Université Paris Diderot - Paris 7
UFR de Mathématiques

Mémoire d'habilitation à diriger des recherches

Hervé Fournier

Complexité du calcul de polynômes et de problèmes géométriques

---

Rapporteur interne :
Arnaud Durand

Rapporteurs externes :
Peter Bürgisser
Dima Grigoriev

Jury :
Olivier Bournez
Arnaud Durand
Christoph Dürr
Etienne Grandjean
Dima Grigoriev
Nitin Saxena

Table des matières

Introduction

(English version of the introduction is given below.)

La complexité a pour objet de préciser les ressources nécessaires pour résoudre des problèmes algorithmiques. En complexité booléenne, les ressources de base sont le temps de calcul ou l'espace mémoire, pour lesquels il existe des théorèmes de hiérarchie. D'autres ressources moins naturelles ont été définies pour capturer des problèmes particuliers : citons le non-déterministe avec la classe NP pour les problèmes d'optimisation combinatoire, ou encore le comptage défini par Valiant [Val79a] pour lequel le calcul du permanent a des propriétés de complétude.

En complexité algébrique, on cherche à déterminer le nombre d'opérations nécessaires pour calculer des polynômes. La classe VP correspond aux polynômes qui peuvent être calculés avec un nombre polynomial d'opérations. Comme dans le cadre booléen, des ressources moins naturelles sont aussi apparues comme la possibilité de faire des sommes exponentielles, menant à la définition de la classe VNP [Val79b]. Comprendre si ces sommes exponentielles permettent de calculer plus de polynômes correspond au problème $\mathsf{VP} \stackrel{?}{=} \mathsf{VNP}$. La classe VNP contient de nombreux polynômes définis en combinatoire comme le permanent : il n'est donc pas étonnant que cette classe entretienne des liens étroits avec les classes booléennes de comptage. En particulier, les classes de comptage peuvent être définies par arithmétisation à partir de classes booléennes [All04] (ce qui consiste à remplacer les portes *et* et *ou* des circuits booléens par les opérations arithmétiques produit et somme).

Même si la complexité algébrique a connu des avancées rapides ces dernières années (dérandomisation vs. bornes inférieures [KI04] ; simulation de formules arithmétiques par des formules de petites profondeur [AV08] ; bornes inférieures sur des modèles de calcul multilinéaires [Raz09]), les bornes inférieures inconditionnelles n'ont pas été améliorées depuis longtemps. La meilleure borne inférieure sur la taille des circuits d'un polynôme de VNP est en $\Omega(n \log n)$, obtenue par Baur et Strassen [BS83]. Pour progresser dans la compréhension de ces problèmes, une approche consiste à travailler sur des modèles affaiblis. Un résultat récent très fort de ce type est une borne inférieure en $n^{\Omega(\log n)}$ sur la taille des formules multilinéaires calculant le déterminant [Raz09]. Une autre approche consiste à comparer les problèmes entre eux, ce qu'on peut voir comme des bornes inférieures conditionnelles. Les travaux de Bürgisser [Bür00, chapitre 4] sur les parties booléennes montrent que des séparations de classes booléennes ont pour conséquence $\mathsf{VP} \neq \mathsf{VNP}$ sur $\mathbb{C}$.

La ligne de recherche décrite ci-dessus vise à abtenir des bornes inférieures superpolynomiales (voire exponentielles). Pour de nombreux problèmes concrets, on connaît des algorithmes polynomiaux, sans toutefois que les bornes inférieure et supérieure ne coïncident. Par exemple, on sait que le calcul du diamètre d'un ensemble de points de $\mathbb{R}^d$ pour $d > 3$ a une complexité $\Omega(n \log n)$, à comparer aux meilleurs algorithmes qui sont proches d'être quadratiques. Dans ce cadre, les bornes inférieures sont obtenues le plus souvent par la borne de Ben-Or [BO83]. Comparer les problèmes entre eux est une autre approche : par exemple, il est conjecturé que le problème 3-Sum (étant donnés une liste de nombres réels, décider si trois d'entre eux ont une somme nulle)

a une complexité quadratique. Réduire 3-Sum à un autre problème constitue donc une borne inférieure conditionnelle sur ce dernier. Une telle liste de problèmes difficiles pour 3-Sum est dressée dans [GO95]. Une dernière approche consiste à explorer des modèles faibles comme c'est le cas dans les algorithmes à base de partitions de l'espace définis par Erickson [Eri96]. Enfin, il faut garder à l'esprit qu'améliorer les algorithmes peut être aussi la solution pour réduire l'écart entre bornes supérieure et inférieure.

Les recherches présentées dans les deux premiers chapitres de ce mémoire se situent dans les domaines évoqués ci-dessus ; une troisième partie porte sur les formules booléennes aléatoires, un domaine plus éloigné mais où des notions de complexité interviennent aussi.

La complexité algébrique est l'objet du premier chapitre. Le premier problème abordé traite de la question de la complexité de déterminer et de compter les monômes d'un polynôme représenté par un circuit [FMM14]. Ces travaux se situent dans la lignée de ceux de Wagner sur la complexité de problèmes combinatoires quand les entrées sont représentées de façon compacte [Wag86]. D'autres travaux concernent les bornes inférieures. Nous avons étudié la possibilité d'obtenir des bornes inférieures en $\Omega(n^k)$, pour tout k fixé, pour des polynômes de VNP sur $\mathbb{C}$, et obtenu des résultats conditionnels dans ce cadre [FPdV14]. Nous avons également étudié la taille des formules de profondeur 4, multilinéaires et homogènes, pour le produit itéré de matrices [FLMS14]. Ces bornes sont basées sur des extensions de la technique des dérivées partielles de Nisan et Wigderson [NW97] définies par Kayal [Kay12]. Le dernier problème exposé est de nature un peu différente : il s'agit de la construction de petites familles d'espaces vectoriels avec une propriété de transversalité (tout espace vectoriel de dimension appropriée possède un supplémentaire dans cette famille) [CFGK03, CFKP08]. Des constructions simples sont données dans le cas des corps infinis mais aucune construction déterministe ne semble connue dans le cas d'un corps fini, bien qu'un argument probabiliste très simple atteste de l'existence de telles familles.

Le deuxième chapitre se concentre sur des problèmes de géométrie algorithmique et d'optimisation. Un premier travail est consacré à l'étude du calcul du diamètre d'un polytope de $\mathbb{R}^3$ [FV07] : l'entrée est ici constituée des coordonnées des points et de la structure de leur enveloppe convexe. La question principale est de savoir si la donnée de l'enveloppe convexe permet un calcul plus rapide du diamètre comme cela se produit en dimension 2. Nous avons montré que ce n'est pas le cas en utilisant la borne de Ben-Or. La seconde partie de nos travaux concerne le calcul de fonctions en escalier approximant un ensemble de points du plan [FV11, FV13]. Nous avons obtenu des algorithmes optimaux linéaires ou en $O(n \log n)$ pour plusieurs problèmes de ce type, les meilleurs algorithmes connus précédemment étant quadratiques. Un dernier point concerne le calcul de l'hyperbolicité de Gromov d'une distance sur un ensemble fini [FIV13]. Nous avons relié ce problème au produit de matrices min-max, donnant à la fois un algorithme et une borne inférieure conditionnelle basée sur les techniques de Vassilevska-Williams et Williams [WW10].

Nous traitons d'une problématique bien différente dans une troisième partie. On considère divers modèles de formules booléennes aléatoires, et on cherche à décrire la distribution de probabilité induite sur les fonctions booléennes. Un travail pionnier dans ce domaine est celui de Lefmann et Savický [LS97] sur les fonctions booléennes calculées par des grandes formules aléatoires utilisant les connecteurs *et* et *ou*. En particulier, ce travail met en évidence des liens entre la probabilité d'obtenir une fonction et sa complexité (taille de la plus petite formule calculant cette fonction) ; cependant l'écart entre les bornes supérieure et inférieure est exponentiel. Motivés par des questions de comparaison entre logiques classique et intuitionniste [MTZ00], nous avons étudié la structure des tautologies (formules calculant la fonction *vrai*) construites sur le connecteur de l'implication [FGGZ07, FGGZ10]. Nous avons montré qu'elles ont presque toutes une structure très simple. Puis nous avons généralisé ce résultat au cas des formules calculant une

fonction booléenne quelconque (toujours sur le système de l'implication), mettant en évidence une relation très précise entre probabilité et complexité d'une fonction pour divers modèles de formules aléatoires [FGGG12]. Le dernier problème abordé est l'étude de la distribution de probabilité sur les fonctions booléennes induite par des formules sur les connecteurs *et* et *ou* qui ont une structure équilibrée : ceci signifie que tous les littéraux ont la même profondeur dans la formule. Ce travail peut être vu comme faisant suite aux travaux de Brodsky et Pippenger [BP05] qui ont mené une étude systématique sur les formules équilibrées construites sur un unique connecteur. Notons que c'est sur ce type de formules et en choisissant un connecteur adéquat que Valiant a obtenu des petites formules monotones pour la fonction majorité [Val84].

Complexity aims at quantifying computational resources which are necessary to solve algorithmic problems. In Boolean complexity, basic resources are computational time and space ; for these hierarchy theorems have been proved. Less natural resources have been defined such as non-deterministic time, with NP class capturing combinatorial optimization problems, or counting defined by Valiant [Val79a] and for which computing the permanent has completeness properties.

In algebraic complexity, one wants to know how many operations are needed to compute polynomials. The class VP corresponds to polynomials that can be computed with a polynomial number of operations. As in the Boolean setting, less natural resources have appeared such as the possibility to make exponential sums, leading to the definition of the class VNP [Val79b]. The class VNP contains many polynomials defined in combinatorics such as the permanent : thus it is not surprising that this class has tight connections with Boolean counting classes. In particular, counting classes can be defined by arithmetization of boolean complexity classes [All04] (this means that *and* and *or* gates of Boolean circuits are replaced with arithmetic operations product and sum).

Even if big progress has been made in algebraic complexity over the last years (derandomization vs. lower bounds [KI04] ; simulation of general arithmetic formulas by small depth formulas [AV08] ; lower bounds for multilinear models of computation [Raz09]), unconditional lower bounds have not been improved for a long time. The best lower bound on circuit size for polynomials from VNP is $\Omega(n \log n)$, obtained by Baur and Strassen [BS83]. In order to make progress, one approach consists in working on weaker computation models.

One recent strong result of this kind is the $n^{\Omega(\log n)}$ lower bound on the size of multilinear formulas computing the determinant [Raz09]. Another approach consists in comparing problems together ; these results can be seen as conditional lower bounds. For example, Bürgisser [Bür00, chapter 4] proved that separation of some Boolean classes yields $\mathsf{VP} \neq \mathsf{VNP}$ over $\mathbb{C}$.

The line of research described above aims at obtaining superpolynomial lower bounds (or even exponential ones). For many concrete problems, we know polynomial algorithms, but there is a gap between lower bounds and upper bounds. For example, computing the diameter of a point set in $\mathbb{R}^d$ when $d > 3$ has complexity lower bound $\Omega(n \log n)$, while the best known algorithm is only slightly better than quadratic. Lower bounds in this setting are usually obtained using Ben-Or bound [BO83]. Comparing problems together is another approach : for example, the problem 3-Sum (given a list of real numbers, decide if three of them sum to zero) is conjectured to have quadratic complexity. Thus, reducing 3-Sum to another problem can be seen as a conditional lower bound on the latter problem. Such problems which are hard for 3-Sum are listed in [GO95].

One last approach is to explore weakened models : for example Erickson considered some kind of algorithms based on space partitions [Eri96]. In addition, one may not forget that improving algorithms may sometimes be the way to reduce the gap between lower and upper bounds.

Research presented in the first two chapters of this document takes place in the two areas depicted above ; one last chapter is about random Boolean formulas, a different field where complexity notions arise too.

Algebraic complexity works are presented in the first chapter. The first problem dealt with is the complexity of deciding and counting the monomials of a polynomial represented by a circuit [FMM14]. This work is in the spirit of Wagner [Wag86] on the complexity of combinatorial problems when input is represented in a succinct form. Other works are on lower bounds. We have studied the possibility that $\Omega(n^k)$ lower bounds for all fixed k could be obtained for polynomials from the class VNP over $\mathbb{C}$, and obtained conditional results [FPdV14]. We have also studied the size of depth 4 multilinear homogeneous formulas for iterated matrix product [FLMS14]. These bounds are based on an extension of the technique of partial derivatives of Nisan and Wigderson [NW97] defined by Kayal [Kay12]. The last problem which is presented is of a different kind : it is about the construction of small families of linear subspaces possessing some transversality property (every linear subspace of appropriate dimension has a transveral subspace in this family) [CFGK03, CFKP08]. Simple constructions are given in the case of infinite fields ; on the other hand no deterministic construction is known on finite fields, although a simple probabilistic argument shows that small families do exist.

The second chapter is dedicated to algorithmic geometry and optimization problems. The first work is about computing the diameter of a polytope in $\mathbb{R}^3$ [FV07] : the input in this case is made of the coordinate of the points and the combinatorial structure of their convex hull. The main question we investigate is to determine whether this additional information can lead to faster algorithms as it happens in dimension 2. Using Ben-Or bound, we proved that this is not the case. The second problem presented focuses on approximating a point set in the plane by a step function [FV11, FV13]. We have obtained optimal linear or $O(n \log n)$-time algorithms for several problems of this kind, for which all previously known algorithms where quadratic. One last work concerns the computation of Gromov hyperbolicity of a metrics defined over a finite point set [FIV13]. We were able to relate this problem to min-max matrix product, by giving a simple algorithm but also obtaining conditional lower bounds based on techniques by Vassilevska-Williams and Williams [WW10].

The last part is dedicated to a different question. We consider different models of random Boolean formulas, and try to describe the probability they induce on Boolean functions. One pioneer work on this area is the one of Lefmann and Savický [LS97] on Boolean functions computed by large random formulas using connectors *and* and *or*. In particular, this work exhibits some connections between the probability to compute a function and its complexity (the size of the smallest formula computing this function) ; however there is an exponential gap between lower and upper bounds obtained here. Motivated by some questions about comparison of classical and intuitionistic logics [MTZ00], we have investigated the structure of tautologies (formulas computing the function *true*) build on the implication connector [FGGZ07, FGGZ10]. We have shown that they almost surely have a very simple structure. Then we have generalized this result to the case of formulas (build on the implication connector) computing a general Boolean function, showing a tight connection between the probability and the complexity of a function for different models of random formulas [FGGG12]. The last problem considered is the case of random formulas build on *and* and *or* connectors which are balanced : this means that all literals have the same depth in the formula. This work can be seen as a follow-up to Brodsky and Pippenger [BP05] who carried out a systematic study of balanced formulas built on a single

connector. Let's remark that this kind of balanced formula is the one Valiant used, choosing an appropriate connector, to build small monotone formulas for the majority function [Val84].

Chapitre 1

Complexité algébrique

La complexité algébrique a pour objet principal l'étude du nombre d'opérations algébriques, additions et produits, nécessaire pour calculer certains polynômes. Par extension, elle désigne aussi les questions de complexité liées aux représentations des polynômes par des circuits.

1.1 Formules et circuits arithmétiques

Un circuit arithmétique sur un corps $\mathbb{K}$ est un graphe orienté acyclique dont chaque feuille est étiquetée par une des variables $x_1, \ldots, x_n$ ou un scalaire (un élément de $\mathbb{K}$), et dont les nœuds internes sont étiquetés par les opérations $+$ et $\times$. Un tel circuit calcule un polynôme de $\mathbb{K}[x_1, \ldots, x_n]$ (Figure 1.1). La taille d'un circuit est son nombre de nœuds internes.

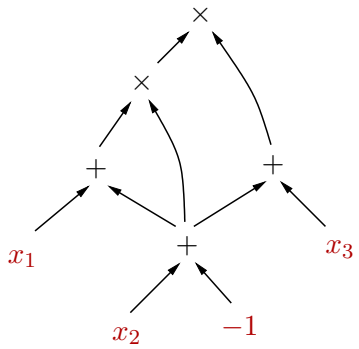

FIGURE 1.1 – Un exemple de circuit arithmétique.

Dans un circuit *sans constantes*, aucun scalaire autre que $0, 1, -1$ n'est autorisé dans les feuilles. En caractéristique nulle, un tel polynôme calcule un polynôme de $\mathbb{Z}[x_1, \ldots, x_n]$ (et en caractéristique non nulle p un polynôme de $\mathbb{F}_p[x_1, \ldots, x_n]$).

Le polynôme calculé par un circuit sans constantes de taille t a un degré majoré par 2^t et des coefficients majorés par 2^{2^t} en valeur absolue. Ces bornes sont du bon ordre de grandeur comme on le voit en calculant x^{2^t} ou $(1+1)^{2^t}$ par t élévations au carré successives. La représentation d'un polynôme par un circuit permet donc de manipuler des polynômes de grands degrés, avec de grands coefficients (même en l'absence de scalaires arbitraires).

Une formule arithmétique est un circuit arithmétique dont le graphe sous-jacent est acyclique. Ainsi, on ne peut utiliser le résultat d'un calcul plusieurs fois comme il est possible de le faire dans un circuit. Par exemple, pour calculer $(x-1)^2$, on doit calculer deux fois le polynôme

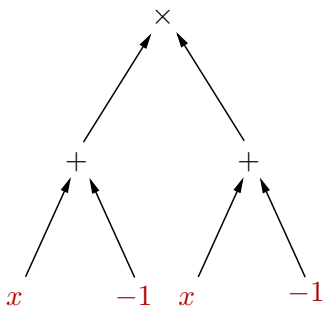

FIGURE 1.2 – Un exemple de formule arithmétique.

$x - 1$ (Figure 1.2), ou calculer le polynôme d'une façon radicalement différente.

Le degré d'un circuit est défini de manière inductive : le degré d'une porte d'entrée est 1, le degré d'une porte d'addition est le maximum des degrés de ses deux portes d'entrée, et le degré d'une porte produit est la somme des degrés de ses deux portes d'entrée. Le degré du circuit, défini comme le degré de sa porte de sortie, est donc une borne supérieure sur le degré du polynôme calculé par ce circuit.

En complexité algébrique, on s'intéresse souvent aux circuits dont le degré est polynomial en la taille (et non exponentiel comme dans l'exemple x^{2^n}). Cette restriction sémantique peut être obtenue par une restriction syntaxique sur les circuits comme l'ont montré Malod et Portier [MP08]. Un circuit est *multiplicativement disjoint* si en toute porte de multiplication, les deux sous-circuits entrants sont disjoints (Figure 1.3).

L'ensemble des suites de polynômes calculés par des circuits de tailles et degrés polynomiaux est noté VP (le corps $\mathbb{K}$ étant sous-entendu), et celui des suites de polynômes calculés par des formules de tailles polynomiales est notée $\mathsf{VP_e}$ (e pour expression). Une conjecture due à Valiant est l'inclusion stricte $\mathsf{VP_e} \subsetneq \mathsf{VP}$.

Nous définissons maintenant un modèle de calcul intermédiaire. Un *branching program* est un graphe orienté acyclique avec deux nœuds distincts s et t. Les arêtes sont étiquetées par des scalaires de $\mathbb{K}$ ou des variables $x_1, \ldots, x_n$. On définit le poids d'un chemin comme le produit des poids de ses arêtes. Le polynôme calculé par un *branching program* est la somme des poids de tous les chemins de s à t.

L'ensemble des suites de polynômes calculés par des *branching programs* de tailles polynomiales a une autre caractérisation en terme de circuits que nous présentons maintenant. Un circuit est *weakly skew* si en toute porte de multiplication, le sous-arbre correspondant à l'une des deux entrées est disjoint de tout le reste du circuit [Tod92, MP08]. Une suite de polynômes est calculée par une suite de *branching programs* de tailles polynomiales si et seulement si elle est calculée par une suite de circuits *weakly skew* de tailles polynomiales. Cette classe est notée $\mathsf{VP_{ws}}$. Parmi les problèmes complets pour cette classe on a le déterminant ou encore le produit itéré de matrices.

Les inclusions suivantes sont vérifiées sur tout corps $\mathbb{K}$:

$$\mathsf{VP_e} \subseteq \mathsf{VP_{ws}} \subseteq \mathsf{VP}.$$

Dans la section 1.4, nous verrons une autre classe définie par Valiant au-dessus de VP.

Un cas particulier de formules est celui des formules de profondeur constante. Dans ce cas, on autorise le fan-in des portes d'addition et de multiplication, c'est-à-dire le nombre

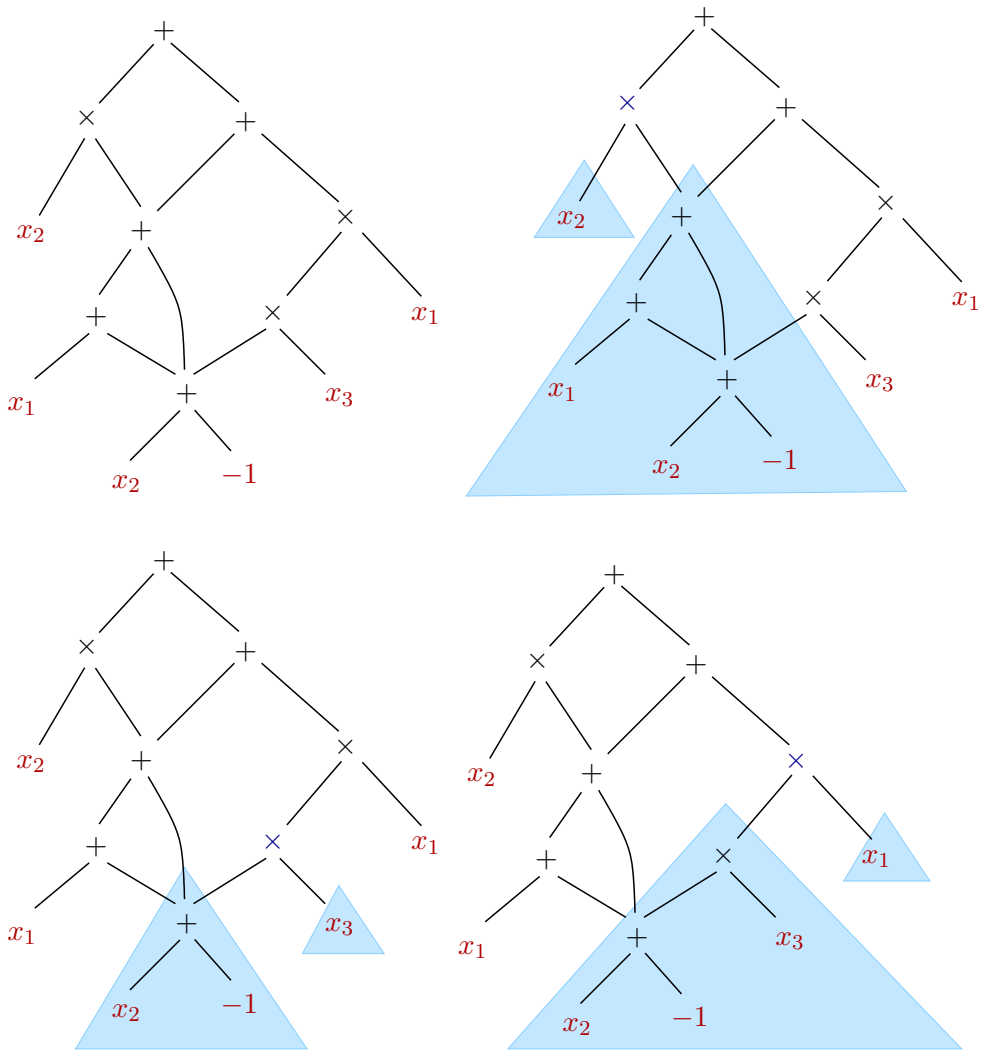

FIGURE 1.3 – Un exemple de circuit multiplicativement disjoint (en haut à gauche) ; par convention, toutes les arêtes sont orientées vers le haut. Le caractère disjoint de chacune des portes de multiplication est mis en évidence dans les trois autres représentations du circuit.

d'arguments de ces portes, à être non borné (si ce n'était pas le cas, une telle formule ne pourrait pas dépendre d'un nombre non constant d'entrées). Nous rencontrerons ce type de formules dans la section 1.5 où nous expliquerons l'importance des formules arithmétiques de profondeur 4.

L'étude des circuits arithmétiques fait apparaître naturellement les classes (booléennes) de comptage, que nous définissons ci-dessous.

1.2 Les classes de comptage

On rappelle que P désigne l'ensemble des langages décidables en temps polynomial. Un langage $L \subseteq \{0,1\}^\star$ est dans NP s'il existe un polynôme $p(n)$ et un langage $A \in \mathsf{P}$ tel que pour tout $x \in \{0,1\}^\star$:

$$x \in L \iff \exists y \in \{0,1\}^{p(|x|)} \ (x,y) \in A.$$

Les classes de comptage s'intéressent au nombre d'éléments y en relation avec x. Précisément, une fonction $f : \{0,1\}^\star \to \mathbb{N}$ est dans $\#\mathsf{P}$ s'il existe un polynôme $p(n)$ et un langage $A \in \mathsf{P}$ tels que pour tout $x \in \{0,1\}^\star$:

$$f(x) = |\{y \in \{0,1\}^{p(|x|)},\ (x,y) \in A\}|.$$

Exemple 1 *La fonction qui à une formule de la logique propositionnelle associe le nombre d'assignements la satisfaisant est dans* $\#\mathsf{P}$*.*

Une fonction $g : \{0,1\}^\star \to \mathbb{Z}$ est dans GapP s'il existe deux fonctions $f, f' \in \#\mathsf{P}$ telle que $g = f - f'$. La classe $\mathsf{C}_=\mathsf{P}$ est l'ensemble des langages $A = \{x,\ g(x) = 0\}$ pour $g \in \mathsf{GapP}$.

Exemple 2 *L'ensemble des couples de formules propositionnelles* (F, G) *tel que* F *et* G *ont le même nombre d'assignements satisfaisables est dans la classe* $\mathsf{C}_=\mathsf{P}$ *(il est même complet pour cette classe).*

On rappelle que le permanent d'une matrice M de taille $n \times n$ est défini par

$$\mathrm{Per}(M) = \sum_\sigma \prod_{i=1}^{n} M_{i,\sigma(i)}$$

où σ parcourt l'ensemble des permutations de $\{1, \ldots, n\}$. Un autre exemple de problème complet pour la classe $\mathsf{C}_=\mathsf{P}$ est obtenu à partir du permanent.

> PER$_=$
> *Entrée :* Matrice $A \in \{0, 1, -1\}^n$, $d \in \mathbb{N}$
> *Problème :* Décider si $\textsc{per}(A) = d$.

Il existe aussi des classes de comptage "modulo p". Par exemple, la classe $\oplus\mathsf{P}$ est l'ensemble des langages $A = \{x,\ f(x) \text{ est impair}\}$ pour une fonction $f \in \#\mathsf{P}$.

La classe PP et la hiérarchie de comptage

La classe PP est définie de la manière suivante : $L \in \mathsf{PP}$ s'il existe $A \in \mathsf{P}$, $f \in \mathsf{FP}$ et un polynôme p tel que

$$x \in L \Leftrightarrow \left|\left\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in A\right\}\right| \geqslant f(x).$$

On peut choisir une fonction f très simple : $L \in \mathsf{PP}$ s'il existe $A \in \mathsf{P}$ et un polynôme p tel que

$$x \in L \Leftrightarrow \left|\left\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in A\right\}\right| > 2^{p(|x|)-1}$$

Autrement dit : $x \in L$ si et seulement si une majorité de y vérifient $A(x,y)$.

Exemple 3 *Le langage* MajSat *est l'ensemble des formules propositionnelles dont strictement plus de la moitié des assignements des variables satisfont la formule. On a* $x \vee y \in$ MajSat *car la proportion des assignements satisfaisables est* $3/4 > 1/2$ *et* $(x \vee y) \wedge z \notin$ MajSat *car la proportion des assignements satisfaisables est* $3/8 \leqslant 1/2$*.*

On rappelle que la hiérarchie polynomiale est définie de la manière suivante : on pose $\Sigma_0 = \mathsf{P}$ et $\Sigma_{k+1} = \mathsf{NP}^{\Sigma_k}$, c'est-à-dire que les langages de Σ_{k+1} sont ceux qui peuvent être décidés en temps polynomial par une machine de Turing non-déterministe utilisant comme oracle un langage du niveau inférieur Σ_k. La hiérarchie polynomiale est alors définie par $\mathsf{PH} = \bigcup_{k\in\mathbb{N}} \Sigma_k$. Une formulation équivalente sans oracles est la suivante : un langage L est dans Σ_k s'il existe un polynôme $p(n)$ et un langage $A \in \mathsf{P}$ tel que pour tout $x \in \{0,1\}^\star$: $x \in L$ si et seulement si

$$\exists y_1 \in \{0,1\}^{p(|x|)}\ \forall y_2 \in \{0,1\}^{p(|x|)}\ \exists y_3 \in \{0,1\}^{p(|x|)}\ \ldots Q y_k \in \{0,1\}^{p(|x|)}\ (x, y_1, \ldots, y_k) \in A,$$

où Q est le quantificateur $\exists$ si k impair et $\forall$ sinon.

La classe de comptage PP a une grande puissance de calcul comme le montre le théorème suivant de Toda : $\mathsf{PH} \subset \mathsf{P}^{\mathsf{PP}}$.

La hiérarchie de comptage est définie par

$$\mathsf{CH} = \mathsf{PP} \cup \mathsf{PP}^{\mathsf{PP}} \cup \mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}} \cup \ldots.$$

Cette hiérarchie a été introduite par Wagner [Wag86] pour classifier la complexité de problèmes combinatoires représentés de manière succincte. L'étude de cette classe a été poursuivie par Torán [Tor88, Tor91].

Que sait-on faire dans la hiérarchie de comptage ?

— PP caractérise la complexité du comptage des solutions des problèmes NP ;
— $\mathsf{NP}^{\mathsf{PP}}$: complexité de nombreux problèmes d'intelligence artificielle et de planning ;
— CH contient tous les problèmes booléens qu'on peut résoudre en faisant des calculs sur $\mathbb{Q}$ avec les opérations $+,-,\times,/,<$ [ABKPM09]. Par exemple :

> Somme de racines carrées
> *Entrée :* $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{N}$
> *Problème :* Décider si $\sum_i \sqrt{a_i} < \sum_i \sqrt{b_i}$.

Dans cet exemple, les racines carrées ne peuvent être calculées exactement mais peuvent être approximées suffisamment pour décider le problème par un nombre polynomial d'opérations arithmétiques [Tiw92].

La hiérarchie de comptage est apparue dans plusieurs autres articles récents. Par exemple, Bürgisser [Bür09] utilise cette classe pour relier les complexités de calculs d'entiers et de polynômes, et Jansen et Santhanam [JS11] — en se basant sur des résultats de Koiran et Perifel [KP11] — l'utilisent pour obtenir des bornes inférieures pour les circuits arithmétiques de profondeur constante.

1.3 Sur les monômes d'un polynôme représenté par un circuit

Bien que la représentation d'un polynôme par un circuit soit compacte comme nous l'avons remarqué, cela n'empêche pas qu'il existe des algorithmes efficaces sur les polynômes représentés par des circuits.

Le déterminant d'une matrice générique peut être calculé par un circuit (et donc le déterminant d'une matrice dont les entrées sont des polynômes donnés par des circuits également). Ceci ne découle pas directement de l'algorithme de Gauss puisque les circuits que nous considérons n'ont pas de divisions (voir par exemple Mahajan et Vinay [MV99]). Toutes les dérivées partielles (c'est-à-dire par rapport à chacune des variables) peuvent être calculées simultanément par

un circuit de taille linéaire comme l'ont montré Baur et Strassen [BS83]. Si on connaît une borne polynomiale sur le nombre de monômes du polynôme calculé par un circuit, Garg et Schost [GS09] ont montré qu'on peut dresser la liste de tous les monômes (avec leurs coefficients) en temps polynomial. Ce résultat s'appuie sur une série de travaux dont nous mentionnons certaines étapes clé ci-dessous. Un premier résultat important a été obtenu par Grigoriev et Karpinski [GK87] pour les polynômes de type permanent en caractéristique nulle. Ceci a été généralisé au cas des polynômes arbitraires en caractéristique nulle par Ben-Or et Tiwari [BT88]. Ces algorithmes permettent de reconstituer le polynôme dans le modèle *black-box* avec un nombre polynomial (en le nombre de variables et le logarithme du degré) d'opérations arithmétiques. Ce dernier résultat est ensuite généralisé à la caractéristique non nulle [GKS90]. Enfin, certains travaux s'attachent à interpoler des fonctions plus générales que les polynômes ; le cas des fonctions rationnelles creuses est traité dans [GKS94].

Définissons maintenant un autre problème important sur les circuits :

> Arithmetic Circuit Identity Testing (ACIT)
> *Entrée :* Un circuit C
> *Problème :* Le polynôme calculé par ce circuit est-il nul ?

On sait que ACIT est dans coRP [Sch79]. La question principale concernant ce problème est l'existence d'un algorithme déterministe en temps polynomial puisque ceci impliquerait des bornes inférieures [KI04].

On s'intéresse à deux problèmes : le problème de savoir si le coefficient d'un monôme d'un polynôme représenté par un circuit est nul, et celui de compter les monômes (qui ont un coefficient non nul). En particulier, nous cherchons à déterminer la complexité de ces problèmes pour divers types de circuits. Dans cette section, les résultats de complétude sont énoncés pour les réductions many-one en espace logarithmique si rien n'est précisé.

Le problème du monôme nul

Le problème du monôme nul, appelé ZMC pour *Zero Monomial Coefficient*, est le suivant.

> ZMC
> *Entrée :* Un circuit C et un monôme m
> *Problème :* Le monôme m a-t-il un coefficient nul dans le polynôme calculé par C ?

Ce problème a été étudié par Koiran et Perifel [KP07]. Ils montrent que quand le degré du circuit est polynomialement borné, ZMC est complet pour $\mathsf{P}^{\#\mathsf{P}}$, pour les réductions appelées "strong non-deterministic Turing reductions". Ces réductions sont peu utilisées et leur définition est obscure. Nous avons obtenu la caractérisation suivante, plus satisfaisante à nos yeux.

Théorème 1 *[FMM14]* ZMC *est* $\mathsf{C}_{=}\mathsf{P}$*-complet à la fois pour les circuits multiplicativement disjoints et les formules.*

Le résultat ci-dessus fournit un nouveau problème complet pour la classe $\mathsf{C}_{=}\mathsf{P}$ qui en possède peu de naturels. [1] Notons que par contraste, la classe $\mathsf{C}_{=}\mathsf{L}$, définie de manière analogue à $\mathsf{C}_{=}\mathsf{P}$ mais pour le comptage en espace logarithmique, possède de nombreux problèmes naturels complets issus de l'algèbre linéaire [ABO99].

Notons que ce résultat de difficulté a trouvé une application dans un article récent de Mittmann, Saxena et Scheiblechner [MSS12] : il y est démontré que la notion de dégénérescence

1. Il est même écrit dans le livre d'Hemaspaandra et Ogihara que la classe $\mathsf{C}_{=}\mathsf{P}$ ne possède aucun problème naturel complet [HO02, p. 293] : ceci est discutable puisque le problème $\textsc{per}_{=}$ à partir duquel nous avons fait la réduction est assez naturel.

considérée dans cet article, à laquelle l'indépendance algébrique en caractéristique non nulle peut être réduite, est difficile par une réduction depuis ZMC.

Un circuit est appelé *monotone* si les seules constantes autorisées sont 0 et 1 (mais pas -1 comme dans le cas général). Regardons maintenant le problème ZMC sur les circuits monotones Dans le cas d'une seule variable, cela est équivalent à décider si la sortie d'un circuit sur des entiers naturels avec des portes $\cup$ et $+$ omet un entier donné. Cela est coNP-complet à la fois pour les formules et les circuits d'après McKenzie et Wagner. Il est facile de généraliser ce résultat aux polynômes en plusieurs variables.

Théorème 2 (McKenzie et Wagner [MW07]) ZMC *est* coNP*-complet à la fois pour les formules monotones et les circuits monotones.*

Koiran et Perifel ont aussi considéré le cas général de ZMC (c'est-à-dire le cas où le degré du circuit donné en entrée n'est pas polynomialement borné). Ils ont montré que ZMC $\in$ CH. Nous améliorons cette borne supérieure.

Théorème 3 *[FMM14] ZMC est dans* $\mathsf{coRP}^{\mathsf{PP}}$.

L'une des questions les plus intrigantes concerne la complexité du degré défini dans [ABKPM09].

> DegSLP
> *Entrée :* Circuit arithmétique C, $d \in \mathbb{N}$
> *Problème :* Décider si le degré du polynôme calculé par C est strictement inférieur à d.

Kayal et Saha [KS11] ont montré que ce problème est dans $\mathsf{coRP}^{\mathsf{PP}}$. Pour l'instant, on ne connaît pas de meilleure borne inférieure que P. Nous définissons ci-dessous une généralisation du problème DegSLP pour lequel nous arrivons à obtenir une borne inférieure par réduction depuis ZMC.

> GapMonSLP
> *Entrée :* Circuit arithmétique C sur une variable X, $a, b \in \mathbb{N}$
> *Problème :* Décider si le polynôme calculé par C ne contient aucun monôme de la forme X^c pour $a \leqslant c \leqslant b$.

Proposition 1 *[FMM14]* GapMonSLP *est équivalent à* ZMC.

Le problème du comptage des monômes

On s'intéresse maintenant à la complexité de compter le nombre de monômes d'un polynôme donné par un circuit. Notons que ce problème est facile si on a une borne polynomiale sur le nombre de monômes, en vertu du résultat d'interpolation de Garg et Schost [GS09] déjà mentionné. Le problème qui consiste à énumérer les monômes d'un polynôme donné, dans le modèle black-box, a été étudié par Strozecki [Str13]. Formellement, le problème est le suivant :

> CountMon
> *Entrée :* un circuit C et $k \in \mathbb{N}$
> *Problème :* Le polynôme calculé par C a-t-il plus de k monômes ?

Notons que le problème ci-dessus n'est pas un problème de comptage à proprement parler, c'est le problème de décision associé. La raison de ce choix est que les résultats obtenus s'expriment plus simplement de cette manière.

Théorème 4 *[FMM14]*

— CountMon *est* $\mathsf{PP}^{\mathsf{PP}}$*-complet. Ce problème est* $\mathsf{PP}^{\mathsf{PP}}$*-difficile même pour les formules de profondeur 4.*

— CountMon *est* $\mathsf{PP}^{\mathsf{NP}}$*-complet à la fois pour les formules monotones et pour les circuits monotones.*

Il existe très peu de problèmes naturels complets pour les niveaux supérieurs de la hiérarchie de comptage. Par exemple, Kwisthout et al. [KBvdG11] donne "le premier problème avec une application pratique démontré comme étant $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complet" (ce problème consiste, étant donné un réseau probabiliste et un entier k, à calculer la k-ième explication la plus probable).

Le cas multilinéaire

Nous étudions maintenant les deux problèmes définis ci-dessus dans le cas particulier de circuits calculant un polynôme multilinéaire. Vérifier qu'un circuit donné en entrée est multilinéaire a une complexité équivalente à celle de ACIT. Nous avons décidé d'inclure ce test de multilinéarité dans la définition de nos problèmes (cela n'a pas d'influence). La définition formelle de nos deux problèmes ainsi que leur complexité sont données ci-dessous.

> ML-ZMC
> *Entrée :* Circuit arithmétique C, monôme m
> *Problème :* Décider si C calcule un polynôme multilinéaire dans lequel le monôme m a un coefficient nul.

> ML-CountMon
> *Entrée :* Circuit arithmétique C, $d \in \mathbb{N}$
> *Problème :* Décider si le polynôme calculé par C est multilinéaire et a au moins d monômes.

Proposition 2 *[FMM14]* ML-ZMC *est équivalent à* ACIT. ML-CountMon *est* PP*-complet (pour les réductions Turing).*

Le cas des circuits en une seule variable

Le problème de tester si un monôme est nul dans le cas de polynômes univariés est relié au problème DegSLP ainsi qu'au problème suivant, également défini par Allender et al. [ABKPM09] : il s'agit de EquSLP, l'équivalent du problème ACIT pour les circuits sans variables.

> EquSLP
> *Entrée :* Circuit arithmétique C calculant un entier
> *Problème :* Décider si l'entier calculé est nul.

La première observation que nous pouvons faire est la suivante.

Proposition 3 *[FMM14] Pour les circuits en une variable multiplicativement disjoints, les problèmes* DegSLP*,* ZMC*,* EquSLP *et* ACIT *sont équivalents (pour les réductions en espace logarithmique). Ces quatre problèmes sont équivalents également dans le cas particulier des circuits monotones.*

Pour caractériser la complexité de ces problèmes dans le cas monotone, nous devons rappeler ce qu'est LOGCFL. Son nom vient du fait que c'est la classe des langages qui peuvent être réduits en espace logarithmique à un langage algébrique. Une autre caractérisation, peut-être plus naturelle et que nous allons utiliser ci-dessous pour définir $\mathsf{C_{=}LOGCFL}$, est que LOGCFL est l'ensemble des langages décidés par des circuits logspace-uniformes semi-bornés AC [Ven91]. De manière équivalente, ce sont aussi les langages décidés par des circuits logspace-uniformes dont chaque porte de conjonction voit ses entrées disjointes (l'équivalent booléen des circuits multiplicativement disjoints).

Proposition 4 *[FMM14] Pour les circuits en une variable multiplicativement disjoints monotones, les problèmes* DegSLP*,* ZMC*,* EquSLP*,* ACIT *et* CountMon *sont* LOGCFL*-complets.*

Il est maintenant nécessaire de rappeler ce qu'est la classe $\mathsf{C_{=}LOGCFL}$. La façon la plus simple de définir la classe $\mathsf{C_{=}LOGCFL}$ est de considérer la version arithmétisée de la classe LOGCFL définie par des circuits. De même que GapP peut être définie comme la différence de deux fonctions #P, on peut définir $\mathsf{GapLOGCFL} = \mathsf{GapSAC^1}$ [All04] comme la différence de deux circuits arithmétiques semi-bornés de profondeur logarithmique sur $\mathbb{N}$, ou simplement en considérant un tel circuit sur $\mathbb{Z}$. De manière équivalente, on peut considérer la différence de deux circuits multiplicativement disjoints sur $\mathbb{N}$ ou un seul circuit multiplicativement disjoint sur $\mathbb{Z}$. Un langage L appartient à $\mathsf{C_{=}LOGCFL}$ s'il existe une fonction $f \in \mathsf{GapLOGCFL}$ telle que $L = \{x \mid f(x) = 0\}$.

Proposition 5 *[FMM14] Pour les circuits en une variable multiplicativement disjoints :*
- DegSLP, ZMC, EquSLP, ACIT *sont* $\mathsf{C_{=}LOGCFL}$*-complets ;*
- CountMon *est* $\mathsf{C_{=}LOGCFL}$*-difficile et est dans* $\mathsf{L^{C_{=}LOGCFL}}$.

À la suite de la publication d'une version préliminaire des résultats présentés dans cette section, des questions similaires ont été étudiées [MRS12] pour des modèles faibles de circuits, des formules ou *branching programs* qui ne peuvent lire chaque variable qu'une ou deux fois (*read-once* ou *read-twice*). Pour ces classes de circuits, la complexité de ces problèmes s'effondre souvent, mais pas toujours.

Perspectives

Les bornes inférieures et supérieures pour le problème CountMon sont très proches mais ne coïncident pas. La complexité de ce problème reste donc à préciser. La borne supérieure en $\mathsf{coRP^{PP}}$ peut-elle être dérandomisée ?

Les techniques développées ici ont trouvé une application dans [MSS12] : il y est démontré que la notion de dégénérescence considérée dans cet article, à laquelle l'indépendance algébrique en caractéristique non nulle peut être réduite, est difficile par une réduction à partir du problème ZMC. Il serait intéressant de déterminer si un tel résultat de difficulté peut être obtenu pour l'indépendance algébrique elle-même.

1.4 Bornes inférieures pour des polynômes uniformes au sens de Valiant

Baur and Strassen [BS83] ont démontré en 1983 que le nombre d'opérations arithmétiques pour calculer le polynôme

$$x_1^n + \ldots + x_n^n$$

est $\Omega(n \log n)$. Ceci est toujours la meilleure borne inférieure connue pour les polynômes uniformes à n variables et de degré $n^{O(1)}$, si la notion d'uniformité considérée est d'avoir des circuits calculés en temps polynomial.

Définition 1 *Soit $s : \mathbb{N} \to \mathbb{N}$. Un langage booléen L est dans $\mathsf{size}(s(n))$ s'il existe une suite de circuits (C_n) tel que C_n décide $L \cap \{0,1\}^n$ et la taille de C_n est $O(s(n))$.*

De même, pour un corps $\mathbb{K}$, on dit que (P_n) est dans la classe $\mathsf{asize}_{\mathbb{K}}(s(n))$ s'il existe une suite de circuits arithmétiques de taille $O(s(n))$ calculant cette suite de polynômes (les feuilles de ces circuits pouvant être étiquetées par des scalaires arbitraires du corps $\mathbb{K}$).

Si aucune condition d'uniformité n'est requise, on connaît des bornes inférieures depuis Lipton [Lip75]. En se basant sur ces travaux et ceux de Strassen [Str74], Schnorr a montré le résultat suivant.

Théorème 5 (Schnorr [Sch78]) *Pour tout k, il existe une famille de polynômes en une variable (P_n) de degré polynomial en n et telle que $(P_n) \notin \mathsf{asize}_{\mathbb{C}}(n^k)$.*

Le point de départ de la technique de Schnorr est de remarquer que le vecteur des coefficients d'un polynôme calculé par un circuit utilisant les constantes complexes $\alpha = (\alpha_1, \ldots, \alpha_p)$ est donné par une application polynomiale en α. Ainsi, calculer un polynôme sans circuit de taille t revient à trouver un point hors de l'image d'une telle application pour un circuit universel pour la taille t. Cette méthode a été étudiée et étendue par Raz [Raz10].

Différentes notions d'uniformité peuvent être considérées, soit en terme de circuits calculant ces polynômes, ou sur la complexité du calcul des coefficients. Par exemple, en étudiant attentivement la preuve de Schnorr, on peut voir que les coefficients des polynômes obtenus peuvent être calculés en temps exponentiel. On s'intéresse dans la suite à des polynômes plus uniformes que cela.

La plupart des résultats de cette section supposent que l'hypothèse de Riemann généralisée (Generalized Riemann Hypothesis, ou GRH) est vraie. Cette hypothèse est nécessaire pour permettre d'exprimer, efficacement et au niveau booléen, des propriétés sur des systèmes polynomiaux sur le corps des nombres complexes. Cette technique est due à Koiran [Koi96]. La raison pour laquelle GRH est utile est qu'elle a permet d'obtenir des résultats précis sur la distribution des nombres premiers.

Bornes inférieures en un polynôme fixé pour les classes booléennes

On s'intéresse dans cette section à des bornes inférieures *en un polynôme fixé* sur la taille des circuits. Autrement dit, pour une classe de complexité $\mathcal{C}$, nous voulons démontrer que :

$$\text{pour tout } k \in \mathbb{N},\ \mathcal{C} \not\subset \mathsf{size}(n^k).$$

Bien sûr, cela n'implique pas $\mathcal{C} \not\subset \bigcup_{k\in\mathbb{N}} \mathsf{size}(n^k) = \mathsf{size}(\mathsf{poly})$. La raison principale de s'intéresser à ce genre de résultats est que c'est un objectif plus modeste.

Les bornes inférieures en n^k sur la taille des circuits ont été étudiées de manière intensive sur les classes booléenne. Le premier résultat de ce type est dû à Kannan.

Théorème 6 (Kannan [Kan82]) *Pour tout $k \in \mathbb{N}$, $\Sigma_2 \not\subset \mathsf{size}(n^k)$.*

Le principe de la démonstration du résultat de Kannan est le suivant :

(i) Il existe $L_n \subset \{0,1\}^n$, $|L_n| = n^{3k}$, tel que L_n n'est pas décidé par un circuit de taille n^{k+1} (par dénombrement) ;

(ii) On peut définir l'appartenance au langage $\bigcup_n L_n$ (pour des langages L_n comme ci-dessus) dans la classe Σ_4 ;

(iii) Ceci montre que $\forall k, \Sigma_4 \not\subset \mathsf{size}(n^k)$;

(iv) Si SAT $\not\subset \mathsf{size}(\mathsf{poly})$ alors $\forall k, \mathrm{SAT} \not\subset \mathsf{size}(n^k)$; sinon $\mathsf{PH} = \Sigma_2$ d'après le théorème de Karp-Lipton et donc $\Sigma_2 = \Sigma_4 \not\subset \mathsf{size}(n^k)$.

Nous allons avoir besoin des classes d'Arthur-Merlin. Un langage L est dans MA s'il existe un polynôme $p(n)$ et $A \in \mathsf{P}$ tel que pour tout x :

$$\begin{cases} x \in L \Rightarrow \exists y \in \{0,1\}^{p(|x|)} \ \Pr_{r \in \{0,1\}^{p(|x|)}}[(x,y,r) \in A] \geqslant 2/3; \\ x \notin L \Rightarrow \forall y \in \{0,1\}^{p(|x|)} \ \Pr_{r \in \{0,1\}^{p(|x|)}}[(x,y,r) \in A] \leqslant 1/3. \end{cases}$$

Un langage L appartient AM s'il existe un polynôme $p(n)$ et $A \in \mathsf{P}$ tel que pour tout x :

$$\begin{cases} x \in L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}}[\exists y \in \{0,1\}^{p(|x|)} \ (x,y,r) \in A] \geqslant 2/3; \\ x \notin L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}}[\exists y \in \{0,1\}^{p(|x|)} \ (x,y,r) \in A] \leqslant 1/3. \end{cases}$$

On rappelle les inclusions $\mathsf{MA} \subseteq \mathsf{AM} \subseteq \mathsf{PH}$.

Pour obtenir des bornes inférieures sur les polynômes, nous suivrons la même stratégie que Kannan, mais le fait que des constantes arbitraires de $\mathbb{C}$ soient autorisées ne permettent plus d'adapter directement les méthodes booléennes. Une variante de Karp-Lipton qu'on utilise pour l'étape (iv) est le théorème suivant de Lund et al. [LFKN90] : si $\mathsf{PP} \subset \mathsf{P/poly}$ alors $\mathsf{CH} = \mathsf{MA}$.

On rappelle ci-dessous les classes booléennes qui possèdent des bornes inférieures en n^k. La classe SPP est définie comme l'ensemble des langages dont la fonction caractéristique est dans GapP : autrement dit, $L \in \mathsf{SPP}$ si et seulement si $\chi_L \in \mathsf{GapP}$.

Bornes inférieures en un polynôme fixé pour les classes booléennes

Classes $\mathcal{C}$ dont on sait qu'elles vérifient la propriété $\forall k, \mathcal{C} \not\subset \mathsf{size}(n^k)$:
— Σ_2 (Kannan [Kan82])
— $\mathsf{MA/1}$ (Santhanam [San09])
— PP (Vinodchandran [Vin05])

Classes dont on ne sait pas si elles possèdent cette propriété :
— MA (et donc NP)
— SPP
— $\oplus\mathsf{P}$

Un article récent de Fortnow et al. [FSW09] précise la frontière entre les classes booléennes qui ont des bornes inférieures en n^k sur la taille des circuits et celles pour lesquelles ce n'est pas connu.

Bornes inférieures sur des polynômes avec des coefficients faciles à calculer, ou faciles à évaluer

On s'intéresse à l'existence de polynômes difficiles (c'est-à-dire sans petits circuits sur $\mathbb{C}$) mais avec des coefficients qui sont "faciles à calculer". La construction de familles uniformes de polynômes sans circuits de taille n^k a été obtenue récemment par Jansen et Santhanam [JS12]. Ils montrent en particulier qu'il existe des polynômes à coefficients dans MA mais qui n'ont pas de circuits arithmétiques de taille n^k sur $\mathbb{Z}$. En supposant vraie GRH, nous étendons leur résultat au cas de polynômes sur le corps des nombres complexes.

Théorème 7 *[FPdV14] On suppose GRH vraie. Pour tout $k \in \mathbb{N}$, il existe une famille (P_n) de polynômes en une variable à coefficients dans $\{0,1\}$ tels que :*

— $\deg(P_n) = n^{O(1)}$ *(degré polynomial) ;*

— *les coefficients de P_n sont calculables dans* PH*. Cela signifie que le langage*

$$\{(1^n, i) \mid \textit{le coefficient de } x^i \textit{ dans } P_n \textit{ est } 1\}$$

est dans PH *;*

— *(P_n) n'a pas de circuits arithmétiques sur $\mathbb{C}$ de taille n^k.*

Si on considère des polynômes à n variables au lieu d'une seule, on obtient des bornes inférieures pour des polynômes à coefficients dans MA.

Corollaire 1 *[FPdV14] On suppose GRH vraie. Pour tout $k \in \mathbb{N}$, il existe une suite de polynômes (P_n) à n variables, à coefficients dans $\{0,1\}$, de degré $n^{O(1)}$, tel que les coefficients sont calculables dans* MA *et $(P_n) \notin \mathsf{asize}_{\mathbb{C}}(n^k)$.*

En fait, la famille de polynômes construite par Jansen et Santhanam est aussi uniforme au sens où ces polynômes peuvent être évalués aux points entiers dans MA. Définissons précisément ce que signifie cette notion.

Définition 2 *Une suite de polynômes $(P_n(x_1, \ldots, x_n))$ peut être évaluée dans la classe de complexité $\mathcal{C}$ si le langage*

$$\{(x_1, \ldots, x_n, i, b) \mid \textit{le } i\textit{-ème bit de } P_n(x_1, \ldots, x_n) \textit{ est } b\}$$

est dans $\mathcal{C}$, où $x_1, \ldots, x_n, i$ sont des entiers donnés en binaire et $b \in \{0,1\}$.

Deux obstacles ne permettent pas d'étendre de façon simple ce résultat aux polynômes complexes. D'une part les constantes complexes interdisent d'adapter directement la méthode de Jansen et Santhanam. D'autre part nous avons besoin du protocole AM de Koiran [Koi96] afin de décider si un système polynomial admet une solution complexe. Le résultat ci-dessous utilise la méthode développée dans Santhanam [San09].

Proposition 6 *[FPdV14] On suppose GRH vraie. Pour tout k, il existe une suite de polynômes (P_n) à n variables, à coefficients dans $\{0,1\}$, de degré $n^{O(1)}$, qui peuvent être évalués dans* AM *et tels que $(P_n) \notin \mathsf{asize}_{\mathbb{C}}(n^k)$.*

Bornes inférieures sur les classes de Valiant

Une notion pertinente d'uniformité est celle issue de la définition de la classe VNP de Valiant, qui capture la complexité du permanent. Le problème principal qu'on étudie ici est celui d'obtenir des bornes inférieures en $\Omega(n^k)$ sur la complexité de polynômes à n variables de VNP. Ceci n'est pas tellement différent de l'approche de la section précédente, qui consistait à obtenir des bornes inférieures sur des polynômes ayant des coefficients faciles à calculer : en effet, en vertu du critère de Valiant, un polynôme est dans la version uniforme de VNP si ses coefficients sont dans GapP.

Bien que MA puisse sembler être une petite classe comparée à GapP (en particulier compte tenu du théorème de Toda $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$), les résultats obtenus à la section précédente ne permettent pas l'obtention de bornes inférieures dans le cadre de VNP. Rappelons les définitions usuelles des classes de Valiant.

Définition 3 (Classes de Valiant) *Soit $\mathbb{K}$ un corps. Une suite (P_n) de polynômes sur $\mathbb{K}$ est dans la classe $\mathsf{VP}_{\mathbb{K}}$ si le degré de P_n est polynomial en n et si (P_n) est calculé par une suite de circuits arithmétiques sur $\mathbb{K}$* de taille polynomiale.

Une suite $(Q_n(x))$ de polynômes sur $\mathbb{K}$ est dans la classe $\mathsf{VNP}_{\mathbb{K}}$ s'il existe une suite $(P_n(x,y)) \in \mathsf{VP}_{\mathbb{K}}$ telle que

$$Q_n(x) = \sum_{y \in \{0,1\}^{\ell_n}} P_n(x,y)$$

où ℓ_n est la longueur de y dans P_n.

La taille de (x,y) est limitée par la taille des circuits pour P_n et est donc polynomiale. Notons que la seule différence entre VP_K et $\mathsf{asize}_{\mathbb{K}}(\mathsf{poly})$ est la contrainte sur le degré de P_n. Si le corps sous-jacent $\mathbb{K}$ est clair d'après le contexte on pourra enlever "$\mathbb{K}$" et écrire VP et VNP.

Les polynômes de VNP

Les réductions usuelles sur les classes de Valiant sont les *projections*. Sur un corps $\mathbb{K}$, un polynôme $P(x_1,\ldots,x_n)$ est une projection de $Q(y_1,\ldots,y_m)$ si $P(x_1,\ldots,x_n) = Q(a_1,\ldots,a_m)$ pour $a_1,\ldots,a_m \in \{x_1,\ldots,x_n\} \cup \mathbb{K}$. Une suite (P_n) se réduit à (Q_n) (via les projections) si P_n est une projection de $Q_{q(n)}$ pour une application polynomialement bornée q.

L'exemple le plus classique de famille complète pour la classe VNP est le permanent :

$$\mathrm{PER}_n(x_{1,1},\ldots,x_{n,n}) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

La somme ci-dessus est effectuée sur toutes les permutations σ de $\{1,\ldots,n\}$. Cependant, cette suite de polynômes (PER_n) n'est complète qu'en caractéristique différente de 2.

Une variante de ces polynômes, où on somme sur les permutations composées uniquement d'un seul cycle, permet d'obtenir une suite de circuits complète en toute caractéristique. Cette suite est notée (HC_n), "HC" correspondant à "Hamiltonian Circuit" et est définie par :

$$\mathrm{HC}_n(x_{1,1},\ldots,x_{n,n}) = \sum_{\sigma} \prod_{i=1}^{n} x_{i,\sigma(i)},$$

où la somme est sur tous cycles $\sigma \in S_n$ (c'est-à-dire sur tous les cycles hamiltoniens du graphe complet sur les sommets $\{1,\ldots,n\}$). Le résultat de complétude en toute caractéristique de la famille (HC_n) est due à Valiant [Val79b].

De nombreux autres problèmes intéressants de VNP sont obtenus comme ci-dessus en considérant une suite de graphes G_n dont les arêtes sont étiquetées par des variables génériques, et une propriété de graphe $\mathcal{P}$. La série génératrice des sous-graphes de G_n possédant la propriété $\mathcal{P}$ est définie par

$$P_n = \sum_{H \subset G_n, H \text{ vérifie } \mathcal{P}} w(H)$$

où $w(H)$ est le produit des étiquettes des arêtes de H. Si la suite de graphes (G_n) est simple à décrire et la propriété $\mathcal{P}$ facile à vérifier, la suite de polynômes (P_n) est dans VNP. Ceci fournit de nombreux autres problèmes complets. Par exemple, la série génératrice des cliques sur la suite des graphes complets (K_n) est VNP-complète [Bür00].

Nous sommes maintenant en mesure de définir les versions uniformes des classes de Valiant que nous allons considérer.

Définition 4 (Version uniforme des classes de Valiant) *Soit $\mathbb{K}$ un corps. Une suite de circuits (C_n) est appelée uniforme si l'encodage (booléen) de C_n peut être calculé en temps $n^{O(1)}$. Une suite de polynômes (P_n) sur $\mathbb{K}$ est dans la classe* unif-$\mathsf{VP}_{\mathbb{K}}$ *si elle est calculée par une suite de circuits arithmétiques sans constante de degré polynomial.*

Une suite de polynômes $(Q_n(x))$ sur $\mathbb{K}$ est dans la classe unif-$\mathsf{VNP}_{\mathbb{K}}$ *si Q_n a n variables $x = x_1, \ldots, x_n$ et s'il existe une suite $(P_n(x, y)) \in$ unif-$\mathsf{VP}_{\mathbb{K}}$ telle que*

$$Q_n(x) = \sum_{y \in \{0,1\}^{\ell_n}} P_n(x, y)$$

où ℓ_n est la longueur y dans P_n.

La condition d'uniformité imposée implique que la taille du circuit C_n dans la définition de unif-VP est polynomiale en n. On remarque que unif-$\mathsf{VP}_{\mathbb{K}}$ et unif-$\mathsf{VNP}_{\mathbb{K}}$ ne dépendent que de la caractéristique du corps $\mathbb{K}$: en effet, comme aucune constante de $\mathbb{K}$ n'est autorisée dans les circuits, ces classes sont égales à celles définies sur le sous-corps premier de $\mathbb{K}$.

Dans la définition de unif-VNP, nous avons choisi d'imposer que Q_n a n variables pour énoncer les résultats de manière concise. Notons tout de même que ce n'est *pas* la convention utilisée habituellement pour les classes non uniformes, où le nombre de variables est limité par la taille (polynomiale) du circuit.

Le "critère de Valiant" s'adapte facilement aux classes uniformes pour donner une caractérisation alternative de unif-VNP.

Proposition 7 (Critère de Valiant) *En caractéristique nulle, une famille (P_n) est dans* unif-VNP *si et seulement si P_n a n variables, un degré polynomial et ses coefficients sont calculables dans* GapP *: autrement dit, la fonction qui à $(c_1, \ldots, c_n)$ associe coefficient de $X_1^{c_1} \cdots X_n^{c_n}$ dans P_n appartient à* GapP*.*

*Il en est de même en caractéristique non nulle p avec des coefficients dans "*GapP $\mod p$*" (ceci est équivalent au fait que pour tout $v \in \mathbb{F}_p$, l'ensemble des monômes qui ont pour coefficient v est dans* $\mathsf{Mod}_p\mathsf{P}$*).*

Nous montrons maintenant comment les bornes inférieures en un polynôme fixé pour VNP sont reliées aux mêmes questions dans le cadre booléen. Par exemple, nous montrons que si NP n'a pas de circuits de taille n^k pour tout k, alors VNP n'a pas de circuits de tailles n^k pour tout k (sous GRH).

Théorème 8 *[FPdV14] On suppose GRH vraie. On suppose que l'une des propositions suivantes est vérifiée :*

1. $\mathsf{NP} \not\subset \mathsf{size}(n^k)$ *pour tout k ;*
2. $\mathsf{C}_{=}\mathsf{P} \not\subset \mathsf{size}(n^k)$ *pour tout k ;*
3. $\mathsf{MA} \subset \mathsf{size}(n^k)$ *pour un certain k ;*
4. $\mathsf{NP} = \mathsf{MA}$.

Alors unif-$\mathsf{VNP} \not\subset \mathsf{asize}_{\mathbb{C}}(n^k)$ *pour tout k.*

Remarquons que le premier point est une conséquence immédiate du second car $\mathsf{coNP} \subseteq \mathsf{C}_{=}\mathsf{P}$. Dans le résultat ci-dessus, nous voyons le phénomène observé dans d'autres contextes : la dérandomisation (ici de la classe MA) implique des bornes inférieures. Dans le cadre des corps finis, le résultat reliant les bornes inférieures sur VNP et les mêmes problèmes booléens est plus précis puisque nous obtenons une équivalence.

Théorème 9 *[FPdV14] Les deux propositions suivantes sont équivalentes :*
— *il existe k tel que* $\mathsf{unif\text{-}VNP}_{\mathbb{F}_2} \subset \mathsf{asize}_{\mathbb{F}_2}(n^k)$;
— $\mathsf{VP}_{\mathbb{F}_2} = \mathsf{VNP}_{\mathbb{F}_2}$ *et il existe k tel que* $\oplus\mathsf{P} \subset \mathsf{size}(n^k)$.

Terminons avec un résultat *non conditionnel* en caractéristique nulle. Nous n'autorisons plus les constantes arbitraires dans les circuits, uniquement les constantes $0, 1, -1$. Pour $s : \mathbb{N} \to \mathbb{N}$, on note $\mathsf{asize}_0(s)$ les suites de polynômes calculés par des circuits sans constante de taille $O(s)$ en caractéristique nulle. L'intérêt vient du fait que le degré des circuits n'est pas polynomialement borné : ainsi, l'utilisation de larges constantes, non arbitraires car construites par des petits circuits, n'est pas prohibée.

Théorème 10 *[FPdV14]* $\mathsf{unif\text{-}VNP} \not\subset \mathsf{asize}_0(n^k)$ *pour tout k.*

Perspectives

Montrer une borne inférieure en $\Omega(n^k)$ sur la taille des circuits booléens pour une fonction $\oplus\mathsf{P}$ permettrait d'obtenir des bornes inférieures pour $\mathsf{unif\text{-}VNP}_{\mathbb{F}_2}$ d'après le théorème 9. Notons que d'après [FSW09], cette question ne relativise pas puisqu'il existe un oracle pour lequel $\oplus\mathsf{P}$ a des circuits de taille $O(n^2)$.

Bien sûr une borne inférieure sur $\oplus\mathsf{P}$ donne aussi une borne inférieure sur $\mathsf{unif\text{-}VNP}$. Une façon alternative de conclure au même résultat serait une borne inférieure sur le bit de rang n^k d'une fonction $\#\mathsf{P}$. C'est une question ouverte qui nous semble intéressante ; en comparaison, on sait que les bits de rang polynomial quelconque des fonctions $\#\mathsf{P}$ contiennent PH et sont donc sans circuits de taille n^k.

1.5 Bornes inférieures pour le produit itéré de matrices

On étudie maintenant la complexité arithmétique du polynôme appelé *produit itéré de matrices*. Précisément, nous améliorons les bornes inférieures existantes pour les formules multilinéaires homogènes de profondeur 4 calculant ce polynôme.

Le produit itéré de matrices

Soient $n, d \geqslant 2$ deux entiers, et $X_1, \ldots, X_d$ les ensembles de variables suivants. Pour $i \in [d] \setminus \{1, d\}$, soit $X_i = \{x^{(i)}_{j,k}, j, k \in [n]\}$; pour $i \in \{1, d\}$, on pose $X_i = \{x^{(i)}_j, j \in [n]\}$. Le produit itéré de matrices, noté $\mathrm{IMM}_{n,d}$ (IMM signifie *Iterated Matrix Multiplication*) est défini par

$$\mathrm{IMM}_{n,d} = \sum_{j_1,\ldots,j_{d-1}} x^{(1)}_{j_1} x^{(2)}_{j_1,j_2} x^{(3)}_{j_2,j_3} \cdots x^{(d-1)}_{j_{d-2},j_{d-1}} x^{(d)}_{j_{d-1}}.$$

Ce polynôme correspond à l'entrée en position $(1, 1)$ du produit de d matrices génériques de taille $n \times n$.

Une façon alternative de définir ce polynôme est de le voir comme un *branching program* algébrique étagé avec $d+1$ étages successifs $V_0, \ldots, V_d$. Les étages 0 et d sont constitués d'un seul sommet : $V_i = \{v^{(i)}\}$ for $i \in \{0, d\}$. Les autre étages contiennent n sommets : $V_i = \{v^{(i)}_1, \ldots, v^{(i)}_n\}$ pour $0 < i < d$. Le graphe contient toutes les arêtes possibles de V_i à V_{i+1} pour $i \in \{0, \ldots, d-1\}$. On étiquette l'arête de $v^{(i-1)}_j$ à $v^{(i)}_k$ par la variable $x^{(i)}_{j,k}$ pour $0 < i < d-1$, l'arête de $v^{(0)}$ à $v^{(1)}_j$ par $x^{(1)}_j$ et l'arête de $v^{(d-1)}_j$ à $v^{(d)}$ par $x^{(d)}_j$. Le produit itéré de matrices est bien le polynôme calculé par ce *branching program*, c'est-à-dire la somme des poids des chemins de $v^{(0)}$ à $v^{(d)}$, le

poids d'un chemin étant le produit des variables étiquetant les arêtes de ce chemin, est égal à $\mathrm{IMM}_{n,d}$.

Une motivation pour étudier les polynômes $\mathrm{IMM}_{n,d}$ est qu'ils sont complets (pour les projections) pour la classe $\mathsf{VP}_{\mathsf{ws}}$ introduite dans la section 1.1.

Différentes variantes des formules arithmétiques

Les formules et circuits arithmétiques ont été définis à la section 1.1. Un circuit est dit étagé si les nœuds de calcul peuvent être partitionnés en niveaux, et chaque porte de calcul reçoit ses entrées du niveau précédent. On adopte ici la convention que les entrées correspondent au niveau 0.

Une formule $\Sigma\Pi\Sigma\Pi$ est une formule étagée où chaque porte des niveaux 1 et 3 est une porte $\times$ et chaque porte des niveaux 2 et 4 est une porte $+$. On utilisera aussi la notation $\Sigma\Pi^{(\alpha)}\Sigma\Pi^{(\beta)}$ pour indiquer que le fan-in des portes de niveau 1 est borné par α et celui des portes de niveau 3 par β.

On rappelle qu'une polynôme est homogène si chacun de ses monômes a le même degré. Une formule arithmétique est appelée *homogène* si chacune de ses portes calcule un polynôme homogène.

De même, une formule arithmétique est appelée *multilinéaire* si chacune de ses portes calcule un polynôme multilinéaire.

Un cas particulier de formule multilinéaire homogène est une formule *ensemble-multilinéaire*. Considérons une partition $S_1, S_2, \ldots, S_d$ de l'ensemble des variables S. Pour $T \subseteq [d]$, on dit qu'un monôme sur les variables de S est *T-multilinéaire* si il c'est produit de variables tel que exactement une variable provienne de chaque S_i pour $i \in T$. Un polynôme est dit T-multilinéaire si c'est une combinaison linéaire de monômes T-multilinéaires. Une formule est dite ensemble-multilinéaire (pour la partition des variables $(S_1, \ldots, S_d)$) si chacun de ses nœuds calcule une formule T-multilinéaire pour un ensemble $T \subseteq [d]$ (qui dépend du nœud). Une formule ensemble-multilinéaire pour $\mathrm{IMM}_{n,d}$ est considérée par rapport à la partition des variables $X_1, \ldots, X_d$ correspondant aux différentes matrices génériques.

La technique des dérivées partielles et extensions

La technique des dérivées partielles a été introduite par Nisan and Wigderson [NW97] pour prouver des bornes inférieures en complexité algébrique, en particulier sur le produit itéré de matrices.

Pour un polynôme $P \in \mathbb{F}[x_1, \ldots, x_n]$, définissons $\partial(P)$ l'espace vectoriel engendré par toutes les dérivées partielles (de tous ordres) du polynôme P, autrement dit :

$$\partial(P) = \mathrm{vect}\left\{ \frac{\partial^{i_1+\ldots+i_n} P}{\partial x_1^{i_1} \ldots \partial x_n^{i_n}} \;\middle|\; i_1, \ldots, i_n \in \mathbb{N} \right\}.$$

La mesure de complexité utilisée par Nisan et Wigderson est la dimension de $\partial(P)$. En utilisant cette technique, couplée avec la technique des restrictions aléatoires, ils ont obtenu des bornes inférieures sur la taille de formules particulières pour le produit itéré de matrices.

Théorème 11 (Nisan et Wigderson [NW97]) *Soit $r \geqslant 1$. Une formule calculant $\mathrm{IMM}_{n,d}$ qui est ensemble-multilinéaire et telle que tout chemin de la racine aux feuilles contient au plus r portes de produit a pour taille n^{d-1} pour $r = 1$ et $\exp(\Omega(d^{1/r}))$ pour $r \geqslant 2$.*

On peut remarquer que la dimension n des matrices n'intervient pas pour $r \geqslant 2$. Pour le cas $r = 2$, nous montrerons une borne en $n^{\Omega(\sqrt{d})}$ pour les formules $\Sigma\Pi\Sigma\Pi$ ensemble-multilinéaires (et même pour les formules multilinéaires homogènes).

On utilise une version plus forte introduite récemment par Kayal [Kay12] et Gupta et al. [GKKS13], c'est la méthode des *dérivées partielles décalées* (*shifted partial derivatives* en anglais).

Pour $k, \ell \in \mathbb{N}$ et un polynôme $f \in \mathbb{F}[x_1, \ldots, x_n]$, on définit

$$\langle \partial_k f \rangle_{\leqslant \ell} = \text{vect}\left\{ x_1^{j_1} \ldots x_n^{j_n} \cdot \frac{\partial^k f}{\partial x_1^{i_1} \ldots \partial x_n^{i_n}} \;\middle|\; i_1 + \ldots + i_n = k,\ j_1 + \ldots + j_n \leqslant \ell \right\}.$$

La mesure de complexité que nous utilisons est $\dim(\langle \partial_k f \rangle_{\leqslant \ell})$. Le fait d'utiliser les dérivées partielles décalées permet de faire la différence entre différentes structures d'espaces vectoriels engendrés par des polynômes comme le montre l'exemple-jouet suivant.

Exemple 4 *Considérons M l'ensemble des monômes $M = \{X^{5i}Y^{5j} \mid i, j \geqslant 0,\ i + j \leqslant 2\}$ (voir figure 1.4). On a $\dim\langle M \rangle_{\leqslant 2} = 36$. En revanche, pour $M' = \{X^{5+i}Y^{5+j} \mid i, j \geqslant 0,\ i + j \leqslant 2\}$, on a $\dim\langle M' \rangle_{\leqslant 2} = 15$ Ainsi, bien que $\dim$ Vect$(M) = \dim$ Vect(M'), le décalage permet de mettre en évidence la différence de structure entre M et M'.*

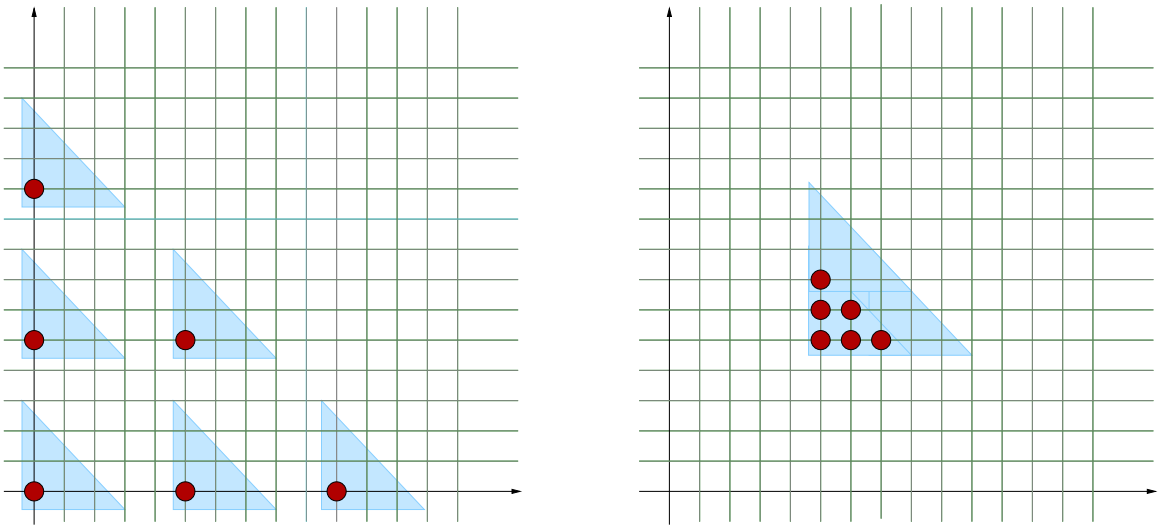

FIGURE 1.4 – Les monômes de M (à gauche) et M' (à droite) sont représentés par des points rouges de $\mathbb{Z}^2$. Les monômes de $\langle M \rangle_{\leqslant 2}$ et $\langle M' \rangle_{\leqslant 2}$ sont représentés par des aplats bleus.

Présentation des résultats et idée de la démonstration

On commence par énoncer un résultat sur une restriction des formules $\Sigma\Pi\Sigma\Pi$. Ce sont les formules $\Sigma\Pi^{(D)}\Sigma\Pi^{(t)}$, où les portes Π aux niveaux 3 et 1 ont un fan-in borné par D et t respectivement. Ce modèle de formules $\Sigma\Pi^{(D)}\Sigma\Pi^{(t)}$ a été l'objet de nombreux résultats récents [GKKS13, KSS13]. Voici ce que nous obtenons.

Théorème 12 *[FLMS14] Soit $n, d, D, t \in \mathbb{N}$ tels que $n \geqslant 10$ et $1 \leqslant t \leqslant d/160$. Toute formule $\Sigma\Pi^{(D)}\Sigma\Pi^{(t)}$ calculant $\mathrm{IMM}_{n,d}$ a pour fan-in à la racine $\left(\frac{n^{3/4}d}{Dt}\right)^{\Omega(d/t)}$. En particulier, si $D = O(d/t)$, alors toute formule $\Sigma\Pi^{(D)}\Sigma\Pi^{(t)}$ calculant $\mathrm{IMM}_{n,d}$ a pour fan-in à la racine $n^{\Omega(d/t)}$.*

Notre résultat principal concerne les formules multilinéaires homogènes.

Théorème 13 *[FLMS14] Pour n et d assez grands tels que $d = n^{O(1)}$, toute formule $\Sigma\Pi\Sigma\Pi$ multilinéaire homogène calculant $\mathrm{IMM}_{n,d}$ a pour taille $n^{\Omega(\sqrt{d})}$.*

Décrivons rapidement la stratégie pour obtenir des bornes inférieures sur la taille de certains types de formules $\Sigma\Pi\Sigma\Pi$ calculant $\mathrm{IMM}_{n,d}$.

Dans le travail de Gupta et al. [GKKS13], la méthode des dérivées partielles décalées a été utilisée pour montrer que n'importe quelle formule $\Sigma\Pi^{(O(n/t))}\Sigma\Pi^{(t)}$ (pas nécessairement homogène) calculant le permanent ou le déterminant de taille $n \times n$ a pour taille $\exp(\Omega(n/t))$. Leur preuve est constituée de deux étapes. D'abord, ils montrent que l'espace des dérivées partielles décalées d'une formule $\Sigma\Pi^{(D)}\Sigma\Pi^{(t)}$, pour certaines valeurs des paramètres D et t, a une *petite* dimension. Ensuite, il est montré que les dimensions de $\langle \partial_k \mathrm{Det}_n \rangle_{\leqslant \ell}$ et $\langle \partial_k \mathrm{Per}_n \rangle_{\leqslant \ell}$ sont grandes (pour des valeurs de paramètres k et ℓ appropriées).

On montre une borne inférieure sur la dimension de l'espace des dérivées partielles décalées de $\mathrm{IMM}_{n,d}$. Ceci permet d'obtenir une borne inférieure en $n^{\Omega(d/t)}$ pour les formules $\Sigma\Pi^{(D)}\Sigma\Pi^{(t)}$ calculant $\mathrm{IMM}_{n,d}$. En fait, on montre un résultat un peu plus fort : l'espace des dérivées partielles décalées est grand même pour des restrictions bien choisies de $\mathrm{IMM}_{n,d}$ (les restrictions que nous considérons ici consistent à remplacer certaines variables par 0). Ceci implique des bornes inférieures pour des formules $\Sigma\Pi^{(D)}\Sigma\Pi^{(t)}$ calculant ces restrictions de $\mathrm{IMM}_{n,d}$.

Afin d'obtenir la borne inférieure sur les formules multilinéaires homogènes, on se ramène aux formules $\Sigma\Pi\Sigma\Pi^{[t]}$ en utilisant la technique des *restrictions aléatoires* (déjà utilisées dans ce contexte dans [NW97], même si les restrictions utilisées ici sont bien différentes). Ceci est assez naturel, puisqu'une restriction aléatoire remplaçant n'importe quelle variable par 0 avec une probabilité suffisante supprime un monôme de haut degré avec une probabilité proche de 1. La difficulté consiste à définir des restrictions aléatoires qui laissent l'espace des dérivée partielles décalées de grande dimension.

Réduction à la profondeur 4

Les formules de profondeur 4 sont d'une importance particulière compte tenu du résultat d'Agrawal et Vinay [AV08] qui montre que pour certaines questions de bornes inférieures, il est suffisant de considérer ce type de formules plutôt que des circuits généraux. Ce résultat a été amélioré ensuite par Koiran [Koi12] puis Tavenas :

Théorème 14 (Tavenas [Tav13]) *Un polynôme de degré d en N variables calculé par un circuit de taille s peut aussi être calculé par une formule de profondeur 4 de taille $\exp(O(\sqrt{d \log(ds) \log N}))$. En particulier, tout polynôme de degré $d = N^{O(1)}$ qui a un circuit de taille $N^{O(1)}$ a une formule de profondeur $\Sigma\Pi\Sigma\Pi$ (et même $\Sigma\Pi^{(O(\sqrt{d}))}\Sigma\Pi^{(O(\sqrt{d}))}$). De plus, si le polynôme est homogène, la formule de profondeur 4 l'est également.*

Lien avec les réductions à la profondeur 4

Parallèlement à l'amélioration du théorème de réduction à la profondeur 4 de Tavenas (voir encadré), les bornes inférieures pour les formules $\Sigma\Pi^{(O(\sqrt{d}))}\Sigma\Pi^{(O(\sqrt{d}))}$ ont connu des progrès récents : Gupta, Kamath, Kayal et Saptharishi [GKKS13] ont obtenu des bornes inférieures en $\exp(\Omega(\sqrt{n}))$ sur la taille des formules $\Sigma\Pi^{(O(\sqrt{n}))}\Sigma\Pi^{(O(\sqrt{n}))}$ calculant le permanent ou le déterminant de taille $n \times n$. Remarquons que cela donne une borne inférieure en $\exp(\Omega(\sqrt{d}))$

pour un polynôme de la classe VP et de degré d ; à comparer à la borne de Tavenas en $\exp(O(\sqrt{d}\log N))$ où $N = n^2$. Après le résultat de Gupta et al. cité ci-dessus, le résultat suivant a été obtenu.

Théorème 15 (Kayal, Saha et Saptharishi [KSS14]) *Il existe une suite de polynômes (P_n) appartenant à* VNP*, P_n étant de degré $d = n$ en $N = n^2$ variables, tel que toute formule $\Sigma\Pi^{(O(\sqrt{d}))}\Sigma\Pi^{(O(\sqrt{d}))}$ calculant P_n a pour taille $\exp(\Omega(\sqrt{d}\log N))$.*

Notons que la borne inférieure fait apparaître un facteur $\log N$ qui ne se trouve pas dans la borne de Gupta et al. Ainsi, améliorer le résultat de Tavenas ou la borne inférieure de [KSS14] aurait pour conséquence de séparer VP de VNP.

Le théorème 13 permet d'obtenir les mêmes bornes inférieures que Kayal, Saha et Saptharishi mais pour le produit itéré de matrices, c'est-à-dire un polynôme de VP et non VNP. Ceci montre que la borne de Tavenas ne peut être améliorée pour les formules multilinéaires homogènes.

Application aux formules régulières

Rappelons tout d'abord les notions de degré et de degré syntaxique. On définit le degré d'un nœud d'une formule comme le degré du polynôme qu'il calcule. Le degré syntaxique d'un nœud est défini inductivement de la façon suivante. Le degré syntaxique d'un nœud d'entrée est 1 ; le degré syntaxique d'une porte étiquetée + est le maximum des degrés syntaxiques de ses deux portes d'entrée. Le degré syntaxique d'une porte étiquetée × est la somme des degrés syntaxiques de ses deux portes d'entrée. Par définition, le degré syntaxique d'une formule est celle de sa porte de sortie.

Une formule est appelée *régulière* si elle vérifie les conditions suivantes :

— la formule est étagée, chaque étage étant formé de portes + uniquement, ou de portes × uniquement, ou d'entrées ;
— sur un étage donné, toutes les portes ont le même fan-in ;
— le degré syntaxique de la formule est au plus deux fois le degré de la formule.

Les formules régulières ont été étudiées par Kayal, Saha, et Saptharishi [KSS13]. Ils ont montré l'existence d'un polynôme VNP de degré d sur N variables pour laquelle toute formule a pour taille $N^{\Omega(\log d)}$.

Dans cet article, la question est aussi posée de savoir si tout polynôme de degré d en N variables calculé par un *branching program* algébrique de taille polynomiale a une formule régulière de taille $N^{o(\log d)}$. Nous répondons à cette question par la négative.

Corollaire 2 *[FLMS14] Pour tout $n, d \in \mathbb{N}$ assez grands, toute formule régulière pour $\mathrm{IMM}_{n,d}$ a pour taille $n^{\Omega(\log d)}$.*

La borne ci-dessus est optimale puisque la construction usuelle d'une formule de taille $n^{O(\log d)}$ pour $\mathrm{IMM}_{n,d}$ fournit une formule régulière.

Perspectives

Les progrès suivant cette ligne de recherche sont très rapides. Mentionons en particulier l'obtention de bornes inférieures superpolynomiales pour les formules de profondeur 4 homogènes (non nécessairement multilinéaires) pour le produit itéré de matrices [KLSS14].

Une question intéressante serait de généraliser le théorème 13 aux formules de profondeur fixée, par exemple aux formules de profondeur 6.

1.6 Construction d'une famille de sous-espaces supplémentaires

Soit $\mathbb{K}$ un corps infini et $\mathcal{F}$ une famille de sous-espaces vectoriels de dimension r de $\mathbb{K}^n$. On dit que $\mathcal{F}$ a la propriété de transversalité $\mathcal{P}_{n,r}(\mathbb{K})$, ou est une *famille transversale*, si tout sous-espace vectoriel de $\mathbb{K}^n$ de dimension $(n-r)$ possède un supplémentaire dans $\mathcal{F}$. Autrement dit, pour tout sous-espace vectoriel $E \subset \mathbb{K}^n$ de dimension $n-r$, il existe $F \in \mathcal{F}$ tel que $E \cap F = \{0\}$.

Il est facile de voir l'existence de familles finies $\mathcal{F}$ possédant la propriété de transversalité. Par exemple, étant donné une base de $\mathbb{K}^n$, on peut prendre la famille de tous les sous-espaces vectoriels engendrés par r éléments de cette base. Ceci donne une famille transversale de cardinalité $\binom{n}{r}$. En fait Chistov [Chi85] a montré qu'il existe des familles transversales de cardinal $r(n-r)+1$. Dans la suite, on pose $\tau(n,r) = r(n-r)+1$.

La question de construire des familles de supplémentaires efficacement a été soulevée en relation avec la complexité du calcul du rang d'une matrice. L'existence de petites familles de supplémentaires, à petits coefficients, donnée dans [Chi85] est non effective : l'algorithme qui en découle pour le calcul du rang est donc non uniforme. Un algorithme parallèle uniforme pour le rang d'une matrice a été proposé peu après par Mulmuley [Mul86], mais sans résoudre la question de la construction efficace d'une famille de supplémentaires. Cette propriété de transversalité a aussi été utilisée en théorie de la complexité [CNS96] sur les corps finis et en géométrie algébrique effective [RV02] sur les corps infinis.

On s'intéresse dans cette section à donner des algorithmes en temps polynomial pour construire des familles transversales. On commence par donner une telle construction sur les corps infinis ; puis on s'intéresse au cas des corps finis. Enfin, on discute la complexité d'un problème relié, qui est de savoir si des points de $\mathbb{R}^n$ sont en position générale. C'est le problème qui a motivé ces travaux initialement.

Construction sur les corps infinis

On donne maintenant une construction explicite de familles transversales de cardinalité $\tau(n,r) = 1 + r(n-r)$ dans le cas d'un espace vectoriel sur un corps de caractéristique nulle. Étant donné $d \in \mathbb{N}$, on note $v_d(x)$ le vecteur de Vandermonde $(1, x, \ldots, x^d)$. Plus généralement, étant donné $\bar{\alpha} = (\alpha_1, \ldots, \alpha_p) \in \mathbb{N}^p$, on note $v_{\bar{\alpha}}(x) = (x^{\alpha_1}, \ldots, x^{\alpha_p})$.

Proposition 8 *[CFGK03] Soit $\mathbb{K}$ un corps de caractéristique 0. La famille $(V_i)_{0 \leqslant i \leqslant r(n-r)}$ où $V_i = \mathrm{Vect}\{v_{n-1}(i), \ldots, v_{n-1}(i+r-1)\}$ a la propriété $\mathcal{P}_{n,r}(\mathbb{K})$.*

On remarque que les vecteurs de base des éléments de la famille ci-dessus ont une écriture en binaire de taille polynomiale. On s'intéresse dans ce qui suit aux corps infinis de caractéristique non nulle : alors que la construction précédente est valide en caractéristique nulle uniquement, les deux constructions données maintenant fonctionnent en caractéristique quelconque. On note $\mathrm{ord}(\theta)$ l'ordre du groupe multiplicatif engendré par θ.

Proposition 9 *[CFGK03] Soit $\mathbb{K}$ un corps infini (de caractéristique arbitraire). Soit $(z_i)_{0 \leqslant i \leqslant r(n-r)}$ une famille d'éléments non nuls distincts de $\mathbb{K}$. Si $\mathrm{ord}(\theta) \geqslant n$, la famille $(V_i)_{0 \leqslant i \leqslant r(n-r)}$ où*

$$V_i = \mathrm{vect}\{v_{n-1}(z_i), v_{n-1}(\theta z_i), \ldots, v_{n-1}(\theta^{r-1} z_i)\}$$

a la propriété $\mathcal{P}_{n,r}(\mathbb{K})$.

On donne maintenant une famille un peu différente possédant aussi la propriété de transversalité.

Proposition 10 *[CFGK03] Soit $\mathbb{K}$ un corps infini. Soit $a_0, \ldots, a_{n-1}$ des éléments distincts de $\mathbb{K}$. Pour $0 \leqslant i \leqslant r-1$, on définit le vecteur colonne $v_i(t) = (a_0^i, a_1^i t, a_2^i t^2, \ldots, a_{n-1}^i t^{n-1})$. Soit $(z_i)_{0 \leqslant i \leqslant r(n-r)}$ des éléments non nuls distincts de $\mathbb{K}$. La famille $(V_i)_{0 \leqslant i \leqslant r(n-r)}$ où $V_i = \text{vect}\{v_0(z_i), \ldots, v_{r-1}(z_i)\}$ a la propriété $\mathcal{P}_{n,r}(\mathbb{K})$.*

Sur un corps algébriquement clos $\mathbb{K}$, on sait qu'on ne peut pas construire une famille possédant la propriété $\mathcal{P}_{n,r}(\mathbb{K})$ si $\mathbb{K}$ est de cardinal strictement inférieur à $\tau(n,r)$ (ceci vient de la géométrie algébrique : $r(n-r)$ est la dimension de la grassmannienne des sous-espaces vectoriels de dimension r de $\mathbb{K}^n$). Les constructions sont donc optimales du point de vue de la cardinalité pour les corps algébriquement clos. En revanche, nous ne connaissons pas de borne inférieure non triviale sur les autres corps, par exemple $\mathbb{R}$ ou $\mathbb{Q}$. Nous pouvons cependant remarquer que le cas réel est différent du cas complexe, comme le montre l'exemple ci-dessous.

Remarque 1 *Considérons la famille des sous-espaces vectoriels de $\mathbb{R}^4$ de dimension 2 engendrés par les colonnes des matrices suivantes (de taille 4×2) :*

$$\begin{pmatrix} I \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ I \end{pmatrix}, \begin{pmatrix} I \\ I \end{pmatrix}, \begin{pmatrix} R_{\pi/2} \\ I \end{pmatrix}.$$

Ci-dessus I est la matrice identité de taille 2×2 et $R_{\pi/2}$ est la matrice de rotation d'angle $\pi/2$. Cette famille a la propriété $\mathcal{P}_{4,2}(\mathbb{R})$ et est de taille $4 < \tau(4,2)$.

Construction sur les corps finis

On vient de voir des constructions de familles de taille $n^{O(1)}$ avec la propriété $\mathcal{P}_{n,r}(\mathbb{K})$ sur n'importe quel corps infini $\mathbb{K}$. Sait-on construire rapidement des familles transversales sur les corps finis ? C'est la question que nous étudions maintenant.

Les constructions données dans les propositions 9 et 10 sont en fait valides dès que le corps $\mathbb{K}$ est assez grand comparé à n comme le montre une inspection des preuves correspondantes. Précisément, ces constructions fonctionnent si $|\mathbb{K}| = \Omega(n^2)$. Sans réussir à trouver de construction explicite de familles transversales sur les corps finis, nous améliorons la borne précédente en donnant des constructions de familles transversales en temps polynomial pour $|\mathbb{K}| = \Omega(n)$. En effet, considérons le $\mathbb{F}_q$-espace vectoriel $\mathbb{F}_q^n$. Par la proposition 10, étant donnés $0 \leqslant r \leqslant n$ et $q = p^{\nu_0}$ (avec p premier), si $q > r(n-r)+1$ alors on peut construire une famille $\mathcal{F}$ de taille $|\mathcal{F}| = r(n-r)+1$ ayant la propriété $\mathcal{P}_{n,r}(\mathbb{F}_q)$. On améliore l'inégalité $q > r(n-r)+1$ en montrant qu'on sait construire une famille (un peu plus grande) pour $q \geqslant \lfloor n/2 \rfloor$.

Théorème 16 *[CFKP08] Soit $q = p^{\nu_0} \geqslant \min\{r, n-r\}$ où $0 \leqslant r \leqslant n$. On peut construire une famille $\mathcal{F}$ ayant la propriété $\mathcal{P}_{n,r}(\mathbb{F}_q)$ avec $|\mathcal{F}| \leqslant \tau(n,r)^3$. De plus, l'algorithme calculant $\mathcal{F}$ fonctionne en temps polynomial en n, p et ν_0.*

Généralisons un peu la notion de famille transversale. Considérons deux corps $\Bbbk \subset \mathbb{K}$. Un sous-espace vectoriel V de $\Bbbk^n$ définit d'une façon naturelle un sous-espace vectoriel de $\mathbb{K}^n$: l'espace vectoriel $\hat{V}$ engendré par V. Soit $\mathcal{F}$ une famille de sous-espaces vectoriels de $\Bbbk^n$ de dimension r. On dit que $\mathcal{F}$ a la propriété $\mathcal{P}_{n,r}(\Bbbk, \mathbb{K})$ si pour tout sous-espace vectoriel $W \subset \mathbb{K}^n$ de dimension $(n-r)$, il existe $V \in \mathcal{F}$ tel que $\hat{V} \cap W = \{0\}$.

Un argument probabiliste simple montre qu'il existe des familles de taille $O(\tau(n,r))$ ayant la propriété $\mathcal{P}_{n,r}(\mathbb{F}_q)$, où la constante dans le grand-O est indépendante de q. On peut montrer un résultat de même nature pour la propriété plus forte $\mathcal{P}_{n,r}(\mathbb{F}_q, \overline{\mathbb{F}}_q)$.

Proposition 11 *[CFKP08] Il existe une famille de taille $O(n(n-r)r^2)$ qui a la propriété $\mathcal{P}_{n,r}(\mathbb{F}_q, \overline{\mathbb{F}}_q)$, où la constante cachée est indépendante de q.*

La propriété ci-dessus est obtenue par un argument probabiliste en utilisant les bornes sur le nombre de conditions de signe d'un ensemble de polynômes (voir l'encadré ci-dessous).

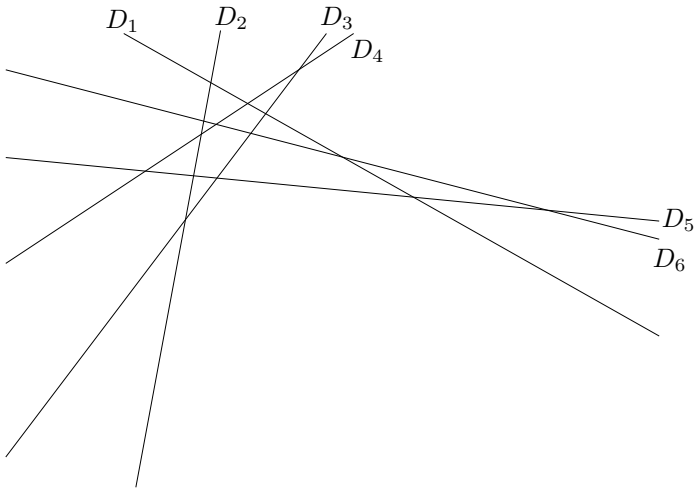

FIGURE 1.5 – Arrangement de 6 droites en position générique dans le plan

Borne sur le nombre de conditions de signe

Combien de régions délimitent m droites du plan en position générique ? Dans l'exemple de la figure 1.5, les 6 droites $D_1, \ldots, D_6$ délimitent 22 régions dans le plan.

Pour $\theta \in \mathbb{R}$ on note $\sigma(\theta)$ le signe de θ défini par $\sigma(0) = 0$ et $\sigma(\theta) = \theta/|\theta|$ sinon. Considérons m droites du plan $D_1, \ldots, D_m$ comme dans la figure 1.5. Notons ℓ_i l'équation de la i-ème droite. En un point $x \in \mathbb{R}^2$ du plan, on considère le vecteur de signe

$$v(x) = (\sigma(\ell_1(x)), \ldots, \sigma(\ell_m(x))) \in \{-1, 0, 1\}^m.$$

Le nombre de régions délimitées par les n droites $D_1, \ldots, D_m$ est le nombre de conditions de signe à coordonnées dans $\{-1, 1\}$ effectivement réalisées, c'est-à-dire

$$\text{Nombre de régions} = \left|\{v(x), x \in \mathbb{R}^2\} \cap \{-1, 1\}^m\right|.$$

Alors que le nombre de conditions de signe potentiel est de 2^m, le nombre de conditions de signe réalisées est $O(m^2)$.

Dans le cas de polynômes réels, des bornes sur le nombre de conditions de signe ont été obtenues par Grigoriev [Gri88] puis améliorées par Basu, Pollack et Roy (voir par exemple [BPR06]).

Ce type de borne se généralise au cas des conditions de signe réalisées par des polynômes de petit degré dans des espaces de dimension supérieure sur des corps ordonnés ou non. Sur les corps non ordonnés tels que $\mathbb{C}$, les conditions de signe sont à valeur dans "$= 0$" et "$\neq 0$". Voici la borne utilisée dans la démonstration de la proposition 11.

Théorème 17 *[RBG01] Considérons $f_1, \ldots, f_m$ un ensemble de polynômes à n variables (avec $m \geqslant n$) sur le corps $\mathbb{F}$, chaque polynôme étant de degré total au plus d. Le nombre de conditions de signe réalisées par le vecteur $(f_1(x), \ldots, f_m(x))$, quand $x \in \mathbb{F}^n$, est majoré par $\binom{md}{n}$.*

On donne maintenant la construction de familles transversales possédant la propriété $\mathcal{P}_{n,r}(\mathbb{F}_q, \overline{\mathbb{F}}_q)$. Encore une fois, les constructions données sur les corps infinis ont cette propriété et conviennent dès que $q > r(n-r)+1$. C'est le cas des "petit corps" qui nous intéresse ici.

Théorème 18 *[CFKP08] Soit $0 < \varepsilon < 1/2$, $0 \leqslant r \leqslant n$ deux entiers et $q = p^{\nu_0} \geqslant \varepsilon n$. Soit $n \geqslant \varepsilon^{-3}$. On peut construire une famille $\mathcal{F}$ possédant la propriété $\mathcal{P}_{n,r}(\mathbb{F}_q, \overline{\mathbb{F}}_q)$ de taille $|\mathcal{F}| = O(q^{2\lceil r/q \rceil}\tau(n,r))$ (où la constante cachée est absolue). L'algorithme pour calculer $\mathcal{F}$ fonctionne en temps polynomial en n, p, ν_0 et $|\mathcal{F}|$; il est donc polynomial en $n^{1/\varepsilon}$, p et ν_0.*

Bien sûr on voudrait, pour toute puissance de nombre premier q, avoir un algorithme qui étant donné n et r construit une famille $\mathcal{F}$ avec la propriété $\mathcal{P}_{n,r}(\mathbb{F}_q)$ (ou même la propriété plus forte $\mathcal{P}_{n,r}(\mathbb{F}_q, \overline{\mathbb{F}}_q)$) en temps $n^{O(1)}$, ce qui impliquerait $|\mathcal{F}| = n^{O(1)}$. À notre connaissance aucun algorithme réalisant cette construction n'est connu à ce jour.

Tester si des points sont en position générale

Considérons le problème suivant :

> NullDet
> *Entrée :* $n, m \in \mathbb{N}$, une matrice A de taille $n \times m$
> *Problème :* Existe-t-il un sous-déterminant de taille $n \times n$ de A égal à 0 ?

La question de la NP-complétude de NullDet s'est posée dès 1982 [CKM82]. L'une des motivations vient de la géométrie algorithmique : il correspond effectivement à vérifier si m points de $\mathbb{R}^n$ sont ou non en position générale (la matrice dont les colonnes sont les coordonnées des points est une instance positive de NullDet si les points ne sont pas en position générale). Il est utile de pouvoir vérifier efficacement si les points sont ou non en position générale. C'est pour cette raison que la question est à nouveau soulevée dans [DGH98] en 1998. Un problème de même nature est le suivant :

> κ-Det
> *Entrée :* $n, m \in \mathbb{N}$, une matrice A de taille $n \times m$ à coefficients dans $\mathbb{Z}$
> *Problème :* Existe-t-il un sous-déterminant de taille $n \times n$ de A égal à κ ?

Dyer, Gritzmann and Hufnagel [DGH98] ont montré que κ-Det est NP-complet pour $\kappa \neq 0$. Mais leur preuve ne s'applique pas au cas particulier $\kappa = 0$, qui correspond au problème NullDet défini ci-dessus. En fait, la complexité du problème NullDet avait déjà été résolu par Khachiyan [Kha95]. Une preuve différente est donnée par Erickson [Eri99]. Les méthodes développées pour construire les familles de supplémentaires permettent de donner une preuve alternative :

Proposition 12 *[Kha95, Eri99, CFGK03]* NullDet *est* NP*-complet pour les réductions many-one en temps polynomial.*

Disons un mot de la méthode. Considérons le problème suivant :

> Max-FLS
> *Entrée :* m points de $\mathbb{Z}^n$ et $k \in \mathbb{N}$
> *Problème :* Décider s'il existe un hyperplan homogène contenant au moins k de ces points.

On sait que le problème est NP-complet [AK95]. On peut reformuler ce problème en terme de matrice : si on considère la matrice $A \in \mathbb{Z}^{n \times m}$ dont les colonnes sont les coordonnées des points d'entrée, la question est de savoir s'il existe k colonnes linéairement dépendantes. La proposition 8 donne immédiatement la NP-difficulté de ce problème pour des réductions de type *truth-table*. Une analyse plus fine permet de montrer la difficulté pour les réductions de type *many-one* comme énoncé dans la proposition 12.

Perspectives

Il serait intéressant de comprendre la complexité de NullDet dans le modèle des arbres de calcul algébriques. Par exemple, existe-t-il des arbres de calcul algébriques de profondeur polynomiale pour résoudre ce problème, comme il en existe pour d'autre problèmes NP-complets de nature géométrique (par exemple le Knapsack peut être décidé par des arbres de calcul algébriques de profondeur polynomiale [Mei93, Mey84, Mey88]).

Chapitre 2

Géométrie algorithmique et optimisation

Cette partie traite de la complexité de problèmes de géométrie et d'optimisation tels que le calcul du diamètre euclidien d'un ensemble de points de l'espace, ou le calcul d'une fonction en escalier à distance minimale d'un ensemble de points du plan. On connaît des algorithmes polynomiaux (souvent quadratiques ou cubiques) pour les problèmes considérés dans ce chapitre ; le but ici n'est donc pas de prouver des bornes inférieures exponentielles comme dans le chapitre précédent, mais de préciser au mieux la complexité de ces problèmes. Le modèle de la machine de Turing (réelle) n'est pas adapté à ces questions fines et c'est donc la machine RAM, présentée ci-dessous, qui est utilisée.

2.1 RAM sur les réels, arbres de calcul algébriques et borne de Ben-Or

La machine RAM réelle est le modèle de calcul généralement utilisé pour analyser la complexité des problèmes de géométrie algorithmique [PS85].

C'est une machine à accès indirect (c'est-à-dire qu'on peut lire le contenu d'une case mémoire en temps constant à partir de son adresse) où chaque case mémoire peut contenir un entier ou un nombre réel. Cette machine peut effectuer des comparaisons et les opérations arithmétiques $(+, -, \times, /)$ entre deux nombres réels ou deux nombres entiers en un pas de calcul. Cependant, il n'est pas autorisé de convertir un nombre réel en entier (par exemple avec la fonction partie entière). Les valeurs entières sont utilisées pour faire de l'adressage indirect, mais les nombres réels ne peuvent pas être utilisés pour cela. Par défaut, une RAM réelle ne peut utiliser que les constantes entières et réelles 0 et 1, mais il est aussi possible de considérer des machines pouvant utiliser des constantes réelles arbitraires.

Décrivons maintenant le modèle des arbres de calcul algébriques. Ces arbres décident une partie de $\mathbb{R}^n$ (pour $n \in \mathbb{N}$ fixé) de la façon suivante. Notons l'entrée $x = (x_1, x_2, \ldots, x_n)$. Un arbre de calcul algébrique est un arbre unaire-binaire où chaque nœud est soit un nœud de calcul (unaire), soit un nœud de branchement (binaire), soit une feuille. Un nœud de calcul u est étiqueté par une opération arithmétique $\{+, -, \times, /, \sqrt{\cdot}\}$ et chacune de ses entrées est une constante réelle, une entrée x_i ou le résultat d'un calcul obtenu en un nœud ancêtre de u. En chaque nœud de branchement v, on compare la valeur obtenue en un nœud ancêtre de v à 0 par l'une des opérations $\{>, \geqslant, =\}$. Selon le résultat de ce test, le programme va dans l'un ou l'autre des fils de v. Ainsi, sur une entrée x, le programme suit un chemin de l'arbre de calcul qui termine dans une feuille. Les feuilles sont étiquetées par *accepte* ou *rejette*. On dit qu'un

arbre décide l'ensemble $W \subset \mathbb{R}^n$ si on atteint une feuille étiquetée *accepte* sur chaque entrée $x \in W$ et une feuille étiquetée *rejette* sinon.

Le résultat suivant est très utile pour obtenir des bornes inférieures :

Théorème 19 (Ben-Or [BO83]) *Un arbre de calcul algébrique qui décide un ensemble $W \subset \mathbb{R}^n$ a une profondeur $\Omega(\log(\#W) - n)$, où $\#W$ est le nombre de composantes connexes de W.*

Cette borne peut être étendue aux machines RAM réelles qui ne prennent que des nombres réels comme entrée. En effet, supposons qu'une machine RAM réelle (avec des constantes réelles arbitraires) décide un ensemble $W \subset \mathbb{R}^n$. Comme l'entrée est composée de n nombres réels et pas d'entiers, la machine RAM ne peut faire des accès mémoire qu'à des adresses fixées (indépendantes de l'entrée, dépendant uniquement du chemin de calcul). Ainsi, tous les calculs et les branchements de la machine RAM peuvent être dépliés dans un arbre de calcul algébrique qui décide W, et dont la profondeur correspond au temps de calcul de la machine RAM. Une borne inférieure sur la profondeur des arbres de calcul algébriques décidant W se traduit donc par la même borne inférieure sur le temps de calcul d'une machine RAM décidant W.

2.2 Complexité du diamètre d'un polytope en dimension 3

Soit P un ensemble de n points de $\mathbb{R}^d$. Calculer le diamètre de cet ensemble

$$\mathrm{diam}(P) = \max\{\mathrm{d}(x, y) \mid x, y \in P\}$$

pour la distance euclidienne $\mathrm{d}(\cdot, \cdot)$ est un problème fondamental en géométrie algorithmique qui a été beaucoup étudié. Si $P \subset \mathbb{R}^2$, le diamètre de P peut être calculé en temps $O(n \log n)$ [PS85], ce qui est optimal dans le modèle des arbres de calcul algébriques. Le cas du diamètre en dimension 3 est resté ouvert plus longtemps : Clarkson et Shor [CS89] on été les premiers à donner un algorithme probabiliste optimal en temps $O(n \log n)$ pour calculer le diamètre d'un ensemble de n points de $\mathbb{R}^3$, et Ramos [Ram01] a réussi à obtenir un algorithme déterministe avec le même temps de calcul.

La borne inférieure en $\Omega(n \log n)$ sur le calcul du diamètre d'un ensemble de points $P \subset \mathbb{R}^2$ ne tient plus si P est donné comme les sommets d'un polygone convexe ordonnés le long du bord : dans ce cas il existe un algorithme en temps $O(n)$ calculant le diamètre [PS85]. Comme motivation, nous rappelons ci-dessous l'algorithme et les bornes inférieures en dimension 2.

La question principale à laquelle nous nous sommes intéressés est de savoir si la donnée de la structure de l'enveloppe convexe de P dans $\mathbb{R}^3$ permet aussi de calculer le diamètre d'un ensemble de points plus rapidement.

Diamètre d'un polygone de $\mathbb{R}^2$

Une réduction simple à partir du problème *set disjointness* (étant donnés deux ensembles $A, B \subset \mathbb{R}$, décider si $A \cap B = \emptyset$) montre que le problème du diamètre d'un ensemble de n points de $\mathbb{R}^2$ a une complexité $\Omega(n \log n)$ (et il existe un algorithme réalisant cette borne).

Cette borne inférieure n'est plus valide lorsqu'on se donne l'enveloppe convexe d'un ensemble de points de $\mathbb{R}^2$. On peut dans ce cas calculer le diamètre en temps $O(n)$. La technique consiste à appuyer deux droites parallèles ℓ et ℓ' sur le polygone et les faire tourner (voir figure 2.1). Le maximum de la distance entre ℓ et ℓ' est le diamètre du polygone. De plus, les points de l'enveloppe convexe étant donnés dans l'ordre, l'algorithme est bien réalisé en temps linéaire.

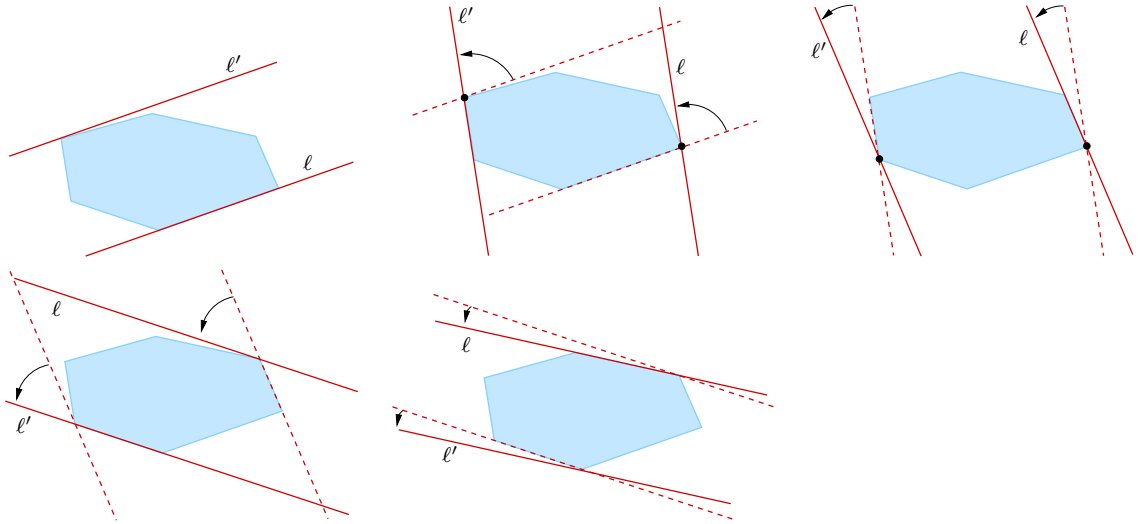

FIGURE 2.1 – Calcul du diamètre d'un polygone en faisant tourner deux droites parallèles.

Borne inférieure sur le problème du diamètre d'un polytope en dimension 3

On va montrer qu'on ne peut pas calculer le diamètre d'un polytope de $\mathbb{R}^3$ plus rapidement que celui d'un ensemble de points comme c'est le cas en dimension 2. Ainsi, il faut un temps $\Omega(n \log n)$ pour calculer le diamètre d'un polytope de $\mathbb{R}^3$. Cette borne inférieure montre que l'algorithme de Ramos [Ram01] est optimal pour calculer le diamètre d'un polytope de $\mathbb{R}^3$.

Précisons la façon dont un polytope est donné en entrée. La *structure combinatoire* d'un polytope $\mathbb{R}^3$, ou 3-polytope, est l'ensemble des relations d'inclusion entre les sommets, les arêtes, et les faces de ce polytope. Ainsi, on suppose ci-dessous qu'un polytope est donné par :

- les coordonnées de tous les sommets du polytope ;
- pour chaque face, la liste des sommets de cette face dans l'ordre du bord de cette face.

Théorème 20 *[FV07] Un arbre de calcul algébrique qui décide si un polytope de $\mathbb{R}^3$ à n sommets a un diamètre inférieur à 1 a une profondeur $\Omega(n \log n)$.*

La démonstration du théorème 20 repose sur la borne de Ben-Or. Pour cela nous avons construit un polytope de $\mathbb{R}^3$ avec les caractéristiques suivantes (voir figure 2.2) : les sommets sont donnés par des points fixes

$$A = \{a_j \mid j \in \{-n, \dots, n\}\} \text{ et } C = \{c_j^i \mid j \in \{-n, \dots, n\}, i \in \{-1, 1\}\}$$

et des points mobiles

$$B(\bar{\beta}) = \{b_j(\beta_j) \mid -n+1 \leqslant j \leqslant n-1\}.$$

Ici chaque paramètre réel β_j varie dans un intervalle $I \subset \mathbb{R}$ fixé. Le polytope $P(\bar{\beta})$ est l'enveloppe convexe de $A \cup B(\bar{\beta}) \cup C$. Ce polytope a les propriétés suivantes :

- La structure de l'enveloppe convexe de $A \cup B(\bar{\beta}) \cup C$ ne dépend pas de $\bar{\beta}$.
- $\operatorname{diam}(A \cup B(\bar{\beta}) \cup C) = \operatorname{diam}(A, B(\bar{\beta}))$.
- L'ensemble $\{b_j(\beta_j) \mid \beta_j \in I \text{ et } \operatorname{diam}(A, \{b_j(\beta_j)\}) < 1\}$ a au moins $2n$ composantes connexes (figure 2.3). Il en découle que

  $$\{\bar{\beta} \in I^{|\bar{\beta}|} \mid \operatorname{diam}(P(\bar{\beta})) < 1\}$$

  est formé de $n^{\Theta(n)}$ hypercubes disjoints.

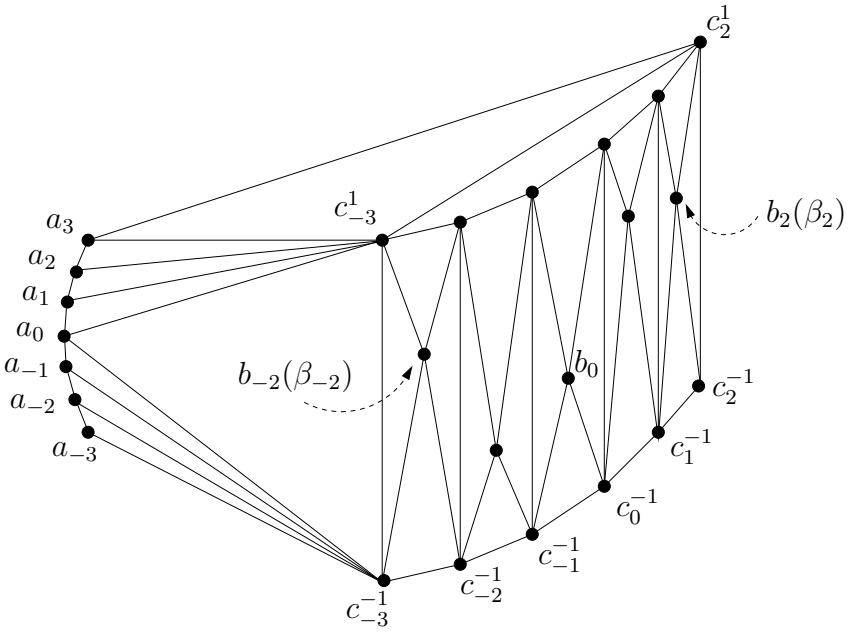

FIGURE 2.2 – La structure du 3-polytope : les ensembles $A, B(\bar{\beta})$ et C pour $n = 3$.

Notons D_n l'ensemble des polytopes de diamètre au plus 1 et S_n l'ensemble des polytopes $P(\bar{\beta})$. Le dernier point ci-dessus permet de montrer que la complexité de $D_n \cap S_n$ est $\Omega(n \log n)$ par la borne de Ben-Or. Comme S_n peut être décidé en temps linéaire (puisque chaque point $b_i(\beta_i)$ décrit un arc de cercle), on conclut que la complexité de D_n (i.e. celle des polytopes de diamètre inférieur à 1) est aussi $\Omega(n \log n)$.

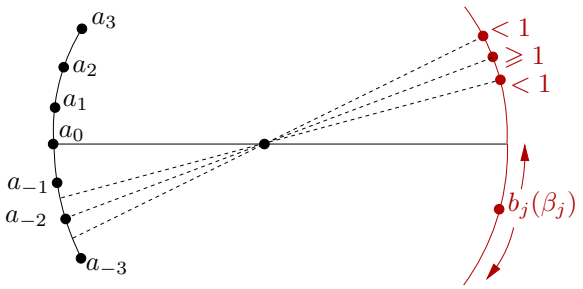

FIGURE 2.3 – La distance entre A et $b_j(\beta_j)$ pour n=3.

Corollaire 3 *[FV07] Une* RAM *réelle (pouvant utiliser des constantes réelles arbitraires) qui décide si un polytope de* $\mathbb{R}^3$ *à* n *sommets a un diamètre inférieur à* 1 *fonctionne en temps* $\Omega(n \log n)$.

Diamètre en dimension supérieure et problème de Hopcroft

Que sait-on sur la complexité de décider le diamètre d'un ensemble de points de $\mathbb{R}^d$ pour $d > 3$? Il n'existe pas de borne précise comme pour les dimensions 2 et 3. La seule borne inférieure connue est $\Omega(n \log n)$ comme en dimension 3 et le meilleur algorithme est dû à Matoušek et

Schwarzkopf [MS93] : le diamètre d'un ensemble de n points de $\mathbb{R}^d$ peut être calculé en temps $O(n^{2-2/(\lceil d/2 \rceil+1)} \log^{O(1)} n)$.

Nous aller comparer le problème du diamètre au problème de Hopcroft défini comme ceci :

> HOPCROFT
> *Entrée :* n points et n droites dans $\mathbb{R}^2$
> *Problème :* Décider si un point se trouve sur une droite.

Ce problème de Hopcroft est un bien connu [Eri96]. Matoušek [Mat93] a donné un algorithme en temps $O\left(n^{4/3} 2^{O(\log^* n)}\right)$ pour décider le problème de Hopcroft, et aucun algorithme en temps $O(n^{4/3})$ n'est connu pour ce problème. La borne inférieure connue est de $\Omega(n \log n)$ sur la profondeur des arbres de calcul algébriques. De plus, Erickson a montré une borne inférieure en $\Omega(n^{4/3})$ sur un modèle de calcul affaibli. Ainsi, exhiber une réduction du problème de Hopcroft à un autre problème suggère que ce problème est difficile à résoudre en temps $o(n^{4/3})$. Erickson a obtenu plusieurs réductions pour divers problèmes géométriques [Eri95] ; par exemple, il a montré que le lancer de rayons dans un terrain polyédral, ou décider si l'intersection de demi-espaces de $\mathbb{R}^5$ sont plus durs que le problème de Hopcroft.

On compare maintenant des problèmes de calcul de diamètre au problème de Hopcroft. Une variante du diamètre d'un ensemble de points, appelé *diamètre rouge-bleu* est la suivante : les points sont partitionnés en deux ensembles, rouges et bleus, et on demande de calculer le maximum de la distance entre un point rouge et un point bleu.

Proposition 13 *[FV07] Il existe une réduction en temps linéaire du problème de Hopcroft au diamètre rouge-bleu dans $\mathbb{R}^6$, ou au diamètre dans $\mathbb{R}^7$, avec une machine* RAM *réelle utilisant la constante $\sqrt{2}$. De même, il existe une réduction en temps linéaire du problème de Hopcroft au problème du diamètre rouge-bleu dans $\mathbb{R}^9$, ou au diamètre dans $\mathbb{R}^{10}$, pour une machine* RAM *réelle sans constantes.*

Notons que l'algorithme de Matoušek pour le diamètre dans $\mathbb{R}^7$ donne une borne supérieure en $O(n^{1.6} \log^{O(1)} n))$.

Perspectives

Les résultats concernant le diamètre d'un ensemble de points en dimension supérieure n'ont pas de raison d'être optimaux : la meilleure borne inférieure sur le problème de Hopcroft est $\Omega(n \log n)$, et même s'il n'existe pas d'algorithme connu en temps $o(n^{4/3})$ pour ce problème, le résultat de la proposition 13 n'est pas satisfaisant puisque le meilleur algorithme pour le diamètre rouge-bleu dans $\mathbb{R}^6$ est en temps $O(n^{3/2} \log^{O(1)} n)$. En revanche, le diamètre rouge-bleu dans $\mathbb{R}^4$ peut être résolu en temps $O(n^{4/3} \log^{O(1)} n)$, donc il serait très intéressant de réduire Hopcroft à ce problème (de manière un peu similaire, Erickson [Eri95] demande si le diamètre euclidien dans $\mathbb{R}^4$ est plus dur que de tester si une intersection de demi-espaces $\mathbb{R}^5$ est vide).

2.3 Approximation d'un ensemble de points du plan par une fonction en escalier

Une fonction $f : [a, b) \to \mathbb{R}$ est appelée *fonction en escalier à k marches* s'il existe une suite de réels $a = a_0 < a_1 < \cdots < a_k = b$ tels que f est constante sur chaque intervalle $[a_i, a_{i+1})$. On note $\mathrm{d}(p, f)$ la distance verticale d'un point $p \in \mathbb{R}^2$ à f ; autrement dit, pour $p = (x, y)$, on a $\mathrm{d}(p, f) = |f(x) - y|$. Soit P un ensemble de n points de $\mathbb{R}^2$. On définit $\mathrm{d}(P, f)$ la distance de P à f par

$$\mathrm{d}(P, f) = \max\{\mathrm{d}(p, f) \mid p \in P\}.$$

On considère le problème suivant :

> MIN-DIST
> *Entrée :* Un entier k et un ensemble P de n points de $\mathbb{R}^2$
> *Problème :* Calculer une fonction en escalier à k marches f^* qui minimise $\varepsilon^* = \mathrm{d}(P, f^*)$.

Le problème de décision associé au problème ci-dessus est le suivant :

> MIN-DIST-DECISION
> *Entrée :* Un ensemble P de n points de $\mathbb{R}^2$ triés selon leur abscisse, un entier k et $\varepsilon > 0$
> *Problème :* Décider s'il existe une fonction en escalier f à k marches tel que $\mathrm{d}(P, f) \leqslant \varepsilon$.

Si l'on souhaite minimiser le nombre de marches k pour une distance donnée ε, Diáz-Báñez et Mesa [DBM01] ont montré qu'on peut le faire en temps $O(n)$ avec une approche gloutonne. Cet algorithme peut être utilisé comme algorithme de décision pour le problème MIN-DIST, et comme la distance optimale ε^* dans le problème MIN-DIST est la demi-différence des ordonnées de deux points de P, trier ces valeurs et faire une recherche dichotomique a permis à Diáz-Báñez et Mesa [DBM01] d'obtenir un algorithme en $O(n^2 \log n)$. Ceci a été amélioré à $O(n^2)$ par Wang [Wan02], et Mayster et Lopez [ML06] ont obtenu un algorithme en temps $O(n \log n + k^2 \log(\frac{n}{k})^2)$ pour le même problème. Puis Guha et Shim [GS07a] ont trouvé un algorithme en $O(n + k^2 \log^3 n)$ quand les points sont donnés en ordre trié selon leur abscisse.

Ci-dessous nous donnons un algorithme simple en $O(n \log n)$ pour le problème MIN-DIST. Cet algorithme est optimal quand les points donnés en entrée ne sont pas triés. Notons en particulier que tous les algorithmes précédents [CD06, GS07a, ML06, DBM01, Wan02] sont quadratiques, c'est-à-dire en $\Omega(n^2)$, $\Omega(k^2)$, ou $\Omega(nk)$. L'amélioration que nous avons apportée est particulièrement significative pour les bases de données où n et k peuvent être suffisamment grands pour qu'une complexité quadratique soit rédhibitoire.

Algorithme pour le cas non pondéré

Un algorithme glouton permet de démontrer ceci :

Lemme 1 *[DBM01] Il existe un algorithme en temps $O(n)$ pour décider le problème* MIN-DIST-DECISION.

Définissons E l'ensemble des demi-différences des ordonnées des points de P :

$$E = \left\{ \frac{y - y'}{2} \mid (x, y) \in P \text{ and } (x', y') \in P \right\}.$$

Il est facile de vérifier que $\varepsilon^* \in E$. Une façon de résoudre le problème MIN-DIST est de trier E en temps $O(n^2 \log n)$, puis de faire une recherche dichotomique pour déterminer ε^* en utilisant l'algorithme de décision du lemme 1. Pour éviter un temps quadratique, nous allons utiliser la technique de recherche dans une matrice triée (voir encadré).

La technique de recherche dans une matrice triée de Frederickson et Johnson

Une matrice est dite triée si pour tout $i \leqslant i'$ et $j \leqslant j'$, $M_{i,j} \leqslant M_{i',j'}$. Supposons qu'on se donne une matrice triée M. Soit $g : \mathbb{R} \to \{\text{TRUE}, \text{FALSE}\}$ une fonction monotone au sens

suivant : si $x < y$ et $g(x) = \text{TRUE}$, alors $g(y) = \text{TRUE}$. Notre but est de trouver le plus petit élément $M_{i,j}$ de la matrice M vérifiant $g(M_{i,j}) = \text{TRUE}$. Nous appelons ce problème le *problème d'optimisation*. Le *problème de décision* consiste à calculer g : étant donné $x \in \mathbb{R}$, calculer $g(x)$.

On suppose dans la suite qu'on peut accéder aux éléments de M en temps constant. De plus on suppose que le problème de décision peut être résolu en temps D. Le problème d'optimisation peut être résolu de façon naïve en temps $O(n^2 \log n + D \log n)$ en triant $\{M_{i,j} \mid 1 \leqslant i, j \leqslant n\}$ puis en effectuant une recherche dichotomique. La technique de Frederickson et Johnson permet d'obtenir :

Théorème 21 (Frederickson et Johnson [FJ84, Fre91]) *Étant donné une matrice triée de taille $n \times n$ et un algorithme de décision en temps D, le problème de recherche dans une matrice triée peut être résolu en temps $O(n + D \log n)$.*

L'algorithme maintient une collection $\mathcal{M}$ de sous-matrices de M. Initialement, on pose $\mathcal{M} = \{M\}$. À chaque étape, on partitionne chaque matrice $\mathcal{M}$ en quatre matrices carrées de mêmes tailles. On ne calcule pas ces matrices explicitement ce qui prendrait un temps quadratique : on représente ces matrices par les indices des lignes et des colonnes de la matrice initiale correspondant à chaque sous-matrice. Ensuite, on élimine environ la moitié de ces matrices (comme expliqué ci-dessous), puis on itère ce procédé jusqu'à ce qu'il ne reste qu'un nombre constant de matrices de tailles 1×1. Ces éléments sont testés avec l'algorithme de décision.

Il reste à expliquer comment on supprime des sous-matrices. Pour chaque sous-matrice de $\mathcal{M}$, on calcule le plus petit élément (qui se trouve dans le coin supérieur gauche). On calcule ensuite la médiane $\lambda_{\min}$ de tous ces éléments. On calcule de même la médiane $\lambda_{\max}$ de tous les plus grands éléments des sous-matrices de $\mathcal{M}$. Supposons que $g(\lambda_{\min}) = \text{TRUE}$. Alors on peut supprimer toutes les sous-matrices dont le plus petit élément est plus grand que $\lambda_{\min}$. On traite de façon symétrique le cas $g(\lambda_{\min}) = \text{FALSE}$, et on fait de même avec $\lambda_{\max}$.

L'utilisation de la technique de recherche dans une matrice triée est rendue possible par la remarque suivante :

$$E = Y + (-Y) \text{ où } Y = \left\{ \frac{y}{2} \mid (x, y) \in P \right\}.$$

On commence par trier Y pour obtenir une suite croissante (b_i) telle que $Y = \{b_1 \leqslant b_2 \cdots \leqslant b_n\}$. Ensuite, on représente les éléments de E comme les entrées d'une matrice triée M définie par $M_{i,j} = \frac{1}{2} b_i - \frac{1}{2} b_{n+1-j}$. La technique de recherche dans une matrice triée, en utilisant l'algorithme de décision décrit au lemme 1, permet d'obtenir le résultat suivant :

Théorème 22 *[FV11] Le problème* Min-Dist *peut être décidé en temps $O(n \log n)$.*

Une réduction simple à partir du tri permet de voir que cet algorithme est optimal (quand on suppose que P n'est pas donné en ordre trié).

Cas d'une entrée triée

On suppose maintenant que l'ensemble de points $P = \{p_i = (x_i, y_i) \mid 1 \leqslant i \leqslant n\}$ est donné trié selon l'ordre des abscisses : pour tout i, $x_i < x_{i+1}$. Dans ce cas, nous avons le résultat suivant.

Théorème 23 *[FV11] Le problème* Min-Dist *avec une entrée triée peut être résolu en temps linéaire.*

Le résultat ci-dessus se base sur une généralisation du problème de partitionnement d'un chemin de Frederickson [Fre91]. Un autre ingrédient est la structure de données de Gabow et al. [GBT84] pour reporter la valeur maximum d'un intervalle d'un tableau en temps $O(1)$ (après un calcul initial en temps linéaire).

Généralisation aux entrées pondérées

Un point pondéré dans le plan est un triplet $p = (x, y, w) \in \mathbb{R}^3$ où $(x, y) \in \mathbb{R}^2$ représente les coordonnées de p et $w > 0$ est un poids associé à p. On utilise maintenant la distance d'un point pondéré $p = (x, y, w)$ à une fonction f définie par $\mathrm{d}'(p, f) = w \cdot |f(x) - y|$. Pour un ensemble de points pondérés P, on définit

$$\mathrm{d}'(P, f) = \max\{\mathrm{d}'(p, f) \mid p \in P\}.$$

On définit alors le problème :

> WEIGHTED-MIN-DIST
> *Entrée :* Un entier k et un ensemble P de n points pondérés de $\mathbb{R}^2$
> *Problème :* Calculer une fonction en escalier à k marches f^* qui minimise $\varepsilon^* = \mathrm{d}'(P, f^*)$.

Une telle approximation est souvent appelée histogramme. Elle est beaucoup utilisée en base de données où on cherche une représentation compacte d'un ensemble de points qui puisse tenir en mémoire, pour optimiser les temps de requête [GS07b].

Le cas pondéré a été étudié par Guha et Shim [GS07b] ; ils ont donné un algorithme en temps $O(n \log n + k^2 \log^6 n)$. Lopez et Mayster [LM08] ont ensuite proposé un algorithme en $O(n^2)$, ce qui est plus rapide pour les grandes valeurs de k. Avec Antoine Vigneron [FV11], nous avons proposé le premier algorithme sous-quadratique en $O(n \log^4 n)$ pour ce problème. Chen and Wang [CW09] l'ont amélioré en un algorithme en temps $O(\min(n \log^2 n, n \log n + k^2 \log^2 \frac{n}{k} \log n \log\log n))$. Enfin, un algorithme probabiliste en temps $O(n \log n)$ a été obtenu par Liu [Liu10]. Ce temps est optimal puisque nous avons remarqué à la section précédente que ce problème nécessite déjà un temps $\Omega(n \log n)$ dans le cas non pondéré. Nous nous sommes ensuite penchés sur la dérandomisation de l'algorithme de Liu.

Théorème 24 *[FV13] Étant donné un ensemble de n points pondérés du plan et un entier $k > 0$, une fonction f en escalier à k marches minimisant la distance $\mathrm{d}'(P, f)$ peut être calculée en temps déterministe $O(n \log n)$.*

Le résultat ci-dessus est obtenu en combinant des idées de travaux précédents sur ce problème [GS07b, KSM07] avec la technique de recherche paramétrée améliorée due à Cole [Col87].

L'algorithme obtenu ici a essentiellement un intérêt théorique, puisque l'algorithme de Cole qu'il utilise repose sur les réseaux de tri de profondeur $O(\log n)$, et toutes les constructions connues ont des algorithmes mettant en jeu de larges constantes [AKS83, Pat90] (voir aussi Knuth [Knu98]).

Quand les points sont donnés non triés, l'algorithme ci-dessus est optimal puisque c'était déjà le cas dans le cas non pondéré. Une question intrigante est de savoir s'il existe un algorithme en temps $o(n \log n)$ quand les points sont donnés triés dans l'ordre des abscisses, comme cela se produit dans le cas non pondéré.

Le théorème 24 a une application directe au problème du k-centre sur la droite réelle, défini de la manière suivante.

k-Center
Entrée : Un ensemble de n points $r_1 < \cdots < r_n \in \mathbb{R}$ avec des poids $w_1, \ldots, w_n$
Problème : Calculer un ensemble de k centres $c_1, \ldots, c_k \in \mathbb{R}$ qui minimise

$$\max_{i \in \{1,\ldots,n\}} w_i \cdot \mathrm{d}(r_i, \{c_1, \ldots, c_k\}).$$

On peut réduire le problème du k-centre sur la droite réelle au calcul d'une fonction en escalier en temps linéaire de la façon suivante. Étant donné une instance du k-centre, on considère les points $p_i = (i, r_i)$ pour $i = 1, \ldots, n$ avec les mêmes poids $w_1, \ldots, w_n$. Les ordonnées des k marches d'une fonction optimale pour approximer ces points réalisent un k-centre optimal. Ainsi, on peut résoudre le problème k-Center défini ci-dessus en temps $O(n \log n)$, ce qui améliore un résultat récent de Chen et Wang [CW11].

Cas d'une entrée bruitée

Dans cette section, on considère une généralisation du problème Min-Dist où on autorise h points à être à une distance plus que ε de P. L'idée de pouvoir enlever un certain nombre de points de l'entrée est de rendre l'algorithme plus robuste au bruit. Le problème est donc défini comme ceci.

Outlier
Entrée : Un ensemble de points $P \subseteq \mathbb{R}^2$, $h, k \in \mathbb{N}$
Problème : Une fonction f à k marches et un sous-ensemble $P' \subseteq P$ avec $|P'| \leqslant h$ minimisant $\mathrm{d}(P \setminus P', f)$.

Quand $|P| = n$, une approche dynamique permet d'obtenir un algorithme en temps $O(n^2 h^2)$. Ce qui est intéressant est le cas où h est beaucoup plus petit que n ; on cherche dans ce cas à améliorer la dépendance en n.

Proposition 14 *[FV11] Le problème* Outlier *peut être résolu en temps $O(n \log n \cdot h^2)$. Si les entrées sont triées selon les abscisses, on peut le résoudre en temps $O(n \cdot h^2 \log h)$.*

Le résultat ci-dessus dans le cas trié est obtenu en appliquant la méthode de Frederickson déjà mentionnée.

2.4 Hyperbolicité d'une métrique

Gromov a introduit la notion d'hyperbolicité d'un espace métrique [BS00, Gro87] en utilisant la condition suivante.

(Condition sur quatre points.) Un espace métrique (M, d) est dit δ-hyperbolique pour $\delta \geqslant 0$ s'il vérifie la *condition des quatre points* : Pour tout $x, y, z, t \in M$, les deux plus grandes sommes de distances parmi $d(x, y) + d(z, t)$, $d(x, z) + d(y, t)$, et $d(x, t) + d(y, z)$ diffèrent d'au plus 2δ.

L'*hyperbolicité de Gromov* δ^* de (M, d) est la plus petite valeur $\delta^* \geqslant 0$ telle que (M, d) est δ^*-hyperbolique.

Cette définition permet de calculer l'hyperbolicité d'un espace métrique sur n points en temps $\Theta(n^4)$ en parcourant tous les quadruplets de points. Ceci n'est pas acceptable pour certaines applications aux réseaux [CCL12]. Connaître l'hyperbolicité est important car les temps de calcul et espaces mémoire nécessaires pour certains algorithmes sur les espace hyperboliques sont souvent exprimés en fonction de l'hyperbolicité de Gromov [CDE+12, CE07, KL06].

Pour tout $x, y, r \in M$, le *produit de Gromov* de x et y au point r est défini comme

$$(x|y)_r = \frac{1}{2}\left(d(x,r) + d(y,r) - d(x,y)\right).$$

Le point r est appelé *point de base*. L'hyperbolicité de Gromov peut être définie en terme de produits de Gromov, plutôt qu'avec la condition des quatre points ci-dessus. Ainsi, un espace métrique (M, d) est δ-hyperbolique si et seulement si pour tout $x, y, z, r \in M$

$$(x|z)_r \geqslant \min\{(x|y)_r, (y|z)_r\} - \delta.$$

L'hyperbolicité de Gromov δ^* est la plus petite valeur de δ qui satisfait la propriété ci-dessus. Autrement dit :

$$\delta^* = \max_{x,y,z,r} \left\{\min\{(x|y)_r, (y|z)_r\} - (x|z)_r\right\}.$$

On définit l'hyperbolicité δ_r au point de base r par

$$\delta_r = \max_{x,y,z} \left\{\min\{(x|y)_r, (y|z)_r\} - (x|z)_r\right\}.$$

Ainsi $\delta^* = \max_r \delta_r$.

L'exposant de la multiplication de matrice μ est défini comme la borne inférieure de l'ensemble des réels $\omega > 0$ tel que le produit de deux matrices de tailles $n \times n$ peut être calculé en temps $O(n^\omega)$, les opérations arithmétiques ayant un coût unitaire [vzGG03]. La meilleure borne connue est inférieure à 2.373 [Wil12]. Dans la suite, on note ω un nombre réel tel qu'on peut multiplier deux matrices réelles de tailles $n \times n$ en temps $O(n^\omega)$.

Complexité du calcul de l'hyperbolicité à un point de base fixé

On considère le problème de décision suivant.

> Fixed-Base Hyperbolicity
> *Entrée :* Une métrique sur n points, un point r et $\alpha \geqslant 0$
> *Problème :* Décider si $\delta_r > \alpha$.

Précisons qu'on ne demande pas de vérifier que l'entrée est une métrique (aucun algorithme en $o(n^3)$ n'est connu pour cela [WW10]). Une métrique d'arbre est une métrique pour laquelle $\delta_r = 0$ pour un point de base r (ou de manière équivalente pour tout point de base r). On sait décider en temps $O(n^2)$ si une métrique est une métrique d'arbre [Ban90].

Il semble qu'il n'y ait pas d'autre algorithme connu pour calculer l'hyperbolicité au point de base r que la méthode naïve consistant à parcourir tous les triplets de points de l'espace [CFHM12], soit une complexité en $O(n^3)$. On donne ci dessous un algorithme exact basé sur le produit (max,min) de matrices, dont on discute ensuite l'optimalité.

Le produit (max,min) de deux matrices réelles A, B, noté $A \otimes B$, est défini par :

$$(A \otimes B)_{ij} = \max_k \min\{A_{ik}, B_{kj}\}.$$

Remarquons que par dualité, calculer un produit (max,min) est équivalent à calculer un produit (min,max). Duan et Pettie [DP09] on donné un algorithme en temps $O(n^{(3+\omega)/2})$ pour calculer le produit (max,min) de deux matrices de tailles $n \times n$. Une remarque simple permet d'obtenir la borne suivante pour le calcul de l'hyperbolicité.

Proposition 15 *[FIV13] Le problème* Fixed-Base Hyperbolicity *peut être décidé en temps* $O(n^{(3+\omega)/2})$.

En fait, nous allons montrer que le calcul de l'hyperbolicité est intimement lié au produit (max,min). D'après la proposition ci-dessus, toute amélioration de la complexité du produit (max,min) de matrices permet d'améliorer la complexité du calcul de l'hyperbolicité. Nous allons montrer une réciproque partielle : tout amélioration du calcul de l'hyperbolicité pour un point de base fixé en dessous d'un exposant fixé permet d'obtenir un meilleur algorithme pour le produit (min,max).

Notre principal outil est un résultat de Vassilevska Williams et Williams [WW10] cité ci-dessous dans le cas spécial des structures (min,max). Nous devons d'abord définir la notion de triangle négatif dans un graphe triparti.

Définition 5 (Triangle négatif d'un graphe triparti) *Étant donné un graphe triparti* $G = (I \cup J \cup K, E)$ *avec des poids* $w : E \to \mathbb{R}$*, un triangle* $(i, j, k) \in I \times J \times K$ *est appelé* négatif *si* $\max\{w_{i,k}, w_{k,j}\} + w_{i,j} < 0$.

Remarquons que la définition ci-dessus n'est pas symétrique en I, J, K : seuls I et J sont interchangeables.

Théorème 25 (Vassilevska-Williams et Williams) *[WW10] Soit* $T(n)$ *une fonction telle que* $T(n)/n$ *est croissante. Supposons qu'on puisse décider si un graphe triparti à* n *nœuds contient un triangle négatif en temps* $T(n)$*. Alors le produit (min,max) de deux matrices de tailles* $n \times n$ *peut être effectué en temps* $O(n^2 T(n^{1/3}) \log W)$*, où* W *est la valeur absolue de la plus grande valeur de la matrice produit.*

Nous pouvons montrer ceci.

Lemme 2 *[FIV13] L'existence d'un triangle négatif dans un graphe triparti peut être réduit à l'hyperbolicité en base fixée en temps quadratique.*

Ceci permet d'obtenir le résultat suivant.

Proposition 16 *[FIV13] Si l'hyperbolicité en base fixée peut être décidée en temps* $O(n^\nu)$*, avec* $\nu \geqslant 2$*, alors le produit (max,min) de deux matrices de taille* $n \times n$ *peut être décidé en temps* $O(n^{2+\nu/3} \log n)$.

La proposition ci-dessus implique qu'un algorithme en $O(n^\nu)$ avec $\nu < 3(\omega - 1)/2$ pour l'hyperbolicité en base fixée donnerait un algorithme en $O(n^\tau)$ avec $\tau < (3+\omega)/2$ pour le produit (max,min). Comme la meilleure borne connue sur le produit de matrices donne $3(\omega-1)/2 > 2.05$, tout algorithme en $O(n^{2.05})$ pour l'hyperbolicité en base fixée mènerait à un meilleur algorithme pour le produit (max,min).

Approximations de l'hyperbolicité

Notre but est maintenant de calculer une approximation de l'hyperbolicité de Gromov δ^* d'une métrique sur n points. On sait que l'hyperbolicité δ_r par rapport à n'importe quel point de base r est une 2-approximation de δ^* [BS00]. Ainsi, l'algorithme précédent fournit une 2-approximation. A-t-on un algorithme plus efficace si l'on n'a pas besoin d'une approximation aussi précise ? Gromov [Gro87] (voir aussi Chepoi et al. [CDE+12, Theorem 1] et le livre de Ghys et de la Harpe [GdlH90, Chapter 2]) ont montré que tout espace métrique (M, d) peut être plongé dans un arbre T avec une erreur additive $2\delta \log_2 n$, et un tel arbre peut être construit en temps quadratique. Plus précisément, si on note d_T la distance dans l'arbre T, alors

$$d(a, b) - 2\delta^* \log_2 n \leqslant d_T(a, b) \leqslant d(a, b) \text{ pour tout } a, b \in M.$$

Ceci nous a permis d'obtenir un algorithme d'approximation avec un facteur $O(\log n)$:

Proposition 17 *[FIV13] On peut calculer les approximations suivantes de l'hyperbolicité δ^* d'une distance sur n points :*

- *avec un facteur 2 en temps $O(n^{(3+\omega)/2})$;*
- *avec un facteur $2\log_2 n$ en temps $O(n^2)$.*

Travaux ultérieurs et problème ouvert

Suite à nos travaux, Duan [Dua14] a adapté l'algorithme du produit (max,min) au cas spécifique utile ici afin d'obtenir deux algorithmes d'approximation pour le calcul de l'hyperbolicité d'une métrique sur n points :

- une $(1+\varepsilon)$-approximation en temps $(1/\varepsilon)n^{\omega+1}\log^{O(1)}(n)$ (où ω est un réel tel que deux matrices (booléennes) de tailles $n \times n$ peuvent être multipliées en temps $O(n^\omega)$) ;
- une $(2+\varepsilon)$-approximation en temps $(1/\varepsilon)n^{\omega}\log^{O(1)}(n)$.

Après ces travaux, la question de la complexité du calcul exact de l'hyperbolicité de Gromov demeure.

Chapitre 3

Formules booléennes aléatoires

L'étude de la distribution de probabilité induite par des formules booléennes aléatoires a été initiée par Paris et al. [PVW94]. Ce travail se concentrait sur des formules construites sur les connecteurs *et* et *ou*. Lefman et Savický [LS97] ont obtenu dans ce cadre des bornes inférieure et supérieure sur la probabilité d'une fonction booléenne sur ces modèles, bornes dépendant de la complexité de cette fonction.

C'est la ligne de recherche que nous avons poursuivi sur les formules construites sur le connecteur de l'implication. Dans un premier temps, nous avons étudié les tautologies, ceci étant motivé par des questions de comparaison entre les logiques classique et intuitionniste soulevées par Zaionc [MTZ00]. Puis nous nous sommes penchés sur l'estimation de la probabilité d'une fonction booléenne quelconque.

Dans la littérature, d'autres travaux se sont concentrés sur les fonctions calculées par des formules équilibrées (c'est-à-dire où toutes les feuilles sont à la même hauteur) sur un seul connecteur. C'est par exemple cette technique qui a permis à Valiant [Val84] d'obtenir une petite formule calculant la fonction majorité. Cette ligne de recherche à été poursuivie par Brodsky et Pippenger [BP05] qui ont effectué une étude systématique de la distribution de probabilité sur les fonctions booléennes obtenue en itérant un unique connecteur. C'est dans cette perspective que nous avons travaillé en considérant les formules équilibrées toujours, mais donc les nœuds internes sont étiquetés de manière uniforme par les connecteurs *et* et *ou*.

3.1 Formules booléennes aléatoires sur l'implication

Formules sur le connecteur de l'implication

On considère des formules sur les variables $\{x_1, \ldots, x_k\}$ et le seul connecteur de l'implication. Une telle formule peut être représentée par un arbre binaire dont les nœuds internes sont étiquetés par l'implication (notée $\to$) et les feuilles par des variables de l'ensemble $\{x_1, \ldots, x_k\}$. On note cet ensemble de formules $\mathcal{F}_k$. À chaque formule A on associe la fonction booléenne qu'elle calcule, notée $[A]$. On appelle tautologie une formule A vérifiant $[A] = \mathit{Vrai}$.

Toute formule $A \in \mathcal{F}_k$ s'écrit de manière unique

$$A = A_1 \to (A_2 \to (\ldots \to (A_p \to r(A)) \ldots))$$

où $A_i \in \mathcal{F}_k$ et $r(A) \in \{x_1, \ldots, x_k\}$ – voir figure 3.1. On note une telle formule $A_1, \ldots, A_p \to r(A)$. Les formules $A_1, \ldots, A_p$ sont appelées les *prémisses* de A, et la feuille la plus à droite $r(A)$ le *but* de A. De manière analogue, on définit les prémisses et le but de toute sous-formule de A ; les but des prémisses de A sont appelés les *sous-buts* de A (les sous-buts A sont $r(A_1), \ldots, r(A_p)$).

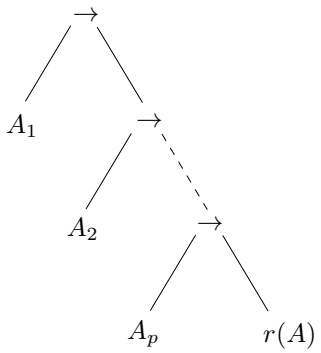

FIGURE 3.1 – Décomposition d'une formule le long de sa branche droite.

Fraction limite

On définit la taille $|A|$ d'une formule A comme le nombre de ses feuilles. Notre but est de quantifier la fraction des arbres calculant une fonction donnée f. Pour préciser cette notion, nous devons introduire la définition suivante.

Définition 6 *La* fraction limite *d'un sous-ensemble $\mathcal{A} \subseteq \mathcal{F}_k$ des formules est définie comme*

$$\mu_k(\mathcal{A}) = \lim_{n\to\infty} \frac{|\{A \in \mathcal{A} \ : \ |A| = n\}|}{|\{A \in \mathcal{F}_k \ : \ |A| = n\}|}$$

si la limite existe.

On note $\mathcal{F}_k(f)$ l'ensemble des formules de $\mathcal{F}_k$ calculant une fonction booléenne f. Ainsi $\mathcal{F}_k(f) = \{A \in \mathcal{F}_k, [A] = f\}$ et on définit la fraction limite d'une fonction f *comme la fraction limite de $\mathcal{F}_k(f)$.*

La première question qui se pose est l'existence de cette fraction limite.

Proposition 18 *[FGGG12] Pour toute fonction booléenne f, la fraction limite $\mu_k(\mathcal{F}_k(f))$ est bien définie.*

La proposition ci-dessus est une conséquence simple du théorème de Drmota-Lalley-Woods (voir par exemple le livre de Flajolet et Sedgewick [FS09]). Dans la suite, $\mu_k(\mathcal{F}_k(f))$ est abrégé en $\mu_k(f)$.

Séries génératrices et analyse combinatoire

Soit $(a_n)_{n\in\mathbb{N}}$ une suite de nombres réels. La *série génératrice ordinaire* de (a_n) est la série formelle $\sum_{n=0}^{\infty} a_n z^n$. En considérant z comme une variable complexe, on sait que cette série converge vers une fonction $f(z)$ dans un disque ouvert maximal $\{z \in \mathbb{C} : |z| < R\}$ où R est le rayon de convergence de la série. La fonction $f(z)$ est appelée la *fonction génératrice ordinaire* de (a_n). Pour une fonction $g(z)$ analytique dans un disque contenant 0, on note $[z^n]\ g(z)$ le coefficient de z^n dans le développement en série de $g(z)$ en 0.

Beaucoup de questions concernant le comportement asymptotique de la suite (a_n) peuvent être résolues en étudiant le comportement de la fonction génératrice f près du cercle des

complexes de module R. C'est cette approche qui est utilisée pour obtenir les fractions limites des classes de formules considérées dans cette section.

On considère la série génératrice énumérant les formules d'une classe donnée par rapport à leur taille (nombre de feuilles de l'arbre représentant la formule), en utilisant des fonctions génératrices à une variable z. Ceci nous permet d'obtenir une fonction génératrice $\phi(z)$ pour la classe considérée. Si $f_k(z)$ est la fonction génératrice de toutes les formules sur k variables $\mathcal{F}_k$, la fraction limite est donnée par $\lim_{n\to\infty}[z^n]\phi(z)/[z^n]f_k(z)$. On peut parfois donner la valeur exacte de cette limite, ou un équivalent lorsque $k \to \infty$.

Pour obtenir la série formelle (et donc la série génératrice) d'une classe de formules données, il est parfois possible d'appliquer des constructions combinatoires à des formules de base, ces constructions se traduisant de façon simple sur les séries génératrices. Nous rappelons maintenant les trois constructions de ce type les plus simples. Si A et B sont deux classes d'objets combinatoires de séries génératrices $f_A(z)$ et $f_B(z)$, l'union disjointe $A \sqcup B$ a pour série génératrice $f_A(z) + f_B(z)$. Le produit cartésien $A \times B$ (avec la taille d'un élément (a, b) égale à la somme des tailles de a et de b) est donné par le produit $f_A(z)f_B(z)$. Enfin, les suites finies (de longueur quelconque) d'éléments de A est énumérée par $1/(1 - f_A(z))$.

La structure simple des tautologies

On appelle tautologie une formule qui calcule la fonction *Vrai*.

Définition 7 *Soit $\mathcal{S}_k$ l'ensemble des formules sur les variables $\{x_1, \ldots, x_k\}$ de la forme*

$$T = A_1, \ldots, A_p \to r(T),$$

où l'un (au moins) des A_i est égal à $r(T)$. Ces formules sont des tautologies que nous appellerons simples.

Théorème 26 *[FGGZ07] Asymptotiquement (quand le nombre de variables booléennes k tend vers l'infini), toutes les tautologies sont simples. Autrement dit,*

$$\lim_{k\to\infty} \frac{\mu_k(\mathcal{S}_k)}{\mu_k(\mathit{Vrai})} = 1.$$

On en déduit

$$\mu_k(\mathit{Vrai}) = \frac{1}{k} + O\left(\frac{1}{k^2}\right).$$

Ceci répond à une question posée dans [MTZ00], la motivation invoquée dans cet article étant de comparer logiques classique et intuitionniste.

Logique intuitionniste

Une tautologie au sens classique est une fonction qui s'évalue à vrai pour toute valeur des variables booléennes. Une formule du calcul propositionnel est une tautologie au sens intuitionniste (sur le connecteur de l'implication seul) si on peut démontrer cette formule à l'aide des règles de déduction suivantes :

— Axiomes

$$\frac{}{\Gamma, A \vdash A}$$

— Introduction de $\to$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

— Élimination de $\to$ (Modus Ponens)

$$\frac{\Gamma \vdash A \quad \Gamma \vdash A \to B}{\Gamma \vdash B}$$

Toute tautologie au sens intuitionniste est une tautologie au sens classique mais la réciproque n'est pas vraie comme le montrer la formule de Pierce :

$$((x \to y) \to x) \to x.$$

On définit Cl_k l'ensemble des tautologies classiques, c'est-à-dire les formules qui s'évaluent à *Vrai* pour toute valuation. On sait que $\mu_k(Cl_k)$ est bien défini (de même que la fraction limite des formules calculant n'importe quelle fonction booléenne fixée, comme il est expliqué plus loin). On définit de même Int_k l'ensemble tautologies intuitionnistes. Cette fois, on ne sait pas montrer que cet ensemble a une fraction limite bien définie ; c'est pourquoi nous posons $\mu_k^-(Int_k) = \liminf_{n\to\infty} \frac{|Int_k^n|}{|\mathcal{F}_k^n|}$.

Corollaire 4 *[FGGZ07] Asymptotiquement (quand $k \to \infty$), les tautologies (classiques) sont intuitionnistes, c'est-à-dire :*

$$\lim_{k\to\infty} \frac{\mu_k^-(Int_k)}{\mu_k(Cl_k)} = 1.$$

Toujours en considérant comme seul connecteur l'implication, mais en autorisant maintenant des littéraux positifs ou négatifs, est-il toujours vrai que presque toutes les tautologies ont une structure très simple ? Nous allons montrer que c'est effectivement le cas.

On note $\tilde{\mathcal{T}}_k$ les tautologies dans ce système, sur les variables $\{x_1, \ldots, x_k\}$. Il est facile de démontrer que $\mu_k(\tilde{\mathcal{T}}_k)$ est bien défini.

Définition 8 *On définit les tautologies* simples $\tilde{\mathcal{S}}_k$ *comme l'union des deux familles suivantes :*

— *Les formules de la forme $A_1, \ldots, A_p \to \alpha$ où $A_i = \alpha$ pour un $i \in \{1, \ldots, p\}$; autrement dit, ces formules sont de la forme*

$$\ldots, \ell, \ldots \to \ell$$

pour un littéral ℓ.

— *Les formules de la forme $A_1, \ldots, A_p \to \alpha$ tel que A_i et A_j sont deux littéraux opposés pour $i, j \in \{1, \ldots, p\}$; ce sont les formules du type*

$$\ldots, \ell, \ldots, \bar{\ell}, \ldots \to \cdot$$

où $\bar{\ell}$ est la négation de ℓ (i.e. $\bar{\ell} = \bar{x}_i$ si $\ell = x_i$ et $\bar{\ell} = x_i$ si $\ell = \bar{x}_i$).

Théorème 27 *[FGGZ10] Dans le modèle des formules sur l'implication avec littéraux positifs et négatifs, asymptotiquement (quand $k \to \infty$), toutes les tautologies sont simples :*

$$\lim_{k\to\infty} \frac{\mu_k(\tilde{\mathcal{S}}_k)}{\mu_k(\tilde{\mathcal{T}}_k)} = 1.$$

Expansions valides d'une formule et fraction limite d'une fonction

On s'intéresse maintenant à la fraction limite d'une fonction booléenne quelconque. Pour cela, on commence par définir trois règles, appelées *règles d'expansion*, qui à partir d'une formule A, permettent d'obtenir une formule plus grande calculant la même fonction booléenne.

Définition 9 *Soit A une formule et B une sous-formule dont on note ν la racine.*

- *Une formule A' est appelée expansion valide de A par une tautologie au nœud ν si A' est obtenue en remplaçant B par $C \to B$ dans A, où C est une tautologie. Bien sûr A' calcule la même fonction que A puisque $[C \to B] = [B]$.*
- *Soit $\alpha \in \{x_1, \ldots, x_k\}$. Si remplacer B par $C \to B$ donne une formule A' calculant la même fonction que A* pour toute formule C de but α, *alors A' est appelée expansion valide du type "but α" au nœud ν.*
- *Soit $\alpha \in \{x_1, \ldots, x_k\}$. Si remplacer B par $C \to B$ donne une formule A' calculant la même fonction que A* pour toute formule C ayant une prémisse égale à α, *alors A' est appelée expansion valide du type "prémisse α" au nœud ν.*

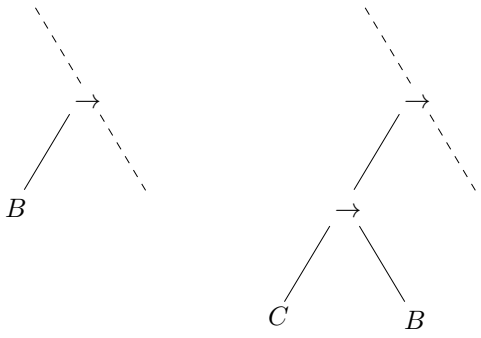

FIGURE 3.2 – Expansion valide avec la sous-formule C à la racine de B.

La figure 3.2 représente une formule après une expansion valide à la racine de B.

Définition 10 *Étant donné une formule A, on définit $E(A)$ comme l'ensemble des formules obtenues par une (unique) expansion (parmi les trois définies ci-dessus). Pour $\mathcal{A} \subseteq \mathcal{F}_k$, on pose $E(\mathcal{A}) = \bigcup_{A \in \mathcal{A}} E(A)$.*

Définition 11 *Étant donné une formule A, on définit $\lambda(A)$ comme le nombre de types différents d'expansions valides de A ; précisément, c'est le nombre de couples (ν, α) avec ν un nœud de A et $\alpha \in \{x_1, \ldots, x_k\}$ tels qu'une expansion de type "but α" est valide au nœud ν de A, plus le nombre de couples (ν, α) avec ν un nœud de A et $\alpha \in \{x_1, \ldots, x_k\}$ tels qu'une expansion de type "prémisse α" est valide au nœud ν de A, plus $2|A| - 1$ (ce qui compte le nombre d'expansions possibles par des tautologies, une en chacun des $2|A| - 1$ nœuds de la formule A).*

Pour une fonction booléenne f dépendant d'un nombre fini de variables de $\{x_i \mid i > 0\}$ et pouvant être calculée avec le connecteur de l'implication, on définit sa complexité $L(f)$ comme la taille de la plus petite formule (sur l'implication) la calculant.

Étant donné une fonction booléenne f, on note $\mathcal{M}_f$ l'ensemble des formules de taille minimale (c'est-à-dire de taille $L(f)$) pour f. On définit

$$\lambda(f) = \sum_{M \in \mathcal{M}_f} \lambda(M).$$

On peut montrer que $\lambda(f)$ ne dépend pas du nombre de variables k de l'espace ambiant.

Théorème 28 *[FGGZ07, FGGG12] La fraction limite de la fonction Vrai vérifie*

$$\mu_k(\mathit{Vrai}) = \frac{1}{k} + O\left(\frac{1}{k^2}\right).$$

Pour une fonction booléenne f différente de Vrai, presque tous les arbres calculant f sont obtenus à partir d'une unique expansion d'un arbre minimal de f :

$$\mu_k(f) \sim \mu_k(E(\mathcal{M}_f)).$$

En conséquence, la fraction limite de f est asymptotiquement (quand $k \to \infty$) égale à :

$$\mu_k(f) = \frac{\lambda(f)}{4^{L(f)}\ k^{L(f)+1}} + O\left(\frac{1}{k^{L(f)+2}}\right).$$

On peut donner les bornes suivantes sur l'entier $\lambda(f)$. Pour une fonction booléenne f différente de *Vrai* et qui a ℓ variables essentielles (c'est-à-dire qui dépend effectivement de ℓ variables) :

$$2(2L(f)-1)|\mathcal{M}_f| \leqslant \lambda(f) \leqslant (1+2\ell)(2L(f)-1)|\mathcal{M}_f|.$$

Le modèle des processus de branchement

On considère maintenant la distribution de probabilité induite sur les fonctions booléennes par un processus critique de Galton-Watson. Dans ce modèle, les probabilités qu'un nœud ait 0 ou 2 fils est égale à 1/2 et l'étiquetage des feuilles est uniforme indépendant dans $\{x_1, \ldots, x_k\}$. Il est connu qu'une formule est presque sûrement finie dans ce modèle [AN72].

Cette distribution de probabilités introduite dans [CFGG04] pour les formules construites sur les connecteurs *et* et *ou* peut être adaptée au cas des formules construites sur l'implication. Pour une formule A on obtient :

$$\pi_k(A) = \mathbb{P}(\text{structure de } A) \cdot \mathbb{P}(\text{étiquetage of } A) = \frac{1}{2^{2|A|-1}\ k^{|A|}}.$$

Dans ce modèle, remarquons que la probabilité $\pi_k(\mathcal{A})$ est bien définie pour un sous-ensemble quelconque de l'ensemble des formules $\mathcal{A}$. On définit la probabilité d'une fonction f comme $\pi_k(f) = \pi_k(\{A \in \mathcal{F}_k \mid [A] = f\}) = \sum_{[A]=f} \pi_k(A)$.

Proposition 19 *[FGGG12] La probabilité des tautologies satisfait :*

$$\pi_k(\mathit{Vrai}) = \frac{1}{2k} + O\left(\frac{1}{k^2}\right).$$

Pour une fonction booléenne f différente de Vrai,

$$\pi_k(f) = \frac{|\mathcal{M}_f|}{2^{2L(f)-1}k^{L(f)}} + O\left(\frac{1}{k^{L(f)+1}}\right).$$

Travaux ultérieurs

La comparaison entre logiques classique et intuitionniste a été étendue au système propositionnel complet par Genitrini et Kozic [GK12]. Des résultats précis entre complexité et probabilité des fonctions dans le cadre des connecteurs *et/ou* ont été donnés par Kozik [Koz08] puis généralisés par Genitrini et Mailler [GM14].

3.2 Formules équilibrées sur les connecteurs *et/ou*

Considérons une formule booléenne aléatoire sur les connecteurs *et* et *ou* qui soit équilibrée (c'est-à-dire que cette formule a toutes ses feuilles à la même hauteur). On se propose d'étudier la distribution des fonctions booléennes calculées par de telles formules aléatoires.

Distribution induite par les formules équilibrées

On considère des formules sur k variables $x_1, \ldots, x_k$, avec les connecteurs *et* and *ou*. On représente de telles formules par des arbres dont les feuilles sont étiquetées par les variables et les nœuds internes par les connecteurs logiques. On dit qu'une formule est *équilibrée* si toutes les feuilles sont à la même hauteur.

Soit $k > 0$, $p \in [0,1]$ et μ_0 une distribution de probabilité sur $H_0 = \{x_1, \ldots, x_k\}$. Pour tout $n \geqslant 1$, soit $H_n = \{h_1 \wedge h_2,\ h_1 \vee h_2 \mid h_1, h_2 \in H_{n-1}\}$, et μ_n la distribution de probabilité sur H_n définie par $\mu_n(h_1 \wedge h_2) = p\ \mu_{n-1}(h_1)\mu_{n-1}(h_2)$, et $\mu_n(h_1 \vee h_2) = (1-p)\ \mu_{n-1}(h_1)\mu_{n-1}(h_2)$. Ainsi, H_n est l'ensemble des formules équilibrées de profondeur n dont les nœuds internes sont étiquetés indépendamment par $\wedge$ et $\vee$ selon une loi de Bernoulli de paramètres p, et dont les feuilles sont étiquetées par les variables $x_1, \ldots, x_k$ selon la distribution de probabilité μ_0.

Toute formule booléenne définit de manière naturelle une fonction booléenne. On note $\mathcal{B}_k$ l'ensemble des fonctions booléennes sur les variables $x_1, ..., x_k$. La suite (μ_n) induit une suite de distributions de probabilité (π_n) sur $\mathcal{B}_k$ de la manière suivante : pour tout $f \in \mathcal{B}_k$,

$$\pi_n(f) = \sum_{\{h \in H_n \ \mid\ h \text{ définit } f\}} \mu_n(h).$$

Notre but est d'étudier l'existence d'une distribution limite à (π_n) et de la décrire.

On note $\prec$ l'ordre partiel (strict) sur $\{0,1\}^k$ défini par $(a_1, \ldots, a_k) \prec (b_1, \ldots, b_k)$ si $a_i \leqslant b_i$ pour tout i et $a \neq b$. Pour $a, b \in \{0,1\}^k$, on note $\inf\{a, b\}$ l'infinimum de a et b pour cet ordre. Étant donné une distribution de probabilité μ_0 sur $\{x_1, \ldots, x_k\}$, on définit le *poids* d'un point $a = (a_1, \ldots, a_k) \in \{0,1\}^k$ comme

$$\omega(a) = \mu_0(x_1).a_1 + \ldots + \mu_0(x_k).a_k.$$

Notons que $\omega(a)$ est un réel de l'intervalle $[0,1]$. Pour $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{R}^k$ et $\theta \in \mathbb{R}$, la *fonction seuil affine* $T_{\alpha,\theta}$ est la fonction booléenne sur $\{0,1\}^k$ définie par

$$T_{\alpha,\theta}(a) = 1 \ \Leftrightarrow\ \alpha_1.a_1 + \ldots + \alpha_k.a_k \geqslant \theta.$$

Théorème 29 *[FGG09] Soit $k \geqslant 1$, $p \in [0,1]$, et μ_0 une distribution de probabilité sur $\{x_1, \ldots, x_k\}$ telle que $\mu_0(x_i) > 0$ pour tout i. La suite de distributions (π_n) possède une limite, décrite comme il suit :*

- *Si $\wedge$ a une probabilité supérieure à $\vee$ ($p > 1/2$), le support de la distribution limite est réduit à $x_1 \wedge \ldots \wedge x_k$.*
- *De même, si $\vee$ a une probabilité supérieure à $\wedge$ ($p < 1/2$), le support est réduit à $x_1 \vee \ldots \vee x_k$.*
- *Si $\wedge$ et $\vee$ ont même probabilité ($p = 1/2$), la distribution limite suit la loi de $T_{\mu_0,U}$ où U est une variable aléatoire uniforme sur $[0,1]$.*

Dans le cas $p = 1/2$, la distribution limite est ainsi concentrée sur les fonctions seuil de la forme $\{T_{\mu_0,\theta} \mid \theta \in \mathbb{R}\}$. On peut décrire cette loi limite π de manière plus explicite : soit $\theta_0 = 0 < \theta_1 < \theta_2 < \cdots < \theta_s = 1$ les différents poids des points de $\{0,1\}^k$; pour $i \in \{1, \ldots, s\}$, $\pi(T_{\mu_0,\theta_i}) = \theta_i - \theta_{i-1}$.

La distribution limite dans le cas $p = 1/2$ a une interprétation géométrique naturelle. Soit $h_0, h_1, \ldots, h_s$ les différents hyperplans réels orthogonaux à μ_0 et intersectant les points de l'hypercube $\{0,1\}^k$, avec h_{i+1} juste au-dessus de h_i. La distribution limite π est concentrée sur les fonctions seuil définies par les hyperplans $h_1, \ldots, h_s$; de plus, la probabilité de h_i est proportionnelle à la distance euclidienne $\mathrm{d}(h_i, h_{i-1})$ (i.e. elle est égale à $\mathrm{d}(h_i, h_{i-1})/\mathrm{d}(h_0, h_s)$). Ceci est illustré par la figure 3.3 dans le cas de deux variables.

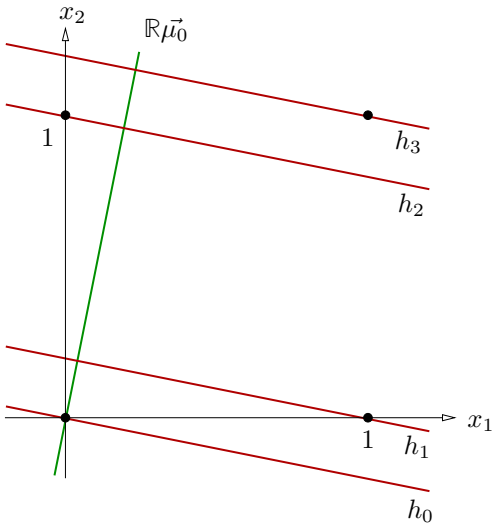

FIGURE 3.3 – Hyperplans définissant les fonctions seuil affines de la distribution limite de (π_n) pour les paramètres $p = \frac{1}{2}$, $\mu_0(x_1) = \frac{1}{5}$ et $\mu_0(x_2) = \frac{4}{5}$.

Le cas uniforme est une conséquence immédiate du théorème 29.

Corollaire 5 *[FGG09] Pour $p = 1/2$ et μ_0 uniforme sur $H_0 = \{x_1, \ldots, x_k\}$, la suite (π_n) admet une distribution limite qui est uniforme sur les k fonctions seuil $x_1 + x_2 + \ldots + x_k \geqslant i$, où $i \in \{1, \ldots, k\}$.*

C'est une conséquence facile du théorème 29 de considérer un modèle de formules où les feuilles peuvent être étiquetées par des fonctions booléennes quelconques. La suite de probabilités induite par les arbres *et/ou* équilibrés de profondeur croissante admet toujours une limite obtenue par substitution dans la distribution limite décrite ci-dessus. Un cas particulier celui où on étiquette les feuilles par des littéraux positifs et négatifs :

Corollaire 6 *[FGG09] Pour $p = 1/2$ et μ_0 uniforme sur $H_0 = \{x_1, \bar{x}_1, \ldots, x_k, \bar{x}_k\}$, la suite (π_n) a pour distribution limite la loi uniforme sur $\{vrai, faux\}$.*

Il est intéressant de noter que la distribution limite est concentrée sur un petit nombre de fonctions, au plus 2^k ; en particulier, ce nombre de fonctions est égal à k (resp. 2) dans le cas des littéraux positifs uniquement (resp. avec des littéraux positifs et négatifs). Ceci est à comparer au modèle des formules non équilibrées, où toutes les formules expressibles apparaissent dans la distribution limite.

Vitesse de convergence

On analyse la vitesse de convergence par rapport au nombre d'itérations, pour un nombre de variables et des valeurs de p et de μ_0 fixées. Soit

$$\|\pi_n - \pi\| = \max_{f \in \mathcal{B}_k} |\pi_n(f) - \pi(f)|.$$

Si $\|\pi_n - \pi\| = 2^{-\Theta(n)}$, on dit que la vitesse de convergence est *linéaire*. La vitesse de convergence est appelée *logarithmique* si $\|\pi_n - \pi\| = \Theta(1/n)$.

On suppose à nouveau que μ_0 satisfait $\mu_0(x_i) > 0$ pour tout $i \in \{1, \ldots, k\}$. Remarquons que dans le cas $k = 1$, toutes les expressions calculent la même fonction x_1. On suppose donc $k > 1$. Le cas $p \in \{0, 1\}$ correspond à l'utilisation d'un seul connecteur logique ; il est facile de voir que $\|\pi_n - \pi\| = 2^{-\Theta(2^n)}$ dans ce cas. On peut maintenant énoncer le résultat sur la vitesse de convergence dans les cas non triviaux.

Proposition 20 *[FGG09]*

- *Pour $p \notin \{0, 1/2, 1\}$, la vitesse de convergence de (π_n) est linéaire ;*
- *Pour $p = 1/2$, la vitesse de convergence de (π_n) est linéaire si tous les points de $\{0, 1\}^k$ ont des poids distincts ; sinon, elle est logarithmique.*

Il serait intéressant d'étudier la vitesse de convergence par rapport au nombre de variables. Cependant, nous ne pensons pas que la convergence serait assez rapide pour retrouver l'existence de petites formules monotones pour la fonction majorité. Remarquons qu'on ne connaît pas l'existence de petites formules monotones pour les fonctions seuil obtenues ici [Ser07].

Chapitre 4

Perspectives

Certaines perspectives de recherche directement reliées à des travaux présentés ont été données à la fin des sections correspondantes. Nous proposons quelques autres directions ci-dessous.

4.1 Bornes inférieures géométriques et calcul de polynômes

Il n'y a pas de bornes inférieures en $\omega(n \log n)$ pour des problèmes de géométrie algorithmique usuels. Les bornes inférieures plus fortes considèrent des problèmes avec beaucoup d'éléments géométriques (comme la borne en $\Omega(n^2)$ pour le problème du sac-à-dos [DL78, BO83] dont la restriction à la dimension n est l'union de 2^n hyperplans), ou sont obtenus sur des modèles de calcul faibles (par exemple la borne en $\Omega(n^{4/3})$ pour le problème de Hopcroft due à Erickson [Eri96]).

Cette limite en $n \log n$ est assez naturelle : on ne connaît pas non plus de borne inférieure en $\omega(n \log n)$ pour les polynômes de VNP [Raz09], alors que les problèmes géométriques classiques comme le problème de Hopcroft ou 3-Sum reviennent à décider des polynômes de la classe VP, une classe à priori bien plus petite. Cette remarque soulève deux questions.

Est-il possible de dépasser la borne $\Omega(n \log n)$ pour la version uniforme de VNP ? C'est ce que nous avons exploré dans l'article [FPdV14]. Même si cela semble être un problème difficile, il n'est pas exclu qu'il soit possible d'obtenir des bornes inférieures en $\Omega(n^k)$ pour ces polynômes (sous l'hypothèse GRH). Une piste à explorer est d'utiliser pour cela la puissance de la hiérarchie de comptage puisqu'il est possible de travailler sous l'hypothèse $\mathsf{VP} = \mathsf{VNP}$, qui implique $\mathsf{CH} = \mathsf{MA}$.

Une autre question est de savoir si on peut relier les bornes inférieures dans le modèle de Valiant et les bornes inférieures pour les problèmes géométriques. C'est le cas pour les problèmes géométriques qui sont l'union de zéros d'un petit nombre de polynômes, comme le problème suivant.

> 3-Sum
> *Entrée :* $x_1, x_2, \ldots, x_n \in \mathbb{R}$
> *Problème :* Existe-t-il i, j, k distincts tel que $x_i + x_j + x_k = 0$?

En effet, calculer le polynôme $\Pi_{i,j,k}(x_i + x_j + x_k)$ permet de décider 3-Sum. Mais cette relation ne tient plus pour les problèmes tels que le diamètre. La raison vient du fait que les tests du type "$x > 0$?" ne sont pas possibles dans le modèle de Valiant. En utilisant la construction de fractions rationnelles approximant la fonction signe due à Newman [New64], on peut montrer qu'une borne inférieure en $\omega(n \log n)$ pour une 2^{-n}-approximation du diamètre en dimension fixée donnerait une borne inférieure en $\omega(n \log n)$ pour la version uniforme de VP. Est-il possible

d'obtenir ce type de résultats pour la version exacte du diamètre ? Ce type de relation entre Valiant et le modèle de calcul Blum-Shub-Smale a été obtenu par Koiran et Perifel [KP09] pour de grandes classes de complexité. Leur méthode repose sur la simulation d'arbres de petite profondeur pour la localisation décrits par Grigoriev [Gri00]. Une piste à explorer est d'essayer de faire ce type de simulation pour les petites classes capturant les problèmes de géométrie algorithmique.

4.2 Bornes inférieures explicites : les modèles multilinéaires

Comme nous l'avons déjà mentionné, on ne connaît pas de borne inférieure en $\omega(n \log n)$ sur la taille des circuits généraux pour des polynômes à n variables de la classe VNP uniforme. Ainsi, une direction pour progresser sur les bornes inférieures consiste à considérer des modèles de calcul plus faibles, comme des circuits d'une nature particulière. C'est ce que nous avons fait à la section 1.5 en se concentrant sur les formules de profondeur 4.

Une autre restriction est celle des circuits *multilinéaires*, c'est-à-dire les circuits dont toutes les portes calculent un polynôme multilinéaire. Notons que la classe des polynômes multilinéaires contient des polynômes jouant un rôle important en complexité algébrique, comme le permanent et le déterminant. La recherche sur ces modèles multilinéaires a été très active ces dernières années après le résultat de Raz [Raz06] séparant formules et circuits multilinéaires. Ce résultat a été obtenu en montrant qu'une petite formule multilinéaire ne peut pas calculer de *polynôme de haut rang*. Un polynôme $P(\bar{x})$ est appelé de haut rang si, pour une partition aléatoire uniforme des variables $\bar{x} = \bar{y} \cup \bar{z}$ avec $|\bar{y}| = |\bar{z}| = |\bar{x}|/2$, la matrice des dérivées partielles du polynôme $P(\bar{y}, \bar{z})$ est de degré maximal (ou proche) avec grande probabilité. Un espoir serait de généraliser cela au cas des *branching programs*. En effet, notre intuition est que les *branching programs* multilinéaires ne peuvent pas calculer des polynômes de haut rang. Un tel résultat séparerait les *branching programs* multilinéaires des circuits.

Notons que les formules et *branching programs* arithmétiques multilinéaires ont été séparés [DMPY12]. En fait le premier résultat important obtenu sur les modèles multilinéaires est une borne inférieure super-polynomiale sur la taille des formules multilinéaires calculant le déterminant [Raz09]. Nous souhaitons poursuivre cette étude en étudiant la taille des *branching programs* multilinéaires calculant le déterminant (une tentative récente pour obtenir de telles bornes est présentée dans [Jan08]). Cette direction de recherche est particulièrement attractive puisque le déterminant peut être calculé par des *branching programs* taille polynomiale mais qui produisent des monômes non linéaires lors des calculs intermédiaires (voir par exemple Mahajan et Vinay [MV99]). En suivant l'approche de Raz, nous avons l'intention d'étudier la distribution du rang de la matrice des dérivées partielles du polynôme calculé par un *branching program* multilinéaire fixé (pour des partitions/projections des variables à définir convenablement). Un objectif ambitieux est de démontrer une borne inférieure super-polynomiale sur la taille des *branching programs* multilinéaires calculant le déterminant.

Il existe en fait deux notions de multilinéarité. Celle décrite ci-dessus, où chaque porte intermédiaire calcule un polynôme multilinéaire, en est la version *sémantique*. Une condition plus forte, appelée multilinéarité *syntaxique*, serait qu'aucune variable ne puisse être en commun dans les deux sous-arbres d'une porte de multiplication ; ou pour les *branching programs*, qu'une même variable n'apparaisse pas deux fois le long d'un chemin. Ces deux notions coïncident sur les formules, au sens où une formule sémantiquement multilinéaire peut être remplacée par une formule syntaxiquement multilinéaire de même taille et calculant le même polynôme. Il serait intéressant de comparer ces deux notions pour des circuits plus généraux.

4.3 Autour des problèmes du degré et de PosSLP

Le problème PosSLP défini dans [ABKPM09] est le suivant.

> PosSLP
> *Entrée :* Un circuit arithmétique calculant un entier à partir de l'unique constante -1
> *Problème :* L'entier calculé par ce circuit est-il positif ?

On rappelle que ce problème est dans la hiérarchie de comptage, et on peut lui réduire le problème de la somme des racines carrées défini à la section 1.2. Même si l'article [ABKPM09] fait un pas important en montrant que le problème PosSLP se situe dans une classe plus petite que PSPACE, la borne supérieure qui se trouve dans un niveau supérieur de la hiérarchie de comptage pourrait être encore améliorée (intuitivement, une borne supérieure en $\mathsf{PH}^{\mathsf{PP}}$ ne semble pas inaccessible, sachant qu'on sait tester si deux circuits calculent le même nombre par un algorithme probabiliste en temps polynomial). Étudier la distribution des nombres calculés par des circuits d'une taille donnée pourrait permettre de faire des progrès.

Par ailleurs, il est intrigant de n'avoir aucune autre borne inférieure que P pour DegSLP et PosSLP. On propose ci-dessous d'étudier une version particulière du problème DegSLP. Comme ce problème fait intervenir des questions combinatoires sur les *matchings*, on peut espérer mieux le comprendre que le cas général.

Cette question est de déterminer le degré du déterminant d'une matrice à coefficients dans $\mathbb{Z}[x]$. La difficulté potentielle vient du fait qu'on autorise des degrés exponentiels ; par exemple, les entrées de la matrice sont des polynômes creux. Un cas particulier déjà pertinent est le suivant.

> DegDet
> *Entrée :* Une matrice carrée A dont les entrées sont de la forme $x^{a_{i,j}}$, où x est une variable et les $a_{i,j}$ sont de entiers écrits en binaire
> *Problème :* Calculer le degré de $\det A$.

Comme le déterminant peut être calculé par un petit circuit, c'est un cas particulier de DegSLP. Notons que le calcul du degré du déterminant d'une matrice de polynômes est une question possédant un intérêt pratique [HS99] : montrer une borne supérieure sur la complexité de DegDet serait donc fructueux.

Remarquons que si on appelle D le degré maximum, sur toutes les permutations π, de la somme des $a_{i,\pi(i)}$, D est une borne supérieure sur le degré de $\det A$. Et on sait déterminer en temps polynomial si D est bien le degré de $\det A$ ou si la somme des monômes de degré maximal s'annule [Mur95]. Les instances potentiellement difficiles sont donc celles où les monômes de plus haut degré (D, $D-1$, etc.) s'annulent.

Travaux présentés

[CFGK03] A. Chistov, H. Fournier, L. Gurvits, and P. Koiran. Vandermonde matrices, NP-completeness, and transversal subspaces. *Foundations of Computational Mathematics*, 3(4) :421–427, 2003.

[CFKP08] A. Chistov, H. Fournier, P. Koiran, and S. Perifel. On the construction of a family of transversal subspaces over finite fields. *Linear Algebra Appl.*, 429(2-3) :589–600, 2008.

[FGG09] H. Fournier, D. Gardy, and A. Genitrini. Banlanced and/or trees and linear threshold functions. In *6th SIAM Workshop on Analytic Algorithmics and Combinatorics*, pages 51–57, 2009.

[FGGG12] H. Fournier, D. Gardy, A. Genitrini, and B. Gittenberger. The fraction of large random trees representing a given Boolean function in implicational logic. *Random Structures and Algorithms*, 40(3) :317–349, 2012.

[FGGZ07] H. Fournier, D. Gardy, A. Genitrini, and M. Zaionc. Classical and intuitionistic logic are asymptotically identical. In *CSL*, pages 177–193, 2007.

[FGGZ10] H. Fournier, D. Gardy, A. Genitrini, and M .Zaionc. Tautologies over implication with negative literals. *Mathematical Logic Quarterly*, 56(4) :338–396, 2010.

[FIV13] H. Fournier, A. Ismail, and A. Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *CoRR*, abs/1210.3323v3, 2013.

[FLMS14] H. Fournier, N. Limaye, G. Malod, and S. Srinivasan. Lower bounds for depth 4 formulas computing iterated matrix multiplication. In *STOC*, pages 128–135, 2014.

[FMM14] H. Fournier, G. Malod, and S. Mengel. Monomials in arithmetic circuits : Complete problems in the counting hierarchy. *Computational Complexity*. To appear. (Preliminary version in STACS 2012).

[FPdV14] H. Fournier, S. Perifel, and R. de Verclos. On fixed-polynomial size circuit lower bounds for uniform polynomials in the sense of valiant. *Information and Compuatation*. To appear. (Preliminary version in MFCS 2013).

[FV07] H. Fournier and A. Vigneron. A tight lower bound for computing the diameter of a 3D convex polytope. *Algorithmica*, 49(3) :245–257, 2007. (Preliminary version in LATIN 2006).

[FV11] H. Fournier and A. Vigneron. Fitting a step function to a point set. *Algorithmica*, 60(1) :95–109, 2011. (Preliminary version in ESA 2008).

[FV13] H. Fournier and A. Vigneron. A deterministic algorithm for fitting a step function to a weighted point-set. *Inf. Process. Lett.*, 113(3) :51–54, 2013.

Bibliographie

[ABKPM09] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the Complexity of Numerical Analysis. *SIAM J. Comput.*, 38(5) :1987–2006, 2009.

[ABO99] E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2) :99–126, 1999.

[AK95] Edoardo Amaldi and Viggo Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theor. Comput. Sci.*, 147(1&2) :181–210, 1995.

[AKS83] Miklós Ajtai, János Komlós, and Endre Szemerédi. Sorting in $c \log n$ parallel sets. *Combinatorica*, 3(1) :1–19, 1983.

[All04] E. Allender. Arithmetic circuits and counting complexity classes. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 33–72. Dept. Math., Seconda Univ. Napoli, Caserta, 2004.

[AN72] K. B. Athreya and P. E. Ney. *Branching Processes*. Springer, 1972.

[AV08] Manindra Agrawal and V. Vinay. Arithmetic circuits : A chasm at depth four. In *FOCS*, pages 67–75. IEEE Computer Society, 2008.

[Ban90] Hans-Jürgen Bandelt. Recognition of tree metrics. *SIAM J. Discrete Math.*, 3(1) :1–6, 1990.

[BO83] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 80–86, 1983.

[BP05] A. Brodsky and N. Pippenger. The boolean functions computed by random boolean formulas or how to grow the right function. *Random Structures and Algorithms*, 27 :490–519, 2005.

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, second edition, 2006.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theor. Comput. Sci.*, 22 :317–330, 1983.

[BS00] M. Bonk and O. Schramm. Embeddings of Gromov hyperbolic spaces. *Geometric and Functional Analysis*, 10 :266–306, 2000.

[BT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 301–309, 1988.

[Bür00] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2000.

[Bür09] P. Bürgisser. On Defining Integers And Proving Arithmetic Circuit Lower Bounds. *Computational Complexity*, 18(1) :81–103, 2009.

[CCL12] Nathann Cohen, David Coudert, and Aurélien Lancin. Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Research Report RR-8074, INRIA, September 2012.

[CD06] F. Chazal and S. Das. An efficient algorithm for fitting rectilinear x-monotone curve to a point set in a plane. Technical report, August 2006. Available at `http://math.u-bourgogne.fr/IMB/chazal/publications.htm`.

[CDE+12] Victor Chepoi, Feodor F. Dragan, Bertrand Estellon, Michel Habib, Yann Vaxès, and Yang Xiang. Additive spanners and distance and routing labeling schemes for hyperbolic graphs. *Algorithmica*, 62(3–4) :713–732, 2012.

[CE07] Victor Chepoi and Bertrand Estellon. Packing and covering *delta* -hyperbolic spaces by balls. In *APPROX-RANDOM*, pages 59–73, 2007.

[CFGG04] B. Chauvin, P. Flajolet, D. Gardy, and B. Gittenberger. And/Or trees revisited. *Combinatorics, Probability and Computing*, 13(4-5) :475–497, July-September 2004.

[CFHM12] Wei Chen, Wenjie Fang, Guangda Hu, and Michael W. Mahoney. On the hyperbolicity of small-world networks and tree-like graphs. In *Proceedings of the 23rd International Symposium on Algorithms and Computation (ISAAC)*, pages 278–288, 2012.

[Chi85] Alexander Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *Proc. 5th International FCT Conference*, volume 199 of *Lecture Notes in Computer Science*, pages 9–13. Springer-Verlag, 1985.

[CKM82] R. Chandrasekaran, S. N. Kabadi, and K. Murty. Some NP-complete problems in linear programming. *Operations Research Letters*, 1 :101–103, 1982.

[CNS96] Jin-Yi Cai, Ashish V. Naik, and D. Sivakumar. On the existence of hard sparse sets under weak reductions. In *STACS*, pages 307–318, 1996.

[Col87] Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34(1) :200–208, 1987.

[CS89] K. Clarkson and P. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4 :387–421, 1989.

[CW09] Danny Z. Chen and Haitao Wang. Approximating points by a piecewise linear function : I. In *ISAAC*, pages 224–233, 2009.

[CW11] Danny Z. Chen and Haitao Wang. Efficient algorithms for the weighted k-center problem on a real line. In *ISAAC*, pages 584–593, 2011.

[DBM01] José Miguel Díaz-Báñez and Juan A. Mesa. Fitting rectilinear polygonal curves to a set of points in the plane. *European Journal of Operational Research*, 130(1) :214–222, 2001.

[DGH98] Martin E. Dyer, Peter Gritzmann, and Alexander Hufnagel. On the complexity of computing mixed volumes. *SIAM J. Comput.*, 27(2) :356–400, 1998.

[DL78] David P. Dobkin and Richard J. Lipton. A lower bound of the $\frac{1}{2}n^2$ on linear search programs for the knapsack problem. *J. Comput. Syst. Sci.*, 16(3) :413–417, 1978.

[DMPY12] Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff. Separating multilinear branching programs and formulas. In *STOC*, pages 615–624, 2012.

[DP09] Ran Duan and Seth Pettie. Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms (SODA)*, pages 384–391, 2009.

[Dua14] Ran Duan. Approximation algorithms for the gromov hyperbolicity of discrete metric spaces. In *LATIN*, pages 285–293, 2014.

[Eri95] J. Erickson. On the relative complexities of some geometric problems. In *Proceedings of the 7th Canadian Conference on Computational Geometry*, pages 85–90, 1995.

[Eri96] Jeff Erickson. New lower bounds for Hopcroft's problem. *Discrete & Computational Geometry*, 16(4) :389–418, 1996.

[Eri99] Jeff Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28(4) :1198–1214, 1999.

[FJ84] G. N. Frederickson and D. B. Johnson. Generalized selection and ranking : Sorted matrices. *SIAM Journal on Computing*, 13(1) :14–30, 1984.

[Fre91] G. N. Frederickson. Optimal algorithms for tree partitioning. In *Proc. 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 168–177, 1991.

[FS09] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[FSW09] Lance Fortnow, Rahul Santhanam, and Ryan Williams. Fixed-polynomial size circuit bounds. In *IEEE Conference on Computational Complexity*, pages 19–26, 2009.

[GBT84] H. N. Gabow, J. L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. In *Proc. 16th Annual ACM Symposium on Theory of Computing*, pages 135–143, 1984.

[GdlH90] E. Ghys and P. de la Harpe. *Sur les groupes hyperbolic d'apres Mikhael Gromov*, volume 83 of *Progress in Mathematics*. Birkhäuser, 1990.

[GK87] Dima Grigoriev and Marek Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC (extended abstract). In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 166–172, 1987.

[GK12] Antoine Genitrini and Jakub Kozik. In the full propositional logic, 5/8 of classical tautologies are intuitionistically valid. *Ann. Pure Appl. Logic*, 163(7) :875–887, 2012.

[GKKS13] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the chasm at depth four. In *Proceedings of the Conference on Computational Complexity (CCC)*, 2013.

[GKS90] Dima Grigoriev, Marek Karpinski, and Michael F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. Comput.*, 19(6) :1059–1063, 1990.

[GKS94] Dima Grigoriev, Marek Karpinski, and Michael F. Singer. Computational complexity of sparse rational interpolation. *SIAM J. Comput.*, 23(1) :1–11, 1994.

[GM14] Antoine Genitrini and Cécile Mailler. Equivalence classes of random boolean trees and application to the catalan satisfiability problem. In *LATIN*, pages 466–477, 2014.

[GO95] Anka Gajentaan and Mark H. Overmars. On a class of o(n2) problems in computational geometry. *Comput. Geom.*, 5 :165–185, 1995.

[Gri88] Dima Grigoriev. Complexity of deciding tarski algebra. *J. Symb. Comput.*, 5(1/2) :65–108, 1988.

[Gri00] Dima Grigoriev. Topological complexity of the range searching. *J. Complexity*, 16(1) :50–53, 2000.

[Gro87] M. Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *math. sci. res. ins. publ.*, pages 75–263. Springer, 1987.

[GS07a] S. Guha and K. Shim. A note on linear time algorithms for maximum error histograms. *IEEE Transactions on Knowledge and Data Engineering*, 19(7) :993–997, 2007.

[GS07b] Sudipto Guha and Kyuseok Shim. A note on linear time algorithms for maximum error histograms. *IEEE Trans. Knowl. Data Eng.*, 19(7) :993–997, 2007.

[GS09] S. Garg and E. Schost. Interpolation of polynomials given by straight-line programs. *Theor. Comput. Sci.*, 410(27-29) :2659–2662, 2009.

[HO02] L. A. Hemaspaandra and M. Ogihara. *The complexity theory companion*. Springer Verlag, 2002.

[HS99] D. Henrion and M. Sebek. Improved polynomial matrix determinant computation. *IEEE Transactions on Circuits and Systems I*, 46(10) :1307–1308, 1999.

[Jan08] Maurice J. Jansen. Lower bounds for syntactically multilinear algebraic branching programs. In *MFCS*, pages 407–418, 2008.

[JS11] M. Jansen and R. Santhanam. Permanent Does Not Have Succinct Polynomial Size Arithmetic Circuits of Constant Depth. In *ICALP*, pages 724–735, 2011.

[JS12] Maurice J. Jansen and Rahul Santhanam. Stronger lower bounds and randomness-hardness trade-offs using associated algebraic complexity classes. In *STACS*, pages 519–530, 2012.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3) :40–56, 1982.

[Kay12] Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 19 :81, 2012.

[KBvdG11] J. Kwisthout, H. L. Bodlaender, and L. C. van der Gaag. The complexity of finding kth most probable explanations in probabilistic networks. In *SOFSEM*, pages 356–367, 2011.

[Kha95] Leonid Khachiyan. On the complexity of approximating extremal determinants in matrices. *J. Complexity*, 11(1) :138–153, 1995.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, 13 :1–46, 2004.

[KL06] Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *FOCS*, pages 119–132, 2006.

[KLSS14] Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An exponential lower bound for homogeneous depth four arithmetic formulas. In *STOC*, 2014.

[Knu98] Donald E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley Professional, May 1998.

[Koi96] Pascal Koiran. Hilbert's Nullstellensatz is in the polynomial hierarchy. *J. Complexity*, 12(4) :273–286, 1996.

[Koi12] Pascal Koiran. Arithmetic circuits : The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448 :56–65, 2012.

[Koz08] Jakub Kozik. Subcritical pattern languages for and/or trees. In *Fifth Colloquium on Mathematics and Computer Science*, Discrete Math. Theor. Comput. Sci. Proc., AI, pages 437–448. Assoc. Discrete Math. Theor. Comput. Sci., Nancy, 2008.

[KP07] P. Koiran and S. Perifel. The complexity of two problems on arithmetic circuits. *Theor. Comput. Sci.*, 389(1-2) :172–181, 2007.

[KP09] Pascal Koiran and Sylvain Perifel. VPSPACE and a transfer theorem over the reals. *Computational Complexity*, 18(4) :551–575, 2009.

[KP11] P. Koiran and S. Perifel. Interpolation in Valiant's Theory. *Computational Complexity*, pages 1–20, 2011.

[KS11] N. Kayal and C. Saha. On the Sum of Square Roots of Polynomials and related problems. In *IEEE Conference on Computational Complexity*, pages 292–299, 2011.

[KSM07] Panagiotis Karras, Dimitris Sacharidis, and Nikos Mamoulis. Exploiting duality in summarization with deterministic guarantees. In *KDD*, pages 380–389, 2007.

[KSS13] Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. *Electronic Colloquium on Computational Complexity (ECCC)*, 20 :91, 2013.

[KSS14] Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *STOC*, 2014.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *FOCS*, pages 2–10, 1990.

[Lip75] Richard J. Lipton. Polynomials with 0-1 coefficients that are hard to evaluate. In *FOCS*, pages 6–10, 1975.

[Liu10] Jin-Yi Liu. A randomized algorithm for weighted approximation of points by a step function. In *COCOA (1)*, pages 300–308, 2010.

[LM08] Mario A. Lopez and Yan Mayster. Weighted rectilinear approximation of points in the plane. In *LATIN*, pages 642–653, 2008.

[LS97] H. Lefmann and P. Savický. Some typical properties of large And/Or Boolean formulas. *Random Structures and Algorithms*, 10 :337–351, 1997.

[Mat93] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(2) :157–182, 1993.

[Mei93] Stefan Meiser. Point location in arrangements of hyperplanes. *Inf. Comput.*, 106(2) :286–303, 1993.

[Mey84] Friedhelm Meyer auf der Heide. A polynomial linear search algorithm for the n-dimensional knapsack problem. *J. ACM*, 31(3) :668–676, 1984.

[Mey88] Friedhelm Meyer auf der Heide. Fast algorithms for N-dimensional restrictions of hard problems. *J. ACM*, 35(3) :740–747, 1988.

[ML06] Y. Mayster and M. A. Lopez. Approximating a set of points by a step function. *Journal of Visual Comunication and Image Represententation*, 17(6) :1178–1189, 2006.

[MP08] G. Malod and N. Portier. Characterizing Valiant's algebraic complexity classes. *J. Complexity*, 24(1) :16–38, 2008.

[MRS12] M. Mahajan, B. V. Raghavendra Rao, and K. Sreenivasaiah. Identity testing, multilinearity testing, and monomials in read-once/twice formulas and branching programs. In *MFCS*, pages 655–667, 2012.

[MS93] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete and Computational Geometry*, 10(2) :215–232, 1993.

[MSS12] J. Mittmann, N. Saxena, and P. Scheiblechner. Algebraic Independence in Positive Characteristic – A p-Adic Calculus. *ArXiv e-prints*, February 2012.

[MTZ00] M. Moczurad, J. Tyszkiewicz, and M. Zaionc. Statistical properties of simple types. *Mathematical Structures in Computer Science*, 10(5) :575–594, 2000.

[Mul86] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. In *STOC*, pages 338–339, 1986.

[Mur95] Kazuo Murota. Computing the degree of determinants via combinatorial relaxation. *SIAM J. Comput.*, 24(4) :765–796, 1995.

[MV99] Meena Mahajan and V. Vinay. Determinant : Old algorithms, new insights. *SIAM J. Discrete Math.*, 12(4) :474–490, 1999.

[MW07] P. McKenzie and K. W. Wagner. The complexity of membership problems for circuits over sets of natural numbers. *Computational Complexity*, 16(3) :211–244, 2007.

[New64] D. J. Newman. Rational approximation to $|x|$. *Michigan Math. J.*, 11(1) :11–14, 1964.

[NW97] Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3) :217–234, 1997.

[Pat90] Mike Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5(1) :65–92, 1990.

[PS85] F. Preparata and I. Shamos. *Computational geometry : An introduction*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 2nd edition, 1985.

[PVW94] J. B. Paris, A. Vencovská, and G. M. Wilmers. A natural prior probability distribution derived from the propositional calculus. *Annals of Pure and Applied Logic*, 70 :243–285, 1994.

[Ram01] E. Ramos. An optimal deterministic algorithm for computing the diameter of a three-dimensional point set. *Discrete and Computational Geometry*, 26 :233–244, 2001.

[Raz06] Ran Raz. Separation of multilinear circuit and formula size. *Theory of Computing*, 2(1) :121–135, 2006.

[Raz09] R. Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *Journal of the ACM (JACM)*, 56(2) :8, 2009.

[Raz10] Ran Raz. Elusive functions and lower bounds for arithmetic circuits. *Theory of Computing*, 6(1) :135–177, 2010.

[RBG01] Lajos Rónyai, László Babai, and Murali K. Ganapathy. On the number of zero-patterns of a sequence of polynomials. *Journal of the American Mathematical Society*, 14(3) :717–735, 2001.

[RV02] Marie-Françoise Roy and Nicolai Vorobjov. The complexification and degree of a semi-algebraic set. *Mathematische Zeitschrift*, 239(1) :131–142, 2002.

[San09] Rahul Santhanam. Circuit lower bounds for merlin–arthur classes. *SIAM J. Comput.*, 39(3) :1038–1061, 2009.

[Sch78] Claus-Peter Schnorr. Improved lower bounds on the number of multiplications/divisions which are necessary of evaluate polynomials. *Theor. Comput. Sci.*, 7 :251–261, 1978.

[Sch79] A. Schönhage. On the Power of Random Access Machines. In *ICALP*, pages 520–529, 1979.

[Ser07] R. A. Servedio. Every linear threshold function has a low-weight approximator. *Computational Complexity*, 16(2) :180–209, 2007.

[Str74] Volker Strassen. Polynomials with rational coefficients which are hard to compute. *SIAM J. Comput.*, 3(2) :128–149, 1974.

[Str13] Yann Strozecki. On enumerating monomials and other combinatorial structures by polynomial interpolation. *Theory Comput. Syst.*, 53(4) :532–568, 2013.

[Tav13] Sébastien Tavenas. Improved bounds for reduction to depth 4 and 3. In *Mathematical Foundations of Computer Science (MFCS)*, 2013.

[Tiw92] Prasoon Tiwari. A problem that is easier to solve on the unit-cost algebraic RAM. *J. Complexity*, 8(4) :393–397, 1992.

[Tod92] S. Toda. Classes of Arithmetic Circuits Capturing the Complexity of Computing the Determinant. *IEICE Transactions on Information and Systems*, E75-D :116–124, 1992.

[Tor88] J. Torán. Succinct Representations of Counting Problems. In *AAECC*, pages 415–426, 1988.

[Tor91] J. Torán. Complexity Classes Defined by Counting Quantifiers. *J. ACM*, 38(3) :753–774, 1991.

[Val79a] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3) :410–421, 1979.

[Val79b] L.G. Valiant. Completeness classes in algebra. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 249–261. ACM, 1979.

[Val84] L. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5 :363–366, 1984.

[Ven91] H. Venkateswaran. Properties that Characterize LOGCFL. *J. Comput. Syst. Sci.*, 43(2) :380–404, 1991.

[Vin05] N. V. Vinodchandran. A note on the circuit complexity of PP. *Theor. Comput. Sci.*, 347(1-2) :415–418, 2005.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra (2. ed.)*. Cambridge University Press, 2003.

[Wag86] K. W. Wagner. The Complexity of Combinatorial Problems with Succinct Input Representation. *Acta Informatica*, 23(3) :325–356, 1986.

[Wan02] D. P. Wang. A new algorithm for fitting a rectilinear x-monotone curve to a set of points in the plane. *Pattern Recognition Letters*, 23(1-3) :329–334, 2002.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *STOC*, pages 887–898, 2012.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654, 2010.