrix $p(b \times n-b+1)$ with b rows and $n-b+1$ columns, where each element $p_{ij}(0 \leq i < b, 0 \leq j < n-b+1)$ corresponds to the propositional variables set by the function `newVar()` (line 4). In line 5, we generate the positive clauses of (10). The initialization of the matrix p of variables together with the generation of the positives clauses of (10) are done in lines 1-8. From line 9 to line 20, the algorithm build all the clauses of (11) and (9). Note that by inverting the rows and columns of the matrix, the clauses of (9) are generated as $(x_{i+j-1} \vee \neg p_{ij})$ (line 18) which is clearly more simple.

Algorithm 1: Pigeon-Hole-Based CNF encoding.

Require: A cardinality constraint $\sum_{i=1}^{n} x_i \geqslant b$
1: **for** $(i = 1;\ i \leq b;\ i++)$ **do**
2: $\quad c = \emptyset$
3: $\quad$ **for** $(j = 1; j \leq (n-b+1); j++)$ **do**
4: $\quad\quad p_{ij} = newVar()$
5: $\quad\quad c \leftarrow c \cup p_{ij}$
6: $\quad$ **end for**
7: $\quad \mathcal{F} \leftarrow \mathcal{F} \cup c$
8: **end for**
9: **for** $(i = 1;\ i \leq b;\ i++)$ **do**
10: $\quad c = \emptyset$
11: $\quad$ **for** $(j = 1;\ j \leq (n-b+1);\ j++)$ **do**
12: $\quad\quad c \leftarrow c \cup p_{ij}$
13: $\quad\quad$ **if** $(i \leq b-1$ && $j \leq n-b)$ **then**
14: $\quad\quad\quad c \leftarrow c \cup \neg p_{(i+1)j}$
15: $\quad\quad\quad \mathcal{F} \leftarrow \mathcal{F} \cup c$
16: $\quad\quad\quad c \leftarrow c \setminus \{\neg p_{(i+1)j}\}$;
17: $\quad\quad$ **end if**
18: $\quad\quad \mathcal{F} \leftarrow \mathcal{F} \cup \{\neg p_{ij}, x_{i+j-1}\}$
19: $\quad$ **end for**
20: **end for**
21: **return** $\mathcal{F}$

4 ENHANCING $PH\mathcal{P}_B^N$ BY REDUCING THE SIZE OF THE CNF IN TERMS OF LITERALS

In this section we propose an enhancement of the previous encoding in order to reduce the size of the CNF i.e. the total number of literal occurrences. Indeed, until now we considered the complexity w.r.t. the number of variables and clauses needed to encode the cardinality constraint into CNF. In the sequel, we propose an interesting enhancement of our encoding by reducing the size of the CNF. The following proposi-

tion states the number of literals occurrences needed to obtain $ph\mathcal{P}_n^b$.

Proposition 2. *$|ph\mathcal{P}_b^n|$ is in $O(n^3)$*

Proof. The pigeon-hole encoding is obtained through three set of clauses (9), (10) and (11). The number of literals of (9) is equals to $(b-1)\times(2+\ldots+(n-b+1)) = (b-1)\times((n-b+1)\times(\frac{n-b+2}{2})-1)$. For (10) the size of the clauses is $b\times(n-b+1)$. Finally the total size of the clauses of (11) is $2\times b\times(n-b+1)$. By considering the worst case, where $b=\frac{n}{2}$, we deduce that the size of $ph\mathcal{P}_b^n$ is in $O(n^3)$. □

According to Proposition 2, the number of literal occurrences is in $O(b\times(n-b)^2)$ in the worst case. Consequently, for b near $\frac{n}{2}$ and for large values of n, the encoding leads to huge CNF formula. Furthermore, as many clauses of (10) and (11) are of large size, this will slow down the unit propagation process. To overcome this drawback, we propose a more compact representation of (10) and (11) allowing to reduce the complexity in terms of literal occurrences from $O(n^3)$ to quadratic in the worse case. To this end, we propose to make use of the mining based compression approach of CNF formulae proposed in (Jabbour et al., 2013). The compression approach is obtained using an original combination of data mining techniques with the well known Tseitin's encoding (Tseitin, 1968). The proposed approach called Mining4CNF uses itemset mining techniques to detect hidden structures in the CNF and use them to reduce the size of the CNF formula. More precisely, the approach allows to derive frequent (appearing many times in the formula) sub-clauses. Such sub-clauses are then substituted by a fresh variable, while adding a new boolean function representing such sub-clauses (Tseitin encoding). Also, in (Jabbour et al., 2013), the authors show that this compression technique achieves significant compression rate on many CNF instances including some specialized constraints such as the AtMostOne Constraint ($\sum_{i=1}^{n} x_i \leqslant 1$). To illustrate such approach, let us consider the following example.

Example 2. *Let Φ be the formula containing the following 10 clauses:*

$$\begin{array}{rrr} x_0 \vee \neg x_4, & x_0 \vee \neg x_5, & x_0 \vee \neg x_6, \\ \neg x_3 \vee \neg x_4, & \neg x_3 \vee \neg x_5, & \neg x_3 \vee \neg x_6, \\ \neg x_0 \vee x_1 & \vee & x_4 \vee x_5 \vee x_6 \\ x_3 & \vee & x_4 \vee x_5 \vee x_6 \\ \neg x_1 \vee x_2 & \vee & x_4 \vee x_5 \vee x_6 \\ \neg x_2 \vee x_3 & \vee & x_4 \vee x_5 \vee x_6 \end{array}$$

Mining4CNF first enumerates some interesting (or frequent) sub-clauses, and use them to compress the CNF formula in the second step. Suppose that $(x_4 \vee x_5 \vee x_6)$ is a frequent sub-clause, the formula Φ can be rewritten as:

$$\begin{array}{rrr} x_0 \vee \neg x_4, & x_0 \vee \neg x_5, & x_0 \vee \neg x_6, \\ \neg x_3 \vee \neg x_4, & \neg x_3 \vee \neg x_5, & \neg x_3 \vee \neg x_6, \\ \neg x_0 \vee x_1 & \vee & \mathbf{y} \\ x_3 & \vee & \mathbf{y} \\ \neg x_1 \vee x_2 & \vee & \mathbf{y} \\ \neg x_2 \vee x_3 & \vee & \mathbf{y} \\ \neg \mathbf{y} & \vee & x_4 \vee x_5 \vee x_6 \end{array}$$

As we can remark, an implication $\mathbf{y} \rightarrow x_4 \vee x_5 \vee x_6$ is sufficient, as the sub-clause $(x_4 \vee x_5 \vee x_6)$ occurs with positive polarity. This enhancement is introduced by Plaisted and Greenbaum that essentially produces a subset of Tseitin's representation (Plaisted and Greenbaum, 1986).
In this simple example, the original formula contains 31 literals, while the new formula involves only 27 literals. As the compression process is based on the Tseitin encoding, the transformation preserves satisfiability.

Before presenting how Mining4CNF can be adapted to compress our Pigeon-Hole based encoding of cardinality constraints, let us recall the clauses encoded in (11) and (10). Equation 11 expresses a relation between two successives rows in the matrix representation. In order to simplify its representation the indices are then changed below since some literals are proved to be false. The clauses of (11) present a staircase form.

Let us fix $k=(n-b)$ to simplify the new matrix representation as follows:

$$\begin{pmatrix} & & p_{b1} & p_{b2} & \cdots & p_{b(n-b)} & p_{b(k+1)} \\ & & & \vdots & & & \\ \rightarrow & p_{21} & p_{22} & \cdots & p_{2k} & p_{2(k+1)} & \\ \rightarrow & p_{11} & p_{12} & \cdots & p_{1k} & p_{1(k+1)} & \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ & x_1 & x_2 & \cdots & \cdots & \cdots & \cdots & x_n \end{pmatrix}$$

The clauses linking the two rows pointed by arrows in the above matrix are:

$$\begin{array}{l} p_{11} \vee \neg p_{21} \\ p_{11} \vee p_{12} \vee \neg p_{22} \\ p_{11} \vee p_{12} \vee p_{13} \vee \neg p_{23} \\ \vdots \\ p_{11} \vee p_{12} \vee \ldots \vee p_{1k} \vee \neg p_{2k} \end{array}$$

So such clauses form a triangle. Note that the clauses of (10) corresponds to the rows of the matrix. For the compression purposes, we add to each triangle one positive clause (in bold font) from (10) as follows:

$$
\begin{array}{l}
p_{11} \vee \neg p_{21} \\
p_{11} \vee p_{12} \vee \neg p_{22} \\
p_{11} \vee p_{12} \vee p_{13} \vee \neg p_{23} \\
\vdots \\
p_{11} \vee p_{12} \vee \ldots \vee p_{1k} \vee \neg p_{2k}
\end{array}
$$

$$
\mathbf{p_{11} \vee p_{12} \vee \ldots \vee p_{1k} \vee p_{1(k+1)} \vee \bot}
$$

To obtain the set of all clauses encoded by (10) and (11), we add conjunctively all the triangles (clauses) that can be generated from each two successives rows of the matrix. As we can observe, the number of triangles is $b-1$ while the number of rows (positive clauses) is b. By adding one positive clause of (10) to its corresponding triangle, the following positive clause $(p_{b1} \vee \ldots \vee p_{b(k+1)})$ remains.

Based on this sets of clauses (in the form of triangles), we can observe that there they contain many frequent sub-clauses. For example, the sub-clauses $c = (p_{11} \vee p_{12} \vee \ldots \vee p_{1\frac{k+1}{2}})$ appears $(k+1)/2$ times. For the simplicity of the presentation, we consider k an odd number. Applying Mining4CNF approach leads to the substitution each sub-clause in all clauses where it appears with a new variable α. To preserve satisfiability, we have to add the clause $(p_{11} \vee p_{12} \vee \ldots \vee p_{1\frac{k+1}{2}} \vee \neg\alpha)$. This process allows to substitute $(\frac{k+1}{2})^2$ literals with $(\frac{k+1}{2} + \frac{k+1}{2} + 1)$ literals. Consequently, the size reduction in terms of number of literals is $(\frac{k+1}{2})^2 - (\frac{k+1}{2} + \frac{k+1}{2} + 1) = \frac{k^2}{4} - \frac{k}{2} - \frac{7}{4}$ literals. After replacing such sub-clause and adding the new clause, one can remark that the new derived formula can be divided into the two following formulae:

$$
\begin{array}{l}
p_{11} \vee \neg p_{21} \\
p_{11} \vee p_{12} \vee \neg p_{22} \\
p_{11} \vee p_{12} \vee p_{13} \vee \neg p_{23} \\
\vdots \\
p_{11} \vee p_{12} \vee \ldots \vee p_{1\frac{k}{2}} \vee \neg p_{2\frac{k}{2}} \\
p_{11} \vee p_{12} \vee \ldots \vee p_{1\frac{k}{2}} \vee p_{1\frac{k+1}{2}} \vee \neg\alpha
\end{array}
$$

and

$$
\begin{array}{l}
\alpha \vee \neg p_{2\frac{k+1}{2}} \\
\alpha \vee p_{1\frac{k+3}{2}} \vee \neg p_{2\frac{k+3}{2}} \\
\alpha \vee p_{1\frac{k+3}{2}} \vee p_{1\frac{k+5}{2}} \vee \neg p_{2\frac{k+5}{2}} \\
\vdots \\
\alpha \vee p_{1\frac{k+3}{2}} \vee p_{1\frac{k+5}{2}} \vee p_{1\frac{k+7}{2}} \vee \ldots \vee p_{1k} \vee \neg p_{2k} \\
\alpha \vee p_{1\frac{k+3}{2}} \vee p_{1\frac{k+5}{2}} \vee \neg p_{1\frac{k+7}{2}} \vee \ldots \vee p_{1k} \vee \neg p_{1(k+1)} \vee \bot
\end{array}
$$

Interestingly, partitioning the original formula (triangle) as two formulae (triangles) allows us to define a recurrence relation that we describe later.

In the following, we formally describe how to compact the equations (10) and (11), using the sets of clauses (in the form of triangles). Let us first define the function f as follows:

$$
f(x_1,\ldots,x_n,y_1,\ldots,y_n) = \bigwedge_{i=1}^{n} (\neg y_i \vee \bigvee_{j=1}^{i} x_j)
$$

It is straightforward to conclude that the clauses linking two rows in the matrix can be expressed using the function f. For rows number 1 and 2 (with arrows on the left hand side of the matrix), it can be defined as $f(p_{11},\ldots,p_{1(k+1)},p_{21},\ldots,p_{2k},\bot)$. Each application of f corresponds to a triangle of clauses (see above). Then the clauses of equations (10) and (11) can be rewritten using f as:

$$
\left(
\begin{array}{ll}
(p_{b1} \vee \ldots \vee p_{b(k+1)}) & \wedge \\
\bigwedge_{i=1}^{b-1} f(p_{i1},\ldots,p_{i(k+1)},p_{(i+1)1},\ldots,p_{(i+1)k},\bot) &
\end{array}
\right)
$$

Then, the general CNF formula of $ph\mathcal{P}_b^n$ can be defined as:

$$
\left(
\begin{array}{ll}
(p_{b1} \vee \ldots \vee p_{b(k+1)}) & \wedge \\
\bigwedge_{i=1}^{b-1} f(p_{i1},\ldots,p_{in},p_{(i+1)1},\ldots,p_{(i+1)k},\bot) & \wedge \\
\bigwedge_{i+j=k+1} (x_k \vee \neg p_{ij}) &
\end{array}
\right)
$$

Algorithm 2 describes how to compress $f(x_1,\ldots,x_n,y_1,\ldots,y_n)$. It apply a greedy approach to replace frequent sub-clauses by choosing the one allowing to maximize the reduction rate. Note that, when n is less than 5 (line 1), compacting f do not leads to any improvement in terms of number of literals.

Algorithm 2: $f(x_1,\ldots,x_n,y_1,\ldots,y_n)$.

1: **if** $n \leq 5$ **then**
2: **return** $\bigwedge_{i=1}^{n}(\neg y_i \vee \bigvee_{j=1}^{i} x_j)$
3: **end if**
4: k = (int) (n / 2)
5: $\alpha \leftarrow newVar()$
6: **return** $f(x_1,\ldots,x_{k-1},x_k,y_1,\ldots,y_{k-1},\neg\alpha) \wedge f(\alpha,x_{k+1},\ldots,x_n,y_k,\ldots,y_n)$

Proposition 3. *Using Mining4CNF compression approach, $|ph\mathcal{P}_b^n|$ is in $O(n^2)$.*

Proof. Let us now show the complexity of our new encoding after the applications of the compression approach. Note that the number of literals occurrences encoded in $f(p_{i1},\ldots,p_{i(k+1)},p_{(i+1)1},\ldots,p_{(i+1)k},\bot)$ is in $O(k)$. Indeed, let us denote by $\mathcal{L}(n)$ the number

of literal occurrences of $f(x_1,\ldots,x_n,y_1,\ldots,y_n)$. According to Algorithm 2, $\mathcal{L}(n)$ satisfies the following recurrence equation:

$$\mathcal{L}(n) = 2 \times \mathcal{L}(\frac{n}{2}) \quad (12)$$

$$\mathcal{L}(5) = 20. \quad (13)$$

It is clear that $\mathcal{L}(n)$ is in $O(n)$. Then, using such algorithm, the number of literal occurrences of f is reduced from quadratic to linear. As in $ph\mathcal{P}_b^n$, f is used $(b-1)$ times, then the complexity of $ph\mathcal{P}_b^n$ in term of number of literal occurrences becomes quadratic, i.e., $O(b \times (n-b))$. □

Let us note that the compression approach increases the number of fresh variables and clauses. However, the complexity of our Pigeon-Hole based encoding $ph\mathcal{P}_n^b$ in term of additional clauses and literals remains the same. Indeed, the number of new variables and clauses added using Algorithm 2 does not exceed n. Then the complexity of $ph\mathcal{P}_n^b$ is in $O(b \times (n-b))$ variables and $O(b \times (n-b))$ clauses.

5 EXPERIMENTS

In this section, we carried out an experimental evaluation of the performance of our enhanced Pigeon-Hole based encoding of the cardinality constraint. The primary goal is to assess the competitiveness of our proposal. For this purpose, we compared the performances using *QMaxSAT* [1] solver used to solve MaxSAT instances. Let us recall that `QMaxSAT` uses the encoding defined in (Bailleux and Boufkhad, 2003). It is also based on minsat [2] solver. We use the last version of minisat. We denote by *QMaxSAT-PHP* the version of QMaxSAT where the encoding (Bailleux and Boufkhad, 2003) is substituted with the enhanced pigeon-holed based encoding. We considered the instances of Partial MaxSAT competition 2015 [3]. All the experiments are done on a cluster Intel Xeon quad-core avec 32GB of RAM et 2.66 Ghz. The time out is fixed to 30 minutes.

Figure 1 represents the results obtained on instances encoding partial MaxSAT problems belonging to crafted category. Each dot (x,y) represents the number of instances x solved in less than y seconds. As we can see, `QMaxSAT-PHP` is more efficient than classical QMaxSAT. It solves 32 instances more. Furthermore, there exists some classes of instances where our approach is clearly the best e.g., `AES*`, `s38584*`.

[1] https://sites.google.com/site/qmaxsat/

[2] http://minisat.se/

[3] http://www.maxsat.udl.cat/15/

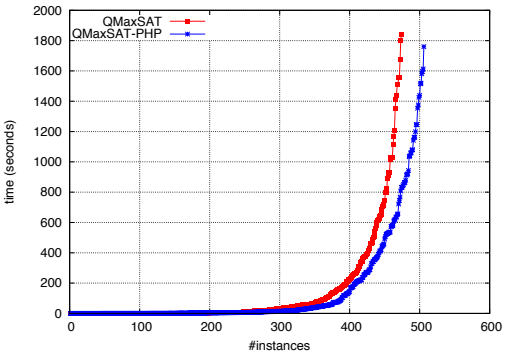

Figure 1: Results on crafteds instances.

Figure 2 represents the results obtained on instances encoding partial MaxSAT problems belonging to application category. In contrast to crafted instances, here classical QMaxSAT outperforms our solver. It solves 51 instances more. Furthermore, as for crafted case, there exists a set of classes where QMaxSAT is the best e.g., `splitedReads*`, `b20_C-mbd14-0202*`, `b20-s_PathRelaxation_Set_FS*`. However, there exists classes where our solver is better e.g., `atcoss*`.

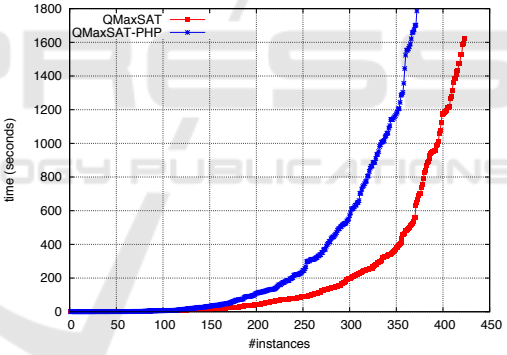

Figure 2: Results on applications instances.

6 CONCLUSION AND FUTURE WORKS

In this paper, we proposed an enhancement of the Pigeon-Hole based encoding of cardinality constraints into CNF. The new encoding is competitive as it remains in $O(b(n-b))$ variables and clauses. Interestingly, we demonstrate that mining-based compression techniques can achieve substantial reduction in the size of the encoding. This opens a promising perspective on how to extend the reasoning applied in this paper to other kinds of constraints (e.g. global constraints). Experimental results shows that our new encoding is competitive on crafted instances.

The generalization of our reasoning to encode general pseudo Boolean constraint to CNF is also a short term perspective. Finally, we plan to conduct an experimental evaluation of our Pigeon-Hole based encoding w.r.t. the well-known encodings.

REFERENCES

Aloul, F. A., Ramani, A., Markov, I. L., and Sakallah, K. A. (2003). Solving difficult instances of boolean satisfiability in the presence of symmetry. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(9):1117–1137.

Aloul, F. A., Sakallah, K. A., and Markov, I. L. (2006). Efficient symmetry breaking for boolean satisfiability. *IEEE Trans. Computers*, 55(5):549–558.

Asín, R., Nieuwenhuis, R., Oliveras, A., and Rodríguez-Carbonell, E. (2009). Cardinality networks and their applications. In *12th International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, pages 167–180.

Bailleux, O. and Boufkhad, Y. (2003). Efficient cnf encoding of boolean cardinality constraints. In *9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, pages 108–122.

Bailleux, O., Boufkhad, Y., and Roussel, O. (2009). New encodings of pseudo-boolean constraints into cnf. In *SAT'2009*, pages 181–194.

Benhamou, B. and Sais, L. (1992). Theoretical study of symmetries in propositional calculus and applications. In *11th International Conference on Automated Deduction (CADE'1992)*, volume 607 of *Lecture Notes in Computer Science*, pages 281–294. Springer.

Benhamou, B. and Sais, L. (1994). Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12(1):89–102.

Cook, S. A. (1976). A short proof of the pigeon hole principle using extended resolution. *SIGACT News*, 8(4):28–32.

Crawford, J. (1992). A theorical analysis of reasoning by symmetry in first order logic. In *Proceedings of Workshop on Tractable Reasonning, AAAI*, pages 17–22.

Crawford, J. M., Ginsberg, M. L., Luks, E. M., and Roy, A. (1996). Symmetry-breaking predicates for search problems. In *KR*, pages 148–159.

Eén, N. and Sörensson, N. (2006). Translating pseudo-boolean constraints into sat. *JSAT*, 2(1-4):1–26.

Gent, I. P., Petrie, K. E., and Puget, J.-F. (2006). Chapter 10 symmetry in constraint programming. In F. Rossi, P. v. B. and Walsh, T., editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 329 – 376. Elsevier.

Jabbour, S., Saïs, L., and Salhi, Y. (2014). A pigeon-hole based encoding of cardinality constraints. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2014, Fort Lauderdale, FL, USA, January 6-8, 2014*.

Jabbour, S., Sais, L., Salhi, Y., and Uno, T. (2013). Mining-based compression approach of propositional formulae. In *CIKM*, pages 289–298.

Krishnamurthy, B. (1985). Shorts proofs for tricky formulas. *Acta Informatica*, 22:253–275.

Kullmann, O. (1997). On a generalization of extended resolution. *Discrete Applied Mathematics*, 34:73–95.

Plaisted, D. A. and Greenbaum, S. (1986). A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2(3):293–304.

Puget, J. (1993). On the satisfiability of symmetrical constraint satisfaction problems. In *proceedings of ISMIS*, pages 350–361.

Silva, J. P. M. and Lynce, I. (2007). Towards robust cnf encodings of cardinality constraints. In *13th International Conference on Principles and Practice of Constraint Programming (CP 2007)*, pages 483–497.

Sinz, C. (2005). Towards an optimal cnf encoding of boolean cardinality constraints. In *11th International Conference on Principles and Practice of Constraint Programming (CP 2005)*, pages 827–831.

Tseitin, G. (1968). On the complexity of derivations in the propositional calculus. In Slesenko, H., editor, *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pages 115–125.

Walsh, T. (2006). General symmetry breaking constraints. In *12th International Conference on Principles and Practice of Constraint Programming (CP 2006)*, pages 650–664.

Warners, J. P. (1996). A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*.