Compression pour l'analyse de séquences*

E.Rivals
Laboratoire d'Informatique Fondamentale de Lille
U.A. 369 du C.N.R.S., Université de LILLE I
59655 Villeneuve d'Ascq. FRANCE.
email : rivals@lifl.fr

22 novembre 1995

Résumé

Deux objectifs ont guidé la rédaction de cet article.

Tout d'abord, la volonté de piquer la curiosité des personnes qui gèrent de grands ensembles de données génétiques, en montrant explicitement les résultats de notre travail sur la compression de fichiers de banques génétiques. Ces résultats évaluent le gain en place disque ou en temps de transmission qu'offre la compression.

Ensuite, nous désirons aborder la problématique de la compression de texte comme outil d'analyse de séquences. Il s'agit de montrer la compression sous son versant d'outil de compréhension du message génétique, parce qu'elle en permet une analyse multi-critères. Un algorithme de compression basé sur une régularité est un critère de classification des séquences. Ce second volet représente notre sujet majeur de recherche.

1 Introduction

Nous abordons deux cadres différents d'utilisation de la compression. Dans le premier, nous poursuivons un objectif pragmatique de compression de fichiers de données génétiques. Ces fichiers proviennent de banques de données dites "à plat", i.e. devant être manipulables par ordinateur, mais aussi par un utilisateur non assisté d'une machine. Pour cette raison, ces fichiers ne sont pas compressés. En partant de ce constat, nous proposons une méthode de compression capable d'éviter le stockage ou la transmission d'une grande proportion de ces données. Cette méthode code séparément les commentaires en langue naturelle et la séquence génétique.

*Published in Maxime Crochemore editor, *Actes de la rencontre pour les Recherches de Motifs dans les Séquences, GREG, Marseille feb-94*. Institut Gaspard Monge, Université de Marne La Vallée, 1994.

Le second cadre d'utilisation est exclusivement focalisé sur la compression de séquences génétiques, comme moyen d'investigation et de compréhension du message génétique. Nous présentons tout d'abord la problématique générale de la compression de texte, puis notre démarche méthodologique d'utilisation de la compression pour l'analyse de séquences. Ensuite vient l'état des développements actuels, suivis de résultats comparatifs, puis les perspectives de notre recherche.

2 Compression de fichiers de données génétiques

2.1 Les motivations

Des projets de recherche nationaux et internationaux, unissent des centaines des laboratoires pour l'étude et le séquençage de plusieurs génomes. Le rythme de séquençage actuel prévoit un doublement des données disponibles chaque année. En outre, un texte commente chaque séquence produite par un laboratoire pour qu'elle serve à d'autres biologistes moléculaires. Etant donné, l'état de nos connaissances et la diversité des fonctions codées par les gènes, ces commentaires évoluent, changent de forme et d'importance ; les années à venir verront la création de multiples bases de données spécifiques qui adapteront les commentaires à leur domaine de "compétence". Ce mouvement a déjà commencé et nous citons l'exemple de la base LIGM dédiée aux séquences d'Immunoglobulines et conçue à Montpellier.

Ces deux constatations, ainsi que le fait qu'elle n'existe pas, nous motive pour créer une méthode de compression spécifique à ce type de fichiers, tout en étant indépendante du format et des commentaires choisis. Nous voulons par cette tentative susciter des interrogations de la part des gestionnaires de base de données, sur le gain potentiel de place disque et de durée de télécommunication, qui résulte de l'utilisation d'un compresseur.

2.2 La méthode

La figure 1 montre un exemple de fichier EMBL, aussi appelé entrée EMBL, auquel nous avons supprimé quelques lignes par commodité. Le fichier se divise en deux groupes de lignes. Celles débutant par six caractères blancs, contiennent la séquence qui fait l'objet du fichier ; toutes les autres concernent la mise en forme et les commentaires, elles sont écrites en langue naturelle. D'une part, la compression de texte est un domaine fort bien connu et de nombreux algorithmes lui sont dédiés. D'autre part, nous voyons qu'une séquence génétique ressemble peu à un texte en langue naturelle et nous savons qu'elle se comprime fort mal en comparaison de ceux-ci, confère la sous-section 3.3.2. Dès lors, nous proposons de séparer ces deux types de données présents dans un tel fichier, pour leur appliquer un algorithme de compression adapté.

```
ID      HSA00A022 standard; RNA; PRI; 313 BP.
XX
AC      Z15193;
XX
DT      29-SEP-1992 (Rel. 33, Created)
DT      29-SEP-1992 (Rel. 33, Last updated, Version 1)
XX
DE      H. sapiens partial cDNA sequence; clone 00A02; strand (+)
DE      read.
XX
KW      partial cDNA sequence; transcribed sequence fragment.
XX
OS      Homo sapiens (human)
OC      Eukaryota; Animalia; Metazoa; Chordata; Vertebrata; Mammalia;
OC      Theria; Eutheria; Primates; Haplorhini; Catarrhini; Hominidae.
XX
RN      [1]
RP      1-313
RA      Genexpress;
RT      ;
RL      Submitted (25-SEP-1992) to the EMBL Data Library by:
RL      Genethon, B.P. 60, 91002 Evry Cedex France and Genetique
XX
FH      Key             Location  Qualifiers
FH
XX
SQ      Sequence 310 BP; 94 A; 55 C; 61 G; 100 T; 0 other;
        gggaaaggga  ccccacttgc  tcccatgccc  acctcaagaa  aaaacataaa  acaatttttt
        ttaaaaaaga  aaagaaatct  acctcagttg  acaggattcc  acctttaggg  tttcttcaac
        ttttaagtct  tacctgttga  gtgtaacttt  tgtagcatct  tgcttttcca  agcaagctag
        tgaggcatga  cagagcagaa  gtctgtaaat  gtccctgtga  tggcctcttt  ctagcatgtt
        gcagttttat  ttttaataaa  ttggtaagtg  aaatgaatgt  aaaggtaatt  tgtgtacggt
        tttagacatg
//
```

FIG. 1 - *Une entrée de la base EMBL*

Pour le texte en langue naturelle, nous choisissons d'appliquer une méthode dynamique de Lempel-Ziv basé sur l'algorithme dit LZ77, parce qu'elle donne de bons résultats en moyenne sur ce type de textes, confère [BCW90]. Par contre, nous ne pouvons agir de même avec les données génétiques, car il n'existe pas d'algorithme standard pour elles. L'idée la plus simple consiste à transcoder la séquence de son format actuel ASCII 7 bits, en un format où quatre bases consécutives sont codées dans un octet. Le nouveau format code une base sur deux bits : 'A' sur '00', 'C' sur '01', etc...

L'algorithme agit donc en deux phases :

- séparation des deux types de données dans deux fichiers différents
- compression du texte par un méthode LZ77 et simple transcodage de la séquence.

Pour que l'on puisse retrouver le fichier original par décompression, il convient de garder les indications de l'emplacement de la séquence dans le reste du texte. Nous avons décidé de conserver ces informations sous forme de pointeur dans le fichier texte et non dans le fichier séquence. En effet, avec l'option contraire, ce dernier ne serait plus un texte sur un alphabet de quatre lettres et perdrait par là, sa caractéristique d'être propice au transcodage de quatre bases sur un octet.

Si nous avons dans un fichier :

CC Pribnow box in position 195 "TAAATT";

nous le séparons ainsi, dans le fichier texte :

CC Pribnow box in position 195 "# 6.";

où "# 6." indique qu'il y a 6 caractères de la séquence à insérer à cette position, tandis que dans le fichier séquence, on place :

TAAATT

On considère que toute suite d'au moins 5 bases consécutives, appartient à la séquence. Pour les repérer, il suffit de compter qu'un nucléotide est suivi ou précédé de 4 autres nucléotides dans le fichier original. La séparation est donc de complexité linéaire en temps et constante en espace, la compression aussi.

2.3 Les résultats

Nous présentons ici des résultats de compression de fichiers en provenance de la base de l'EMBL et de GenBank. Le logiciel employé pour la compression de texte pur se nomme gzip, est gratuit et disponible sur Unix. Le tableau comparatif est construit de la manière suivante (de gauche à droite) :

- le nom du fichier compressé
- la taille initiale du fichier en Kilo-octets
- sa taille après compression par Separe_Compresse en Kilo-octets

- le taux de compression réalisé par Separe_Compresse, en pourcent. Il donne la proportion de réduction obtenue par la méthode (ici Separe_Compresse). Il est calculé ainsi, si T_i est la taille initiale du fichier et T_f sa taille après compression :

$$1 - (T_f / T_i)$$

- le taux de compression réalisé par gzip, en pourcent
- le gain de compression de Separe_Compresse par rapport à gzip, en pourcent

Fichier	Taille initiale Ko	Taille finale Ko	Separe_Compresse %	gzip %	Gain %
EMBL					
genexpress.embl.33	1.891	163	91	88	25
genexpress.embl.34	766	59	92	89	24
genexpress.embl.35	211	18	92	87	30
genexpress.embl.36	1203	89	93	89	27
genexpress.embl.38	419	29	93	90	28
GenBank					
gb.diff.daily.01-Jan-1994	478	101	79	72	25
gb.diff.daily.02-Jan-1994	1.977	322	84	79	22
gb.diff.daily.03-Jan-1994	187	5	97	96	29
gb.diff.daily.16-Dec-1993	1.180	211	82	76	24
gb.diff.daily.17-Dec-1993	482	106	78	70	26

2.4 Commentaires, avantages et inconvénients

Le premier constat est évident : les fichiers de données génétiques se compressent énormément, et ce quel que soit le format choisi : EMBL ou GenBank. En autre, notre méthode tire un grand bénéfice de la séparation des deux types de données contenus dans ces fichiers : les commentaires en langue naturelle et la séquence génétique, qui est une suite de bases prises dans un alphabet de quatre lettres. Par rapport à un algorithme classique de type LZ77 (gzip), le taux de réduction moyen supplémentaire dépasse les 26%. A l'utilisation, cela se traduit par un gain appréciable en place disque ou en temps de transmission sur un réseau.

Citons quelques avantages et inconvénients. Tout d'abord, l'algorithme est efficace quant à la réduction de la taille mémoire occupée, mais surtout il est indépendant du format des fichiers de la base de données et peut donc être utilisé avec les formats existants, comme avec un format d'une nouvelle base.

Par ailleurs, nous soulignons qu'il est applicable aux fichiers de séquences génétiques, mais non à ceux contenant des séquences protéiques. Il présente une amélioration par rapport à l'utilisation d'un compresseur classique, de sorte qu'il peut être bénéfique de changer. Néanmoins, l'utilisation à grande échelle de notre méthode nécessite la prise en compte d'aspects pratiques importants:

comme la possibilité d'indexer les données compressées, ou encore la compatibilité de celles-ci avec les logiciels actuels de manipulation de fichiers de données génétiques.

3 Compression pour l'analyse de séquences

L'objectif de cette section est avant tout d'expliquer pourquoi la compression est un outil d'étude d'un type de texte et pourquoi il peut s'appliquer à l'analyse du génome. Pour cela, nous détaillons tout d'abord la problématique de la compression et son application à la génétique, puis nous abordons l'état actuel des travaux faits en ce sens (dont les notres), pour illustrer comment le problème peut se résoudre pratiquement. Ensuite, nous comparons et commentons les résultats de deux algorithmes de compression. Enfin, nous terminerons par les perspectives de notre travail.

3.1 La problématique de la compression

Nous définissons informellement la compression d'un texte, comme l'étude d'un type de régularité, pour recoder le texte de manière plus courte. Comme exemple de régularités, nous pouvons citer la présence de facteurs répétés ou la répartition des symboles du texte. Le problème en soit est comment identifier, repérer et recoder de manière avantageuse une régularité du texte. On peut y déceler deux buts : la compression pour la réduction ou la compression pour la compréhension de la structure du texte. L'avantage est que la compression ne trompe pas : si régularité elle trouve, régularité il y a. Elle est algorithmiquement objective, chose plus difficile pour un observateur humain.

Nous allons examiner plus en détail quelques exemples précis pour illustrer notre propos.

3.1.1 Quelques exemples de compression

Huffman La méthode d'Huffman exploite une régularité globale du texte, qui est l'équirépartition de l'emploi des symboles dans le texte. Le codage consiste à remplacer chaque symbole par un code binaire d'autant plus court (resp. long) que le symbole est fortement (resp. faiblement) représenté. A priori, il faut des codes binaires de taille n, où $n = \lceil \log_2 |A| \rceil$, pour un texte écrit à l'aide d'un alphabet de taille $|A|$. En employant le code d'Huffman, on atteint une limite théorique de compression avec des codes de taille fixe : l'entropie de la source.

Kolmogorov Une méthode de compression qui associe à une séquence binaire, le plus court programme capable d'engendrer cette dite séquence sur une Machine de Turing Universelle, est optimale. La longueur de ce plus court programme s'appelle la complexité de Kolmogorov de cette séquence, et constitue la limite théorique de compression possible de cette séquence. Cette méthode, quoique non praticable, exploite toutes les régularités quel qu'en soit le type.

Compression par détection de répétitions Prenons en exemple un type de régularité plus intéressante pratiquement : la présence de facteurs répétés dans un texte (par facteur, nous entendons une sous-séquence contiguë du texte). Toute apparition du facteur autre que la première, peut être remplacée par un pointeur sur cette dernière. Ainsi, pour un facteur assez long, le texte est compressé. Nous appelons cette méthode : compression par détection de répétitions. Pour celle-ci, aucune limite théorique spécifique n'a été déterminée.

3.1.2 Compression comme mesure de régularités

La méthode d'Huffman et la compression par détection de répétitions montrent qu'un algorithme de compression mesure la "présence de régularités" dans un texte. La compression par détection de répétitions, non seulement isole les facteurs répétés du texte (comme peut le faire un algorithme de pattern-matching), mais surtout mesure la quantité de répétitions dans le texte.

On peut généraliser cette utilisation de la compression à bien d'autres régularités. En fait, toute présence de régularités réelles dans un texte, doit permettre de le compresser. Un algorithme de compression quantifie la présence de véritables régularités, en ce sens qu'il exploite celles qui lui permettent de compresser.

Il s'agit d'une mesure objective, capable de distinguer une régularité exploitable, d'une "régularité due au hasard" (cette dernière expression est un non-sens, puisque par définition le hasard n'engendre aucune régularité ; mais nous signifions qu'elle peut apparaître comme une régularité pour un expérimentateur humain tentant de calculer cette mesure). Cette objectivité constitue le grand avantage de la méthode.

Dès lors, pour comprendre une classe de texte, on peut choisir des types de régularité pertinents pour son étude, construire un algorithme de compression basé sur un type précis de régularité, et enfin, en mesurer la présence sur les exemples de textes à analyser. On engendre ainsi, pour chaque type de régularité et d'algorithme associé, une classification des textes utile à leur compréhension, classification que l'on peut qualifier de multi-critères.

Cette démarche méthodologique constitue notre approche d'étude des séquences génétiques (qui sont des textes sur un alphabet de quatre lettres). Dans leur cas, les régularités pertinentes sont diverses et représentent autant d'axes de recherche. Nous envisageons les répétitions exactes ou approximatives parce qu'elles proviennent des mécanismes de réplication présents dans l'évolution des génomes ; ainsi que les répétitions inversées et symétriques ou palindromes génétiques, car ils permettent les appariements dits "en tige" de portions d'ARN et ont donc un sens pour le biologiste moléculaire. On peut aussi envisager la répartitions des codons à l'intérieur des séquences codantes, déjà étudiée statistiquement dans [Dan93b], et bien d'autres régularités encore.

L'objectif de nos recherches n'est donc pas de compresser pour réduire, comme il en est question dans la deuxième section, mais de compresser pour comprendre.

3.2 Etat actuel des travaux

Nous allons décrire deux exemples de travaux de compression de séquences génétiques, comme illustration pratique de la démarche expliquée ci-dessus. En premier, nous décrivons ceux de l'équipe de l'INRIA Rocquencourt, i.e. leur algorithme Biocompress et son amélioration Biocompress 2. Puis nous examinons les divers compresseurs fruits du travail du LIFL. Ensuite, nous comparons l'efficacité de compression de ces deux méthodes.

3.2.1 Biocompress

Développé à Rocquencourt, Biocompress résulte de la première tentative dans notre communauté, de compression de séquences biologiques.

Biocompress est basé sur le parcours séquentiel de gauche à droite (ou $5' - 3'$) de la séquence. Il détecte et exploite deux types de régularités déjà citées : répétitions exactes et palindromes génétiques. D'une séquence sous forme ASCII, il produit un fichier binaire.

A chaque position dans la séquence, Biocompress :

- recherche le plus long palindrome et le plus long facteur dans la partie de la séquence déjà examinée (i.e. dans une fenêtre allant du premier caractère au caractère courant), apparaissant aussi à la position courante

- si l'un des deux est suffisament long, il est remplacé dans le texte par un code indiquant sa longueur et sa position de première apparition. Notons ce code (l, p)

La condition "suffisament long" signifie : le code (l, p) que l'on désire substituer à une apparition du facteur, est-il plus court que le facteur lui-même ? Si c'est le cas, on choisit le code de remplacement et la position suivante examinée est la première à la suite du facteur recodé. Sinon, le caractère courant est gardé tel quel et le caractère suivant devient position courante.

Pour la recherche de facteur, Biocompress utilise un automate donnant les positions d'apparition de tous les facteurs de longueur inférieure ou égale à 8. Par commodité, appelons facteur courant, la sous-séquence débutant à la position courante. Ainsi, pour chercher les apparitions précédentes du facteur courant, on cherche les positions d'apparition de ses préfixes grâce à l'automate, et seulement si elles existent, on effectue un comparaison lettre à lettre entre les facteurs commencant à ces positions et le facteur courant. Le même système est employé pour la recherche des palindromes.

Nous ne détaillons pas les codages utilisés par l'algorithme, pour plus de précision se reporter à [GT93a]. Remarquons cependant, le codage global comprend alternativement une portion de séquence dite non codée (i.e. simplement transcodée d'une base sur deux bits), et des réapparitions de facteurs remplacés par un code (l, p). En outre, Biocompress exploite des facteurs éventuellement imbriqués dans un autre ou le chevauchant.

3.2.2 Biocompress 2

Il s'agit d'une deuxième version de Biocompress. Les portions de séquence non codées, simplement transformées en binaire dans la version 1, peuvent bénéficier d'un codage arithmétique d'ordre deux. En effet, il peut être avantageux de coder ces portions par une technique de compression classique. Après comparaison de ces méthodes sur des textes génétiques (dont les résultats se trouvent dans [GT93b]), le choix fut arrêté sur un codage arithmétique dynamique d'ordre 2 (algorithme trouvé dans [Nel91]).

3.2.3 Notre approche

Les algorithmes que nous développons montrent une approche progressive et différente du problème. Progressive, parce que les programmes traitent des problèmes de plus en plus complexes ; différente dans le choix des occurrences des régularités compressées.

Principalement, deux types de régularités ont été envisagé :

- les facteurs répétés ou répétitions directes
- les palindromes génétiques ou répétitions inverses et symétriques selon les appariements de Watson-Crick.

Nous avons d'abord traité l'une séparemment de l'autre, chacune a son algorithme, puis les deux simultanément dans un algorithme plus général. Nous allons décrire l'algorithme qui compresse en n'exploitant que les répétitions directes. Les autres programmes sont calqués sur le même modèle. Les deux premiers compresseurs dans l'ordre chronologique de conception, i.e. les algorithmes de compression par détection de répétitions et par détection de palindromes, sont expliqués dans [Riv93], on y trouve une description détaillé de la procédure de recherche des régularités.

Les occurrences précises des régularités, i.e. les facteurs répétés et les palindromes génétiques, sont recherchés exhaustivement à l'aide de l'arbre des suffixes du texte, mieux connu sous le nom de Suffix Tree. C'est une volonté initiale, que d'effectuer la compression en connaissance de toutes les opportunités de réduction (chaque facteur répété différent représente une opportunité de réduction).

Pour compresser, il faut choisir une combinaison de certaines de ces opportunités et en laisser d'autres. La compression se rapproche d'un problème d'optimisation combinatoire. Dans notre cas, il s'agit de trouver une bonne combinaison de régularités dans un espace de recherche extrêmement grand. L'exploration de tout cet espace est exclu, et c'est à priori le seul moyen de déterminer si une combinaison est la meilleure.

Nous avons opté pour une heuristique de choix simplificatrice. D'une part, nous écartons tout chevauchement d'un facteur sur un autre ou sur lui-même. D'autre part, les facteurs sont examinés dans l'ordre décroissant de taille. L'heuristique décrite ici avec les facteurs, est aussi employée dans les autres algorithmes.

Le choix d'un facteur pour compresser le texte n'est jamais remis en cause. Celui-ci est supposé de taille suffisament grande, pour que le remplacement d'une de ces réapparitions, apporte un gain de compression. A cet effet, chaque algorithme est paramétré par la taille minimale des facteurs recherchés, pour être ensuite examinés.

Tous les facteurs définitivement retenus, ont leurs apparitions marquées dans le texte au fur et à mesure du choix. De sorte que, lorsqu'on examine le facteur suivant, on sait aux endroits où il apparait, si un autre facteur a déjà été choisi. Après l'achèvement de tous les choix, on procède au codage en parcourant le texte de gauche à droite. Grâce aux marquages effectués, on sait pour chaque position, lequel des trois cas se présente :

- s'il y a un facteur retenu ou non
- dans ce cas, s'il s'agit d'une première apparition ou d'une autre

Cela donne lieu à deux procédures de codage :

- les zones non répétées ou pour une première apparition d'un facteur, la sous-séquence correspondante est transcodée telle quelle (chaque base est associée à un code de 2 bits)
- une nième apparition d'un facteur est remplacée par le numéro de la première apparition de son facteur, dans l'ordre du parcours de gauche à droite.

Pour conclure cette description, nous soulignons trois faits très importants :

- le recodage des réapparitions de facteur est dynamique et n'entraine en aucun cas le stockage d'une table de correspondance des codes utilisés
- ce codage n'est pas loin d'être optimal, puisque l'on peut difficilement faire plus court qu'en numérotant les facteurs choisis
- bien sûr, toutes ces portions de séquence et ces codes de remplacement sont auto-délimités de manière à les distinguer les uns des autres.

En résumé, le schéma de nos algorithmes comporte quatre phases :

1. repérage de toutes les occurrences des régularités, grâce au Suffix Tree du texte ou de son complémentaire inversé biologique
2. classement de ces occurrences par ordre décroissant de longueur
3. examen selon cet ordre des occurrences et choix des plus avantageuses pour la compression suivant l'heuristique expliquée plus haut
4. codage en binaire du texte complet

3.3 Résultats comparatifs et commentaires

3.3.1 Comparaison des compresseurs Biocompress 1 et 2, Compgd+

Ici, nous présentons une comparaison des algorithmes de l'équipe de l'INRIA et des notres, sous forme d'un tableau de taux de compression. La compétition oppose 3 compresseurs : Biocompress, Biocompress2 pour l'INRIA et Compgd+ pour le LIFL.

L'architecture de Compgd suit le modèle exposé ci-dessus en exploitant conjointement les répétitions directes et les palindromes génétiques : il peut donc exploiter les mêmes opportunités de compression que Biocompress. Parfois, il s'avère avantageux de sur-compresser le résultat de Compgd par une méthode d'Huffman ou de Lempel-Ziv version 77 (nous nommons cet algorithme Compgd+).

Le tableau se divise en quatre colonnes, celles de gauche indique le nom et la taille de la séquence d'ADN compressée, tandis que les trois autres donnent les taux de compression en pourcent obtenus avec, respectivement de gauche à droite, Biocompress, Biocompress2 et Compgd+.

Si nous appelons, T_i la taille initiale de la séquence en nombre de bases et T_f la taille en octets du fichier final, alors le taux de compression se calcule ainsi :

$$1 - \frac{T_f * 4}{T_i}$$

Le facteur 4 permet de comparer sans tenir compte de la réduction obtenue en effectuant un simple transcodage de la séquence (comme c'est le cas dans Separe_Compresse).

Tableau de comparaison des taux de compression.

Séquence	Taille	Biocompress	Biocompress2	Compgd+
mipacga	100000	0,48	6,24	5,77
mpomtcg	186608	1,39	3,11	3
chntxx	155844	16,26	19,14	17,78
mpocpcg	121024	8,44	15,76	13,56
yscchrIII	315357	1,62	3,87	3
humghcsa	66495	32,75	34,63	19,66
humhbb	73323	3,22	6,16	5
humhdabcd	58864	3,86	6,15	5
humdystrop	38770	0,00	3,69	3
humhprtb	56737	2,27	4,67	4
vaccg	191737	7,91	11,93	8
hs5hcmvcg	229354	5,90	7,60	6,30

3.3.2 Commentaires

Bien que nos travaux aient débuté plus tard que ceux de l'INRIA, nous sommes arrivés rapidement à des resultats semblables en termes de taux de compression. Ceux-ci sont du même ordre de grandeur, en dépit d'une approche complétement différente. Ce résultat est logique dans la mesure où les régularités abordées sont identiques, relativement aisées à repérer et qu'il existe un recodage assez naturel sous forme de pointeur vers la première apparition.

Les taux de compression de Compgd+ s'intercalent entre ceux de Biocompress et ceux de son amélioration. Cependant, deux séquences échappent à cette remarque :

- **vaccg** se compresse de trois pourcents de moins avec Compgd+
- **humghcsa**, qui contient énormément de facteurs répétés (il s'agit du plus haut taux de compression), est bien mieux compressé à Paris qu'à Lille. Une cause probable est l'incapacité de Compgd+ à traiter les chevauchements et imbrications de ces nombreux facteurs. Nous sommes en train de remédier à ce problème de deux manières :
  1. par la constitution d'un algorithme réitérable et donc spécialement apte à traiter les chevauchements et les imbrications de facteurs
  2. par l'étude du codage des répétitions approximatives qui offre des perspectives allant au-delà des taux de compression actuels (travaux d'Olivier Delgrange).

Une remarque de grande importance : le tableau ne reflète pas du tout le taux de compression réalisable sur une séquence prise au hasard dans une banque. Celles testées ici sont les plus dociles à notre connaissance, d'autres se compriment peu ou pas du tout par nos méthodes, en raison de leur faible longueur et du manque de régularités. Les taux sont particulièrement bas par rapport au taux de compression des textes en langue naturelle.

3.4 Perspectives

Nous envisageons deux sortes de développements et d'améliorations : l'un au niveau de la performance de compression, l'autre au niveau de l'utilisabilité des compresseurs.

Le premier objectif vise à améliorer la capacité de compression des nos algorithmes, en se concentrant d'abord sur ceux qui n'exploitent qu'une seule régularité, mais qui dans ce cadre, ne laisseraient passer aucune opportunité de compression. Pour cela, nous pensons faire migrer nos algorithmes d'un codage hétérogène vers un codage homogène. Ceci en séparant le produit de la compression en deux :

- d'un côté, les codes de compression correspondants aux auto-délimitations et aux pointeurs de remplacement
- de l'autre, les portions de séquence non codées, concaténées selon leur ordre d'apparition de gauche à droite.

Ainsi pourrons-nous itérer l'algorithme sur la séquence restante et de ce fait exploiter les facteurs et palindromes imbriqués ou se chevauchant. Une autre approche consiste à varier l'heuristique de choix des régularités, actuellement basée sur la taille. Le choix peut tenir compte d'autres caractéristiques des régularités qui influent sur le taux de compression : tel que le nombre d'apparitions, le fait qu'elles apparaissent en tandem, etc... D'autres régularités étudiées à l'heure actuelle, semblent prometteuses, citons les répétitions avec mutations ou répétitions approximatives.

Sans attendre d'avoir un ensemble d'outils de compression complétement achevés, nous engageons une série de tests d'utilisation de ces compresseurs. Avec de la compression exacte, nous voulons examiner la classification de séquences induites par chacun de nos algorithmes, tandis que la compression avec perte d'information nous permet de tester des hypothèses de répartition des caractéristiques (hydrophobicité, appartenance à un motif structural pour les acides aminés, etc...).

Dans un cadre plus axé sur l'utilisabilité de nos algorithmes, nous avons commencé le développement sur station de travail SUN d'une plateforme d'expérimentation de compression. Elle permet de comparer :

- les taux de compression entre différentes séquences
- les taux de diverses combinaisons de compresseur sur une ou plusieurs séquences.

Cette interface graphique est écrite à partir de X-View.

D'autre part, nous voulons rendre notre compression plus "lisible", en trouvant un langage de description intermédiaire qui reflète les choix effectués pour la compression d'une séquence. On pourrait y voir où sont les facteurs compressés le long de la séquence, leur taille ainsi que leur enchaînement. Il s'agit donc d'expliquer la phase de compression à l'utilisateur, comme peut le faire un module d'explication avec le raisonnement produit par un système expert.

En combinant ces développements, nous désirons créer un outil agréable à manipuler et suffisament intelligent pour expliciter le résultat des expériences qu'il permet d'exécuter. Nous espérons qu'il trouvera par là, des débouchés et des utilisations entre les mains des biologistes.

Références

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison Wesley, Reading, MA, 1974.

[Apo85] Alberto Apostolico. The myriad virtues of subword tree. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Word.* Springer-Verlag, 1985.

[BCW90] Bell, Cleary, and Witten. *Text Compression.* Prentice Hall, 1990.

[Cre76] E. Mac Creight. A space-economical suffix tree construction algorithm. *Journal of the Association of Computing Machinery*, 23(2):262–272, April 1976.

[Cro] Maxime Crochemore. Off-line exact serial string searching. Rapport IGM 1992-1.

[Dan93a] Antoine Danchin. Genomes and Informatics. *Biochimie*, 75(5), 1993. Tous les articles, numéro spécial.

[Dan93b] Antoine Danchin. Le séquençage des petits génomes. *La Recherche*, 1993.

[Deh93] Patrice Dehais. *LIGM Database : Guide de l'utilisateur.* LIRMM, septembre 1993.

[Del94a] Jean-Paul Delahaye. *Information, complexité et hasard.* Hermès, Paris, 1994.

[Del94b] Jean-Paul Delahaye. Virus. *Pour la Science*, 1994.

[Fri91] W. H. Fridman. *L'immunologie.* Institut Merieux, 1991.

[Gro93] Roberto Grossi. *On Suffix Trees and Related Pattern Matching Problems.* PhD thesis, Università di Pisa, 1993.

[GT93a] Stéphane Grumbach and Fariza Tahi. Compression of DNA Sequences. In *Data Compression Conference*, 1993.

[GT93b] Stéphane Grumbach and Fariza Tahi. A New Challenge for Compression Algorithms: Genetic Sequences. *Journal of Information Processing and Management*, 1993.

[Lew92] Benjamin Lewin. *Gènes*, volume III. Médecine et Sciences Flammarion, 1992.

[LV93] Ming Li and Paul M.B. Vitanyi. *Introduction to Kolmogorov Complexity*. Springer-Verlag, 1993.

[Mil93] Aleksandar Milosavljeviĉ. Discovering Sequence Similarity by the Algorithmic Significance. In *Intelligent Systems for Molecular Biology*, pages 284–291. AAAI Press, 1993.

[MJ93a] Aleksandar Milosavljeviĉ and Jerzy Jurka. Discovering Simple DNA Sequences by the Algorithmic Significance Method. *CABIOS*, 9(4):407–411, 1993.

[MJ93b] Aleksandar Milosavljeviĉ and Jerzy Jurka. Discovery by Minimal Length Encoding: a Case Study in Molecular Evolution. *Machine Learning*, 12:69–87, 1993.

[Nel91] Mark Nelson. *The Data Compression Book*. M&T Publishing Inc., 1991.

[Riv93] Éric Rivals. Compression de séquences génétiques. Mémoire de DEA informatique, septembre 1993.

[SB92] Maxime Singer and Paul Berg. *Gènes et génomes*. Vigot, 1992.

[Wei73] P. Weiner. Linear pattern matching algorithms. In *Conf. Record of the 14th Annual Symposium on Swithcing and Automata Theory*, 1973.