Efficient CNF encoding of Boolean cardinality constraints

Olivier Bailleux[1] and Yacine Boufkhad[2]

[1] LERSIA, Université de Bourgogne
Avenue Alain Savary, BP 47870
21078 Dijon Cedex
olivier.bailleux@u-bourgogne.fr
[2] LIAFA, Université Paris 7
Case 7014 - 2, place Jussieu F-75251 Paris Cedex 05
Yacine.Boufkhad@liafa.jussieu.fr

Abstract. In this paper, we address the encoding into CNF clauses of Boolean cardinality constraints that arise in many practical applications. The proposed encoding is efficient with respect to unit propagation, which is implemented in almost all complete CNF satisfiability solvers. We prove the practical efficiency of this encoding on some problems arising in discrete tomography that involve many cardinality constraints. This encoding is also used together with a trivial variable elimination in order to re-encode parity learning benchmarks so that a simple Davis and Putnam procedure can solve them.

1 Introduction

Many types of constraints that appear in real world problems have no natural expression in the propositional satisfiability. The cardinality constraint over a set of Boolean variables (i.e., a constraint on the number of variables that can be assigned the value 1) is one of these. The encoding problem that we address is: given a set E of Boolean variables (called input variables) subject to a cardinality constraint requiring that at least μ and at most ρ of them can be equal to 1, build a CNF formula $\Psi(E,\mu,\rho)$ over a set of variables including E such that $\Psi(E,\mu,\rho)$ can be satisfied by a truth assignment if and only if the values assigned to variables in E by this truth assignment satisfy the cardinality constraint. In this paper, we propose an efficient CNF encoding of the cardinality constraint.

While there is no general definition of a good encoding, there are at least some common sense conditions that such an encoding must fulfill. The first one is that the size of the formula must be kept relatively small with respect to E and the second one is that the formula must be adapted to the kind of solver to be used. Our encoding requires $O(n\,log(n))$ variables and $O(n^2)$ clauses of length at most 3, where $n = |E|$. This encoding is also efficient in the sense that unit propagation restores the generalized arc consistency on the variables in E.

The straightforward way of encoding of cardinality constraints is based on a bit adder that adds one by one the variables in E, as in [6]. The result of the addition is represented in the usual binary representation of integers, and the Boolean variables that compose it are constrained by some clauses so that the integer they represent is in the prescribed range. While the size of the CNF formula generated by this encoding remains reasonable, its main disadvantage is that a SAT solver based on unit propagation needs to have all the variables in E assigned a value in order to check the cardinality constraint. Even if the constraint is violated by a partial assignment, unit propagation alone does not generate an empty clause. In the encoding described in this paper, the key feature is a unary representation of integer variables that can represent not only the value, if known, of an integer but also the interval where it falls if the Boolean variables in its unary representation are partially assigned. A bit adder based on this representation allows the derivation of the interval where a variable c falls, given the intervals of variables a and b such that $c = a + b$. We obtain a CNF formula where unit propagation derives all the consequences of every assignment with respect to the generalized arc consistency. In particular, it derives an empty clause whenever a partial assignment to the input variables is inconsistent with the cardinality constraint.

In order to subject this encoding to an application where there are these types of constraints, we have tested it on a problem arising in 2-D discrete tomography. The specific problem that we address is the reconstruction of a pattern lying in a 2-D grid, given its projections in four directions. The projection in some direction is the number of points belonging to the pattern in that direction. The reconstruction problem is to find a pattern that complies with the given projections in every direction. This amounts to finding an assignment to the Boolean variables representing the cells of the grid given many cardinality constraints. Each one of these cardinality constraints, represents the fact that the number of cells belonging to the pattern in some direction, is equal to the projection in that direction. This is a good test for our encoding scheme since each variable is involved in four different cardinality constraints. We compared, using some instances of discrete tomography, the performance of our encoding solved by the state of the art SAT solver zchaff [15] versus a commercial constraint solver.

The cardinality constraints also appear in parity learning instances. These benchmarks have been the center of a challenge to solvers. We show that they can be solved easily using a basic DP procedure, by separating them into two parts: an XOR-CNF formula and a formula containing a mixture of cardinality constraint and some XOR-CNF relations.

At this point, one may make some general remarks on the issue of encoding into CNF. The benchmarks that are extensively used to assess the efficiency of SAT algorithms are generally taken as they are. The issue of finding the best encodings or even the pertinence of encoding them into SAT formulas is rarely brought up. In the challenges that are organized for SAT, only two categories of submission are welcomed: solvers and benchmarks. The issue of improving the proposed encodings in ignored. As the most interesting benchmarks are the hardest ones, ignoring the encoding may have this consequence: some intrinsically easy problems may be made hard by an inappropriate encoding, and, as they are hard, they may be considered as interesting benchmarks. If the final goal is the practical solving of hard real world problems and not only to make solvers overcome inappropriate encodings by rediscovering in the CNF formulas some deductions that are obvious in the original problem, then a careful encoding is crucial and must be considered as a third type of contribution beside solvers and benchmarks in the challenges organized for SAT.

The paper is organized as follows. The encoding is described in Section 2, where correctness and efficiency are proved. Then in Section 3 the encoding is applied to the discrete tomography problems after a brief description of these problems. In Section 4, the learning parity instances are revisited in the light of this new encoding of cardinality constraints.

2 Efficient CNF encoding of cardinality constraints

We give first the notations that will be used throughout this paper. The truth values TRUE and FALSE of propositional logic will be denoted 0 and 1. An *instantiation* or a *truth assignment* I of a set V of propositional variables is a function that maps each variable $v \in V$ to a non empty set $I(v) \subseteq \{0,1\}$. A variable v is said to be *fixed* to 0 or *assigned the value* 0 by an instantiation I if $I(v) = \{0\}$, fixed to 1 if $I(v) = \{1\}$, and free if $I(v) = \{0,1\}$. In non-ambiguous contexts, $v = 1$ denotes $I(v) = \{1\}$ and $v = 0$ denotes $I(v) = \{0\}$. An instantiation I of V is said to be *complete* if it fixes all the variables in V. The instantiations that are not complete are said to be *partial*.

For any CNF formula Φ and any instantiation I of a subset of the variables of Φ, $\Phi|_I$ denotes the formula obtained by replacing the variables that are fixed by I with their truth values.

A *unit clause* is a clause that includes only one literal. *Unit propagation* denotes the process that fixes each variable occurring in a unit clause in such a way as to satisfy this clause, up until the empty clause is produced or no unit clause remains.

2.1 The problem

The goal is to translate a cardinality constraint over a set E of Boolean variables into a CNF formula. The cardinality constraint specifies that the number p of variables fixed to 1 among a set E of Boolean variables is at least μ and at most ρ. The CNF formula $\Psi(E,\mu,\rho)$ is defined on a set $V \supset E$ of propositional variables. The variables in $V \setminus E$ are called *encoding variables*.

The encoding must be *correct* in the sense that for any complete instantiation I of E, $\Psi(E,\mu,\rho)|_I$ is *satisfiable* if and only if I satisfies the cardinality constraint.

The encoding must also be *efficient* in the sense that for any partial instantiation I of E, unit propagation on $\Phi(E,\mu,\rho)|_I$ must restore the generalized arc consistency on the variables in E, specifically:

- if more than ρ variables in E are fixed to 1 or if more than $n-\mu$ variables in E are fixed to 0 then unit propagation produces the empty clause,
- else if ρ variables in E are fixed to 1 then unit propagation fixes to 0 all the other variables in E,
- else if $n-\mu$ variables in E are fixed to 0 then unit propagation fixes to 1 all the other variables in E.

2.2 Encoding principle

The proposed encoding uses a unary representation of integers. The value of an integer x such that $0 \leq x \leq n$ is represented by 1 x times followed by 0 $n-x$ times. An integer variable v with domain $0..n$ is represented by a set $V = \{v_1, v_2, \ldots, v_n\}$ of n propositional variables. Each possible value of v is encoded as a complete instantiation of V, as described above. If $v = x$, then $v_1 = 1$, $v_2 = 1$, ..., $v_x = 1$ and $v_{x+1} = 0$, ..., $v_n = 0$. A partial instantiation of V is said to be *pre-unary* if for each $v_i = 1$, $v_j = 1$ for any $j < i$ and for each $v_i = 0$, $v_j = 0$ for any j, $i \leq j \leq n$. A unary instantiation is then a complete pre-unary instantiation.
The main advantage of such a representation is that the integer variable can be specified as belonging to an interval. Indeed, the inequality $x \leq v \leq y$ is specified by the partial pre-unary instantiation of V that fixes to 1 any v_i such that $i \leq x$ and fixes to 0 any v_j such that $j \geq y+1$.
Conversely, any partial pre-unary instantiation I of V is related to an integer interval. The bounds of this interval will be denoted $min(I)$ and $max(I)$. We underline that the classical binary representation of integers does not allow one to specify such membership relations as our representation does.

Example : With $n = 6$, if I is a partial instantiation such that $v_1 = v_2 = 1$, $v_5 = v_6 = 0$, and v_3, v_4 are free, then $min(I) = 2$ and $max(I) = 4$. Then the corresponding integer variable v is such that $2 \leq v \leq 4$.

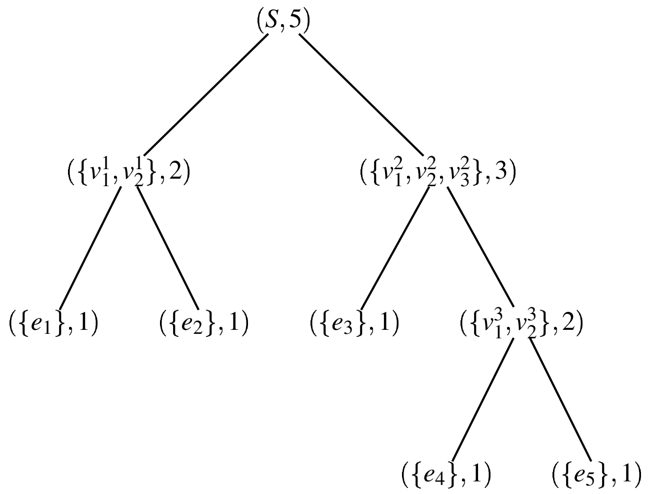
The number of variables fixed to 1 by an instantiation I will be denoted $N(I)$. When I is the unary representation of an integer, $N(I)$ is then the value of this integer.
The encoding of a cardinality constraint on a set E of variables is done in two parts : a *totalizer* and a *comparator*.

The totalizer The totalizer is a CNF formula $\Phi(E)$ defined on 3 sets of variables :

- $E = \{e_1, \ldots, e_n\}$: the set of *input variables*,
- $S = \{s_1, \ldots, s_n\}$: the set of *output variables*,
- a set L of variables called *linking variables*.

These sets of variables can be described by a binary tree built as follows. We start from a isolated node labeled by the integer n and we proceed iteratively: to each terminal node labeled by $m > 1$, we connect two children labeled by $\lfloor m/2 \rfloor$ and $m - \lfloor m/2 \rfloor$, respectively. This procedure produces a binary tree with n leaves labeled by 1. Next, each variable in E is allocated to a leaf in a bijective way. The set S of output variables is allocated to the root node. To each internal node labeled by an integer m, a set of m new variables is allocated which will be used to represent a unary value belonging to $1..m$. The union of the set of variables allocated to the internal nodes is the set L of linking variables.

Example : for $n = 5$, $E = \{e_1, e_2, e_3, e_4, e_5\}$, and $S = \{s_1, s_2, s_3, s_4, s_5\}$ the following tree is obtained:



In this example, the set of linking variables is $L = \{v_1^1, v_2^1, v_1^2, v_2^2, v_3^2, v_1^3, v_2^3\}$.
We will now define a set of clauses that ensures that in any complete instantiation of the variables of the totalizer, the set of variables related to any non-leaf node r with children a and b encodes the unary representation of $\alpha + \beta$, where α and β are the integers encoded by the sets of variables related to a and b.
Let r be an internal node related to children a and b. Let $R = \{r_1, \ldots, r_m\}$ be the set of variables related to r, $A = \{a_1, \ldots, a_{m_1}\}$ be the set of variables related to a, and $B = \{b_1, \ldots, b_{m_2}\}$ be the set of variables related to b. The following conjunction of clauses is related to the node r:

$$\bigwedge_{\substack{0 \leq \alpha \leq m_1 \\ 0 \leq \beta \leq m_2 \\ 0 \leq \sigma \leq m \\ \alpha+\beta=\sigma}} (C_1(\alpha,\beta,\sigma) \wedge C_2(\alpha,\beta,\sigma)) \qquad (1)$$

with the following notations:

$$a_0 = b_0 = r_0 = 1, a_{m_1+1} = b_{m_2+1} = r_{m+1} = 0$$

$$C_1(\alpha,\beta,\sigma) = (\overline{a_\alpha} \vee \overline{b_\beta} \vee r_\sigma)$$

$$C_2(\alpha,\beta,\sigma) = (a_{\alpha+1} \vee b_{\beta+1} \vee \overline{r_{\sigma+1}})$$

Notice that $C_1(\alpha,\beta,\sigma)$ is the CNF representation of the relation $\sigma \geq \alpha+\beta$ and $C_2(\alpha,\beta,\sigma)$ is the CNF representation of the relation $\sigma \leq \alpha + \beta$.
The obtained formula is simplified by removing the clauses including the constant 1 and reducing the clauses including the constant 0. Notice that each clause includes at most three literals.

Lemma 21 (forward propagation) *Let $\Phi(E)$ be a totalizer with n input variables. If p input variables are fixed to 1, q input variables are fixed to 0, and all the other* *variables of the totalizer are free then the partial instantiation I_S of S obtained after unit propagation in $\Phi(E)$ is pre-unary and such that $min(I_S) = p$ and $max(I_S) = n - q$.*

Proof : By induction on the number n of input variables.
For $n = 1$ the totalizer includes only one variable that is either the input and the output variable. The property is then obvious. Now let us consider that the property is true for any $n < \nu$.
Let $A = \{a_1, a_2, ..., a_{\nu_A}\}$ denote the set of the $\nu_A = \lfloor \nu/2 \rfloor$ variables associated to the first child of the root and $B = \{b_1, b_2, ..., b_{\nu_A}\}$ denote the set of the $\nu_B = \nu - \lfloor \nu/2 \rfloor$ variables associated to the other child. Considering the leaves of the tree below each child of the root, we get a partition of the input set E into two disjoint subsets E_A and E_B. Let p_A denote the number of variables fixed to 1 in E_A, p_B denote the number of variables fixed to 1 in E_B, q_A denote the number of variables fixed to 0 in E_A and q_B denote the number of variables fixed to 0 in E_B. Clearly, we have $p = p_A + p_B$ and $q = q_A + q_B$. Let I_A and I_B denote the instantiations of A and B obtained after unit propagation in $\Phi(E)$.
It follows from the induction hypothesis that I_A and I_B are pre-unary, $min(I_A) = p_A$, $max(I_A) = \nu_A - q_A$, $min(I_B) = p_B$, $max(I_B) = \nu_B - q_B$.
It is easy to see that for every t such that $1 \leq t \leq p$ there exist t_A, $t_A \leq min(I_A)$, and t_B, $t_B \leq min(I_B)$ such that $t = t_A + t_B$. Thanks to the clause $\overline{a_{t_A}} \vee \overline{b_{t_B}} \vee s_t$ of the C_1 type associated to the root node, unit propagation fixes to 1 the variable s_t. Consequently, all the variables s_1 to s_p are set to 1. Thanks to the clauses C_2 associated to the root node, we can prove by similar arguments that unit propagation fixes to 0 the variables s_{n-q+1} to s_n.
For the variables s_t such that $p < t \leq n - q$, it is easy to see that for every couple (t_A, t_B) such that $t_A + t_B = t$, either $t_A > min(I_A)$ or $t_B > min(I_B)$. Consequently, at least one of the variables a_{t_A} or b_{t_B} is not equal to 1. Then no clause $\overline{a_{t_A}} \vee \overline{b_{t_B}} \vee s_t$ can be reduced to the unit clause s_t. The same arguments applied to the causes C_2 can be used to prove that the clause $\overline{s_t}$ can not be produced. Then the variables s_t such that $p < t \leq n - q$ are all free. So I_S is pre-unary, $min(I_S) = p$, and $max(I_S) = n - q$.
Then I_S is the unary representation of the smallest interval containing $N(I_E)$.

Lemma 22 (backward propagation) *Let $\Phi(E)$ be a totalizer with n input variables. If:*

- *p input variables are fixed to 1, q input variables are fixed to 0 ($p + q < n$), the remaining input variables being free,*
- *and the output variables s_{p+1} to s_{n-q} are all fixed to the same value ε (0 or 1)*

then all the input variables remaining free are instantiated to ε by unit propagation.

Proof : By induction on the number n of input variables.
For $n = 1$ the property is obvious. Now let us consider that the property is true for any $n < \nu$.
We use the same notations as in the proof of the previous lemma. If we consider solely the assignment to the input variables, the unit propagation assigns to the output variables the values such that $min(I_S) = p$ and $max(I_S) = n - q$. Suppose that s_{p+1} to s_{n-q} variables are assigned the value 0. Let t any integer such that $1 \leq t \leq n - p$. Clearly, we

have $s_{p+t} = s_{p_A+p_B+t} = 0$. If $p_A + t \leq \nu_A$, there is a clause of type C_1 $\overline{a_{p_A+t}} \vee \overline{b_{p_B}} \vee s_{p+t}$. By unit propagation, this clause assigns the value 0 to a_{p_A+t}. Then every variable a_{p_A+t} such that $p_A + t \leq \nu_A$ is assigned the value 0. By the same argument we can prove that every variable b_{p_B+t} such that $p_B + t \leq \nu_B$ is also assigned the value 0. By the induction hypothesis, all the free input variables below the nodes A and B (i.e. all free input variables) are assigned 0.
Similar arguments using the clauses of type C_2 allow the derivation of the conclusion that all the free input variables are assigned 1, if the variables s_{p+1} to s_{n-q} are assigned the value 1.

The comparator The comparator is a set of unary clauses that are satisfied if and only if the instantiation of the input variables of the totalizer represents an interval that matches with the cardinality constraint. Then the constraint $\mu \leq N(E_I) \leq \rho$ will be specified as follows:

$$\bigwedge_{1 \leq i \leq \mu} (s_i) \bigwedge_{\rho+1 \leq j \leq n} (\overline{s_j}) \tag{2}$$

We denote by $\Psi(E, \mu, \rho)$, the conjunction of the CNF formula representing the totalizer and the CNF representing the comparator.

2.3 Correctness and efficiency of the encoding

The correctness and the efficiency of the proposed CNF encoding of the cardinality constraint follows directly from Lemma 21 and 22.

Theorem 23 *The CNF encoding of a cardinality constraint described in the section 2.2 is correct and efficient.*

Proof

1. **Correctness:** It follows from Lemma 21 that for any complete instantiation of the input variables, unit propagation fixes all the other variables of the totalizer in such a way that the instantiation of the output variables is the unary value of the number of input variables fixed to 1. Then the conjunction of the totalizer and the comparator is satisfiable if and only if the number of input variables fixed to 1 belongs to the interval encoded by the unary clauses of the comparator.
2. **Efficiency:** Let p be the number of input variables fixed to 1, q be the number of input variables fixed to 0 ($p + q < n$), μ be the lower bound and ρ be the upper bound of the interval encoded by the comparator. It follows from lemma 21 that unit propagation fixes s_1 to s_p to 1 and fixes s_{n-q+1} to s_n to 0. In addition, the clauses of the comparator allow unit propagation to fix $s_{\rho+1}$ to s_n to 0 and to fix s_1 to s_μ to 1.
   If $p > \rho$ then unit propagation fixes $s_{\rho+1}$ to 1, which is in conflict with the clause $(\overline{s_{\rho+1}})$ of the comparator, thus the empty clause is produced.
   In the same way, if $q > n - \mu$ then unit propagation fixes s_μ to 0, which is in conflict with the clause (s_μ) of the comparator.

If $p = \rho$ then, because the $n - p$ output variables $s_{\rho+1}$ to s_{n-q} are fixed to 0 by the unit clauses of the comparator, it follows from Lemma 22 that unit propagation fixes the free input variables to 0 .
If $q = n - \mu$ then, because the $n - q$ output variables s_1 to s_μ are fixed to 1 by the unit clauses of the comparator, it follows from Lemma 22 that unit propagation fixes the free input variables to 1.

2.4 Complexity issues

The binary tree used to specify the totalizer has $\Theta(\log n)$ levels. Each of these levels requires n linking (or output) variables, thus the totalizer includes $\Theta(n \log n)$ encoding variables.
For sake of simplicity, let us consider that n is a power of 2. For each node related to a set of m linking (or output) variables, there are less than $2m^2$ clauses. Let us number the levels from 1 to l, where l is the number of the root level. For any i such that $1 \leq i \leq l$, the level $l - i$ includes 2^i nodes, each related to $n/2^i$ variables. Then the level $n - i$ includes less than $2^i(2(n/2^i)^2) = 2n^2/2^i$ clauses. So the totalizer includes $O(n^2)$ clauses. Given that the root node of the totalizer requires $\Omega(n^2)$ clauses, and that the comparator requires n clauses, the encoding requires $\Theta(n^2)$ clauses.
Clearly enough, if the cardinality constraint is $\rho \leq N(I_E) \leq \mu$, all the properties described above remain true if any variable with rank upper than μ is initially fixed to 0. This allows one to simplify the formula, using unit propagation, and then reduce its size. If μ is the same order of magnitude as n, this simplification does not change the size complexity of the encoding.

In the worst case, because unit propagation must fixe all the encoding variables, restoring the generalized arc consistency requires time $\Theta(n \log n)$. This time complexity is not optimal, given that a dedicated algorithm using the rules described section 2.1 can restore the generalized arc consistency of a Boolean cardinality constraint in time $O(n)$.

3 Discrete tomography problems

We apply the encoding of the cardinality constraints described in the previous section to a problem arising in discrete tomography. Tomography is a non-destructive method used to examine the interior of solid opaque objects. It consists of sending X-rays at different angles through the object and recording the attenuation at the opposite side. The attenuation reflects the density of the object in a given direction. The problem is then to reconstruct the studied object's image using the attenuation of the X-rays. Tomography is used in many fields ranging from medical imagery to geology and as described in Gardner et al [10] in the determination of the crystalline structure using high resolution transmission electron microscopy. Several mathematical tools have been developed for solving the reconstruction problem in the continuous case. We focus on the reconstruction of 2-D objects given their discrete projections in 4 directions.
This problem have been investigated under various conditions in [17, 4, 19, 9, 13, 1, 21, 3]. Recently Gardner et al [10] proved the NP-Completeness of the problem of testing the existence and the uniqueness of a pattern given its projections in at least 3 directions.

In the 2-D discrete case, the pattern to be reconstructed lies in a grid having n rows and m columns. Each cell of the grid, i.e. a unitary square $[i,i+1] \times [j,j+1]$, is either black, *filled*, if it belongs to the pattern or white, *empty*, if it does not. We will denote the cell located at the intersection of row i and column j by $c_{i,j}$. The i-th row projection and the j-th column projection of the pattern are the numbers of filled cells in the i-the row and the j-th column respectively. The vertical and horizontal projections of a pattern in a grid $n \times m$ are denoted by two vectors $H = (h_1,...,h_i,...,h_n) \in \mathbf{N}^n$ and $V = (v_1,...,v_j,...,v_m) \in \mathbf{N}^m$, h_i and v_j being the i-th row projection and the j-th column projection respectively. Similarly, the k-th diagonal projection is the number of filled cells among the cells $c_{i,j}$ such that $i+j = k+1$. The l-th antidiagonal projection is the number of filled cells among the cells $c_{i,j}$ such that $m+i-j = l+1$. Figure 1 shows the projections of the pattern **6**.

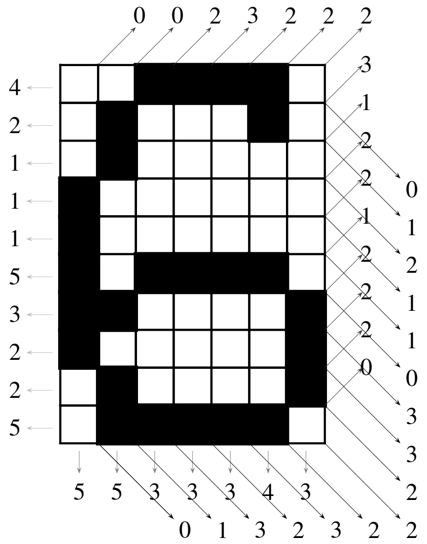


Fig. 1. A pattern representing 6 and its projections, i.e. the number of black cells, in the four directions, yielding the vectors $H = \{4,3,1,1,1,5,3,2,2,5\}$ for the horizontal lines, $V = \{5,5,3,3,3,4\}$ for the vertical lines, $S\{0,0,2,3,2,2,2,3,1,2,2,1,2,2,2,0\}$ for the diagonal 45 degrees directions ,$T\{0,1,2,1,1,0,3,3,2,2,2,3,2,3,1,0\}$ for the diagonal -45 degrees directions.

We adresse specifically this NP-Complete [10] problem :
RECONSTRUCT : Given m,$n \in N$, 4 vectors $H = (h_1,h_2,...,h_m)$, $V = (v_1,v_2,...,v_n)$, $S = (s_1,s_2,...,s_{m+n-1})$ and $T = (t_1,t_2,...,t_{m+n-1})$, is there a pattern $\mathcal{P}$ falling in a $m \times n$ grid such that the horizontal, vertical, diagonal, and antidiagonal projections are respectively H, V, S and T?
We convert this problem into SAT. Every cell $c_{i,j}$ in the grid is represented by a Boolean variable $x_{i,j}$ such that :

$$x_{i,j} = \begin{cases} 1 \text{ if } c_{i,j} \text{ is filled} \\ 0 \text{ if } c_{i,j} \text{ is empty} \end{cases}$$

An instance $\mathcal{R}(n,m,H,V,S,T)$ of **RECONSTRUCT** is then encoded into a SAT instance $\mathcal{F}(n,m,H,V,S,T)$ that is the conjunction of cardinality constraints in every direction. Namely, $\mathcal{F} = \mathcal{H}(n,m,H) \wedge \mathcal{V}(n,m,V) \wedge \mathcal{S}(n,m,S) \wedge \mathcal{T}(n,m,T)$ where :

$$\begin{aligned}
\mathcal{H}(n,m,H) &= \bigwedge_{i=1..n} \Psi(\{x_{i,j}, j = 1..m\}, h_i, h_i) \\
\mathcal{V}(n,m,V) &= \bigwedge_{j=1..m} \Psi(\{x_{i,j}, i = 1..n\}, v_j, v_j) \\
\mathcal{S}(n,m,S) &= \bigwedge_{k=1..m+n-1} \Psi(\{x_{i,j}, i + j = k+1\}, s_k, s_k) \\
\mathcal{T}(n,m,T) &= \bigwedge_{k=1..m+n-1} \Psi(\{x_{i,j}, m + i - j = k+1\}, t_k, t_k)
\end{aligned}$$

In order to test the efficiency of encoding this problem into CNF formula, we have used two types of instances, randomly generated instances and hand-crafted instances. The latter are obtained by drawing a pattern as done for Figure 1 and computing the projections in each direction in order to get an instance of **RECONSTRUCT**. Note that by solving the instance obtained, we do not necessarily get the pattern used to generate it. The uniqueness of the the pattern corresponding to some projections can be obtained by augmenting their number, but this is not the purpose of this work. We have also used test instances generated by randomly filling every cell with a prescribed probability p. Not surprisingly, some of our experiments not reported here have shown that the most difficult instances to reconstruct were the instances built using a probability $p = 0.5$.
The few experimental results presented in this section meet two aims. First, verify that the CNF encoding can be competitive with a commercial constraint programming system for solving hand-crafted instances of the discrete tomography problem. Second, address the *scalability* of the CNF encoding on the discrete tomography problem. To this end, we compare the efficiency of solving it with the CNF encoding against the efficiency of solving it using a dedicated solver. All experiments concerning the CNF encoding were done with the ”state of the art” SAT solver zchaff [15].

CNF encoding versus CHIP This comparison is based on two series of instances derived from the hand-crafted images shown figure 3. Each instance of size $n \times n$ consists of the n first lines and the n first columns of the related image.
Two solving methods are compared for each instance : CHIP V5 [5], the commercial constraint programming system from COSYTEC, and zchaff. The cardinality constraints were translated into the CHIP language by using the ChipAmong constraint [2]. The default heuristic of CHIP was used. Table 1 gives the run times required for solving each instance on a SUN workstation clocked at 450 MHz.
Clearly, thanks to the proposed CNF encoding of the cardinality constraints, our test instances can be solved with zchaff in the same run time as CHIP.

CNF encoding versus dedicated solver The preceding results show that our CNF encoding of cardinality constraints allows zchaff to outperform the general integer constraint programming system CHIP on some instances of the discrete tomography problem, but these results are restricted to a few test instances. In addition, CHIP is not specialized in solving cardinality constraints over Boolean variables.

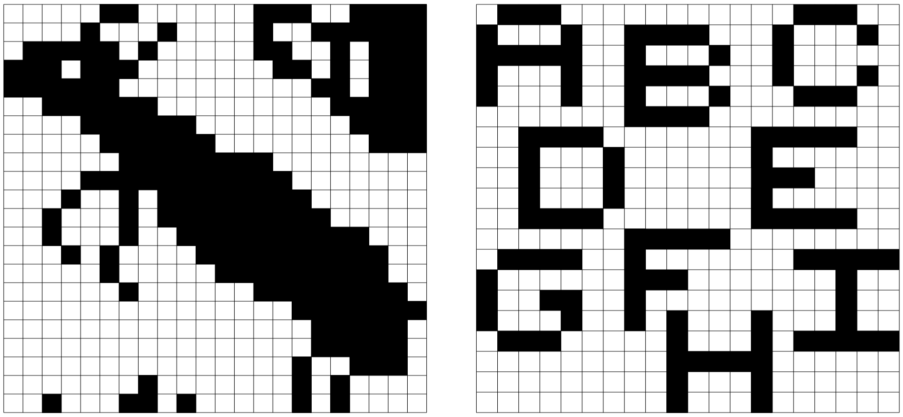


Fig. 2. The two patterns : mouse and letters, used in the comparison of CNF encoding plus zchaff versus CHIP

In order to give an idea of the scalability of the proposed encoding scheme, we will now compare it with a solver dedicated to the discrete tomography problem. To this end, we developed an enumerative solver that maintains generalized arc consistency at each node in the search tree and uses the following heuristic to select the branching variable and value:

For each projection, let P be the number of 1 not yet assigned, V be the number of 0 not yet assigned and U be the number of free variables. Of course $U = V + P$. The weight $w_1(v)w_2(v)$ is assigned to each variable v, where

- $w_1(v)$ is the sum of the $e^{(P/U)}$ for the four projections related to v,
- $w_2(v)$ is the sum of the $e^{(V/U)}$ for the four projections related to v.

Instance	CHIP V5	zchaff
mouse-12	0.04	0.09
mouse-14	0.37	0.30
mouse-16	1.60	0.41
mouse-18	2.84	4.45
mouse-20	19.8	35.4
mouse-22	>3600	92.0
letters-12	0.06	0.60
letters-14	1.41	0.66
letters-16	242	66.1
letters-18	>3600	19.8
letters-20	>3600	620

Table 1. CPU in seconds for solving instances of the discrete tomography problem with CHIP V5 and zchaff.

The branching variable v is chosen among those of maximum weight. The first branching value is 1 *iff* $w_2(v) > w_1(v)$.
Figure 3 compares the run times of zchaff and the dedicated solver on a PC running at 2 GHz, for randomly generated instances of the discrete tomography problem. It is not very surprising that the dedicated solver clearly outperforms the association of zchaff and CNF encoding, but, interestingly, the ratio between the two run times does not grow accordingly to the size of the problem. We think that such a result is extremely promising in terms of scalability and tractability of cardinality constraints under CNF encoding.

Size of the grid	Dedicated solver CPU	Encoding + zchaff CPU	ratio
15×15	0.05	2.7	54
16×16	0.04	10.6	265
17×17	0.07	19	271
18×18	0.11	24	218
19×19	0.15	56	373
20×20	0.33	85	258
21×21	0.27	127	470
22×22	1.42	136	96
23×23	1.8	136	76
24×24	4.1	215	52
25×25	8.6	267	31

Table 2. Dedicated solver versus CNF-encoding+zchaff on random patterns (100 instances for each row). The fact that the instances are satisfiable introduces great variations in the ratios. CPU is given in seconds.

4 Parity learning instances revisited

The instances arising from the parity learning problem on 32 bits have been proposed by Crawford [6] for the DIMACS challenge [12]. These instances appeared to be very challenging and none of the algorithms existing at that time were able to solve them. Later, in a paper by Selman & al [18], developing efficient algorithms for these instances is presented as one of the ten challenges in propositional reasoning and search. Following this challenge, two algorithms that solve the par32 instances were published, one by Warners and Van Maaren [20] and the other by Li [14]. The two algorithms do some specific transformations exploiting the special structure of the par32 instances.
In short, we recall only the parity learning problem expression. The reader may refer to [6] for a full and detailed description. We give the sketch of the encoding that we have made for these instances and the experimental results.
Given m sets of subscripts A_i ($1 \leq i \leq m$) such that $A_i \subseteq \{1,2,...,n\}$ and m bits y_1, y_2,...,y_m find n bits, a_1, a_2, ..., a_n such that among the m bits b_i defined as

$$b_i = y_i \oplus a_{k_1} \oplus a_{k_2} \oplus ...$$

where $k_j \in A_i$, at most e are equal to 1.

When $e = 0$ or more generally in the case where all the b_i's are assigned a value, the problem is reduced to find an assignment to an XOR-CNF formula which is known to be polynomial. Even if XOR-SAT is polynomial, a straightforward encoding of the XOR clauses, as it is done in [6], lead to a problem that is difficult for standard SAT solvers. XOR-SAT is polynomial because variable elimination can be done without adding any XOR clause. Indeed, by performing a trivial variable elimination all the a_i's can be eliminated. For example, from $b_i = y_i \oplus a_{k_1} \oplus a_{k_2} \oplus ...$, one can deduce $a_{k_1} = b_i \oplus y_i \oplus a_{k_2} \oplus ...$ and then a_{k_1} can be replaced everywhere by $(b_i \oplus y_i \oplus a_{k_2} \oplus ...)$. When $m > n$ all the a_i's can be eliminated. In general we have $m > n$, which is the case of the most difficult instances of this problem, indeed in [6] m is empirically chosen to be $m = 2n$. We obtain, then, an XOR-CNF formula, formed by the remaining $m - n$ equations, involving only the b_i's subject to the cardinality constraint that at most e of them are equal to 1. We encode then, the XOR-CNF formula using the same method described in [6] and the cardinality constraint using the method of Section 2. We denote the resulting formula by F_C. The n equations giving the a_i's are encoded as usual into a formula F_X.

It is easy to see that given the values of the b_i's, unit propagation on F_X assigns all the a_i's the correct values. Clearly, the satisfiability of the parity learning instance is equivalent to the satisfiability of F_C. The satisfiability of F_C is the hard part. By finding a solution that satisfies F_C, if any, one can derive the correct values of the a_i's by a straightforward unit propagation performed on F_X after assigning to the variables common to F_C and F_X the values they have in this solution.

One may argue that the variable elimination used in the above encoding is partly a solving procedure. An answer this argument is that an analogous preprocessing, simulating this variable elimination done on the XOR equations, can be done by a specific algorithm on the benchmarks as they were encoded by [6]. However while this variable elimination is trivial on the initial problem, discovering it on the encoded formula without knowing the initial problem, is much harder. The algorithm proposed by [20] uses a two phase algorithm as we do by separating the problem into two formulas. However, while they use a sophisticated techniques working on the CNF formulas, we use a trivial method to separate the hard part from the easy one by working directly on the initial problem. This fits with the idea, explained in the introduction, about the importance of not taking the SAT benchmarks as they are but to try to improve their encoding.

We have re-encoded the parity learning instances of the DIMACS benchmark database and we have used, to solve them, a simple Davis and Putnam procedure [8, 7] based on unit propagation and mom's heuristic for variable selection. The results are summarized in table 4, the performances of the basic DP are compared to zchaff and Eqsatz [14]. The latter has been designed for solving these instances. The aim of these experiments is to show that these instances are not intrinsically hard but their apparent hardness comes from their intial encoding.

5 Related work

The only example of polynomial size CNF encoding of Boolean cardinality constraint that we found in the literature is the one used by Crawford to propositionalize parity

Instance of parity	Before re encoding			After re encoding		
learning	Eqsatz	DP	zchaff	Eqsatz	DP	zchaff
par32-1.cnf	308	-	-	42	24	0.9
par32-2.cnf	11	-	-	37	106	205
par32-3.cnf	1241	-	-	2	49	806
par32-4.cnf	190	-	-	60	150	2
par32-5.cnf	2771	-	-	3	1	0.9

Table 3. The performances of basic DP, Eqsatz and zchaff on the parity learning instances before and after re-encoding them. The performances are given in terms of CPU time in seconds on a Pentium 2Ghz PC under Linux

learning problem [6]. Like us, Crawford uses a set of clauses that totalizes the number of bits set to 1. But because integers are represented in base two, this totalizer does not allow unit propagation to restore generalized arc consistency.

In fact, encoding constraints into CNF in such a way that unit propagation restores arc-consistency is a very recent research topic. To the best of our knowledge, the first contribution in this field is the paper of Gent [11] on the CNF encoding of binary constraints. The results of Gent are not directly comparable with ours because we do not address the same kind of constraints. It is however interesting to note that, unlike our encoding, Gent's encoding allows unit propagation to restore arc-consistency with an optimal worst case time complexity.

6 Conclusion

In this paper we proposed a new CNF encoding scheme for Boolean cardinality constraints, which allows unit propagation to maintain generalized arc consistency of the encoded constraints. We experimentally showed that, using this encoding method, a SAT solver can address discrete tomography problems and be competitive with a general constraint programming system (Cosytec CHIP), and even with a dedicated solver. We also showed that, associated with a technique of trivial variable elimination, the proposed encoding scheme allows one to drastically improve the efficiency of solving the par32 problem. This problem was hitherto considered as very hard, essentially because of its encoding.

These results confirm that in the area of solving problems under CNF encoding, the encoding scheme is as important as the solver. Then it is very important to define which properties a ”good” CNF encoding must verify. The encoding scheme proposed in this paper connects generalized arc consistency in the input problem to unit propagation in the encoded problem. As a research perspective, this could be extended to cardinality constraints on non-binary domains, like the *among* constraint of [2] or the global cardinality constraint of [16].

We think it could be useful to revisit the encoding schemes currently used in the SAT benchmarks and, in a more general way, to propose new tools for efficient CNF encoding of usual global and arithmetic constraints, in the spirit of Gent's work [11] on binary constraints.

References

1. E. Barcucci, A. DelLungo, M. Nivat, and R. Pinzani, *Reconstructing convex polyominoes from horizontal and vertical projections*, Theoret. Comput. Sci., (1996), pp. 321–347.
2. N. Beldiceanu and E. Contjean, *Introducing global constraints in CHIP*, Mathematical and Computer Modelling, 12 (1994), pp. 97–123.
3. Y. Boufkhad, O. Dubois, and M. Nivat, *Reconstructing (h, v)-convex 2-dimensional patterns of objects from approximate horizontal and vertical projections*, Theoret. Comput. Sci., 290(3) (2003), pp. 1647–1664.
4. S. Chang, *The reconstruction of binary patterns from their projections*, Comm. ACM, (1971), pp. 21–25.
5. COSYTEC SA, *CHIP C++ Library, Reference Manual, Version 5.4*, October 2001.
6. J. Crawford, *Instances of learning parity function.* http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/Benchmarks/SAT/DIMACS/PARITY/descr.html.
7. M. Davis, G. Logemann, and D. Loveland, *A machine program for theorem proving*, Communications of the ACM, 5 (1962), pp. 394–397.
8. M. Davis and H. Putnam, *A computing procedure for quantification theory*, Journal of the ACM, 7 (1960), pp. 201–215.
9. A. DelLungo, *Polyominoes defined by two vectors*, Theoret. Comput. Sci., 127 (1994), pp. 187–198.
10. R. G. Gardner, P. Gritzmann, and D. Prangenberg, *Ont the computational complexity of reconstructing lattice sets from their x-rays*, Discrete Mathematics, (1999), pp. 45–71.
11. I. P. Gent, *Arc consistency in sat*, in Proceedings of the Fifteenth European Conference on Artificial Intelligence (ECAI 2002), 2002.
12. D. Johnson and M. Trick, eds., *Second DIMACS implementation challenge : cliques, coloring and satisfiability*, vol. 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1996.
13. A. Kuba, *The reconstruction of two-directionaly connected binary patterns*, Comput. Graph. Image Process, 27 (1984), pp. 249–265.
14. C. Li, *Integrating equivalency reasoning into davis-putnam procedure*, in AAAI: 17th National Conference on Artificial Intelligence, AAAI / MIT Press, 2000.
15. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, *Chaff: Engineering an efficient sat solver*, in 39th Design Automation Conference, June 2001.
16. J.-C. Régin, *Generalized arc consistency for global cardinality constraint*, in AAAI 96, 1996, pp. 209–215.
17. H. Ryser, *Combinatorial Mathematics*, The Carus Mathematical Monographs, 1963.
18. B. Selman, H. A. Kautz, and D. A. McAllester, *Ten challenges in propositional reasoning and search*, in Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97), 1997, pp. 50–54.
19. Y. Wang, *Characterization of binary patterns and their projections*, IEEE Trans. Compt., C-24 (1975), pp. 1032–1035.
20. J. Warners and H. van Maaren, *A two phase algorithm for solving a class of hard satisfiability problems*, Op. Res. Lett., 23(3-5) (1999), pp. 81–88.
21. G. Woeginger, *The reconstruction of polyominoes from their orthogonal projections*, tech. rep., TU Graz, 1996.