> facilement ce supplément dans la somme, car: *Card*(A union B) = *Card*(A) + *Card*(B) – *Card*(A 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 3×5×5×5, d'où 375000= 3×5×5×5×1000. Il restera alors à factoriser 1000, ce qui donne 2×2×2×5×5×5, d'où 375000=3×5×5×5×2×2×2×5×5×5. La complexité (en nombre d'opérations élémentaires) de la résolution de A et B est plus faible que *Comp*(A) + *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 *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 *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 *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:
$Kolmo(\text{Union}(A_i)) \leq Kolmo(A_1) + \ldots + Kolmo(A_k)$.

L'idée de la démonstration est simple:
- les programmes les plus courts qui engendrent $A_1, A_2, \ldots, A_k$, peuvent être mis bout à bout;
- ils constituent alors un programme P qui engendre le *tout* (c'est-à-dire 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