Définir et mesurer la complexité :

La théorie algorithmique de l'information

Jean-Paul Delahaye

Laboratoire d'informatique fondamentale de Lille (UMR CNRS 8022)

Fleurance, le 3 août 2013

Documents : **http://www2.lifl.fr/~delahaye/Complexe/**

L'univers se complexifie ?

L'évolution biologique passe de formes simples à d'autres plus complexes ?

Les objets technologiques (voitures, ordinateurs, téléphones, etc.) sont de plus en plus complexes ?

Les sciences proposent des théories de plus en plus complexes ?

Qu'est-ce que cette *complexité* ? Comment la mesurer ?

Vieilles questions !

Étonnant renouvellement des solutions proposées
(en particulier grâce à la théorie du calcul).

De nombreuses sciences rencontrent la ***complexité*** : oui.

Mais cela conduit-il à une science de la ***complexité*** ? C'est loin d'être évident.

Il semble quand même que la ***théorie de la calculabilité*** (née dans les années 1936 des travaux de Turing, Gödel, Church, Kleene) fournit un fondement **général et universel** à une théorie de la complexité : ***la théorie algorithmique de l'information*** (née vers 1965).

Elle concerne **toutes les disciplines**... mais elle ne prétend pas y résoudre tous les problèmes.

Il en va comme pour ***la géométrie et à la topologie*** qui fournissent un fondement **général et universel** à tout ce qu'on peut dire et penser sur l'espace (et les espaces) dans toutes les disciplines.

La différence vient de ce que ***la théorie algorithmique de l'information*** n'est pas aujourd'hui très connue et que les outils qui en permettent l'application commencent juste à être mis au point.

On se limitera aux objets numériques.

Est-ce que tout se ramène à eux ?

La suite de 50 chiffres

00000000000000000000000000000000000000000000000000

est plus simple que :

26535897932384626433832795028841971693993751058209

Le *simple* s'oppose au *complexe*.

Pourtant ***complexe*** signifie :

sans ordre, sans régularité, aléatoire

ou signifie :

organisé, fortement structuré, riche en information

Il y a deux types de complexité :

la ***complexité aléatoire*** (le désordre)

la ***complexité organisée*** (l'ordre non trivial et structuré)

La deuxième suite :

26535897932384626433832795028841971693993751058209

apparaît comme un exemple de **complexité aléatoire**.

Il s'agit en fait de la suite des décimales de π à partir de la sixième (les cinq premières sont 14159).

On doit donc la considérer comme un exemple de **complexité organisée :**

π est déterminé, unique et n'est sujet à aucune variation.

Certes, sa structure est cachée, mais il n'est en rien désordonné.

La **complexité aléatoire** n'est pas facile à distinguer de la **complexité organisée**.

Autre exemple de **complexité organisée** :

0110100110010110100101100110100110010110011010010110100110010110
1001011001101001011010011001011001101001100101101001... ... ...

C'est la suite de Thue-Morse. Elle est définie par :

0

01

0110

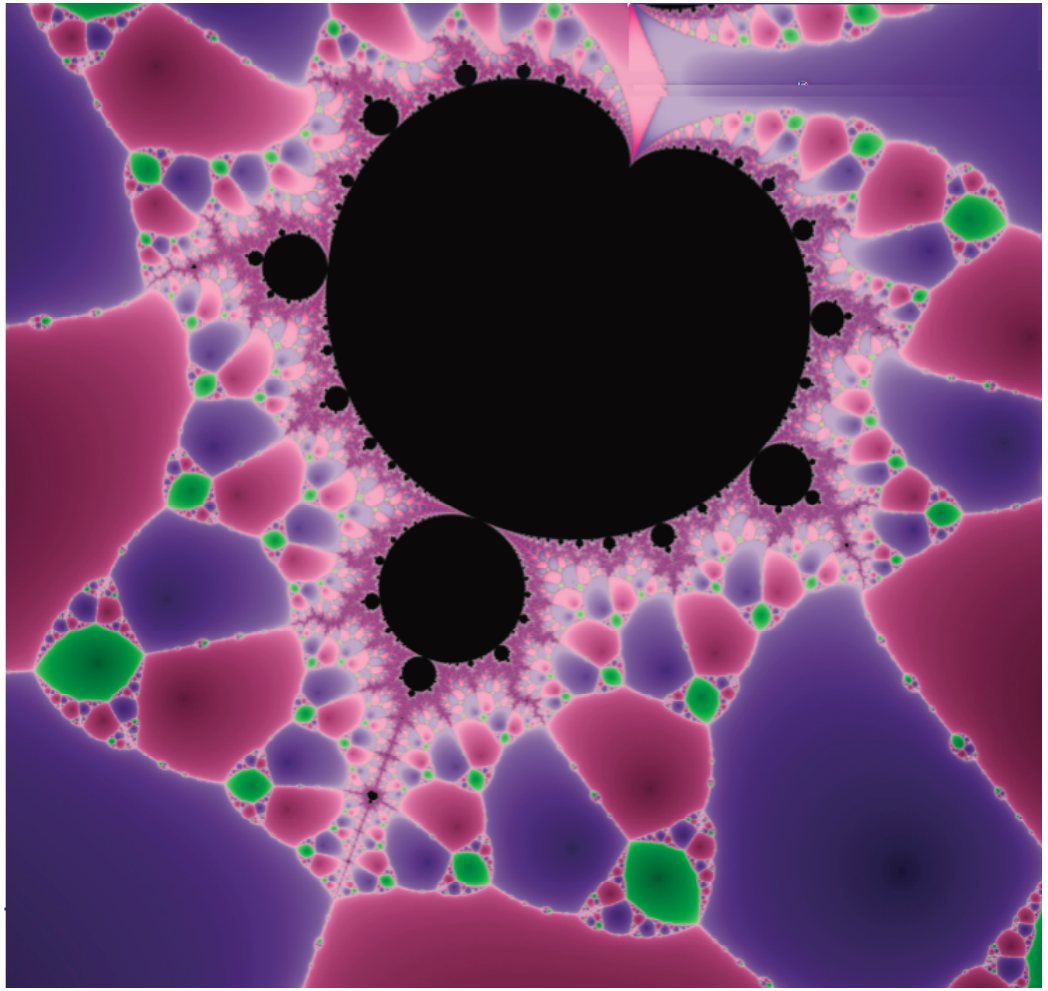
01101001

0110100110010110

....

$$tm(n+1) = tm(n)\ \overline{tm(n)}$$

Autre exemple de complexité organisée : un fractal (ou plutôt l'image d'un fractal)



Le concept de complexité aléatoire a été défini vers 1965 sous le nom de :

complexité de Kolmogorov

ou de ***Chaitin-Kolmogorov***, ou ***complexité algorithmique***

Définition.

> La **complexité de Kolmogorov**, K(*s*), d'une suite binaire finie ***s*** est :
>
> K(*s*) = *longueur du plus court programme* ***s**** *qui engendre* ***s***

Pour le plus court programme, ***s****, on parle aussi de ***programme minimal*** de ***s*.**

• Idée : «*est simple ce qui se décrit brièvement*».

• Pas de prise en compte du temps de calcul (... mais des variantes existent)

• Pourquoi se ramener à des suites finies de 0 et de 1 ?

En se plaçant au bon niveau de détail cela semble toujours possible (musique, image, cinéma, etc.)

• Dans la définition, on suppose que *s** est écrit lui-même en binaire.

• La définition est-elle **robuste** ?

Est-ce qu'en changeant de langage de programmation, on ne change pas la complexité mesurée ?

Théorème d'invariance

Si L et M sont deux langages universels, et si on note $K_L(s)$ (respectivement $K_M(s)$) la complexité de Kolmogorov quand on utilise L (respectivement M) comme langage de référence, alors il existe une constante C_{LM} telle que pour toute suite binaire finie $\boldsymbol{s}$:

$$| K_L(\boldsymbol{s}) - K_M(\boldsymbol{s}) | < C_{LM}$$

Idée : On utilise la possibilité d'écrire en L un interpréteur pour M, et réciproquement.

Exemple 1

Une suite d'un milliard de '0' est une suite simple :

$$s = 0\ 0\ 0\ \ldots\ 0 \qquad K(s) < 100$$

Programme pour s :

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

Exemple 2

La complexité de Kolmogorov de la suite des 10^9 premiers **chiffres binaires de π** est faible.

```
s = (1 1 .0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1
0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0 1
0 0 0 1 0 1 1 1 0 0 0 0 0 ...)
```

$$K(s) < 1000$$

Un programme C de 158 caractères qui donne 2400 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);}
```

$$\pi = \sum_{i=0}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!}.$$

Dik Winter and Achim Flammenkamp

15,000 decimal digits of π:

```
a[52514],b,c=52514,d,e,f=1e4,g,h;
main(){for(;b=c-=14;h=printf("%04d", e+d/f))
  for(e=d%=f;g=--b*2;d/=g)d=d*b+f*(h?a[b]:f/5),a[b]=d%--g;}
```

$$\pi = 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7}\left(\cdots\left(2 + \frac{i}{2i+1}\left(\cdots\right)\right)\right)\right)\right)$$

En Haskell (évaluation paresseuse, lambda-calcul, stream) :

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)
```

Exemple 3

Une suite tirée au hasard a toutes les chances d'avoir une forte complexité de Kolmogorov.

• Parmi toutes les suites de 0 et de 1 de longueur n, n fixé,

- moins d'une suite sur 1024 a une complexité < ***n* - 10**, c'est-à-dire peut être comprimée de plus de 10 digits ;

- moins d'une suite sur un million a une complexité < ***n* - 20**, c'est-à-dire peut être comprimée de plus de 20 digits.

- etc.

Raisonnement :

Les programmes de longueur i sont au plus 2^i.

Il y a donc au plus $1 + 2 + 4 + ... + 2^{k-1} = 2^k - 1 < 2^k$ programmes de longueur $< k$

Parmi les suites de longueur $\boldsymbol{n}$, il y en a donc moins de 2^{n-k} dont la longueur du programme minimal est $< n - k$

La proportion des suites binaires s de longueur n compressibles de k bits est donc au plus :

$$2^{n-k} / 2^n = 1 / 2^k$$

Une suite tirée au hasard est très rarement compressible.

Compression

- Si vous tirez au hasard une suite de caractères (parmi 256) et que vous appliquez un algorithme de compression de texte, vous n'obtiendrez rien (en fait, le texte compressé sera plus long !).

- Attention : si vous composez un texte avec des **0** et de **1** tirés au hasard, n'importe quel compresseur de texte le réduira d'environ 7/8 (car chaque 0 ou 1 est codé que 8 bits).

Les utilisateurs d'algorithmes de compression de textes constatent qu'on peut toujours comprimer un texte.

Ils peuvent avoir l'illusion que tout est compressible. C'est faux.

Les textes usuels ne sont (presque) jamais aléatoires et c'est pour cela qu'on peut les comprimer.

Exemple 4

On prend une suite tirée au hasard de **500 millions de chiffres binaires** :

`s = 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 ...`

et qu'on double chaque chiffre

`s' = 00111100111100111111000011001 1 ...`

La complexité de Kolmogorov de `s'` est d'environ 500 millions

Est-il est facile de calculer la complexité de Kolmogorov d'une suite ***s*** ?

Comment s'y prendre ?

Mauvaise nouvelle.

K(*s*) n'est pas calculable

En effet, la fonction ***s*** —> K(***s***) n'est pas récursive (c'est-à-dire pas calculable par algorithme)

• aucun algorithme ne peut, pour toute suite ***s*** qu'on lui fournit en données, calculer en temps fini le nombre entier K(***s***).

Pire !

- si T est une **théorie formelle** fixée et qu'elle ne démontre que des énoncés arithmétiques justes, alors elle ne démontre qu'un nombre fini d'énoncés de la forme :

$$\text{« } \mathbf{K}(s) = n \text{ »}$$

En particulier : il existe un certain k_T tel que la théorie **T** ne peut démontrer aucun résultat du type :

$$\text{« } \mathbf{K}(s) = n \text{ »} \quad \text{avec} \quad n \geq k_T.$$

*Une théorie formelle n'a accès qu'à un nombre fini de résultats du type «*la complexité de ***s*** est ***n****» ;*

- *tous sauf un nombre fini d'entre eux sont des* **indécidables au sens de Gödel** *pour la théorie.*

Bonne nouvelle.

K(*s*) est approchable

Il existe une suite d'algorithmes $\mathbf{K_0}, \mathbf{K_1}, \ldots, \mathbf{K_n}, \ldots$ tels que pour tout ***s*** :

la suite $\mathbf{K_0}(s), \mathbf{K_1}(s), \ldots \mathbf{K_n}(s), \ldots$ est décroissante et convergente vers $\mathbf{K}(s)$

Comme les $\mathbf{K_n}(s)$ et $\mathbf{K}(s)$ sont des entiers, cela signifie que pour tout ***s***, la suite des valeurs approchées $\mathbf{K_0}(s), \mathbf{K_1}(s), \ldots, \mathbf{K_n}(s), \ldots$ est du type :

143, 23, 22, 20, 20, 20, 15, 15, 15, 15, 15, 8, 8, 8, ... , 8, ...

stationnaire et égale à $\mathbf{K}(s)$ *à partir d'un certain point.*

• En pratique, pour évaluer **K(*s*)**, on utilise des algorithmes de compression (sans pertes).

La taille du fichier comprimé de *s* par un algorithme C de compression est une valeur approchée de K(*s*)

• L'indécidabilité a pour conséquence qu'on ne peut jamais être certain d'être proche de **K(*s*)** (une régularité *non vue* par l'algorithme de compression utilisé est peut-être présente dans ***s***)

Complexité de Kolmogorov des images.

Taille du fichier png d'une image.

Compression sans perte, donc approximation de K(image).

Images du volcan Bromo (Ile de Java, Indonésie).

Complexité relative et distance informationnelle

Soit *t* une suite binaire fixée :

K(*s* | *t*) =

longueur du plus court programme s* qui engendre s

en utilisant si nécessaire les données présentes dans *t*

K(***s*** | ***s***) ≈ 0

K(***s*** | ***t***) ≈ K(***s***) si t est une suite aléatoire n'ayant aucun rapport avec s

Plus ***t*** et ***s*** ont des contenus liés, plus K(***s*** | ***t***) est petit.

En utilisant des algorithmes de compression de données (sans pertes), on en déduit des **distances** (**pseudo-distances, mesures de similarité**) qui donnent d'intéressants résultats.

- **Arbres phylogénétiques** à partir de séquences génétiques (Varré, Delahaye, Vitanyi, Cilibrasi, ...);

- Arbre représentant les parentés entre les **langues indoeuropéennes** en partant des traductions de ***La déclaration universelle des Droits de l'Homme*** (Cilibrasi, Vitanyi) ;

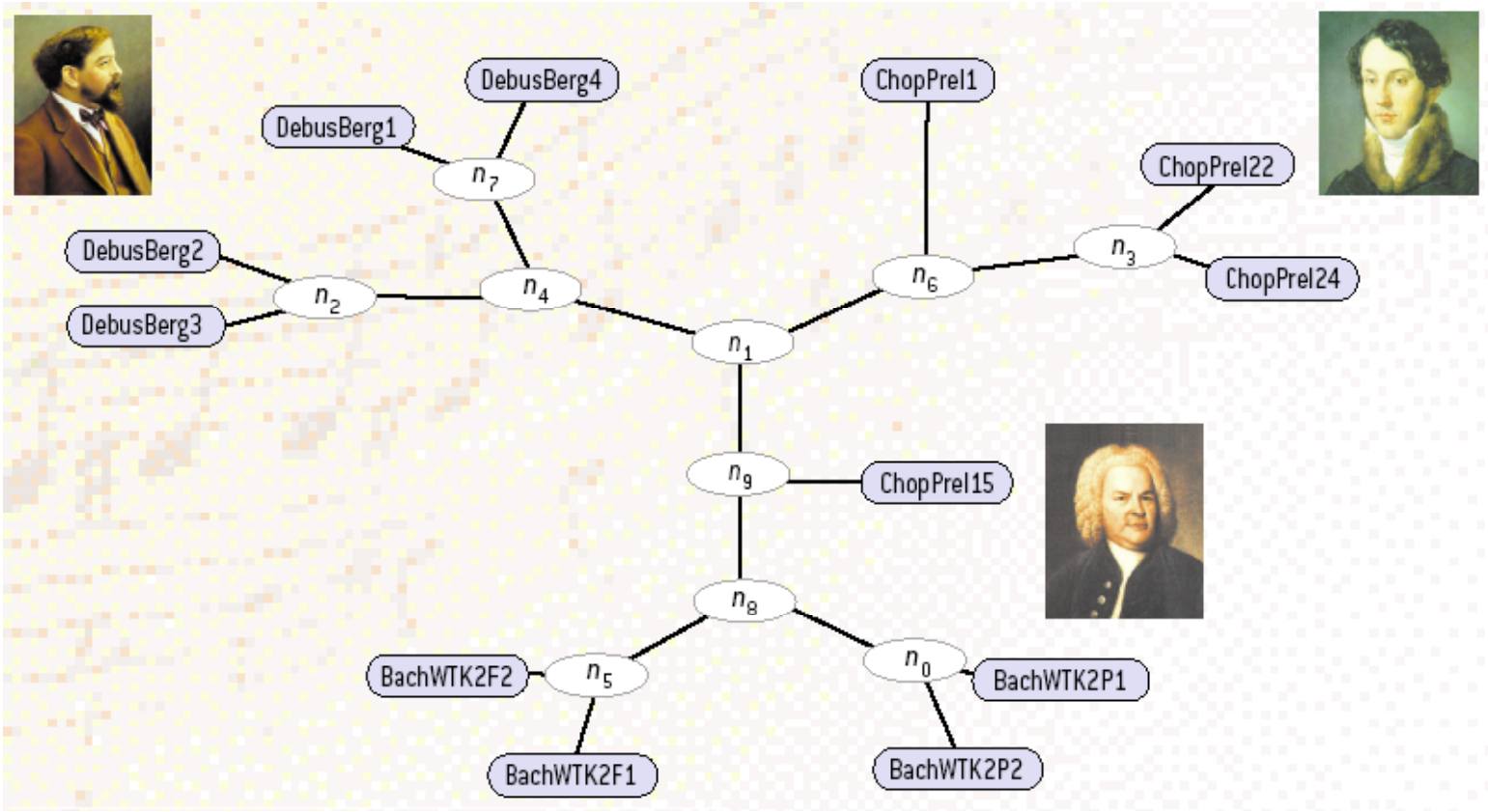
- Comparaison de textes littéraires (Cilibrasi, Vitanyi) ;

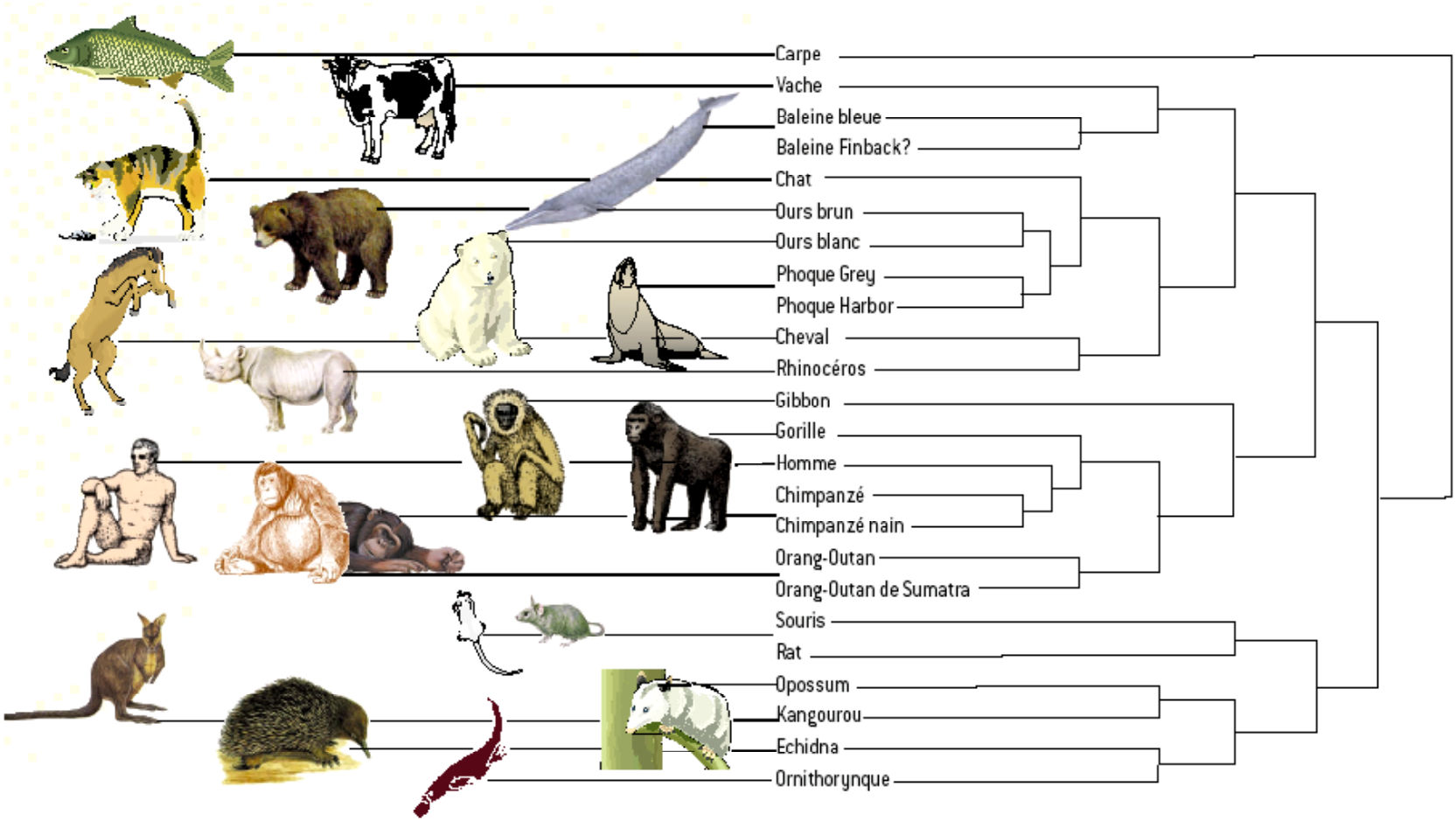
- Applications à la classification de **morceaux de musique** (Cilibrasi, Vitanyi, de Wolf) ;

- Applications au **traitement d'images et de séquences vidéo** (Leclercq, Delahaye).

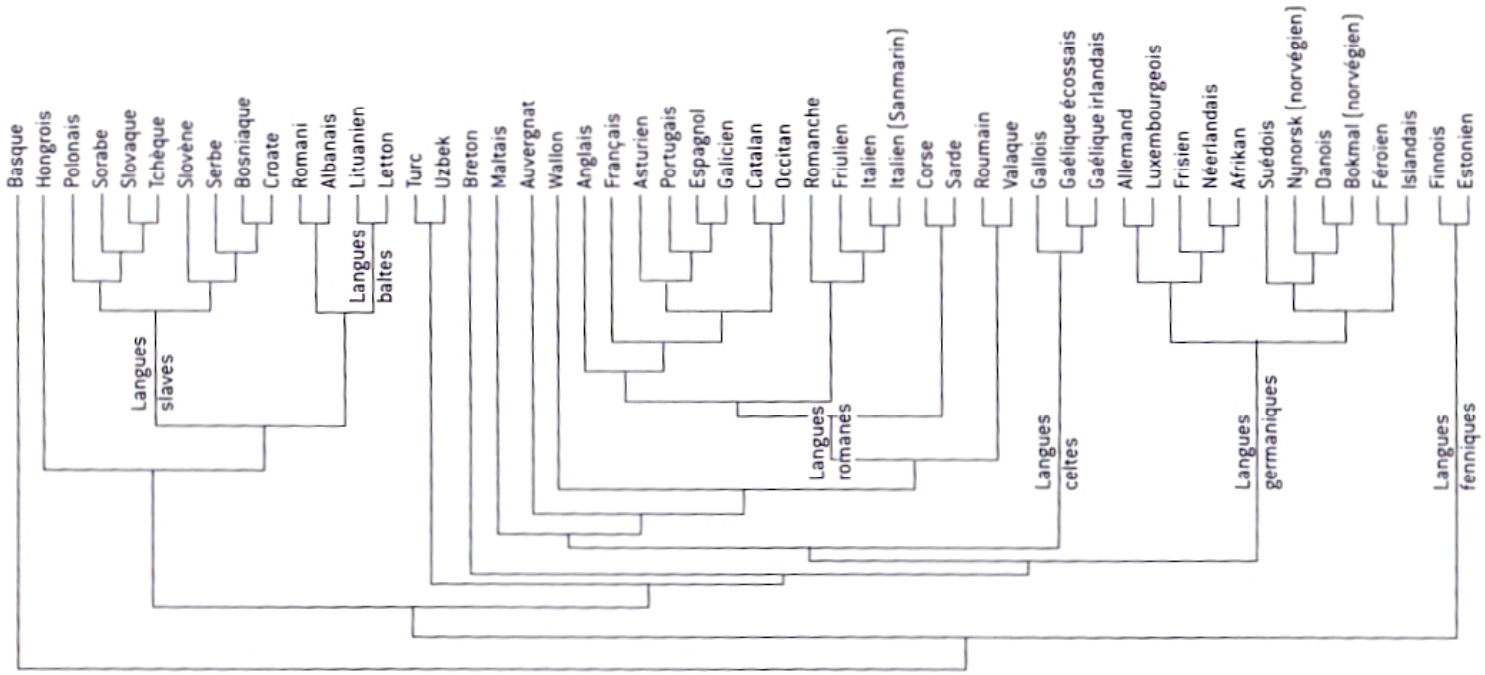
- Détection du **spam** (Gilles Richard).

- etc.


DebusBerg4
DebusBerg1
n_7
DebusBerg2
n_2
n_4
DebusBerg3
ChopPrel1
ChopPrel22
n_3
n_6
ChopPrel24
n_1
n_9
ChopPrel15
n_8
BachWTK2F2
n_5
n_0
BachWTK2P1
BachWTK2F1
BachWTK2P2



Arbre reconstituant l'évolution de 24 espèces de mammifères obtenus par l'algorithme de compression appliqué aux séquences de l'ADN mitochondriale. Cet arbre concorde bien avec les résultats classiques des paléontologues.



Classification des langues. Application aux langues indo-européennes à partir des traductions dans chacune des langues de la *Déclaration universelle des droits de l'Homme*.

Les résultats de la théorie donnent un sens précis aux égalités suivantes :

complexité de Kolmogorov = complexité aléatoire

incompressibilité = imprévisibilité = absence de structure

Aujourd'hui les concepts de la **théorie de la complexité de Kolmogorov** sont utilisés par les **physiciens** :

- définition de l'entropie d'un micro-état ;
- résolution du paradoxe du **démon de Maxwell** (R. Landauer, Ch. Bennett, W. Zurek)
- thermodynamique du calcul, calculateur réversible, etc.

Utilisation en **épistémologie**, **psychologie**, en **finance**, etc.

En **mathématiques**, on en tire aussi de nouvelles techniques de démonstration.

Important pour le fondement de la **théorie des probabilités** :

définition de la notion de suite aléatoire.

Revenons au problème de la définition de la complexité organisée

Affirmation :

Pour définir la complexité organisée, la complexité de Kolmogorov ne convient pas.

Argument

	cristal	π (10^9 digits)	ordinateur	nuage
K	*faible*	*moyen-*	*moyen +*	*grand*

Les objets les plus ***organisés*** sont en colonne 3.

Ilya Prigogine.

Une idée ?

L'exemple de π suggère de tenir compte de la **«quantité de calculs»** contenue dans un objet :

L'organisation —*la complexité organisée*— n'est-ce pas la marque que contient un objet attestant qu'il est le *résultat d'un long processus d'élaboration, ou de calcul* ?

Troisième tentative pour exprimer le "*contenu en calcul*" d'un objet fini : ***la profondeur logique de Bennett.***

La profondeur logique de Bennett de la suite ***s*** est définie par :

P(***s***) = *temps de calcul du programme minimal de* ***s***

Idée est mentionnée dans un article de G. Chaitin de 1977 qui l'attribue à Bennett.

Articles de Bennett en 1986, 1987, 1988, 1990, ...

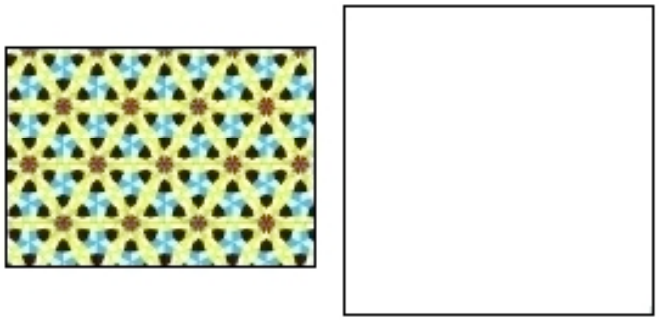
La définition de Bennett est-elle satisfaisante ?

pour une suite ***s*** de longueur n :

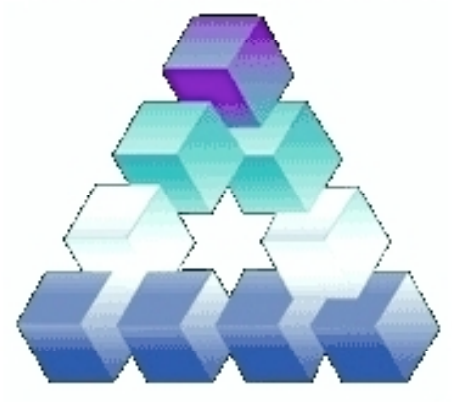
K varie de 0 à n (environ) alors que P varie de n à n^2, n^3, 2^n, ...

	cristal	π (10^9 digits)	ordinateur	nuage
K	faible	moyen-	moyen +	grand
P	faible	moyen	grande	faible

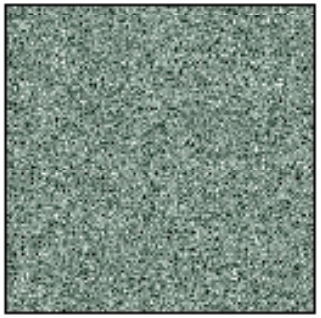
L'incompressibilité entraîne la rapidité du calcul à partir du programme minimal.



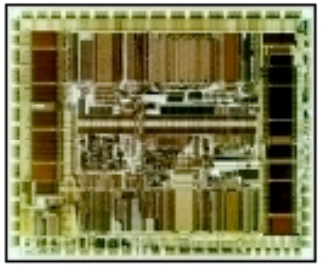
Simplicité



Complexité aléatoire



Complexité organisée



Considérations en faveur de l'identification :

profondeur de Bennett = complexité organisée

(a) Théorème d'invariance.

(b) Origine probable d'un objet.

Imaginons que l'on découvre le premier milliard de digits de π écrits sur le tronc d'un arbre à l'aide de barres plus ou moins longues codant 0 et 1.

On va supposer que ces digits ont été écrits à partir de «quelque chose qui a à voir avec π».

On va sans doute faire l'hypothèse d'un canular : quelqu'un à l'aide d'un programme, forcément assez court a calculé ces décimales et les a inscrites sur le tronc de l'arbre.

Si vraiment l'hypothèse du canular peut être écartée, on supposera qu'un mécanisme physique bien précis lié à des lois physiques (assimilables à un programme court) a réalisé le calcul de ces décimales.

L'origine vraisemblable d'un objet profond,
ne peut être attribuée
qu'à un de ses *programmes minimaux* ou *presque minimaux*.

(c) La loi de croissance lente

Cette hypothétique loi énonce que :

la complexité organisée augmente lentement en fonction du temps

Toute bonne notion formalisant la notion de **complexité organisée** doit satisfaire cette loi.

Est-ce le cas pour la profondeur logique de Bennett ?

Théorème (Bennett 1988)

La profondeur des objets présents dans un univers déterministe est susceptible d'augmenter au fur et à mesure qu'il y a du calcul qui s'effectue,

mais ne peut augmenter que lentement avec le temps.

Dit autrement, celui qui accepte l'identification :

complexité organisée = profondeur logique de Bennett

dispose d'une "explication" à la lenteur de l'apparition de la complexité organisée dans tout univers déterministe peu profond à l'origine.

Calcul de la profondeur logique de Bennett

(Hector Zenil, Jean-Paul Delahaye, Jean-François Colonna)

Si s^* est approché par le compressé de s

alors le temps de décompression de s^*

est une valeur approchée de $P(s)$

Exemple : 7 images classées par K croissant et P croissant.

Conclusion

- Il faut soigneusement distinguer la complexité aléatoire et la complexité organisée.

- L'identification **profondeur = complexité organisée** est une proposition intéressante. Elle constitue un pas en avant important dans la compréhension de la complexité.

- Les applications se développent grâce aux algorithmes de compression de données.


Vème Marathon des sciences