Ministère de l'Education Nationale
Université de Montpellier II
Place Eugène Bataillon
34095 Montpellier Cedex 5

TP FMIN105
Algorithmique / Complexité / Calculabilité

Rapport (Décembre 2011)

Travail préparé par :

Thibaut Marmin
Clément Sipieter
William Dyce

Table des matières

Chapitre 1

Partie théorique

1.1 Algorithmique

1.2 Complexité

1.2.1 SAT $\propto$ 3–SAT

(a) **Énoncé de SAT** :

Données :	$\mathcal{V} = \{v_1, v_2 \dots v_n\}$	*Ensemble de n variables*
	$\mathcal{C} = \{c_1, c_2, c_3 \dots c_m\}$	*Ensemble de m clauses*
où	$c_i = (l_{i1} \vee l_{i2} \vee \dots \vee l_{ik})$	*Clauses de k littéraux*
avec	$l_{ij} = v$ ou $\neg v$	*avec* $v \in U$

Problème : existe-il au moins une affectation des variables telle que chaque clause de $\mathcal{C}$ soit vrai.

Énoncé de 3–SAT :

3–SAT est identique au problème SAT avec $k = 3$.

Données :	$\mathcal{V} = \{v_1, v_2, v_3 \dots v_n\}$
	$\mathcal{C} = \{c_1, c_2, c_3 \dots c_m\}$
où	$c_i = (l_{i1} \vee l_{i2} \vee l_{i3})$
avec	$l_{ij} = v$ ou $\neg v$

(b) La réduction du problème SAT peut être définit en montrant que chaque clause c de $\mathcal{C}$ peut-être transformée en un ensemble de clauses $\mathcal{C}'$ tel que pour toute affectation rendant vrai l'ensemble des clauses de $\mathcal{C}$, on peut trouver une affectation rendant vrai chaque clause de $\mathcal{C}'$. Chaque clause de $\mathcal{C}'$ devant être de taille exactement 3. La réciproque doit également être montrée.

Définissons les réductions :

$\boldsymbol{k = 1}$

Soit ci_1 une clause de taille 1, on a $ci_1 = (l)$. Ajoutons deux variables $v_1, v_2 \notin \mathcal{V}$ et transformons la clause c en quatre clauses. On obtient l'ensemble $\mathcal{C}_1 = \{c_1, c_2, c_3, c_4\}$ avec :

$$c_1 = (l \vee v_1 \vee v_2)$$

$$c_2 = (l \vee v_1 \vee \neg v_2)$$

$$c_3 = (l \vee \neg v_1 \vee v_2)$$

$$c_4 = (l \vee \neg v_1 \vee \neg v_2)$$

$\boldsymbol{k = 2}$

Soit ci_2 une clause de taille 2, on a $ci_2 = (l_1 \vee l_2)$. Ajoutons une variable $v \notin \mathcal{V}$ et transformons la clause c en deux clauses. On obtient l'ensemble $\mathcal{C}_2 = \{c_1, c_2\}$ avec :

$$c_1 = (l_1 \vee l_2 \vee v)$$

$$c_2 = (l_1 \vee l_2 \vee \neg v)$$

$\boldsymbol{k = 3}$

La clause ci_3 ne subit pas de transformation.

$$\mathcal{C}_3 = \{ci_3\}$$

$k > 3$

Soit la clause $ci_k = (l_1 \vee l_2 \vee \cdots \vee l_k)$. On ajoute $(k-3)$ nouvelles variables $(v_1, v_2 \ldots v_{k-3})$.

$$\mathcal{C}_k = \underbrace{(l_1 \vee l_2 \vee v_1)}_{c_1} \bigwedge_{i=1}^{k-4} \left[\underbrace{(\neg v_i \vee l_{i+2} \vee v_{i+1})}_{c_{i+1}} \right] \wedge \underbrace{(\neg v_{k-3} \vee l_{k-1} \vee l_k)}_{c_{k-2}}$$

Montrons que SAT est vrai si et seulement si 3–SAT est vrai :

SAT → 3–SAT

- Soit une interprétation I_1 qui satisfasse la clause ci_1 :

$$val(I_1, ci_1) = val(I_1, l) = vrai$$

Prenons une interprétation I_1' avec $val(I_1, l) = val(I_1', l)$, peu importe les affectations de v_1 et v_2, l étant présent dans toutes les clauses de $\mathcal{C}_1$:

$$val(I_1', \mathcal{C}) = vrai$$

- Soit une interprétation I_2 qui satisfasse la clause ci_2 :

$$\exists i, val(I_2, l_i) = vrai$$

Prenons une interprétation I_2' avec :

$$val(I_2, l_1) = val(I_2', l_1)$$

$$val(I_2, l_2) = val(I_2', l_2)$$

Peu importe l'affectation de v dans I_2', on a $val(I_2', \mathcal{C}_2) = vrai$.

- Soit une interprétation I_k qui satisfasse la clause ci_k :

$$\exists i, val(I_k, l_i) = vrai$$

Prenons une interprétation I_k' telle que :

$$\begin{aligned} val(I_k, l_i) &= val(I_k', l_i) \\ \forall j \in [1; (i-2)], val(I_k', v_j) &= vrai \\ \forall j \in [(i-1); (k-3)], val(I_k', v_j) &= faux \end{aligned}$$

On obtient :

$$val(I_k', \mathcal{C}_k) = vrai$$

3–SAT → SAT

- Prenons une interprétation I_1 telle que $val(I_1, \mathcal{C}_1) = vrai$.
Sans perte de généralité, on suppose que :

$$val(I_1, v_1) = val(I_1, v_2) = vrai$$

La clause c_4 de $\mathcal{C}_1$ ne peut être satisfaite que si $val(I_1, l) = vrai$.
On a donc :

$$val(I_1, ci_1) = vrai$$

– Prenons une interprétation I_2 telle que $val(I_2, \mathcal{C}_2) = vrai$.
Sans perte de généralité on suppose que :

$$val(I_2, v) = vrai$$

La clause c_2 de $\mathcal{C}_2$ ne peut être satisfaire que si $val(I_2, (l_1 \vee l_2)) = vrai$.
On a donc :

$$val(I_2, ci_2) = vrai$$

– Prenons une interprétation I_k telle que $val(I_k, \mathcal{C}_k) = vrai$ et montrons qu'il existe forcément un i tel que $val(I_k, l_i) = vrai$.
Supposons que l'interprétation I_k est modèle de $\mathcal{C}_k$ avec

$$\forall i \in [1; k], val(I_k, l_i) = faux$$

$$\Rightarrow val(I_k, v_1) = vrai \text{ (dans } c_1)$$

Donc :

$$\begin{array}{lrl} & \forall i \in [1; (k-4)], val(I_k, v_{i+1}) & = vrai \\ \Rightarrow & val(I_k, v_{k-3}) & = vrai \\ \Rightarrow & val(I_k, c_{k-2}) & = faux \\ \Rightarrow & val(I_k, \mathcal{C}_k) & = faux \end{array}$$

Pour que l'interprétation I_k satisfasse $\mathcal{C}_k$, il doit exister un $i \in [1; k]$ tel que $val(I_k, l_i) = vrai$.
On a donc :

$$val(I_k, ci_k) = vrai$$

(c) Le point (b) définit la réduction de SAT vers 3–SAT. Afin de montrer la NP-Complétude de 3–SAT, montrons que la réduction s'effectue en un temps polynomial.
Soit :

k la taille de la clause initiale,

v_k le nombre de variables à ajouter pour obtenir des clauses de taille 3,

w_k le nombre de clauses de taille 3 obtenues à partir de la clause initiale.

$$\begin{array}{ll} v_3 = 0 & w_3 = 1 \\ v_4 = 1 & w_4 = 2 \\ v_5 = 2 & w_5 = 3 \\ \vdots & \vdots \end{array}$$

Pour tout $k > 3$:

$$v_k = v_{\lceil \frac{k}{2} \rceil + 1} + v_{\lfloor \frac{k}{2} \rfloor + 1} + 1$$

$$w_k = w_{\lceil \frac{k}{2} \rceil + 1} + w_{\lfloor \frac{k}{2} \rfloor + 1}$$

$v_k = \theta(k)$, donc borné par la taille de F. La réduction s'effectue donc en un temps polynomial.

Il est possible de réduire le problème SAT à 3–SAT en un temps polynomial, SAT étant NP-complet, 3–SAT l'est aussi.

(d) Soit $\mathcal{C}$ un ensemble de clause à n_v variables avec n_1 clauses de taille 1, n_2 clauses de taille 2, n_3 clauses de taille 3, n_4 clauses de taille 4 et n_5 clauses de taille 5. Calculons le nombre de variables et le nombre de clauses obtenues après réduction (respectivement n'_v et n'_c).
Les points (b) et (c) permettent de déterminer pour une clause de taille k, le nombre de clause obtenues et le nombre de variables ajoutées après réduction. On peut donc en déduire la tableau suivant :

Taille de la clause dans $\mathcal{C}$	1	2	3	4	5
Nombre de clauses	n_1	n_2	n_3	n_4	n_5
Nombre de variables ajoutées par clause	2	1	0	1	2
Nombre de variables ajoutées au total	$2n_1$	n_2	0	n_4	$2n_5$
Nombre de clauses obtenues par clause	4	2	1	2	3
Nombre de clauses obtenues au total	$4n_1$	$2n_2$	n_3	$2n_4$	$3n_5$

On a donc :

$$n'_v = n_v + 2n_1 + n_2 + n_4 + 2n_5$$
$$n'_c = 4n_1 + 2n_2 + n_3 + 2n_4 + 3n_5$$

1.2.2 3–SAT $\propto$ 2–SAT ?

Cette réduction repose sur un principe qui consiste à décomposer une clause de taille k en plusieurs clauses de tailles inférieures.

Soit une clause $c = (l_1 \vee l_2 \vee l_3)$ une clause de taille 3 et I une interprétation qui satisfait c.

Cas 1 : décomposons cette clause en deux clauses c_1 et c_2 de tailles 1 et 2 :

$$\begin{aligned} c_1 &= (l_1) \\ c_2 &= (l_2 \vee l_3) \end{aligned}$$

Pour montrer l'équivalence 3–SAT $\leftrightarrow$ 2–SAT, il faut ajouter une variable v aux deux clauses créées :

$$\begin{aligned} c_1 &= (l_1 \vee v) \\ c_2 &= (l_2 \vee l_3 \vee \neg v) \end{aligned}$$

On a donc la clause c_2 de taille 3.

Cas 2 : décomposons cette clause en trois clauses c_1, c_2 et c_3 de taille 1 :

$$\begin{aligned} c_1 &= (l_1) \\ c_2 &= (l_2) \\ c_3 &= (l_3) \end{aligned}$$

Pour montrer l'équivalence 3–SAT $\leftrightarrow$ 2–SAT, il faut ajouter deux variables v_1 et v_2 aux trois clauses créées :

$$\begin{aligned} c_1 &= (l_1 \vee v_1 \vee \neg v_2) \\ c_2 &= (l_2 \vee \neg v_1 \vee v_2) \\ c_3 &= (l_3 \vee v_1 \vee v_2) \end{aligned}$$

On a donc également des clauses de taille 3. La réduction définie ci-avant ne permet donc pas la réduction de 3–SAT vers 2–SAT.

1.2.3 2–SAT, un problème polynomial

1. coucou
2. coucou

1.3 Calculabilité

1. La stratégie d'énumération des couples d'entier peut être visualisée sur un graphique en suivant les diagonales successives comme sur l'image[1] suivante :

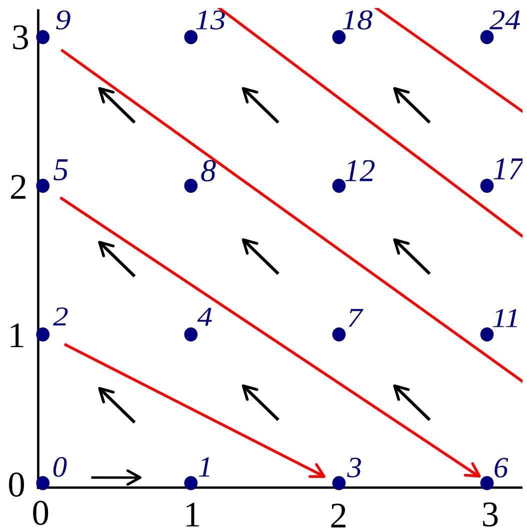

FIG. 1.1 – La fonction de couplage de Cantor établit une bijection de $\mathbb{N} * \mathbb{N}$ dans $\mathbb{N}$.

Soit $(x, y) \in \mathbb{N} * \mathbb{N}$ un couple. On trie par ordre lexicographique $(x + y)$. Ainsi on obtient le tableau suivant :

(x, y)	$(0,0)$	$(1,0)$	$(0,1)$	$(2,0)$	$(1,1)$	$(0,2)$	$(3,0)$	$(2,1)$	$(1,2)$	$(0,3)$	...
$(x + y)$	0	1	1	2	2	2	3	3	3	3	...
$c_2(x + y)$	0	1	2	3	4	5	6	7	8	9	...

2. **Fonction de codage**

$$c_2(x, y) = \frac{(x + y)(x + y + 1)}{2} + y$$

Fonctions de décodage Les fonctions de décodage ne peuvent pas être décrites sous la forme de formules arithmétiques. Elles nécessitent

[1] Image provenant de Wikipedia, ce fichier est disponible selon les termes de la licence Creative Commons.

l'algorithme suivant :

Algorithme 1: CalculXY(z)

Données : z *// Rang du couple (x,y)*

```
début
    s ← 0;
    t ← 0;
    tant que s ⩽ z faire
        s ← t*(t+1)/2;
        t ← t + 1;
    t ← t − 2;
    z ← t*(t+1)/2;
    y ← z − s;
    x ← t − y;
    retourner Couple(x,y);
```

3. **Codage des triplets** : il peut avoir lieu de manière récursive :

$$c_3(x, y, z) = c_2(x, c_2(y, z))$$

Généralisation au codage des k-uplets :

$$c_k(x_1, x_2, \dots, x_k) = c_2(x_1, c_{k-1}(x_2, \dots, x_k))$$

Avec : $$c_2(x, y) = \frac{(x + y)(x + y + 1)}{2} + y$$

4. Prenons une suite $r = (r_1, r_2, r_3, \dots)$ qui énumère les réels de l'intervalle $[0; 1]$, puis créons un réel x compris dans cet intervalle, tel que si la n$^{\text{ième}}$ de r_n est égale à 1, la n$^{\text{ième}}$ décimale de x est égale à 2. Dans la cas contraire, la n$^{\text{ième}}$ décimale de x est égale à 1.

On obtient sur cet exemple :

r_1	=	0	,	**4**	2	9	6	4	6	1	…
r_2	=	0	,	2	**7**	3	2	9	4	0	…
r_3	=	0	,	6	4	**1**	1	5	1	2	…
r_4	=	0	,	3	0	5	**9**	0	4	3	…
r_5	=	0	,	9	1	3	3	**1**	8	2	…
r_6	=	0	,	0	2	0	8	3	**2**	7	…
r_7	=	0	,	2	5	7	3	6	4	**0**	…
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
				$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
x	=	0	,	1	1	2	1	2	1	1	…

Le réel x ne peut pas être énuméré par la suite r car il diffère de sa première décimale dans r_1, de sa deuxième décimale dans r_2, … de sa n$^{\text{ième}}$ décimale dans r_n. Pourtant le réel x est clairement dans l'intervalle $[0; 1]$.

L'ensemble des éléments de l'intervalle $[0; 1]$ ne sont donc pas dénombrables, donc pas énumérables. On ne peut donc pas trouver de fonction de codage pour cet ensemble.

On peu généraliser à l'ensemble $\mathbb{R}$: $[0; 1]$ étant inclus dans $\mathbb{R}$, et $[0; 1]$ n'étant pas dénombrable, l'ensemble $\mathbb{R}$ n'est pas dénombrable.

Chapitre 2

Partie pratique