Enumérer tous les modèles de formules booléennes par poids croissant*

Nadia Creignou **Frédéric Olive** **Johannes Schmidt**
Laboratoire d'Informatique Fondamentale de Marseille,
CNRS UMR 6166 - Aix-Marseille Université
{nadia.creignou,frederic.olive,johannes.schmidt}@lif.univ-mrs.fr

Résumé

Nous nous intéressons à l'énumération des modèles de formules booléennes par poids croissant. Le poids d'un modèle est le nombre de variables assignées à 1. Un algorithme est considéré comme efficace dans ce contexte s'il énumère les modèles l'un après l'autre dans le bon ordre en maintenant un délai polynomial entre deux solutions successives. Nous nous plaçons dans le cadre de Schaefer. L'énumération des modèles a déjà été abordée dans ce cadre, mais sans prise en compte de l'ordre d'énumération. Cette contrainte a une incidence cruciale sur la complexité. Nous obtenons une nouvelle classification dichotomique de la complexité de ce problème. D'une part nous présentons de nouveaux algorithmes à délai polynomial pour énumérer les modèles des formules de Horn et des formules 2-XOR par poids croissant ; d'autre part, nous prouvons que pour les formules ne relevant pas de l'un des deux types susmentionnés, il n'existe pas de tels algorithmes à délai polynomial, à moins que $\mathrm{P} = \mathrm{NP}$.

1 Introduction

Cet article traite de l'algorithmique et de la complexité de l'*énumération*, c'est-à-dire de la génération de toutes les solutions d'un problème donné. Cette question suscite un intérêt croissant depuis une dizaine d'années, du fait de l'explosion de la taille des données traitées par les ordinateurs dans les applications les plus quotidiennes. Pensons par exemple aux requêtes dans des bases de données, qui admettent fréquemment des volumes énormes de réponses. Présenter ces réponses de manière incrémentale efficace est un enjeu d'importance croissante. Ainsi, un utilisateur de moteur de recherche attend que les premiers résultats d'une recherche par mots clé soient instantanément produits. Au titre des autres champs d'application de l'énumération, citons notamment la résolution de contraintes, la recherche opérationnelle, la fouille de données, l'exploration du web, la bioinformatique et la linguistique computationnelle (voir e.g. [17, 8, 1]).

Du fait de la quantité de solutions possiblement générées par les algorithmes d'énumération, la taille de leur sortie est souvent bien plus importante que celle de leur entrée. Par conséquent, le temps polynomial mesure mal la performance de tels algorithmes, et on préfère quantifier la *régularité* de ces types de processus, plutôt que la durée totale de leur exécution. Ainsi, c'est la notion de *délai polynomial* qui est habituellement acceptée comme formalisation de la "faisabilité", en matière d'algorithmes d'énumération : un algorithme d'énumération est dit *à délai polynomial* si le temps écoulé entre deux solutions successives est polynomial en la taille de l'entrée.

Depuis le résultat inaugural de Schaefer [19], le point de vue des problèmes de satisfaction de contraintes (CSP) sur la théorie de la complexité a fait la preuve de sa pertinence (voir par exemple [5]). En offrant un cadre unifié intimement lié à divers problèmes issus des bases de données, il a permis de nombreux résultats de classification relatifs à différentes notions de complexité (voir [7] pour une synthèse). Le présent papier s'inscrit dans cette ligne de recherche.

Pour définir un CSP booléen non uniforme, on fixe un ensemble fini de relations booléennes Γ, appelé langage de contraintes. Une Γ-formule est alors une conjonction de clauses dont la forme est dictée par Γ. Le problème de l'énumération se formule alors comme suit : une Γ-formule étant fixée, peut-on énumérer efficacement tous ses modèles ? Dans [4], Creignou et

*Accepté à SAT'11. Avec le soutien de l'ANR *ENUM* 07-BLAN-0327-04.

Hébrard ont abordé cette question en prouvant un premier résultat de classification sur l'énumération relative aux Γ-formules : si Γ est Horn, dual-Horn, bijonctive ou affine, il existe un algorithme à délai polynomial qui énumère les modèles de toute Γ-formule ; sinon, un tel algorithme n'existe pas, à moins que P = NP.

Mais ce dernier résultat ignore un aspect important de la conception des algorithmes d'énumération : la spécification de l'ordre dans lequel les solutions sont générées. Or cet aspect est déterminant : dans de nombreux cas, on ne peut pas se permettre de générer l'intégralité des solutions, et l'on cherche plutôt à produire les plus "importantes" relativement à une *métrique* particulière, incarnée par cet ordre. Dans d'autres cas, on cherche une solution satisfaisant des propriétés supplémentaires complexes, et on génère les solutions par *ordre de préférence* jusqu'à en trouver une acceptable (voir quelques exemples dans [15, 26]). Par ailleurs, il s'avère que l'ordre imposé dans un problème d'énumération affecte profondément la complexité du problème. Johnson et al. [10] ont ainsi prouvé que les indépendants maximaux d'un graphe peuvent être énumérés dans l'ordre lexicographique par un algorithme à délai polynomial, alors qu'un tel algorithme n'existe pas pour l'ordre anti-lexicographique, à moins que P = NP.

Dans ce papier, nous spécifions l'ordre d'énumération. Nous traitons de l'énumération des modèles d'une formule *relativement à leur poids*, c'est-à-dire aux nombre de variables qu'ils évaluent à 1. Le poids est un paramètre naturel dans les CSP booléens [16, 18, 6], qui peut être assimilé au *coût* du modèle. Notre point de vue renvoie de ce fait aux nombreux travaux sur l'énumération par coût décroissant [25, 15, 26]. Ainsi, la question centrale de ce papier est : peut-on énumérer efficacement les modèles d'une Γ-formule par poids croissant ? Nous répondons à cette question par un résultat de classification dichotomique : si l'ensemble Γ de relations est Horn ou Krom, il existe un algorithme à délai polynomial qui énumère les modèles de toute Γ-formule par poids croissant ; sinon, un tel algorithme n'existe pas, à moins que P = NP. La preuve de ce résultat s'appuie sur de nouveaux algorithmes d'énumération pour les CSP booléens, en particulier pour les formules de Horn.

L'article est organisé comme suit : dans la Section 2, nous rappelons le vocabulaire de base des CSP booléens et de l'énumération. Nous énonçons aussi notre résultat principal, le Théorème 2.2. La preuve de ce théorème est détaillée au cours des sections suivantes. La Section 3 présente des algorithmes d'énumération efficaces, dans les cas où Γ est Krom ou Horn. Dans la Section 4, nour prouvons le versant négatif du Théorème 2.2 : pour toute relation Γ qui n'est ni Horn ni Krom, l'existence d'un algorithme à délai polynomial pour l'énumération des modèles d'une Γ-formule par poids croissant impliquerait P = NP. Nous concluons et pointons quelques questions ouvertes dans la Section 5.

2 Matériel

2.1 Langages de contraintes et Γ-formules

Une *relation logique* d'arité k est une relation $R \subseteq \{0,1\}^k$. Par abus de notation, nous ne différencions pas une relation et son symbole de prédicat. Une *contrainte* C est une formule $C = R(x_1, \ldots, x_k)$, où R est une relation logique k-aire et où les x_i sont des variables non nécessairement distinctes. Si u et v sont deux variables, on note $C[u/v]$ la contrainte obtenue à partir de C en remplaçant chaque occurrence de v par u. Si V est un ensemble de variables, $C[u/V]$ désigne la contrainte obtenue à partir de C par substitution de u à chaque occurrence de toute variable de V.

Une assignation m de valeurs de vérité aux variables *satisfait* la contrainte C si $\big(m(x_1), \ldots, m(x_k)\big) \in R$. Un *langage de contrainte* Γ est une ensemble fini de relations logiques. Une Γ-*formule* φ est une conjonction de contraintes écrites avec les seules relations logiques de Γ. C'est donc une formule du premier ordre sans quantificateur. On note $\mathrm{Var}(\varphi)$ l'ensemble de variables apparaissant dans φ. Une Γ-formule φ est satisfaite par une assignation $m : \mathrm{Var}(\varphi) \to \{0,1\}$ si m satisfait simultanément toutes les contraintes de φ. L'assignation m est alors un *modèle* de φ.

Un ordre canonique sur les variables étant fixé, une assignation s'interprète naturellement comme un ensemble de tuples. L'ensemble des modèles d'une formule φ constitue alors une relation logique R_φ que l'on confondra le plus souvent avec φ elle-même.

Au cours de l'article, nous nous référerons à la terminologie de Schaefer [19] pour classifier les relations booléennes. Une relation booléenne R est *Horn* (resp. *dual-Horn*) si R peut être définie par une formule de Horn (resp., dual-Horn), c'est-à-dire une formule sous forme normale conjonctive (CNF pour abréger), dont chaque clause contient au plus un litéral positif (resp., au plus un litéral négatif). Une relation R est *bijonctive* si elle peut être définie par une formule 2-CNF. Une relation R est *affine* si elle peut être définie par une formule *affine*, i.e. par une conjonction de XOR-clauses (obtenue par un XOR de plusieurs variables plus éventuellement la constante 1). Une telle formule peut aussi être vue comme un système d'équations linéaires sur GF[2]. Une relation est *Krom* (on dit aussi *affine de largeur* 2) si elle est définissable par

une conjonction de clauses, chacune d'entre-elles étant soit une clause unaire, soit une 2-XOR-clause (obtenue par un XOR de 2 variables plus éventuellement la constante 1). Une telle formule peut aussi être vue comme un système d'équations linéaires sur GF[2] avec au plus deux variables par équation. Une relation R est 0-*valide* (resp. 1-*valide*) si $R(0,\ldots,0)=1$ (resp. $R(1,\ldots,1)=1$). Finalement, un langage de contrainte Γ est Horn (resp. dual-Horn, bijonctif, affine, Krom, 0-valide, 1-valide) si chaque relation de Γ est Horn (resp. dual-Horn, bijonctive, affine, Krom, 0-valide, 1-valide). Nous disons qu'un langage de contrainte est *Schaefer* s'il est Horn, dual-Horn, bijonctif, ou affine.

On dispose de critères efficaces pour déterminer l'appartenance d'une relation aux classes pré-citées. Chacune de ces classes est en effet caractérisée par ses polymorphismes (voir e.g. [7] pour une description détaillée). Rappelons brièvement ces propriétés. Les opérations de conjonction, disjonction et addition appliquées sur des k-uplets booléens le sont coordonnée par coordonnée. Ainsi pour $m, m', m'' \in \{0,1\}^k$:

$$\begin{aligned} m \wedge m' &= (m[1] \wedge m'[1], \ldots, m[k] \wedge m'[k]) \\ m \vee m' &= (m[1] \vee m'[1], \ldots, m[k] \vee m'[k]) \\ m \oplus m' &= (m[1] \oplus m'[1], \ldots, m[k] \oplus m'[k]) \end{aligned}$$

où $m[i]$ est la i-ème coordonnée du vecteur m et $\oplus$ est l'opérateur XOR (le "ou exclusif"). Pour une relation logique R, les *propriétés de clôture* suivantes caractérisent la structure de R.

- (R est Horn) ssi $(m, m' \in R \Rightarrow m \wedge m' \in R)$.
- (R est dual-Horn) ssi $(m, m' \in R \Rightarrow m \vee m' \in R)$.
- (R est affine) ssi $(m, m', m'' \in R \Rightarrow m \oplus m' \oplus m'' \in R)$.
- (R est affine et 0-valide) ssi $(m, m' \in R \Rightarrow m \oplus m' \in R)$.

Le problème de la satisfaisabilité pour les Γ-formules, noté $\textsc{Sat}(\Gamma)$, fût d'abord étudié par Schaefer [19] qui établit sa fameuse classification dichotomique : si Γ est Schaefer ou 0-valide ou 1-valide, alors $\textsc{Sat}(\Gamma)$ est dans P ; sinon $\textsc{Sat}(\Gamma)$ est NP-complet. La complexité de la recherche d'une solution non triviale (i.e. d'un modèle qui évalue au moins une variable à 0 et une variable à 1), $\textsc{Sat}^*(\Gamma)$, est traité dans [4] : si Γ est Schaefer, alors $\textsc{Sat}^*(\Gamma)$ est dans P ; sinon, $\textsc{Sat}^*(\Gamma)$ est NP-complet. Depuis, des classifications de la complexité de nombreux autres problèmes algorithmiques relatifs aux Γ-formules ont été obtenues (voir [7] pour une synthèse).

2.2 Enumération

Nous traitons donc ici de l'énumération par poids croissant des modèles de formules booléennes généralisées, le poids d'une assignation étant le nombre de variables qu'il envoie sur 1. Ce problème peut être formulé comme suit :

Problème :	$\textsc{Enum-Sat}_w(\Gamma)$
Entrée :	une Γ-formule φ.
Sortie :	générer tous les modèles de φ par poids croissant.

On dit qu'un algorithme $\mathcal{A}$ calcule le problème d'énumération $\textsc{Enum-Sat}_w(\Gamma)$ si, pour une formule φ donnée, $\mathcal{A}$ génère les modèles de φ l'un après l'autre, par poids croissant, sans doublons, et s'arrête après la dernière sortie.

Le temps polynomial ne fournit pas une évaluation pertinente de l'efficacité des algorithmes d'énumération, parce que la sortie de tels algorithmes est de taille potentiellement exponentielle en la taille de leur entrée. Dans [10] plusieurs notions d'efficacité sont présentées. Le moins que l'on puisse exiger, est que l'énumération s'effectue en *temps total polynomial*, c'est-à-dire que le temps requis pour générer toutes les solutions soit polynomial en la taille de l'entrée *et* en le nombre de solutions (i.e. en la taille de la sortie). Une propriété attendue d'un algorithme d'énumération est qu'il commence à retourner des solutions le plus tôt possible et, plus généralement, qu'il génère les solutions de façon régulière, avec un délai borné entre deux sorties consécutives. On dit qu'un algorithme d'énumération est *à délai polynomial* si le temps écoulé avant la première sortie et le délai entre deux sorties consécutives sont bornés par un polynôme $p(n)$ en la taille de l'entrée. Notons qu'un tel algorithme génère les k premières solutions en temps $k \cdot p(n)$. Ceci est une proprieté importante des algorithmes à délai polynomial, parce que même si l'on ne dispose pas du temps nécessaire à l'énumération de *toutes* les solutions, du moins les k premières solutions seront-elles énumérées rapidement. Si $\textsc{Enum-Sat}_w(\Gamma)$ est calculable par un algorithme à délai polynomial, nous écrivons $\textsc{Enum-Sat}_w(\Gamma) \in \text{delayP}$.

Pour caractériser la complexité en espace, nous ignorons l'espace utilisé pour écrire les sorties, seul l'espace requis pour stocker des résultats intermédiaires est pris en compte. On exige parfois d'un algorithme d'énumération qu'il s'exécute en *espace polynomial*, i.e. que l'espace qu'il utilise pendant tout son calcul soit polynomial en la taille de l'entrée. Cette demande est restrictive, même pour un algorithme à délai polynomial. En effet certains algorithmes à délai polynomial, en particulier quand un ordre spécifique est imposé, s'appuient sur des structures de données de taille exponentielle en celle de la donnée (voir [10, 14, 24]). Ceci vaut notamment pour l'algorithme 2 décrit dans la Section 3.

Pourtant tout algorithme à délai polynomial utilise un *espace polynomial incrémental*. Ceci signifie que

l'espace requis pour générer les k premières solutions est borné par le produit de k et d'un polynôme en la taille de l'entrée.

2.3 Résultat principal

La complexité de l'énumération des modèles de formules booléennes en l'absence d'un ordre préalablement spécifié a été étudiée dans [4]. Une approche alternative, utilisant les polymorphismes partiels, a été développée plus récemment dans [21].

Théorème 2.1 *(*Enumération des modèles [4]*.) Si* Γ *est Schaefer, alors il existe un algorithme à délai polynomial qui génère tous les modèles d'une* Γ*-formule. Sinon, un tel algorithme n'existe pas, à moins que* $\mathrm{P} = \mathrm{NP}$.

Dans cet article nous nous intéressons à l'énumération des modèles par poids croissant. Bien sûr, quand Γ est Schaefer, le Théorème 2.1 permet d'exécuter cette tâche en *temps total polynomial*. Il suffit de générer d'abord toutes les solutions via l'algorithme à délai polynomial qui sous-tend la preuve de ce théorème, puis de trier les solutions afin de les produire par poids croissant. Cependant, une telle procédure interdit tout contrôle sur la régularité de l'énumération : s'il y a un nombre exponentiel de modèles, la première solution est produite après un temps exponentiel. Il s'avère nécessaire de développer des techniques plus élaborées pour réaliser une énumération efficace dans cet ordre. Les ensembles de relations qui admettent un bon comportement quant à cette tâche ne coïncident pas avec celles de Schaefer. Notre théorème principal détaille cette situation, énonçant un nouveau résultat dichotomique pour l'énumération des modèles par poids croissant.

Théorème 2.2 *(*Enumération par poids croissant*.) Si* Γ *est* $Horn$ *ou* $Krom$*, alors il existe un algorithme à délai polynomial qui génère tous les modèles d'une* Γ*-formule par poids croissant. Sinon, un tel algorithme n'existe pas, à moins que* $\mathrm{P} = \mathrm{NP}$.

Ce résultat est prouvé dans les deux prochaines sections.

3 Algorithmes d'énumération efficaces

Les algorithmes d'énumération efficaces proposés dans [4] (voir Théorème 2.1) sont implicitement basés sur l'auto-réductibilité ([23, 13, 20]). Par conséquent ces algorithmes énumèrent les modèles dans l'ordre lexicographique et n'utilisent qu'un espace polynomial. Enumérer les solutions dans l'ordre du poids croissant exige de nouveaux algorithmes. Les deux classes de langages de contrainte examinées dans cette section, $Krom$ et $Horn$, demandent des approches algorithmiques différentes. En effet elles se distinguent par la complexité de leur problème associé k-Ones, dans lequel, étant donné une formule et un entier k, on veut savoir s'il existe un modèle avec exactement k uns. Pour des formules $Krom$ ce problème est dans P, alors que pour des formules de $Horn$ il est NP-complet (voir [6]). En conséquence, dans le cas des formules $Krom$ on peut exploiter le fait que k-Ones est résoluble en temps polynomial pour obtenir un algorithme d'énumération efficace. En revanche, les formules de $Horn$ demandent une autre stratégie. Il est naturel d'utiliser une structure de données qui maintient un ensemble ordonné et qui permet des opérations efficaces d'insertion et d'extraction. Ainsi, l'algorithme que nous allons présenter pour le cas $Horn$ utilise une file de priorité. Cette méthode a déjà été utilisée dans e.g. [10, 14]. Comme dans ces articles, notre file de priorité permet l'insertion et l'extraction de l'élément de tête en temps logarithmique en la taille de l'entrée. Ainsi la taille de la file peut augmenter de façon exponentielle tandis qu'un délai polynomial est toujours maintenu.

Proposition 3.1 *Si* Γ *est Krom, alors il existe un algorithme à délai polynomial et en espace polynomial qui énumère tous les modèles d'une* Γ*-formule par poids croissant.*

Preuve : Soient Γ un ensemble de relations Krom et φ une Γ-formule. On peut supposer sans perte de généralité que φ ne contient pas de clauses unitaires. Ainsi chaque clause de φ exprime soit l'égalité, soit la non-égalité entre deux variables. En utilisant la transitivité de l'égalité et le fait que dans le cas booléen $a \neq b \neq c$ implique $a = c$, on peut identifier des classes d'équivalence de variables telles que deux classes quelconques ou bien sont indépendantes, ou bien correspondent à des valeurs de vérité opposées. On appelle *cluster* une paire (A, B) de classes correspondant à des valeurs de vérité opposées, B pouvant être vide. Il est immédiat que deux clusters quelconques sont nécessairement indépendants l'un de l'autre et donc afin d'obtenir un modèle de φ, il suffit de choisir pour chaque cluster (A, B) ou bien $A = 1, B = 0$ ou bien $A = 0, B = 1$. Dans la suite nous supposons que φ est satisfaisable (dans le cas contraire nous détecterons une contradiction en contruisant les clusters). Soit $n \geq 1$ le nombre de clusters, alors le nombre de modèles est 2^n. La contribution de chaque cluster en terme de poids est ou bien $|A|$ ou bien $|B|$ (il peut également arriver que $|A| = |B|$). Représentons un modèle par un n-uplet $s \in \{0, 1\}^n$ indiquant pour chaque cluster l'affectation qui est choisie. Dans le cas où $|A| \neq |B|$ nous indiquons

par 0 l'affectation de plus faible poids et par 1 l'autre. Ainsi $(0, 0, \ldots, 0)$ représentera un modèle de poids minimal, alors que $(1, 1, \ldots, 1)$ représentera un modèle de poids maximal. Pour l'énumération nous considérons seulement la différence de poids $\big||A| - |B|\big|$ de chaque cluster puisque nous pouvons au final soustraire le poids d'un modèle minimal. En posant $(w_1, \ldots, w_n)$ pour représenter les différences de poids de chacun des clusters nous sommes finalement ramenés au problème d'énumération suivant :

Problème :	UNARY-SUBSET-SUM
Entrée :	Une suite d'entiers positifs ou nuls $(w_1, \ldots, w_n) \in \mathbb{N}^n$ codés en unaire
Sortie :	énumérer tous les n-uplets $s \in \{0,1\}^n$ par poids $\delta(s)$ croissant, où $\delta(s) = \Sigma_{i=1}^n s_i \cdot w_i$

Observons que la taille de l'entrée est donnée par $\max(n, W)$ où $W := \Sigma_{i=1}^n w_i$ puisque les entiers sont codés en unaire. De plus la réduction de notre problème initial au problème UNARY-SUBSET-SUM est bien calculable en temps polynomial puisque la somme totale des poids W est majorée par le nombre de variables de la formule intiale (et donc la représentation des entiers en unaire n'est pas un obstacle). Pour achever la preuve de la proposition il nous reste à montrer que le problème UNARY-SUBSET-SUM est résoluble avec un délai polynomial. Pour ce faire nous utilisons une stratégie de programmation dynamique pour calculer en temps polynomial une matrice $A \in \{0,1\}^{(n+1,\, W+1)}$ telle que $A(i,k) = 1$ si et seulement si en sommant certains des poids $w_1, \ldots, w_i$ on peut atteindre la somme k, où $0 \leq i \leq n,\ 0 \leq k \leq W$. La matrice A est construite en posant tout d'abord $A(0,0) = 1$ et $A(0,k) = 0$ pour tout $k \geq 1$, puis en remplissant les autres champs ligne par ligne selon la règle $A(i,k) = 1$ si et seulement si $A(i-1,k) = 1$ ou $A(i-1, k-w_i) = 1$. Ainsi le calcul de A requiert un temps $O(n \cdot W)$. Ce pré-calcul étant fait, pour chaque poids k d'une solution, nous énumérons toutes ces solutions de poids k. Ceci se fait récursivement en construisant chaque solution, représentée par un mot binaire, à partir du mot vide (ϵ).

Le lecteur pourra se convaincre facilement que l'Algorithme 1 énumère toutes les solutions s du problème UNARY-SUBSET-SUM par poids $\delta(s)$ croissant. Le pré-calcul demande un temps quadratique et le délai est linéaire ensuite. Au final nous obtenons donc bien un algorithme d'énumération à délai polynomial qui utilise un espace quadratique pour l'énumération des modèles d'une formule Krom par poids croissant. □

Algorithm 1 Algorithme pour UNARY-SUBSET-SUM.

```
MAIN(w_1, ..., w_n)
compute A ∈ {0,1}^(n+1, W+1)
for k = 0 to W do
    if A(n,k) = 1 then
        CONSTRUCTSOLUTIONS(n, k, ε)
            /* énumération des solutions de poids k */
end for

CONSTRUCTSOLUTIONS(i, j, s)
if i = 0 then
    output s
else
    if A(i-1, j-w_i) = 1 then
        CONSTRUCTSOLUTIONS(i-1, j - w_i, 1 ∘ s)
        /* ∘ est l'opérateur de concaténation */
    if A(i-1, j) = 1 then
        CONSTRUCTSOLUTIONS(i-1, j, 0 ∘ s)
```

Proposition 3.2 *Si Γ est Horn, alors il existe un algorithme à délai polynomial qui énumère tous les modèles d'une Γ-formule par poids croissant.*

Preuve : Soit φ une Γ-formule. Si Γ est *Horn* alors φ est équivalente à une conjonction de clauses de *Horn*. Nous utilisons une file de priorité Q pour respecter l'ordre par poids croissant et pour éviter des doublons. La commande Q.enqueue(s, k) insère un élément s avec une clé k (un poids). La file trie par clé croissante et insère un élément s seulement s'il n'est pas encore présent dans la file.

Pour plus de lisibilité nous représentons un modèle par l'ensemble de variables assignées à 1. Nous utilisons le fait bien connu que l'intersection de tous les modèles d'une formule de *Horn* est l'unique *modèle minimal* qui peut être calculé en temps polynomial. Pour φ une formule de *Horn* satisfaisable nous indiquons le modèle minimal par $mm(\varphi)$. Remarquons que pour un ensemble de variables $V \subseteq \mathrm{Var}(\varphi)$ la formule $\varphi \wedge V := \varphi \wedge \bigwedge_{v \in V} v$ est encore une formule de *Horn*.

Nous affirmons qu'étant donné une formule de *Horn*, l'Algorithme 2 énumère tous ses modèles par poids croissant avec délai polynomial. Le fait que le délai soit polynomial est immédiat. Selon la définition de la file de priorité, et puisque les modèles m' générés à partir de m en ligne 11 sont toujours de poids supérieur à celui de m, il est également facile de vérifier que les modèles sont générés dans le bon ordre et qu'aucun modèle n'est généré deux fois. Pour prouver qu'aucun modèle n'est omis, il suffit de démontrer que pour tout modèle $m' \neq mm(\varphi)$ il existe un sous-modèle $m \subsetneq m'$ tel qu'en ligne 11 l'algorithme génère m' à partir de m. C'est-à-dire il faut qu'il y ait un $x \in m' \setminus m$ tel que $m' = mm(\varphi \wedge m \wedge x)$. Pour cela nous considérons l'ensemble $H := \{m \mid m \text{ un modèle de } \varphi \text{ et } m \subsetneq m'\}$.

Algorithm 2 Algorithm pour HORN-SAT.

Require: φ a Horn formula
1: **if** φ unsatisfiable **then**
2: return 'no'
3: Q = newPriorityQueue
4: $m := mm(\varphi)$
5: Q.enqueue(m, $|m|$)
6: **while** Q not empty **do**
7: m := Q.dequeue
8: output m
9: **for all** $x \in \mathrm{Var}(\varphi) \setminus m$ **do**
10: **if** $\varphi \wedge m \wedge x$ satisfiable **then**
11: $m' := mm(\varphi \wedge m \wedge x)$
12: Q.enqueue(m', $|m'|$)
13: **end for**
14: **end while**

L'ensemble H n'est pas vide car il contient au moins le modèle minimal $mm(\varphi)$. Un modèle maximal m de H satisfait toutes les conditions requises car il vérifie $m' = mm(\varphi \wedge m \wedge x)$ pour tout $x \in m' \setminus m$.

Finalement soulignons que contrairement à l'Algorithme 1, l'Algorithme 2 utilise potentiellement un espace exponentiel. □

4 Cas difficiles

Dans cette section nous nous intéressons au cas où Γ n'est ni Horn ni Krom. Clairement pour pouvoir énumérer tous les modèles d'une Γ-formule par poids croissant il est nécessaire d'être capable de trouver un modèle de poids minimal en temps polynomial. Comme nous allons le voir ce n'est pas une condition suffisante. En effet il faut également être capable de trouver le deuxième modèle efficacement. Les problèmes suivants vont jouer un rôle important dans notre étude :

Problème :	MIN-ONES(Γ)
Entrée :	une Γ-formule φ, un entier positif W
Question:	Existe-t-il un modèle de φ qui attribue la valeur vrai à au plus W variables ?

Problème :	MIN-ONES*(Γ)
Entrée :	une Γ-formule φ, un entier positif W
Question:	Existe-t-il un modèle de φ non identiquement nul qui attribue la valeur vrai à au plus W variables ?

De la classification obtenue dans [12] pour le problème d'optimisation correspondant on peut déduire ce qui suit.

Proposition 4.1 *Si Γ est 0-valide ou Horn ou Krom, alors* MIN-ONES(Γ) *est dans* P*, sinon* MIN-ONES(Γ) *est* NP*-complet.*

Notre principale contribution dans cette partie est le résultat de complétude suivant. Il prouve de façon évidente que si Γ n'est ni Horn ni Krom, alors il n'existe pas d'algorithme qui énumère tous les modèles d'une Γ-formule par poids croissant avec un délai polynomial à moins que P = NP.

Proposition 4.2 *Soit Γ un ensemble de relations qui n'est ni Horn ni Krom. Alors* MIN-ONES*(Γ) *est* NP*-complet.*

Preuve : Si Γ n'est pas Schaefer, alors SAT*(Γ) est NP-complet [19] et donc *a fortiori* le problème MIN-ONES*(Γ) l'est également. Si Γ n'est pas 0-valide, alors, puisqu'il n'est ni Horn ni Krom, le résultat découle de la NP-complétude de MIN-ONES(Γ) (Proposition 4.1). Ainsi, il reste à étudier les ensembles Γ qui sont Schaefer et 0-valides mais qui ne sont ni Horn ni Krom. Il y a trois cas à analyser.

- Γ est bijonctif et 0-valide mais ni Horn ni Krom.
- Γ est affine et 0-valide mais ni Horn ni Krom.
- Γ est dual-Horn et 0-valide mais ni Horn ni Krom.

Observer qu'une formule sous forme normale conjonctive avec deux littéraux dans chaque clause qui est 0-valide est également Horn. Ainsi le premier cas n'arrive jamais. De plus on peut facilement prouver qu'une relation 0-valide et affine qui n'est pas Horn ne peut-être Krom. Pour cette raison, afin d'établir la preuve de la proposition il est suffisant de prouver successivement la NP-complétude de MIN-ONES*(Γ) dans les deux cas suivants :

1. Γ est affine et 0-valide mais pas Horn, ou
2. Γ est dual-Horn et 0-valide mais ni affine ni Horn.

La NP-complétude de MIN-ONES*(Γ) pour tout ensemble Γ vérifiant une de ces deux conditions est établie respectivement dans les Propositions 4.6 et 4.9. Le schéma de la preuve donné par la Figure 1 convaincra facilement le lecteur qu'au final aucun cas n'a été oublié. □

4.1 Affine, 0-valide, non Horn

Dans cette partie nous traitons les relations qui sont 0-valides et affines mais non Horn. Nous allons prouver que pour une telle relation R, trouver un modèle non identiquement nul de poids minimal d'une R-formule est NP-difficile. Pour ce faire nous avons besoin de quelques lemmes techniques. Les deux premiers sont des résultats de définissabilité, alors que le troisième

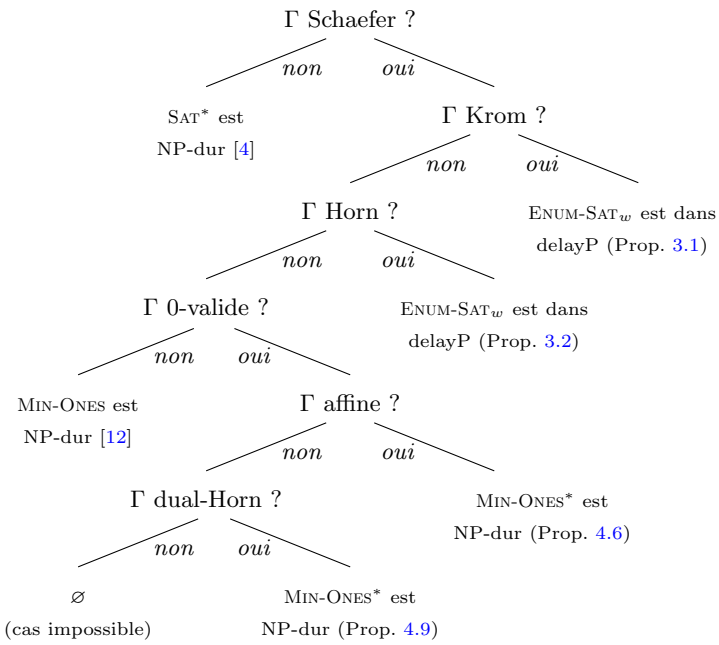

FIGURE 1 – Schéma de la preuve

est un résultat de complétude pour un problème de base.

Une des méthodes les plus populaires pour obtenir des résultats de complexité pour des problèmes liés à la satisfaction de contraintes (notamment pour l'énumération) est d'utiliser des outils issus de l'algèbre universelle. Une correspondance de Galois établit en effet un lien entre le pouvoir d'expressibilité d'un langage de contraintes et son ensemble de polymorphismes ou polymorphismes partiels (voir e.g. [9, 22]). Cependant dans le cas présent il n'est pas nécessaire d'utiliser ces outils très élaborés. En effet les résultats techniques dont nous avons besoin concernent des ensembles de relations très particuliers et peuvent être obtenus à la main.

Dans les preuves de tous les lemmes qui suivent R représente une relation d'arité k et V un ensemble de k variables deux à deux distinctes, mettons $V = \{x_1, \ldots, x_k\}$.

Lemme 4.3 *Soit R une relation qui est 0-valide et affine mais ni Horn ni 1-valide. Alors il existe une R-formule équivalente à $\neg w \wedge (x \oplus y \oplus z = 0)$.*

Preuve : Considérons la contrainte $C = R(x_1, \ldots, x_k)$. Puisque R n'est pas Horn il existe m_1 et m_2 dans R tels que $m_1 \wedge m_2 \notin R$. Etant donné que R est 0-valide et affine nous avons $m_1 \oplus m_2 \in R$. Pour $i, j \in \{0, 1\}$, soit $V_{i,j} = \{x \mid x \in V, m_1(x) = i \wedge m_2(x) = j\}$. Observons que $V_{0,1} \neq \varnothing$ (respectivement, $V_{1,0} \neq \varnothing$), sinon $m_1 \wedge m_2 = m_2$ (resp., $m_1 \wedge m_2 = m_1$), ce qui contredit le fait que $m_1 \wedge m_2 \notin R$. De plus $V_{1,1} \neq \varnothing$, sinon $m_1 \wedge m_2 = \vec{0}$, une contradiction. Considérons la $\{R\}$-contrainte : $M(w, x, y, z) = C[w/V_{0,0}, x/V_{0,1}, y/V_{1,0}, z/V_{1,1}]$. Selon la remarque ci-dessus les trois variables x, y and z apparaissent effectivement dans cette contrainte. Examinons l'ensemble des modèles de M qui attribuent la valeur 0 à w : il contient 0011 (car $m_1 \in R$), 0101 (car $m_2 \in R$), 0110 (car $m_1 \oplus m_2 \in R$) et 0000 (car R est 0-valide). Cependant il ne contient pas 0001 (car par hypothèse $m_1 \wedge m_2 \notin R$). Il ne contient pas non plus 0111. En effet, sinon il contiendrait $0011 \oplus 0101 \oplus 0111$ (car R est affine), qui est équivalent à 0001, contradiction. On peut également déduire que cet ensemble ne contient pas non plus 0010 ni 0100 (car $0000 \oplus 0011 \oplus 0010 = 0001$ et $0110 \oplus 0101 \oplus 0100 = 0111$). Remarquons que puisque R est 0-valide mais non-1-valide $C[w/V] \equiv \neg w$. Considérons alors

$$\varphi(w, x, y, z) = C[w/V] \wedge M(w, x, y, z).$$

Cette R-formule φ est équivalente à $\neg w \wedge (x \oplus y \oplus z = 0)$, ce qui termine la preuve. □

Lemme 4.4 *Soit R une relation qui est 0-valide, 1-valide, affine mais pas Horn. Alors il existe une R-formule équivalente à $(w \oplus x \oplus y \oplus z = 0)$.*

Preuve : Observons qu'une relation R qui est affine et à la fois 0-valide et 1-valide est nécessairement complémentive, i.e., pour tout $m \in R$ nous avons également $\vec{1} \oplus m \in R$. Nous pouvons imiter l'analyse faite dans le lemme précédent et considérer $M(w, x, y, z) = C[w/V_{0,0}, x/V_{0,1}, y/V_{1,0}, z/V_{1,1}]$. Ainsi $\varphi(w, x, y, z) = M(w, x, y, z)$ vérifie $\varphi(w, x, y, z) \equiv (w \oplus x \oplus y \oplus z = 0)$. □

Lemme 4.5 MIN-ONES*$(x \oplus y \oplus z = 0)$ *et* MIN-ONES*$(w \oplus x \oplus y \oplus z = 0)$ *sont NP-complets.*

Preuve : Soit S un système linéaire homogène dans le corps fini à deux éléments GF(2). Il est connu que trouver une solution non identiquement nulle de poids minimal d'un tel système est NP-difficile (voir [2, Théorème 4.1]). Afin de prouver le lemme ci-dessus nous devons prouver que ce problème reste difficile même quand il est restreint à des systèmes dont chaque équation a seulement trois (resp., quatre) variables. Supposons que S a n variables, $x_1, \ldots, x_n$. Afin de réduire le nombre de variables par équation nous introduisons des variables auxiliaires. S'il existe une équation $x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = 0$ pour un certain $k \geq 4$,

nous introduisons une nouvelle variable, y_{i_1,i_2}, et remplaçons l'équation originale par les deux équations $y_{i_1,i_2} \oplus x_{i_1} \oplus x_{i_2} = 0$ et $y_{i_1,i_2} \oplus x_{i_3} \oplus \ldots \oplus x_{i_k} = 0$. Nous réitérons ce procédé jusqu'à ce que toutes les équations aient trois variables. La satisfaisabilité du système est préservée durant cette transformation. Le nombre de variables auxiliaires est majoré par le nombre d'occurrences de variables dans le système initial. Afin de conserver l'information sur le poids des solutions l'idée est d'introduire suffisamment de copies des variables originales, rendant ainsi les variables auxiliaires négligables. Soit N le nombre d'occurrences de variables dans S. Soit f une nouvelle variable qui va jouer le rôle de la constante 0. Pour tout $i = 1, \ldots, n$, nous introduisons N copies de x_i, $x_i^1, \ldots, x_i^N$, et ajoutons les équations $x_i \oplus x_i^j \oplus f = 0$ pour $j = 1, \ldots N$. Finalement nous ajoutons l'équation $f \oplus f \oplus f = 0$, i.e., $f = 0$ (cette équation va assurer que $x_i = x_i^j$ pour tout j). Il y a une correspondance bijective entre les solutions de S et les solutions du système S' ainsi obtenu. De plus S a une solution non triviale de poids au plus W si et seulement si S' a une solution non triviale de poids au plus $W(N+1)+N$. Puisque le système S' peut-être vu comme une $(x \oplus y \oplus z = 0)$-formule, nous avons ainsi prouvé la NP-difficulté de Min-Ones$^*(x \oplus y \oplus z = 0)$.

Réduisons maintenant Min-Ones$^*(x \oplus y \oplus z = 0)$ à Min-Ones$^*(w \oplus x \oplus y \oplus z = 0)$. Soit S un système linéaire homogène à n variables tel que chaque équation comporte exactement trois variables. Soient w et w_i pour $i = 1, \ldots, n+1$ de nouvelles variables. Transformons S en S' comme suit : transformons chaque équation $x \oplus y \oplus z = 0$ en $w \oplus x \oplus y \oplus z = 0$ et ajoutons les $n+1$ equations $w \oplus w \oplus w \oplus w_i = 0$ pour $i = 1, \ldots, n+1$. Les solutions de S' affectant la valeur 0 à w coïncident avec les solutions de S. De plus toute solution de S' affectant la valeur 1 à w a un poids au moins égal à $n+1$. Donc, S a une solution non triviale de poids au plus W ($W \leq n$) si et seulement si S' a une solution non triviale de poids au plus W, c.q.f.d. □

Nous pouvons désormais énoncer notre résultat de difficulté.

Proposition 4.6 *Si R est 0-valide et affine mais non Horn, alors le problème* Min-Ones$^*(R)$ *est* NP-*complet.*

Preuve : La proposition découle des trois lemmes ci-dessus : si R n'est pas 1-valide, alors selon le Lemme 4.3 on peut réduire le problème Min-Ones$^*(x \oplus y \oplus z = 0)$ au problème Min-Ones$^*(R)$ (remplacer chaque contrainte $(x \oplus y \oplus z = 0)$ par la R-formule équivalente à $\neg w \wedge (x \oplus y \oplus z = 0)$, où w est une nouvelle variable). Si la relation R est 1-valide, alors selon le Lemme 4.4 on peut réduire Min-Ones$^*(w \oplus x \oplus y \oplus z = 0)$ à Min-Ones$^*(R)$. Dans les deux cas le Lemma 4.5 permet de conclure. □

4.2 Dual-Horn, 0-valide, ni Krom ni Horn

Dans cette partie nous nous intéressons aux relations qui sont 0-valides et dual-Horn mais ni affines ni Horn. La méthode de preuve n'est pas la même que celle de la section précedente. Néanmoins nous aurons également besoin de quelques résultats techniques intermédiaires.

Lemme 4.7 *Soit R une relation qui est 0-valide et dual-Horn mais ni affine ni 1-valide. alors il existe une R-formule équivalente à $\neg t \wedge (u \to v)$.*

Preuve : Considérons la contrainte $C = R(x_1, \ldots, x_k)$. Observons que puisque R est 0-valide mais non 1-valide, $C[t/V] \equiv \neg t$. Puisque R est 0-valide mais pas affine il existe deux tuples distincts m_1 et m_2 dans R tels que $m_1 \oplus m_2 \notin R$. Puisque R est dual-Horn, nous avons $m_1 \vee m_2 \in R$. Pour $i, j \in \{0, 1\}$, soit $V_{i,j} = \{x \mid x \in V, m_1(x) = i \wedge m_2(x) = j\}$. Observons que $V_{1,1} \neq \varnothing$, sinon $m_1 \vee m_2 = m_1 \oplus m_2$, une contradiction. De plus, puisque $m_1 \neq m_2$ ou bien $V_{0,1}$ ou $V_{1,0}$ est non vide. Supposons dans un premier temps qu'ils sont tous les deux non vides. Considérons la R-contrainte $M(w, x, y, z) = C[w/V_{0,0}, x/V_{0,1}, y/V_{1,0}, z/V_{1,1}]$. Les trois variables x, y and z apparaissent effectivement dans cette contrainte. Examinons l'ensemble des modèles de M qui affectent la valeur 0 à w. Cet ensemble contient 0011 (car $m_1 \in R$), 0101 (car $m_2 \in R$), 0111 (car $m_1 \vee m_2 \in R$) et 0000 (car R est 0-valide), mais ne contient pas 0110 (puisque par hypothèse $m_1 \oplus m_2 \notin R$). L'appartenance de 0100, 0010, 0001 à cet ensemble est ouverte :

- S'il ne contient pas 0100, alors considérons la R-formule $\varphi(t, u, v) := C[t/V] \wedge M(t, u, v, v)$. Son ensemble de modèles est $\{001, 011, 000\}$, et donc $\varphi(t, u, v) \equiv \neg t \wedge (u \to v)$.
- S'il contient 0100, alors il ne contient pas 0010. En effet sinon, puisque R est dual-Horn, il contiendrait également 0110, ce qui fournit une contradiction. Considérons donc la R-formule $\varphi(t, u, v) := C[t/V] \wedge M(t, v, u, v)$. Son ensemble de modèles est $\{001, 011, 000\}$, et donc $\varphi(t, u, v) \equiv \neg t \wedge (u \to v)$.

Si par exemple $V_{0,1} = \varnothing$, alors on doit considérer $M(w, y, z) = C[w/V_{0,0}, y/V_{1,0}, z/V_{1,1}]$. Dans ce cas $\varphi(t, u, v) := C[t/V] \wedge M(t, u, v)$ est équivalent à $\neg t \wedge (u \to v)$. □

Lemme 4.8 *Soit R une relation qui est 0-valide, 1-valide, et dual-Horn mais pas Horn, alors il existe une R-formule équivalente à $(u \to v)$.*

Preuve : Puisque R est dual-Horn mais pas Horn, elle n'est pas complémentive, c'est-à-dire il existe un certain tuple $m \in R$ tel que $m \oplus \vec{1} \notin R$. Considérons alors la contrainte $C = R(x_1, \ldots, x_k)$. Pour $i \in \{0, 1\}$, soit $V_i = \{x \mid x \in V \wedge m(x) = i\}$. Considérons la R-formule $\varphi(u, v) = C[u/V_0, v/V_1]$. Alors $\varphi(u, v) \equiv \neg u \vee v \equiv u \rightarrow v$. □

Proposition 4.9 *Si R est 0-valide et dual-Horn mais n'est ni affine ni Horn, alors le problème* Min-Ones$^*(R)$ *est* NP*-complet.*

Preuve : Soit R une telle relation qui est 0-valide et dual-Horn mais n'est ni affine ni Horn. Soit T la relation unaire constante, $T = \{1\}$. Selon la Proposition 4.1, le problème Min-Ones(R, T) est NP-complet. Réduisons Min-Ones(R, T) à Min-Ones$^*(R)$. Soit φ une $\{R, T\}$-formule, $\varphi = \psi \wedge \bigwedge_{x \in V} T(x)$ où ψ est une R-formule. Soit t une nouvelle variable. Considérons

$$\varphi' = \psi[t/V] \wedge \bigwedge_{x \in \mathrm{Var}(\varphi) \setminus V} x \rightarrow t.$$

Observons que la seule solution qui affecte la valeur 0 à t dans φ' est la solution identiquement nulle. Pour cette raison il est clair que φ a une solution de poids au plus W ($W \geq |V|$) si et seulement si φ' admet une solution non identiquement nulle de poids au plus $W - \mid V \mid +1$. Les deux lemmes ci-dessus permettent d'exprimer φ' comme une R-formule (modulo l'introduction variable supplémentaire qui va toujours prendre la valeur 0 quand R n'est pas 1-valide), la preuve est ainsi complète. □

5 Conclusion

Nous avons classifié la complexité de l'énumération des modèles d'une Γ-formule par poids croissant selon Γ. Nous avons prouvé que dans le cas des problèmes de satisfaction de contraintes booléennes une condition nécessaire et suffisante pour énumérer toutes les solutions par poids croissant avec un délai polynomial est le fait d'être capable de fournir une solution non identiquement nulle de poids minimal en temps polynomial. Remarquons que par dualité, sous l'hypothèse P $\neq$ NP, on peut énumérer tous les modèles d'une Γ-formule par poids *décroissant* avec un délai polynomial si et seulement si Γ est Krom ou dual-Horn. Nous aurions également pu considérer la version pondérée du problème, dans laquelle il y a une fonction de poids $w\colon V \rightarrow \mathbb{N}$ et le poids d'un modèle m est donné par $\sum_{v \in V} w(v) \cdot m(v)$. L'algorithme proposé dans la preuve de la Proposition 3.2 pour les formules de Horn pourrait tout aussi bien traiter cette variante. En revanche tel n'est pas le cas de l'algorithme proposé pour les formules Krom (Proposition 3.1). En effet cet algorithme est basé sur une procédure de programmation dynamique qui est pseudo-polynomiale et qui devient de complexité exponentielle quand les poids ne sont plus polynomialement bornés. Cependant, au prix d'un espace exponentiel, on peut développer un algorithme d'énumération à délai polynomial pour les formules Krom en utilisant une file de priorité comme cela a été proposé dans le cas des formules de Horn. Le fait d'exiger que l'énumération soit faite par poids croissant a révélé de nouveaux algorithmes d'énumération pour les problèmes de satisfaction de contraintes booléennes, différents de ceux proposés jusque-là. L'algorithme développé pour les formules de Horn requiert potentiellement un espace exponentiel, il serait souhaitable bien sûr d'éviter cela. Il serait intéressant de savoir s'il est possible de réduire cet espace utilisé au prix peut-être d'une augmentation de la complexité en temps ou à l'inverse si cette quantité d'espace exigée est inhérente à cet ordre particulier (par poids croissant) pour les formules de Horn. Une autre piste de recherche est l'étude de l'énumération par poids croissant des solutions de problèmes de satisfaction de contraintes sur des domaines finis de cardinalité supérieure à 2. L'énumération des solutions de tels problèmes a déjà été étudiée dans [3, 21]. Comme mentionné dans [21] le fait d'exiger un ordre pré-défini dans l'énumération pourrait être la clé pour découvrir de nouveaux algorithmes. Dans [11] les auteurs ont étudié la complexité du problème Max-Sol(Γ) qui généralise le problème Max-Ones à des domaines finis arbitraires. Ils ont identifié deux classes de langages de contraintes pour lesquelles ce problème peut-être résolu efficacement. Ces deux classes peuvent être vues comme des généralisations substantielles et non triviales des classes pour lesquelles le problème Max-Ones est efficacement résoluble sur le domaine booléen, c'est-à-dire, Krom, 1-valide et dual-Horn. Il serait intéressant d'examiner ces classes plus en détail afin de déterminer si elles peuvent donner lieu à des algorithmes à délai polynomial pour l'énumération des solutions par poids croissant.

Remerciements

Nous remercions Arnaud Durand pour nous avoir fourni une précieuse référence, [2].

Références

[1] G. Bagan, A. Durand, and E. Grandjean. On acyclic conjunctive queries and constant delay enumeration. In J. Duparc and T. A. Henzinger, edi-

tors, *CSL*, volume 4646 of *LNCS*, pages 208–222. Springer, 2007.

[2] A. Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(46), 1997.

[3] D. A. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3) :219–229, 2004.

[4] N. Creignou and J.-J. Hébrard. On generating all solutions of generalized satisfiability problems. *Theoretical Informatics and Applications*, 31(6) :499–511, 1997.

[5] N. Creignou, Ph. G. Kolaitis, and H. Vollmer, editors. *Complexity of Constraints - An Overview of Current Research Themes*, volume 5250 of *LNCS*. Springer, 2008.

[6] N. Creignou, H. Schnoor, and I. Schnoor. Nonuniform boolean constraint satisfaction problems with cardinality constraint. *ACM Trans. Comput. Log.*, 11(4), 2010.

[7] N. Creignou and H. Vollmer. Boolean constraint satisfaction problems : When does Post's lattice help ? In Creignou et al. [5], pages 3–37.

[8] M. Hagen. Lower bounds for three algorithms for transversal hypergraph generation. *Discrete Applied Mathematics*, 2009.

[9] P. G. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200 :185–204, 1998.

[10] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3) :119–123, 1988.

[11] P. Jonsson and G. Nordh. Introduction to the maximum solution problem. In Creignou et al. [5], pages 255–282.

[12] S. Khanna, M. Sudan, and D. Williamson. A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction. In *Proceedings 29th Symposium on Theory of Computing*, pages 11–20. ACM Press, 1997.

[13] S. Khuller and V. V. Vazirani. Planar graph coloring is not self-reducible, assuming P $\neq$ NP. *Theoretical Computer Science*, 88(1) :183–189, 1991.

[14] B. Kimelfeld and Y. Sagiv. Incrementally computing ordered answers of acyclic conjunctive queries. In O. Etzion, T. Kuflik, and A. Motro, editors, *NGITS*, volume 4032 of *LNCS*, pages 141–152. Springer, 2006.

[15] B. Kimelfeld and Y. Sagiv. Efficiently enumerating results of keyword search over data graphs. *Inf. Syst.*, 33(4-5) :335–359, 2008.

[16] A. A. Krokhin and D. Marx. On the hardness of losing weight. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (1)*, volume 5125 of *LNCS*, pages 662–673. Springer, 2008.

[17] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Scandinavian Workshop on Algorithm Theory*, pages 260–272, 2004.

[18] D. Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2) :153–183, 2005.

[19] T. J. Schaefer. The complexity of satisfiability problems. In *Proccedings 10th Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.

[20] J. Schmidt. Enumeration : Algorithms and complexity. Preprint (2009), available at http://www.thi.uni-hannover.de/fileadmin/forschung/arbeiten/schmidt-da.pdf.

[21] H. Schnoor and I. Schnoor. Enumerating all solutions for constraint satisfaction problems. In W. Thomas and P. Weil, editors, *STACS*, volume 4393 of *LNCS*, pages 694–705. Springer, 2007.

[22] H. Schnoor and I. Schnoor. Partial polymorphisms and constraint satisfaction problems. In Creignou et al. [5], pages 229–254.

[23] C. P. Schnorr. Optimal algorithms for self-reducible problems. In *International Conference on Automata, Languages and Programming*, pages 322–337, 1976.

[24] Y. Strozecki. Enumeration complexity and matroid decomposition. *Phd thesis*, 2010.

[25] V. Vazirani and M. Yannakakis. Suboptimal cuts : Their enumeration, weight and number. In W. Kuich, editor, *Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 366–377. Springer Berlin / Heidelberg, 1992.

[26] L.-P. Yeh, B.-F. Wang, and H.-H. Su. Efficient algorithms for the problems of enumerating cuts by non-decreasing weights. *Algorithmica*, 56(3) :297–312, 2010.