Rapport de stage : Réductions dans le modèle $(\mathbb{R}, +, -, \leq, 0, 1)$

Yann Strozecki

Maître de stage : Peter Bürgisser
Assisté de Peter Scheiblechner et Martin Lotz

25 décembre 2007

Table des matières

Résumé

On va, dans ce rapport, essayer de décrire l'état des connaissances en complexité pour le modèle $(\mathbb{R}, +, -, \leq, 0, 1)$ et de trouver des problèmes complets naturels pour réduction polynômiale dans différentes classes. On s'intéressera plus particulièrement au problème du voyageur de commerce et à la classe $D\sharp P_{add}$, notamment le problème de la complétude pour réduction polynômiale de $\sharp TSP_{\mathbb{R}}$ dans $D\sharp P_{add}$, question posée dans un article de Bürgisser [3]. On va aussi souligner les difficultés qui se posent pour obtenir des résultats fins dans les classes NP_{add} et $D\sharp P_{add}$.

Introduction :
Au cours de ce stage j'ai étudié de deux sujets, dans ce rapport je traite du deuxième et j'ai joins un résumé sur le premier en annexe. Il s'agit dans les deux cas de complexité et de mathématique théorique. Dans l'annexe j'étudie la complexité de la multiplication des matrices et les mathématiques, ici la théorie de la représentation, nous aide à obtenir des résultats de complexité. Dans le deuxième cas, c'est à dire l'étude des classes NP, $D\sharp P$ pour des machines BSS additives, il s'agit au contraire d'utiliser des outils de théorie de la complexités pour étudier des problèmes de nature mathématique, géométrique ou dans cette étude plutôt combinatoire.
Dans le résumé en annexe sur les matrices aucune démonstration n'est donnée, mais dans le rapport principal elles sont fournies dans la mesure du possible, bien qu'on devrait pouvoir le lire en sautant le texte qui suit **Preuve :** et le comprendre.

1 Définitions

Pour que cette étude ait un sens il faut d'abord définir ce qu'est une machine réelle. Il existe plusieurs modèles de calcul équivalents mais on va se contenter des deux principaux qui sont les machines BSS et les familles de circuit. Pour les connaissances qui peuvent être nécessaires sur les modèles se reporter au cours de logique et complexité [12].

Définition 1. Un problème est un sous-ensemble d'un langage. On dit aussi souvent problème pour faire référence à la question de l'appartenance d'un mot à un problème.

1.1 Les modèles de calcul

Le modèle BSS :
La définition suivante est fortement inspirée du livre Complexity and Real Computation [1], chapitre 2 et 3. Ce modèle de calcul par machine BSS ressemble à celui défini par les machines de Turing classiques, mais avec des machines qui manipuleraient des réels.
Une machine BSS est un graphe dirigé connexe avec 5 types de sommet : entrée, sortie, calcul, test, déplacement. Les sommets de calcul sont étiquetés par l'opération qu'ils réalisent, les sommets de déplacement par gauche ou droite et les sommets de test par une relation. Elle travaille sur un ruban biinfini ou chaque case peut prendre une valeur du modèle (réelle dans les cas qui nous intéresserons), et elle agit de la manière suivante :

- Un sommet de calcul lit la valeur des deuxième et troisème cases du ruban et écrit le résultat de l'application de la fonction dans la première case.

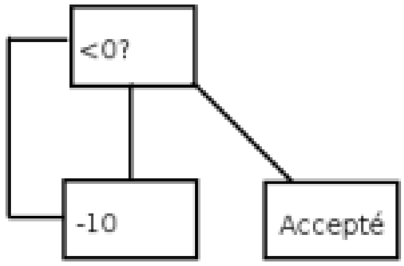

FIG. 1 – Une machine qui est toujours d'accord

- Un sommet de déplacement décale toute les cases d'un cran à gauche ou à droite selon qu'il soit étiqueté droite ou gauche.
- Un sommet de test lis les deux premières cases et selon que leur valeur soit en relation ou non envoie vers deux sommets différents.

Tout les sommets qui ne sont pas de test n'ont qu'un seul successeur. Les sommets d'entrée et de sortie se trouvent naturellement au début et à la fin du graphe, et servent à copier l'entrée sur le ruban et à extraire du ruban la réponse à la fin.

La machine fonctionne étape par étape en exécutant d'abord le sommet d'entrée, puis son successeur et ainsi de suite jusqu'à ce qu'elle arrive sur un sommet de sortie, elle a alors fini son calcul. On peut pour simplifier étiqueter les sommets de calcul par des combinaisons polynômiales des opérations de base du modèle. Dans notre étude les sommets de test vont en faite tester soit l'égalité soit l'infériorité d'un polynôme à 0. La bonne compréhension de ces machines est particulièrement important pour une démonstration ultérieure où l'on simule le fonctionnement d'une telle machine.

On dit qu'une machine calcule y pour une entrée x si on peut lire y sur le ruban après qu'elle ait fonctionné sur un ruban contenant x, en fait une machine calcul une fonction de $\mathbb{R}^\infty \to \mathbb{R}^\infty$. On dit qu'une machine résous un problème, si elle calcule la fonction indicatrice du problème.

Famille de circuit :
Pour une description plus détaillée de ce modèle de calcul voir [12] et [10]. Il existe un modèle alternatif,dans lequel on décrit une 'machine' par une famille de circuits donnée par une machine de Turing. Un circuit est un graphe connexe sans cycle orienté, dont les sommets sont étiquetés par

- par des variables, des constantes du modèle ou des paramètres pris dans le corps dans lequel on travail, ici $\mathbb{R}$, si ce sont des sommets d'entrée .
- par des numéros d'identification si ce sont des sommets de sortie.
- par des fonctions du modèle (ici des opérations arithmétiques), ou par des relations (ici égalité ou comparaison)

Les sommets étiquetés par une fonction prennent un certain nombre d'entrée et ont une arrête de sortie renvoyant le résultat de la fonction. Les sommets étiquetés par une relation prennent deux entrées et renvoie 1 si elles sont

comparables, 0 sinon. Pour calculer avec un circuit il suffit de le parcourir en réalisant les opérations liées aux portes. On dit que qu'un circuit calcule y pour une entrée x, si en effectuant les calculs liés au circuit on peut lire y sur les sommets de sortie.
On dit qu'une famille de circuit (donnée par un nombre de fini de constante et une machine de de Turing classique) décide un problème Π si pour tout x, la machine de Turing produit un circuit qui ne dépends que de la taille de x et des constantes, et ce circuit calcule 1 pour $x \in \Pi$, 0 sinon.

1.2 Généralités sur les classes de complexité

Toutes les définitions suivantes sont indépendantes du modèle. Une entrée est un mot fini, appartenant à $\mathbb{R}^\infty$ dans notre cas, sa taille est son nombre de "lettres" dans $\mathbb{R}$. On notera parfois la taille de x, $|x|$.

Définition 2 (Classe P)**.** On dit qu'un problème est dans P si il existe un machine qui décide de l'appartenance de x à ce problème en un nombre d'étape inférieur à un polynôme en $|x|$. Pour les familles de circuit il s'agit aussi de borner le temps d'exécution de la machine de Turing qui fournit le circuit pour une taille n de l'entrée par un polynôme en n, ce qui borne également le nombre de porte de ces circuit de manière polynômiale.

Maintenant qu'on a introduit P, on peut définir NP !

Définition 3 (Classe NP)**.** On dit qu'un problème Π est dans NP, si

$$x \in \Pi \Leftrightarrow \exists y, (x, y) \in L$$

avec $L \in P$ et y de taille polynômiale en x. On définit DNP en ajoutant la contrainte $y \in \{0,1\}^{p|x|}$ où p est un polynôme.

Bien sûr, dans le cas classique (mot de $\{0,1\}^\infty$) DNP est la même chose que NP. On appelle souvent y un témoin pour x, car c'est souvent une 'solution' à une instance d'un problème donné par x qu'on vérifie ensuite de manière polynômiale.

On peut aussi définir NP comme l'ensemble des problèmes résolus de manière non déterministe par une machine BSS fonctionnant en temps polynômial.

Définition 4 (Classe $\sharp P$)**.** On dit qu'une fonction f est dans $\sharp P$ si elle peut se définir comme $f(x) = |\{y|(x, y) \in L\}|$ où y est polynômial en $|x|$ et $L \in P$.

De même f est dans $D\sharp P$ si $f(x) = |\{y \in \{0,1\}^{p|x|}|(x, y) \in L\}|$ où y est polynômial en $|x|$ et $L \in P$.

En fait les fonction de $\sharp P$ (respectivement $D\sharp P$) comptent les bons témoins d'un problème NP (respectivement DNP) pour un x donné, on parlera donc de problèmes qui 'comptent'. Ces fonctions dépendent en fait du problème polynômial et du polynôme associé à un problème NP. On peut en avoir plusieurs pour un même problème NP mais sauf précision à un problème NP est associé un problème polynômial et un polynôme 'habituel' ce qui définit une version qui compte 'habituelle'.

Définition 5 (Partie booléenne)**.** On appelle partie booléenne d'un problème Π de décision son intersection avec $\{0,1\}^\infty$, et on le notera $BP(\Pi)$. Si C est une classe on notera $BP(C)$ la classe des parties booléennes des problèmes de C.

Définition 6 (Machine sans paramètre)**.** On peut distinguer une sous-classe de NP, celle des problèmes dont les machines n'utilisent pas d'autres constantes que 1 et 0, on notera cette classe NP^0.

$TSP_\mathbb{R}$ par exemple appartient à NP^0_{add}, car pour une instance x et un témoin y qui indique le chemin dans le graphe défini par x, il n'y a qu'à faire la somme des poids des arrêtes qui constituent ce chemin pour tester (x,y) et ceci s'effectue par une machine polynômiale sans constante.

Définition 7 (Réduction polynômiale)**.** Soient deux problèmes Π et Π', on dit que Π se réduit polynômialement à Π' si il existe une fonction f calculable en temps polynômiale telle que $x \in \Pi \Leftrightarrow f(x) \in \Pi'$. On dira parfois que Π se réduit à Π' dans P_{add} pour dire qu'il s'agit d'une réduction dans la classe P_{add}. De la même manière on dit que la fonction F_1 se réduit polynômialement à F_2 si il existe f calculable en temps polynômial telle que $F_1(x) = F_2(f(x))$.

Définition 8 (Réduction Turing)**.** Soient deux problèmes Π et Π', on dit que Π se réduit Turing à Π' si il existe une machine polynômiale, qui décide Π avec oracle Π'.

Naturellement réduction polynômiale implique réduction Turing, mais l'inverse n'est pas vrai.

Définition 9 (Complétude)**.** Si tous les problèmes appartenant à une classe C se réduisent polynômialement (respectivement Turing) à Π et que $\Pi \in C$, on dit que Π est C complet pour réduction polynômiale (respectivement Turing).

1.3 Étude de la complexité dans le modèle $(\mathbb{R}, +, -, \leq, 0, 1)$

Dans la suite on utilisera principalement le modèle $(\mathbb{R}, +, -, \leq, 0, 1)$, c'est à dire qu'on peut calculer et tester des polynômes linéaires,avec des petits entiers comme coefficient. Dans ce modèle les tests sont fait sur la relation inférieur ou égale à zéro. Mais on peut aussi tester l'égalité à zéro en testant aussi que l'opposé est inférieur ou égal à zéro, et en combinant on obtient une comparaison stricte.

On utiliseras l'indice *add* pour préciser qu'on est dans ce modèle et on rajoutera en exposant = si on ne fait de test que sur des égalités, c'est à dire qu'on est dans le modèle $(\mathbb{R}, +, -, =, 0, 1)$. Pour le modèle classique (booléen) on utiliseras le nom des classes sans indice, ni exposant.

1.4 Liste des problémes étudiés

1. SAT : Etant donné une formule propositionnelle Φ, existe-il une affectation booléenne des variables qui la rendre vraie.
2. $2SAT$:Etant donné une formule propositionnelle Φ dont chaque clause ne contient que deux variables, existe-il une affectation booléenne des variables qui la rendre vraie.
3. $CSAT$(Circuit Satisfaction) :Etant donné un circuit arithmétique, existe-il une entrée qui satisfasse le circuit.
4. $BCSAT$(Boolean Circuit Satisfaction) :Etant donné un circuit arithmétique, existe-il une entrée booléenne qui satisfasse le circuit.
5. BIP(Boolean Inequality Problem) :Etant donné un système d'équations et d'inéquations linéaires sans constante, dont les coefficients sont réels, existe-il une affectation booléenne des variables qui satisfassent le système.
6. BEP(Boolean Equality Problem) :Etant donné un système d'égalités et d'inégalités linéaires sans constante, dont les coefficients sont réels, existe-t-il une affectation booléenne des variables qui satisfassent le système.
7. $KNAP$(Knapsack) :Etant donné un poids w et des coefficients λ_i existe il une affection booléenne des variables x_i telle que $\sum_i xi\lambda_i = w$.
8. $PM_{\mathbb{R}}$(Perfect Matching) : Etant donné un graphe biparti G avec des poids réels sur les arrêtes et un poids $w \in \mathbb{R}$, existe-t-il un couplage parfait de poids au plus w.
9. $TSP_{\mathbb{R}}$(Traveling Salesman Problem) : Etant donné un graphe complet avec des poids réels sur les arrêtes et un poids $w \in \mathbb{R}$, existe-t-il un cycle hamiltonien de poids au plus w.
10. $TSP_{\mathbb{R}}(d)$:Même chose mais le poids w est fixé à d.
11. $TSP'_{\mathbb{R}}$:Même chose mais le graphe n'est pas nécessairement complet.
12. $LTSP_{\mathbb{R}}$(Local Traveling Salesman Problem) :Etant donné un graphe complet avec des poids réels sur les arrêtes et ainsi que des sous ensembles d'arêtes $(E_1, E2\dots)$ et des poids $(w1, w2\dots)$, existe-t-il un chemin hamiltonien tel que son intersection avec chaque Ei soit de poids inférieur ou égal à w_i.
13. $LTSP'_{\mathbb{R}}$:Même problème mais on s'autorise aussi des contraintes de type inférieur strict à un certain poids.

Bien sur on va aussi considérer les versions qui "comptent" de ces problèmes quand c'est naturel, qu'on notera avec un $\sharp$ devant le nom du problème.

2 Résultats préliminaires

2.1 Motivations

Un des buts du stage était de montrer que le problème $\sharp TSP_{\mathbb{R}}$ est $\sharp P_{add}$ complet pour une réduction polynômiale. Ce problème s'étant révélé plus difficile que prévu, on a essayé de fournir d'autre problèmes $\sharp P_{add}$ complets qui nous rapprocherait de ce but.
On peut dans la même optique étudier le problème $\sharp KNAP_{\mathbb{R}}$ dans un modèle affaibli $\sharp P^{=}_{add}$, c'est à dire avec des branchements sur $=$ seulement. On va aussi essayer de donner des raisons pour lesquelles, pour certains problèmes, les réductions polynômiales sont difficiles ou impossibles.
La motivation de cette étude vient de l'existence de nombreux problémes $\sharp P_{add}$ complet pour une réduction Turing, et de la méconnaissance de problème complet pour réduction polynomiale qui pourtant est bien plus puissante comme on le verra plus tard. Sans doute la classification à Turing réduction près n'est pas assez fine, comme semble l'indiquer le résultat $D\sharp PH_{add} \subseteq FP^{\sharp P}_{add}$, c'est à dire que tout problème de $D\sharp PH_{add}$ se Turing réduit à un problème de $\sharp P$.

2.2 $DNP_{add} = NP_{add}$

On accepte le résultat suivant démontré dans [1] chapitre 21.

Lemme 1. *Soit un polyèdre P non vide de $\mathbb{R}^n$ définit par*

$$A_1 y \leq b_1, A_2 y < b_2$$

où $A_1 \in \mathbb{Z}^{N_1 \times n}$ et $A_2 \in \mathbb{Z}^{N_2 \times n}$. Les composantes des matrices A_1 et A_2 sont de taille bornés par L. Alors il existe un $x \in P$ tel que,

$$\forall i, x_i = \sum_j u_{ij} b_{1j} + \sum_j v_{ij} b_{2j} + w_i$$

où les coefficients u_{ij}, vij, w_i sont des rationnels de taille bornée par $(Ln)^c$ pou une certaine constante c.

Théorème 1. *$DNP_{add} = NP_{add}$*

Démonstration. Soit Π un problème NP_{add}, considérons un mot a, et un témoin y tels que $a \in \Pi \Leftrightarrow (a, y) \in L$ où $L \in P$. Le calcul sur a, y peut être représenter par un arbre de décision linéaire tel que pour une taille fixée de

a, on a toujours le même arbre. Satisfaire cet arbre équivaut à vérifier un système d'inégalité comme celle du lemme précédent. Donc en appliquant le lemme sur le système en considérant a comme une constante, on a l'existence d'un x qu'on peut décrire par des rationnels de longueur bornée.

On peut donc tester l'appartenance de a avec un témoin de taille polynômial et appartenant à $\{0,1\}^{\infty}$. On choisit comme témoin les rationnels permettant de construire x. On peut calculer en temps polynômial x et vérifier que $(a,x) \in L$ en temps polynômial. On vient de démontrer qu'on peut dans tout les cas passer d'un témoin réel à un témoin binaire donc $DNP_{add} = NP_{add}$. □

Remarque 1 (De l'intérêt d'étudier quand même DNP_{add})**.** En fait si $DNP_{add} = NP_{add}$, on a $D\sharp NP_{add} \neq \sharp NP_{add}$, ce qui se voit aisément car les fonctions de $\sharp NP_{add}$ peuvent prendre la valeur ∞ mais pas celle de $D\sharp NP_{add}$. La question de trouver un rapport étroit entre ces deux classes est intéressante, on sait que c'est les mêmes à Turing réduction prêt mais ça ne parait pas assez précis car on a aussi $\sharp P$ qui est égal à ces deux classes à Turing réduction près.

Remarque 2. NP_{add} et NP^0_{add} sont équivalent à réduction polynômiale dans P_{add} près. Il suffit en effet de considérer une machine de P_{add} qui décide d'un problème de NP_{add} avec l'aide d'un témoin, de transformer ses paramètres en variables, et on obtient une machine de P^0_{add} qui décide un problème de NP^0_{add}. Pour réduire le premier problème au deuxième, il suffit de considérer la fonction de P_{add} qui affecte les paramètres qu'on a retiré aux variables correspondantes. De la même manière on a $\sharp P_{add}$ qui se réduit polynômialement à $\sharp P^0_{add}$.

Remarque 3. A la lumière des remarques précédentes, on voit que quand on veut montrer la NP_{add} complétude d'un problème, il suffit de lui réduire tous les problèmes de DNP^0_{add}.

2.3 Turing complétude à la pelle

Théorème 2. *On a l'inclusion de classe suivante : $NP_{add} \subseteq P^{NP}_{add}$, c'est à dire que tous les problèmes NP (classique) complets pour Turing réduction sont NP_{add} complets pour Turing réduction.*

Cette démonstration s'inspire d'un article de Bürgisser [4] et utilise la construction de Meyer auf der Heide suivante . Soit $H_{s,n}$ l'ensemble des polynômes linéaires dans $\mathbb{R}^n$ de coefficients a_i tels que $\sum |a_i| \leq 2^s$. Soit $F_{s,n}$ l'ensemble des ensembles appelés faces $F = \cap_{f \in H_{s,n}} \{x \in \mathbb{R}^n | sg(f(x)) = \sigma(f)\}$ où σ est une fonction de $H_{s,n} \to \{-1,0,1\}$. En fait on découpe $\mathbb{R}^n$ en face ou cellule par des hyperplans, définis par des entiers pas trop grand. On va admettre les résultats suivants.

Soit M une machine sans constante réelle, à entrée dans $\mathbb{R}^\infty \times \{0,1\}^\infty$, fonctionnant en temps polynômial en la taille du premier argument. La relation

$$R := \{(F,y) \in F_t \times \{0,1\}^\infty | \forall x \in F, M \text{ accepte (x,y) } \}$$

peut être vérifiée en temps polynômial.

On utilise un résultat important de Fournier et Koiran [8] :
Pour tout polynôme t le problème de la localisation d'un point est dans $(FP^0_{add})^{NP}$. C'est à dire qu'on peut calculer un point rationnel de taille bornée polynômialement, appartenant à la même face $F \in F_{t(n),n}$ que l'entrée avec une machine polynômiale sans constante faisant appel à un oracle classique de NP. En fait on sait pour chaque entrée réelle lui associer sa face par un point codé avec des booléens.

Démonstration. Supposons que $A \in NP_{add}$, par le théorème 1, $A \in DNP_{add}$,

$$x \in A \Leftrightarrow \exists y \in \{0,1\}^{p|x|} | (x,y) \in B$$

où p est un polynôme et B un problème de P. Soit M_B une machine décidant en temps polynômial l'appartenance à M et α le n-uplet des constantes non triviales utilisées par la machine.

$$C = \{(x,z,v) \in \mathbb{R}^\infty \times \{0,1\}^\infty \times \mathbb{R}^n | M_B \text{ accepte } (x,z) \text{en remplaçant } \alpha \text{par } v\}$$

On a bien sûr $C \in P^0_{add}$ et $(x,z) \in B \Leftrightarrow (x,z,\alpha) \in C$. On veut montrer $A \in P^{NP}_{add}$, il suffit de montrer que l'ensemble $A' = \{(x,v) | \exists z, (x,z,v) \in C\}$ appartient à $(P^0_{add})^{NP}$. On peut donc supposer que $B \in P^0$ et appliquer le premier lemme. On obtient que le problème

$$R = \{F \in F_t | \forall x \in F, M_B \text{accepte } (x,z)\}$$

est dans NP.

Maintenant on peut décrire un algorithme de décision de A, qui prouve que $A \in P^{NP}_{add}$. On localise l'entrée x sur une face F (réalisé en $(FP^0_{add})^{NP}$ par la proposition précédente), puis on décide si $F \in R$ par un appel d'oracle dans NP. On a donc testé, l'appartenance de F à R ce qui équivaut à tester si x appartient à A. □

Remarque 4. Ce résultat est vrai pour tout la hiérarchie polynômiale (PH).

Théorème 3. *On a l'égalité suivante :* $FP^{\sharp P_{add}}_{add} = FP^{D\sharp P_{add}}_{add} = FP^{\sharp P}_{add}$.

Preuve :
La démonstration est similaire à celle du théorème précédent. Soit $\phi \in D\sharp P_{add}$,

$$\phi(x) = |\{y | \exists z, (x,y,z) \in B\}|$$

Soit M_B une machine de P_{add} qui décide de l'appartenance à B et qu'on peut supposer comme précédemment sans constante. On a par ailleurs que la fonction $\psi(F) = |\{y|\forall x \in F, \exists z, (x, y, z) \in B\}|$ est calculable dans $\sharp P$.
Enfin l'algorithme qui à une entrée x calcule la face F de $F_{t(n),n}$ correspondante, puis par un appel à un oracle de $\sharp P$ calcule $\psi(F)$, nous donne $\phi(x)$ et prouve $A \in FP_{add}^{\sharp P}$.

Corollaire 1. *Ce résultat nous donne un grand nombre de problèmes $\sharp P_{add}$ ou ce qui revient au même $D\sharp P_{add}$ complets pour réduction Turing. La plupart des versions qui comptent des problèmes présentés sont donc complètes pour Turing réduction.*

Théorème 4. *Le problème $BCSAT$, c'est à dire la satisfaction d'un circuit par une entrée booléenne est DNP_{add} complet et $\sharp BCSAT$ est $D\sharp NP_{add}$ complet.*

C'est en fait un résultat indépendant du modèle, la démonstration suivante fonctionnant quel que soit le modèle (voir [12]).

Démonstration. Soit un problème $\Pi \in DNP_{add}$, c'est à dire que

$$x \in \Pi \Leftrightarrow \exists y \in \{0,1\}^{p|x|}, tq(x, y) \in L$$

où L est un problème de P_{add}. Il existe donc une famille de circuit donné par une machine de Turing polynômiale qui décide L. On considère la fonction qui à une entrée x associe le circuit correspondant dans la famille qui décide L avec x considéré comme une constante, elle peut être calculé polynômialement car on obtient le circuit polynômialement. Alors la satisfaction de ce circuit par un y nous donne $x \in \Pi$ donc on a réduit polynômialement Π à $BCSAT$. On remarque que le nombre de y qui satisfont le circuit correspond à $\sharp BCSAT(x)$ mais aussi à $\sharp\Pi(x)$. □

La démonstration reste la même dans le cas d'un témoin $y \in \mathbb{R}$ c'est à dire pour les classes NP_{add} et $\sharp NP_{add}$
Ceci fourni le seul exemple de problème complet pour réduction polynômiale que j'ai pu d'abord trouver, mais on va s'attacher à en trouver d'autre. On a par exemple le problème DIM, i.e. décider si la dimension d'un ensemble donné par un système d'inégalité est inférieur à un entier donné.
Il faut aussi citer des problèmes de nature géométrique, pour lesquels ont été démontrer des résultats de NP_{add} complétude pour réduction polynômiale [4] :

- $EADH_{add}$ Etant donné un circuit de décision, décider si l'ensemble reconnu par ce circuit est clos pour la topologie euclidienne.
- $ZDENSE_{add}$ Etant donné un circuit de décision, décider si l'ensemble reconnu par ce circuit est dense pour la topologie de Zariski.

– $UNBOUNDED_{add}$ Etant donné un circuit de décision, décider si l'ensemble reconnu par ce circuit n'est pas borné dans $\mathbb{R}^n$.

La démonstration de ces résultats repose principalement sur l'étude fine des chemins acceptants dans les circuits de décision. Tout ces problèmes sont complets mais ils n'ont pas de versions qui comptent intéressantes.

3 Difficulté de trouver des réductions polynômiale pour $D\sharp P_{add}$

3.1 Des problèmes non complets ?

Lemme 2. *Si un problème est $\sharp P_{add}$ complet pour une réduction polynomiale, alors le problème de NP_{add} qui l'a engendré est NP_{add} complet pour la même réduction. De même pour les classes avec témoin booléen ($D\sharp P_{add}$ et DP_{add}).*

Démonstration. Soient $\Pi, \Pi' \in NP_{add}$ avec $\sharp\Pi, \sharp\Pi'$ les problèmes qui comptent correspondants. On suppose que $\sharp\Pi'$ est $\sharp P_{add}$ complet pour une réduction polynomiale et on a donc :

$$x \in \Pi \Leftrightarrow \sharp\Pi(x) \neq 0 \Leftrightarrow \sharp\Pi'(f(x)) \neq 0 \Leftrightarrow f(x) \in \Pi'$$

On voit donc que Π est NP_{add} complet pour réduction polynômiale. □

On sait grâce à Valiant [11] que $\sharp 2SAT$ est complet pour Turing réduction dans $\sharp P$.

Donc le problème $\sharp 2SAT$ est $D\sharp P_{add}$ complet pour réduction Turing en utilisant un théorème précédant. Mais on sait aussi que $2SAT$ est dans P et donc aussi dans P_{add}, donc par le lemme précèdent, $\sharp 2SAT$ ne peut pas être $D\sharp P_{add}$ complet pour réduction polynômiale, sauf si $P_{add} = NP_{add}$.

Remarque 5. On peut résumer cela en remarquant qu'il doit exister des problèmes $\sharp P_{add}$ complet pour réduction Turing mais pas pour réduction polynômiale.

Remarque 6. Si un problèmes de NP_{add} se réduit polynômialement dans P^0_{add} à un autre problème alors leurs parties booléennes se réduisent de la même manière par une réduction polynômiale classique. Ceci vient du fait que $BP(P^0_{add}) = P$, c'est à dire qu'on peut remplacer la fonction de réduction dans P^0_{add} par une fonction dans P.

3.2 Ensembles creux

Définition 10. Soit $S \subset \mathbb{R}^\infty$, on dit qu'il est creux si la dimension topologique de $S \cap \mathbb{R}^n$ est inférieure à un polynôme en $\log(n)$.

Conjecture :
Il n'y a pas d'ensemble creux complet dans NP_{add} pour réduction polynômiale sauf si $P_{add} = NP_{add}$. Cette conjecture est tirée de [5] et de [7].

Remarque 7. Les problèmes classiques NP complets sont par un théorème précédent NP_{add} complet pour Turing réduction, mais comme ils sont tous creux, cela laisse planer un sérieux doute sur leur complétude pour réduction polynômiale.

Heureusement $TSP_{\mathbb{R}}$ et toutes ses variantes ne sont pas creux, ce qui nous laisse un espoir d'obtenir leur complétude pour réduction polynômiale.

4 Satisfaction de formules et de systèmes

Les résultats importants de cette section portent sur la complétude de BIP pour une réduction polynômiale ainsi que de la complétude de la version qui "compte". Ces résultats sont principalement inspirés par un article de Cucker et Matamala [6], légèrement simplifié et modifié, pour bien voir la complétude du problème qui compte correspondant.

4.1 Description du fonctionnement d'une machine par des formules

On va d'abord décrire le fonctionnement d'une machine BSS polynômiale par des formules. Puis on va montrer l'équivalence de la satisfaction de ces formules à la satisfaction d'une instance de BIP. On a vu que c'était suffisant pour montrer la NP_{add} complétude d'un problème de lui réduire tous les problèmes de DNP^0_{add}.
Soit un problème Π appartenant à DNP^0_{add}, on sait que $x \in \Pi \Leftrightarrow \exists y, (x, y) \in L$. Comme L est un langage reconnu en temps polynômial, on peut exhiber une machine BSS sans paramètre qui décide de l'appartenance $(x, y) \in L$ en un temps inférieur à p, ou p est polynômial en la taille de x.
On va donc introduire des variables $Z_{1,-p} \dots Z_{t,i} \dots Z_{p,p}$ qui représentent l'état des cases du ruban de la machine au cours de son calcul (la première variable indique l'étape du calcul et la deuxième la position de la case sur le ruban). On numérote tous les sommets du graphe de cette machine. On introduit également Y_t qui indique sur quel sommet du graphe la machine doit s'exécuter à l'étape $t + 1$. Enfin la fonction β représente les arrêtes du graphe : elle fournit en fonction d'un sommet, le sommet suivant, et si il y a deux choix dans le cas d'un sommet de test on notera $\beta^>$ et $\beta^\leq$ les deux possibilités. On simule l'initialisation de la machine avec x (de taille n) et le témoin y (de taille k) comme entrée de la manière suivante :

$$\phi_{init} = (Z_{1,1} = x_1 \wedge \dots Z_{1,n+1} = y_1 \wedge \dots Z_{1,n+k+1} = 0 \wedge \dots Z_{p,p} = 0 \wedge Y_1 = \beta(1))$$

à l'étape t, les changements peuvent être décrit par les formules suivantes :

– pour un sommet de calcul m,

$$Y_{t-1} = m \wedge Y_t = \beta(m) \wedge Z_{t,1} = Z_{t,2} * Z_{t,3} \bigwedge_{j \neq 1} Z_{t+1,j} = Z_{t,j}$$

où $*$ représente $+$ ou $-$ car on a à faire à une machine additive.

– pour un sommet de déplacement à droite m,

$$Y_{t-1} = m \wedge Y_t = \beta(m) \bigwedge_{j=-p+1}^{p} Z_{t+1,j} = Z_{t,j-1}$$

si c'est un noeud de déplacement à gauche on décale dans l'autre sens.

– pour un sommet de test m,

$$Y_{t-1} = m \wedge (Y_t = \beta^{>}(m) \wedge Z_{t,1} > 0) \vee (Y_t = \beta^{\leq}(m) \wedge Z_{t,1} \leq 0) \bigwedge_{j=-p}^{p} Z_{t+1,j} = Z_{t,j}$$

Pour décrire l'étape t du calcul, il suffit de dire qu'il passe par un sommet, c'est à dire qu'on fait la disjonction des formules précédentes sur tous les sommets.
Enfin l'acceptation s'écrit ainsi : $\phi_{accept} = (Zp, 1 = 1 \bigwedge_{j \neq 1} Z_{p,j} = 0)$
On a donc 'simulé' le fonctionnement de la machine, ce qui fait que l'acceptation de x est équivalent à

$$\exists y_1 \ldots \exists y_p \exists Z_{2,-p} \ldots \exists Z_{p,p} \in \mathbb{R}, \, (\phi_{init} \wedge (\bigwedge_t \phi_t) \wedge \phi_{accept})$$

On remarque que le nombre d'assignation qui permet de satisfaire cette formule est exactement le nombre de y qui satisfont $(x, y) \in L$ puisque une fois fixé les y_i, les $Z_{i,j}$ sont fixés par le fonctionnement de la machine. La deuxième remarque satisfaisante est que cette formule est de taille polynômiale en $|x|$. On veut maintenant ramener la satisfaction de ces formules à la satisfaction d'un système d'inéquation.

4.2 Complétude de BIP

Décrivons d'abord précisément le problème BIP.

Définition 11. Soit l la fonction indicatrice des réels positifs. BIP est la satisfaction des formules suivantes par des variables booléennes x_i :

$$\forall j < n, \, l(\sum_i x_i \lambda_{i,j}) = \tau_i$$

Ou $\tau_i \in \{0, 1\}$ et $\lambda_{i,j} \in \mathbb{R}$ sont les données du problème.

Théorème 5. *BIP est NP_{add} complet pour réduction polynômiale et $D\sharp P_{add}$ complet pour réduction polynômiale.*

Démonstration. On va commencer par remplacer les formules ϕ_t par un système d'égalité qui est satisfiable si et seulement si les formules sont satisfiables. On va par exemple remplacer la partie qui décrit les noeuds de calcul par le système suivant :
$\sum_{m \in C} \sigma_{t,m}(Y_{t-1} - m) = 0$
$\sum_{m \in C} \sigma_{t,m}(Y_t - \beta(m)) = 0$
$\sum_{m \in C} \sigma_{t,m}(Z_{t,1} - (Z_{t,2} * Z_{t,3})) = 0$
$\sum_{m \in C} \sigma_{t,m}(Z_{t,2} - Z_{t-1,2}) = 0$
$\vdots$
$\sum_{m \in C} \sigma_{t,m}(Z_{t,p} - Z_{t-1,p}) = 0$
et on impose aussi les conditions suivantes pour tout $m \in C$
$sgn(Y_{t-m}) = \sigma'_{t,m}$
$sgn(1 - \sigma'_{t,m}) = \sigma_{t,m}$
On fait ainsi pour tout les noeuds ce qui permet de trouver un système dont la satisfaction est équivalent à la satisfaction des formules ϕ_t, dont la taille n'est pas plus que polynômiale en la taille des ϕ_t.
Le système que nous avons construit contient des variables réelles mais on peut exprimer celles-ci comme combinaison linéaire des constantes contenues dans la machine qu'on veut simuler. Les coefficients de ces combinaisons linéaires sont des entiers de taille bornée polynômialement par construction. Donc on peut coder ces entiers par des booléens et notre problème se ramène bien à celui de la satisfaction par des booléens d'un système.
La simulation de la machine BSS se fait par un nombre polynômial de formule de taille polynômiale, et le passage aux inégalités est toujours polynômial, donc la complétude de BIP est bien à réduction polynômiale près. $\square$

5 Parlons de voyageurs de commerce

5.1 Plusieurs problèmes équivalents

Remarque 8. Les problèmes $PM_{\mathbb{R}}$ et $TSP_{\mathbb{R}}$ sont équivalents (à réduction polynômiale près). De même $TSP_{\mathbb{R}}$, $TSP'_{\mathbb{R}}$ et $TSP_{\mathbb{R}}(d)$ sont équivalents (à réduction polynômiale près).

En fait si dans un graphe bipartite on confond les deux sommets en regard, un arrangement est équivalent à un chemin hamiltonien donc $x \in PM_{\mathbb{R}} \Leftrightarrow x \in TSP_{\mathbb{R}}$.

De même les différentes variantes de $TSP_{\mathbb{R}}$, avec un graphe non complet ou avec la contrainte sur le poids donnée comme une variable, sont équivalentes.

En effet pour réduire le problème où toutes les arrêtes ne sont pas présentes il suffit d'affecter aux arrêtes manquantes un poids 'infini'. On réalise cela en ajoutant la valeur absolue de tous les poids négatifs des arrêtes ainsi que la contrainte de poids si elle est positive, ainsi tout chemin passant par une de ces arrêtes ne satisfera pas la contrainte de poids.

Pour réduire le problème où la contrainte w fait partie de l'entrée à celui ou elle est fixée à d, il suffit de rajouter au graphe deux sommets reliés par un arc de poids $w - d$ et de relier cette excroissance à tout les sommets d'un côté, et à un sommet quelconque de l'autre. On vérifie facilement que cette construction marche et conserve aussi le nombre de chemin possible.

Remarque 9. On peut facilement réduire la satisfaction d'une inéquation ou d'une disjonction d'inéquations à $TSP_{\mathbb{R}}$.

5.2 Peut on réduire BIP à $TSP_{\mathbb{R}}$?

Remarque 10. $TSP_{\mathbb{R}}$ revient à la satisfaction de pour tout k

$$\sum_{i \neq j} \lambda_{i,j} x_{i,j} + \lambda \leq 0$$

$$\sum_{i} x_{i,k} = 1$$

$$\sum_{j} x_{k,j} = 1$$

où les $\lambda_{i,j}$ sont les poids des arrêtes et λ le poids maximal d'un chemin.

On considère ainsi $TSP_{\mathbb{R}}$ comme un sous problème de BIP, malheureusement réduire BIP à ce sous problème paraît difficile.
On va quand même réduire BIP à un problème qui parait plus proche en apparence à $TSP_{\mathbb{R}}$, c'est à dire $LTSP'_{\mathbb{R}}$.

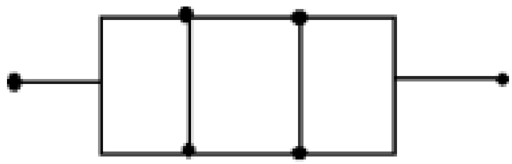

FIG. 2 – Un graphe fort disgracieux

Théorème 6. *$LTSP'_{\mathbb{R}}$ est NP_{add} complet pour réduction polynômiale et $\sharp LTSP'_{\mathbb{R}}$ est aussi $D\sharp P_{add}$ complet pour cette réduction polynômiale.*

Démonstration. On a une instance de BIP définie par des $\lambda_{i,j}$ et des τ_i. Il faut utiliser une astuce qui sert notamment dans la réduction de $3SAT$ à TSP dans le livre de Papadimitriou [9]. On va utiliser le graphe de la figure ci-dessus, on remarque qu'on peut prendre exactement deux chemins différents pour le traverser en passant une fois seulement par chaque sommet. On met donc autant de ces graphes en série qu'il y a de x_i et chaque choix de trajet dans un de ses sous graphes va correspondre au choix de $x_i = 0$ ou $x_i = 1$. On affecte en conséquence les poids sur ce sous-graphe : 0 sur toutes les arrêts correspondant à un trajet et les $\lambda_{i,j}$ sur l'autre trajet (pour i fixé). En fixant convenablement les contraintes à réaliser, c'est à dire les sous-ensembles d'arêtes E_i et le choix de $< ou \leq$ selon que $\tau_i = 0$ ou 1, on a équivalence entre l'instance de BIP et l'existence d'un chemin hamiltonien avec ces contraintes. Cette preuve fait correspondre à chaque manière de satisfaire le système un chemin hamiltonien différent, donc on a aussi le résultat de complétude pour $\sharp LTSP'_{\mathbb{R}}$. □

Mais réduire $LTSP'_{\mathbb{R}}$ à $LTSP_{\mathbb{R}}$ (on ne s'autorise que des $\leq$) pour une réduction polynômiale est plus difficile qu'il n'y parait. De même pour réduire $LTSP_{\mathbb{R}}$ à $TSP_{\mathbb{R}}$, il n'y a pas de solution simple. On a en fait identifié les deux principales difficultés, c'est à dire est on capable de passer des conditions locales à une condition globale et peut on gérer des réels qui sont arbitrairement proches ? On va introduire de nouveaux problèmes, dans un nouveau modèle dans lequel cette dernière difficulté ne se pose pas.

5.3 Un peu de théorie des modèles

Toute les bases nécessaires à la compréhension ce passage peuvent être trouvé dans [12]. On considère la théorie définie par :

$$\mathrm{Th}(\langle \mathbb{R}, +, -, \leq \rangle, \mathbb{R}) \cup_{n \in \mathbb{N}} \{\exists x \forall y, z, (x * n = 1) \wedge (0 < c < x) \wedge ((y \leq z) \Rightarrow (y + c \leq z + c))\}$$

La multiplication par de x par n a un sens, ce n'est que l'addition n fois de x. De même l'infériorité stricte est facilement obtenue en imposant l'infériorité large et en niant l'égalité. c est une nouvelle constante du langage.

Pour tout sous ensemble fini de ces formules, on sait que $\mathbb{R}$ avec les opérations usuelles les satisfaits. En effet si n_0 est le maximum des n tels qu'on ait $\exists x \forall y, z, (x * n = 1) \wedge (0 < c < x) \wedge ((y \leq z) \Rightarrow (y + c \leq z + c))$ dans notre ensemble fini de formule, il suffit de prendre $1/(n_0 + 1)$ pour le satisfaire. Donc par compacité on a l'existence d'un modèle qui satisfait toutes ces formules et donc d'un c infiniment petit compatible pour l'addition et la relation inférieur ou égale.
Ce nouveau modèle ressemble fortement à celui précédemment étudié, on va l'apeller $\tilde{\mathbb{R}}$. C'est en fait le même que $\mathbb{R}$ sauf pour l'existence d'un point non nul minimum dans $\mathbb{R}^+$. On peut vérifier facilement que la version de BIP dans ce modèle est aussi complète. En effet la démonstration où l'on simule une machine BSS reste la même, car on a juste rajouté des constantes dans ce que sait manipuler la machine. Par contre on va pouvoir prouver que le problème $LTSP_{\mathbb{R}}$ dans ce modèle est complet, attention ce n'est plus exactement le même problème, d'ailleurs on va l'appeler $LTSP_{\tilde{\mathbb{R}}}$, car le langage reconnu est dans $\tilde{\mathbb{R}}$.

Théorème 7. *$LTSP_{\tilde{\mathbb{R}}}$ est complet dans $NP_{\tilde{\mathbb{R}}}$ pour réduction polynômiale.*

Démonstration. On remarque que

$$\forall x, y \in \tilde{\mathbb{R}},\ x \leq y \Leftrightarrow x < y - c$$

Donc on va transformer toutes nos inégalités strictes en inégalités larges de la manière suivante :

$$\forall j,\ \sum_i \lambda_{i,j} x_i < 0 \ \rightarrow\ \forall j,\ \sum_i \lambda_{i,j} x_i \leq c$$

On a par la remarque précédente que la satisfaction des deux systèmes est équivalente. Puis par la même construction que pour montrer que $LTSP'_{\mathbb{R}}$ est complet, en utilisant uniquement des ≤ 0 ou des $\leq c$, on obtient le résultat recherché. De plus on voit que $\sharp LTSP_{\tilde{\mathbb{R}}}$ est complet dans $D\sharp P_{\tilde{\mathbb{R}}}$ pour la même réduction polynômiale. □

Remarque 11. Le problème soulevé par l'existence de ces inférieurs stricts et esquivé en changeant de modèle, n'existe pas pour les entiers. En effet dans les entiers on connaît le 'pas' minimum entre deux entiers quelconques, de même on le connaît pour des rationnels de taille bornée, et on peut alors transformer des inférieurs stricts en large et vice-versa avec seulement une addition. D'ailleurs on peut voir notre nouveau modèle comme le produit de $\mathbb{R}$ et $\mathbb{Z}$ avec l'ordre lexicographique, c'est à dire qu'on se débrouille pour quantifier à quel point deux 'réels' sont proches.

Pour conclure, remarquons qu'il n'existe aucune démonstration de NP_{add} complétude Turing spécifique à un des problèmes précédents (comme TSP).

On a seulement un outil théorique qui nous donne ce résultat pour une grande partie d'entre eux (théorème 1) !On ne peut donc pas s'inspirer de ces réductions pour en trouver d'autes plus fines.
Peut on trouver des réductions plus intelligentes qui discriminent bien tous les problèmes considérés ? Comme cette réduction doit être plus fine que la Turing réduction, il va être difficile de démontrer des résultats de complétude !

6 Et les modèles proches ?

6.1 Un peu d'égalité

Les problèmes classiques complet (pour réduction polynômiale ou Turing) dans NP ne sont pas complets (pour réduction polynômiale ou Turing) dans $NP^{=}_{add}$. Ces résultats sont un corollaire des théorèmes démontrés dans [5] et [7]. En effet on sait que dans ce modèle, les problèmes creux ne sont ni Turing complet, ni complet pour réduction polynômiale.
Il est intéressant de trouver dans ce modèle des problèmes NP complet pour réduction polynômiale, en adaptant la démonstration de la complétude BIP, on obtient que BEP est $NP^{=}_{add}$ complet. La version qui compte de BEP, $\sharp BEP$ est complète pour réduction polynômiale dans $D\sharp P^{=}_{add}$.
On peut se demander si on peut utiliser ce résultat de complétude pour prouver des résultats sur $KNAP$ ou des problèmes proches comme on l'a fait sur $TSP_{\mathbb{R}}$. On peut par contre certainement déduire la complétude pour réduction dans $P^{=}_{add}$ de $LTSP^{=}_{\mathbb{R}}$, le même problème que $LTSP_{\mathbb{R}}$ avec des conditions d'égalités et pas d'infériorité.

6.2 Etudier les corps

Que se passe-t-il quand on ajoute la multiplication aux opérations possibles ? Ce modèle est sans doute le plus intéressante, car le plus naturel avec toutes les opérations de $\mathbb{R}$. On travail sur des ensembles algébriques plutôt que linéaire mais les questions sont assez similaires. Il y a plusieurs résultat de base sur la complétude dans $NP_{\mathbb{R}}$, on peut donner en exemple les problèmes suivants :

- $DIM_{\mathbb{R}}$ Etant donné un ensemble semi-algébrique et un entier, décider si la dimension de l'ensemble est inférieur à l'entier.
- $SAS_{\mathbb{R}}$ Etant donné un ensemble semi-algébrique, décider si il est vide ou non.
- $FEAS_{\mathbb{R}}$ Etant donné un polynôme réel de plusieurs variables, a-t-il ou non une racine réelle.

Il est particulièrement intéressant de remarquer que les problèmes $\sharp SAS_{\mathbb{R}}$ et $\sharp FEAS_{\mathbb{R}}$ sont Turing complets pour $\sharp P_{\mathbb{R}}$. De plus, si on donne l'ensemble semi-algébrique sous forme normale conjonctive, $\sharp SAS_{\mathbb{R}}$ est $\sharp P_{\mathbb{R}}$ complet pour réduction polynômiale. Pour une approche plus détaillée de ce domaine on peut consulter par exemple *Counting Complexity Classes for Numeric Computation II : Algebraic and Semialgebraic Sets* [2]. Il serait intéressant de trouver des liens entre les résultats dans ce modèle et ceux du modèle additif.

7 Problèmes ouverts

Ici je rappelle quelques problèmes ouverts intéressants de ce domaine, notamment ceux qui n'ont pas trouvé une solution satisfaisante dans ce rapport.

- Comme cela n'a pas été résolu, est ce que $LTSP_{\mathbb{R}}$, $TSP_{\mathbb{R}}$ sont NP_{add} complets pour des réductions polynômiales ? Que dire surtout de la complétude pour réduction polynômiale de $\sharp TSP_{\mathbb{R}}$ dans $D\sharp P_{add}$, qui était notre but au début de l'étude.
- Existe-il une réduction pour laquelle certains des problèmes Turing NP_{add} complets sont complets et d'autres non. Elle permettrait de séparer en plusieurs classes les problèmes qu'on a étudié.
- Trouver un rapport entre $D\sharp P_{add}$ et $\sharp P_{add}$ plus précis que $D\sharp P_{add} \in \sharp P_{add}$, ainsi que dans le modèle affaibli sans la comparaison.

Références

[1] L.Blum, F.Cucker, M.Shub and S.Smale. *Complexity and Real Computation*, Sprinter Verlag, 1998.

[2] P.Bürgisser, F.Cucker. *Counting Complexity Classes for Numeric Computation II : Algebraic and Semialgebraic Sets.*2003

[3] P.Bürgisser, F.Cucker. *Counting Complexity Classes for Numeric Computations I : Semi-Linear Sets.* 2003

[4] P.Bürgisser, F.Cucker, P.J. de Naurois. *The complexity of semilinear problems in succint representation.*2005

[5] F.Cucker, P.Koiran, M.Matamala. *Complexity and Dimension.* 1996

[6] F.Cucker, M.Matamala. *On digital Nondeterminism.* 1996

[7] H.Fournier. *Sparse NP-complete problems over the reals with addition.* 2000.

[8] H.Fournier, P.Koiran. *Lower bounds are not easier over the reals : Inside PH.* 2000

[9] C.H.Papadimitriou. *Computational Complexity*, Addison-Wesley, 1994.

[10] B.Poizat *Les petits cailloux*, Nur Al-Mantiq War-Ma'Rifah (Aléas), 1995.

[11] L.Valiant. *The Complexity of Enumeration and Reliability Problems.* 1979

[12] F.Wagner, N.Portier. *Cours de logique et complexité.*