Rapport: Cook's versus Valiant's Hypothesis, Peter Bürgisser

Omar Fawzi

17 novembre 2006

1 Introduction

Le but de cet article est d'effectuer une comparaison entre les classes de compléxité Turing (**P**, **NP** ...) et les classes de compléxité algébrique définies par Valiant (**VP**, **VNP**, ...). Cet article montre que l'hypothèse de Valiant à savoir $\mathbf{VP} \neq \mathbf{VNP}$ est lié à l'hypothèse de Cook sur la compléxité Turing $\mathbf{P} \neq \mathbf{NP}$. En effet Peter Bürgisser montre que si $\mathbf{VP}_K = \mathbf{VNP}_K$ pour un certain corps K, alors la hiérarchie polynomiale s'effondrerai au niveau 2. L'article montre en fait un résultat plus fort, les classes $\mathbf{VNP}_K$ et $\mathbf{VP}_K$ sont encadrées par des classes de compléxité classique non-uniforme, c'est à dire en s'autorisant un conseil de taille polynomiale.

Les résultats les plus difficiles sont les inclusions des classes Valiant dans les classes classiques. La difficulté de cette partie tient au fait que le modèle Valiant est non-uniforme, mais cette non-uniformité est non bornée dans le sens où l'on peut avoir des constantes aussi grandes que l'on veut puisque dans le modèle algebrique il n'y a pas de notion de taille pour un élement du corps. Pour résoudre ce problème, on a recours à la géométrie algébrique. L'auteur démontre un théorème de géométrie algébrique dont on ne présentera pas la preuve qui utilise beaucoup d'outils de ce domaine. On montrera quand même pourquoi penser à avoir recours à de telle méthode et le lien avec l'élimination des constantes.

2 Définitions et énoncés des résultats

Pour pouvoir comparer le modèles de calcul des SLP avec le calcul Turing, il est nécessaire de faire une limitation pour le modèle Valiant. En effet, vu que la taille des constantes n'importe pas dans ce modèle, des fonctions constantes sur $\{0,1\}^n$ égales à une valeur très grandes devant n (par exemple plus qu'exponentiel en n, 2^{2^n}) ne peuvent surement pas etre calculé par une machine de Turing en temps polynomiale. De plus pour définir une notion de taille d'un élément du corps on se ramène à des entiers. On définit ainsi la partie booléenne d'une p-famille comme ceci :

Définition 1. On définit une string fonction comme une suite de fonction (ϕ_n) telle que $\phi_n : \{0,1\}^n \to \{0,1\}^{m(n)}$ avec $m(n)$ bornée polynomialement. Soit (f_n) une p-famille avec $f_n \in K[X_1, \ldots, X_n]$. On dit que (f_n) a une partie booléene si il existe une string fonction (ϕ_n) telle que :

– Si K est de caractéristique nulle

$$\forall x \in \{0,1\}^n, f_n(x) = \sum_{i=0}^{m(n)} \phi_{n,i}(x) 2^{i-1}$$

– Si K est de caractéristique p, il faut que $m(n) = m = E(\log p) + 1$

$$\forall x \in \{0,1\}^n, f_n(x) = \sum_{i=0}^{m} \phi_{n,i}(x) 2^{i-1} [p]$$

Remarque.
– On confondra souvent une string fonction (ϕ_n) et la fonction $\mathbb{N} \to \mathbb{N}$ qui lui est associée.
– Notons que quelque soit le corps K de caractéristique nulle considéré, on impose que les valeurs sur $\{0,1\}^n$ soit entières (les entiers ont bien un sens en caractéristique 0). Notons aussi que si (f_n) a une partie booléene alors on a $f_n(x) < 2^{m(n)}$ pour $x \in \{0,1\}^n$. De plus si f_n ne prend que des valeurs entières positives cette condition devient suffisante. On peut aussi se poser la question est ce que etant donné une partie booléene on

peut reconstruire le polynome f_n. La partie booléene fixe 2^n valeur de la fonction f_n, ce qui ne suffit pas en général à définir un polynome de degré quelconque.
- Pour le cas de caractéristique p, la motivation de cette définition n'est pas très claire mais il faut bien imposer une restriction pour les corps infini de caractéristique p.

En ce qui concerne maintenant le coté classique, il faut regarder les classes non-uniforme de fonctions (et non simplement de langages). On donne donc quelques définitions

Définition 2. On définit les classes de compléxité suivante
- Pour une classe $\mathbf{C}$, on définit $\mathbf{C}/poly$ l'ensemble des fonctions f tels qu'il existe une fonction $\alpha : \mathbb{N} \to \{0,1\}^*$ vérifiant $|\alpha(n)|$ polynomialement bornée et une fonction $g \in \mathbf{C}$ tels que

$$f(x) = g(x, \alpha(|x|)$$

- Les classes $\mathbf{FNC}^i$ les string fonctions calculables par des circuits de taille polynomiale et de profondeur $O(\log^i n)$
- La classe $\mathbf{FP}$ est l'ensemble des fonctions calculables en temps polynomiale par une machine de Turing. Notons que ces fonctions sont des string fonctions.
- La classe $\#\mathbf{P}$ est la classe de comptage associé à $\mathbf{NP}$. Une fonction $f \in \#\mathbf{P}$ si il existe une machine de Turing non déterministe sur une entrée x a $f(x)$ calcul acceptant de x. Notons que ces fonctions sont aussi des string fonctions. On définit aussi $\#_p\mathbf{P}$ pour p premier comme les fonctions de $\#\mathbf{P}$ modulo p.

On définit aussi la hiérarchie polynomiale dont on aura besoin pour exprimer les résultats qui permettent de comparer l'hypothèse de Valiant et celle de Cook.

Définition 3 (Hiérarchie polynomiale)**.** On définit la hiérarchie polynomiale comme ceci

$$\Sigma_0 = \Delta_0 = \mathbf{P}$$

puis pour $i \geq 0$,

$$\Sigma_{i+1} = \mathbf{NP}^{\Sigma_i}$$

On dit que le hiérarchie polynomiale s'effondre au niveau k si $\Sigma_k = \Sigma_{k+1}$. Dans ce cas on a pour tout $l \geq k, \Sigma_l = \Sigma_k$. Notons qu'on a $\Sigma_1 = \mathbf{NP}^{\mathbf{P}} = \mathbf{NP}$ puisqu'il suffit de simuler l'oracle.

Le résultat principal de l'article est le suivant

Théoreme 1. *Sous l'hypothèse de Riemann généralisé, on a*

1. *Si K est de caractéristique nulle*

$$\boldsymbol{FNC}^1 \subset BP(\boldsymbol{VP}_K) \subset \boldsymbol{FNC}^3$$

$$\#\boldsymbol{P}/poly \subset BP(\boldsymbol{VNP}_K) \subset \boldsymbol{FP}^{\#\boldsymbol{P}}/poly$$

2. *Si K est fini de caractéristique p alors*

$$\boldsymbol{FNC}^1 \subset BP(\boldsymbol{VP}_K) \subset \boldsymbol{FNC}^2$$

$$\#_p\boldsymbol{P}/poly = BP(\boldsymbol{VNP}_K)$$

Remarque.
- C'est seulement dans le cas infini que l'on utilise l'hypothèse de Riemann généralisée
- Pour le cas infini, ce sont les inclusions de droite qui sont plus difficile et utilisent un résultat de géométrie algébrique et un résultat de parallélisation, pour le cas fini on a pas le problème des constantes. Mais après pour voire les conséquences de $\mathbf{VNP}_K = \mathbf{VP}_K$ le cas fini est plus difficile puisqu'il faut se ramener à $\mathbf{NP}/poly$.

Nous allons donner les grandes ligne de la démonstration de ce théorème sans toutefois rentrer dans la démonstration de résultat intermédiaires compliqués. On va essayé aussi de voire pourquoi on a recours à des résultats de géométrie algébrique.

Nous commençons donc par examiner les conséquences de ce théorème en terme de lien entre l'hypothèse de Cook et celle de Valiant.

3 Conséquences du théorème

Du théorème 1 on peut déduire immédiatement que si pour K de caractéristique nulle, si $\mathbf{VP}_K = \mathbf{VNP}_K$ alors on a

$$\#\mathbf{P}/poly \subset BP(\mathbf{VNP}_K) = BP(\mathbf{VP}_K) \subset \mathbf{FNC}^3/poly \subset \mathbf{FP}/poly \subset \#\mathbf{P}/poly$$

En particulier pour les classes de langages on a $\mathbf{P}/poly = \mathbf{NP}/poly$. Il a été montré dans [KJ82] que ceci implique que que la hiérarchie polynomiale s'effondrerait au niveau 2, c'est à dire que $\mathbf{P}^{\mathbf{NP}} = \mathbf{NP}^{\mathbf{NP}}$. On a donc

Pour le cas de caractéristique p il faut travaillé un peu plus que pour aboutir à la même conclusion. Il faut trouver un lien entre $Mod_p\mathbf{NP}$ qui est la classe de langages associé à $\#_p\mathbf{P}$ i.e. les langages de la forme $\{x \in \{0,1\}^* | \phi(x) \equiv 1[p]\}$ pour $\phi \in \#\mathbf{P}$.

On a alors

Théoreme 2. *Pour tout p premier*

$$\boldsymbol{NP}/poly \subset Mod_p\boldsymbol{NP}/poly$$

Preuve. Il suffit de montrer que $\mathbf{NP} \subset Mod_p\mathbf{NP}/poly$. Ensuite pour $\mathbf{NP}/poly$, il suffit de rajouter le même conseil à la machine qui simule. Soit L un langage de $\mathbf{NP}$. Alors par $\mathbf{NP}$-complétude de SAT, pour tout $x \in L$ il existe une formule booléene ϕ_x telle que

$$\#\phi_x > 0 \Leftrightarrow x \in L$$

où $\#\phi$ désigne le nombre d'assignements qui satisfont ϕ.

Si on arrive à trouver une fonction qui à une formule ϕ exprimé en forme normale conjonctive (cnf) associe une formule χ exprimé en forme normale conjonctive telle que $\#\phi_x > 0 \Leftrightarrow \#\chi \equiv 1[p]$ alors c'est bon. Il faut bien sur que cette transformation s'effectue en temps polynomial avec conseil polynomial, et que la taille de χ soit polynomiale en celle de ϕ.

On utilise un résultat de Valiant et Vazirani dans [VV86] qui dit qu'il existe une fonction γ calculable en temps polynomiale qui à une cnf ϕ ayant n variables et une chaine de longueur n associe $\gamma : (\phi, w \in \{0,1\}^n) \mapsto \Phi$ qui vérifie les 2 propriétés suivantes :

- $\forall w, \#\phi = 0 \Rightarrow \#\gamma(\phi, w) = 0$
- si $\#\phi > 0$ il y a une proportion de w qui envoie sur Φ avec $\#\Phi \neq 1$. Plus précisement si on met sur l'ensemble des $w \in \{0,1\}^n$ la distribution uniforme alors on a

$$\#\phi > 0 \Rightarrow P(\{w \in \{0,1\}^n, \gamma(\phi, w) \neq 1\}) \leq 1 - \frac{1}{4n}$$

Que peut on faire avec ce Φ ? Si on était sûr qu'il existe un w tel que $\forall \phi, \gamma(\phi, w) = 1$, ce serait bon, on n'a qu'a prendre w comme conseil (qui est de taille n). On a $P(\gamma(\phi, w) = 1) \geq \frac{1}{4n}$, ceci signifie qu'il existe un w qui convient, mais celui-ci peut être différent suivant ϕ. L'idée est donc en utilisant plusieurs w et en combinant les formules obtenus, on diminue la probabilité que le nombre d'assignements satisfaisant la formule résultante ne soit pas congrus à 1 modulo p.

Fixons d'abord ϕ, on prend $q \in \mathbb{N}$ mots w_i de taille n et on regarde leur image par γ, cela nous donne des cnf Φ_i. Alors on peut montrer qu'on sait trouver en temps polynomial une cnf χ de taille polynomiale qui vérifie

$$\#\chi = 1 + \prod_{i=1}^{q}(p - 1 + \#\Phi_i)$$

Pour montrer ceci on montre qu'on peut trouver une cnf qui fasse le produit, la somme et n'importe quelle constante. Notons que si $\#\chi \equiv 1[p]$ alors pour tout i, $\#\Phi_i \equiv 1[p]$. Supposons maintenant que $\#\phi > 0$, on suppose de plus que l'on choisit les w_i de manière indépendante, alors on a

$$P(\#\chi \neq 1[p]) = P(\forall i, \#\Phi_i \neq 1) \leq (1 - \frac{1}{4n})^q$$

Résumons ce qu'on a jusqu'à présent, pour chaque cnf possible de taille n on a une fraction des images plus grande que $1 - (1 - \frac{1}{4n})^q$ qui vont sur des χ telles que $\#\chi \equiv 1[p]$. Mais les w_i en question qui donnent cette partie des images dépendent de la formule ϕ choisit, notons $E_\phi = \{(w_i), \#\chi_{\gamma(\phi, w_i)} \equiv 1[p]\}$ cet ensemble. Comment faire en

sorte qu'il existe au moins une liste $w_1, \ldots, w_q$ qui envoit **toutes** les cnfs ayant n variables sur une certaine formule χ qui vérifie $\#\chi \equiv 1[p]$? En d'autres termes comment s'assurer que $\bigcap_\phi E_\phi$ est non vide ?

Si on fait en sorte que pour tout ϕ, $|^c E_\phi| < \frac{2^{nq}}{N}$ où N est le nombre de forme normale conjonctive ayant n variables, alors c'est gagné, en effet dans ce cas la réunion ne peut pas faire l'espace entier. Comment assurer cela, il faut imposer que $2^{nq}(1-\frac{1}{4n})^q < \frac{2^{nq}}{N}$ soit

$$N(1-\frac{1}{4n})^q < 1$$

Pour cela on peut augmenter q mais pas trop quand même, il faut que q reste polynomial en la taille de l'entrée. Pour voir si c'est possible, il faut évaluer N. Au lieu d'evaluer N, on va s'interesser à N_a le nombre N_a de cnfs de taille a. Une cnf de taille a se code en espace polynomiale en a, il existe donc c tel que $N_a \leq 2^{a^c}$. Soit a la taille de notre cnf, $a \geq n$. On en déduit qu'il suffit de choisir q dépendant de a, un polynome assez grand en a pour qu'on ait l'inégalité. En effet, $(1-\frac{1}{4n})^q = (1-\frac{1}{4n})^{nq/n} \leq e^{\frac{-1}{4}\frac{q}{n}}$ peut écraser N_a avec q bien choisi.

Comment va procéder notre machine ? D'abord étant donnée une cnf ϕ avec un conseil $w_1, \ldots, w_{q(|\phi|)}$, elle calcule l'image par la fonction de Valiant et Vazirani de chacun des w_i, ensuite on construit χ qui a les bonnes propriétés i.e. $\#\phi_x > 0 \Leftrightarrow \#\chi \equiv 1[p]$. Notons que le conseil ne dépend que de la taille de l'entrée, et qu'il est de longueur polynomiale (un nombre polynomial de mots de longueur inférieur à la taille de l'entrée. □

En utilisant ce résultat, on aboutit à la conclusion exactement de la même façon que pour la caractéristique nulle.

Corollaire 1. *Si on a $\boldsymbol{VP}_K = \boldsymbol{VNP}_K$ pour K de caractéristique nulle ou fini alors on aura*

$$\boldsymbol{P}/poly = \boldsymbol{NP}/poly$$

et la hiérarchie polynomiale s'effondre au niveau 2.

4 Démonstration du théorème

Nous allons d'abord commencer par les inclusions de gauche, qui ne nécessite pas l'hypothèse de Riemann.

1. $\mathbf{FNC}^1 \subset BP(\mathbf{VP}_K)$
   Soit (C_n) une famille de circuits de taille $s(n)$ et de profondeur $d(n) = O(\log n)$ calculant la string fonction (ϕ_n). On cherche un polynome f_n calculé par un SLP de taille polynomiale vérifiant

   $$\forall x \in \{0,1\}^n, f_n(x) = \sum_{i=0}^{m(n)} \phi_{n,i}(x) 2^{i-1}$$

   De plus pour s'assurer que (f_n) est bien dans $\mathbf{VP}_K$ il faut verifier que $deg(f_n)$ est polynomialement borné en n. Pour ceci, il suffit de remplacer les opérateur booléen $\wedge$, $\vee$ et $\neg$ par des additions et des multiplications dans K. En utilisant les égalités $x \wedge y = xy, x \vee y = xy - x - y, \neg x = 1 - x$ on obtient alors un circuit SLP de taille $s'(n) \leq 3s(n)$ et de profondeur $d'(n) \leq 2d(n)$. Ce programme SLP calcule bien un polynome f_n qui vérifie bien la propriété demandée. De plus on a $deg(f_n) \leq 2^{d'(n)}$ qui est polynomialement borné donc $(f_n) \in \mathbf{VP}_K$. Notons que cette preuve marche pour tout corps, en effet on n'utilise que les 2 éléments neutres.
2. $\#\mathbf{P}/poly \subset BP(\mathbf{VNP}_K)$
   On commence par montrer que $\#\mathbf{P} \subset BP(\mathbf{VNP}_K)$. Soit $\phi \in \#\mathbf{P}$, alors avec la même démonstration que celle de la **NP**-complétude de 3SAT, on montre que pour tout x il existe une formule booléene ψ_x sous forme 3-normale-conjonctive ayant $p(|x|)$ variables calculé en temps polynomiale telle que

   $$\phi(x) = |\{y \in \{0,1\}^{p(|x|)}, \psi_x(y) = 1\}|$$

   Ce que l'on veut c'est trouver $(f_n) \in \mathbf{VNP}_K$ i.e. s'ecrivant $f_n(x) = \sum_{e \in \{0,1\}^{p(n)}} g_n(x,e)$ avec $(g_n) \in \mathbf{VP}_K$ et calculant ϕ. Cette somme a l'aire de bien convenir vu qu'on veut compter le nombre de valeurs qui satisfont une formule. Le problème est que le ψ_x dépend de x, il faudrait que celui-ci ne dépende que de la taille de x, puisque qu'on a une fonction pour chaque taille.

En fait en regardant la preuve de Cook, on peut voir que la formule booléene calculée ψ_x ne dépend de x que par l'intermédiare de variable $X_1, \ldots, X_n$ en imposant $X_1 = x_1, \ldots, X_n = x_n$. Et tout le reste de la construction ne dépend que de la machine de Turing qui calcule la fonction ϕ. On peut donc dire qu'il existe des formules booléenes ψ_n telles que

$$\forall x \in \{0,1\}^n, \phi(x) = |\{y \in \{0,1\}^{p(n)}, \psi_n(x,y) = 1\}|$$

En exprimant chaque clause par un polynome de degré au plus 3 (une clause $u \vee v \vee w$ peut s'exprimer en un polynome en u, v et w en utilisant les égalités $u \wedge v = uv, u \vee v = uv - u - v, \neg u = 1 - u$), on voit que l'on peut traduire cette formule booléene ψ_n en un polynome g_n (on fait le produit des polynomes obtenus pour chaque clause) telle que

$$\forall x \in \{0,1\}^n, \psi_n(x) = 1 \Leftrightarrow g_{n+p(n)}(x) = 1$$

Ce polynome est bien calculable en temps polynomial et a un degré $\leq 3p(n)$. En posant donc

$$f_n(X) = \sum_{y \in \{0,1\}^{p(n)}} g_{n+p(n)}(X, y)$$

on a bien

$$\forall x \in \{0,1\}^n, f_n(x) = |\{y \in \{0,1\}^{p(n)}, \psi_n(x,y) = 1\}| = \phi(x)$$

On a donc démontrer que $\#\mathbf{P} \subset BP(\mathbf{VNP}_K)$, il reste maintenant à considérer les conseils. Par définition il existe $\Phi \in \mathbf{VNP}_K$ et $\alpha(n)$ de taille polynomialement bornée telles que $\phi(x) = \Phi(\langle x, \alpha(|x|)\rangle)$. Notons $t(n)$ la taille de l'encodage $\langle x, \alpha(|x|)\rangle$, on a clairement $t(n)$ qui est polynomialement borné. On applique ce qu'on a montrer à Φ, on a donc pour $x \in \{0,1\}^n$

$$\phi(x) = \Phi(\langle x, \alpha(n)\rangle) = f_{t(n)}(\langle x, \alpha(n)\rangle)$$

On pose alors $g_n(X) = f_{t(n)}(\langle x, \alpha(n)\rangle)$ et vu que $\langle ., . \rangle$ est polynomiale on a que (g_n) est une p-projection de (f_n) donc $(g_n) \in \mathbf{VNP}_K$ et sa partie booléene est bien ϕ.

Notons qu'on a utilisé ici que K est de caractéristique nulle, lorsqu'on a défini f_n par une somme. Mais on voit bien que pour K de caractéristique p on aura avec exactement la même preuve $\#_p\mathbf{P}/poly \subset BP(\mathbf{VNP}_K)$

3. $BP(\mathbf{VP}_K) \subset \mathbf{FNC}^3$

   D'abord pour K de caractéristique nulle. La difficulté pour montrer ce résultat ou un résultat de ce type c'est que les constantes utilisé par le SLP qui calcule (f_n) peuvent être "très grandes" en fonction de n et aussi que ces constantes appartiennent au corps K en général (et ce ne sont pas que des entiers), donc on ne pourra pas coder ces constantes dans un conseil polynomial pour une machine de Turing. L'idée est donc de se ramener du corps K qui est infini à un corps fini $\mathbb{F}_{p_n}$ avec p_n assez grand pour que $f_n(x) \mod p_n = f_n(x)$ pour $x \in \{0,1\}^n$ et assez petit pour qu'on puisse coder les constantes de $\mathbb{F}_{p_n}$ dans un espace polynomiale.

   On commence par supposer que les constantes utilisés sont entières, ceci pour montrer l'idée de réductions des constantes. On part donc de $(f_n) \in \mathbf{VP}_K$ qui a une partie booléene. On veut trouver une machine de Turing M qui sur l'entrée $x, \alpha(|x|)$ donne la sortie $f_{|x|}(x)$. On prend pour M une machine que simule le calcul du SLP qui est codé par $\alpha(|x|)$. Cette machine fonctionne bien en temps polynomiale pourvu que la description du SLP soit polynomiale. Il reste donc à bien décrire le SLP en question, prendre le SLP de f_n ne marche pas puisque qu'il peut y avoir des constantes trop grande. Mais maintenant on va utiliser le fait que (f_n) a une partie booléene. On a donc l'existance d'un polynome $t(n)$ tel que $\forall x \in \{0,1\}^n, f_n(x) < 2^{t(n)}$. On prend donc simplement pour SLP celui qui calcule f_n en remplaçant les constantes par leur reste modulo $2^{t(n)}$. On a donc trouver le conseil α qui convient. Ceci montre qu'avec qu'un circuit SLP avec des constantes entières peut se calculer en temps polynomiale non-uniforme.

   Maintenant comment l'article montre il dans le cas général l'inclusion $BP(\mathbf{VP}_K) \subset \mathbf{FNC}^3$? D'abord on utilise un résultat de parallélisation démontré dans [VSBR83].

   Théoreme 3. *Soit f un polynome a n variables de degré d et de compléxité $L(f)$. Alors f peut être calculée par un circuit SLP de taille $O(d^6 L(f)^3)$ et de profondeur $O(\log(dL(f))\log(d) + \log(n))$.*

On déduit de ce théorème que f_n peut se calculer par un SLP de profondeur $O(\log^2(n))$. Notons $y_1, \ldots, y_m$ les constantes utilisés. Et notons $F_n(X, Y)$ le polynome obtenu en remplaçant les constantes y_i par des variables Y_i. On cherche maintenant des $y'_1, \ldots, y'_m$ qui soit des petits entiers et qui vérifie aussi $\forall x \in \{0,1\}^n, F_n(x, y'_1, \ldots y'_m) = f_n(x)$. Pour cela on considère le système S_n suivant

$$F_n(x, Y) - f_n(x) = 0$$

On sait que ce système a une solution y, mais notre but est de trouver un petite solution entière. L'ensemble des solutions de ces équations ce que l'on appelle une variété algébrique. La géométrie algébrique est l'étude de ses variétés. En utilisant des notions de géométrie algébrique, l'article démontre le théorème suivant

Théoreme 4. *Soit $f_1, \ldots, f_s \in \mathbb{Z}[X_1, \ldots, X_n]$ des polynomes de degré borné par d et de poids borné par w. (Le poids d'un polynome est la somme des valeurs absolues de ses coefficients). On note S le système*

$$f_1 = 0, \ldots, f_s = 0$$

Soit S un système qui a au moins une solution sur $\mathbb{C}$. On note $\pi_S(x)$ le nombre de nombres premiers $p \leq x$ tel que S ait au moins une solution dans $\mathbb{F}_p$. On a

$$\pi_S(x) \geq \frac{\pi(x)}{d^{O(n)}} - x^{1/2}\log(wx)$$

On souhaite donc appliquer ce théorème à notre système afin d'obtenir un p_n qui convient. D'abord on montre que $deg(F_n(x, Y) - f_n(x))$ et $\log(wt(F_n(x, Y) - f_n(x)))$ sont bornées par $2^{O(\log^2(n))}$. Ensuite il est important de remarque que le polynome $F_n(x, Y) - f_n(x)$ est à coefficient entier. Puis, en utilisant le fait que $\pi(x) \sim \frac{x}{ln(x)}$ on a pour en choisisant c convenablement et pour n suffisament grand,

$$\pi_{S_n}(2^{n^c}) > 2^{t(n)}$$

Il existe donc un nombre premier p_n tel que S_n soit résoluble dans $\mathbb{F}_p$ qui vérifie $2^{t(n)} \leq p_n \leq 2^{n^c}$. On a donc un p_n qui vérifie les conditions qu'on voulait.

Il nous faut maintenant construire un circuit qui fait des calculs dans $\mathbb{F}_{p_n}$, notons que l'addition et la multiplication dans $\mathbb{F}_p$ se fait avec des circuits de taille $O(\log p)$ et de profondeur $O(\log\log p)$ [KR]. Donc chaque porte d'addition ou de multiplication dans le circuit SLP va être remplacé par un circuit booléen de taille $O(\log p)$ et de profondeur $O(\log\log p)$. On voit donc que la taille du circuit va rester polynomiale puisque les constantes y'_i sont aussi de taille polynomiale. La profondeur du circuit va être muliplié par $O(\log\log p_n)$ ce qui va donner un circuit de profondeur $O(\log^3(n))$.

Remrquons que le résultat de parallélisation permet d'obtenir l'inclusion dans $\mathbf{FNC}^3$ mais sans ce résultat on peut montrer l'inclusion dans $\mathbf{FP}/poly$ certes plus faible mais suffisante pour montrer le corollaire 1.

Notons que si K est fini on n'a qu'a simule le SLP directement puisqu'il y a un nombre fini constantes qu'on peut coder. En utilisant le résultat de parallélisation on peut se ramener à un circuit de profondeur $O(\log^2(n))$. D'où l'inclusion dans $\mathbf{FNC}^2$ si K est fini.

4. $BP(\mathbf{VP}_K) \subset \mathbf{FP}^{\#\mathbf{P}}/poly$

   La preuve que $BP(\mathbf{VP}_K) \subset \mathbf{FP}^{\#\mathbf{P}}/poly$ pour $car(K) = 0$ du théorème 1 est du même genre, elle est basé sur le théorème 4.

   Et pour K fini montrons que $BP(\mathbf{VNP}_K) \subset \#_p\mathbf{P}/poly$, ce qui montrera que $BP(\mathbf{VP}_K) = \#_p\mathbf{P}/poly$. On note $\mathbb{F}_{p^e}$ le corps de cardinal p^e. On peut toujours considérer ce corps comme un espace vectoriel sur $\mathbb{F}_p$. Soit maintenant $(f_n) \in BP(\mathbf{VNP}_{\mathbb{F}_{p^e}})$, on peut écrire $f_n(X) = \sum_y g_{u(n)}(X, y)$ avec $(g_n) \in \mathbf{VP}_{\mathbb{F}_{p^e}}$. On peut décomposer $g_n(x)$ dans une base $(b_1 = 1, b_2, \ldots b_e)$ de $\mathbb{F}_{p^e}$. On a alors $g_n(x) = \sum_{i=1}^n g_{n,i}(x)b_i$ avec les $g_{n,i}$ des polynomes à coefficients dans $\mathbb{F}_p$. Mais on sait que sur $\{0,1\}^n$, f_n ne prend que des valeurs dans $\mathbb{F}_p$ on en déduit donc que On a donc

$$\forall x \in \{0,1\}^n, f_n(x) = \sum_e \sum_{i=1}^n g_{u(n),i}(x, e)b_i = \sum_e g_{u(n),1}(x, e)$$

   Mais les $(g_{n,i})_n$ sont dans $\mathbf{VP}_{\mathbb{F}_p}$ donc par l'inclusion précedente dans $\mathbf{FP}/poly$. Il existe donc ϕ qui calcule $BP(g_n)$. On a donc

$$\forall x \in \{0,1\}^n, f_n(x) = \sum_e \phi(x,e) \mod p$$

Pour finir, il suffit juste de montrer le lemme suivant.

Lemme 1. *Si une string fonction (ϕ_n) est dans **FP**/poly, alors (Φ_n) définie par*

$$\Phi_n(x) = \sum_{i=1}^{m(n)} \sum_{e \in \{0,1\}^{u(n)-n}} \phi_{u(n),i}(x_1, \dots x_n, e_1, \dots, e_{u(n)}) 2^{i-1}$$

est dans $\#\mathbf{P}$/poly.

Preuve. Montrons pour cela qu'il existe un langage R calculable en temps polynomial tel que pour tout n,

$$\forall x \in \{0,1\}^n, \Phi_n(x) = \sum_e \phi_n(x_1, \dots x_n, e_1, \dots, e_{u(n)}) = |\{y, (x,y) \in R\}|$$

On définit R comme suit $R = \bigcup R_n$ avec

$$(x,y,z,e) \in R_n \Leftrightarrow \begin{cases} x \in \{0,1\}^n, y \in \{0,1\}^{m(u(n))}, z \in \{0,1\}^{m(u(n))}, e \in \{0,1\}^{u(n)-n} \\ \exists i \in \{1, \dots, m(u(n))\}, \forall k, y_k = \delta_{ik} \text{ et } \forall k \geq i, z_k = 0 \text{ et } \phi_{u(n),i}(x,e) = 1 \end{cases}$$

On voit ici que la variable y sert à compter i, la variable z sert à multiplier par 2^{i-1}. On voit bien que R vérifie la propriété cherchée, de plus il est clair que R est calculable en temps polynomial. □

On a donc une fonction $\Phi(x) = \sum_e \phi(x,e) \in \#\mathbf{P}/poly$ qui vérifie $\forall x \in \{0,1\}^n, f_n(x) = \Phi(x) \mod p$. Ce qui signifie la partie booléene de (f_n) qui est $\Phi \mod p$ est dans $\#_p\mathbf{P}/poly$.

5 Commentaires

J'ai essayé dans ce rapport de présenter les idées importantes de l'article (excepté la partie géométrie algebrique qui m'a semblé difficile) et d'éclaircir quelque démonstrations qui ne sont pas très détaillé. J'ai essayé aussi d'introduire des transitions pour les définitions et les idées que je trouvais un peu "brutales". En effet, les choses apparaissait déconnectés au début, par exemple l'idée d'utiliser un tel théorème de géométrie algébrique semblait "sortir du chapeau". Mais après une meilleur compréhension de l'article, il se trouve que tout ceci est très naturel, c'est ce que j'ai essayé de montrer dans ce rapport.

Au niveau du contenu, je trouve que la notion de partie booléenne n'est pas très naturelle. Il est clair que vu que le modèle Valiant est non-uniforme, il est naturel qu'on impose l'égalité sur les entrées de même taille, mais je me demande pourquoi on impose la positivité et pourquoi on ne prend pas simplement l'ensemble des fonctions telles qu'il existe un polynome $t(n)$ tel que $\forall x \in \{0,1\}^n, f_n(x) \in \{-2^{t(n)}, \dots, 2^{t(n)}\}$. Notons que les preuves de cet articles marche tout aussi bien dans ce cas. On voit aussi que la partie booléenne d'une fonction ne contient pas toute l'information de celle-ci. En effet comme je l'ai mentionné dans la remarque, on ne peut pas en général revenir au polynome à partir de sa partie booléenne. Mais je ne crois pas que l'on puisse eviter ceci.

La question qui reste non résolue est le cas où K est infini de caractéristique p.

Références

[KJ82] R.M. Karp, R.J. Lipton, Turing machines that take advice, Logic and Algorithmic : An international Symposium held in honor of Ernst Specker, Monogr. No. 30 de l'Enseign. Math., 1982, pp. 255-273.

[VSBR83] L.G. Valiant, S. Skyum, S. Berkowitz, C. Rackoff, Fast parallel computation of polynomials using few processors, SIAM J. Comp. 12(4) (1983) 641-644.

[VV86] L.G. Valiant, V.V. Vazirani, NP is as easy as detecting unique solutions, Theoret. Comp. Sci. 47 (1986) 85-93.

[KR] R.M. Karp, V. Ramachandran, Parallel algorithms for shared-memory machines, in : J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. A, Elsevier, Amsterdam, 1990, pp. 869-941 (Chapter 17)