Choisir un encodage CNF de contraintes de cardinalité performant pour SAT

T. Delacroix
IMT Atlantique - Dépt. LUSSI, Brest, France
thomas.delacroix@imt-atlantique.fr

Résumé

Cet article répond à une double problématique : (1) comment choisir un encodage CNF pour des contraintes de cardinalité de type $\#k(x_1, ..., x_n)$ où $\#$ peut être l'un des symboles $\leq, =, \geq$; (2) déterminer un encodage CNF performant pour les contraintes de cardinalité plus générales $\in \mathcal{K}(x_1, ..., x_n)$ où $\mathcal{K} \subset [\![0, n]\!]$. Pour ce faire, on introduit d'abord un nouvel encodage séquentiel bidirectionnel. On décrit alors un processus pour choisir l'encodage le plus performant pour une contrainte de cardinalité donnée s'appuyant sur une comparaison de différents encodages pour tous les cas possibles de valeurs n et k. Enfin, on montre que l'encodage séquentiel bidirectionnel permet de répondre à la problématique (2).

Mots Clef

CNF, SAT, encodage, contraintes de cardinalité.

Abstract

This article has a double aim : (1) define a process for choosing the most efficient CNF encoding for cardinality constraints of type $\#k(x_1, ..., x_n)$ where $\#$ is one of the following symbols $\leq, =, \geq$; (2) determine an efficient CNF encoding for the more general cardinality constraints of type $\in \mathcal{K}(x_1, ..., x_n)$ where $\mathcal{K} \subset [\![0, n]\!]$. In order to do this, we introduce a new sequential bidirectional encoding. We then describe a process for choosing the most efficient encoding for a given cardinality constraint based on a comparison of different encodings for all possible values of n and k. Finally, we show that the sequential bidirectional encoding can be used to reach our second aim.

Keywords

CNF, SAT, encoding, cardinality constraints.

1 Introduction

Parmi les solveurs modernes en programmation sous contrainte les plus performants, on trouve aujourd'hui un certain nombre de solveurs CNF-SAT. Les palmarès récents du MiniZinc Challenge en témoignent [13, 11].
Un solveur CNF-SAT permet d'obtenir une valuation pour laquelle une expression logique sous forme conjonctive normale (CNF) donnée est satisfaite lorsqu'il en existe une. Pour une contrainte particulière, la performance du solveur dépend donc à la fois de l'algorithme de résolution du solveur et à la fois de la façon dont la contrainte est exprimée sous forme CNF.

De nombreux travaux se sont donc penchés sur la question de savoir comment exprimer des contraintes classiques des CSP sous forme CNF de manière performante pour les solveurs CNF-SAT [1, 2, 3, 4, 5, 7, 10, 14, 15]. C'est le cas notamment pour les contraintes de cardinalité [1] de type $\#k(x_1, ..., x_n)$ où $\#$ peut être l'un des symboles $\leq, =, \geq$. En effet, un encodage naïf de ces contraintes contient, dès que n augmente, beaucoup trop de clauses pour pouvoir être utilisé de manière raisonnable en pratique. De nombreux encodages CNF ont donc été proposés, fonctionnant tous sur le même principe général : des variables supplémentaires sont introduites de manière à réduire drastiquement le nombre de clauses.

Parmi les travaux existants, on trouve des comparaisons des différents encodages proposés [2, 9, 12]. Toutefois, ces comparaisons ne sont pas exhaustives. En effet, l'accent y est généralement mis sur le comportement des encodages lorsque n tend vers l'infini. Or, en pratique, on peut également être amené à considérer de très nombreuses contraintes de cardinalité de faible dimension. Une comparaison exhaustive des encodages existants s'impose donc afin d'essayer d'optimiser au maximum l'étape du choix de l'encodage dans la résolution SAT et cela constitue un des éléments central de cet article. On s'aperçoit alors qu'il n'y a pas un encodage plus performant que tous les autres mais de nombreux encodages performants selon les paramètres du problème. On montre également qu'il est possible de combiner des encodages pour obtenir de meilleures performances.

Par ailleurs, on introduit dans cet article un nouvel encodage : l'encodage séquentiel bidirectionnel. Cet encodage, dont la définition est assez naturelle, est particulièrement adapté pour considérer des contraintes de cardinalité plus complexes, notamment les contraintes de cardinalité correspondant à un intervalle. On montre par ailleurs qu'il permet de considérer des contraintes de cardinalité de type $\in \mathcal{K}(x_1, ..., x_n)$ où $\mathcal{K} \subset [\![0, n]\!]$ de manière performante ce qui représente une nouveauté.

1. Le terme contrainte de cardinalité est utilisé ici selon la nomenclature standard dans le contexte SAT et ne doit pas être confondu avec d'autres notions telles que celle de *global cardinality constraint* utilisée dans le contexte de la programmation sous contrainte.

2 Encodage séquentiel bidirectionnel

Dans la suite, on considère des entiers n et k tels que $n \geq 2$ et $k \in [\![1, n-1]\!]$, les autres cas étant évidemment triviaux. La démarche qui mène à définir l'encodage proposé dans cette section s'apparente à la démarche qui mène à l'encodage séquentiel proposé par Carsten Sinz dans [12]. En effet, dans l'article précité, l'auteur définit les sommes partielles $s_i = \sum_{m=1}^{i} x_m$ et considère le j-ième bit $s_{i,j}$ de la représentation unaire de s_i. Il transpose alors ces bits en variable booléenne dans un encodage CNF pour aboutir à l'encodage ci-dessous pour la contrainte de cardinalité $\leq k(x_1, ..., x_n)$. Par la suite, on le désigne par le nom d'encodage séquentiel unidirectionnel et on le note $SeqU^n_{\leq k}$.

$$
\left.
\begin{array}{l}
(\neg x_1 \vee s_{1,1}) \\
(\neg s_{1,j}) \qquad \forall j \in [\![1, k]\!] \\
(\neg x_i \vee s_{i,1}) \\
(\neg s_{i-1,1} \vee s_{i,1}) \\
\left.
\begin{array}{l}
(\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) \\
(\neg s_{i-1,j} \vee s_{i,j}) \\
(\neg x_i \vee \neg s_{i-1,k})
\end{array}
\right\} \forall j \in [\![1, k]\!] \\
(\neg x_n \vee \neg s_{n-1,k})
\end{array}
\right\} \forall i \in [\![1, n[\![
$$

$(SeqU^n_{\leq k})$

Cependant, cette transposition contient une réduction qui aboutit à une perte d'information entre la variable $s_{i,j}$ tel qu'elle est encodée dans $SeqU^n_{\leq k}$ par rapport au bit $s_{i,j}$ décrit précédemment. En effet, le bit $s_{i,j}$ est équivalent à $s_i \geq j$. Or l'encodage $SeqU^n_{\leq k}$ donne $(s_i \geq j) \implies s_{i,j}$ mais pas l'implication réciproque. Cette perte d'information est volontaire car elle entraine un encodage plus restreint de la contrainte $\leq k(x_1, ..., x_n)$. Toutefois, ce choix n'est pas forcément judicieux lorsque l'on considère une contrainte $= k(x_1, ..., x_n)$ ou deux contraintes $\geq k_1(x_1, ..., x_n)$ et $\leq k_2(x_1, ..., x_n)$ définissant un intervalle.

L'encodage CNF qui suit permet d'encoder exactement l'ensemble des équivalences $(s_i \geq j) \iff s_{i,j}$ pour tout $i \in [\![1, n]\!]$ et pour tout $j \in [\![1, k+1]\!]$. On appelle encodage séquentiel bidirectionnel cet encodage et on le note $SeqB^n_{\#k}$.

$$
\left.
\begin{array}{l}
(x_1 \vee \neg s_{1,1}) \\
(\neg x_i \vee s_{i,1}) \, \forall i \in [\![1, n]\!] \\
(\neg s_{j-1,j}) \quad \forall j \in [\![1, k+1]\!] \\
\left.
\begin{array}{l}
(\neg s_{i-1,j} \vee s_{i,j}) \\
(x_i \vee s_{i-1,j} \vee \neg s_{i,j})
\end{array}
\right\} \forall j \in [\![1, k+1]\!] \\
\left.
\begin{array}{l}
(s_{i-1,j-1} \vee \neg s_{i,j}) \\
(\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j})
\end{array}
\right\} \forall j \in [\![1, k+1]\!]
\end{array}
\right\} \forall i \in [\![1, n]\!]
$$

$(SeqB^n_{\#k})$

À partir de cet encodage, il est très facile d'obtenir la contrainte de cardinalité $\leq k(x_1, ..., x_n)$. En effet, il suffit de rajouter la clause $\neg s_{n,k+1}$. De même, la contrainte $\geq k(x_1, ..., x_n)$ s'obtient simplement par le rajout de la clause $s_{n,k}$. Enfin, la contrainte $= k(x_1, ..., x_n)$ s'obtient par le rajout de ces deux clauses. On note $SeqB^n_{\leq k}$, $SeqB^n_{\geq k}$ et $SeqB^n_{=k}$ les encodages respectifs correspondants.

3 Choisir son encodage

3.1 Comparaisons des encodages

En plus des encodages décrits précédemment, on va également considérer l'encodage naïf ($N^n_{\leq k}$) défini par :

$$
\bigwedge_{\mathbf{i} \in \mathcal{C}_n^{k+1}} \bigvee_{j=1}^{k+1} \neg x_{i_j} \qquad (N^n_{\leq k})
$$

où $\mathbf{i} = (i_1, ..., i_{k+1})$ est une combinaison appartenant à l'ensemble $\mathcal{C}_n^{k+1}$ des combinaisons de $k+1$ éléments de $[\![1, n]\!]$. On rajoute l'encodage proposé par Bailleux & Boufkhad dans [3] ($BB^n_{\geq k_1, \leq k_2}$). On note que les auteurs précités ne donnent pas d'expression explicite de leur encodage mais décrivent plutôt un algorithme permettant de le construire.

D'autres encodages de la littérature [15] ne sont pas considérés car ils ne satisfont pas la condition de performance relative à la propagation unitaire décrite initalement dans [3] (i.e. il ne permettent pas de vérifier la contrainte sur une valuation partielle des variables x_i).

Par ailleurs, par manque de temps, nous n'avons pas intégré ici d'encodage à base de réseaux [1, 2, 7]. En effet, nous souhaitons d'abord vérifier que la génération de tels encodages est bien linéaire en leur nombre total de clauses (ou de littéraux), de manière à ce que la comparaison soit valable. Ce travail reste donc à compléter sur ce point. En effet, de tels encodages peuvent comporter un nombre total de clauses inférieur à ceux des encodages considérés ici pour un certain nombre de valeurs de n et k. Ceci est notamment le cas de l'encodage de la contrainte $\leq k(x_1, ..., x_n)$ proposé par Asín et al. dans [2] dont le nombre total de clauses est égal à :

$$
-3m + 6mK + \frac{3}{4}mK\log_2(K) + \frac{3}{4}mK\log_2(K) - 3K - \frac{3}{2}K\log_2(K)
$$

avec $K = 2^{\lceil \log_2(k) \rceil}$ et $m = \left\lceil \frac{n}{K} \right\rceil$.

Le tableau 1 donne le nombres de clauses (avec le détail en fonction de la taille en nombre de littéraux de ces clauses) ainsi que le nombre de variables auxilaires pour chacun de ces encodages. Les valeurs figurant dans ce tableau ont été recalculées à partir des descriptions de ces encodages dans les articles précités [12, 3] ainsi que le présent article.

Dans la suite de cet article, on utilise les informations de ce tableau pour permettre de choisir l'encodage le mieux adapté aux différents cas étudiés. On fera également usage de la règle suivante pour obtenir un encodage d'une contrainte $\geq k(x_1, ..., x_n)$ en considérant l'encodage pour la contrainte $\leq k(x_1, ..., x_n)$ via l'utilisation de la règle suivante :

$$
\geq k(x_1, ..., x_n) \iff \leq (n-k)(\neg x_1, ..., \neg x_n) \quad (1)
$$

De même, on pourra obtenir un encodage de la contrainte $= k(x_1, ..., x_n)$ en combinant différents encodages via l'utilisation de la constation suivante :

$$= k(x_1, ..., x_n) \iff (\leq k(x_1, ..., x_n) \wedge \geq k(x_1, ..., x_n)) \quad (2)$$

Encodage	Nombre de clauses composées de m littéraux		Nombre de variables auxiliaires
$SeqU^n_{\leq k}$	$m = 1$	$k - 1$	$nk - k$
	$m = 2$	$nk + 2n - 2k - 2$	
	$m = 3$	$nk - n - 2k + 2$	
	Total	$2nk + n - 3k - 1$	
$SeqB^n_{\#k}$	$m = 1$	k	$nk + n$
	$m = 2$	$2nk + 2n - 2k$	
	$m = 3$	$2nk + n - 2k - 1$	
	Total	$4nk + 3n - 3k - 1$	
$N^n_{\leq k}$	$m = k + 1$	$\binom{n}{k+1}$	0
$BB^n_{\geq k_1, \leq k_2}$ [2]	$m = 1$	$n - k_2 + k_1$	$n \log_2(n)$
	$m = 3$	$n^2 + 2n \log_2(n) + n - 2$	
	Total	$n^2 + 2n \log_2(n) + 2n - k_2 + k_1 - 2$	

TABLE 1 – Nombre et tailles de clauses pour chaque encodage considéré

3.2 Contrainte $\leq k(x_1, ..., x_n)$

Dans cette secion, on compare différents encodages de $\leq k(x_1, ..., x_n)$.

Conditions sur n et k par encodage			
$SeqU^n_{\leq k}$	$BB^n_{\geq 0, \leq k}$	$SeqB^{\neg n}_{\geq n-k}$	$N^n_{\leq k}$
$n \leq 5$ et $k \in$			
$\emptyset$	$\emptyset$	$\emptyset$	$[\![1, n[\![$
$6 \leq n \leq 8$ et $k \in$			
$[\![1, k_1[\![$	$\emptyset$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = n - 4$.			
$9 \leq n \leq 13$ et $k \in$			
$[\![1, k_1[\![$	$\emptyset$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = n - 3$.			
$14 \leq n \leq 30$ et $k \in$			
$[\![1, k_2[\![$	$\emptyset$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = n - 3$ et $k_2 = \lceil \frac{2}{3}(n+1) \rceil$.			
$31 \leq n \leq 36$ et $k \in$			
$[\![1, k_2[\![$	$\emptyset$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = n - 2$ et $k_2 = \lceil \frac{2}{3}(n+1) \rceil$.			
$37 \leq n$ et $k \in$			
$[\![1, k_3[\![$	$[\![k_3, k_2[\![$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = n - 2$, $k_2 = \left\lceil \frac{3n^2 - 2n \log_2(n) - 2n + 2}{4(n-1)} \right\rceil$ et $k_3 = \left\lceil \frac{n^2 + 2n \log_2(n) + n - 1}{2(n-1)} \right\rceil$.			

TABLE 2 – Encodage de $\leq k(x_1, ..., x_n)$ donnant le nombre total minimal de clauses en fonction de n et k

2. Les valeurs données ici sont exactes si n est une puissance de 2.

On remarque d'abord que pour une telle contrainte, l'encodage $SeqU^n_{\leq k}$ est clairement toujours plus performant que l'encodage $SeqB^n_{\leq k}$ donc on peut exclure ce dernier de notre comparaison. Par contre, en utilisant (1), on peut considérer l'encodage $SeqB^n_{\geq n-k}$ appliqué à $(\neg x_1, ..., \neg x_n)$. On note $SeqB^{\neg n}_{\geq n-k}$ cet encodage.
On cherche donc à comparer les encodages $N^n_{\leq k}$, $SeqU^n_{\leq k}$, $SeqB^{\neg n}_{\geq n-k}$, $BB^n_{\geq 0, \leq k}$. Une analyse complète des différents nombres totaux de clauses pour chacun de ces encodages permet de déterminer l'encodage offrant le plus petit nombre de clauses en fonction de n et k. On a réalisé cette analyse ici et regroupé les résultats dans le tableau 2. Ce tableau décrit, en fin de compte, une partition de l'ensemble des valeurs potentielles de n et k en 4 parties, chacune correspondant aux valeurs de n et k pour lesquelles l'encodage en colonne est optimal (pour le critère du nombre de clauses considéré ici).

Conditions sur n et k par encodage			
$SeqU^n_{\leq k}$	$BB^n_{\geq 0, \leq k}$	$SeqB^{\neg n}_{\geq n-k}$	$N^n_{\leq k}$
$n \leq 5$ et $k \in$			
$\emptyset$	$\emptyset$	$\emptyset$	$[\![1, n[\![$
$6 \leq n \leq 7$ et $k \in$			
$[\![1, k_1[\![$	$\emptyset$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = n - 3$.			
$8 \leq n \leq 10$ et $k \in$			
$[\![1, k_1[\![$	$\emptyset$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = n - 2$.			
$11 \leq n \leq 27$ et $k \in$			
$[\![1, k_2[\![$	$\emptyset$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = n - 2$ et $k_2 = \left\lceil \frac{10n^2 - 3n - 3}{3(5n-6)} \right\rceil$.			
$28 \leq n \leq 148$ et $k \in$			
$[\![1, k_2[\![$	$\emptyset$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = n - 1$ et $k_2 = \left\lceil \frac{10n^2 - 3n - 3}{3(5n-6)} \right\rceil$.			
$149 \leq n$ et $k \in$			
$[\![1, k_3[\![$	$[\![k_3, k_2[\![$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = n - 1$, $k_2 = \left\lceil \frac{7n^2 - 6n \log_2(n) - 6n + 4}{10(n-1)} \right\rceil$ et $k_3 = \left\lceil \frac{3n^2 + 6n \log_2(n) + 3n - 7}{5n - 8} \right\rceil$.			

TABLE 3 – Encodage de $\leq k(x_1, ..., x_n)$ donnant le nombre total minimal de littéraux en fonction de n et k

Le tableau 2 permet de choisir un encodage en fonction du nombre de clauses de cet encodage mais il ne prend pas en compte la taille de ces clauses (i.e. le nombre de littéraux par clause). Or la taille des clauses dans une contrainte CNF peut avoir une influence importante sur la rapidité d'un solveur SAT sur cette contrainte. Comme l'évaluation d'une valuation d'une contrainte CNF est linéaire par rapport à son nombre total de littéraux, on pourrait également considérer le nombre total de littéraux de chacun de encodages plutôt que leur nombre de clauses comme critère pour choisir un encodage. Même si le nombre total de clauses est généralement utilisé comme critère de comparaison des encodages dans l'état de l'art précité, on penche plutôt pour l'utilisation du nombre total de littéraux. En

tout état de cause, on présentera systèmatiquement par la suite les valeurs obtenues pour chacun de ces deux critères. Le tableau 3 permet de déterminer l'encodage avec le plus petit nombre total de littéraux.

3.3 Contrainte $\geq k(x_1, ..., x_n)$

Conditions sur n et k par encodage			
$N^{\neg n}_{\leq n-k}$	$SeqB^n_{\geq k}$	$BB^n_{\geq k,\leq n}$	$SeqU^{\neg n}_{\leq n-k}$
$n \leq 5$ et $k \in$			
$[\![1, n[\![$	$\emptyset$	$\emptyset$	$\emptyset$
$6 \leq n \leq 8$ et $k \in$			
$[\![1, 5[\![$	$\emptyset$	$\emptyset$	$[\![5, n[\![$
$9 \leq n \leq 13$ et $k \in$			
$[\![1, 4[\![$	$\emptyset$	$\emptyset$	$[\![4, n[\![$
$14 \leq n \leq 30$ et $k \in$			
$[\![1, 4[\![$	$[\![4, k_1[\![$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = \lceil \frac{n-1}{3} \rceil$.			
$31 \leq n \leq 36$ et $k \in$			
$[\![1, 3[\![$	$[\![3, k_1[\![$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = \lceil \frac{n-1}{3} \rceil$.			
$37 \leq n$ et $k \in$			
$[\![1, 3[\![$	$[\![3, k_2[\![$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = \left\lceil \frac{n^2-2n\log_2(n)-n+1}{2(n-1)} \right\rceil$ et $k_2 = \left\lceil \frac{n^2+2n\log_2(n)-2n-2}{4(n-1)} \right\rceil$.			

TABLE 4 – Encodage de $\geq k(x_1, ..., x_n)$ donnant le nombre total minimal de clauses en fonction de n et k

Conditions sur n et k par encodage			
$N^{\neg n}_{\leq n-k}$	$SeqB^n_{\geq k}$	$BB^n_{\geq k,\leq n}$	$SeqU^{\neg n}_{\leq n-k}$
$n \leq 5$ et $k \in$			
$[\![1, n[\![$	$\emptyset$	$\emptyset$	$\emptyset$
$n = 6$ et $k \in$			
$\{1, 2, 3, 5\}$	$\emptyset$	$\emptyset$	$k = 4$
$n = 7$ et $k \in$			
$[\![1, 4[\![$	$\emptyset$	$\emptyset$	$[\![4, n[\![$
$8 \leq n \leq 10$ et $k \in$			
$[\![1, 3[\![$	$\emptyset$	$\emptyset$	$[\![3, n[\![$
$11 \leq n \leq 27$ et $k \in$			
$[\![1, 3[\![$	$[\![3, k_1[\![$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = \left\lceil \frac{5n^2-15n+3}{3(5n-6)} \right\rceil$.			
$28 \leq n \leq 148$ et $k \in$			
$[\![1, 2[\![$	$[\![2, k_1[\![$	$\emptyset$	$[\![k_1, n[\![$
où $k_1 = \left\lceil \frac{5n^2-15n+3}{3(5n-6)} \right\rceil$.			
$149 \leq n$ et $k \in$			
$[\![1, 2[\![$	$[\![2, k_2[\![$	$[\![k_2, k_1[\![$	$[\![k_1, n[\![$
où $k_1 = \left\lceil \frac{2n^2-6n\log_2(n)-11n+7}{5n-8} \right\rceil$ et $k_2 = \left\lceil \frac{3n^2+6n\log_2(n)-4n-4}{10(n-1)} \right\rceil$.			

TABLE 5 – Encodage de $\geq k(x_1, ..., x_n)$ donnant le nombre total minimal de littéraux en fonction de n et k

Dans cette section, on compare 5 encodages pour la contrainte $\geq k(x_1, ..., x_n)$. Par l'équivalence (1), on obtient bien la contrainte souhaitée en appliquant les encodages $N^n_{\leq n-k}$, $SeqU^n_{\leq n-k}$ et $SeqB^n_{\leq n-k}$ à $(\neg x_1, ..., \neg x_n)$. On note respectivement $N^{\neg n}_{\leq n-k}$, $SeqU^{\neg n}_{\leq n-k}$ et $SeqB^{\neg n}_{\leq n-k}$ ces trois encodages. On considère par ailleurs les deux encodages $SeqB^n_{\geq k}$ et $BB^n_{\geq k,\leq n}$.
Comme précédemment, on détermine les encodages donnant le nombre total minimal de clauses (tableau 4) ainsi que les encodages donnant le nombre total minimal de littéraux (5) en fonction de n et k. On remarque que, dans ce cas, c'est l'encodage $SeqB^{\neg n}_{\leq n-k}$ qui est écarté systématiquement.

3.4 Contrainte $= k(x_1, ..., x_n)$

Dans cette section, on compare 7 encodages différents de $= k(x_1, ..., x_n)$ parmi lesquels 2 sont des encodages mixtes entre deux encodages différents. Il s'agit des encodages :

1. $BB^n_{=k}$ (i.e. $BB^n_{\geq k,\leq k}$) ;
2. $SeqB^n_{=k}$;
3. $SeqB^{\neg n}_{=n-k}$;
4. $SeqU^n_{=k}$ (i.e. $SeqU^n_{\leq k}$ et $SeqU^{\neg n}_{\leq n-k}$) ;
5. $N^n_{=k}$ (i.e. $N^n_{\leq k}$ et $N^{\neg n}_{\leq n-k}$) ;
6. $NS^n_{=k}$ (i.e. $N^n_{\leq k}$ et $SeqU^{\neg n}_{\leq n-k}$) ;
7. $SN^n_{=k}$ (i.e. $SeqU^n_{\leq k}$ et $N^{\neg n}_{\leq n-k}$).

Comme dans les deux sections précédentes, on a determiné les encodages donnant le nombre total de clauses minimal en fonction de n et k ainsi que celui donnant le nombre total de littéraux minimal en fonction des mêmes paramètres. Chacun de ces résultats étant difficilement synthétisable en un seul tableau, on renvoie aux annexes pour le détail pour tous les entiers $n \in [\![6, 17]\!]$.

Encodage	Condition sur k	Proportion en $n \to +\infty$
$SN^n_{=k}$	$1 \leq k < 3$	0%
$SeqB^n_{=k}$	$3 \leq k < k_2$	25%
$BB^n_{=k}$	$k_2 \leq k < k_1$	50%
$SeqB^{\neg n}_{=n-k}$	$k_1 \leq k < n-2$	25%
$NS^n_{=k}$	$n-2 \leq k < n$	0%
où $k_1 = \left\lceil \frac{3n^2-2n\log_2(n)-2n+3}{4n-3} \right\rceil$ et $k_2 = \left\lceil \frac{n^2+2n\log_2(n)-n-3}{4n-3} \right\rceil$.		

TABLE 6 – Encodage de $= k(x_1, ..., x_n)$ donnant le nombre total minimal de clauses en fonction de k pour $n \geq 18$

Pour $n \leq 5$, c'est l'encodage naïf $N^n_{=k}$ qui donne les plus petits nombres de clauses et de littéraux quel que soit k. Les deux tableaux 6 et 7 donnent les résultats pour $n \geq 18$. On remarquera que les encodages $SeqU^n_{=k}$ et $N^n_{=k}$ en sont absents. Par ailleurs, on a indiqué, pour chaque encodage, la limite de la proportion à n donné de valeurs différentes de k pour laquelle cet encodage est optimal lorsque n tend vers l'infini.

Encodage	Condition sur k	Proportion en $n \to +\infty$
$SN^n_{=k}$	$k = 1$	0%
$SeqB^n_{=k}$	$2 \leq k < k_2$	30%
$BB^n_{=k}$	$k_2 \leq k < k_1$	40%
$SeqB^{\neg n}_{=n-k}$	$k_1 \leq k < n-1$	30%
$NS^n_{=k}$	$k = n-1$	0%
où $k_1 = \left\lceil \frac{7n^2 - 6n\log_2(n) - 6n + 1}{10n-9} \right\rceil$ et $k_2 = \left\lceil \frac{3n^2 + 6n\log_2(n) - 3n - 1}{10n-9} \right\rceil$.		

TABLE 7 – Encodage de $= k(x_1, ..., x_n)$ donnant le nombre total minimal de littéraux en fonction de k pour $n \geq 18$

3.5 Contrainte $\geq k_1, \leq k_2(x_1, ..., x_n)$

On étudie ici le cas d'une contrainte donnant un intervalle. On compare 7 encodages de cette contrainte, correspondant aux cas généraux des encodages de $= k(x_1, ..., x_n)$ de la section précédente. Il s'agit des encodages :

1. $BB^n_{\geq k_1, \leq k_2}$;
2. $SeqB^n_{\geq k_1, \leq k_2}$;
3. $SeqB^{\neg n}_{\geq n-k_2, \leq n-k_1}$;
4. $SeqU^n_{\geq k_1, \leq k_2}$ (i.e. $SeqU^n_{\leq k_2}$ et $SeqU^{\neg n}_{\leq n-k_1}$);
5. $N^n_{\geq k_1, \leq k_2}$ (i.e. $N^n_{\leq k_2}$ et $N^{\neg n}_{\leq n-k_1}$);
6. $SN^n_{\geq k_1, \leq k_2}$ (i.e. $SeqU^n_{\leq k_2}$ et $N^{\neg n}_{\leq n-k_1}$);
7. $NS^n_{\geq k_1, \leq k_2}$ (i.e. $N^n_{\leq k_2}$ et $SeqU^{\neg n}_{\leq n-k_1}$).

On ne fait pas ici une comparaison exhaustive de toutes les valeurs possibles de k et n comme on a pu le faire pour les contraintes précédentes car cela nous semble trop fastidieux et difficilement exposable de manière concise. On donne simplement les expressions des nombres totaux de clauses et nombres totaux de littéraux pour chacun de ces encodages. Ces expressions peuvent être évaluées au cas par cas pour déterminer une valeur minimale et choisir l'encodage associé lorsque n est faible.

Encodage	Nombre total de clauses
$BB^n_{\geq k_1, \leq k_2}$	$n^2 + 2n\log_2(n) + 2n - k_2 + k_1 - 2$
$SeqB^n_{\geq k_1, \leq k_2}$	$4nk_2 + 3n - 3k_2 + 1$
$SeqB^{\neg n}_{\geq n-k_2, \leq n-k_1}$	$4n^2 - 4nk_1 + 3k_1 + 1$
$SeqU^n_{\geq k_1, \leq k_2}$	$2n^2 + 2n(k_2 - k_1) - n - 3(k_2 - k_1) - 2$
$N^n_{\geq k_1, \leq k_2}$	$\binom{n}{k_2+1} + \binom{n}{k_1-1}$
$SN^n_{\geq k_1, \leq k_2}$	$\binom{n}{k_1-1} + 2nk_2 + n - 3k_2 - 1$
$NS^n_{\geq k_1, \leq k_2}$	$\binom{n}{k_2+1} + 2n^2 - 2nk_1 - 2n + 3k_1 - 1$

TABLE 8 – Nombre total de clauses pour les encodages de la contrainte $\geq k_1, \leq k_2(x_1, ..., x_n)$

Sinon, en dehors de certains effets de bord quand $k_1 \leq 2$ ou bien $k_2 \geq n-2$, on peut voir que, pour n suffisamment grand, ce sont les trois premiers encodages qui sont les plus performants. On peut alors utiliser le protocole suivant pour choisir un encodage :

Si	$\min(k_2, n-k_1) \geq \frac{3}{10}n$	**choisir** $BB^n_{\geq k_1, \leq k_2}$
Sinon si	$k_2 \leq \min\left(\frac{3}{10}n, n-k_1\right)$	**choisir** $SeqB^n_{\geq k_1, \leq k_2}$
Sinon		**choisir** $SeqB^{\neg n}_{\geq n-k_2, \leq n-k_1}$

(3)

Encodage	Nombre total de littéraux
$BB^n_{\geq k_1, \leq k_2}$	$3n^2 + 6n\log_2(n) + 4n - k_2 + k_1 - 6$
$SeqB^n_{\geq k_1, \leq k_2}$	$10nk_2 + 7n - 9k_2 - 1$
$SeqB^{\neg n}_{\geq n-k_2, \leq n-k_1}$	$10n^2 - 10nk_1 - 2n + 9k_1 - 1$
$SeqU^n_{\geq k_1, \leq k_2}$	$5n^2 + 5n(k_2 - k_1) - 7n - 9(k_2 - k_1) + 2$
$N^n_{\geq k_1, \leq k_2}$	$(n-k_2)\binom{n}{k_2} + k_1\binom{n}{k_1}$
$SN^n_{\geq k_1, \leq k_2}$	$k_1\binom{n}{k_1} + 5nk_2 + n - 9k_2 + 1$
$NS^n_{\geq k_1, \leq k_2}$	$(n-k_2)\binom{n}{k_2} + 5n^2 - 5nk_1 - 8n + 9k_1 + 1$

TABLE 9 – Nombre total de littéraux pour les encodages de la contrainte $\geq k_1, \leq k_2(x_1, ..., x_n)$

Le protocole ci-dessus utilise le critère du nombre total de littéraux minimal. Pour utiliser le critère du nombre total de clauses, il suffit de remplacer $\frac{3}{10}$ par $\frac{1}{4}$.

3.6 Contrainte $\in \mathcal{K}(x_1, ..., x_n)$

Dans cette section, on considère un sous-ensemble $\mathcal{K} = \{k_1, ..., k_m\} \subset [\![0, n]\!]$ avec $m \geq 2$ et $k_1 < k_2 < ... < k_m$. Le cas général d'une contrainte d'appartenance du cardinal à un sous-ensemble $\mathcal{K}$ se distingue entièrement des cas précédents. En effet, la proposition $k \in \mathcal{K}$ est équivalente à la disjonction $\bigvee_{i=1}^{m} [k = k_i]$. Ainsi, à part le cas particulier d'un intervalle traité précédemment, on ne peut pas se ramener directement à une conjonction des encodages CNF décrits précédemment. Or obtenir une contrainte CNF à partir d'une disjonction de contraintes CNF nécessite une opération supplémentaire. Effectuer cette opération est prohibitif si l'on considère directement des disjonctions des encodages précédents. Toutefois, on peut également considérer ces disjonctions au sein d'un encodage séquentiel bidirectionnel unique ce qui permet de rendre cette opération tout à fait envisageable.

En effet, la contrainte suivante est bien un encodage de la contrainte $\in \mathcal{K}(x_1, ..., x_n)$.

$$SeqB^n_{\#} \wedge \left(\bigvee_{i=1}^{m} (s_{n,k_i} \wedge \neg s_{n,k_i+1}) \right) \tag{4}$$

Cette encodage n'est pas une contrainte CNF mais on peut se ramener à une contrainte CNF équisatisfaisable facilement en introduisant m variables supplémentaires y_0, ..., y_{m-1} [8]. On définit ainsi l'encodage $SeqB^n_{\in \mathcal{K}}$ de la contrainte $\in \mathcal{K}(x_1, ..., x_n)$ comme étant la contrainte CNF ci-dessous :

$$SeqB^n_{\#} \quad \wedge \quad \bigwedge_{i=1}^{m} \left(s_{n,k_i} \vee \neg y_{i-1} \vee \left(\bigvee_{j=i}^{m-1} y_j \right) \right)$$
$$\wedge \quad \bigwedge_{i=1}^{m} \left(\neg s_{n,k_i+1} \vee \neg y_{i-1} \vee \left(\bigvee_{j=i}^{m-1} y_j \right) \right)$$
$$\wedge \quad y_0$$
$$(SeqB^n_{\in \mathcal{K}})$$

Cette contrainte a un nombre total de clauses égal à :

$$4nk_m + 3n - 3k_m + 2m \tag{5}$$

et un nombre total de littéraux égal à :

$$10nk_m + 7n - 9k_m + m^2 + 3m - 2 \tag{6}$$

Enfin, on peut remarquer que l'encodage $SeqB^n_{\in \overline{\mathcal{K}}}$ (où $\overline{\mathcal{K}}$ est le complèmentaire de $\mathcal{K}$) appliqué à $(\neg x_1, ..., \neg x_n)$ donne également la contrainte $\in \mathcal{K}(x_1, ..., x_n)$. On note $SeqB^{\neg n}_{\in \overline{\mathcal{K}}}$ cet encodage. Il peut éventuellement être plus performant que l'encodage précédent selon $\mathcal{K}$. Son nombre total de clauses et son nombre total de littéraux s'obtiennent en remplaçant k_m par $\max(\overline{\mathcal{K}})$ et m par $n - m$ dans (5) et (6).

Le pire des cas est atteint pour une contrainte de type k pair. On a alors un nombre de clauses de l'ordre de $4n^2$ et un nombre de littéraux de l'ordre de $10.5n^2$ ce qui est tout à fait raisonnable. Par ailleurs, à notre connaissance, il s'agit du seul encodage CNF de la littérature de la contrainte $\in \mathcal{K}(x_1, ..., x_n)$.

4 Conclusion

Lorsque l'on recherche un algorithme optimisé pour un problème donné, il est rare de découvrir un seul algorithme qui soit le plus optimisé pour chacune des occurrences de ce problème. Pour le problème d'une résolution CNF-SAT d'une contrainte de cardinalité, on peut remarquer que chacun des encodages d'une contrainte de cardinalité considérés dans cet article est préférable aux autres pour au moins quelques valeurs différentes de k et n. Par ailleurs, les comportements des différentes solutions à l'infini ne préjugent en rien quant à leurs comportements en faible dimension. Par exemple dans le cas des contraintes $\leq k(x_1, ..., x_n)$ et $\geq k(x_1, ..., x_n)$, l'encodage proposé par Bailleux et Boufkhad est préférable (selon le critère du nombre total de littéraux) dans environ 10% des cas lorsque n est très grand mais n'est préférable en aucun cas lorsque $n < 149$. Afin de pouvoir déterminer l'encodage réellement le mieux adapté à une contrainte et des paramètres donnés, il est nécessaire de passer par une étude exhaustive des cas comme cela a été réalisé dans cet article.

On pourrait objecter que, si n est petit alors la taille de l'encodage le sera également et qu'en conséquence, même si on reste loin de l'optimum possible, on pourra négliger l'accroissement. Cet argument ne tient toutefois pas longtemps car un gain fixe sur une opération peut être non négligeable, d'autant plus si cette opération est répétée de nombreuses fois au cours d'un même processus.

Par ailleurs, le nouvel encodage séquentiel bidirectionnel qui est proposé dans cet article offre une performance nettement améliorée dans certains cas ainsi que des perspectives nouvelles.

En effet, dans le cas d'une contrainte de type $= k(x_1, ..., x_n)$, les nombres totaux de clauses et de littéraux sont au mieux quadratiques en n pour les autres encodages et ceci quel que soit k. Toutefois, dans un certain nombre de problèmes, k est fixé indépendamment de n. Les nombres totaux de clauses et de littéraux de l'encodage séquentiel bidirectionnel sont alors linéaires en n. Cet encodage a ainsi pu être mis à profit dans le cas de la résolution SAT d'un problème d'emploi du temps en BTS [6] qui comporte un certain nombre de contraintes de type $= 3(x_1, ..., x_n)$.

Cet encodage offre également de nouvelles perspectives via son application aux contraintes de type $\in \mathcal{K}(x_1, ..., x_n)$ pour lesquelles il n'existait pas avant, à notre connaissance, d'encodage performant.

Enfin, on peut noter que sa présentation explicite (semblable à celle de l'encodage séquentiel dans [12] et qui a sûrement contribué à sa popularité) en permet une implémentation directe.

Le travail de recherche présenté dans cet article est d'ordre théorique. Si les critères du nombre total de clauses ou du nombre total de littéraux sont généralement pertinents, ils ne suffisent pas à déterminer l'encodage le mieux adapté qui pourra dépendre non seulement du problème considéré mais également du solveur utilisé. Il appelle donc d'autres travaux qui permettront de mettre en pratique (tels que [6]) et de valider ces résultats. Il sera également prolongé de manière à intégrer au processus de choix des encodages à base de réseaux dont l'encodage de $\leq k(x_1, ..., x_n)$ défini dans [2].

5 Annexe

k \ n	6	7	8	9	10	11	12	13	14	15	16	17
1	6	6	6	6	6	6	6	6	6	6	6	6
2	5	6	6	6	6	6	6	6	6	6	6	6
3	5	5	6	6	6	6	6	6	6	6	6	6
4	5	5	5	4	2	2	2	2	2	2	2	2
5	7	7	7	4	1	1	1	2	2	2	2	2
6		7	7	7	3	1	1	1	1	1	1	2
7			7	7	7	3	1	1	1	1	1	1
8				7	7	7	3	3	1	1	1	1
9					7	7	7	3	3	1	1	1
10						7	7	7	3	3	1	1
11							7	7	7	3	3	3
12								7	7	7	3	3
13									7	7	7	3
14										7	7	7
15											7	7
16												7

TABLE 10 – Encodage de la contrainte $= k(x_1, ..., x_n)$ donnant le nombre total minimal de clauses pour $n \in [\![6, 17]\!]$

Les tableaux 10 et 11 donnent, en fonction de k et n, les encodages de $= k(x_1, ..., x_n)$ pour $n \in [\![6, 17]\!]$ ayant un

k \ n	6	7	8	9	10	11	12	13	14	15	16	17
1	6	6	6	6	6	6	6	6	6	6	6	6
2	6	6	6	6	6	6	6	6	6	6	6	2
3	5	6	4	2	2	2	2	2	2	2	2	2
4	7	7	4	4	4	2	2	2	2	2	2	2
5	5	7	4	4	4	4	2	2	2	2	2	2
6		7	7	3	4	4	4	4	2	2	2	2
7			7	7	3	3	3	4	4	4	2	2
8				7	7	7	3	3	3	4	4	4
9					7	7	3	3	3	3	3	4
10						7	7	3	3	3	3	3
11							7	7	3	3	3	3
12								7	7	3	3	3
13									7	7	3	3
14										7	3	3
15											7	3
16												7

TABLE 11 – Encodage de la contrainte $= k(x_1, ..., x_n)$ donnant le nombre total minimal de littéraux pour $n \in [\![6, 17]\!]$

nombre total minimal de clauses et de littéraux respectivement. Les encodages sont indiqués dans le tableau par un numéro qui correspond à leur numérotation dans la section 3.4.

Références

[1] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks and their applications. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 167–180. Springer, 2009.

[2] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks : a theoretical and empirical study. *Constraints*, 16(2) :195–221, 2011.

[3] O. Bailleux and Y. Boufkhad. Efficient cnf encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 108–122. Springer, 2003.

[4] O. Bailleux, Y. Boufkhad, and O. Roussel. A translation of pseudo-boolean constraints to sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2 :191–200, 2006.

[5] O. Bailleux, Y. Boufkhad, and O. Roussel. New encodings of pseudo-boolean constraints into cnf. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 181–194. Springer, 2009.

[6] T. Delacroix. Planifier l'épreuve e5 à l'aide d'un solveur sat. In *APIA, Conférence Nationale sur les Applications Pratiques de l'Intelligence Artificielle*, 2018.

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

[8] ENS. Concours d'admission - composition d'informatique - a -, 2016.

[9] A. M. Frisch and P. A. Giannaros. Sat encodings of the at-most-k constraint. some old, some new, some fast, some slow. In *Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation*, 2010.

[10] J. Marques-Silva and I. Lynce. Towards robust cnf encodings of cardinality constraints. *Principles and Practice of Constraint Programming–CP 2007*, pages 483–497, 2007.

[11] MiniZinc. Minizinc challenge 2017 results, 2017.

[12] C. Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. *CP*, 3709 :827–831, 2005.

[13] P. J. Stuckey, T. Feydy, A. Schutt, G. Tack, and J. Fischer. The minizinc challenge 2008–2013. *AI Magazine*, 35(2) :55–60, 2014.

[14] N. Tamura, M. Banbara, and T. Soh. Compiling pseudo-boolean constraints to sat with order encoding. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 1020–1027. IEEE, 2013.

[15] J. P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2) :63–69, 1998.