Mesurer la complexité des objets numériques

Jean-Paul Delahaye [1]

« L'image d'un monde antique baignant dans le chaos n'est pas neutre. Elle accrédite la notion d'une croissance progressive de la complexité au cours des ères. Elle relie l'apparition de l'intelligence et de la conscience au déroulement de l'histoire. Elle restitue l'être humain dans ce vaste mouvement d'organisation de la matière à l'échelle cosmique. »

Hubert Reeves, Dernières nouvelles du cosmos, Éditions du Seuil, Paris, 2002, pp. 224 et 225.

Quand on considère les différentes étapes de l'évolution de l'univers depuis le Big Bang telles qu'elles sont décrites par la cosmologie contemporaine, on est tenté de dire — comme Hubert Reeves aime y insister — qu'il se *complexifie*. De même, on affirme que les écosystèmes terrestres comportent des éléments de plus en plus nombreux et variés interagissant de façon de plus en plus subtile et délicate. À une échelle de temps plus réduite, on parle encore de la *complexification* des sociétés humaines, des échanges sociaux et commerciaux, des objets technologiques et des théories scientifiques. Mais de quoi s'agit-il exactement ? Qu'est-ce qui précisément augmente quand on parle de complexité croissante ? Peut-on mesurer ou au moins définir de manière rigoureuse cette "complexité" et en faire un concept mathématique ? La question préoccupe les chercheurs depuis longtemps et une multitude de méthodes et d'idées ont été proposées pour définir et évaluer la complexité des objets artificiels et naturels, des processus et des interactions.

Dans cet article, nous nous attacherons uniquement au problème de la définition et de la mesure de la complexité des *objets numériques* (textes, images, contenus

1. Université des Sciences et Technologies de Lille, Laboratoire d'Informatique Fondamentale de Lille, UMR 8022 CNRS, Bât M3-ext, 59655 Villeneuve d'Ascq Cedex. E-mail : delahaye@lifl.fr.

de disques durs, de disques optiques, de clef USB, de téléphone portable, etc.). De manière indirecte, cela concernera tous les objets du monde réel, dès lors qu'on accepte de les représenter par des données numériques. Nous n'aborderons pas la question de savoir si certains objets sont impossibles à *numériser*, ni la question presque équivalente de savoir si une quantité infinie d'informations peut se trouver présente dans un volume fini d'espace. La question de la complexité des algorithmes (classes P, NP, PSPACE, EXPTIME, etc.) est un autre sujet qui n'a la plupart du temps qu'un sens asymptotique, alors que justement nous voulons parler d'objets finis.

Nous allons voir qu'à côté de mesures simples comme l'espace mémoire occupé par l'objet sur un support numérique (mesure très insatisfaisante), ou l'entropie de Shannon (guère meilleure contrairement à ce que certains imaginent), d'autres mesures sont plus sérieuses car elles prennent en compte la richesse combinatoire, ou mieux encore se fondent sur la théorie du calcul.

Nous soutiendrons que la *complexité de Kolmogorov* (ou *complexité de Chaitin-Kolmogorov*) est l'une des plus importantes notions introduites pour mesurer la complexité des objets numériques et qu'il est possible de la mettre en œuvre grâce aux algorithmes de compression (sans pertes). Nous défendrons cependant l'idée que la *profondeur logique de Charles Bennett* (notion liée à la complexité de Kolmogorov) est la piste la plus sérieuse pour définir et mesurer la *complexité structurelle* des objets numériques. Grâce à des résultats accumulés année après année depuis son introduction en 1977, la profondeur logique de Bennett apparaît être la définition scientifique la meilleure dès lors qu'on ne confond pas "contenu incompressible d'information" (mesurée par la complexité de Kolmogorov) avec "richesse d'organisation" ou "complexité structurelle". La mise au point d'outils qui réussissent, au moins dans certains cas, à évaluer la profondeur logique de Bennett conduit à en envisager des applications.

La taille mesurée en bits ou en octets

Bien sûr, la mesure la plus simple de la complexité d'un objet numérique est la taille de l'espace mémoire (mesuré en bits ou en octets) qu'on utilise pour le conserver (sans chercher à le compresser). Si on suit cette idée, une image de dix millions de pixels, chacun codé par un octet (au total donc : 10 Mo), sera considérée comme dix fois plus complexe qu'une image d'un million de pixels (1 Mo). C'est une première proposition naturelle et simple, mais on réalise immédiatement qu'elle n'est pas satisfaisante puisqu'une image toute blanche avec une telle mesure serait considérée aussi complexe qu'une image de même taille représentant un paysage ou une puce électronique ou n'importe quoi, ce qui n'est pas du tout conforme à ce qu'on attend d'une mesure de complexité (quel que soit le sens qu'on donne à ce mot).

L'entropie

L'entropie de Shannon d'un texte ou fichier s composé de n symboles pris dans un alphabet à k symboles est définie par :

$$H(s) = n\Big(-\sum_i p_i \log p_i\Big)$$

où i décrit les k symboles de l'alphabet et p_i est la fréquence d'utilisation de chacun des symboles dans s. Si $p_i = 0$, on considère que $p_i \log p_i = 0$. Le logarithme utilisé est celui en base 2, c'est-à-dire, celui pour lequel $\log 2 = 1$.

Reprenons l'exemple de l'image toute blanche. Dans le cas d'un alphabet à 256 caractères, pour coder une image de n pixels, on trouve que l'image blanche (chaque pixel est représenté par le même caractère) a pour entropie 0 : la fréquence du caractère codant le blanc est 100%, et les autres symboles ont pour fréquence 0%.

Dans le cas d'une image aléatoire, chacun des symboles a une fréquence proche de $1/256$, et donc l'entropie d'un fichier aléatoire de n symboles est approximativement égale à :

$$n\Big(-\sum \frac{1}{256}\log\frac{1}{256}\Big) = n\Big(-\log\frac{1}{256}\Big) = 8n.$$

C'est la quantité d'information (mesurée en bits : $8n$ bits = n octets) contenue dans l'image (sans compression), c'est-à-dire la taille de l'espace mémoire occupé par l'image.

Dans le cas d'une image représentant un paysage ou un objet réel, l'entropie de Shannon sera intermédiaire entre celle d'une image blanche, et celle des images aléatoires qui sont les images ayant une entropie maximale. Si une image est composée pour moitié de pixels blancs et pour moitié de pixels noirs, l'entropie sera

$$-n\sum_{i=0,1}\frac{1}{2}\log\frac{1}{2} = n$$

qui à nouveau représente la quantité d'information à laquelle on peut réduire l'image quand au lieu de coder chaque pixel par un octet, on le code par un seul bit d'information.

D'une façon générale, l'entropie de Shannon indique la quantité d'information mesurée en bits qu'il faut, au minimum, pour coder l'image en utilisant un code de type "remplacer chaque suite de p symboles par une suite de bits plus ou moins longue d'une manière réversible, c'est-à-dire de telle façon que le décodage ne présente pas d'ambiguïté" (codage de Huffman).

Exemple. Si le symbole 'a' est utilisé avec la fréquence $\frac{1}{2}$, 'b' avec la fréquence $\frac{1}{4}$, 'c' avec la fréquence $\frac{1}{8}$, 'd' avec la fréquence $\frac{1}{8}$, alors on codera 'a' par 0, 'b' par 10, 'c' par 110, 'd' par 111 (on réserve les codes courts pour les symboles les

plus fréquents). Avec les fréquences indiquées, un fichier de n pixels comporte $\frac{n}{2}$ 'a' (demandant chacun 1 bit), $\frac{n}{4}$ 'b' (demandant chacun 2 bits), $\frac{n}{8}$ 'c' (demandant chacun 3 bits), $\frac{n}{8}$ 'd' (demandant chacun 3 bits). Pour le fichier complet, on arrive à environ $\frac{n}{2}+\frac{2n}{4}+\frac{3n}{8}+\frac{3n}{8}=\frac{7n}{4}$ bits. Le codage proposé est optimal parmi les codages envisagés et ce qu'il donne est exactement ce qu'indique l'entropie :

$$-n\Big(\sum \frac{1}{2}\log\frac{1}{2}+\frac{1}{4}\log\frac{1}{4}+\frac{1}{8}\log\frac{1}{8}+\frac{1}{8}\log\frac{1}{8}\Big)=n\Big(\frac{1}{2}+\frac{2}{4}+\frac{3}{8}+\frac{3}{8}\Big)=\frac{7n}{4}.$$

Sur tout cela on consultera [16].

À première vue, l'entropie de Shannon apparaît comme une mesure assez satisfaisante de complexité. En y regardant de plus près, on découvre cependant que cette mesure attribue la même complexité

– d'une part, à une image composée "d'une moitié droite toute noire, et d'une moitié gauche toute blanche",

– et d'autre part, à une image utilisant "50% de pixels blancs et 50% de pixels noirs disposés aléatoirement" ou représentant "un objet non trivial (paysage, personnage, machine, etc.) qui comporte 50% de pixels blancs et autant de pixels noirs".

C'est absurde ! Ces images ont de toute évidence des complexités différentes : la première n'est pas complexe du tout, alors que les images d'objets réels ont certainement une assez grande complexité (quel que soit le sens qu'on donne à ce mot). L'entropie de Shannon ne convient pas comme mesure de complexité, car la fréquence d'utilisation des différents symboles ne détermine pas leur ordre dans un texte ou leur position dans une image, alors que la complexité d'un texte ou d'une image dépend évidemment de cet ordre ou de ces positions.

D'autres mesures inspirées de l'entropie de Shannon sont envisageables pour tenter de résoudre son utilisation directe insatisfaisante. On peut par exemple considérer l'entropie de Shannon des couples de deux symboles consécutifs, ou des carrés de 4 pixels extraits de l'image qu'on analyse. Ce qu'on obtient sera un peu meilleur, mais ne sera jamais pleinement satisfaisant car la complexité d'une image ne dépend pas seulement de la statistique sur les carrés de 4 pixels qui, comme les pixels, peuvent pour une statistique fixée être rangés simplement (par exemple en regroupant les carrés identiques) ou composer des motifs nombreux et variés, donc plus complexes.

Mesures combinatoires particulières

Pour les textes en langue naturelle, la richesse du vocabulaire est une marque naturelle de leur complexité. De même, le nombre de sous-mots d'une taille donnée, p, pris dans un fichier s en mesure la richesse, donc d'une certaine façon la complexité. Cette "variété combinatoire" qui cette fois prend en compte l'ordre des symboles est facile à mesurer puisqu'il suffit juste de mener un décompte.

Si p est un entier fixé et s une suite (finie ou infinie) de symboles pris dans un alphabet fixé (à k symboles), on appelle *fonction de complexité* (ou *complexité combinatoire*) de s, la fonction $p \to \mathrm{cc}_s(p)$ qui à p associe le nombre de mots différents de p symboles qu'on trouve dans s. Bien sûr :

$$0 \leq \mathrm{cc}_s(p) \leq k^p.$$

La fonction de complexité combinatoire de la suite `000....0...` est constante et égale à 1. Si une suite est périodique de période r alors $\mathrm{cc}_s(p) = r$ pour p assez grand. Les suites périodiques ne sont pas très complexes, et cela se voit sur leur fonction de complexité qui est bornée. On montre d'ailleurs qu'il y a équivalence entre les propriétés "être ultimement périodique" et "avoir une fonction cc_s bornée".

Un remarquable résultat récent dû à Boris Adamczewski et Yann Bugeaud [1] affirme que les nombres irrationnels algébriques ont des développements en base b ($b \geq 2$) dont la fonction de complexité combinatoire, cc, vérifie :

$$\lim_{p\to\infty} \frac{\mathrm{cc}(p)}{p} = \infty.$$

Ce résultat exprime que les développements en base b de ces nombres sont complexes et en particulier plus complexes que ceux des nombres rationnels qui sont ultimement périodiques, ce qui implique pour leurs développements décimaux que :

$$\lim_{p\to\infty} \frac{\mathrm{cc}(p)}{p} = 0.$$

Le résultat n'est pas une surprise, mais on attendait une démonstration et dans ce domaine tout résultat est difficile ! Notons qu'il ne peut pas y avoir de résultat du même type pour les nombres transcendants dont certains ont des fonctions de complexité combinatoire linéairement croissantes.

Même si ce type de mesure combinatoire de la complexité est intéressant, on ne peut pas s'en satisfaire. De toute évidence, la suite de chiffres décimaux suivante (les chiffres des entiers les uns derrière les autres) :

`01234578910111213141516171891202122232425 26...`

introduite par Champernowne en 1933 n'est pas complexe. En effet, elle se décrit très brièvement, on la programme sans mal, et quelqu'un qui en voit les premiers termes devine aisément les suivants. Pourtant sa complexité combinatoire est maximale : non seulement $\mathrm{cc}_s(p) = 10^p$ pour chaque p, mais on prouve que toutes les séquences m de longueur p (de chiffres de 0 à 9) apparaissent une infinité de fois et ont pour fréquence limite d'apparition $\frac{1}{10^p}$ (on dit que le nombre réel dont l'écriture en base 10 est cette suite de chiffres est un nombre *normal*).

Mesurer la complexité d'une suite s (finie ou infinie) par la variété des sous-mots qu'on y rencontre ne convient pas lorsqu'on aborde des objets un peu subtils dont la suite de Champernowne n'est qu'un exemple parmi une infinité d'autres. La variété

des sous-mots d'une suite numérique, s, est un indice de sa complexité et évaluer cette variété donne une certaine indication de cette complexité (d'où l'intérêt du résultat sur les nombres algébriques irrationnels), mais pas d'une manière entièrement satisfaisante. Il faut aller plus loin !

La complexité cc est utilisée pour établir des propriétés des générateurs pseudo-aléatoires : si elle est élevée, on a une assurance que le générateur n'est pas trop mauvais. D'autres mesures combinatoires du même type sont considérées par les chercheurs de ce domaine et sont utilisées en cryptographie. Chacune pourrait se présenter comme une mesure de complexité, dont malheureusement on constaterait qu'elle est insatisfaisante dès qu'on a affaire à des séquences possédant des régularités un peu cachées ou subtiles (comme les chiffres du nombre π). Voir par exemple [5].

Pour tester les générateurs pseudo-aléatoires, une multitude de tests statistiques sont aussi utilisés. Chacun d'eux pourrait donner une mesure de complexité en utilisant le principe suivant : meilleure est la satisfaction au test, plus grande est la complexité de la séquence testée. Nous ne donnons pas de détails sur ces mesures et renvoyons à la publication [40].

Aussi utiles en pratique qu'ils soient, ces tests statistiques ne sont pas satisfaisants dans l'absolu. En effet, la suite des chiffres de la constante π (ou de n'importe quel nombre irrationnel algébrique) en base 2 (ou autre) passe ces tests de manière satisfaisante : aucune exception n'a jamais été trouvée. Elle est donc maximalement complexe à leurs yeux. Pourtant, du fait d'une définition simple et de l'existence de méthodes assez rapides pour calculer les chiffres de cette suite, elle ne peut pas être considérée comme la plus complexe possible : les tests statistiques se trompent au sujet de sa complexité. Nous allons y revenir.

Pour se sortir de cette profusion de méthodes de mesures de la complexité des objets numériques, dont pourtant aucune n'est vraiment satisfaisante, une idée un peu plus abstraite était nécessaire. C'est de la théorie de la calculabilité qu'elle est venue.

La complexité de Kolmogorov

Une idée implicite dans les tentatives de mesure évoquées jusqu'à présent est que "le simple est ce qui se décrit brièvement, et le complexe ce qui à l'inverse ne peut se décrire brièvement". C'est l'idée de la de complexité de Kolmogorov introduite en 1965 par Andrei Kolmogorov (et sous des formes presque équivalentes, indépendamment et à peu près au même moment, par Ray Solomonoff et Gregory Chaitin). Pour plus de détail sur l'histoire de cette notion, on consultera la référence absolue sur le sujet [35] ou, en moins encyclopédique, [17, 19, 20].

Donnons très rapidement quelques précisions sur cette mesure de complexité qui ne possède plus les défauts des mesures évoquées jusqu'ici.

La *complexité de Kolmogorov*, $K(s)$, d'une suite binaire finie s est par définition la longueur du plus court programme qui engendre s, c'est-à-dire du plus court programme qui produit l'impression de s sur l'écran de l'ordinateur, sur l'imprimante ou dans un fichier.

Le plus court programme en question s'appelle le *programme minimal pour s*, on le notera s^*. S'il y a plusieurs programmes de taille minimale donnant s, on prend pour s^* le premier dans l'ordre lexicographique. Les langages que mentionne la définition sont les langages informatiques de programmation : langage C, langage Java, langage Fortran, langage LISP, etc.

Plusieurs questions se posent à la lecture de la définition.

– Comment faire pour les suites qui ne sont pas binaires ?

Moyennant une conversion (équivalente au passage de la base de numération b à la base 2), il est clair qu'on se ramène facilement à des suites binaires et des programmes binaires.

– Est-ce que la définition dépend du langage de programmation choisi ?

Oui bien sûr, mais assez peu. En effet, si on ne considère que des machines puissantes (c'est-à-dire équivalentes à des machines de Turing universelles), alors le résultat suivant répond à la question et donne sa consistance à la notion de complexité de Kolmogorov en montrant que la définition de $K(s)$ varie faiblement quand on change de langage de référence dans la définition de $K(s)$.

Théorème d'invariance. *Si U et V sont deux machines de Turing universelles (machines qui pour la donnée $[i,d]$ calculent ce que la machine de Turing numéro i calcule pour la donnée d), et si on note $K_U(s)$ (respectivement $K_V(s)$) la complexité de Kolmogorov quand on utilise la machine U (respectivement V) comme machine de référence, alors il existe une constante $C_{U,V}$ indépendante de s, telle que pour toute suite finie s :*

$$|K_U(s) - K_V(s)| < C_{U,V}.$$

Ce résultat est essentiel puisqu'il signifie "qu'à une constante additive près", le changement de langage de programmation (ou de machine de Turing universelle, ce qui revient au même) ne modifie pas la complexité de Kolmogorov d'une suite s. Que vous utilisiez des programmes écrits en langage Java ou en Pascal, Fortran ou C (chacun est un langage universel, chacun est équivalent à une machine de Turing universelle), la complexité que vous définissez en mesurant la longueur des plus courts programmes restera sensiblement la même, et sera donc très sensiblement la même si s est une suite assez longue.

L'idée de la définition de Kolmogorov est la formalisation mathématique de : "est complexe ce qu'on ne peut pas définir brièvement". Avant les développements de la théorie de la calculabilité (Gödel, Church, Turing, etc.), cette idée ne pouvait pas être

mise en œuvre dans le champ mathématique. La notion de longueur d'une définition (dans un langage formel) n'est pas invariante, même à une constante additive près. C'est bien la mise au point de la théorie de la calculabilité qui a rendu mathématisable l'idée intuitive et naturelle que simple peut signifier "définissable en peu de symboles" et complexe "impossible à définir brièvement".

Donnons des exemples pour éclairer la définition.

(a) *Une suite répétitive élémentaire.*

La complexité de Kolmogorov d'une suite s, composée d'un milliard de '0' (ou d'une image toute blanche) est très petite. En effet, écrire un programme du type :

```
pour i variant de 1 jusqu'à 1000000000 imprimer 0
```

(programme traduisible dans tout langage de programmation) prend peu de symboles. Puisque la longueur d'un tel programme est de quelques dizaines de bits, 200 tout au plus, cela signifie que : $K(s) < 200$.

Une suite binaire de moins de 200 bits définit une suite binaire d'un milliard de bits : la représentation par un programme (vue comme une définition) fait gagner un espace considérable. La définition de s par programme a une longueur inférieure à 0,00002% de la longueur de l'objet s représenté. Un milliard de fois '0' est une longue suite, mais conformément à ce qu'on attend, c'est une suite simple.

(b) *Un milliard de chiffres binaires de π.*

La complexité de Kolmogorov de la suite s composée d'un milliard de chiffres binaires de π est relativement faible. On peut en effet calculer s avec un programme assez court en utilisant une des nombreuses formules de série que l'analyse mathématique propose pour définir ou calculer π. Très concrètement, ce que nous savons de π montre que :

$$K(s) < 2000.$$

Cette seconde suite est donc un peu moins simple que la première, mais au sens de la complexité de Kolmogorov, elle reste particulièrement simple puisque sa complexité de Kolmogorov est inférieure à 0,0002% de sa longueur.

Voici un programme en langage C de 158 caractères qui engendre 1600 décimales de π :

```
int a=10000,b,c=8400,d,e,f[8401],g;main(){for(;b-c
;)f[b++]=a/5;for(;d=0,g=c*2;c-=14,printf("%.4d",e+
d/a),e=d%a)for(b=c;d+=f[b]*a,f[b]=d%--g,d/=g--,--b
;d*=b);}
```

En Haskell le programme suivant donne autant de digits que la mémoire de la machine le permet :

```
pi = g(1,0,1,1,3,3) where
  g(q,r,t,k,n,l) = if 4*q+r-t<n*t
```

```
then n : g(10*q,10*(r-n*t),t,k,div(10*(3*q+r))t-10*n,l)
else g(q*k,(2*q+r)*l,t*l,k+1,div(q*(7*k+2)+r*l)(t*l),l+2)
```

Ce programme, dû à Jeremy Gibbons [25], comporte 200 caractères (en comptant les blancs et les retours à la ligne).

C'est donc bien un programme de moins de 2000 chiffres binaires qu'il faut pour calculer un milliard de chiffres binaires de π. Le programme obtenu par ce moyen n'est pas nécessairement le plus court possible permettant d'avoir un milliard de chiffres binaires de π, mais sa longueur est plus grande que celle du programme le plus court qui sert à définir K(s).

(c) *Une suite de chiffres au hasard.*

Une suite de chiffres binaires tirés au hasard — par exemple en lançant une pièce de monnaie non truquée — a toutes les chances d'être complexe au sens de Kolmogorov, c'est-à-dire d'avoir une complexité de Kolmogorov élevée. En effet, un argument de dénombrement montre précisément que :

parmi toutes les suites binaires de longueur n (n fixé), moins d'une suite sur 2^k a une complexité de Kolmogorov inférieure à $n-k$.

On note que le nombre de programmes corrects (c'est-à-dire dont la syntaxe est conforme aux exigences du langage choisi pour définir K) de longueur inférieure à $n-k$ est au plus 2^{n-k}.

Il y a au plus 2 programmes corrects de longueur 1, 4 de longueur 2, 8 de longueur 3, ..., 2^{n-k-1} de longueur $n-k-1$, et la somme de ces nombres vaut $2^{n-k}-2$. En conséquence, la proportion de suites de longueur n possédant un programme minimal de longueur inférieure à $n-k$ est inférieure à :

$$\frac{2^{n-k}}{2^n} = \frac{1}{2^k}.$$

Un peu plus de précision est même possible : une suite binaire s de longueur n, prise au hasard, possède, avec une probabilité très forte, une complexité de Kolmogorov, K(s), d'environ n.

En effet :

(1) d'une part, le résultat précédent montre que K(s), sauf cas exceptionnels (exponentiellement rares), ne sera pas nettement inférieur à n, et

(2) d'autre part, les programmes du type `imprimer "001001001..."` (leur longueur est approximativement n) montrent que jamais K(s) n'est sensiblement supérieur à n, la longueur de s.

Le programme minimal d'une suite s peut être considéré comme une version compressée de s et en pratique ce sont d'ailleurs les algorithmes de compression de données sans perte qui constituent les meilleurs outils d'évaluation de K(s).

(d) *Un exemple mixte.*

Voici un quatrième exemple qui fera comprendre un peu plus en profondeur ce qu'est la complexité de Kolmogorov. Il s'agit d'un exemple où l'aléatoire et l'organisation sont mélangés.

On prend une suite générée aléatoirement de 500 millions de chiffres binaires :

$$s = 01101010100001010000 1000\ldots$$

D'après ce que nous avons vu, sa complexité de Kolmogorov est environ 500 millions. On transforme cette suite de 500 millions de chiffres binaires en une suite s' de 1 milliard de chiffres binaires en doublant chaque 0 et en doublant chaque 1 :

$$s' = 00111100110011001100000000110011000000001100000 0\ldots$$

La suite s commence par 011, la suite s' commence donc par 001111.

La complexité de Kolmogorov de cette suite d'un milliard de chiffres binaires est à peu près 500 millions. En effet :

– Elle ne peut pas être moins, car si on disposait d'un programme de moins de 500 millions de bits produisant s', on pourrait en le modifiant légèrement obtenir un autre programme de moins de 500 millions de bits environ produisant s (le nouveau programme P' serait la copie du premier P, mais n'imprimerait qu'un digit sur deux, moyennant une petite modification qui n'en change que très peu la longueur). Or cela n'est pas possible car cela signifierait que sa complexité de Kolmogorov est inférieure à 500 millions, ce qui n'est pas le cas.

– Elle ne peut pas être beaucoup plus, car en mettant dans un programme la donnée des 500 millions de bits de s et en plaçant quelques instructions pour que le programme recopie chaque bit de s deux fois de suite, on a un programme de longueur 500 millions environ qui produit s'.

Ici une structure simple — le doublement de chaque digit —, mélangée à de l'aléatoire donne une suite de longueur 1 milliard, ayant une complexité de 500 millions. D'une manière générale, toute régularité (statistique, algébrique, arithmétique, etc.) dans une suite de longueur n, fait rapidement baisser sa complexité de Kolmogorov.

Trouver des programmes minimaux est très difficile et on montre que la complexité de Kolmogorov n'est pas calculable par algorithme. Cependant tout algorithme de compression de données peut être vu comme un calculateur approché de la complexité de Kolmogorov. S'il repère bien les régularités du fichier numérique qu'on lui soumet et les exploite au mieux la taille du fichier qu'il produit est une approximation satisfaisante de la complexité de Kolmogorov. Bien sûr, aucun algorithme de compression ne repère toutes les régularités possibles qu'on peut trouver

dans un fichier et le résultat d'incalculabilité mentionné signifie précisément qu'aucun algorithme ne réussira jamais à repérer toutes les régularités qu'on peut mettre dans une suite ou un fichier.

La méthode de calcul de la complexité de Kolmogorov par les algorithmes de compression de données — et plus spécialement la mesure de la complexité de Kolmogorov relative — a conduit à de nombreuses applications : classification de textes [14], de musiques, reconstitution d'arbres phylogénétiques [44, 29], détection de spam [10], étude des séries de cours boursiers, repérage de flux anormaux d'informations et d'intrusions, etc.

La complexité de Kolmogorov est très ambitieuse puisqu'elle revient à prendre en compte toute régularité présente dans un fichier (et pas seulement certaines régularités statistiques, comme le fait l'entropie de Shannon), on dit qu'elle est *universelle*. Cela a une double conséquence.

(1) Elle est impossible à calculer exactement pour tout fichier numérique (ce qui ne l'empêche pas d'être approchable). C'est ennuyeux mais pas si grave comme l'attestent les mises en œuvre pratiques nombreuses à l'aide d'algorithmes de compression.

(2) Elle n'a pas les défauts de toutes les mesures combinatoires, statistiques mentionnées précédemment qui ne captent qu'une partie — souvent assez réduite — de la notion générale de complexité en n'envisageant que des catégories limitées de régularités.

Les nombreuses applications mises en œuvre récemment prouvent que l'avantage de l'universalité, associée à l'effectivité des algorithmes de compression de données — qui sont de petites machines intelligentes travaillant à repérer des régularités — est aujourd'hui en train de l'emporter sur les conceptions ad hoc, souvent étroites et toujours faciles à prendre en défaut, qu'on a utilisées sans, bien souvent, en comprendre les insuffisances graves.

Cependant la complexité de Kolmogorov est une mesure de la complexité au sens de "contenu incompressible d'information". Elle n'est pas une mesure de "richesse en structure". D'ailleurs les suites aléatoires qui n'ont aucune structure sont pour la complexité de Kolmogorov les plus complexes.

C'est qu'en fait, il existe deux notions intuitives très différentes de complexité que le langage courant confond et associe à tort.

– Il y a une notion de complexité comme richesse en information que la variété combinatoire — ou mieux, l'incompressibilité — mesure et dont la version ultime est la complexité de Kolmogorov.
– Il y a une notion de complexité comme richesse en structure, comme quantité d'organisation, comme difficulté à être élaboré ou calculé, comme information de valeur. Cette notion est plus ou moins indépendante de la première

notion. C'est cette seconde notion qu'on évoque — sans nécessairement en avoir conscience — quand on parle de complexification.

La première notion est clairement celle que la complexité de Kolmogorov formalise et on peut considérer que le succès de cette formalisation est total. La seconde notion que pour l'instant nous n'avons évoquée que de manière informelle peut-elle être définie rigoureusement dans un cadre mathématique ? Si on y arrive, cela donnera une base mathématique à l'idée de la complexification. La réponse est que la profondeur logique de Charles Bennett (et une série de notions plus ou moins liées) se présentent comme de telles mesures de richesse, de valeur, d'utilité de l'information. Nous en donnons une présentation rapide plus loin.

L'image souvent défendue de l'univers passant d'un désordre total (complexité aléatoire maximum) à un ordre de plus en plus grand et donc à une forme de mort (parfois appelée "mort thermique") par uniformisation est une image fausse. L'univers évolue en créant, par des processus assimilables à du calcul, des objets de plus en plus complexes, mais la complexité dont il s'agit alors est celle liée à l'organisation, à la variété et la richesse des structures. Il n'y a pas deux pôles opposés dont les extrêmes seraient "l'aléa" et "l'uniforme" (la complexité aléatoire et la simplicité totale), mais trois pôles dont les points extrêmes sont "l'aléa", le "simple absolu ou le rien", et le "très organisé" (probablement sans maximum).

La profondeur logique de Bennett

Formulée en 1977 par Charles Bennett mais rendue précise en 1988 [11] la proposition de définition d'une mesure de la complexité structurelle se fonde sur l'idée d'un "contenu en calcul", qui s'oppose à l'idée d'un "contenu en information". Pour mesurer ce contenu en calcul, l'idée naturelle de considérer le temps de calcul du programme le plus rapide donnant s ne conduit qu'à une notion triviale (car ne dépendant pour l'essentiel que de la longueur de s). Il faut être un peu plus subtil et ne considérer que le programme minimal (ou les programmes courts dans les versions paramétrées de la notion).

La *profondeur logique* (ou *profondeur de Bennett*), $\mathrm{P}(s)$, de la suite binaire finie s est définie par :

$$\mathrm{P}(s) = \textit{temps de calcul du programme minimal } s^* \textit{ de } s.$$

L'unité de mesure de ce temps de calcul est le pas élémentaire de calcul pour le modèle de machine que l'on considère, ou une unité de temps d'exécution d'un programme si on se réfère à $\mathrm{K}(s)$ en se basant sur des langages universels de programmation.

L'article principal sur le sujet est celui publié en 1988 par Charles Bennett [11], mais on lira avec intérêt les autres textes de Bennett [12, 13]. L'article de 1988 donne les détails de la théorie de la profondeur logique. Il explore non seulement la définition donnée ci-dessus, mais plusieurs variantes.

La définition de profondeur logique proposée par Bennett est-elle satisfaisante et doit-on croire qu'on dispose avec elle d'une bonne mesure de complexité organisée (ou complexité structurelle), complémentaire de la notion de complexité de Kolmogorov qui, on l'a vu, mesure la complexité aléatoire ? La discussion est délicate, les arguments nombreux et variés et la conclusion n'est pas considérée unanimement comme acquise.

Commençons notre examen en reprenant l'exemple de la suite aléatoire ($K(s) \approx$ longueur(s)) qui nous a conduit à conclure que la complexité de Kolmogorov n'est pas le bon concept de complexité organisée : une suite aléatoire n'est pas organisée et possède pourtant une forte complexité de Kolmogorov. Que donne la profondeur logique de Bennett pour une telle suite s, obtenue par tirage au sort ?

Une telle suite, s, possède par définition une complexité de Kolmogorov $K(s)$ équivalente à sa longueur. Il en résulte que son programme minimal est sans doute équivalent à un programme du type "imprimer s" ou est proche d'un tel programme. Or, pour toute machine raisonnable et pour tout langage de programmation raisonnable exécuté par une telle machine, le temps d'exécution d'un programme du type "imprimer s" sera linéaire ou quasi-linéaire, c'est-à-dire petit (puisque le temps de calcul d'une suite de longueur n ne peut pas être inférieur à n en nombre de pas de calcul). Autrement dit, une suite aléatoire possède toujours une faible profondeur logique de Bennett : lorsqu'une suite n'a pas de structure et est totalement désordonnée, il n'y a rien de mieux à faire que d'écrire s dans le programme qui doit la produire (qui est donc du type "imprimer s") et alors, un tel programme fonctionne rapidement. Nous attendions cette propriété qui faisait défaut à la complexité de Kolmogorov.

Il ne s'agit pas là d'un argument définitif, mais il est particulièrement frappant que le principal problème de la complexité de Kolmogorov (elle est maximale pour les suites aléatoires pourtant sans organisation), défaut qui conduit à l'écarter comme mesure de complexité structurelle, disparaisse quand on prend en considération la profondeur logique de Bennett : la profondeur logique d'une suite aléatoire est minimale comme elle l'est pour une suite simple (par exemple composée uniquement de '0').

Une analyse heuristique de la profondeur de Bennett nous conduit aussi à la conclusion que, plus un objet est finement structuré, plus le temps nécessaire au passage de sa description la plus compressée (son programme minimal) à l'objet lui-même sera long. De ce fait — et dans le cas des chiffres de π les arguments peuvent être rendus précis — la profondeur logique de Bennett semble bien convenir : très faible pour une longue suite répétitive de 0 ou une suite incompressible ($K(s)$ très faible ou $K(s)$ maximum) ; moyenne pour une suite comme celle des chiffres de π (ayant un certain contenu en calcul) ; elle est élevée dans le cas d'un objet fortement structuré ou possédant une organisation complexe. C'est exactement ce que l'on attendait.

Schématiquement, il y aurait donc quatre sortes d'objets :

(1) les objets K-simples et peu profonds : une suite uniforme de '0', une suite répétant un même motif (cristal), etc. ;

(2) les objets K-simples et profonds : la suite des chiffres de π, la suite de Prouhet-Thue-Morse, etc. ;

(3) les objets aléatoires et peu profonds : les suites typiques de longueur n ;

(4) les objets aléatoires et profonds : microprocesseurs, ordinateurs, cellules vivantes, villes, organisations sociales, écosystèmes, etc.

Progrès récents

Bien qu'assez lentement, une série d'avancées dans la compréhension et l'utilisation de la profondeur logique de Bennett ont eu lieu depuis sa définition en 1977. Il faut les voir comme la confirmation que l'idée de Bennett est bonne et que, même si rien n'est très facile dans ce domaine soumis à des nombreuses "indécidabilités", il faut poursuivre son étude qui de toutes les façons, ne serait-ce que d'une manière abstraite, nous aide à saisir la nature de la complexité dont notre compréhension intuitive doit maintenant devenir mathématique et scientifique.

Des versions mathématiques plus avancées et plus précises de la notion de profondeur logique de Bennett ont été proposées par Bennett [11]. Ces versions ne se contentent pas de faire intervenir le programme le plus court produisant s, mais prennent en compte tous les programmes produisant s qui ne peuvent être compressés de plus de k digits ou même tous les programmes produisant s mais en accordant plus de poids aux programmes les plus courts qui sont considérés comme des origines plus probables de l'objet s. Ces versions possèdent de meilleures propriétés que P(s) dont celle d'autoriser la démonstration d'une "loi de croissance lente" qui indique que la complexité structurelle ne peut pas croître rapidement. Malheureusement, ces définitions plus satisfaisantes sur un plan abstrait sont plus délicates à mettre en application que la définition de base que nous avons notée P(s).

Parmi les tentatives de définition d'une mesure de complexité structurelle, certaines sont issues de la théorie du calcul tout en étant différentes de celle de Bennett. Ces autres notions sont en compétition avec l'idée de Bennett et il est donc nécessaire de comprendre s'il s'agit de notions liées les unes aux autres. La notion de "sophistication" introduite par Moshe Koppel et Henri Atlan [31, 32, 33], la notion de "complexité effective" de Seth Lloyd et Murray Gell-Mann [24] et la notion de "facticité" de Adriaans [4] se fondent toutes les trois sur l'idée d'une séparation possible entre la composante aléatoire et la composante organisée d'un objet. L'idée des définitions proposées par ces chercheurs sous des formes mathématiques qui en masquent parfois le côté naturel est la suivante. Quand on examine un programme produisant un objet, il est possible — au moins dans certain cas — d'y reconnaître

deux parties : l'une représente ce qui fixe et décrit la structure de l'objet, l'autre est ce qui habille la structure par des éléments de moindre importance ou même aléatoires.

Considérons par exemple une suite finie s de paires de chiffres binaires 00 ou 11 dont la succession est aléatoire :

$$s = \texttt{00 11 11 00 00 00 11 00 11 00 00 11 00 11 11}$$

Le plus court programme s^* produisant cette suite comportera deux parties assez faciles à identifier, car il sera de la forme :

```
imprimer en doublant chaque chiffre la suite 011000101001011.
```

La structure régulière de s est exprimée par la partie "imprimer en doublant chaque chiffre". La composante désordonnée se trouve quant à elle dans la partie descriptive et incompressible de 0 et de 1 qui termine le programme.

Lorsqu'on aura affaire à une suite s plus structurée que celle de notre exemple, la longueur de la partie non aléatoire sera plus grande. C'est cette longueur qui, selon Koppel, Atlan, Lloyd et Gell-Mann, Adriaans, mesure la richesse en organisation — la complexité structurelle — de la suite s.

La partie du programme où on a inscrit la séquence sans doubler les digits `011000101001011` est d'autant plus longue que la séquence s comporte beaucoup d'éléments. C'est la partie aléatoire de la séquence et sa longueur est en gros la complexité de Kolmogorov de s.

Des difficultés techniques rendent délicate la formulation mathématique de cette idée de séparation en deux parties d'un programme court donnant s, mais ce que nous venons d'expliquer est l'essence de l'idée de la sophistication, de la complexité effective et de la facticité. Puisque ces définitions sont aussi naturelles sans doute que la définition de la profondeur logique de Bennett, la question était posée de savoir si un lien existe entre ces deux visions, en apparence assez différentes, de la complexité structurelle. Un début de réponse a été donné en 2010 dans un remarquable article [8] de Nihat Ay, Markus Müller, et Arleta Szkola du Max Plank Institute de Leipzig. Les chercheurs y établissent formellement qu'une grande complexité effective entraîne toujours une grande profondeur logique, confirmant indirectement le bien fondé de la définition de Bennett.

Les recherches récentes ont conduit à un second progrès de la théorie qui provient de la mise au point de versions calculables de la profondeur logique [6, 7, 22, 34]. Ces versions font des hypothèses simplificatrices et restrictives sur les processus de calcul, mais aboutissent à des définitions débarrassées de la gênante incalculabilité de P(s).

Autre motif de satisfaction pour ceux qui pensent que le concept de profondeur logique de Bennett ou les concepts voisins sont une clef pour comprendre le monde réel a été de le voir reconnu dans des travaux scientifiques liés à ce qui dans le monde est une sorte de grand calcul : l'évolution biologique. Antoine Danchin [18] de l'Institut Pasteur, Guillaume Baptist [9] de l'Université de Grenoble et John Collier [15]

de l'université de Durban en Afrique du Sud défendent tous les trois indépendamment l'un de l'autre l'idée que la profondeur logique de Bennett est un concept utile et important en biologie. En effet, elle permet d'éviter certaines considérations trop naïves sur la complexification des êtres vivants et elle éclaire l'idée d'un progrès dans les lignées d'organismes vivants — progrès qui ne serait que l'accumulation de résultats de calculs.

Expérimentation

Parmi les dernières avancées dans la démonstration que la profondeur logique de Bennett est le bon concept mathématique permettant de définir et de mesurer la complexité structurelle, divers travaux de nature expérimentale ont conduit à des débuts de validation pratique de l'idée de Bennett dont les fondements intuitifs ne font pas de doute, mais dont il restait à montrer qu'elle possède aussi un potentiel d'application.

Un premier travail [45] a consisté à mesurer vraiment le temps de calcul du plus court programme produisant une image. Mené par Hector Zenil de l'Université de Sheffield, Cedric Gaucherel du Département d'Ecologie de l'Institut Français de Pondichéry en Inde et moi-même, la méthode évalue le temps de décompression nécessaire pour des images représentant divers objets plus ou moins structurés. Ce temps de décompression est assimilable à $P(s)$ car le fichier compressé de s est une version approchée de s^* et il en résulte que le calcul pour passer du fichier compressé au fichier complet est assimilable au calcul qui s'effectue à partir de s^* pour revenir à s, c'est-à-dire $P(s)$. Les résultats de ces mesures de temps de calcul ont été utilisés pour ordonner les images par complexité structurelle croissante. La classification obtenue est proche de celle attendue. Elle place au début du classement les images parfaitement simples ou totalement aléatoires, et à la fin du classement les images des objets les plus riches en structures.

Un second travail [42], fondé sur l'énumération massive de petites machines de Turing, a été mené par Fernando Soler-Toscano de l'Université de Séville, Hector Zenil, Nicolas Gauvrit de l'Université de Paris VII et moi-même. Il a permis d'évaluer pour des courtes séquences de '0' et de '1', leur complexité de Kolmogorov et leur profondeur logique de Bennett. Ces calculs montrent, comme on s'y attendait, que les deux notions de complexité sont indépendantes et que la profondeur logique de Bennett est faible aussi bien pour les suites simples (10 fois '0' par exemple) que pour les suites d'apparence aléatoire. À nouveau la coïncidence entre théorie et calcul approché montre que malgré l'indécidabilité théorique des notions, on peut tenter d'en faire quelque chose, et que ce qu'on trouve alors est conforme à ce que l'intuition attend. Ces expérimentations et applications à la classification des objets numériques selon leur complexité structurelle ne sont que balbutiantes, mais pourraient se révéler réellement utiles, pourvu que l'idée qu'il y a deux complexités bien

différentes (liées respectivement au *contenu en information* et au *contenu en calcul*) fasse son chemin, et que dans toutes les disciplines où on les rencontre, on cherche à les mesurer par les techniques de compression sans perte. Ces dernières sont susceptibles de progrès sans limite (on n'aura jamais d'outil parfait de détection des régularités, mais en disposant de capacité de calcul croissante on réussira de mieux en mieux) et devraient donc conduire à des outils de mesure de plus en plus fins et robustes des deux complexités identifiées et formalisées par la théorie de la calculabilité.

On le voit la définition et la mesure de la complexité sont des sujets difficiles dont on a sans doute pas encore identifié toutes les clefs. Les progrès récents sont cependant remarquables sur tous les aspects du problème :

(1) démonstration de la complexité (combinatoire) des développements décimaux de certains nombres réels ;

(2) meilleure maîtrise des générateurs de suites pseudo-aléatoires, grâce aux outils évaluant concrètement la complexité de leurs productions ;

(3) définition d'un notion universelle de complexité aléatoire (la complexité de Kolmogorov) et mise au point d'outils en permettant la mesure et l'utilisation ;

(4) identification d'une dualité longtemps restée confuse entre complexité aléatoire et complexité structurelle avec propositions de définitions mathématiques précises (dont la profondeur logique de Bennett) ;

(5) résultats et méthodes de calcul approché en cours de mise au point pour cette notion dont l'importance conceptuelle et pratique est devenue évidente.

Références

Il existe des centaines d'articles et de livres abordant le thème de la définition et de la mesure de la complexité. Nous nous sommes astreints à n'en mentionner qu'un peu moins d'une cinquantaine en choisissant ceux directement évoqués dans l'article ou ceux les plus utiles. Dans le texte de notre article, certaines notions et propositions de mesures de complexités que nous considérons comme trop liées à un domaine particulier, mal fondées ou conduisant à des impasses n'ont pas été mentionnées du tout. Sans pouvoir prétendre ici à l'exhaustivité (impossible sur ce sujet) nous indiquons quelques références concernant ces notions et autres propositions de mesure de complexité [2, 3, 23, 36].

[1] B. Adamczewski, Y. Bugeaud, On the Complexity of Algebraic Numbers I. Expansions in Integer Bases, *Annals of Mathematics, Second Series*, 165(2), 547–565, 2007.

[2] C. Adami, N. Cerf, Physical Complexity of Symbolic Sequences, *Physica D* 137, 62—69, 2000.

[3] C. Adami, Sequence Complexity in Darwinian Evolution, *Complexity* 8(2), 49–56, 2003.

[4] P. Adriaans, Facticity as the Amount of Self-Descriptive Information in a Data Set, arxiv.org/abs/1203.2245 , 2012.

[5] R. Ahlswede, L. Khachatrian, C. Mauduit, A. Sárközy, A Complexity Measure for Families of Binary Sequences, *Periodica Mathematica Hungarica* 46(2), 107–118, 2003.

[6] L. Antunes, L. Fortnow, D. van Melkebeek, N. Vinodchandran, Computational Depth : Concept and Applications, *Theor. Comput. Sci.* 354(3), 391–404, 2006.

[7] L. Antunes, A. Matos, A. Souto, P. Vitányi, Depth as Randomness Deficiency, *Theory Comput. Syst.* 45, 724–739, 2009.

[8] N. Ay, M. Müller, A. Szkola, Effective Complexity and its Relation to Logical Depth, *IEEE Trans. Inform. Theory* 56, 4593–4607, 2010.

[9] G. Baptiste, Réseau de régulation génique chez Escherichia coli, Thèse Université de Grenoble, 2012.

[10] S. Belabbes, G. Richard, On Using SVM and Kolmogorov Complexity for Spam, Filtering Proceedings of the Twenty-First International FLAIRS, 2008.

[11] C. Bennett, Logical Depth and Physical Complexity, In : *A half-Century Survey on the Universal Turing Machine*, 227-257, Oxford University Press, Inc., New York, 1988.

[12] C. Bennett, How to Define Complexity in Physics, and Why, in W. H. Zurek (ed.), *Complexity, Entropy, and the Physics of Information*, 1991.

[13] C. Bennett, What Increases When a Self-organizing System Organizes Itself ? Logical Depth to the Rescue, `http://dabacon.org/pontiff/?p=5912`, 2012.

[14] R. Cilibrasi , P. Vitányi, Clustering by compression, *IEEE Transactions on Information Theory*, 2005.

[15] J. Collier, Information in Biological Systems, *Handbook of Philosophy of Science*, V8, Philosophy of Information, eds. Pieter Adriaans and Johan van Benthem, Dordrecht, Elsevier, 763-787, 2008.

[16] T. Cover, J. Thomas, *Elements of Information Theory*, Wiley-Interscience, 2006.

[17] G. Chaitin, *Algorithmic Information Theory*, Cambridge, Cambridge University Press, 1987.

[18] A. Danchin, *La Barque de Delphes. Ce que révèle le texte des génomes*, Ed. Odile Jacob, Paris, 1998.

[19] J.-P. Delahaye, *Information, complexité et hasard*, Hermès, Paris, 1994, 1999.

[20] J.-P. Delahaye, *Complexité aléatoire et complexité organisée*, Editions Quae, 2009.

[21] J.-P. Delahaye, H. Zenil, Numerical Evaluation of Algorithmic Complexity for Short Strings : A Glance into the Innermost Structure of Randomness, *Applied Mathematics and Computation* 219, 63–77, 2012.

[22] D. Doty, P Moser, *Feasible Depth, Computation and Logic in the Real World*, 2007.

[23] D. Feldman, J. Crutchfield, Measures of Statistical Complexity : Why ?, *Physics Letters A* 238, 244–252, 1998.

[24] M. Gell-Mann, S. Lloyd, Information Measures, Effective Complexity, and Total information, *Complexity* 2, 44-–52, 1996.

[25] J. Gibbons, Unbounded Spigot Algorithms for the Digits of Pi, *The American Mathematical Monthly* 113(4), 318–328, 2006.

[26] P. Grassberger, Randomness, Information, and Complexity, arxiv.org/abs/1208.3459, 2012.

[27] P. Grunwald, P. Vitányi, Shannon Information and Kolmogorov Complexity, arXiv :cs/0410002v1, 2004.

[28] D. Juedes, J. Lathrop, J. Lutz, Computational Depth and Reducibility, *Theor. Comput. Sci.* 132, 37-–70, 1994.

[29] A. Kaltchenko, Algorithms for Estimating Information Distance with Application to Bioinformatics and Linguistics, *Electrical and Computer Engineering*, 2004.

[30] A. Kolmogorov, Three Approaches to the Definition of the Concept Quantity of information, *Probl. Peredachi Inf.* 1(1), 3-–11, 1965.

[31] M. Koppel, Complexity, Depth, and Sophistication, *Complexity* 1, 1087–1091, 1987.

[32] M. Koppel, Structure, In : *A half-Century Survey on the Universal Turing Machine*, 235–252, Oxford University Press, Inc., New York, 1988.

[33] M. Koppel, H. Atlan, An Almost Machine-independent Theory of Program-length Complexity, Sophistication, and Induction, *Information Sciences* 56(1), 23–33, 1991.

[34] J. Lathrop, J. Lutz, Recursive Computational Depth, *Information and Computation* 153(2), 139-–172, 1999.

[35] M. Li, P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, Third Edition, Springer Verlag, 2008.

[36] S. Lloyd, H. Pagels, Complexity as Thermodynamic Depth, *Annals of Physics* 188, 186–213, 1988.

[37] J. McAllister, Effective Complexity as a Measure of Information Content, *Philosophy of Science* 70, 302–307, 2003.

[38] M. Mitchell, *Complexity Guided Tour*, Oxford University Press, Chapitre 7, 2009.

[39] H. Reeves, *Dernières nouvelles du cosmos*, Éditions du Seuil, Paris, 2002.

[40] A. Rukhin et al., A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications, *NIST Special Publication* 800(22), 2010.

[41] C. Shannon, W. Weaver, *The Mathematical Theory of Communications*, Univ. of Illinois Press, Urbana, 1949.

[42] F. Soler-Toscano, H. Zenil, J.-P. Delahaye, N. Gauvrit, Correspondence and Independence of Numerical Evaluations of Algorithmic Information Measures, arxiv.org/abs/1211.4891, 2012.

[43] A. Teixeira, A. Matos, A. Souto, L. Antunes, Entropy Measures vs. Kolmogorov Complexity, *Entropy* 13(3), 595–611, 2011.

[44] J.-S. Varré, E. Rivals, J.-P. Delahaye, The Transformation Distance : a Dissimilarity Measure Based on Movements of Segments, *Bioinformatics* 15(3), 194–202, 1999.

[45] H. Zenil, J.-P. Delahaye, C. Gaucherel, Image Characterization and Classification by Physical Complexity, *Complexity* 17(3), 26–42, 2012.