# TP1 INF8770

## COMPARAISON ET CARACTÉRISATION DE MÉTHODES DE CODAGE

VALENTIN BOUIS (1733927), BENJAMIN HEINEN(1780391)

fpa

# Question 1

**Hypothèse 1 :**

On suppose que moins il y aura de teintes de couleur différentes (exemple : image en noir et blanc), plus l'algorithme de codage par paires d'octets sera efficace. En effet il sera capable de remplacer les répétitions de couleurs par un autre symbole. Dans le cas d'une image en noir et blanc, les répétitions cote a cote de noir et de blanc seraient alors remplacées par un autre symbole, ce qui diminuerait grandement la taille de l'image. Au contraire pour l'algorithme de codage prédictif, l'objectif est d'obtenir le plus de zéro possible dans l'erreur. Ainsi pour une image en noir et blanc on essayerait de transformer les 1 (blanc) en 0 (noir), mais les valeurs initialement à 0 risquent également d'être modifié. On pense donc que le codage par paires d'octets compressera mieux une image avec peu de teintes de couleurs que le codage prédictif.

**Hypothèse 2 :**

Plus l'image aura de couleurs distinctes qui ne se répètent pas selon un motif, plus le codage prédictif compressera mieux que le codage par paires d'octets. En effet le codage par paires d'octets ne sera pas capable de simplifier des duos de pixels s'il ne trouve pas de motif répété.

Voici un exemple :

*Code : 0 1 2 3 4 5*

*Version paires d'octets : pas de simplification possible (pas de pair identique)*

*Version prédictif : 0 0 0 0 0 0 dans le cas ou x(n) = x(n-1) - 1*

On voit alors que dans le cas extrême ou aucun motif n'apparait l'algorithme de codage prédictif est bien meilleur que celui par paires d'octets.

**Hypothèse 3 :**

On pense que le temps d'exécution de l'algorithme de codage par paires d'octets sera plus lent que l'algorithme de codage prédictif lors de la compression. En effet le codage par paires d'octets nécessite un algorithme récursif ainsi que d'établir un dictionnaire

des combinaisons de 2 pixels possibles ainsi que leur nombre d'occurrences. En revanche le codage prédictif ne nécessite seulement que le calcul de l'erreur.

**Hypothèse 4 :**

La dernière hypothèse que nous formulons correspond à la taille occupée dans la mémoire lors de la compression. En effet, on pense que celle-ci sera plus faible pour l'algorithme de codage prédictif, en admettant qu'on combine celui-ci à l'algorithme de Huffman. Ainsi, le dictionnaire nécessaire à la compression par paires d'octets sera plus grand puisqu'il y a plus de combinaisons de symboles (chaque symbole apparait dans 2 combinaisons) que de symboles.

## Question 2

**Test hypothèse 1 :**

Pour tester l'efficacité de compression, nous allons tout simplement comparer la valeur du quotient de compression pour chaque algorithme. Celui-ci est égal à :

*Taille de l'image initiale/Taille de l'image compressée*

On regardera alors ce quotient pour chaque exemplaire d'image afin de voir si l'algorithme de codage par paires d'octets est bien plus efficace que celui prédictif selon les situations.

De plus, nous tracerons un graphique représentant les performances de compression pour des images allant d'une taille de 2x2 pixel a une taille de 1024x1024 pixels.

**Test hypothèse 2 :**

De la même manière que lors de l'hypothèse 1, nous allons comparer la valeur du quotient de la compression pour les différents algorithmes de codage. Ce quotient sera toujours égal à :

*Taille de l'image initiale/Taille de l'image compressée*

On pourra alors comparer selon les différents algorithmes et les différents type d'image. On utilisera également un graphique permettant d'analyser les performances de compression des différents algorithmes.

**Test hypothèse 3 :**

Pour tester le temps d'exécution de chacun des algorithmes, nous allons tout simplement comparer le temps pris par chacun pour compresser une image. Pour cela nous utiliserons le module Time de python. Nous donnerons la même image à traiter aux deux algorithmes. Nous lancerons le chronomètre au commencement de l'algorithme (après l'import de l'image) et l'arrêterons quand celui-ci aura fini de compresser l'image.

De plus, nous utiliserons un graphique qui permet de comparer la vitesse d'exécution des algorithmes de codage. Celui-ci devrait permettre de mettre en lumière certain

**Test hypothèse 4 :**

Afin de comparer la taille prise en mémoire par chaque algorithme, nous allons utiliser la librairie psutil de python. Celle-ci nous permet d'obtenir de nombreuses informations liées à la consommation en mémoire du programme grâce à la méthode memory_info(). Nous insérerons les données pertinentes dans un tableau puis nous les comparerons.

## Question 3

Tout d'abord, nous avons réalisé l'algorithme par paires d'octets en nous inspirant de celui que nous avions à notre disposition sur le Github du cours. Néanmoins, nous l'avons modifié afin de pouvoir traiter une image. Afin de simplifier les calculs, nous avons fait le choix de choisir une image uniquement composée de teintes de gris, ce qui permet de réduire la taille de l'image par trois. En effet, celle-ci n'est composé que d'un octet par pixel, au lieu des trois que l'on a classiquement avec le modèle RGB.

Par la suite, nous avons fait le choix de choisir une image de petite taille (64x64 pixel) plutôt qu'une grande image. Ce choix a été fait de façon expérimentale, lorsque l'on s'est rendu compte que l'algorithme par paires d'octets prenait beaucoup de temps.

Ensuite, nous avons commencé à développer l'algorithme prédictif. Cela fut assez rapide en reprenant l'exemple pris sur le Github du cours. Néanmoins, il ne serait pas raisonnable de comparer le codage prédictif avec le codage par paires d'octet. Ainsi, afin de les rendre comparable, nous avons appliqué le codage d'Huffman au codage prédictif. Pour ce faire, nous avons également fait le choix de récupérer l'implémentation du codage Huffman présente sur le Github du cours.

De plus, afin de diversifier les tests possibles, nous avons codé des images directement en python. Ainsi, une image composée d'une alternance de blanc et de noir sera un tableau de deux dimensions composées de zéro et d'un, pour lequel on change de nombre à chaque index du tableau. De même, une image composée de blanc sur la première moitié puis de noir sur la seconde moitié sera une image avec une première moitié avec que des zéros, puis une seconde moitié avec que des uns.

Tableau 1 : Tigre, taille 33024

| | Prédictif + Huffman | Paires Octets |
|---|---|---|
| Taille compressée | 17669 | 8724 |
| Temps exécution (s) | 0,4 | 1880,54 |
| Mémoire vive utilisée (Mo) | 72,97 | 61,39 |

Tableau 2 : Noir et blanc (alterné), taille 33020

| | Prédictif + Huffman | Paires Octets |
|---|---|---|
| Taille compressée | 33020 | 8 |
| Temps exécution (s) | 1,34485 | 0,2498 |
| Mémoire vive utilisée (Mo) | 76,05 | 70,95 |

Tableau 3 : Noir et blanc (alterné), taille 90

| | Prédictif + Huffman | Paires Octets |
|---|---|---|
| Taille compressée | 90 | 5 |
| Temps exécution (s) | 0,22321 | 0,00178 |
| Mémoire vive utilisée (Mo) | 72,49 | 70,77 |

Tableau 4 : Noir et blanc (continu), taille 33020

| | Prédictif + Huffman | Paires Octets |
|---|---|---|
| Taille compressée | 33020 | 16 |
| Temps exécution (s) | 1,17671 | 0,60958 |
| Mémoire vive utilisée (Mo) | 75,00 | 71,33 |

Tableau 5 : Noir et blanc (continu), taille 90

| | Prédictif + Huffman | Paires Octets |
|---|---|---|
| Taille compressée | 114 | 10 |
| Temps exécution (s) | 0,23677 | 0,005 |
| Mémoire vive utilisée (Mo) | 72,48 | 71,04 |

Tableau 6 : Dégradé, taille 49152

| | Prédictif + Huffman | Paires Octets |
|---|---|---|
| Taille compressée | 18807 | 8352 |
| Temps exécution (s) | 0,38237 | 2490,66 |
| Mémoire vive utilisée (Mo) | 73,31 | 66,37 |

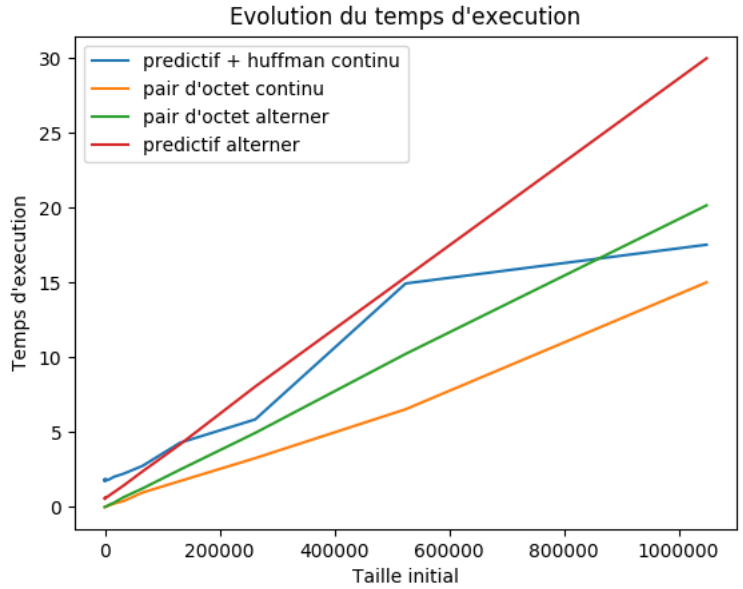


*Figure 1 : Évolution du temps d'exécution en fonction de la taille*

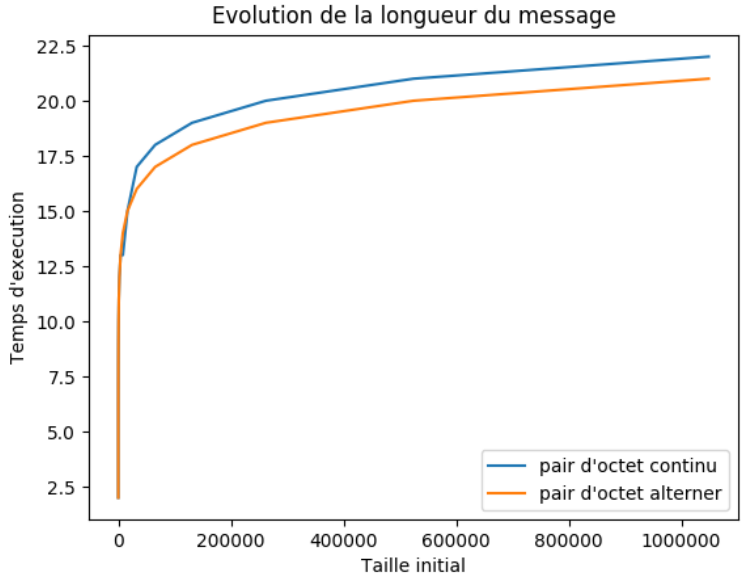


*Figure 2 : Évolution de la longueur du message en fonction de la taille*

## Question 4

Dans cette partie, nous analyserons les résultats obtenus selon trois catégories : la compression du message, le temps d'exécution de l'algorithme de codage et la mémoire utilisé. Ensuite, nous comparerons les résultats obtenus à nos hypothèses de la première question.

**Compression du message**

Tout d'abord, dans le tableau 1, nous pouvons constater que pour une image composée de motif (un lion dans notre image), l'algorithme par paires d'octet compresse deux fois mieux le message que l'algorithme prédictif mélangé au codage Huffman, ce qui s'avère très intéressant.

Dans les tableaux 2 et 3, qui affichent les résultats pour une image avec une alternance de noir et de blanc, on se rend compte de quelque chose de particulièrement intrigant. Si le codage par paires d'octets réduit considérablement la taille du message, passant de 33020 caractères à 8, le codage prédictif mélangé au codage Huffman ne réduit absolument pas la taille du message. En réalité, ce résultat semble très naturel. En effet, compte tenu du fait que notre message ne contient que des 0 et des 1, même si le codage prédictif permettait, avec une formule optimale, d'avoir que des 0, chaque symbole du message tiendrait sur un bit, comme pour le message original. Ainsi, le codage Huffman ne serait pas capable d'améliorer la compression. En effet, le codage Huffman permet d'optimiser le nombre nécessaire de bits à chaque symbole. Comme chaque symbole tient déjà sur un bit, cela ne pourrait pas améliorer la compression du message.

Par la suite, on constate dans les tableaux 4 et 5 que le codage prédictif réagit exactement de la même façon que dans les deux tableaux précédents. En revanche, dans le tableau 5, qui ne comporte que 90 éléments, on peut tout de même remarquer que la compression échoue totalement, augmentant ainsi la taille du message. Cela pourrait être améliorer en choisissant une formule plus approprier pour déterminer l'erreur. En effet, la formule actuel produit trois symboles différents dans le message, ce qui s'encode en deux bits. Cela augmente donc légèrement la taille du message.

Enfin, dans le tableau 6, on se rend compte que le codage prédictif, toujours couplé au codage de Huffman, diminue la taille du message d'un facteur proche de 2,60, tandis que le message par paires d'octets diminue la taille du message par un facteur d'environ 5,90.

Finalement, il apparait que dans la figure 2, on peut voir que l'algorithme par paires d'octet évolue toujours avec la même courbe, ne prenant pas en compte si l'image est composé d'une alternance de pixel de couleur opposé ou d'une moitié de pixel d'une couleur tandis que l'autre serait d'une autre couleur.

**Temps d'exécution**

Tout d'abord, dans le tableau 1, qui permet de coder une image composée d'un motif représentant un tigre, on se rend compte que le codage par paires d'octet prend un temps considérable par rapport au codage prédictif couplé au codage Huffman. En effet, le codage par paires d'octet prend environ 4700 fois plus de temps.

En revanche, dans le cas d'une image composé d'une alternance de noir et de blanc, on peut constater, comme présenté dans les tableaux 2 et 3, que le codage par paires d'octet est le plus rapide, et ce qu'importe la quantité d'information. De plus, le facteur diminue proportionnellement à la longueur du message. Par conséquent, il se peut qu'à partir d'une certaine taille, l'utilisation d'un codage prédictif couplé à un codage Huffman soit plus rapide. Avec les données que nous avons récoltées, on se rend compte que lorsque l'on a 33020 données, le codage par paires d'octet est environ 5 fois plus rapide. De même, quand le message a une longueur de 90, il est 125 fois plus rapide.

Ce constat se reproduit dans les tableaux 4 et 5, qui permettent d'analyser les données pour des images composées d'une moitié blanche et d'une autre moitié noire. Ainsi, on se rend compte que pour une image ayant une longueur d'environ 30020 éléments, le codage pair d'octet est environ deux fois plus rapide. De plus, s'il y a moins d'élément, la différence de temps est nettement plus grande. En effet, lorsqu'il y a 90 pixels dans l'image, le codage par pair d'octet est environ 47 fois plus rapide.

Finalement, il est intéressant de comparer les courbes représentant les temps d'exécutions des différents types de codage présent dans la figure 1. On se rend ainsi compte que, que l'image soit constitué d'une alternance de noir et de blanc ou qu'elle contienne une moitié blanche et une autre moitié noire, le codage prédictif couplé au codage Huffman est plus lent que le codage par pair d'octet. De plus, le traitement d'une image ayant des couleurs par alternance est toujours plus lent qu'une image étant divisé en deux couleurs. Néanmoins, on peut supposer que sur des très grandes quantités de données, le codage prédictif puisse devenir plus rapide que le codage par paires d'octet, compte tenu de l'inclinaison des courbes dans le cas d'une image composé d'une moitié de blanc et d'une autre moitié de noire. Cela ne semble en revanche ne pas être vrai dans le cas où il y a une alternance de couleur.

On peut expliquer le temps considérable que prend le codage par paires d'octets pour une image complexe (autre que noir et blanc) grâce au dictionnaire qu'il maintient à jour à chaque paire de symbole parcouru de la chaine originale. En effet pour un grand nombre de symboles (image en couleur / niveau de gris), le nombre de combinaisons de symboles grandit également, et ainsi le parcours du dictionnaire pour ajouter/modifier une paire de symbole devient plus difficile.

Concernant la rapidité du codage par paires d'octets dans le cas d'images à deux teintes (noir et blanc), c'est encore une fois la taille dictionnaire qui joue un rôle important. Les seuls motifs possibles étant 11 10 01 00 le parcours de ce dernier se fait très rapidement.

**Mémoire vive utilisée**

Pour la mémoire vive utilisée, il faut prendre en compte deux choses : La mémoire prise par l'image de départ et celle prise par les différents objets crées par l'algorithme. Pour une même image la différence dans nos tableaux montre alors quel algorithme consomme le plus de mémoire pendant son exécution.

La première chose qu'on remarque est que l'algorithme prédictif + Huffman utilise plus de mémoire que l'algorithme de codage par paires d'octets pour des images avec peu de symboles (entre 5% et 8% de plus). Cette différence s'explique par le fait que le dictionnaire utilisé dans le codage par paires d'octets est très petit puisqu'il existe que 4 combinaisons de symbole possible (11, 10, 01, 00). Au contraire, le codage prédictif va avoir besoin d'un tableau pour stocker l'erreur, un tableau pour stocker l'image prédite, et un autre pour appliquer l'algorithme d'Huffman (même si celui-ci sera très léger pour 2 symboles).

Deuxièmement, si on regarde les tableaux 3 et 5, on se rend compte que l'algorithme prédictif consomme la même quantité de mémoire peu importe si les pixels blancs et noirs sont alternés ou non. Ce n'est pas le cas pour le codage par paires d'octets qui consommera moins de mémoire pour des pixels noir et blanc alternés, ce qui fait du sens puisque le motif est tout le temps le même dans le cas des pixels alternés (01 et 10 dans le dictionnaire).

**Liens avec nos hypothèses**

Nos résultats corroborent notre **hypothèse 1** puisque le taux de compression pour les images en noir et blanc est bien meilleur pour l'algorithme de codage par paires d'octets. Par exemple pour l'image qui alterne un pixel blanc et un pixel noir, on trouve un taux de compression de 18 pour « paires d'octet » et de 1 (pas de compression) pour « prédictif ».

En revanche l'**hypothèse 2** semble erronée car nos images qui ne présentent pas de motifs récurrents ou un faible nombre de couleurs (image du tigre / image dégradée) ne sont pas mieux compressées par l'algorithme de codage prédictif.

Il semblerait que nous n'ayons pas fourni suffisamment d'information pour **l'hypothèse 3**. En effet selon la complexité de l'image (nombre de couleur et motifs), on obtient une vitesse d'exécution très différente : Pour une image ayant une grande diversité de couleurs, l'algorithme de codage par paires d'octet n'est pas intéressant. En effet, il prend plus de 30 minutes à s'exécuter pour une image ayant peu de pixel, et cela malgré le fait que l'on utilise uniquement des teintes de gris au lieu de la structure classique RGB, tandis que le codage prédictif est très efficace en se terminant en 1 seconde environ. Néanmoins, pour des image simple composé de peu de couleur, le codage par paires d'octet s'avère bien plus rapide que le codage prédictif.

Il semblerait que nous nous sommes fourvoyer en posant notre **hypothèse 4**. Nous pensions que la taille occupée en mémoire serait moindre pour l'algorithme prédictif, couplé au codage Huffman. Néanmoins, on se rend compte qu'elle est plus élevée pour toutes nos images. Nous allons essayer de déterminer la raison de ce résultat.

Tout d'abord, la combinaison de codage prédictif dispose du poids de l'arbre de Huffman ainsi que le poids du message. On peut également considérer le poids de la formule permettant de calculer l'erreur. Néanmoins, cela nous semble non significatif, car nous supposons que c'est très faible par rapport à la mémoire totale utilisée par l'algorithme.

En contrepartie, le codage par paires d'octet utilise le poids du dictionnaire, qui nous semble plus important que le poids de l'arbre de Huffman. En revanche, puisque le message est nettement mieux compressé dans chacune des situations présentées, la différence entre le poids du message dans les deux algorithmes explique sans aucun doute la différence de poids observé dans nos tableaux.