RECHERCHE

Compression et compréhension des séquences de nucléotides

Stéphane Grumbach, Fariza Tahi

I.N.R.I.A.
Rocquencourt, BP 105
78153 Le Chesnay
FRANCE
E-mail: {Stephane.Grumbach , Fariza.Tahi}@inria.fr

Travail réalisé dans le cadre du projet "Recherche dans les Séquences" du Groupement de Recherches et d'Etudes sur les Génomes, GREG.

RÉSUMÉ. *Les séquences d'ADN contiennent un fort pourcentage de redondances, dont les répétitions "palindromiques", que les algorithmes universels de compression ne détectent pas. Nous présentons l'algorithme "*Biocompress-2*" de compression des séquences de nucléotides, qui est à notre connaissance, l'algorithme le plus efficace sur notre corpus de séquences d'ADN. Les taux de compression ne sont pas satisfaisants, mais nous montrons que cette technique permet la découverte de régularités pertinentes, et présente un lien avec les techniques de construction d'arbres phylogéniques et de prédiction de structures secondaires d'ARN.*

ABSTRACT. *DNA sequences contain a high percentage of redundancies, such as "palindromic" repetitions, that universal compression algorithms fail to detect. We present the algorithm "*Biocompress-2*" to compress nucleotide sequences, which, to the best of our knowledge, is the most efficient algorithm on our corpus of DNA sequences. The rates are not satisfactory, but this technique leads to the discovery of relevant regularities, and presents a connection with the techniques of construction of phylogenetic trees and prediction of secondary structures of RNAs.*

MOTS-CLÉS : *compression de données, séquences d'ADN et d'ARN, régularités, palindromes, structure secondaire d'ARN, phylogénie.*

KEY WORDS : *data compression, DNA and RNA sequences, regularities, palindromes, RNA secondary structure, phylogeny.*

1. Introduction

Il existe différentes méthodes formelles pour analyser les séquences génétiques et détecter des régularités. Parmi elles, on peut en particulier citer l'analyse statistique et la reconnaissance de motifs ("pattern matching"). Nous nous proposons d'utiliser les techniques de compression de données pour étudier les séquences de nucléotides. La compression de données répond à une motivation pratique et théorique : pratique par la possibilité de réduire la taille des données, pour faciliter le stockage et l'échange, et théorique par la possibilité de détecter des régularités.

Le type de données génétiques sur lesquelles nous avons travaillé sont des séquences de nucléotides, ADN et ARN. Ces séquences peuvent être considérées comme des *textes* sur un alphabet à quatre lettres $\{A, C, G, T\}$ (notons que T est remplacé par U dans le cas de l'ARN). Ces textes peuvent être très longs, en particulier lorsqu'il s'agit de génomes complets. Le génome humain, par exemple, est constitué de trois milliards de paires de bases. De telles longueurs ne sont pas séquencées à l'heure actuelle. Dans nos exemples, nous avons considéré la plus longue séquence disponible (315357 caractères), le chromosome III de la levure (YSCCHRIII) [OLI 92].

Plusieurs algorithmes ont été proposés pour la compression de textes [LEL 87, STO 88]. Il existe deux approches principales, l'approche statistique et l'approche substitutionnelle. Dans la première, les symboles sont codés suivant la probabilité de leur apparition ; les mots de code courts sont utilisés pour le codage des symboles les plus fréquents, tandis que les mots de code longs sont utilisés pour les symboles les moins fréquents [HUF 52]. Dans la deuxième approche, des mots de tailles différentes sont codés par une référence (un pointeur) à l'une de leurs précédentes apparitions dans le texte [LEM 76].

Nous avons testé de nombreux algorithmes de compression de textes sur un échantillon de séquences d'ADN. Ceux basés sur l'approche substitutionnelle donnent tous des taux de compression négatifs (la taille du texte obtenu est supérieure à la taille du texte initial), tandis que ceux basés sur l'approche statistique donnent des taux de compression en général positifs mais très faibles. L'algorithme le plus efficace sur les séquences d'ADN est le codage arithmétique d'ordre 2, dont le taux de compression sur notre échantillon est en moyenne de 3.86%.

Nous avons développé deux algorithmes de compression (sans perte d'information) des séquences d'ADN et d'ARN : un algorithme purement substitutionnel *Biocompress* [GRU 93], et une extension avec des primitives statistiques, *Biocompress-2* [GRU 94]. Essentiellement substitutionnel, *Biocompress-2* est basé sur la détection et le codage des répétitions. Sa particularité par rapport aux autres algorithmes basés sur cette approche est la détection et le codage d'un type particulier de répétitions, propres aux séquences d'ADN et d'ARN, les *répétitions palindromiques*. Les palindromes jouent un rôle important dans la formation de la structure secondaire de l'ARN. En effet, la structure secondaire est le résultat de repliements de la séquence primaire dus à des appariements de palindromes.

Le (relatif) succès de *Biocompress-2* provient de : la détection et le codage des palindromes, la détection des facteurs longs et éloignés, l'utilisation de mots de code ne pouvant être plus longs que la séquence qu'ils codent, et enfin, l'utilisation du codage arithmétique d'ordre 2 lorsque des répétitions significatives n'ont pu être détectées. A notre connaissance, *Biocompress-2* donne les meilleurs taux de compression des séquences d'ADN. Le taux moyen de compression sur notre échantillon (choisi pour la longueur des séquences) est de 10.24%, deux fois et demi meilleur que le codage arithmétique d'ordre 2.

Nous présentons aussi un mode alternatif de compression, dit vertical. Contrairement au mode de compression classique (horizontal) où chaque séquence est comprimée indépendemment des autres, dans le mode vertical, une séquence est comprimée par rapport à une autre séquence. Ce mode peut conduire à des taux de compression plus significatifs que le précédent.

La compression des séquences d'ADN et des séquences d'ARN est intéressante d'un point de vue théorique. Dans le mode horizontal, elle conduit à un concept d'entropie des séquences génétiques, et permet la détection de régularités. Dans le mode vertical, elle peut permettre une mesure du taux de similitude entre des séquences d'espèces différentes, et aider ainsi à la construction d'arbres phylogéniques.

Le papier est organisé comme suit. Dans la section suivante, nous rappelons les principes des algorithmes de compression de textes, et interprétons leurs effets sur les séquences d'ADN. Nous présentons l'algorithme *Biocompress-2* dans la Section 3, puis une analyse de nos résultats ainsi que quelques perspectives en Section 4.

2. Algorithmes classiques

Les algorithmes de compression existant considèrent en entrée des fichiers ASCII. Une séquence d'ADN, de par son alphabet à quatre lettres, peut être stockée dans un fichier ASCII avec une taille égale à 25% de sa taille initiale, chaque caractère pouvant être codé sur deux bits au lieu d'un octet. Les taux de compression doivent donc être calculés au prorata de la cardinalité de l'alphabet.

Soient E la séquence d'entrée sur un alphabet à k lettres, et S la séquence de sortie sur un alphabet à k' lettres. Le taux de compression est donné par l'expression :

$$1 - \frac{\mid S \mid}{log_{k'}(k)\mid E \mid} = 1 - \frac{log_2(k')\mid S \mid}{log_2(k)\mid E \mid}.$$

Les expressions $log_2(k)$ et $log_2(k')$ expriment les facteurs de conversion des alphabets respectifs des séquences E et S en l'alphabet binaire, ce qui permet d'avoir des textes exprimés sur un même alphabet, et donc des longueurs comparables.

En particulier, les taux de compression d'une séquence d'ADN réalisés par les algorithmes classiques de compression doivent être calculés par la formule :

$$1 - \frac{4 \mid S \mid}{\mid E \mid},$$

car k étant égale à 4 (alphabet de la séquence d'ADN) et k' égale à 256 (alphabet ASCII de la séquence de sortie), $log_2(k) = 2$ et $log_2(k') = 8$, d'où $\frac{log_2(k')}{log_2(k)} = 4$.

Comme nous l'avons mentionné plus haut, il existe deux approches différentes de compression de textes, l'approche statistique et l'approche substitutionnelle. L'une des plus anciennes techniques de compression est basée sur la première approche. Il s'agit de l'algorithme de Huffman, qui consiste à coder les caractères suivant la probabilité de leur apparition. Les fréquences d'apparition des caractères sont contenues dans une table qui peut être construite statiquement avant l'étape de codage (modèle statique), ou dynamiquement avec des mises-à-jour pendant l'étape de codage (modèle adaptatif). Cet algorithme, aussi bien dans le modèle adaptatif que dans le modèle statique, échoue sur les séquences d'ADN. La raison de cet echec est simple. En effet, il est difficile de trouver un codage des quatre lettres A, C, G et T qui permet d'avoir des mots de code de taille inférieure à deux bits. D'autre part, les probabilités d'occurrence des caractères sont très proches. Néanmoins, si des blocs de quatre caractères sont codés sur un octet avant l'étape de compression, l'algorithme de Huffman tend à donner des taux de compression positifs (voir Figure 3 pour les taux de compression de *compact*, codage de Huffman adaptatif). Notons que dans ce cas, le taux de compression est donné par : $1 - \frac{|S|}{|E|}$, les séquences S et E étant sur l'alphabet ASCII ($k = k' = 256$, d'où $\frac{log_2(k')}{log_2(k)} = 1$).

Un autre algorithme basé sur l'approche statistique, le codage arithmétique [WIT 87], donne de meilleurs résultats sur les séquences d'ADN. C'est une variante du codage de Huffman. Dans Huffman, chaque caractère est codé séparément par un mot de code dont la longueur est proche du minimum théorique $-log_2(p_u)$, p_u étant la probabilité associée au caractère [LEL 87]. Or, la quantité $-log_2(p_u)$ n'est pas toujours un nombre entier. Dans ce cas, le mot de code affecté au caractère est l'entier immédiatement supérieur à $-log_2(p_u)$, d'où résulte une certaine redondance résiduelle. Pour pallier cet inconvénient, la méthode arithmétique propose d'affecter le codage sur des groupes de caractères, en représentant ces derniers par des valeurs numériques comprises entre 0 et 1.

Le codage arithmétique admet également deux modèles, le modèle statique et le modèle adaptatif. D'autre part, la séparation entre l'étape de calcul des fréquences et l'étape de codage permet d'avoir un modèle d'ordre supérieur, qui consiste à calculer la fréquence d'un caractère en considérant les caractères qui le précèdent, l'ordre étant le nombre de caractères précédents pris en compte. Dans le cas des séquences d'ADN, la probabilité d'apparition d'un caractère dépend de manière assez significative des caractères qui le précèdent [WAT 89, CUR 89]. Les taux de compression obtenus sur notre échantillon de séquences d'ADN par l'algorithme arithmétique d'ordre supérieur, *arith-n*, donné dans [NEL 91], atteignent leur maximum pour l'ordre $n = 2$. Les taux sont très similaires pour l'ordre 1 et 3 et décroissent après l'ordre 4 (voir Figure 3 pour les taux de compression de *arith-n* pour n=2).

Contrairement aux algorithmes basés sur l'approche statistique, ceux basés sur l'approche substitutionnelle donnent tous des taux de compression négatifs. Cette approche, appelée aussi codage par facteurs (les facteurs sont les différentes sous-séquences de la séquence initiale), consiste à coder les facteurs en les remplaçant par une référence à l'une de leurs précédentes occurrences. Les auteurs de cette approche, Ziv et Lempel, ont proposé deux méthodes pour rechercher les facteurs et les coder. Dans la première, utilisée pour la première fois dans l'algorithme *LZ77* [ZIV 77], les occurrences du facteur à coder sont recherchées dans une fenêtre, et le mot de code utilisé est une paire d'entiers constituée de la position et de la longueur de la première occurrence trouvée. Dans la deuxième méthode, utilisée pour la première fois dans l'algorithme *LZ78* [ZIV 78], le facteur est recherché dans un dictionnaire contenant les facteurs initialement codés, et est ensuite codé par le numéro qui lui est attribué dans le dictionnaire, dans le cas où il y apparaît.

Différentes extensions ont été proposées aux algorithmes *LZ77* et *LZ78*. Parmi elles, nous avons testé la méthode *LZSS* (extension de *LZ77*) [STO 82, NEL 91], ainsi que *LZW* [WEL 84, NEL 91] et *compress* (extensions de *LZ78*). L'algorithme *compress*, disponible dans l'environnement UNIX [THO 85], est basé sur le schéma *LZC* [BEL 89], légère variante de *LZW* (de par son implémentation). Le taux moyen de compression de *LZSS* sur notre échantillon de séquences d'ADN est de -36.15%, et celui de *compress* est de -7.49% (voir Figure 3 pour les taux de compression de *compress*).

L'inefficacité des algorithmes de codage par facteurs sur les séquences d'ADN est due à plusieurs raisons. D'abord, les mots de code utilisés ont été élaborés pour le codage de textes sur l'alphabet ASCII, d'où leur inadéquation au codage de textes sur des alphabets courts. D'autre part, la longueur et le nombre de facteurs détectés par ces algorithmes, dans le cas des séquences d'ADN, ne sont pas toujours satisfaisants. Les algorithmes basés sur le dictionnaire peuvent perdre un pourcentage important de facteurs répétés, car seuls les facteurs codés sont stockés dans le dictionnaire et il n'est possible de détecter un facteur répété que si une de ses occurrences apparaît dans le dictionnaire.

textes	tailles	Algorithmes			
		Statistiques		Substitutionnels	
		compact	*arith-2*	*LZSS*	*compress*
papier.tex	47683	37.66%	63.06%	54.46%	52.90%
Biocompress-2.c	20862	35.55%	69.51%	62.62%	56.67%
Biocompress-2.exe	204800	24.61%	42.66%	38.29%	24.65%

Figure 1. *Taux de compression d'algorithmes classiques sur des textes*

Les algorithmes utilisant une fenêtre présentent également des insuffisances, dues essentiellement à la petite taille (fixe) de la fenêtre. Si s est la largeur de la fenêtre et i est la position courante, la fenêtre est comprise entre le $(i-s)^{\text{ème}}$ et le $i^{\text{ème}}$ caractère. L'utilisation de petites fenêtres est efficace sur des textes dont la redondance est locale, mais ceci n'est pas le cas dans les séquences d'ADN où les redondances peuvent être éloignées et aussi très longues (pouvant ainsi dépasser la taille de la fenêtre).

Les résultats montrent l'inefficacité des algorithmes classiques sur les séquences d'ADN. Pourtant, ces mêmes algorithmes donnent des taux de compression très satisfaisants sur des textes classiques (voir Figure 1 pour les taux de compression réalisés par *compact, arith-2, LZSS* et *compress* sur le fichier Latex de ce papier et sur les fichiers source et exécutable du programme *Biocompress-2*).

3. L'algorithme Biocompress-2

L'algorithme *Biocompress-2* (voir Figure 2) a été conçu pour la compression des séquences d'ADN. Il peut également être utilisé sur les séquences d'ARN mais pas sur les séquences d'amino-acides des protéines. Les séquences protéiques sont sur un alphabet à vingt caractères et possèdent des propriétés complètement differentes des séquences d'ADN et d'ARN. *Biocompress-2* code un texte sur l'alphabet $\{$A, C, G, T$\}$ en une séquence binaire. Il est basé sur la première approche du codage de Ziv et Lempel, à laquelle il associe un codage statistique, le codage arithmétique d'ordre 2.

L'algorithme Biocompress-2 détecte et code deux types de répétitions, les répétitions simples et les répétitions palindromiques, c'est-à-dire les répétitions inverses et images, A étant l'image de T et vice versa, et C étant l'image de G et vice versa. Par exemple, soit la séquence $AACTTT$, sa répétition palindromique est $AAAGTT$, et le couple ($AACTTT$, $AAAGTT$) forme un “palindrome”.

```
début
i = 1;                                  * i : position courante *
tant que (i ≤ n)                        * n : longueur de la séquence *
  faire début
    trouver le plus long facteur f
      commençant à une position inférieure à i
      et égalant le facteur commençant à la position i
    soit l1 la longueur et p1 la position de f

    trouver le plus long facteur f'
      commençant à une position inférieure à i
      formant un palindrome avec le facteur commençant à la position i
    soit l2 la longueur et p2 la position de f'

    si | (l1, p1) | +c > 2l1
      alors
        si | (l2, p2) | +c > 2l2
          alors stocker le caractère de la position i dans un buffer;
                i = i + 1;
          sinon
            si buffer ≠ vide
              alors envoyer le codage arithmétique du contenu du buffer;
                    vider le buffer;
            fsi
            envoyer le mot de code (l2, p2);
            i = i + l2;
        fsi
      sinon
        si buffer ≠ vide
          alors envoyer le codage arithmétique du contenu du buffer;
                vider le buffer;
        fsi
        si (| l1 |>| l2 | ou | (l2, p2) | +c > 2l2)
          alors envoyer le mot de code (l1, p1);
                i = i + l1;
          sinon envoyer le mot de code (l2, p2);
                i = i + l2;
        fsi
    fsi
  fin
fin
```

Figure 2. *L'algorithme Biocompress-2*

Le processus de détection et de codage des facteurs est réalisé "on-line" par *Biocompress-2*. Ceci consiste en : un parcours séquentiel de la séquence à coder (de gauche à droite) ; une recherche du plus long facteur commençant à la position courante tel qu'il apparaît dans la partie de la séquence déjà parcourue et donc codée ; puis un codage de ce facteur. Dans le cas de détection de palindromes, nous recherchons le plus long facteur commençant à la position courante, tel que son image apparaît dans le sens contraire au sens de lecture, dans la partie de la séquence déjà parcourue.

Le facteur (simple ou palindrome) ainsi trouvé est codé par un mot de code correspondant à une paire d'entiers (l, p), tel que l est la longueur du facteur et p la position de la précédente occurrence. Dans le cas des facteurs simples, la position considérée est la position absolue (calculée par rapport au début de la séquence) du premier caractère de la première occurrence. Dans le cas des facteurs palindromes, p est la position relative (calculée par rapport à la position courante) du dernier caractère de la dernière occurrence du facteur.

Chaque caractère A, C, G ou T peut être codé sur deux bits (respectivement par 00, 01, 10, 11). En codant un facteur par la paire (l, p), il arrive souvent que ce mot de code soit plus long que la taille du facteur (dont les caractères sont codés sur deux bits). Pour éviter toute extension de la séquence initiale, nous codons un facteur par la paire (l, p) seulement si la taille de cette dernière est inférieure à la taille du facteur, c'est à dire $\mid (l, p) \mid + \; c \leq 2l$, tel que $\mid (l, p) \mid$ est la taille de la représentation binaire de (l, p) et c est une constante proportionnelle à la taille des bits de contrôle. Si aucun facteur ne satisfait cette condition, le caractère se trouvant à la position courante est rajouté à un buffer, et la recherche est répétée avec les facteurs commençant à partir du caractère suivant. On obtient ainsi une alternance de suites de paires (l, p) et de suites de caractères non codés.

Sur chaque suite de caractères est lancé un codage statistique, le codage arithmétique d'ordre 2. La table contenant les fréquences des différents caractères est mise à jour dynamiquement au niveau de chaque suite de caractères et est maintenue pour la prochaine suite de caractères. Dans le cas où la taille du mot de code délivré par le codage arithmétique est supérieure à la taille de la suite de caractères, ces derniers sont codés chacun sur deux bits.

La séquence de sortie contient finalement trois types de mots de code : des caractères sur des paires de bits, des valeurs numériques correspondant aux mots de code délivrés par le codage arithmétique, et des paires d'entiers correspondant aux couples (l, p). Devant chaque suite contenant le même type de mots de code est inséré un mot de code de contrôle, représenté par un entier donnant le nombre d'éléments de la suite. Ce mot de code est utilisé pour indiquer le changement de sémantique du codage, et permet ainsi la distinction entre les trois types de suites. D'autre part, un bit de contrôle placé au début de chaque couple (l, p) permet de différencier un facteur palindrome (bit = 1) d'un facteur simple (bit = 0).

Par exemple, la séquence suivante :

```
AACTGTTGTTGTTAGAACAACAGTT
      ++++++>
   ++++++>
               *********>
<*********
```

est codée dans l'expression :

```
6 AACTGT 1 0 (7,4) 2 AG 1 1 (10,5).
```

Dans cet exemple de séquence d'ADN, il est clair que le premier facteur ayant une précédente occurrence de longueur supérieure ou égale à 5 (5 étant dans cet exemple la longueur minimale d'un facteur à partir de laquelle nous supposons intéressant de le coder par le couple (l, p)) apparaît à la position 7 (représenté dans le schéma par la flèche +++++ >). Il est de longueur 7 et sa précédente occurrence se trouve à la position 4 (ce qui traduit leur chevauchement). Il est donc codé par le mot de code $(7, 4)$, qui est précédé du bit 0 étant donné que c'est un facteur simple. Le facteur intéressant à coder après celui-ci est un facteur palindrome apparaissant à la position 16 (représenté par la flèche ********* >). Il est de longueur 10 et sa précédente occurrence (la seule) est à la position $p = 5$, p étant la longueur séparant le dernier caractère de cette occurrence de la position courante. Le facteur est donc codé par le mot de code $(10, 5)$, précédé du bit 1. Notons que le codage arithmétique n'a pas été utilisé dans cet exemple.

Il existe dans les séquences de nucléotides certaines régions sur lesquelles *Biocompress-2* agit très positivement, et plus particulièrement lorsqu'il s'agit d'une suite de répétitions du même motif. Par exemple, cette structure de micro satellite : *GTGTGTGTGTGTGTGT* fréquemment rencontrée dans les séquences génomiques est codée comme suit :

2 GT 1 0 (14,1).

L'efficacité de la compression dépend de l'espace requis pour la représentation des entiers correspondant aux différents types de mots de code. Chaque mot de code, selon sa spécificité relative aux séquences d'ADN, est représenté de manière particulière. Les différentes représentations sont décrites dans [GRU 94].

Contrairement aux algorithmes classiques qui, généralement, ne détectent qu'une partie des facteurs existant dans le texte, *Biocompress-2* détecte tous les facteurs, quelque soient leur taille et leur position. Une fenêtre de taille variable est définie, comprise entre la position initiale de la séquence et la position courante. Pour la détection des facteurs se trouvant dans cette fenêtre, *Biocompress-2* utilise des structures de données simples. Il s'agit de deux automates construits on-line et en temps linéaire, l'un reconnaissant les facteurs simples et l'autre reconnaissant les facteurs inverses. Les deux automates sont des arbres 4-aires (quasi) complets, de profondeur h ($h = 8$ dans nos tests).

Le choix des ces automates a été influencé par le fait que dans les séquences d'ADN suffisamment longues (ce qui est le cas de notre échantillon), tous les mots courts (en général de longueur inférieure ou égale à 6 ou 7) apparaissent.

Chaque automate reconnaît donc tous les facteurs (simples ou inverses selon l'automate considéré), de taille inférieure ou égale à h, de la partie de la séquence déjà codée. Les positions de ces facteurs sont stockées dans l'automate. Considérons un facteur commençant à la racine de l'arbre et aboutissant à un noeud u. Si sa taille est inférieure à h, seulement la position de sa première occurrence (ou dernière s'il s'agit de l'automate des facteurs inverses) est stockée dans u. Si sa taille est égale à h, le noeud u, qui est dans ce cas une feuille, contient la liste des positions respectives de toutes ses occurrences. Les facteurs de taille supérieure à h sont détectés par une recherche sur l'automate suivie d'une recherche sur la séquence d'entrée.

Des automates plus complexes de reconnaissance de facteurs (simples) ont été utilisés dans des algorithmes de compression déjà existant, tels que l'automate des facteurs [BLU 85, CRO 88] utilisé dans [ZIP 92], et l'arbre des suffixes [McC 76] utilisé dans [ROD 81] et [FIA 89]. Ces deux structures sont construites en temps linéaire par rapport à la taille de la séquence et permettent une recherche des facteurs également linéaire, par un parcours séquentiel de la séquence de gauche à droite. Néanmoins, elles présentent l'inconvénient d'occuper une grande place mémoire lorsqu'il s'agit de comprimer des textes longs. Pour éviter des débordements, elles sont souvent réinitialisées au cours de la compression, d'où la perte de certains facteurs, essentiellement les plus éloignés. Dans la compression des séquences d'ADN, il est important de détecter tous les facteurs. La solution choisie par *Biocompress-2* est de réduire la taille de l'automate, au dépend du temps de recherche des facteurs.

Les deux structures citées (l'automate des facteurs et l'arbre des suffixes) ne reconnaissent que les facteurs simples, et sont difficilement adaptables pour la reconnaissance des facteurs inverses. Par contre, l'arbre des suffixes présenté dans [WEI 72, CHE 84], construit par un parcours de droite à gauche de la séquence, est facilement adaptable pour la construction de l'arbre des préfixes inversés (reconnaissant donc les facteurs inverses), en parcourant dans ce cas la séquence de gauche à droite.

4. Compression et compréhension

Les taux de compression obtenus avec l'algorithme *Biocompress-2* sur un échantillon de séquences d'ADN sont donnés dans le tableau des taux de compression (voir Figure 3) (où *Biocompress-2* est noté par *biocp-2*), ainsi que ceux de *compress*, *compact*, *arith-2*, et *Biocompress* (noté *biocp*). Les taux de compression de *Biocompress* et *Biocompress-2* sont donnés par : $1 - \frac{|S|}{2|E|}$, la sortie de ces derniers étant des fichiers binaires (E est la séquence d'ADN en entrée, et S la séquence binaire en sortie). Les taux de *compress* et *arith-2* sont donnés

séquences [1]	taille	Algorithmes				
		compress	*compact*	*arith-2*	*biocp*	*biocp-2*
MIPACGA	100314	−5.81%	5.09%	6.37%	0.48%	6.24%
MPOMTCG	186608	−10.11%	0.57%	1.72%	1.39%	3.11%
CHNTXX	155844	−9.36%	1.91%	3.31%	16.26%	19.14%
MPOCPCG	121024	−373%	6.46%	8.17%	8.44%	15.76%
YSCCHRIII	315357	−9.41%	1.69%	2.48%	1.62%	3.87%
HUMGHCSA	66495	−9.68%	0.66%	3.11%	32.75%	34.63%
HUMHBB	73323	−9.73%	1.72%	4.08%	3.22%	6.16%
HUMHDABCD	58864	−11.48%	0.06%	2.87%	3.86%	6.15%
HUMDYSTROP	38770	−11.66%	0.90%	3.80%	0.00%	3.69%
HUMHPRTB	56737	−10.12%	0.97%	3.56%	2.27%	4.67%
VACCG	191737	−8.37%	2.82%	5.10%	7.91%	11.93%
HS5HCMVCG	229354	−10.65%	0.58%	1.76%	5.90%	7.60%

Figure 3. *Tableau des taux de compression*

par : $1 - \frac{4|S|}{|E|}$, et ceux de *compact* par : $1 - \frac{|S|}{|E|}$.

Sur le tableau, nous pouvons noter que *Biocompress-2* réalise, pour chaque séquence de l'échantillon, un taux de compression proche du taux réalisé par *Biocompress* sur la même séquence, auquel est rajouté le taux réalisé par *arith-2*. Ceci démontre la quasi parfaite intégration des deux méthodes substitutionnelle et statistique.

1. Les séquences d'ADN du tableau incluent des génomes complets, des gènes et des virus. Voici leur description:
Génomes complets de deux mitochondries et deux chloroplastes, et la séquence complète du chromosome III de la levure:
MIPACGA, mitochondrie du podospora anserina;
MPOMTCG, mitochondrie du marchantia polymorpha;
CHNTXX, chloroplaste du tabac;
MPOCPCG, chloroplaste du marchantia polymorpha;
YSCCHRIII, saccharomyces cerevisiae (la plus longue séquence disponible).
Gènes humains:
HUMGHCSA, hormone de croissance de l'homme et gènes de chorionic somatomammotropin;
HUMHBB, région de la beta globine de l'homme sur le chromosome 11;
HUMHDABCD, séquence d'ADN humaine composée de 3 cosmids (HDAB, HDAC, HDAD);
HUMDYSTROP, gène dystrophine de l'homme;
HUMHPRTB, phosphoribosyltransferase hypoxanthine de l'homme (HPRT).
Génomes complets de virus:
VACCG, virus vaccinia;
HS5HCMVCG, race des cytomegalovirus humains AD169.

Les résultats de *Biocompress-2* sont variables. En général, les palindromes jouent un rôle important. Ceci est en particulier le cas pour la séquence CHNTXX (correspondant au génome du chloroplaste du tabac [SHI 86]), qui contient une répétition palindromique longue de plus de 25000 caractères. Dans le cas du gène HUMGHCSA, le succès de *Biocompress-2* est dû au nombre important de répétitions (environ sept cents facteurs de tailles supérieures à vingt caractères).

Grâce aux techniques utilisées dans la compression des séquences d'ADN, nous avons pu mettre en évidence des régularités de l'ADN. Avant de comprimer les séquences de nucléotides, nous avons réalisé la recherche des répétitions simples et palindromiques, mais aussi inverses (exemple: $AAGTCT$ et $TCTGAA$) et images (exemple: $AAGTCT$ et $TTCAGA$). Nous avons constaté que ces deux types de répétitions (répétitions inverses et images) apparaissent en très faible quantité par rapport aux deux premiers (répétitions simples et palindromiques). Ceci peut s'expliquer par le fait que les répétitions simples et surtout les répétitions palindromiques ont plus de signification biologique. En effet, les répétitions simples peuvent correspondre à une duplication du même gène et donc à une transcription multiple de ce dernier. C'est le cas par exemple, pour les eukaryotes, du gène formé de l'ARNr 18S et de l'ARNr 28S qui est dupliqué plusieurs fois sur un même brin d'ADN. Pour les répétitions palindromiques, en plus du rôle important qu'elles jouent dans la formation des structures secondaires d'ARN, elles correspondent sans doute au phénomène biologique suivant : une séquence d'ADN est orientée et correspond à l'un des brins d'une double hélice (sens de lecture 5'-3'). L'autre brin est son image, et a une orientation opposée. L'existence d'un palindrome (f, f') sur un brin implique l'occurrence du même palindrome sur le brin image, et donc l'occurrence des mêmes facteurs f et f' sur les deux brins (voir Figure 4). Cette régularité purement formelle confirme l'existence de deux brins images l'un de l'autre, de sens de lecture opposés. D'autre part, certaines séquences d'ADN présentent une duplication de parties codantes sur les deux brins, comme c'est le cas par exemple pour la séquence génomique du chloroplaste du tabac (CHNTXX), qui présente une répétition palindromique de plus de 25000 nucléotides, correspondant à un groupe de plusieurs gènes.

Les taux relativement faibles obtenus par *Biocompress-2* nous conduisent à aborder le problème de la compression sous un autre angle. Nous discutons brièvement un autre mode de compression, la compression verticale, qui consiste à comprimer une séquence par rapport à une autre. Ce mode de compression n'a de sens que si les deux séquences sont très similaires. Or, les bases de données génétiques contiennent une grande quantité de séquences similaires, telles que le même gène (associé à une même fonction biologique) dans différentes espèces (les gènes de l'hémoglobine en sont un exemple). Une séquence de référence peut être stockée dans la base de données, tandis que d'autres séquences seront stockées sous une forme comprimée par rapport à la première.

```
               f                f'
           --------->      --------->
5' -------AAGTTCGAGT------ACTCGAACTT----- 3'     brin d'ADN


3' -------TTCAAGCTCA------TGAGCTTGAA----- 5'     brin image
           <---------      <---------
               f'               f
```

Figure 4. *Apparition d'un même facteur sur les deux brins d'ADN*

Pour ce mode de compression, nous n'avons actuellement que des résultats partiels. Nous avons mis en oeuvre un algorithme de compression verticale, *Biocompress-vertical*, qui comprime une séquence S_1 par rapport à une séquence S_2 en codant les facteurs qui apparaissent dans S_1 par une référence à l'une de leurs occurrences dans S_2. La référence est représentée, de la même manière que dans *Biocompress-2*, par le couple (l, p), où l est la longueur du facteur à coder et p la position de sa première occurrence dans S_2 (position calculée par rapport au début de la séquence S_2). Notons que dans ce cas, la fenêtre considérée concerne toute la séquence S_2, structurée en un automate semblable à celui utilisé dans *Biocompress-2*.

Dans certains cas, nous obtenons de bons taux de compression, mais cela reste très lié au degré de similitude entre les séquences, et surtout à la manière dont sont réparties les différences que présentent entre elles les séquences, différences traduites par des substitutions, insertions ou suppressions de nucléotides. En effet, si celles-ci sont éparpillées, les taux de compression résultant sont moins bons que si elles sont groupées. Ceci est dû essentiellement au codage. Lorsque les différences sont groupées, la région dans laquelle elles apparaissent n'est pas codée mais le reste contient de grands facteurs intéressants à coder. Lorsque les différences sont réparties, les facteurs sont en plus grand nombre mais de petite taille, par conséquent, d'un coût de codage plus élevé.

La compression verticale permet ainsi de définir une *distance* entre des séquences (deux à deux), calculée en fonction du nombre et de la longueur des facteurs communs. Cette *distance* n'est pas tout à fait la même que celle définie en biologie, celle-ci étant mesurée par le nombre de différences que présentent entre elles les séquences, indépendamment de leurs positions respectives. Néanmoins, comme nous pouvons le constater dans ce qui suit, il existe une certaine corrélation entre les deux.

La compression verticale peut être liée au problème de construction d'arbres phylogéniques. La Figure 5 représente un fragment d'un arbre phylogénique de

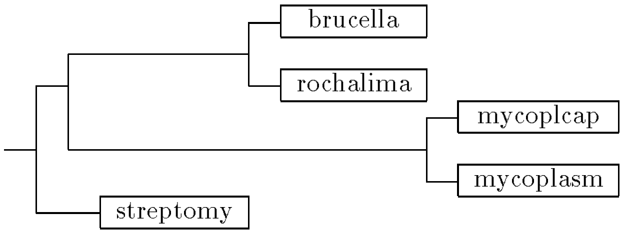

Figure 5. *Fragment d'un arbre phylogénique*

quelques espèces de bactéries, construit à partir de similitudes entre leur ARN ribosomal 16S (ARNr 16S) [GOU 93]. Deux espèces sont proches dans l'arbre (la *distance* les séparant est faible) si leurs séquences diffèrent peu.

Les résultats de *Biocompress-vertical*, entre des paires de séquences d'ARNr 16S des espèces de bactéries données dans la Figure 5, sont donnés ci-dessous.

	streptomy	brucella	rochalima	mycoplasm	mycoplcap
streptomy	99.18%	9.73%	13.18%	5.67%	5.63%
brucella	10.85%	99.12%	55.95%	7.49%	7.31%
rochalima	13.39%	52.91%	99.16%	8.10%	7.73%
mycoplasm	5.74%	6.69%	8.04%	99.17%	84.90%
mycoplcap	5.78%	6.50%	7.75%	85.50%	99.17%

Les taux reflètent les distances entre les séquences dans le fragment de l'arbre phylogénique de la Figure 5. Par exemple, la compression de la séquence du *mycoplcap* par rapport à la séquence du *mycoplasm* donne un taux de 85.50%, tandis que sa compression par rapport à la séquence de la *streptomy* donne un taux de 5.78%.

Nous travaillons actuellement sur des extensions de *Biocompress-2* dans différentes directions. Des répétitions dans les gènes sont généralement des répétitions avec trous, c'est-à-dire des répétitions présentant quelques différences, ces dernières pouvant être des substitutions, insertions ou suppressions de nucléotides. La présence de trous diminue considérablement l'efficacité de l'algorithme *Biocompress-2* qui ne les prévoit pas. Pour cela, il serait intéressant de pouvoir détecter et coder ce type de répétitions. Les répétitions à trous ont également une pertinence dans les structures secondaires d'ARN. En effet, les palindromes à trous forment le plus souvent des "hélices", qui constituent des entités importantes dans les structures secondaires.

D'autre part, la structure secondaire des ARNs est basée non seulement sur les appariements $A-T$ et $C-G$, mais aussi sur l'appariement $T-G$ (qui est néanmoins moins important que les deux premiers). Ceci implique que pour un mot donné, il est possible d'avoir plusieurs facteurs palindromes. Par exemple, le mot $AACATG$ peut être apparié aux mots suivants : $CATGTT, TATGTT,$

CGTGTT, ou *TGTGTT*. Il serait donc également intéressant de détecter ces types de palindromes. En effet, ceci augmenterait le nombre de facteurs trouvés, ainsi que leur taille, et donc potentiellement les taux de compression.

La structure secondaire est elle même une régularité intéressante pour la compression. Elle peut être utilisée pour le codage de manière similaire des séquences ayant la même structure secondaire.

Des régions codantes et non codantes dans les gènes ont déjà été séparées par des analyses statistiques de la périodicité de l'alternance des purines et des pyrimidines [ARQ 90]. L'entropie résultant du degré de la compression peut fournir de nouveaux moyens pour la séparation des régions codantes et non codantes.

Finalement, les techniques de compression peuvent servir à l'optimisation des algorithmes sur les séquences. Elles ont initialement été utilisées pour faciliter le stockage et la transmission des données. Elles peuvent également être utilisées pour l'optimisation d'autres algorithmes en réduisant la taille de leurs données d'entrée. Nous pensons plus particulièrement à la possiblité de développer des algorithmes d'alignement travaillant directement sur des séquences comprimées sous le mode vertical.

5. Bibliographie

[ARQ 90] Arquès D.G. et Michel C.J., Periodicities in coding and noncoding regions in genes, *Journal of Theoretical Biology*, 143, 1990, p. 307-318.

[BEL 89] Bell T., Witten I.H. et Cleary J.G., *Modeling for Text Compression, ACM Computing Surveys*, 21, *n*° 4, 1989, p. 557-591.

[BLU 85] Blumer A., Blumer J., Ehrenfeucht A., Haussler D. Chen M.T. et Seiferas J., The smallest automaton recognizing the subwords of a text, *Theoret. Comput. Sci.*, 40, 1985, p. 31-56.

[CHE 84] Chen M. T. et Seiferas J., Efficient and Elegant Subword Tree Construction, *Proceeding of the NATO Advanced Research Workshop on Combinatorial Algorithms on Words*, 1984, p. 97-107.

[CRO 88] Crochemore M., String matching with constraints, *Mathematical Foundations of Computer Science* (1988), Chytill, Janiga&Koubek eds, Springer-Verlag, p. 44-58.

[CUR 89] Curnow R. et Kirkwood T., Statistical analysis of deoxyribonucleic acid sequence data - a review, *J. Royal Statistical Society*, 152, 1989, p.199-220.

[FIA 89] Fiala E.R. et Greene D.H., Data compression with finite windows, *Communications of the ACM*, 32, *n*° 4, 1989, p. 490-505.

[GOU 93] Gouy M., Communication personnelle, 1993.

[GRU 93] Grumbach S. et Tahi F., Compression of DNA sequences, *Proc. IEEE Symp. on Data Compression*, 1993, p. 340-350.

[GRU 94] Grumbach S. et Tahi F., A new challenge for compression algorithms: Genetic sequences, *Journal of Information Processing and Management*, 30, *n*° 6, 1994.

[HUF 52] Huffman D.A., A method for the construction of minimum-redundancy codes, *Proceeding IRE*, 40, 1952, p. 1098-1101.

[LEL 87] Lelewer D.A. et Hirschberg D.S., Data compression, *ACM Computing Surveys*, 19, n^o 3, 1987, p. 261-287.

[LEM 76] Lempel A. et Ziv J., On the complexity of finite sequences, *IEEE Trans. Inform. Theory*, 22, n^o 1, 1976, p. 75-81.

[McC 76] McCreight E.M., A space-economical suffix tree construction algorithms, *J. ACM*, 23, n^o 2, 1976, p. 262-272.

[NEL 91] Nelson M., *The Data Compression Book*, M&T Publishing Inc., 1991.

[OLI 92] Oliver S.G., et al., The complete DNA sequence of yeast chromosome III, *Nature*, 357, 1992, p. 38-46.

[ROD 81] Rodeph M., Pratt V.R. et Even S., Linear Algorithm for Data Compression via String Matching, *Journal of the association for computing matchinery*, 28, n^o 1, 1981, p. 16-24.

[SHI 86] Shinozaki K., Ohme M., anaka M., Wakasugi T., et Hayashida N., The complete nucleotide sequence of the tobacco chloroplast genome : its gene organisation and expression, *The EMBO Journal*, 5, n^o 9, 1986, p. 2043-2049.

[STO 82] Storer J.A. et Szymansk T.G., Data compression via textual substitution, *Journal of the ACM*, 29, n^o 4, 1982, p. 928-951.

[STO 88] Storer J.A., *Data Compression, methods and theory*, Computer Science Press, 1988.

[THO 85] Thomas S.W. et al., Compress (version 4.0) program and documentation.

[WAT 89] Waterman M.S., *Mathematical Methods for DNA Sequences*, CRC Press, 1989.

[WEI 72] Weiner P., Linear pattern matching algorithms, *IEEE Symposium on Switching and Automata Theory*, 14, 1972, p. 1-11.

[WEL 84] Welch T.A., A technique for high performance data compression, *Computer*, 1984, p. 8-19.

[WIT 87] Witten J.H., Neal R.M. et Cleary J.G., Arithmetic Coding for Data Compression, *communications of the ACM*, 30, n^o 6, 1987, p. 520-540.

[ZIP 92] Zipstein M., Data compression with factor automata, *Theoretical Computer Science*, 1992, p. 213-221.

[ZIV 77] Ziv J. et Lempel A., A universal algorithm for sequential data compression, *IEEE Transactions on Information Theory*, 23, n^o 3, 1977, p. 337-343.

[ZIV 78] Ziv J. et Lempel A., Compression of individual sequences via variable-rate coding, *IEEE Transactions on Information Theory*, 24, n^o 5, 1978, p. 530-536.