Séminaire de Théorie spectrale et géométrie

Laurent Bessières

Les travaux de Nabutovsky et Weinberger sur la complexité de l'espace des variétés riemanniennes

Séminaire de Théorie spectrale et géométrie, tome 19 (2000-2001), p. 77-91

<http://www.numdam.org/item?id=TSG_2000-2001__19__77_0>

Séminaire de théorie spectrale et géométrie
GRENOBLE
Volume **19** (2001) 77–91

LES TRAVAUX DE NABUTOVSKY ET WEINBERGER SUR LA COMPLEXITÉ DE L'ESPACE DES VARIÉTÉS RIEMANNIENNES

Laurent BESSIÈRES

1. Introduction

Nous allons présenter une partie des travaux de [NW] sur la complexité de l'espace des métriques riemanniennes d'une variété compacte.

Soit M^n une variété compacte. On travaillera sur les espaces suivants :

- $\mathrm{Met}(M)$ est l'ensemble des espaces métriques homéomorphes à M, équipé de la distance de Gromov-Hausdorff (voir [GLP]).
- $R_1(M)$ est l'ensemble des variétés riemanniennes (M,g), de courbure sectionnelle bornée en valeur absolue par 1, à difféomorphisme de M près.
- $\mathcal{M}_1(M)$ est la fermeture de $R_1(M)$ dans $\mathrm{Met}(M)$. Un élément de $\mathcal{M}_1(M)$ a une structure de variété et une métrique de classe C^1.

$$R_1(M) \subset \mathcal{M}_1(M) \subset \mathrm{Met}(M)\,.$$

On étudie la fonctionnelle diamètre $\mathrm{diam} : \mathcal{M}_1(M) \to R^+$ et la topologie des niveaux $\mathrm{diam}^{-1}[0;d]$, $d > 0$.

Informellement, le résultat principal de [NW] est : en dimension $\geqslant 5$, $\mathrm{diam}^{-1}[0;d]$ possède un nombre $\geqslant e^{cd^n}$ de composantes connexes ayant les propriétés suivantes :

a) Le volume est minoré par une constante universelle $V_n > 0$. En particulier, chacune de ces composantes admet un minimum local pour le diamètre.

b) Pour relier deux telles composantes par un chemin dans $\mathcal{M}_1(M)$, on doit passer par des diamètres croissant plus vite que toute fonction calculable de d.

Précisons ce dernier point. Une fonction $\phi : \mathbb{N} \to \mathbb{N}$ est dite **calculable** s'il existe

Classification math. : 53C20, 53C23, 03D40, 03D15.

un algorithme T (informellement un programme d'ordinateur, formellement une machine de Turing) qui calcule ϕ (voir section 2, remarque 1). L'assertion b) dit que pour toute fonction calculable ϕ, pour tout réel d assez grand, il existe un nombre $\geqslant e^{cd^n}$ de composantes connexes de $\mathrm{diam}^{-1}[0,d]$, où le volume est minoré par V_n, et telles que tout chemin dans $\mathscr{M}_1(M)$ entre deux telles composantes doit passer par des diamètres $\geqslant \phi([d])$.

En fait, on démontrera une version discrète de cette assertion. Appelons δ-chaîne une suite finie de points de $\mathscr{M}_1(M)$ tels que deux points successifs sont à distance de Gromov-Hausdorff $\leqslant \delta$, $\delta > 0$. Une δ-composante de $\mathscr{M}_1(M)$ sera alors une classe pour la relation d'équivalence : "être relié par une δ-chaîne". Le paramètre δ peut-être calculé pour satisfaire la propriété suivante. Dans l'espace $\mathscr{M}_{1,v,d}$ de toutes les variétés riemanniennes à géométrie bornée (*i.e.*, |courbure| $\leqslant 1$, volume $\geqslant v$, diamètre $\leqslant d$) de dimension n fixée, deux variétés à distance $d_{GH} \leqslant \delta$ sont difféomorphes et même 1,01 lipschitz équivalentes. Précisons que $\delta(d,v)$ est calculable en fonction des paramètres v,d ([Che],[P]).

Formellement, le résultat principal de [NW] est :

THÉORÈME 1.1. — *Pour toute variété compacte M de dimension $\geqslant 5$, il existe des constantes $C_n > 0$, $V_n > 0$ telles que pour toute fonction calculable ϕ, il existe $d_\phi > 0$ tel que pour tout $d > d_\phi$, pour $\delta = \delta(\phi([d]), \frac{V_n}{100})$,*

a) $\mathrm{diam}^{-1}[0,d]$ *possède plus de $e^{C_n d^n}$ δ-composantes où le volume est $\geqslant V_n$.*

b) Ces composantes ne se relient pas par des δ-chaînes dans $\mathrm{diam}^{-1}[0,\phi([d])]$.

Idées de la démonstration

L'idée la plus surprenante de [NW] est de relier la topologie de $\mathrm{diam}^{-1}[0,d]$ et la notion de complexité de Kolmogorov à temps borné d'un ensemble $I \subset \mathbb{N}$. Cette complexité est une fonction $K_I^t : \mathbb{N} \to \mathbb{N}$ définie par $K_I^t(N)$ = taille minimale (en nombre de lettres) d'un algorithme qui détermine $I \cap \{1,\ldots,N\}$, en un nombre de pas de calcul inférieur à $t(N)$, où t est une fonction calculable raisonnable (voir section 2, 3). Remarquons que la taille d'un tel algorithme est majorée par $N + cte$: on peut inclure les réponses dans l'algorithme.

Le point de départ est de trouver un ensemble $I \subset \mathbb{N}$, dont la complexité K_I^t croit au moins linéairement pour toute fonction calculable t (théorème de Barzdin, voir 2.1).

Ensuite, une partie importante de [NW] (que nous admettrons) consiste à associer à cet ensemble une famille de variétés riemanniennes. Pour cela, Nabutovsky et Weinberger renforcent un théorème de Boone-Novikov et obtiennent le résultat suivant :

THÉORÈME 1.2. — *Soit M_0^n une variété compacte, $n \geqslant 5$. Il existe un algorithme qui construit pour tout $k \in \mathbb{N}$, une n-variété riemannienne (M_k,g_k), telle que :*

a) M_k a la même homologie que M_0. De plus, soit M_k est difféomorphe à M_0, soit le volume simplicial vérifie $||M_k|| > ||M_0||$.

b) Soit $I \subset \mathbb{N}$ l'ensemble des indices k tel que M_k est difféomorphe à M_0. Alors I n'est pas calculable et pour toute fonction calculable croissante t, $K_I^t(N)$ croît au moins linéairement.

c) Il existe des constances $c_1 > 0$, $c_2 > 0$ et $c_3 > 0$ telles que pour tout entier naturel k,

- $|K(g_k)| \leqslant 1$
- *le rayon de convexité est* $\geqslant 1$
- $\mathrm{diam}(g_k) \leqslant c_1(\ln(k+1))^{\frac{1}{n}}$
- $c_2.\ln(k+1) \leqslant \mathrm{vol}(g_k) \leqslant c_3.\ln(k+1)$.

La preuve du théorème 1.1 se fait alors par l'absurde en supposant non thm 1.1 et thm 1.2. On construit pour une suite d'entiers strictement croissante $(N_i)_i$,une fonction calculable t et une suite d'algorithmes A_{N_i} tels que

a) A_{N_i} résoud le problème "M_k est difféomorphe à M_0?" pour $1 \leqslant k \leqslant N_i$ en moins de $t(N_i)$ pas.

b) $\dfrac{\text{taille de } A_{N_i}}{N_i} \underset{i \to +\infty}{\longrightarrow} 0$

et cela contredit donc 1.2(b).

Nous allons donner une présentation rapide de la théorie de la calculabilité et démontrer le théorème de Barzdin dans la section 2. Nous parlerons brièvement du travail de Nabutovsky et Weinberger sur le théorème de Boone-Novikov dans la section 3. La section 4 sera consacrée à la démonstration du théorème 1.1.

2. Algorithmes-Calculabilité-Complexité

Le concept de base de la théorie de la calculabilité est la machine de Turing. Nous n'allons pas utiliser la définition formelle mais l'équivalent informel : l'algorithme ou le programme. Pour information, la définition d'une machine de Turing est en Annexe. On appellera **algorithme** une suite finie d'instructions, écrites dans un langage donné, par exemple le français. Le modèle exemplaire est le programme d'ordinateur Turbo-Pascal. Les instructions seront composées d'un nombre fini d'opérations élémentaires (par exemple de calcul), que nous ne préciseront pas. Nous distinguerons seulement l'instruction **fin**. Les **entrées** de l'algorithme seront les entiers $k \in \mathbb{N}$. Pour chaque entrée k, un algorithme T aura deux fonctionnements possibles :

a) soit, après avoir exécuté un nombre fini d'opérations, l'algorithme **s'arrête** sur l'instruction **fin** et produit une sortie $T(k) \in \mathbb{N}$. Dans ce cas, l'entrée est **acceptée**.

b) soit l'algorithme **ne s'arrête pas**. Cette dernière possibilité peut arriver par exemple s'il ne trouve pas l'instruction suivante à exécuter ou au contraire s'il exécute indéfiniment une même boucle d'instructions. En fait, on va supposer que seule la deuxième possibilité arrive. Cela sera utile lorsqu'on considérera l'énumérabilité. En pratique, on

peut imaginer qu'une instruction "**ou boucle infinie**" est ajoutée à tous les embranchements de l'algorithme.

La **longueur** de l'algorithme sera le nombre de lettre de l' algorithme dans l'alphabet binaire. Le **temps de calcul** pour une entrée acceptée sera le nombre d'opérations élémentaires effectuées (ou nombre de pas).

Essayons d'approcher la notion de complexité d'un ensemble $I \subset \mathbb{N}$. Commençons par les ensembles les moins complexes.

DÉFINITION 1. — *On dit qu'un ensemble $I \subset \mathbb{N}$ est calculable s'il existe un algorithme qui détermine, pour tout $k \in \mathbb{N}$, si $k \in I$.*

REMARQUE 1. — *On peut aussi dire qu'une fonction $\phi : \mathbb{N} \longrightarrow \mathbb{N}$ est calculable s'il existe un algorithme $T : \mathbb{N} \longrightarrow \mathbb{N}$, **qui accepte toute entrée**, et tel que $\phi = T$. Un ensemble $I \subset \mathbb{N}$ est calculable si et seulement si la fonction caractéristique χ_I est calculable.*

Tous les ensembles finis sont calculables. $2\mathbb{N}$ est calculable, etc...

Observons que les notions pratiques de temps de calcul ou de capacité de mémoire n'ont ici aucune importance. On ne fait que des considérations de principe. Presque tous les ensembles $I \subset \mathbb{N}$ ne sont pas calculables. Le plus important est donné par le fameux "problème d'arrêt des machines de Turing". Observons tout d'abord que l'ensemble de tous les algorithmes est dénombrable : les algorithmes sont de longueur finie et peuvent être classés par ordre alphabétique. Appelons T_k le k-ième algorithme de la liste.

L'ensemble $I = \{k \in \mathbb{N}, T_k(k)$ s'arrête$\}$ n'est pas calculable. En effet, supposons que I soit calculable. Il existe donc un algorithme T tel que $T(k) = 1$ si $T_k(k)$ s'arrête et $T(k) = 0$ sinon. Considérons alors l'algorithme T' suivant : pour chaque entrée $k \in \mathbb{N}$, T' calcule $T(k)$ puis, si $T(k) = 0$, T' s'arrête en produisant la sortie 0, et si $T(k) = 1$, T' ne s'arrête pas. Cet algorithme apparaît dans la liste de tous les algorithmes sous un nom $T' = T_k$. On obtient la contradiction pour l'entrée k. Si $T'(k) = T_k(k)$ s'arrête, alors $T(k)$ vaut 1 et $T'(k)$ ne s'arrête pas. Et si $T'(k) = T_k(k)$ ne s'arrête pas, alors $T(k) = 0$ et $T'(k)$ s'arrête.

Cependant, il existe un algorithme qui ne s'arrête que pour les entiers $k \in I$. En effet, il existe un algorithme universel U, qui peut simuler tous les algorithmes : U produit T_k le k-ième élément de la liste ordonnée des algorithmes, puis exécute $T_k(k)$. Donc U ne s'arrête que lorsque $T_k(k)$ s'arrête. On dit alors que I a la propriété d'être **énumérable**.

DÉFINITION 2. — *On dit qu'un ensemble $I \subset \mathbb{N}$ est **énumérable** s'il existe un algorithme qui admet I comme ensemble des entrées acceptées.*

Le nom énumérable est justifié par la propriété suivante : I est un ensemble énumérable si et seulement si il existe un algorithme E tel que $E(\mathbb{N}) = I$. Autrement dit, I est produit par un algorithme, mais pas nécessairement dans l'ordre. Justifions d'abord l'équivalence, dans le cas non trivial ou I est infini. Si I est énumérable, il existe un algorithme T qui ne va s'arrêter que pour les entrées $k \in I$. On considère l'algorithme

E fonctionnant grossièrement comme suit : E exécute pour des valeurs croissantes de i, les i premiers pas de calculs de $T(1),\ldots,T(i)$, jusqu'à ce qu'il trouve une entrée n_1 pour laquelle $T(n_1)$ s'est arrêtée. Il sort $E(1) = n_1$. Il continue sans répéter $T(n_1)$ jusqu'à ce qu'il trouve $n_2 = E(2)$, etc... Les sorties de E sont exactement les entrées acceptées par T. Réciproquement, étant donné un algorithme E tel que $E(\mathbb{N}) = I$, alors posons $T(k) = 1$ dès que k apparaît dans la liste $E(1),\ldots,E(i)$, et $T(k)$ ne s'arrête pas sinon.

Donc un ensemble calculable est énumérable, et l'ensemble des entrées acceptées de $T_k(k)$ est énumérable mais non calculable. En fait, un ensemble I est calculable si et seulement si I et $\mathbb{N} - I$ sont énumérables : il suffit de lancer en parallèle l'énumération de I et celle de $\mathbb{N} - I$. Cela revient également à dire qu'il existe un algorithme E qui produit I dans l'ordre.

Essayons maintenant de quantifier la complexité d'un ensemble $I \subset \mathbb{N}$. Si I est calculable, on peut définir la complexité K_I comme **la longueur minimale d'un algorithme qui détermine** I. De manière équivalente, on peut demander que l'algorithme produise une suite de 0 et de 1 représentant les réponses. L'entrée de l'algorithme n'a ici aucune importance. En pratique c'est 0.

Si I n'est pas calculable, il n'existe pas de tel algorithme. On considère alors, pour $N \in \mathbb{N}$, la complexité $K_I(N) = K_{I\cap[1,\ldots,N]}$ de $I \cap [1,\ldots,N]$ comme une fonction de N. Pour chaque N, c'est **la longueur minimale d'un algorithme qui détermine** $I \cap [1,\ldots,N]$. Observons que $K_I(N) \leqslant N + \text{const}$. En effet, on peut inclure dans l'algorithme une liste de N 0 et 1 représentant les N réponses. Si I est énumérable, on peut améliorer l'inégalité en $K_I(N) \leqslant \ln_2(N) + \text{const}$, c'est dire trouver une suite d' algorithmes A_N, de longueur $\leqslant \ln_2(N) + \text{const}$, qui déterminent $I \cap [1,\ldots,N]$. Soit T_k un algorithme qui admet comme entrées les éléments de I. Soit n l'entier $1 \leqslant n \leqslant N$, accepté par T_k, qui maximise le temps de calcul de $T_k(n)$. Il est facile d'imaginer un algorithme, contenant les données k et n, solution. Il suffit de lancer les exécutions de $T_k(1),\ldots,T_k(N)$ pendant un temps de calcul inférieur à celui de l'exécution de $T_k(n)$, et de noter celles qui s'arrêtent. La longueur du numéro k est une constante, la longueur de n en binaire est bornée par $\ln_2(N)$ d'ou longueur$(A_N) \leqslant \ln_2(N) + \text{const}$.

On peut interpréter le temps de calcul d'un algorithme comme une mesure de la difficulté de son travail. On peut alors se demander ce que devient la complexité si l'on contraint ce temps de calcul par une fonction calculable. Il est raisonnable de ne considérer que des fonctions de croissance assez rapide, au moins linéaires en N.

DÉFINITION 3. — *Soit un ensemble $I \subset \mathbb{N}$ et t une fonction calculable sur $\mathbb{N}$. La* ***complexité de Kolmogorov à temps borné*** *$K_I^t(N)$ est la longueur minimale d'un algorithme qui calcule $I \cap \{1,\ldots,N\}$, en moins de $t(N)$ pas.*

On a encore $K_I^t(N) \leqslant N + \text{const}$ et $K_I^t(N) \leqslant \text{const}$ si et seulement si I est calculable. Par contre, si I n'est qu'énumérable, la majoration en $\ln_2(N)$ ne tient plus. Reprenons l'argument précédent avec l'algorithme A_N. Le point clé est la donnée de T_k et de l'entrée n acceptée par T_k qui maximise le nombre de pas parmi $[1,\ldots,N]$. On appelle généralement "busy beaver" (castor travailleur!) la fonction de N qui donne ce temps de

calcul maximal :

$$\beta(N) = \text{nombre de pas de } T_k(n) .$$

Cette fonction croit plus vite que toute fonction calculable. En effet, si elle était bornée par une fonction calculable, il suffirait d'exécuter T_k pendant ce nombre calculable de pas pour déterminer I. Le temps de calcul de A_N est grosso-modo $\beta(N)$. Or pour estimer $K_I^t(N)$, on ne doit considérer que les algorithmes de nombre de pas $\leqslant t(N)$, où t est calculable. Donc l'algorithme A_N n'est plus correct pour majorer $K_I^t(N)$. En fait, borner le nombre de pas augmente substanciellement la taille de l'algorithme :

THÉORÈME 2.1 (Barzdin, [Bar]). — *Il existe un ensemble $I \subset \mathbb{N}$, énumérable, tel que pour toute fonction calculable t,*

$$\exists C_t > 0, \exists N_t, \forall N > N_t, \quad K_I^t(N) > \frac{N}{C_t} .$$

Essentiellement, cet ensemble est si complexe qu'on ne peut pas faire beaucoup mieux, pour déterminer $I \cap \{1, \ldots, N\}$, que d'inclure les N réponses dans l'algorithme A_N.

REMARQUE 2. — *La donnée de $I \subset \mathbb{N}$ est équivalente à la donnée d'un mot $\omega \in \{0;1\}^{\mathbb{N}}$ représentant la fonction caractéristique de I sur $\mathbb{N}$. Les définitions de calculabilité, énumérabilité, etc., se transportent aux mots $\omega \in \{0;1\}^{\mathbb{N}}$.*

Preuve du théorème 2.1 ([Bar], [ZL]). Trouver cet ensemble revient à trouver un mot $\omega \in \{0,1\}^{\mathbb{N}}$ énumérable, tel que pour tout N, le plus petit algorithme qui produit $\omega_1 \ldots \omega_N$ en temps $\leqslant t(N)$ soit de longueur $> \frac{N}{C_t}$. On va construire cet ω en juxtaposant suffisamment de mots x^i de taille 2^i, de complexité maximale. La raison du 2^i apparaîtra plus loin.

LEMME 2.2. — *Pour tout entier n, il existe un mot $x \in \{0,1\}^n$, pour lequel*

$$K_x \geqslant n \tag{1}$$

Preuve du lemme 2.2. Le nombre de mots x en 0 et en 1, de longueur n, est 2^n. Le nombre de mots x en 0 et en 1, de longueur n, de complexité $K_x < n$ est inférieur au nombre d' algorithmes de longueur $< n$. En effet, un même mot peut être produit par plusieurs algorithmes mais un algorithme ne peut produire qu'un des mots. Le nombre des tels algorithmes est $\leqslant 2 + \cdots + 2^{n-1} = 2^n - 2$ (dans l'alphabet binaire). Donc il existe un mot x non produit part ces algorithmes. D'ou (1). □

On a donc, pour toute fonction calculable t, l'existence d'un mot x de longueur n tel que $n \leqslant K_x \leqslant K_x^t(n)$. L'avantage d'introduire t est que la construction de ce mot est alors algorithmique :

LEMME 2.3. — *Étant donné une fonction calculable t, il existe un algorithme qui, à tout entier n, fait correspondre un mot x de longueur n pour lequel*

$$K_x{}^t(n) \geqslant n \tag{2}$$

Preuve du lemme 2.3. L'algorithme commence par produire tous les mots de longueur n, puis il produit et exécute tous les algorithmes de longueur $< n$ pendant $t(n)$ pas. Il compare les mots et les sorties. □

On peut essayer de raffiner cet algorithme pour lui faire admettre également t comme entrée. On va vu que étant donné k, l'algorithme universel U peut produire l'algorithme T_k. On a vu également qu'il n'y a pas d'algorithme pour décider quand T_k va s'arrêter. Par conséquent, on ne sait pas produire uniquement les fonctions calculables. Si l'on remplace le t du lemme précédent par un T_k quelconque, l'algorithme peut ne pas s'arrêter. En considérant maintenant des mots de longueur $n = 2^i$, on a montré le

LEMME 2.4. — *Il existe un algorithme A de deux variables $(k,i) \in \mathbb{N} \times \mathbb{N}$ tel que si $T_k(2^i)$ s'arrête, $A(k,i)$ est un mot x^i de longueur 2^i tel que*

$$K^t_{x^i}(2^i) \geqslant 2^i \tag{3}$$

et si $T_k(2^i)$ ne s'arrête pas, $A(k,i)$ ne s'arrête pas.

On commence maintenant la construction de ω. Soit T un algorithme qui exécute en parallèle croissant la construction de $x^1,\ldots,x^i$ en suivant le schéma suivant. Toutes les lettres de x^i reçoivent initialement la valeur 0. T détermine k la plus grande puissance de 2 qui divise i, *i.e.*, $i = 2^k(2q+1)$. Alors si $T_k(2^i)$ s'arrête, x^i reçoit la valeur $A(k,i)$ définie par le lemme 2.4. Le mot $\omega = x^1 \cup x^2 \cup \cdots \cup x^i \ldots$ ainsi construit est énumérable, *i.e.*, on peut trouver un algorithme T' qui s'arrête pour l'entrée l si et seulement si $\omega_l = 1$. Vérifions que ce mot est la solution du théorème 2.1. Soit $t = T_k$ une fonction calculable, N très grand par rapport à k. Considérons le plus long mot x^i inclus dans $\omega_1 \ldots \omega_N$, où i est de la forme $2^k(2q+1)$. On a d'une part $1 + 2 + \cdots + 2^i \leqslant N$ et d'autre part $1 + 2 + \cdots + 2^i + 2^{i+2^{k+1}} > N$. Cela implique $2^{i+2^{k+1}+1} - 1 \geqslant N$, soit encore

$$2^i \geqslant \frac{N+1}{2^{2^{k+1}} - 1} \tag{4}$$

En utilisant (3), on obtient

$$K^t(\omega,N) \geqslant K^t(\omega,2^i) \geqslant K^t(x^i,2^i) \geqslant 2^i \geqslant \frac{N}{C_k} \tag{5}$$

où C_k est une constante ne dépendant que de k, c'est-à-dire de t. □

3. Renforcement du théorème de Boone-Novikov

Je n'ai pas lu la démonstration du théorème 1.2. Elle fait appel en particulier à la notion de groupe arithmétique. Je vais seulement donner quelques idées.

D'abord le théorème de Novikov ([R]). On part d'une machine de Turing T dont l'ensemble I des entrées acceptées n'est pas calculable. On lui associe un semi-groupe Ω de présentation finie pour lequel le problème des mots n'est pas résoluble, c'est-à-dire qu'il n'existe pas d'algorithme qui détermine quels mots du groupe sont triviaux (Post). L'étape suivante est d'obtenir un groupe de présentation finie où le problème des mots n'est pas résoluble (Boone-Novikov). On peut alors construire une famille de groupes G_k qu'on ne sait pas algorithmiquement distinguer du groupe trivial (Rabin). La dernière étape (Novikov) consiste à réaliser ces groupes comme groupes fondamentaux de variétés Σ_k ayant la même homologie que la sphère S^n, $n \geqslant 5$. On sait alors que Σ_k est homéomorphe à la sphère S^n si et seulement si G_k est le groupe trivial.

La partie importante de [NW] consiste à renforcer cette construction pour obtenir des sphères d'homologies Σ_k obéissant à des contraintes supplémentaires.

En premier lieu, la condition d'homéomorphisme est renforcée en difféomorphisme.

Ensuite, si G_k est non trivial, on veut que le volume simplicial vérifie $||\Sigma_k|| > 0$. D'après l'inégalité de Gromov $C_n||M|| \leqslant \text{minvol}(M)$ ([G]), cela implique que Σ ne peut pas avoir de métriques à courbure bornée de volume trop petit. On a même un contrôle uniforme : le théorème d'isolation de Gromov affirme qu'il existe une constante universelle ε_n telle que toute variété riemannienne complète (M,g), du courbure sectionnelle $|K_g| \leqslant 1$, et de volume $\text{vol}_g(M) \leqslant \varepsilon_n$ est nécessairement de volume simplicial nul $||M|| = 0$. Dans la suite, on dira qu'une variété riemannienne satisfaisant ces conditions géométriques est de **petit volume**. Une variété Σ_k de petit volume est donc difféomorphe à la sphère.

En considérant les sommes connexes $M_k = M_0 \sharp \Sigma_k$, on obtient une suite de variétés telle que soit M_k est difféomorphe à M_0, soit $||M_k|| = ||M_0|| + ||\Sigma_k|| > ||M_k||$.

Enfin, chaque variété M_k est munie d'une métrique g_k à courbure sectionnelle bornée, où le diamètre est contrôlé par $\ln(k)^{\frac{1}{n}}$ et le volume par $\ln(k)$. Cela nous permettra de contrôler la longueur des descriptions de ces variétés (en 4.6).

On obtient alors le théorème 1.2.

4. Preuve du théorème principal

On suppose non thm 1.1 et thm 1.2 pour aboutir à une contradiction. Écrivons non thm 1.1 :

$\forall c > 0$, $\exists \phi$ calculable, $\exists (d_i)_{i \in \mathbb{N}}$ tendant vers $+\infty$, tels que le nombre $\text{card}(d_i)$ de δ-composantes de $\text{diam}^{-1}[0,d_i]$ non δ-connectées dans $\text{diam}^{-1}[0,\phi(d_i)]$, et telles que le volume est $\geqslant V_n$ vérifie $\text{card}(d_i) \leqslant e^{cd_i^n}$. Prenons $c > 0$ pour qu'avec la constante c_1 du thm 1.2, on ait $2cc_1 < 1$. Soit N_i l'entier maximal tel que

$$c_1 \ln(N_i + 1)^{\frac{1}{n}} \leqslant d_i \tag{6}$$

On a alors

$$\text{card}(d_i) \leqslant e^{cd_i^n} \sim N_i^{cc_1} \tag{7}$$

On construit pour cette suite d'entiers strictement croissante $(N_i)_i$, une fonction calculable t et une suite d'algorithmes A_{N_i} tels que

a) A_{N_i} résoud le problème "M_k est difféomorphe à M_0?" pour $1 \leqslant k \leqslant N_i$ en moins de $t(N_i)$ pas.

b) la taille de A_{N_i} est $\leqslant N_i^{2cc_1}$.

et cela contredit 1.2(b).

Décrivons l'algorithme hypothétique A_{N_i} sans nous embarrasser de considérations techniques. L'espace sur lequel on veut travailler est $\mathcal{M}_{1,\phi(d_i),\frac{V_n}{100}}$. C'est à dire l'espace de toutes les n-variétés riemanniennes compactes de courbure sectionnelle bornée en valeur absolue par 1, de diamètre majoré par $\phi(d_i)$, et de volume minoré par $\frac{V_n}{100}$. En gros, on veut discrétiser cet espace par un $\frac{\delta}{10}$-réseau $\text{Net}(\delta,\phi(d_i),\frac{V_n}{100})$, où δ est assez petit pour que 10δ- équivalence Hausdorff implique 1.01-équivalence Lipschitz. Ensuite, on approxime M_0 et $M_1,\ldots,M_{N_i}$ par des points de ce réseau. On détermine si M_k est δ-connecté à M_0 ou à une variété de petit volume. Si M_k est δ-connecté à M_0, l'algorithme répond que M_k est difféomorphe à M_0. Si M_k est δ-connecté à une métrique de petit volume, il répond $||M_k|| = 0$ et donc que M_k est difféomorphe à M_0. Si aucun de ces critères n'est positif, on ne peut pas encore conclure que M_k n'est pas difféomorphe à M_0 : il se pourrait que M_k et M_0 soient difféomorphes, mais situées dans des δ-composantes de gros volume différentes. Pour conclure, il suffit que l'algorithme ait une description de toutes les δ-composantes de gros volume contenant une variété difféomorphe à M_0. La négation du théorème 1.1 va permettre justement de majorer le nombre de telles composantes, la longueur de leur description et donc la taille de l'algorithme.

Un détail technique : les estimations de distance de Gromov-Hausdorff sont à, disons, $\frac{\delta}{100}$ près. Des points de $\text{Net}(\delta,\phi(d_i),\frac{V_n}{100})$ distants de $\delta - \frac{\delta}{200}$ peuvent être estimés distants de $\delta + \frac{\delta}{200}$. L'algorithme peut conclure faussement que M_k n'est pas difféomorphe à M_0. Une estimation erronée dans l'autre sens n'est pas préjudiciable. Ce problème est esquivé si l'algorithme détermine des 2δ-composantes à $\frac{\delta}{100}$ près.

Récapitulons. L'algorithme A_{N_i} va contenir grosso-modo deux parties. L'une sera de l'information et va dépendre fortement de N_i. L'autre sera un programme, à peu près constant.

L'information va consister en la donnée d'un représentant $R_l = (M_0,g_l)$ dans chacune des $\delta(d_i,\frac{V_n}{100})$-composantes de gros volume de $\text{diam}^{-1}[0,d_i]$ dans $\mathcal{M}_1(M_0)$. Chaque représentant sera décrit par un tableau des distances entre les points d'un Δ-réseau dans la variété R_l. Le paramètre Δ devra permettre d'estimer les distances de Gromov-Hausdorff entre variétés à $\frac{\delta(d_i,\frac{V_n}{100})}{100}$-près.

Le programme

(a) calcule un $\delta(\phi(d_i),\frac{V_n}{100})$

(b) construit un $\frac{\delta}{10}$-réseau $\mathrm{Net}(\delta,\phi(d_i),\frac{v_n}{100})$ dans $\mathscr{M}_{1,\phi(d_i),\frac{v_n}{100}}$

(c) remplace $R_1,\ldots,R_p$ par des points du réseau

(d) construit $M_1,\ldots,M_{N_i}$ et les remplace par des points du réseau

(e) détermine les 2δ-composantes et conclut.

Il reste à voir que les points (a) à (e) sont effectivement réalisables par un algorithme, et estimer la taille de A_{N_i}.

Lemmes techniques

Calcul du δ

LEMME 4.1. — *Il existe un algorithme qui à tous $(d,v) \in \mathbb{Q}\times\mathbb{Q}$ associe $\delta = \delta(d,v) \in \mathbb{Q}$ tel que dans $\mathscr{M}_{1,d,v}$, 10δ-équivalence Hausdorff implique 1.01-équivalence Lipschitz.*

Preuve du lemme 4.1. Un théorème de Cheeger [Che] montre qu'il n'y a qu'un nombre fini de types de difféomorphisme pour les variétés de $\mathscr{M}_{1,d,v}$. D'ou l'existence d'une constante δ telle que deux variétés δ-GH proches sont difféomorphes. Le fait δ soit calculable est démontré par Peters ([P]), mais sans la formule explicite. Après de laborieux calculs, on a trouvé

$$\delta(d,v) = c_n v^3 e^{-3(n-1)d}$$

avec $c_n = \frac{\pi^3}{\mathrm{vol}(S^n)^3 n^5 20^{2n+5}}$. □

Calcul du réseau Net

Le point (b) demande un peu de préparation. Un algorithme ne travaille sur que sur des entiers. Aussi, il faut définir comment une variété riemannienne peut être codée par des entiers. Le codage doit permettre à l'algorithme de calculer toutes les grandeurs géométriques (courbure, diamètre, volume) avec une précision donnée.

DÉFINITION 4. — *Une variété riemannienne (M,g) a une **structure semi-algébrique** s'il existe un système de coordonnées locales*

$$M \supset U_\alpha \xrightarrow{\phi_\alpha} V_\alpha \subset \mathbb{R}^n$$

*qui a les propriétés suivantes. Les ouverts V_α sont solution d'un nombre fini d'inégalités polynomiales de $\mathbb{R}^n$. Les fonctions de transitions $\phi_{\alpha\beta} = \phi_\beta \circ \phi_\alpha{}^{-1}$ ont un graphe semi-algébrique dans $\mathbb{R}^{2n}$, c'est-à-dire $\{(x,\phi_{\alpha\beta}(x))\} \subset \mathbb{R}^{2n}$ est solution d'un nombre fini d'inégalités polynomiales de $\mathbb{R}^{2n}$. Les coefficients $g_{ij} : V_\alpha \to \mathbb{R}$ de la métrique ont un graphe semi-algébrique dans $\mathbb{R}^{n+1}$ La structure semi-algébrique sera dite de **type fini** si il n'y a qu' un nombre fini d'inégalités polynomiales et que tous les coefficients sont entiers.*

Ainsi, on peut représenter une structure semi-algébrique de type fini par les coefficients de tous ses polynômes. Définissons la complexité semi-algébrique $comp_{sa}(M,g)$ comme la somme des degrés et des modules des coefficients pour une représentation semi-algébrique minimale..

LEMME 4.2. — *Il existe une fonction $N(\delta,d,v)$ calculable telle que $\forall M \in \mathcal{M}_{1,d,v}$, $\exists M_{sa} \in \mathcal{M}_{1,d+1,\frac{v}{2}}$, muni d'une structure semi-algébrique de type fini, vérifiant $d_{GH}(M,M_{sa}) \leqslant \frac{\delta}{10}$ et $comp_{sa}(M_{sa}) \leqslant N(\delta,d,v)$.*

Ce lemme dit qu'il existe une borne uniforme et calculable sur la complexité semi-algébrique nécessaire pour approcher à $\frac{\delta}{10}$ près les variétés de $\mathcal{M}_{1,d,v}$.

Preuve du lemme 4.2 C'est la partie la plus technique. Nous allons donner les grandes idées. Dans un système de coordonnées locales bien choisies, on veut approximer les fonctions de transitions et les g_{ij} par des polynômes, en contrôlant les quantités géométriques. La courbure fait intervenir les dérivées secondes des g_{ij}. Il faut donc approximer au moins en norme C^2. De plus, on veut un contrôle uniforme des coefficients et des degrés des polynômes dans l'espace des variétés riemanniennnes à géométrie bornée. Cela impose d'avoir une borne uniforme C^3 sur les g_{ij}.

Partant d'une variété $(M,g) \in \mathcal{M}_{1,d,v}$, on commence par régulariser le métrique, en utilisant le résultat suivant :

THÉORÈME 4.3 (Bemelmans,Min-Oo,Ruh [BMR]). — *$\exists \varepsilon_n > 0$ tel que $\forall \varepsilon_n > \varepsilon > 0$, $\forall g \in \mathcal{M}_1(M)$, $\exists g_\varepsilon \in \mathcal{M}_1(M)$ telle que $|g - g_\varepsilon|_{C^0} < \varepsilon$ et $\forall k \in \mathbb{N}$, $|\nabla^k R_{g_\varepsilon}|_{C^0} < C(n,\varepsilon,k)$. Les constantes ne dépendent pas de la variété.*

On peut calculer ε pour que $d_{GH}(g,g_\varepsilon) < \frac{\delta}{20}$ et que volume et diamètre soient proches. A partir de maintenant, on travaille avec g_ε qu'on appelle toujours g.

L'étape suivant consiste à se munir de coordonnées harmoniques :

THÉORÈME 4.4 (Jost,Karcher [F]). — *$\exists \rho(d,v) > 0$ tel que $\forall (M,g) \in \mathcal{M}_{1,d,v}$, $\forall B(m,\rho) \subset M$, il existe des coordonnées harmoniques $H : B(m,\rho) \longrightarrow B(0,\frac{11}{10}\rho) \subset \mathbb{R}^n$.*

Toutes les constantes sont calculables en fonction de d et v. Dans ces coordonnées, pour la métrique g choisie, on a $|g_{ij}|_{C^3} \leqslant C(d,v)$ et $|H_{\alpha\beta}|_{C^3} \leqslant C(d,v)$ où $H_{\alpha\beta}$ désigne une fonction de transition. On approxime en norme C^2 les fonctions $H_{\alpha\beta}$ par des fonctions semi-algébriques de type fini. Le degré et les modules des coefficients des polynômes est borné calculablement en fonction de δ, d, v et de la précision. On approxime de même les g_{ij} par des $\tilde{g_{ij}}$ semi-algébriques, en contrôlant calculablement les polynômes. On recolle les métriques à l'aide d'une partition de l'unité semi-algébrique finie. La variété riemannienne obtenue est appelée (M_{sa},g_{sa}). On peut normaliser la courbure pour que $|K_{g_{sa}}| \leqslant 1$, $d_{GH}(M,M_{sa}) \leqslant \frac{\delta}{20}$, $\mathrm{diam}(g_{sa}) \leqslant d+1$ et $\mathrm{vol}(g_{sa}) \geqslant \frac{v}{2}$ si les approximations sont assez fines. La précision nécessaire est calculable. Donc la complexité

semi-algébrique de (M_{sa},g_{sa}) est bornée calculablement en fonction de δ, d et v, ce qui termine la preuve du lemme 4.2. □

LEMME 4.5. — *La construction d'un $\frac{\delta}{10}$-réseau $\mathrm{Net}(\delta,d,v) \subset \mathcal{M}_{1,d,v}$ de variétés semi-algébriques finies est algorithmique.*

Preuve. Le lemme 4.2 montre qu'un tel réseau existe et que la complexité semi-algébrique est bornée. En notant $N = N(\delta,d,v)$, on a donc que toute M_{sa} est codé par un point à coordonnées entières dans un cube $[-N,N]^N$. Pour calculer le réseau, il suffit de déterminer pour quels points à coordonnées entières le codage donne bien une variété puis d'estimer les quantités géométriques. □

Remplacement des R_l

Les représentants R_l sont donnés sous forme de tableau de distances. Pour trouver un point $\frac{\delta}{10}$ proche dans le réseau $\mathrm{Net}(\delta,\phi(d_i),\frac{V_n}{100})$, il suffit d'estimer les distances de Hausdorff entre R_l et les points du réseau. C'est algorithmique.

Construction de $M_1,\ldots,M_{N_i}$

D'après le théorème 1.2, il existe un algorithme qui produit la liste des M_k. Ces variétés sont produites sous forme semi-algébriques et on peut estimer les distances de Hausdorff avec les points de $\mathrm{Net}(\delta,\phi(d_i),\frac{V_n}{100})$.

Calcul des 2δ-composantes et conclusion

Il suffit d'estimer les distances de Hausdorff entre points de $\mathrm{Net}(\delta,\phi(d_i),\frac{V_n}{100})$ puis le volume de chaque variété.

Longueur de A_{N_i}

LEMME 4.6. — *Pour N_i assez grand, la longueur de l'algorithme A_{N_i} est $\leqslant N_i^{2cc_1}$.*

Preuve. La longueur de l'algorithme est essentiellement le nombre de bits nécessaires pour coder les représentants R_l. Le programme de test reste le même.

Le nombre de représentants R_l est

$$\mathrm{card}(d_i) \leqslant N_i^{cc_1} \tag{8}$$

(début section 4, (7)).

Chaque représentant est contrôlé géométriquement par $\mathrm{diam}(R_l) \leqslant d_i$ et $c_2 \ln(N_i + 1) \leqslant \mathrm{vol}(R_l) \leqslant c_3 \ln(N_i + 1)$. On ne va considérer que les i assez grands pour que $\mathrm{vol}(R_l) \geqslant V_n$. On prend $\Delta = \frac{\delta(d_i,V_n)}{1000}$, soit $\Delta = c_4 e^{-3(n-1)d_i}$ (cf lemme 4.1). On

considère un Δ-réseau minimal $\subset R_l$, c'est dire que les Δ-boules centrées sur le réseau recouvrent la variété R_l, et que les $\frac{\Delta}{2}$-boules sont deux à deux disjointes. Un argument standard de comparaison des volumes permet de majorer le nombre N de points d'un tel réseau :

$$N \leqslant c_5 \frac{\mathrm{vol}(R_l)}{\mathrm{vol}_{S^n}(B(\Delta))} \tag{9}$$

$$\leqslant c_6 \frac{\mathrm{vol}_{\mathscr{H}^n}(B(d_i))}{\Delta^n} \tag{10}$$

$$\leqslant c_7 \frac{e^{(n-1)[\ln(N_i+1)]^{\frac{1}{n}}}}{\Delta^n} \tag{11}$$

En remplaçant Δ, on obtient

$$N \leqslant c_8\, e^{(n-1)(n+1)[\ln(N_i+1)]^{\frac{1}{n}}} \tag{12}$$

L'information stockée dans l'algorithme est le tableau des distances des points de ce réseau, soit N^2 distances. Chaque distance est comprise entre $\frac{\Delta}{2}$ et d_i et est donnée avec une précision de $\frac{\Delta}{100}$. Le nombre de bits nécessaires pour coder une distance est donc de l'ordre de

$$\ln(d_i) - \ln(\Delta) \leqslant c_8 \left(\ln\ln(N_i+1) + \ln(N_i+1)^{\frac{1}{n}}\right) \tag{13}$$

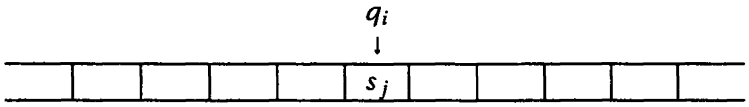
Pour N_i assez grand, on déduit de (8),(12) au carré et (13)

$$\text{taille de } A_{N_i} \leqslant c_9 {N_i}^{cc_1} e^{2(n-1)(n+1)\ln(N_i+1)^{\frac{1}{n}}} \ln(N_i+1)^{\frac{1}{n}} \tag{14}$$

$$\leqslant {N_i}^{2cc_1} \text{ pour } N_i \text{ assez grand. } \square \tag{15}$$

Appendice : Machine de Turing (Voir [D],[R])

L'idée d'une machine de Turing est la suivante. La machine n'a qu'un nombre fini d'états $q_1,\ldots,q_r$. On dispose d'une bande infinie découpée en cases. Chaque case peut être vide ou contenir un symbole $s \in S = \{s_1,\ldots,s_p\}$. La machine, selon son état interne peut lire une seule case à la fois, se déplacer sur la case de gauche ou celle de droite, écrire un des symboles $s \in S = \{s_1,\ldots,s_p\}$ sur la case lue, et changer d'état. L'opération effectuée est déterminée par l'état et le symbole lu.



Formellement, soit deux alphabets $Q = \{q_1,\ldots,q_r\}$ (les états) et $S = \{s_1,\ldots,s_p\}$ (les symboles). Une machine de Turing T est la donnée d'un nombre fini de quadruplets de la forme

a) $q_i s_j s_k q_l$

b) $q_i s_j L q_l$

c) $q_i s_j R q_l$

tels que les deux premiers termes sont différents deux à deux. La forme a) signifie que à l'état q_i, si la machine lit le symbole s_j, elle écrit par dessus le symbole s_k et se met dans l'état q_l. La forme b) (resp. c) signifie que à l'état q_i, si la machine lit le symbole s_j, elle se déplace sur la case de gauche (resp. droite) et se met dans l'état q_l.

Le fait que parmi les quadruplets, les deux premières lettres soient différentes deux à deux signifie que l'opération effectuée est déterminée par l'état et le symbole lu.

Une **description instantanée** est un mot en les s_j, q_i, contenant au plus une lettre de Q, non à droite. Exemple : $s_1 s_2 q_1 s_3$.

Une machine de Turing T envoie une description instantanée α sur une description instantanée β si $\alpha = P q_i s_j R$, $\beta = P q_l s_k R$ et $q_i s_j s_k q_l \in T$. (la machine lit s_j à l'état q_i, écrit s_k et passe à l'état q_l). On note $\alpha \to \beta$.

Un **calcul** est une suite de descriptions instantanées

$$\alpha_0 \to \alpha_1 \to \cdots \to \alpha_n$$

Décidons que l'état de départ est q_1 et l'état d'arrêt est q_0. On dit alors qu'une fonction $f : \mathbb{N} \to \mathbb{N}$ est calculable s'il existe une machine de Turing T telle que, $\forall k$,

$$\alpha = \quad q_1 \underbrace{s_1 \ldots s_1}_{\text{k fois}} \to \cdots \to \quad q_0 \underbrace{s_1 \ldots s_1}_{\text{f(k) fois.}}$$

Par exemple, la fonction $f(k) = k+1$ est réalisée en considérant $S = \{s_0,s_1\}$, $Q = \{q_0,q_1\}$ et $T = \{q_1 s_1 L q_1, q_1 s_0 s_1 q_0\}$.

Bibliographie

[Bar] J.M. BARZDIN, *Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set*, Soviet Math. Dokl., 9 (1968), 1251–1254.

[BMR] J. BEMELMANS, M. MIN-OO, E.RUH, *Smoothing riemannian metrics*, Math. Z., 188 (1984), 69–74.

[Cha] I. CHAVEL, *Riemannian geometry :a modern introduction*, Cambridge University Press,1995.

[Che] J. CHEEGER, *Finiteness theorems for riemannian manifolds*, Amer. J. Math., 92 (1970), 61–74.

[D] M. DAVIS, *Computability and unsolvability*, Dover Publications, New York 1982.

[F] K. FUKAYA, *Hausdorff convergence of riemannian manifolds and its applications*, in "Recent topics in differential and analytic geometry", ed. T. Ochiai, Adv. Stud. Pure Math. 18-I, Academic Press, Boston, 1990.

[NW] A. NABUTOVSKY , S. WEINBERGER, *Variational problems for Riemannian Functionals and arithmetic group*, à paraître aux Publications Math. de l'IHES.

[GLP] M. GROMOV, J. LAFONTAINE, P. PANSU, *Structure métrique pour les variétés riemanniennes*, 1981.

[G] M. GROMOV, *Volume and bounded cohomology*, Publications Math. de l'IHES, 56 (1982), 5–99.

[P] S. PETERS, *Convergence of Riemannian manifolds*, Comp. Math., 62 (1987), 3–16.

[R] J.J. ROTMAN, *Introduction to the theory of groups*, Springer,1995.

[ZL] A.K. ZVONKIN, L.A. LEVIN, *The complexity of finite objecst and the development of the concepts of information and randomness by means of the theory of the algorithms*, Russ. Math. Surv. 25(6) (1970),83–129.

Laurent BESSIÈRES
INSTITUT FOURIER
Laboratoire de Mathématiques
UMR5582 (UJF-CNRS)
BP 74
38402 St MARTIN D'HÈRES Cedex (France)
`Laurent.Bessieres@ujf-grenoble.fr`