Satisfiabilité pour le décodage par syndrome

Carl Berton, Sami Cherif, Claire Delaplace

Laboratoire MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

carl.berton@etud.u-picardie.fr, {sami.cherif;claire.delaplace}@u-picardie.fr

Résumé

Le problème du décodage par syndrome d'un code linéaire binaire consiste à trouver une solution de faible poids de Hamming pour un système linéaire bruité défini sur le corps fini à deux éléments. Dans cet article, nous explorons plusieurs modélisations de ce problème basées sur la satisfiabilité. La première repose sur une représentation XNF, qui combine des contraintes clausales utilisant à la fois l'opérateur de disjonction classique (OR) et l'opérateur de disjonction exclusive (XOR). Ce modèle est exploité par des solveurs de satisfiabilité dédiés aux problèmes cryptographiques. Un second modèle proposé repose uniquement sur des contraintes clausales classiques, générant ainsi des formules CNF qui pourraient être résolues à l'aide des solveurs SAT de l'état de l'art. Nous comparons ces approches afin d'évaluer leur efficacité dans la résolution du problème du décodage par syndrome.

Mots-clés

Décodage par syndrome, Satisfiabilité, Cryptographie

Abstract

The syndrome decoding problem for a linear binary code consists in finding a low Hamming weight solution to a noisy linear system defined over the finite field of two elements. In this paper, we explore several SAT-based models for solving this problem. The first approach relies on an XNF representation, which combines clausal constraints using both classical disjunction (OR) and exclusive disjunction (XOR). This model is designed for SAT solvers specialized in cryptographic problems. The second proposed model relies solely on classical clausal constraints, generating CNF formulas that can be solved using state-of-the-art SAT solvers. We compare these approaches to assess their efficiency in solving the syndrome decoding problem.

Keywords

Syndrome Decoding, Satisfiability, Cryptography

1 Introduction

Le problème de satisfiabilité propositionnelle (SAT) consiste à déterminer, étant donné une formule en Forme Normale Conjonctive (CNF), s'il existe une affectation des variables qui la satisfait [8]. SAT est un problème de décision fondamental en informatique et en intelligence artificielle, utilisé pour résoudre une large gamme de problèmes dans divers domaines, tels que la cryptographie [17, 26, 28], la vérification matérielle et logicielle [15], la planification [23], et bien d'autres. Il a été le premier problème à avoir été démontré comme étant NP-complet [9]. Certains problèmes, notamment en cryptanalyse, sont naturellement formulés sous une autre forme appelée forme normale algébrique (ANF). Bien qu'une conversion en forme normale conjonctive (CNF) soit possible, elle peut provoquer une explosion combinatoire du nombre de variables et de clauses [16], compromettant ainsi l'efficacité de la résolution. Pour pallier cette limitation, certains solveurs, tels que CryptoMiniSat [27], utilisent une structure hybride de formules, combinant à la fois des clauses CNF classiques et des clauses exprimées à l'aide de l'opérateur de disjonction exclusive (XOR), un format connu sous le nom de XNF (XOR-CNF) [22].

Dans ce contexte, nous nous intéressons aux problèmes de décodage. En effet, pour assurer la transmission fiable d'informations sur des canaux bruités, on utilise des codes correcteurs, notamment des codes linéaires binaires, qui ajoutent une redondance pour détecter et corriger les erreurs [24]. Les codes utilisés en théorie de l'information sont construits de manière à ce que ce problème soit facile à résoudre, afin de garder une communication fluide. Cependant, décoder un code linéaire binaire dans le cas général est un problème NP-difficile [3]. Les cryptologues se sont intéressés à ce problème dès les années 70 [19], quand McEliece a mis en place un premier schéma de chiffrement à clé publique dont la sécurité repose sur la difficulté de décoder un code linéaire binaire. Avec l'avènement de la cryptographie post-quantique, la cryptographie à base de code s'est beaucoup développée au cours des dernières années avec de nombreux efforts à la fois du point de vue des constructions et des attaques [20, 11, 21].

Un problème qui intéresse particulièrement les cryptologues est celui du décodage par syndrome qui consiste à retrouver le bruit à partir de la matrice de parité du code [4]. Concrètement, ce problème revient à résoudre un système linéaire sous-déterminé modulo 2 avec la contrainte que la solution doit être de poids de Hamming faible. Bien que ce problème semble se modéliser facilement sous forme de contrainte XNF, il n'existe à notre connaissance aucuns travaux qui visent à résoudre le problème du décodage par syndrome en utilisant des solveurs SAT. À noter que d'autres problèmes similaires, tel que le décodage par maximum de vraisemblance, sont connus pour se modéliser sous

forme de problèmes de programmation linéaire en nombres entiers (PLNE) [12]. Il faut toutefois remarquer que ces travaux se focalisent sur des codes particuliers, à savoir des codes type turbo [5] et des codes de parité à faible densité (LDPC) [13], pour lesquels le décodage est plus facile que dans le cas aléatoire qui nous intéresse dans ce papier.

Dans cet article, nous présentons des travaux préliminaires visant à étudier deux approches de modélisation pour le problème de décodage par syndrome. La première repose sur une représentation XNF, qui permet d'exprimer directement les contraintes de parité, complétée par des contraintes clausales classiques pour intégrer la contrainte de poids sur le vecteur d'erreur. La seconde approche consiste à formuler le problème sous un format CNF classique, en proposant deux modélisations alternatives où les contraintes XOR sont réécrites à l'aide de contraintes Pseudo-Booléennes (PB). Cette dernière approche présente l'avantage d'intégrer plus naturellement la spécification du poids d'erreur lors de la reformulation des XOR et donc de réduire la combinatoire de l'espace de recherche. Il est largement admis que les représentations clausales classiques sont moins efficaces que les représentations XNF pour les problèmes cryptographiques. Notre objectif est donc de comparer une modélisation XNF standard à une approche CNF enrichie par des contraintes Pseudo-Booléennes, afin d'évaluer leur performances respectives et d'identifier si une représentation CNF classique peut rester pertinente pour certains types de problèmes cryptographiques comme le décodage par syndrôme.

Cet article est organisé comme suit. Nous introduisons dans la section 2 les notions fondamentales nécessaires à notre étude, en rappelant les différentes représentations clausales utilisées, avant de présenter en détail le problème du décodage par syndrome. Ensuite, nous poursuivons dans la section 3 avec la modélisation de ce problème à la fois en format XNF et CNF classique. Nous analysons les résultats expérimentaux obtenus afin de comparer ces différentes approches dans la section 4. Enfin, on conclut et discute des travaux futurs dans la section 5.

2 Préliminaires

2.1 Satisfiabilité

Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble de variables booléennes qui peuvent prendre les valeurs $Vrai$ ($\top$ ou 1) ou $Faux$ ($\bot$ ou 0). Un littéral est soit une variable $x \in X$, soit sa négation $\overline{x}$. Une formule CNF ϕ est une conjonction de clauses $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, où chaque clause C_j est une disjonction ($\vee$) de littéraux. Une affectation $\alpha : X \to \{True, False\}$ est une fonction associant à chaque variable de X une valeur de vérité. Ainsi, une clause est satisfaite par une interprétation α si au moins l'un de ses littéraux est satisfait par α. Une formule CNF ϕ est satisfaite par une interprétation α si toute ses clauses sont satisfaites par α. Dans ce cas, on dit que la formule est satisfiable et que α est un modèle de ϕ. Le problème de satisfiabilité propositonnelle (SAT) consite à determiner si une formule CNF donnée est satisfiable. Les solveurs SAT modernes sont extrêmement performants et parviennent à résoudre des formules comportant un grand nombre de variables et de clauses en un temps remarquablement court, malgré la difficulté théorique établie du problème. En effet, ces solveurs sont basés sur l'algorithme CDCL [25] (Conflict-Driven Clause Learning) et emploient donc des mécanismes puissants comme l'analyse de conflits , les redémarrages ou encore des heuristiques de branchement dédiées [8].

Une autre forme de représentation particulièrement utilisée pour les problèmes cryptographiques dans le cadre de la satisfiabilité est le format XNF (XOR-CNF). Contrairement aux formules CNF classiques, qui utilisent uniquement des contraintes clausales avec des opérations de disjonction logique, les XNF emploient en plus des clauses basées sur l'opérateur XOR (disjonction exclusive). Une formule XNF est donc une conjonction de clauses CNF et XOR, où chaque clause XOR est une disjonction exclusive de littéraux. Une clause XOR s'écrit donc sous la forme $C = l_1 \oplus l_2 \oplus \cdots \oplus l_k$ où chaque l_i est un littéral, et $\oplus$ représente l'opérateur XOR. Il est possible d'ajouter la constante $\top$ à une clause pour indiquer un littéral qui est toujours satisfait. Contrairement à l'opération OR, une clause XOR est satisfaite par une interprétation α si et seulement si un nombre impair de ses littéraux sont satisfaits. Une formule XNF est donc satisfaite par α si toutes ses clauses classiques et XOR sont satisfaites.

Dans nos modèles, nous utiliserons également des contraintes Pseudo-Booléennes (PB) qui imposent une borne sur une somme pondérée de littéraux comme suit :

$$\left(\sum_{i=1}^{h} a_i \times l_i\right) \circ k$$

où $a_i, k \in \mathbb{N}$ et $\circ \in \{\leq, =, \geq\}$. Dans le cas où tous les poids sont fixés à 1 (et peuvent donc être omis), de telles contraintes sont plus couramment appelées contraintes de cardinalité. Les contraintes PB et de cardinalité sont typiquement utilisées dans les solveurs pour imposer des bornes pertinentes lors de la recherche et peuvent être encodées efficacement sous forme clausale [6].

2.2 Le Problème du Décodage par Syndrome

En communications numériques, la transmission fiable d'informations est essentielle. Les canaux de transmission sont souvent bruités, ce qui engendre des erreurs dans les messages reçus. Pour y remédier, on utilise des codes correcteurs, notamment des codes linéaires binaires, qui intègrent une redondance dans la transmission [24].

Soit $\mathbb{F}_2$ le corps fini à deux éléments et soient n, k, et t des entiers tels que $0 < k < n$ et $0 < t < \frac{n}{2}$. Pour tout vecteur $v \in \mathbb{F}_2^n$, on note $\mathrm{wt}(v)$ son poids de Hamming, c'est-à-dire le nombre de coefficients non nuls de v. Étant donné une matrice H de taille $(n-k) \times n$ à coefficients dans $\mathbb{F}_2$, le code linéaire binaire C de matrice de parité H est défini par l'ensemble des vecteurs v appartenant au noyau à droite de H. Autrement dit,

$$C = \{v \in \mathbb{F}_2^n : Hv = 0\} \tag{1}$$

Soit $u \in \mathbb{F}_2^n$, u peut s'exprimer sous la forme :

$$u = v + e$$

où $v \in C$ est un mot de code et e est un vecteur d'erreur, qu'on suppose généralement suffisamment petit pour que le décodage fonctionne. Typiquement, si on se place dans le cadre d'un décodage à distance complète, on a $wt(e) \leq d$ avec :

$$d = \min\{wt(u+v) : (u,v) \in C^2, u \neq v\}$$

On appelle syndrome de u le vecteur $s \in \mathbb{F}_2^{(n-k)}$ tel que

$$Hu = s$$

Comme $v \in C$, on a $Hu = H(v+e) = Hv + He = He$. Autrement dit, le syndrome de u est identique à celui de e. Le décodage par syndrome consiste à décoder u en effectuant les étapes suivantes :

1. Calculer le syndrome $s = Hu$.
2. Résoudre le problème de décodage par syndrome, c'est à dire trouver un vecteur $e \in \mathbb{F}_2^n$ tel que $He = s$ et $\mathrm{wt}(e) \leq t$.
3. Récupérer le mot de code corrigé $v = u + e$.

Il est clair que la difficulté de ce problème se trouve à l'étape 2. Ici, il est demandé de trouver une solution de poids de Hamming faible à un système linéaire sous-déterminé. Il existe des codes particulier pour lesquels ce problème est facile à résoudre, mais dans le cas général, il est NP-difficile. Formellement nous définissons le problème de la manière suivante :

Définition 1 (décodage par syndrome à distance complète) *Étant donné un code linéaire binaire C de longueur n, de dimension k et de distance minimale d représenté par sa matrice de parité $H \in \mathbb{F}_2^{(n-k)\times n}$, un paramètre $t < d$ et un vecteur $s \in \mathbb{F}_2^{n-k}$, resoudre le problème du décodage par syndrome consiste à retrouver $e \in \mathbb{F}_2^n$ tel que $He = s$ et $wt(e) \leq t$.*

Si on considère plutôt des représentants sur $\mathbb{Z}$ des éléments de $\mathbb{F}_2$, un système de décodage par Syndrome, qu'on appellera $Syndrome(k,n,t)$, peut s'écrire comme suit :

$$He = s \mod 2 \tag{2a}$$

$$\sum_{i=0}^{n} e_i \leq t, \tag{2b}$$

où H est une matrice de $n-k$ lignes et n colonnes à coefficients entiers, et e et s sont des vecteurs d'entiers respectivement de n et $n-k$ coefficients. On note $H_{i,j}$ le coefficient sur la i-ème ligne, j-ème colonne de H. Plus précisément, $Syndrome(k,n,t)$ est défini comme suit :

$$E_i : \sum_{\substack{0 \leq j \leq n-1 \\ H_{i,j}=1}} e_j = s_i \mod 2 \quad \forall 0 \leq i \leq n-k-1 \tag{3a}$$

$$\sum_{i=0}^{n-1} e_i \leq t \tag{3b}$$

Les cryptologues s'intéressent en particulier au cas des codes linéaires binaires aléatoires. Ces codes sont connus pour avoir une distance minimale qui atteint la borne de Gilbert-Varshamov, notée d_{GV} et qui satisfait

$$\frac{k}{n} = 1 - \mathcal{H}\left(\frac{d_{GV}}{n}\right),$$

où $\mathcal{H}$ est la fonction d'entropie binaire :

$$\begin{aligned} H :& [0,1] \to [0,1] \\ & x \mapsto -x\log_2(x) - (1-x)\log_2(1-x). \end{aligned}$$

Si de plus le débit du code k/n est proche de 1/2, alors cette instance du problème (1) est difficile.

3 Modélisation du problème de décodage par syndrome

Dans cette section, on s'intéresse à la modélisation du problème de décodage par syndrome grâce à la satisfiabilité propositionnelle. En particulier on introduit des modélisations basées à la fois sur les formats XNF et CNF.

3.1 Modélisation XNF

Le système de décodage par syndrome peut être modélisé en format XNF en utilisant l'opérateur XOR pour vérifier la parité dans les équations E_i où $0 \leq i \leq n-k-1$. Pour simplifier les notations, on dénote $m = n-k-1$ et V_{E_i} l'ensemble des indices des variables présentes dans l'équation E_i. Plus formellement, pour tout $i \in \{0,\ldots,m\}$, on a $V_{E_i} = \{j \mid 0 \leq j \leq n-1 \text{ et } H_{i,j} = 1\}$. Le système $Syndrome(k,n,t)$ peut donc s'écrire sous la forme suivante :

$$\left(\bigoplus_{j \in V_{E_i}} e_j\right) \underset{s_i=0}{\oplus} \top \qquad \forall 0 \leq i \leq m \tag{4a}$$

$$\sum_{j=0}^{n-1} e_j \leq t \tag{4b}$$

On peut également remarquer que lorsque la taille d'une clause XOR est supérieure au poids t, le solveur doit se fier à la contrainte (4b) impliquant des variables qui pourrait apparaître dans d'autres clauses XOR pour éliminer des sous-espaces de la recherche où il ne peut pas y avoir de solution. Pour l'aider à raisonner directement sur les variables d'une seule clause, on peut donc rajouter les contraintes redondantes suivantes :

$$\sum_{j \in V_{E_i}} e_j \leq t \qquad \forall 0 \leq i \leq m \text{ tel que } |V_{E_i}| > t \tag{5}$$

3.2 Modélisation CNF

Il est largement admis que les représentations clausales (avec disjonction classique) sont moins efficaces sur les problèmes cryptographiques que les représentations XNF,

où on admet des clauses avec des disjonctions exclusives. En effet, une conversion classique d'une clause XOR en format CNF sans ajout de variables additionnelles peut avoir un coût exponentiel [16]. Dans notre cas, on veut introduire des modèles CNF dédiés au problème du décodage par Syndrome et qui seraient compétitifs.
Pour cela, on définit, pour $0 \leq i \leq m$, l'ensemble suivant :

$$K_{E_i} = \{0 \leq j \leq \min(|V_{E_i}|, t) \mid j \mod 2 = s_i\}$$

Clairement, ces ensembles représentent les valeurs possibles que peuvent prendre les sommes $\sum_{j \in V_{E_i}} e_j$ dans une équation E_i donnée selon la parité indiquée par s_i. Il est important de noter que ces ensembles prennent naturellement en compte la valeur du poids t pour éliminer les sommes qui ne produisent pas une solution faisable. Ainsi, pour représenter le choix des valeurs possibles dans ces sommes dans les différentes équations, on introduit les variables $x_{i,v}$ pour tout $(i, v) \in \{0, \ldots, m\} \times K_{E_i}$, qui sont affectés à vrai si et seulement si la somme des variables dans l'équation E_i est égale à v. Plus formellement, $x_{i,v}$ est affectée à vrai si et seulement si $\sum_{j \in V_{E_i}} e_j = v$. Le système $Syndrome(k, n, t)$ peut donc se réécrire comme suit :

$$\sum_{j \in V_{E_i}} e_j = \sum_{v \in K_{E_i} \setminus \{0\}} v \cdot x_{i,v} \qquad \forall 0 \leq i \leq m \qquad (6a)$$

$$\sum_{v \in K_{E_i}} x_{i,v} = 1 \qquad \forall 0 \leq i \leq m \qquad (6b)$$

$$\sum_{i=0}^{n-1} e_i \leq t \qquad (6c)$$

La contrainte (6a) assure que la somme des variables dans une équation est égale à l'une des valeurs possibles selon sa parité tandis que la contrainte (6b) est nécessaire pour s'assurer qu'une seule et unique valeur est sélectionnée à la fois pour une équation donnée. On peut réécrire (6a) sous forme de contrainte Pseudo-Booléenne (PB) comme suit :

$$\sum_{j \in V_{E_i}} e_j + \sum_{v \in K_{E_i} \setminus \{0\}} v \cdot \overline{x}_{i,v} = \sum_{v \in K_{E_i}} v \qquad \forall 0 \leq i \leq m \qquad (6a')$$

La représentation ainsi définie peut s'apparenter à la représentation par polytopes introduite dans [10, 29], mais en rajoutant moins de variables et en prenant en compte naturellement le poids induit par le problème de décodage par syndrome dans chaque conversion de clause XOR.
On peut également remarquer que les éléments de l'ensemble K_{E_i} peuvent être déterminés par incréments de 2, vu qu'on raisonne sur la parité. Pour tirer parti de cette caractéristique, on peut adapter le sens sémantique des variables $x_{i,v}$ pour représenter le fait que la somme des variables dans l'équation E_i est inférieure ou égale à v. Plus formellement, $x_{i,v}$ est affectée à vrai si et seulement si $\sum_{j \in V_{E_i}} e_j \leq v$. Cela nous permet donc de reformuler le système syndrome comme suit :

$$\sum_{j \in V_{E_i}} e_j = \sum_{v \in K_{E_i} \setminus \{0,1\}} 2 \cdot x_{i,v} + s_i \qquad \forall 0 \leq i \leq m \qquad (7a)$$

$$\overline{x}_{i,v} \vee x_{i,v-2} \qquad \forall 0 \leq i \leq m, v \in K_{E_i} \setminus \{0, 1, 2, 3\} \qquad (7b)$$

$$\sum_{i=0}^{n-1} e_i \leq t \qquad (7c)$$

La contrainte (7a) constitue désormais une représentation plus naturelle de la parité des sommes, permettant une meilleure intégration des contraintes de parité dans la modélisation CNF du problème de décodage par syndrome. On peut également la réécrire naturellement sous forme de contrainte PB comme indiquée ci-dessous, où max désigne l'opérateur usuel renvoyant l'élément maximal dans un ensemble donné. Quant à la contrainte (7b), elle assure que les valeurs possibles des sommes sont bien sélectionnées de manière cohérente et ordonnée, c'est à dire $x_{i,v} \rightarrow x_{i,v-2}$, en respectant l'incrémentation par 2 imposée par la structure des ensembles K_{E_i}.

$$\sum_{j \in V_{E_i}} e_j + \sum_{v \in K_{E_i} \setminus \{0,1\}} 2 \cdot \overline{x}_{i,v} = max\ K_{E_i} \qquad \forall 0 \leq i \leq m \qquad (7a')$$

Cette représentation se rapproche ainsi davantage de celle proposée dans [14] pour la certification des raisonnements de parité. A noter que la deuxième représentation est non seulement plus naturelle mais elle permet également de réduire la complexité de l'encodage. En effet, la complexité de l'encodage d'une contrainte PB est souvent régie par le nombre de littéraux à sommer et le poids maximal des littéraux. Dans notre cas, le nombre de littéraux à sommer reste de même ordre dans les deux modèles. Plus formellement, pour $0 \leq i \leq m$, on peut remarquer que $|K_{E_i}| \leq |V_{E_i}| \leq n$ où n désigne la longueur du code et ainsi le nombre de littéraux sommés est de l'ordre de $O(n)$ dans les deux modèles CNF. Cependant, dans le premier modèle CNF, le poids maximal est $max\ K_{E_i}$, pouvant atteindre n, tandis que le poids maximal du deuxième modèle est toujours fixe et égal à 2.

4 Évaluation Expérimentale

4.1 Protocole expérimental

Pour évaluer les modèles introduits dans la section précédente, nous avons testé des instances issues du Decoding Challenge [1], avec des valeurs de n comprises entre 10 et 150. Nous notons nos modèles (4), (4-5), (6) et (7) respectivement XNF1, XNF2 , CNF1 et CNF2. Les expérimentations ont été réalisées avec les solveurs CryptoMiniSat [27] pour les instances XNF et CaDiCaL [7] pour les

1. https://decodingchallenge.org/syndrome

Instance		CardNetwork				SortNetwork			
n	**t**	XNF1	XNF2	CNF1	CNF2	XNF1	XNF2	CNF1	CNF2
10	4	163.98	191.47	78.47	78.23	197.40	162.38	**61.84**	72.94
20	5	91.83	61.58	**1.41**	7.50	120.77	86.90	1.54	2.59
30	7	**12.39**	12.46	43.93	26.00	31.74	23.98	35.16	35.19
40	8	47.66	48.67	**8.41**	42.98	99.89	65.42	16.58	27.84
50	9	88.63	95.39	38.32	**24.10**	77.11	120.20	26.18	31.54
60	10	123.83	111.76	**31.16**	48.41	69.28	92.92	33.62	38.30
70	11	137.69	135.75	81.34	96.33	220.39	195.08	126.20	**43.13**
80	12	1831.61	2314.17	752.01	**368.60**	7416.13	1038.27	421.41	542.20
90	13	10788.26 (17)	6859.91	3636.34	2623.91	8000.81 (8)	5556.04	7321.13	**1711.03**
100	14	16510.45 (11)	21601.34 (18)	11696.80 (15)	7479.82 (17)	2977.08 (2)	**18901.42 (18)**	9276.91 (13)	13002.32 (14)
110	16	13202.29 (9)	15629.83 (16)	13770.29 (18)	**10099.52 (18)**	1853.72 (5)	12808.43 (12)	12199.81 (17)	13870.86 (18)
120	17	392.69 (3)	**23613.99 (16)**	8323.67 (12)	7184.07 (13)	6006.44 (5)	20028.54 (14)	13394.07 (12)	6804.34 (9)
130	18	7118.13 (4)	**13741.32 (8)**	3962.57 (3)	2882.16 (4)	4403.01 (3)	9863.13 (6)	7890.08 (5)	1141.45 (1)
140	19	- (0)	9196.25 (4)	**8583.47 (4)**	- (0)	- (0)	3203.19 (1)	1480.40 (2)	3245.78 (3)
150	20	71.35 (1)	1536.56 (1)	**2304.49 (2)**	271.72 (1)	- (0)	- (0)	- (0)	- (0)
SOMME		50580.77 (205)	**95150.45 (243)**	53312.68 (234)	31233.35 (233)	31473.78 (183)	72145.90 (231)	52284.92 (229)	40569.51 (225)

TABLE 1 – Somme des temps de résolution en secondes pour les instances résolues pour chaque taille n et t selon les différentes modélisations et encodages du problème de décodage par syndrome. Si des problèmes n'ont pas été résolus dans la limite du temps alloué, le nombre d'instances résolues est indiqué entre '()'. Les meilleurs résultats sont marqués en gras et soulignés, d'abord en fonction du nombre d'instances résolues, puis, en cas d'égalité, en fonction du temps de résolution.

instances CNF. Tous les tests ont été exécutés sur la plateforme de calcul MatriCS [2] sur la partition bigmem, équipée de 12 serveurs bi-processeur Intel Xeon E5-2680 v4 (2.40 GHz), chacun doté de 512 Go de mémoire, avec une limite de temps fixée à 3600 secondes (soit 1 heure) par instance. Nous avons utilisé la librairie PySAT [3] pour encoder nos modèles ainsi que les contraintes pseudo-booléennes. En particulier, nous avons testé deux encodages de la littérature pour les contraintes de cardinalité présentes dans tous les modèles : SortNetwork [2] et CardNetwork [1]. Pour une contrainte de cardinalité donnée sous la forme $\sum_{i=1}^{h} l_i \leq b$, la complexité en termes de nombres de clauses (et variables) générées est de $O(h \log^2 h)$ pour SortNetwork et de $O(h \log^2 b)$ pour CardNetwork. Ainsi, la complexité pour notre modèle XNF1 est respectivement de l'ordre de $O(n \log^2 n)$ et de $O(n \log^2 t)$ pour ces encodages (de même pour XNF2 avec un facteur n additionnel) . Pour les modèles CNF, nous avons également sélectionné l'encodage BinMerge [18] pour encoder les contraintes pseudo-booléennes (spécifiquement les contraintes (6a') et (7a')), dont la complexité est de l'ordre de $O(h \log^2 h \log W)$ où h désigne le nombre de littéraux dans la somme et W le poids maximal. Ainsi, la complexité de nos modèles CNF1 et CNF2 est respectivement de l'ordre de $O(n \log^3 n)$ et $O(n \log^2 n)$. Enfin, pour chaque valeur de n, chaque modèle et chaque encodage choisi pour les contraintes de cardinalité, nous avons généré 20 instances avec des graines aléatoires distinctes, totalisant ainsi 2400 instances du problème de décodage par syndrome.

4.2 Résultats globaux

Les résultats en termes de nombre d'instances résolues et de temps de résolution sont présentés dans le tableau 1. Pour l'encodage CardNetwork, XNF2 parvient à résoudre le plus grand nombre d'instances (243), avec une légère avance sur CNF1 et CNF2, qui en résolvent respectivement 234 et 233, soit 9 et 10 de moins. Cependant, ces deux modélisations CNF se démarquent par leur meilleure efficacité en termes de temps de résolution, avec des gains d'environ 44% pour CNF1 et 67% pour CNF2 par rapport à XNF2. La modélisation XNF1 restent nettement moins performantes, avec seulement 205 instances résolues. On note également que CNF2 est environ 38% plus rapide que XNF1, tout en résolvant 28 instances supplémentaires. Concernant l'encodage SortNetwork, XNF2 reste la modélisation la plus performante en termes d'instances résolues (231), suivie de CNF1 (229) puis CNF2 (225), avec respectivement un écart de 2 et 6 instances par rapport à XNF2. Toutefois, les performances en termes de temps de résolution sont à l'avantage des modélisations CNF, avec un gain d'environ 28% pour CNF1 et 44% pour CNF2 sur la somme totale des temps de résolution par rapport à XNF2. Comme avec CardNetwork, XNF1 reste en retrait, ne résolvant que 183 instances, soit 42 de moins que CNF2.

De manière générale, les modélisations sont plus performantes en nombre d'instances résolues lorsqu'elles sont associées à l'encodage CardNetwork. Par exemple, XNF2 résout 12 instances de plus avec cet encodage comparé à SortNetwork. Cela peut s'expliquer par le fait que CardNetwork génère moins de variables et de clauses, ce qui rend les instances plus faciles à résoudre. Globalement, CNF2 avec CardNetwork apparaît comme un excellent compromis entre nombre d'instances résolues et temps de résolution, affichant le meilleur temps de résolution total, tout en résolvant 50 instances de plus que XNF1 avec SortNetwork, pourtant deuxième en termes de temps. Enfin, CNF2 avec CardNetwork est environ 41% plus rapide que CNF1 avec le même encodage, tout en ne résolvant qu'une seule instance de moins. Nous remarquons également que XNF2 avec CardNetwork semble être la modélisation la plus ro-

2. https://www.matrics.u-picardie.fr/
3. https://pysathq.github.io/

Instance	CardNetwork				SortNetwork			
n	XNF1	XNF2	CNF1	CNF2	XNF1	XNF2	CNF1	CNF2
10-50	404.49	409.57	170.55	178.81	526.92	458.87	**141.29**	170.11
60-100	72591.84	38222.93	34197.65	**21417.07**	126683.7	32983.73	42379.28	36936.97
110-150	319584.45	261717.95	256544.49	**250837.47**	325463.16	287103.3	265364.36	273462.43
TOTAL	392580.77	300350.45	290912.68	**272433.35**	452673.78	320545.9	307884.92	310569.51

TABLE 2 – Score PAR1 obtenu pour chaque famille d'instances selon les différentes modélisations et encodages du problème de décodage par syndrome. Les meilleurs résultats sont marqués en gras et soulignés.

buste en termes de nombre d'instances résolues, notamment lorsque la difficulté augmente. Elle fait la différence sur les modélisations CNF pour les tailles n=120, 130 et 140, avec respectivement 16, 8 et 4 instances résolues contre 12, 3 et 4 pour CNF1, et 13, 4 et 0 pour CNF2 avec le même encodage.

Du point de vue des instances, nous avons classé notre benchmark en trois catégories : les instances faciles (n de 10 à 50), intermédiaires (n de 60 à 100) et difficiles (n de 110 à 150). Pour chaque famille d'instances, le PAR1 a été calculé en cumulant les temps de résolution et en indiquant la limite de temps pour les instances non résolues. Ces résultats sont présentés dans le tableau 2.

Pour les instances faciles, toutes les modélisations ont permis de résoudre l'ensemble des instances. Quelle que soit l'encodage utilisé, les modélisations CNF surpassent nettement les modélisations XNF. CNF1 avec SortNetwork affiche le meilleur temps, avec un gain d'environ 17% par rapport à CNF2 avec le même encodage (deuxième meilleur temps), et un gain d'environ 65% par rapport à XNF1 avec CardNetwork, qui reste la plus performante parmi les modélisations XNF. Pour les instances intermédiaires, CNF2 associée à CardNetwork surpasse les autres modélisations, avec un gain d'environ 35% par rapport à XNF2 associée à SortNetwork, qui se classe en deuxième position. Bien que CNF2 résolve une instance de moins que XNF2 dans cette catégorie, elle reste significativement plus rapide sur celles qu'elle parvient à traiter. Enfin, en ce qui concerne les instances difficiles, CNF2 avec CardNetwork conserve la première place en termes de score PAR1, suivie de près par CNF1, puis XNF2 avec le même encodage. Même si XNF2 parvient à résoudre davantage d'instances que les deux modélisations CNF, respectivement 6 et 9 instances de plus que CNF1 et CNF2, ces dernières restent plus performantes en termes de temps de résolution sur les instances qu'elles traitent.

Les résultats globaux montrent que les modélisations couplées à l'encodage CardNetwork offrent les meilleures performances. CNF2 avec CardNetwork affiche le score PAR1 le plus bas avec un gain d'environ 6% par rapport à CNF1 et de 9% par rapport à XNF2, tandis que XNF1 associée au même encodage et toutes les modélisations utilisant SortNetwork restent en retrait.

4.3 CNF vs XNF

Dans cette section, nous analysons plus en détails les différences de performances entre les modélisations CNF et XNF. La Figure 1 présente l'évolution du temps moyen de résolution (en secondes) en fonction de la taille des instances (n), pour les différentes modélisations encodées avec CardNetwork. La limite de temps est prise en compte dans le calcul de la moyenne lorsqu'une instance n'est pas résolue dans le temps imparti,.

On observe globalement une croissance très rapide du temps moyen avec la taille des instances. Cette croissance devient particulièrement marquée à partir de $n = 80$, avec des augmentations de $n = 70$ à $n = 80$ d'environ 1231% et 1604% pour XNF1 et XNF2, 824% et 282% pour CNF1 et CNF2. Ces valeurs illustrent l'explosion des temps de calcul, notamment pour les modélisations XNF. Pour les tailles d'instance allant de $n = 10$ à $n = 120$, on observe que les temps moyens des représentations CNF restent systématiquement inférieurs à ceux des XNF, à l'exception de $n = 30$ où la tendance s'inverse. De plus, on observe que XNF1 est nettement moins performant que XNF2, particulièrement à partir de $n = 90$ jusqu'à $n = 120$. On remarque également que CNF2 est constamment en dessous de CNF1 pour $n = 80$ à $n = 120$. Au-delà de $n = 120$, il devient

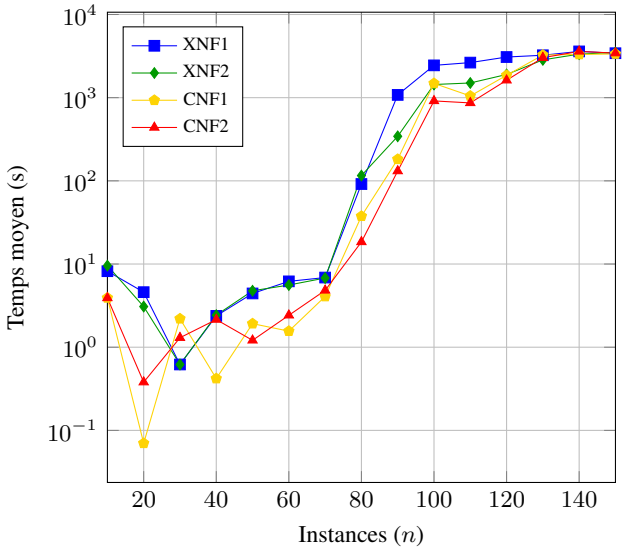

FIGURE 1 – Temps moyens en secondes des différentes modélisations encodées avec CardNetwork du problème de décodage par syndrome. Pour les problèmes non résolus dans le temps imparti, la limite de temps est prise en compte dans la moyenne.

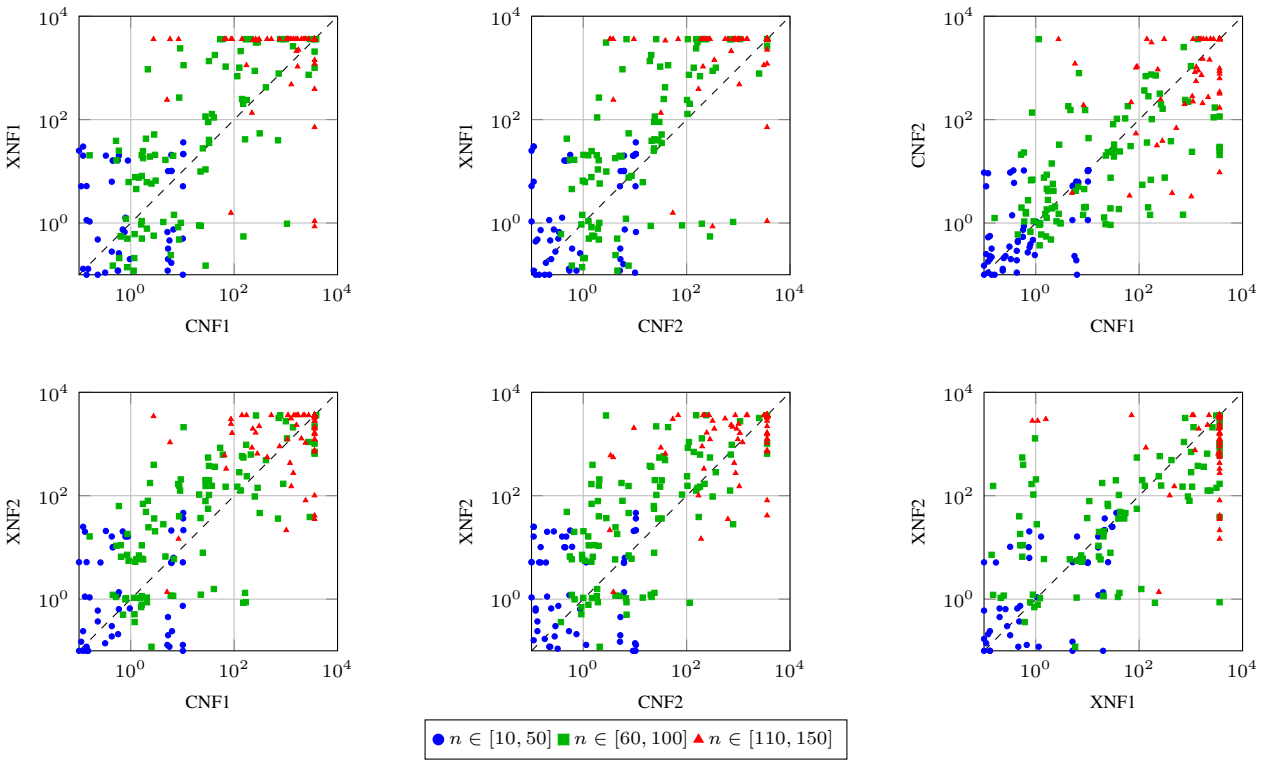

FIGURE 2 – Comparaison des temps de résolution en secondes par instance pour les différentes modélisations avec l'encodage CardNetwork. La limite de temps est prise en compte pour les instances non résolus dans le temps imparti.

plus difficile d'interpréter les courbes, vu la difficulté accrue du problème pour ces valeurs, faisant converger toutes les courbes vers la limite de temps fixée.

La Figure 2 illustre une comparaison des temps d'exécution par instance entre les différentes modélisations encodées avec CardNetwork. Parmi les modèles XNF, XNF1 est systématiquement moins performant, étant surpassé par les autres modélisations dans la majorité des cas. En comparant XNF2 aux modélisations CNF, CNF1 est plus rapide dans 47% des cas contre 37% en faveur de XNF2, et CNF2 l'emporte également dans 47% des cas contre 38% pour XNF2. Cette tendance est particulièrement marquée sur les instances de difficulté intermédiaire, où CNF1 et CNF2 dominent XNF2 respectivement dans 64% et 62% des cas. Entre les deux modèles CNF, CNF2 se montre légèrement supérieur, étant plus rapide que CNF1 dans 43% des cas contre 36% en faveur de CNF1. Cette domination de CNF2 est surtout visible sur les instances de difficulté intermédiaire et élevée.

Les modélisations CNF offrent globalement de meilleures performances que les représentations XNF. Parmi elles, CNF2 se démarque comme la plus performante, en particulier sur les instances de difficulté intermédiaire et élevée. À l'inverse, XNF1 affiche systématiquement les résultats les plus faibles. CNF1 et XNF2 obtiennent de meilleures performances que XNF1, mais restent globalement en retrait par rapport à CNF2 en terme de temps de résolution.

5 Conclusion

Dans cet article, nous avons présenté des travaux préliminaires visant à résoudre le problème du décodage par syndrome en utilisant des solveurs SAT, notamment CryptoMiniSat et CaDiCaL, en modélisant ce problème à l'aide de représentations XNF et CNF. Alors que les représentations XNF sont généralement reconnues comme plus efficaces que les représentations clausales pour les problèmes cryptographiques, nos résultats montrent que, dans le cadre du décodage par syndrome, les représentations CNF introduites offrent des performances comparables et compétitives à celles des représentations XNF.
Ces résultats ouvrent la voie à de nombreuses perspectives. Nos observations suggèrent qu'il serait pertinent d'explorer des stratégies d'encodage hybrides, combinant les avantages des représentations CNF et XNF, dans le but d'améliorer les performances globales, notamment sur les instances les plus complexes. De plus, il pourrait être judicieux d'expérimenter d'autres encodages pour les contraintes de cardinalité et pseudo-booléennes, dans le but d'optimiser davantage les performances. Enfin, il serait intéressant d'étudier des modélisations des variantes du décodage par syndrome. En particulier, le décodage par syndrome avec poids faible [4] s'intéresse à la recherche d'un vecteur d'erreur de poids minimal et on pourrait envisager de modéliser ce problème grâce à la satisfiabilité maximum (MaxSAT) [8], l'extension naturelle de SAT en problème d'optimisation.

4. https://decodingchallenge.org/low-weight

Remerciements

Ce travail est partiellement soutenu par le projet ANR-24-CE23-6126 (BforSAT) financé par l'agence nationale de recherche. Il a bénéficié d'un accès aux ressources HPC de la « Plateforme MatriCS » de l'Université de Picardie Jules Verne, qui est cofinancée par l'Union Européenne avec le Fonds Européen de Développement Régional (FEDER) et le Conseil Régional des Hauts-De-France entre autres.

Références

[1] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks and their applications. In O. Kullmann, editor, *SAT 2009*, volume 5584 of *LNCS*, pages 167–180. Springer, 2009.

[2] K. E. Batcher. Sorting networks and their applications. In *American Federation of Information Processing Societies - AFIPS 1968, Atlantic City, NJ, USA*, volume 32, pages 307–314. Thomson Book Company, Washington D.C., 1968.

[3] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3) :384–386, 1978.

[4] E. R. Berlekamp. *Algebraic coding theory*. McGraw-Hill series in systems science. McGraw-Hill, 1968.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding : Turbo-codes. 1. In *ICC'93*, volume 2, pages 1064–1070. IEEE, 1993.

[6] A. Biere. Pseudo-boolean and cardinality constraints. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, chapter 22, pages 695–730. IOS Press, 2009.

[7] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, and F. Pollitt. CaDiCaL 2.0. In A. Gurfinkel and V. Ganesh, editors, *CAV 2024*, volume 14681 of *LNCS*, pages 133–152. Springer, 2024.

[8] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[9] S. A. Cook. The complexity of theorem-proving procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *STOC 1971*, pages 151–158. ACM, 1971.

[10] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Optimization with parity constraints : From binary codes to discrete integration. In A. E. Nicholson and P. Smyth, editors, *UAI 2013*. AUAI Press, 2013.

[11] A. Esser and E. Bellini. Syndrome decoding estimator. In G. Hanaoka, J. Shikata, and Y. Watanabe, editors, *IACR 2022*, volume 13177 of *LNCS*, pages 112–141. Springer, 2022.

[12] J. Feldman, M. J. Wainwright, and D. R. Karger. Using linear programming to decode binary linear codes. *IEEE Transactions on Information Theory*, 51(3) :954–972, 2005.

[13] R. Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1) :21–28, 1962.

[14] S. Gocht and J. Nordström. Certifying parity reasoning efficiently using pseudo-boolean proofs. In *AAAI 2021*, pages 3768–3777. AAAI Press, 2021.

[15] A. Gupta, M. K. Ganai, and C. Wang. Sat-based verification methods and applications in hardware verification. In M. Bernardo and A. Cimatti, editors, *SFM 2006, Advanced Lectures*, volume 3965 of *LNCS*, pages 108–143. Springer, 2006.

[16] R. G. Jeroslow. On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11(2) :119–124, 1975.

[17] F. Lafitte, J. N. Jr., and D. V. Heule. Applications of SAT solvers in cryptanalysis : Finding weak keys and preimages. *J. Satisf. Boolean Model. Comput.*, 9(1) :1–25, 2014.

[18] N. Manthey, T. Philipp, and P. Steinke. A more compact translation of pseudo-boolean constraints into CNF such that generalized arc consistency is maintained. In C. Lutz and M. Thielscher, editors, *KI 2014*, volume 8736 of *LNCS*, pages 123–134. Springer, 2014.

[19] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44 :114–116, Jan. 1978.

[20] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, and I. Bourges. Hamming quasi-cyclic (hqc). *NIST PQC Round*, 2(4) :13, 2018.

[21] S. Narisada, S. Uemura, H. Okada, H. Furue, Y. Aikawa, and K. Fukushima. Solving mceliece-1409 in one day - cryptanalysis with the improved BJMM algorithm. In N. Mouha and N. Nikiforakis, editors, *ISC 2024*, volume 15258 of *LNCS*, pages 3–23. Springer, 2024.

[22] W. Nawrocki, Z. Liu, A. Fröhlich, M. J. H. Heule, and A. Biere. XOR local search for boolean brent equations. In C. Li and F. Manyà, editors, *SAT 2021*, volume 12831 of *LNCS*, pages 417–435. Springer, 2021.

[23] J. Rintanen. Planning and SAT. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 765–789. IOS Press, 2021.

[24] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3) :379–423, 1948.

[25] J. P. M. Silva and K. A. Sakallah. GRASP - a new search algorithm for satisfiability. In R. A. Rutenbar and R. H. J. M. Otten, editors, *ICCAD 1996*, pages 220–227. IEEE Computer Society / ACM, 1996.

[26] M. Soos, K. Nohl, and C. Castelluccia. Extending SAT solvers to cryptographic problems. In O. Kullmann, editor, *SAT 2009*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009.

[27] M. Soos, K. Nohl, and C. Castelluccia. Extending SAT solvers to cryptographic problems. In O. Kullmann, editor, *SAT 2009*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009.

[28] M. Trimoska, G. Dequen, and S. Ionica. Logical cryptanalysis with WDSat. In *Proceedings of SAT 2021*, Barcelona, Spain, July 2021.

[29] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 223–228, 1988.