*18997 DÉCELLAS Victor*

# Algorithmes de compression de données sans perte

La compression de données est un domaine d'études riche avec énormément d'applications pratiques mais aussi une grande richesse théorique. C'est également un thème à l'intersection entre les mathématiques, via la théorie de l'information ou les ondelettes, et l'informatique : deux sujets qui me passionnent.

Ce sujet est lié au thème de l'année par le fait que la compression et la décompression de données sont des processus qui consistent en une transformation de celles-ci pour les rendre moins lourdes, mais aussi par des questionnements sur comment inverser une altération indésirable issue d'une compression avec pertes.

## Positionnement thématique (ÉTAPE 1) :

- *INFORMATIQUE (Informatique Théorique)*
- *INFORMATIQUE (Informatique pratique)*
- *MATHEMATIQUES (Mathématiques Appliquées)*

## Mots-clés (ÉTAPE 1) :

| **Mots-clés (en français)** | **Mots-clés (en anglais)** |
|---|---|
| *compression* | *compression* |
| *théorie de l'information* | *information theory* |
| *entropie* | *enthropy* |
| *codage arithmétique* | *arithmetic coding* |
| *LZ77* | *LZ77* |

## Bibliographie commentée

La transmission de nombreux fichiers potentiellement très lourd à travers des canaux de transmission avec un débit limité ainsi que la question du stockage de l'information en question, que ce soit dans d'immenses centres de serveurs ou dans de modestes ordinateurs personnels, font qu'il est essentiel de pouvoir compresser des données le plus efficacement possible, que ce soit en terme de mémoire gagnée qu'en temps. De ce fait, nombre de technologies ont étés mises en place pour répondre à ce besoin.

Ces algorithmes sont classés en deux grandes catégories : avec perte et sans perte. Ce TIPE se concentrera sur compression la deuxième classe, bien que les deux domaines connaissent une

recherche florissante, par exemple avec des LLMs (Large Language Model) qui ont montrés avoir des capacités de compression meilleures que les algorithmes "lossless" standards, tels que PNG pour les images ou FLAC pour l'audio. [1]

Un simple argument de cardinalité prouve qu'il est impossible d'avoir un algorithme de compression "parfait" qui compresse strictement et sans perte toute chaîne de caractères. En fait, on sait même que si une chaîne de caractères est strictement compressée par un algorithme sans perte, il existe au moins une autre chaîne dont la version compressée est strictement plus longue qui sa version initiale. [2] En fait, on a même un résultat plus fort qui implique que peu importe l'algorithme choisi, il est impossible de comprimer de manière optimale toutes les chaînes de caractères, c'est-à-dire qu'il existe toujours des chaînes pour lesquelles un autre algorithme comprime plus efficacement la chaîne en question que celui que l'on utilise. [4] [5] Bien qu'en pratique, les fichiers traités ne sont absolument pas des chaînes de caractères quelconques puisqu'ils contiennent beaucoup de motif récurrents, comme par exemple la lettre "e" dans un texte en français ou certaines instructions dans le fichiers exécutables.

Ces techniques de codage sont étudiées via la théorie de l'information, une théorie mathématique qui permet l'étude rigoureuse de la compression de données, mais dont les champs d'applications sont loin de se limiter à ce seul domaine. Le premier théorème de Shanon, notamment, établit une borne inférieure à la longueur moyenne d'un message comprimé : l'entropie. Ce même théorème donne également une majoration de la plus petite longueur moyenne atteignable, elle aussi définie en fonction de l'entropie. [2]

Ce théorème du codage de source est à l'origine d'un classe d'algorithmes de compression nommée "codage entropique" dont l'objectif est d'approcher le plus possible de l'entropie de la source. Les deux principaux algorithmes de ce type sont le codage de Huffman ainsi que le codage arithmétique. On sait que l'algorithme de Huffman est effectivement optimal si l'on se restreint à de l'encodage symbole par symbole, mais aussi que sans cette contrainte, il existe de meilleurs algorithmes, dans le sens où ils parviennent, à avoir des messages compressés dont la longueur moyenne est plus proche de l'entropie. On peut citer le codage arithmétique, qui non seulement atteint l'entropie dans les cas où Huffman y parvient également, mais peut aussi être arbitrairement proche de celle-ci. En encodant par blocs de symboles arbitrairement grands, on peut effectivement arbitrairement s'approche de cette valeur en utilisant le codage de Huffman, mais cette technique est particulièrement lente à cause du nombre considérable de possibilités à prendre en compte.[3] Cependant, l'exécution de tout algorithme entropique nécessite une estimation des fréquences des caractères du string.

Une autre approche fréquemment utilisée est le codage par dictionnaire, qui consiste à chercher dans un string des mots enregistrés dans une structure de donnée nommée "dictionnaire", et à substituer chaque occurrence d'un mot donné par une référence vers ladite structure. L'algorithme de Huffman bloc par bloc est bien un codage par dictionnaire, car chaque mot est remplacé par son chemin correspondant dans un arbre. Cependant, le dictionnaire est la plupart du temps dynamique, comme dans LZ77 et LZ78, c'est-à-dire que l'ensemble des mots enregistrés évolue au fil de la compression. [3]

## Problématique retenue

Avec l'ensemble des techniques susmentionnés, mais aussi avec d'autres qui n'ont pas pus être nommées par manque d'espace, on se posera la question de comment créer des algorithmes de compression sans perte à la fois efficaces et rapides, potentiellement spécialisés dans un type de données précis.

## Objectifs du TIPE du candidat

Je me suis fixé comme objectif d'implémenter une bibliothèque de compression de données sans perte, à la fois polyvalente et efficace. En plus de l'implémentation, il s'agit aussi d'étudier à la fois dans la théorie et la pratique les résultats donnés par ces algorithmes, que ce soit individuellement ou lorsqu'on les combine.

## Références bibliographiques (ÉTAPE 1)

**[1]** EDWARDS BENJ : AI language models can exceed PNG and FLAC in lossless compression, says study *: https://arstechnica.com/information-technology/2023/09/ai-language-models-can-exceed-png-and-flac-in-lossless-compression-says-study/*

**[2]** THOMAS M COVER & JOY A THOMAS : Elements of Information Theory *: chapitre 5 : "Data Compression", chapitre 6 : "Gambling and Data Compression"*

**[3]** GUY E. BLELLOCH : Introduction to Data Compression

**[4]** PETER BRO MILTERSEN : Course notes for Data Compression - 2 Kolmogorov complexity

**[5]** PIERRE CAGNE : Complexité de Kolmogorov