

> facilement ce supplément dans la somme, car:
 $\text{Card}(A \text{ union } B) = \text{Card}(A) + \text{Card}(B) - \text{Card}(A \text{ intersection } B)$.

Dans le cas d'ensembles finis en nombre fini, il n'y a égalité que lorsqu'il n'y a pas d'éléments communs à deux ensembles, ce qui est rare. Ainsi, notre première tentative de formalisation donne et démontre une maxime opposée à celle attribuée à Aristote!

ESSAI 2: LES ALGORITHMES

Prenons pour objets des problèmes algorithmiques concernant les nombres entiers n . Quelques exemples:

- A_1 : Factoriser n ;
- A_2 : Trouver la somme des diviseurs de n ;
- A_3 : Déterminer si n est un nombre premier;
- A_4 : Déterminer si n est un carré parfait;
- etc.

Prenons en guise de *tout* le problème de résoudre l'ensemble des problèmes élémentaires simultanément (pour un n fixé, ou pour un nombre fini de valeurs de n). Comme mesure de complexité, prenons, cela va de soi pour qui s'intéresse aux algorithmes, le nombre d'opérations ou la taille de la mémoire nécessaires pour effectuer la résolution.

Il a par exemple été démontré en 2002 que savoir si un entier n est premier est polynomial en fonction de la taille de n , c'est-à-dire que cela exige une quantité de calculs élémentaires qui croît comme un polynôme dont la variable est n . On est certain que le polynôme en question est de degré inférieur ou égal à 12 et on conjecture qu'il est de degré inférieur ou égal à 6.

Avec ce type de formalisations naturelles pour qui s'occupe d'algorithmes, la maxime attribuée à Aristote est à nouveau inexacte. En effet, la complexité de la résolution du *tout* sera au plus la somme des complexités des *parties*, et sera souvent inférieure. Certains problèmes bénéficieront des calculs faits pour d'autres, ce qui permettra des économies de ressource (en temps et en mémoire) pour qui cherche à traiter les problèmes simultanément.

Examinons un exemple: factoriser à la fois le nombre 375 (problème A) et le nombre 375000 (problème B). On pourra d'abord factoriser 375, ce qui donne $3 \times 5 \times 5 \times 5$, d'où $375000 = 3 \times 5 \times 5 \times 5 \times 1000$. Il restera alors à factoriser 1000, ce qui donne $2 \times 2 \times 2 \times 5 \times 5 \times 5$, d'où $375000 = 3 \times 5 \times 5 \times 5 \times 2 \times 2 \times 2 \times 5 \times 5 \times 5$. La complexité (en nombre d'opérations élémentaires) de la résolution de A et B est plus faible que $\text{Comp}(A) + \text{Comp}(B)$, car on a profité du calcul fait pour A quand on s'est intéressé à B.

ESSAI 3: LA COMPLEXITÉ DE KOLMOGOROV

Lorsqu'on considère des objets numériques finis, une autre idée naturelle est de mesurer leur valeur par la complexité de Kolmogorov.

Par définition, la complexité de Kolmogorov d'un objet numérique A, par exemple une image, que nous noterons $\text{Kolmo}(A)$, est la taille du plus petit programme qui produit A. Produire une image toute noire demande un programme très court, alors que produire l'image d'un visage nécessite un programme assez long. Même si les images ont la même taille en nombre de pixels, la première est moins complexe que la seconde, ce qu'indique correctement la complexité de Kolmogorov.

Cette mesure de complexité est unanimement considérée comme la bonne mesure du «contenu en information» d'un objet numérique. Généralisant l'entropie de Claude Shannon, elle est utilisée en informatique, en physique, en philosophie des sciences, en biologie, en psychologie. En pratique, calculer $\text{Kolmo}(A)$ revient à rechercher son fichier compressé le plus court. Même si l'on ne réussit qu'imparfaitement à trouver ce fichier compressé optimal, disposer de valeurs approchées de $\text{Kolmo}(A)$ grâce aux algorithmes de compression usuels est utile et permet par exemple de classer des données.

Pas de chance, et c'est plus grave ici car il s'agit vraiment d'une mesure de contenu en information, la complexité de Kolmogorov d'un ensemble d'objets numériques finis est inférieure ou égale à la somme des complexités de Kolmogorov des objets pris un à un. C'est un théorème de la théorie:

$$\text{Kolmo}(\text{Union}(A_i)) \leq \text{Kolmo}(A_1) + \dots + \text{Kolmo}(A_k).$$

L'idée de la démonstration est simple:

- les programmes les plus courts qui engendrent $A_1, A_2, \dots, A_k$, peuvent être mis bout à bout;
- ils constituent alors un programme P qui engendre le *tout* (c'est-à-dire $\text{Union}(A_i)$);
- ce programme P n'est peut-être pas le plus court qui donne le *tout*, mais le programme le plus court qui donne le *tout* sera plus court (puisqu'il y a déjà ce programme-là) et donc la complexité du *tout* sera inférieure à la somme des complexités des *parties*. Encore une fois, la théorie démontre une inégalité inverse de celle de la maxime.

Fort de ces exemples, il semblait que jamais on ne pourrait trouver de situations mathématiques où la complexité du *tout* est plus grande que la somme des complexités des objets pris individuellement. Même en cherchant le plus honnêtement possible, quelle que soit la façon (naturelle) de définir et de mesurer la complexité, impossible de trouver un *tout* meilleur que la somme des *parties*!

Précisons que dans cette recherche d'une mesure de complexité satisfaisant la maxime, on a exclu les méthodes factices où l'on place dans le *tout* autre chose que l'ensemble des *parties*. Par exemple, on ne peut pas considérer comme une illustration acceptable de la maxime l'affirmation qu'il y a dans un mot

plus que ce qu'il y a dans l'ensemble des lettres qui le composent.

Il est vrai que dans le mot COMPLEXE, il y a plus que dans la donnée de l'ensemble de ses lettres C, E, L, M, O, P, X, mais c'est bien évidemment parce qu'on ordonne les lettres et qu'on en répète, et que cet ordre et ces répétitions ajoutés aux parties constituent un «plus» qu'on trouve dans le *tout* et qui est absent de la somme des *parties*.

De telles illustrations de la maxime étudiée sont triviales et sans portée, puisqu'elles sont fondées sur un ajout caché quand on constitue le *tout*. Les cas soi-disant spectaculaires de synergie, d'effets cocktail et autres doivent le plus souvent s'interpréter de la même façon. Oui, un vélo est plus que ses parties étalées au sol, mais c'est parce qu'on organise ses parties en les assemblant de la bonne façon et que cette bonne organisation ajoute aux parties.

LA COMPLEXITÉ COMME CONTENU EN CALCUL

La théorie algorithmique de l'information qui détaille ce qu'on peut dire et démontrer sur la complexité de Kolmogorov a introduit une notion qui va nous sauver: la «profondeur logique de Bennett» qui est le temps de calcul du plus court programme qui produit l'objet numérique fini auquel on s'intéresse. C'est une mesure de complexité structurelle, ou encore, une mesure de «contenu en organisation et en calcul», alors que la complexité de Kolmogorov n'est qu'une mesure de «contenu incompressible d'information». Plus un objet est finement structuré, comme l'est par exemple un organisme vivant, plus sa profondeur logique est grande.

Ces deux mesures de complexité, complexité de Kolmogorov et profondeur logique, diffèrent radicalement pour les objets aléatoires dont l'exemple typique est une suite finie de 0 et de 1 obtenue par des tirages successifs à pile ou face. Pour un tel objet aléatoire, la complexité de Kolmogorov est maximale: on ne peut pas le décrire de manière sensiblement plus brève qu'en en donnant les éléments un à un, ce qui est la pire situation pour un objet numérique de taille n , puisque l'objet à produire doit alors être écrit explicitement dans le programme, lequel est nécessairement aussi long que l'objet décrit: une suite aléatoire des bits est incompressible.

À l'inverse, la profondeur logique d'une suite aléatoire de longueur n est minimale pour les suites de longueur n . Une suite aléatoire n'est pas structurée, son contenu en organisation est presque nul. Sa profondeur logique de Bennett est réduite au minimum, puisque exécuter le programme le plus court

2

LA CRYPTOGRAPHIE VISUELLE À LA RESCOURSE

La cryptographie visuelle fournit un exemple incontestable de situation où le tout est plus que la somme des parties, plus précisément où le contenu en structures du tout est bien supérieur à la somme des contenus en structures de chacune de ses deux parties prises isolément.

On considère une image A composée à parts égales de points noirs et blancs disposés aléatoirement. Cette image n'a aucun contenu en structure, ou, dit autrement, sa profondeur logique de Bennett est très faible. On considère une image C ayant, elle, un contenu en structure important: il s'agit ici d'un pavage du plan hyperbolique. On calcule l'image B = A xor C.

Un point est blanc dans B si les points correspondants de A et C sont tous les deux blancs ou tous les deux noirs; sinon, il est noir. L'image B est du même type que l'image A: elle n'a aucun contenu en structure et donc une profondeur très faible. Celui qui dispose de A et de B peut pourtant reconstituer C en calculant $A \text{ xor } B = A \text{ xor } (A \text{ xor } C) = (A \text{ xor } A) \text{ xor } C = C$. Il en résulte que $\text{Profondeur}(A \text{ et } B) \approx \text{Profondeur}(C)$, qui est supérieure à $\text{Profondeur}(A) + \text{Profondeur}(B)$.

Notons que si l'on imprime A et B sur des feuilles transparentes et qu'on les superpose, on voit apparaître l'image D, une forme bruitée de l'image C. C'est le principe de base de la cryptographie visuelle.

