LOSSLESS DIFFERENTIAL COMPRESSION ALGORITHM FOR GENOMIC SEQUENCE DATABASES

By

Heba Mahmoud Mohmmed Affify

A thesis Submitted to the

Faculty of Engineering at Cairo University

In Partial Fulfillment of the

Requirement for the Degree of

DOCTOR OF PHILOSOPHY

In

SYSTEMS AND BIOMEDICAL ENGINEERING

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2012

LOSSLESS DIFFERENTIAL COMPRESSION ALGORITHM FOR GENOMIC SEQUENCE DATABASES

By

Heba Mahmoud Mohmmed Affify

A thesis Submitted to the

Faculty of Engineering at Cairo University

In Partial Fulfillment of the

Requirement for the Degree of

DOCTOR OF PHILOSOPHY

In

SYSTEMS AND BIOMEDICAL ENGINEERING

Under the supervision of

Prof. Dr. Yasser M. Kadah
Biomedical Engineering department
Faculty of Engineering
Cairo University

Associate Prof. Dr.Manal A. Wahed
Biomedical Engineering department
Faculty of Engineering
Cairo University

Dr. Mohamed Islam
Biomedical Engineering department
Faculty of Engineering
Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2012

LOSSLESS DIFFERENTIAL COMPRESSION ALGORITHM FOR GENOMIC SEQUENCE DATABASES

By

Heba Mahmoud Mohmmed Affify

A thesis Submitted to the

Faculty of Engineering at Cairo University

In Partial Fulfillment of the

Requirement for the Degree of

DOCTOR OF PHILOSOPHY

In

SYSTEMS AND BIOMEDICAL ENGINEERING

Approved by

Examining Committee

Prof. Dr. Abdallah S. A. Mohamed **Examiner**
Systems and Biomedical Engineering Department, Cairo University

---

Prof. Dr. Mohamed E. El Adawy **Examiner**
Vice Dean for Education and Undergraduates, Helwan University

---

Prof. Dr. Yasser M. Kadah **Thesis Main Advisor**
Systems and Biomedical Engineering Department, Cairo University

---

Associate Prof. Dr. Manal A.Wahed **Thesis Advisor**
Systems and Biomedical Engineering Department, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2012

بسم الله الرحمن الرحيم

قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ أَنْتَ الْعَلِيمُ الْحَكِيمُ

سورة البقرة آية ٣٢

Engineer: Heba Mahmoud Mohmmed Affify
Date of Birth : 21/07/ 1979
Nationality : Egyptian
E-mail : hebaaffify@yahoo.com
Phone. : 01005757947
Address : 2 El Bahar Elasam St., Giza, Egypt.
Registration Date : 17 / 08 / 2008
Awarding Date : / /
Degree : Doctor of Philosophy
Department : Systems and Biomedical Engineering Department

Supervisors : Prof. Dr. Yasser M. Kadah
Associate Prof. Dr. Manal A. Wahed
Dr. Mohamed Islam

Examiners : Prof. Dr. Abdalla S. A. Mohamed
Prof. Dr. Mohamed E. El Adawy
Prof. Dr. Yasser M. Kadah
Associate Prof. Dr. Manal A. Wahed

Title of Thesis:

LOSSLESS DIFFERENTIAL COMPRESSION ALGORITHM FOR GENOMIC SEQUENCE DATABASES

Key Words: **genomic data**; **compression of DNA sequences**; **phylogenetic trees**

Summary:

The analysis of the huge amounts of genomic data that are continuously been generated puts a number of challenging problems to several research areas and, particularly, to the areas of computational biology and bioinformatics. One of these challenges is related with the problem of generating compression of DNA sequences in an efficient way, because these sequences might be as large as entire genomes (for example, the human genome is composed of about 3000 million bases). The complexity profile of a DNA sequence is a numerical sequence of the different length that indicates a measure of the predictability of each DNA base. Complexity profiles are important because they allow, for example, looking for repetitive structures inside a chromosome or across several chromosomes. These structures are often associated with regulatory functions of DNA. Moreover, the complexity profiles can also be used in finding evolutionary distances and, therefore, in the construction of phylogenetic trees

Table of Contents

List of Tables

List of Figures

List of Abbreviations

DNA	Deoxyribonucleic acid
A	Adenine - DNA Base
T	Thymine - DNA Base
C	Cytosine - DNA Base
G	Guanine - DNA Base
CTW	Context Tree Weighting
XM	Expert Model algorithm
BLAST	Basic Local Alignment Search Tools
BWT	Burrows-Wheeler Transform
PPM	Prediction by Partial Matching
MTF	Move-To-Front
Bpb	Bits per Byte
Bpc	Bit per Char
AMI	Average Mutual Information
BC	BioCompress2
GC	GenCompress
DC	DNACompress
DP	DNAPack
NML	Normalized Maximum Likelihood
CBMs	Compression –Based Distance Measures
CRS	Cambridge Reference Sequence
HTS	High-Throughput Sequencing
NCBI	National Center for Biotechnology Information
SNP	Single Nucleotide Polymorphisms

ACKNOWLEDGEMENT

First and foremost, thanks to God the most gracious. Then I would like to thank Prof. Dr.Yasser Moustafa Kadah; I am privileged to be under his supervision in my post graduate studies, who I am honored to be one of his students. I would like also to thank my teacher Assoc. Prof. Dr. Manal Abdel Wahed, whose input was most beneficial. She has not been only providing me an experienced guidance; but she has also been giving me support and always giving me the confidence and enthusiasm to go on. I would like also to thank Dr. Mohammad Islam, who has helped me during this work. They have given useful advice throughout my work on topics both related and unrelated to research. The many interesting discussions we had were both productive and enjoyable. They have shown me more patience than I deserved.

Finally, I owe some special thanks to my parents & my brothers Dr.Sherif and Mr.Wael, who offered the comments and criticism necessary to produce this work. They provided me with suitable atmosphere to work with moral support I needed. May God bless them always and grant them all their wishes. Also, thanks all of my family and friends. Without their continual love and support, this thesis would never have been written, even if they may not read this acknowledgement.

Abstract

The analysis of the huge amounts of genomic data that are continuously been generated puts a number of challenging problems to several research areas and, particularly, to the areas of computational biology and bioinformatics. One of these challenges is related with the problem of generating compression of DNA sequences in an efficient way, because these sequences might be as large as entire genomes (for example, the human genome is composed of about 3000 million bases). The complexity profile of a DNA sequence is a numerical sequence of the different length that indicates a measure of the predictability of each DNA base. Complexity profiles are important because they allow, for example, looking for repetitive structures inside a chromosome or across several chromosomes. These structures are often associated with regulatory functions of DNA. Moreover, the complexity profiles can also be used in finding evolutionary distances and, therefore, in the construction of phylogenetic trees.

Interestingly, difference compression schemes achieve the efficient compression of entire databases of sequences rather than previous specific compression algorithms. Difference compression schemes which compress entire sets of homologous sequences by encoding only the differences between a genomic sequences and a reference sequence, are developed in our work. One way to improve a data compression algorithm is by trying to find efficient modeling for difference sequences. We focus our study on large sets of sequences that belong to the same class. If two genomes are, e.g., more than 99% identical, it is much more efficient to store one genome as a variation from the other; in which case, only that 1% representing the variation needs to be stored. The decompression process is similar, but reverse when compared with compression to achieve lossless compression. The experimental results place difference sequence and difference locations as a new constituent of the state of the art in DNA sequences compression, achieving less space and less time, than its predecessors.

In this work, we have reviewed the previous art of compression and introduced new method for sequence difference encoding without depending on statistics of sequence set. The previous methods include alignment to find matching strings, entropy

estimation, and methods of encoding. To these existing methods, our proposed algorithm added some features such as a new modeling of differences to generate operation code for reducing the information in huge data sets, use entropy as inductor of reference sequence to improve compression, and updating technique to allow algorithm to add new data sets.

The differential compression algorithm was presented and its compression performance was shown to be optimal under some simplifying assumptions. We used this algorithm's compression performance for huge data by two solutions as a model to which we later compared two solutions to other compression algorithms. First solution treats the whole data set as one unit by using single standard references sequence. Second solution divides whole data set into N small sets to compress each small set individually by using selected variable references based on entropy. The algorithm was applied to three different data sets of genomic sequences, it achieved up to 195-fold compression rate corresponding to 99.4% space saving. This algorithm is successful for some applications such as reference selection, updating new data sets, and compression unknown data set from different species for small data. As a consequence, it does not scale to accept huge data sets in these applications. We experimentally show that these applications of this algorithm take a prohibitive amount of time on data sets as large as 1 MB and conclude that the algorithm's performance is unacceptable for some applications.

Using only previously known coding techniques, our simple algorithm provides a scalable differencing solution for genomic data sets of similar sequences of any size, and suitable speed for compression and decompression. Analysis of the algorithm showed it to perform well on proposed operation code that generates the difference sequence, and poorly on other operations. Also, the simplicity and flexibility of differential compression algorithm could make it an invaluable tool for DNA compression in clinical research. Both entropy estimation and operations code have an adequate combination of compression performance to be a generalized reference selection for many applications in small data sets. Consequently, there are some outstanding concerns about the scalability of this algorithm to choose the right reference under large inputs that need to be resolved by further experimentation.

Chapter 1

Introduction

1.1 Introduction

The theoretical background of compression is provided by information theory that created by Claude Shannon [1], who published fundamental papers on the topic in the late 1940s and early 1950s to show the idea of data compression. Compressing data, of necessity, involves understanding the way information is structured and, if possible, the mechanism by which the information was generated or is destined to be used. Thus, in order to compress speech it helps to know that the speech production process can be modeled by an autoregressive moving average filter excited by a signal with a periodic and a noise-like component. In order to compress images it helps to know that the sensing apparatus cannot discriminate high spatial frequencies. Where it is not possible to explicitly model the information source or sink, the best compression algorithms attempt to extract the way information is organized in the data in an adaptive fashion—in a sense learning the structure which allows for compression. The conceptual tools developed in

the field of source coding that have guided the development of data compression algorithms are thus useful instruments for the analysis of how information is organized in general, and in genomic systems in particular.

Life is strongly associated with organization and structure [2]. Living organisms can be viewed as agents communicating with their environment and storing information necessary for adaptation to the environment. This information storage, both in content and form, is shaped by the process of evolution and transmitted from one generation to the next via the DNA molecule. The complexity of life means that the amount of information contained in the DNA molecule for even simple unicellular organisms is very large and requires efficient storage. Efficient storage would dictate the removal of all redundancy from the data being stored. However, the complexity of life also means that the information transmission has to be accurate as errors would have a disastrous effect on the survival of the organism. This would argue against a decrease in redundancy to allow for some level of robustness in the preservation of information. Another twist, not encountered by most other sources of information, is that the organization of the information in the DNA molecule affects the level of production of the protein molecules that are the machinery of the cell, and can often affect the order in which they are produced. To accommodate all these requirements (and more) the level of organization of information in genomic molecules has to be highly complex.

From the late 1980s onward, the term “bioinformatics” mostly has been used to refer to computational methods for comparative analysis of genomic data. Bioinformatics is the science of using and developing computational tools and algorithms to help solve different biological problems [3]. These problems include similarity searches of unknown DNA / protein sequences, 3D protein structure prediction, protein function prediction and compression of huge data. The extra information obtained from bioinformatics analysis of unknown data can help researchers to design better, or more precise experiments in solving their problems. In bioinformatics, we use existing biological databanks to help

analyze any raw data from various experiments. Therefore, biological database places a vital role in bioinformatics.

The increasing volume of biological data collected in recent years has prompted increasing demand for bioinformatics tools for genomic and proteomic data analysis and compression [4]. Web based bioinformatics application platforms have become one of the popular tools for biological data analysis among the bioscience community. However, these application platforms utilize different stand-alone bioinformatics applications and they use different data sources in different formats.

DNA forms the basis of natural evolution, allowing genetic information to be passed from parents to offsprings. Proteins, formed by the translation of certain parts of DNA, are necessary for the structure and function of cellular life. The amount of DNA and protein being extracted from organisms and sequenced is increasing exponentially [3]. This yields two problems: storage and comprehension.

The problem of storage is practical, useful to study and become dire at present and also in the future. Currently, DNA and protein sequences are usually stored in text format using 8 bits per character, even when a trivial encoding of the data could reduce the length to a half or a quarter of its current size. Indeed, the sequences often consume more than a mere eight bits per character when white space and annotations are added to aid the human reader. Modern storage devices are more than capable of storing vast amounts of sequence data even in this inefficient format.

Despite the prevalence of broadband network connections — especially between universities and research centers — there still exists a need for compact representation of data to speed up transmission. Transferring a single sequence that is millions of characters long may take ten to fifteen minutes over a dial-up connection [3]. In this area the compression of biological sequences may be useful.

Understanding biological sequences has wide applications, from the synthesis of medicines to genetic screening and engineering. The structure of a sequence is important knowledge for its comprehension. If a sequence has a particular property that is shared by another sequence, it is possible that they are related in some way, or that knowledge that applies to one may also be useful for the other. Compression can help to show both the structure of a sequence and how it is related to other sequences.

Recently, data compression can be viewed as a special case of data differencing [5]. Data differencing consists of producing a difference given a source and a target, with producing a target given a source and a difference, while data compression consists of producing a compressed file given a target, and decompression consists of producing a target given only a compressed file. Thus, one can consider data compression as data differencing with empty source data, the compressed file corresponding to a "difference from nothing". This is the same as considering absolute entropy (corresponding to data compression) as a special case of relative entropy (corresponding to data differencing) with no initial data.

Since 1995s onward, the term *differential* compression may used to refer to data differencing. Generally, differencing algorithms achieve compression by finding common sequences between two versions of the same data that can be encoded using a copy reference. A differencing algorithm is an algorithm that finds and outputs the changes made between two versions of the same data by locating common sequences to be copied and unique sequences to be added explicitly. A difference file (Δ) is the encoding of the output of a differencing algorithm. An algorithm that creates a difference file takes as input two versions of a file, a base file and a version file to be encoded, and outputs a difference file representing the incremental changes made between versions.

$$F_{base} + F_{Version} \rightarrow \Delta_{(base, Version)}$$

Reconstruction, the inverse operation, requires the base file and a difference file to rebuild a version

$$F_{base} + \Delta_{(base, Version)} \rightarrow F_{Version}$$

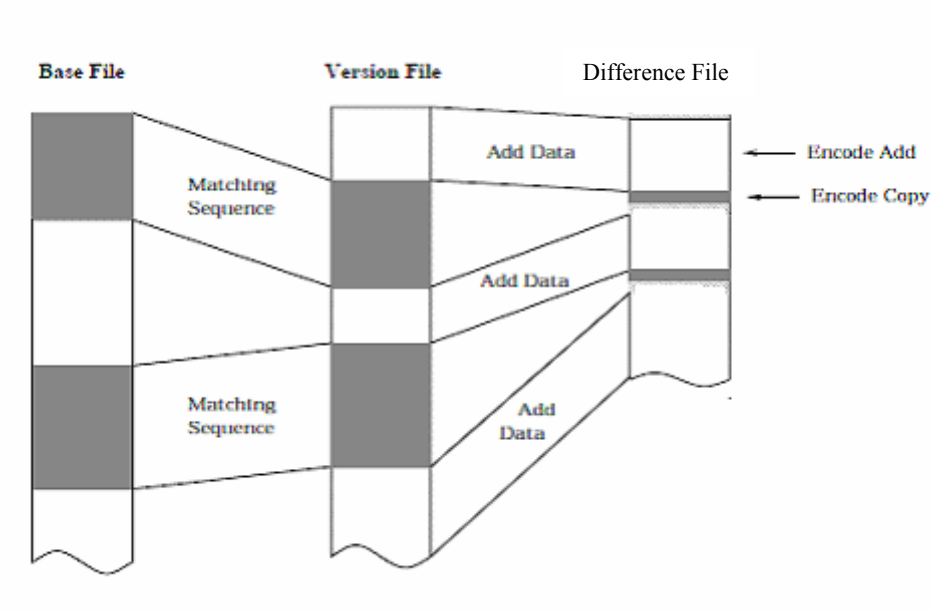


Figure 1.1: An example of encoding difference files

One encoding of a difference file consists of a linear array of editing directives (Figure 1.1). These directives are copy commands, references to a location in a base file where the same data exists, and add commands, instructions to add data into the version file followed by the data to be added. There are other representations including those that represent difference files as linked data structures such as trees or lists, and one based upon matrix algebra. In any representation scheme, a differencing algorithm must have found operations such as copies and adds to be encoded. In distributed file system backup and restore, differential compression would reduce the time to perform file system backup, and decrease network traffic during backup and restore.

Differential compression may be a mean to explore advanced technique for storage of huge genomic data. We found that differential compression of the genomic sequences could open new frontiers in how to calculate the differences and difference locations according to operation codes (add, insert, copy, or delete) and quickly identifying unknown sequence related to set of sequences based on selection of reference sequence. Also, we found that high similarity between sequences provides small difference sequences that improve compression and decompression process by reducing the amount of data transferred.

1.2 Motivation

The increasing productivity of molecular biology in relation to the accumulation of biological information, supported in modern sequencing and assisted by bioinformatics, provides a wide scientific community a number of genomes available in public databases expanding exponentially. In a few years, the volume of biological information available exceed the Tera-bases, and assuming that the sequencing of species give rise to the sequence of individuals, then this value will appear to us ridiculous. Thus, the efficient compression of this type of information is an urgent need to optimize storage and communication [6]. Initial step must be analyzed and measured the complexity of the information contained in genome sequences [7].

From the strictly mathematical point of view, compression implies understanding and agreement of the compressed data [8]. Knowledge of the properties of the information that want to compress is greater than the likelihood of success in obtaining compression [9]. Paradoxically, the knowledge of the properties of information can also result from the application of compression techniques whose initial intention was to capture the regularities and redundancies alike, which ultimately expose confirming or discovering the properties of information [10].

Thus, the primary motivation of this work is to develop knowledge about algorithms that deal with the biological information, especially for database of the similar nucleotide sequences to help in differential compression process.

The post-genomic era brings new challenges to carry out a competent analysis of biological data. This function basically meets the functional genomics adjuvant by bioinformatics. In essence, the task in bioinformatics researchers are faced with is to pass the knowledge of biological information [11]. The research and analysis of regularities of DNA, taken as exact or approximate patterns, is crucial in most applications-oriented functional genomics bioinformatics. The prediction of genes, the discovery of motifs, sequence alignment, compression of sequences or the study of phylogenetic correlations have in its core research and analysis of patterns. In all these applications, the

performance of search algorithms, in terms of efficiency and sensitivity, is determining the quality of the solutions that provide.

In this work we tried to situate the research in terms of not just using bioinformatics algorithms, but also using data independent of the biology of biological origin. Thus, the algorithms developed using the knowledge of biological sequences that depend on the statistics of DNA to achieve the best compression of database of sequences.

The compression of DNA sequences remains a challenging problem, with profound implications in biology and with important technological impact when the use of genomic data will become a daily practice in health and medicine. As such, it will certainly be investigated further due to several reasons:

1- Benefits when storing and transmitting the genome files

2- Possibilities for comparison of entire genomes

3- Discovering statistically significant relationships among various subsequences.

We used the lossless compression (described fully in Chapter 3) for DNA because it is undoubtedly the most important data structures in string processing. This is particularly true if the sequences to be analyzed are very large and do not change. An example of prime importance from the field of bioinformatics is genome analysis, where the sequences under consideration are whole genomes (the human genome, for example, contains more than $3*10^9$ base pairs). The size of the DNA varies greatly in different organisms as shown in the Table 1.1.

In this study, Compression is done without the loss of information by exploiting the redundancy within the similar sequences that based on information theory [1]. Textual data compression, and the associated techniques coming from information theory, are often perceived as being of interest for data communication and storage. However, they are also deeply related to classification and data mining and analysis. In recent years, a substantial effort has been made for the application of textual data

compression techniques to various computational biology tasks, ranging from storage and indexing of large data sets to comparison and reverse engineering of biological networks

Table 1.1: Comparison of different genome sizes

Organism	Genome Size (Bases)	Estimated Genes
Human (*Homo sapiens*)	3 billion	30,000
Laboratory mouse (*M. musculus*)	2.6 billion	30,000
Mustard weed (*A. thaliana*)	100 million	25,000
Roundworm (*C. elegans*)	97 million	19,000
Fruit fly (*D. melanogaster*)	137 million	13,000
Yeast (*S. cerevisiae*)	12.1 million	6,000
Bacterium (*E. coli*)	4.6 million	3,200
Human immunodeficiency virus (HIV)	9700	9

DNA sequencing has had a major impact on life sciences since the wide scale adoption of the Sanger sequencing method [12], With the advent of array-based pyrosequencing in 2005 [13], followed rapidly by new sequencing-by-synthesis and sequencing-by-ligation techniques, there has been an exponential increase in the generation of DNA sequence data [14]. Critically, these newer technologies are developing at a much greater rate than was seen for the older technologies, with the current doubling time for DNA sequence output—as measured by output per unit cost—of around 5 months [14]. In practical terms this has provided a 1,000-fold drop in sequencing costs since 1990 and has made economically possible an increasing number of large data projects. These include the 1,000 Genomes project [15], the International Cancer Genome Project [16] and a variety of projects that use DNA sequences as an assay platform, such as the ENCODE project [17].

Compression is broadly classified into two major topics, although related research or topics. One is intra-species analysis, where the major goal is to explore the information conveyed by the complexity profiles for locating and classifying repetitive structures occurring inside a chromosome or across several chromosomes of the same species. The other topic is inter-species analysis, where the main goal is to use the complexity information for computing evolutionary distances among the species. In both topics, compression process consists of two phases: modeling and coding [18]. Coding phase is used to produce bit sequences such as Huffman coding [19] and Arithmetic coding [20]. Therefore, modeling will play a key role because the reason of the strong connection of compression is modeling: *Good model <=>Good compression.*

There are some features of a good model at the following steps:

1-Based on biological knowledge

2-Simple, few parameters

3-Can give per-symbol information content

4- Efficient algorithm that achieve good compression.

In this work, we tried to use good statistical model that depend on the relationships between two sequences to provide op-code table that refer to the difference sequence to improve the compression method. Also, a good choice for a reference sequence depends on good model.

Generally, compression is useful for digital storage (smaller files means less storage cost) and transmission (less time/bandwidth needed). So, compression is a great tool for genome comparison and for studying various properties of genomes. DNA sequences, which encode life, should be compressible. It is well known that DNA sequences in higher eukaryotes contain many tandem repeats, and essentials genes (like rRNAs) have many copies. It is also proved that genes duplicate themselves sometimes for evolutionary purposes. All these facts conclude that DNA sequences should be compressible. However, compression of DNA sequences is not an easy task [21-23].

1.3 Problem Definition

In the last few years, DNA evidence has started to play a big part in many nations criminal justice systems. It has been used to prove that suspects were involved in crimes and to free people who were wrongly convicted. Several countries, including United States and Britain, have built elaborate databases with hundreds of thousands of unique individual profiles [22]. Currently many countries are establishing DNA database from individual with the history of violent and crimes. Those databases would allow the identification of suspects by simply cross checking the DNA profile of the evidence with those stored in the database. The Combined DNA Identification System (CODIS) connects local states and federal law enforcement agency data banks across the country USA (approximately 2 million profiles). As of march 2005 CODIS has produced 21000 criminal identifications and it has assisted in over 23000 investigations.

Today, increasing genome sequence data of organisms lead DNA database two or three times bigger annually. Thus, it becomes very hard to download and maintain such data in personal local system [23]. The size of current biological databases is rapidly increasing due to continuous sequencing efforts. Some common databases need more than 160GB of disk space (e.g. DNA databank of Japan (DDBJ) [23] and furthermore, additional disk space is needed for the index files of the different retrieval methods (e.g. BLAST [4]). Thus, there is a lot of disk space needed for redundant data. One could now think about efficient compression algorithms to solve this problem.

The problem with DNA based identification system is that it cannot be used for online identification of a person like iris based identification system or thumb print based identification system. But, DNA database of a person may be useful for further criminal investigation. Thus, we may either need to retrieve the DNA data from server for further processing and investigations or we may need to transfer the DNA data from one location to another location for further criminal investigations.

A single person's complete DNA structure may take more than 100 gigabytes of memory. Therefore if we transfer DNA database from one location to another location for further criminal investigations or for any other purpose then the transfer time will be very high. If we download the data from the server for further research work then its download time will be very high. Thus, we need an efficient lossless compression technique to compress the DNA sequence before sending it to another location so that after receiving and decompressing the data at destination end the exactly same DNA sequence should be obtained. Standard compression algorithms (see Table 1.2) are not able to compress DNA sequences because they do not consider special characteristics of DNA sequences (e.g., DNA sequences contain several approximate repeats and complimentary palindromes). The standard compression software such as "compress", "gzip", "bzip2", "winzip" increase the DNA genome file size instead of compressing it. Most of the existing software tools were worked well for English text compression (Bell *et al*. 1990[24]) but not for DNA genomes.

DNA sequences contain only four bases (A: Adenine, T: Thymine, C: Cytosine, G: Guanine). Thus, each base (symbol) can be represented by two bits. However, the standard text compression algorithms such as gzip that based on LZ77 algorithm [25] or compact (Huffman technique) cannot compress DNA sequences; the compressed data need more space than the actual data. On the other hand**,** CTW [26] and Arithmetic coding [20] can compress DNA sequences less than two bits per symbol.

There are some reasons pointed out; this software is designed mainly for English text compression, while the regularities in DNA sequences are much subtler. Generally the windows of the methods based on dictionary have a fixed width of small size. The use of small windows is efficient on text whose redundancy is local. However, in the case of DNA sequence, redundancies may occur at very long distances and factors can be very long. Huffman's code [19] also fails badly on DNA sequences both in the static and adaptive model, because there are only four kind symbols in DNA sequences and the probabilities of occurrence of the symbols are not very different. Obviously, if strings of four characters are encoded on one byte, before compression by using adaptive Huffman,

the average compression ratio will be (1.960 bit per base). Concerning compression ratio, PPM [27] is one of the best compression algorithms in practice. However it cannot compress DNA sequences less than two bits per symbol either.

Table 1.2: Average compression ratio for Standard compression algorithms

Standard compression algorithms	The average compression ratio (Bit/base)
compact (Huffman coding)	2.380
bzip2 (Burrows Wheeler)	2.064
compress (LZW)	2.185
gzip (LZ 77)	2.271
Arithmetic coding	1.952
Context Tree Weighting Method	1.883

Compression of genomic sequences can be divided into two categories:

- Specific compression algorithms developed for efficiently compressing sequence data for the sake of reduced resource consumption (disk space or network usage);
- Investigations of the usefulness of compressibility as a measure of information content, for the purpose of making inferences about sequences (such as the relatedness of two sequences).

In this work, we exploit the differential compression algorithm based on relatedness of similarity sequences, which compresses one sequence relative to another sequence to achieve less disk space. It means that combination of two pervious categories of compression is available in our algorithm.

Specific compression algorithms have been proposed for DNA compression by using particular characteristics such as exact or approximate repeats measures within a single DNA sequence that are based on relations between subsequences only. On

average, the best available compressors such as XM [28], conceived accordingly DNA data specificities, reach 1.7 bits/base, which corresponds to a compression rate of only 15%, and this value is demonstrative of the inherent difficulty. Recently, P. Rajarajeswari, and A. Apparao [29] presented a new compression algorithm named "DNABIT Compress" whose compression rate is below 1.56 bits per base (for Best case) even for larger genome (nearly 2,00,000 characters). Also, the problem is that the existing DNA compression algorithms are too time demanding. For example, the best performing DNA compression techniques, such as XM [28], could take hours for compressing a single human chromosome. Compression gains afforded by these algorithms are ultimately not sufficient to justify their adoption for large databases. While much research has been done on compressing individual DNA sequences, surprisingly little has focused on the compression of entire databases.

Therefore, we need to develop one of these compression algorithms by using special structures of difference sequences to introduce two solutions for compression huge genomic data sets. Also we need to find a simple and fast algorithm to use as a guide to quantify reference sequence. By using reference sequence for each organism, we can use this reference sequence to considerably compress the remaining DNA sequences.

During our ongoing work on the compression of entire databases, we have obtained several encouraging and important results related to DNA modeling based on difference compression. When a sequence, in a set of similar sequences, is differentially compressed, another sequence should be labelled as reference. Instead of storing the original sequences, it is sufficient, and more efficient, to store differences between each sequence and the reference. An important question that needs to be addressed is which sequence should be used as the reference for the sequences to be compressed? Another important question is in which the order should the different sequences be compressed? The second issue is how to update the differentially compressed set of similar sequences, when a new similar sequence is available. The third issue is compression of database from two different species that based on evolution and ordering of species in phylogenetic tree.

1.4 Objectives

Modern biological science produces vast amounts of DNA sequence data. This is fuelling the need for efficient algorithms for sequence compression and analysis. Compression of DNA is interesting for both practical reasons (such as reduced storage and transmission cost) and functional reasons (such as inferring structure and function from compression models). This thesis explores a new efficient algorithm that uses differential compression. It used to estimate the reference of sequence data, update the differentially compressed set, and compression of data set from two different species. Our approach depends on exploitation of high similarity between genomic sequences from the same class to develop difference sequence and renewal entropy estimation in order to improve compression of data set.

This work seeks to develop knowledge of the operation codes through genomic analysis-oriented differential compression of information, this goal should result in compact alternative coding that will make more efficient storage of genomic information and communication.

The following are the core objectives of the proposed algorithm:

- The algorithm should be flexible, in that it should allow any genomic data set to access via the available GenBank.
- The algorithm must achieve lossless compression, where genomic sequences carry important genetic information for clinical research. Lossless is used in cases where it is important that the original and the decompressed data be identical, or where deviations from the original data could be deleterious.
- The most important contribution of this work is the idea of statistical modeling for differences and difference locations. Our statistical modeling is op code table that used to extract information about any redundancy that exists in the data and describe the difference positions in the form of a model. A good model of sequences would achieve excellent compression. Therefore, understanding the input data can lead to efficient ways of representing the information and hence data compression.

- Speed of the algorithm is also an important issue. Because algorithm is expected to be fast in all of its operations.
- Extensibility should be a key feature of this algorithm, as the number of genomic data is growing. The algorithm should be able to add new genomic data without having effect on other components of the algorithm.
- The algorithm should include a collection of essential parameters for evaluating the process of compression such as entropy estimation and size of sequences that depend on the available memory space.

This thesis is divided into two parts for encoding huge data sets. In first part, we try to find a good solution in order to develop difference sequences and compress huge data in less time and less space. In second part, we address the issues of reference sequence selection, when differentially compressing a small set of similar sequences which are combined to from huge data, the order in what the set should be differentially compressed and update the set with new similar sequences whose differences sequences do not obey the same statistical model as that of difference sequences of the original set. Both parts present a new differential compression algorithm that is based on production of difference sequences according to op-code table in order to optimize the compression of homologous sequences in data set. Therefore, the stored data are composed of reference sequence, the set of differences, and differences locations, instead of storing each sequence individually.

In health and medical fields, bioinformatics enables advances in areas such as drug discovery, diagnostics, and disease management [30]. Therefore, compression is important application of bioinformatics that help in medical fields. The goal of compression and decompression of genomic sequence is easy used of huge data to assist researches in their decision making, and avoid errors in diagnosis and the selection of treatment.

1.5 Thesis Overview

The organization of this thesis is as follows: Chapter 1 titled “Introduction” and gives a short account of the problem definition, thesis objective and its organization.

Chapter 2 titled “Background: The Genomic Information” and introduces the background information on various aspects of bioinformatics and how biological data are being managed and used for analysis and compression. It focuses on the structure of DNA molecule and the problem of current ways of practicing bioinformatics that based on the given background information.

Chapter 3 titled “Methods of Analysis and Compression of Genomic Sequence”. It presents for a start the complete workflow for DNA compression, and discusses the coding component of compressing algorithms. It also introduces the literature review of compression of genomic sequences, discusses the problems of them, and analyzing the efficient algorithms in greater detail which serve for differential compression. It describes the different approaches and methodologies for data compression to select, with reasons, the most suitable for genomic information that based on sequences of symbols derived from the quaternary alphabet $\Sigma = \{a,c,t,g\}$.

Chapter 4 titled “Algorithm Design and Implementation” and it describes our proposed algorithm / architecture, which attempts to solve the problem for encoding huge data by two techniques. It gives a detailed walk through of each technique and the various issues / problems incurred during the development process.

Chapter 5 titled “Results and Discussion”. It describes experimental results of data extraction by applying the differential compression algorithm to different databases and compares the compression ratio and time execution for genomic data sets.

Finally, chapter 6 titled “Concluding Remarks”. It presents a brief about the work carried out in this dissertation together with our conclusion and perspective of the future work that could be achieved in this area of research or even related ones.

Chapter 2

Background: The Genomic Information

It is important to fully understanding the state of the art bioinformatics, especially in the extensions of this more involved in the areas covered in this work. The characterization of the language of DNA, particularly with regard to the specificities of repetitive elements that comprise a large proportion of the genome sequences, is also a central theme of this chapter, status justified by their importance as a resource of compression.

2.1 Introduction

In the past, computational biology has opened the way for the current bioinformatics. With the advent of the Internet, a myriad of bioinformatics applications are available online for researchers, ranging from the powerful tools of exploratory genomic databases to the representations of 3D structures of proteins. The daily access to these services prove the immense interest of these applications as well as the bustling research activity in these areas [3]. Bioinformatics is an area of highly multi-disciplinary science and so are, for now, the actors, mostly biologists who made the subsequent learning of computer parts and computer or with the opposite route, if the former took the first steps , the latter are enhancing of developments for bioinformatics [31].

Currently, bioinformatics is concerned mainly the organization, analysis, simulation and knowledge extraction of data produced by recent advances in molecular biology and genetics. As bioinformatics, in a broad sense, supported multi-disciplinary science in biology, mathematics, biostatistics and computing, which seeks to solve the problems based on sequences (DNA, RNA and proteins), a booming business, now that are the major applications of bioinformatics approaches.

With the proliferation of biological databases and the biomedical research using bioinformatics to rival for leadership in the occupation of existing computing resources, we can identify some of the key points where the bioinformatics demand more intensity to establish itself. They are:

- Assembly of DNA fragments;
- Search for genes;
- Analysis and comparison of biological sequences;
- Identification of motifs and patterns;
- Search for homologies or similarities;
- Phylogenetic inference;
- Simulation of genetic processes;
- Inference of protein function and structure;
- Design of proteins;
- Determination of local connection with protein molecules of drugs;
- Etc.

Bioinformatics provides algorithms and applications that address, each time with greater efficiency, the problems listed above. Its evolution is crucial because the volume and complexity of bioinformatics is booming and demand for, for example, find a cure for a particular type of cancer, will necessarily by advances in bioinformatics.

The parallels between the way a computer program works and the biological processes of encoding gene and replication of information has already been established by many of the theorists who addressed the bioinformatics [32, 33]. In fact, the principles established by other theoreticians of computing approach is very considerably the

processes that govern their lives inside the cells. If the DNA is a repository of key information as with the primary hard drive of a computer, the RNA may equate to the RAM through which data is available to process more directly, such as in biological processes maturation of mRNA, preliminary to the formation of proteins [34].

2.2 Bioinformatics

Bioinformatics is the application of computer science and information technology to the field of biology and medicine. Bioinformatics deals with algorithms, databases and information systems, web technologies, artificial intelligence and soft computing, information and computation theory, software engineering, data mining, image processing, modeling and simulation, signal processing, discrete mathematics, control and system theory, circuit theory, and statistics, for generating new knowledge of biology and medicine, and improving & discovering new models of computation (e.g. DNA computing, neural computing, evolutionary computing, swarm-computing, cellular-computing). Java, XML, Perl, C, C++, Python, R, SQL and MatLab are the some of the more prominent software technologies used in this field.

Bioinformatics was applied in the creation and maintenance of a database to store biological information at the beginning of the "genomic revolution", such as nucleotide and amino acid sequences. Development of this type of database involved not only design issues but the development of complex interfaces whereby researchers could both access existing data as well as submit new or revised data.

In order to study how normal cellular activities are altered in different disease states, the biological data must be combined to form a comprehensive picture of these activities. Therefore, the field of bioinformatics has evolved such that the most pressing task now involves the analysis and interpretation of various types of data, including nucleotide and amino acid sequences, protein domains, and protein structures [35]. The actual process of analyzing and interpreting data is referred to as computational biology. Important sub-disciplines within bioinformatics and computational biology include:

- The development and implementation of tools that enable efficient access to, and use and management of, various types of information.
- The development of new algorithms (mathematical formulas) and statistics with which to assess relationships among members of large data sets, such as methods to locate a gene within a sequence, predict protein structure and/or function, and cluster protein sequences into families of related sequences.

The primary goal of bioinformatics is to increase the understanding of biological processes. What sets it apart from other approaches, however, is its focus on developing and applying computationally intensive techniques (e.g., pattern recognition, data mining, machine learning algorithms, and visualization) to achieve this goal. Major research efforts in the field include sequence alignment [4], gene finding [36], genome assembly [37] , drug design [38], drug discovery [39], protein structure alignment [40], protein structure prediction [41], prediction of gene expression [42] and protein–protein interactions [43], genome-wide association studies [44] and the modeling of evolution [45].

The term bioinformatics was coined by Paulien Hogeweg and Ben Hesper in 1978 for the study of informatic processes in biotic systems [46,47]. Its primary use since at least the late 1980s has been in genomics and genetics, particularly in those areas of genomics involving large-scale DNA sequencing.

Bioinformatics now entails the creation and advancement of databases, algorithms, computational and statistical techniques and theory to solve formal and practical problems arising from the management and analysis of biological data. Over the past few decades rapid developments in genomic and other molecular research technologies and developments in information technologies have combined to produce a tremendous amount of information related to molecular biology. It is the name given to these mathematical and computing approaches used to glean understanding of biological processes.

Common activities in bioinformatics include mapping and analyzing DNA and protein sequences, aligning different DNA and protein sequences to compare them and creating and viewing 3-D models of protein structures. There are two fundamental ways of modelling a biological system (e.g. living cell) both coming under Bioinformatic approaches.

1-Static

- Sequences – Proteins, Nucleic acids and Peptides
- Structures – Proteins, Nucleic acids, Ligands (including metabolites and drugs) and Peptides
- Interaction data among the above entities including microarray data and Networks of proteins, metabolites

2-Dynamic

- Systems Biology comes under this category including reaction fluxes and variable concentrations of metabolites
- Multi-Agent Based modelling approaches capturing cellular events such as signalling, transcription and reaction dynamics

2.3 Biological Databases and Their Roles in Bio-Analysis

Twenty-first century biology will be a data-intensive enterprise. Laboratory data will continue to underpin biology's tradition of being empirical and descriptive. In addition, they will provide confirming or disconfirming evidence for the various theories and models of biological phenomena that researchers build. Also, because 21st century biology will be a collective effort, it is critical that data be widely shareable and interoperable among diverse laboratories and computer systems. This section describes the nature of biological data and the requirements that scientists place on data so that they are useful.

2.3.1 Data Heterogeneity

An immense challenge—one of the most central facing 21st century biology—is that of managing the variety and complexity of data types, the hierarchy of biology, and the inevitable need to acquire data by a wide variety of modalities. Biological data come in many types. For instance, biological data may consist of the following [48].

- *Sequences:*

  Sequence data, such as those associated with the DNA of various species, have grown enormously with the development of automated sequencing technology. In addition to the human genome, a variety of other genomes have been collected, covering organisms including bacteria, yeast, chicken, fruit flies, and mice[1]. Other projects seek to characterize the genomes of all of the organisms living in a given ecosystem even without knowing all of them beforehand [49]. Sequence data generally consist of text strings indicating appropriate bases, but when there are gaps in sequence data, gap lengths (or bounds on gap lengths) must be specified as well.

- *Graphs:*

  Biological data indicating relationships can be captured as graphs, as in the cases of pathway data (e.g., metabolic pathways, signaling pathways, gene regulatory networks), genetic maps, and structured taxonomies. Even laboratory processes can

[1] http://www.genome.gov/11006946.

be represented as workflow process model graphs and can be used to support formal representation for use in laboratory information management systems.

- *High-dimensional data:*

Because systems biology is highly dependent on comparing the behavior of various biological units, data points that might be associated with the behavior of an individual unit must be collected for thousands or tens of thousands of comparable units. For example, gene expression experiments can compare expression profiles of tens of thousands of genes, and since researchers are interested in how expression profiles vary as a function of different experimental conditions (perhaps hundreds or thousands of such conditions), what was one data point associated with the expression of one gene under one set of conditions now becomes 10^6 to 10^7 data points to be analyzed.

- *Geometric information:*

Because a great deal of biological function depends on relative shape (e.g., the "docking" behavior of molecules at a potential binding site depends on the three-dimensional configuration of the molecule and the site), molecular structure data are very important. Graphs are one way of representing three-dimensional structure (e.g., of proteins), but ball-and-stick models of protein backbones provide a more intuitive representation.

- *Scalar and vector fields:*

Scalar and vector field data are relevant to natural phenomena that vary continuously in space and time. In biology, scalar and vector field properties are associated with chemical concentration and electric charge across the volume of a cell, current fluxes across the surface of a cell or through its volume, and chemical fluxes across cell membranes, as well as data regarding charge, hydrophobic, and other chemical properties that can be specified over the surface or within the volume of a molecule or a complex.

- *Patterns:*

Within the genome are patterns that characterize biologically interesting entities. For example, the genome contains patterns associated with genes (i.e., sequences of particular genes) and with regulatory sequences (that determine the extent of a

particular gene's expression). Proteins are characterized by particular genomic sequences. Patterns of sequence data can be represented as regular expressions, hidden Markov models (HMMs) [50], stochastic context-free grammars (for RNA sequences), or other types of grammars. Patterns are also interesting in the exploration of protein structure data, microarray data, pathway data, proteomics data, and metabolomics data.

- *Constraints:*

Consistency within a database is critical if the data are to be trustworthy, and biological databases are no exception. For example, individual chemical reactions in a biological pathway must locally satisfy the conservation of mass for each element involved. Reaction cycles in thermodynamic databases must satisfy global energy conservation constraints. Other examples of non local constraints include the prohibition of cycles in overlap graphs of DNA sequence reads for linear chromosomes or in the directed graphs of conceptual or biological taxonomies.

- *Images:*

Imagery, both natural and artificial, is an important part of biological research. Electron and optical microscopes are used to probe cellular and organ function. Radiographic images are used to highlight internal structure within organisms. Fluorescence is used to identify the expressions of genes. Cartoons are often used to simplify and represent complex phenomena. Animations and movies are used to depict the operation of biological mechanisms over time and to provide insight and intuitive understanding that far exceeds what is available from textual descriptions or formal mathematical representations.

- *Spatial information:*

Real biological entities, from cells to ecosystems, are not spatially homogeneous, and a great deal of interesting science can be found in understanding how one spatial region is different from another. Thus, spatial relationships must be captured in machine-readable form, and other biologically significant data must be overlaid on top of these relationships.

- *Models:*

Computational models must be compared and evaluated. As the number of computational models grows, machine-readable data types that describe computational models—both the form and the parameters of the model—are necessary to facilitate comparison among models.

- *Prose:*

The biological literature itself can be regarded as data to be exploited to find relationships that would otherwise go undiscovered. Biological prose is the basis for annotations, which can be regarded as a form of metadata. Annotations are critical for researchers seeking to assign meaning to biological data.

- *Declarative knowledge such as hypotheses and evidence:*

As the complexity of various biological systems is unraveled, machine-readable representations of analytic and theoretical results as well as the underlying inferential chains that lead to various hypotheses will be necessary if relationships are to be uncovered in this enormous body of knowledge.

In many instances, data on some biological entity are associated with many of these types: for example, a protein might have associated with it two-dimensional images, three-dimensional structures, one-dimensional sequences, annotations of these data structures, and so on. All of these different types of data are needed to integrate diverse witnesses of cellular behavior into a predictive model of cellular and organism function. Each data source, from high-throughput microarray studies to mass spectroscopy, has characteristic sources of noise and limited visibility into cellular function. By combining multiple witnesses, researchers can bring biological mechanisms into focus, creating models with more coverage that are far more reliable than models created from one source of data alone. Thus, data of diverse types including mRNA expression, observations of in vivo protein-DNA binding, protein-protein interactions [43], abundance and sub cellular localization of small molecules that regulate protein function (e.g., second messengers), posttranslational modifications, and so on will be required under a wide variety of conditions and in varying genetic backgrounds. In addition, DNA sequence from diverse species will be essential to identify conserved portions of the genome that carry meaning.

2.3.2 Data Organizations

Biological data cover protein and DNA sequences, special coding regions, 3D structures of protein sequences, evolutionary relationships between the organisms.

There are two major organizations that maintain both nucleotide and protein sequence databank. They are the European Molecular Biology Laboratory (EMBL)[2] from Germany and the National Center for Biotechnology Information (NCBI)[3], which is located in the United States. Both parties maintain their own sets of sequence databanks, and they are often being referred to as sequence data banks, such as the SWISS-PROT [51] and TrEMBL [52] protein sequence databank from EMBL, the GenBank [53] nucleotide sequence databank from NCBI and the Protein Data Bank (PDB) [54] from the Brookhaven National Laboratory.

We should clarify the difference between a database and a databank. We often refer to raw sequence data in plain text files as databanks. Databases refer to XML databases or relational databases. Other biological databanks are either linked or derived from sequence databanks. They represent all different aspects of biological data. For example: the PROSITE [55] databank, the PRINTS [56] databank, the ProDom [57] databank of protein domain families and the Pfam [58] databank. Each serves a different purpose. In some cases, unknown protein sequences that are too distantly related to any proteins of known structure detect its resemblance by overall sequence alignment. However, relationships can be revealed by its sequence of a particular cluster of residue types, which is known as a motif, pattern, profile or a fingerprint. Both the PROSITE and the PRINTS databanks contain biologically significant patterns and profiles such that one can use these databanks to determine which known protein family a sequence belongs to, or what domains does the sequence contain. Table 2.1 provides examples of key database resources in bioinformatics.

---

[2] http://www.embl.org/
[3] http://www.ncbi.nlm.nih.gov/

Table 2.1: Examples of Key Database Resources in Bioinformatics

Category	Databases and URLs
Comprehensive data center: broad content including sequence, structure, function, etc.	-NCBI (National Center for Biotechnology and Information): http://www.ncbi.nlm.nih.gov/ -EBI (European Bioinformatics Institute): http://www.ebi.ac.uk/ -European Molecular Biology Laboratory (EMBL): http://www.emblheidelberg.de/ -TIGR (the Institute of Genome Research): http://www.tigr.org/
DNA or protein sequence	- GenBank: http://www.ncbi.nlm.nih.gov/Genbank DDBJ (DNA Data Bank of Japan): http://www.ddbj.nig.acjp/ -PIR (Protein Information Resource): http://pir.georgetown.edu/ -Swiss-Prot: http://www.expasy.ch/sprot/sprot-top.html
Bio-molecular interactions	-BIND (Biomolecular Interaction Network Database): http://www.blueprint.org/bind/bind.php. The contents of BIND include high-throughput data submissions and hand-curated information gathered from the scientific literature.
Genomes: complete genome sequences and related information for specific organisms	-Entrez complete genomes: http://www.ncbi.nlm.nih.gov/Entrez/Genome/org.html -Complete genome at EBI: http://www.ebi.ac.uk/genomes/ -MGD (Mouse Genome Database): http://www.informaticsjax.org/ -SGD (Saccharomyces Genome Database): http://genomewww.stanford.edu/Saccharomyces/
Genetics: gene mapping, mutations, and diseases	-GDB (Genome Database): http://gdbwww.gdb.org/gdb/ -OMIM (Online Mendelian Inheritance in Man): http://www3.ncbi.nlm.nih.gov/Omim/searchomim.html -HGMD (Human Gene Mutation Database): http://archive.uwcm.ac.uk/uwcm/mg/hgmdO.html
Gene expression: microarray and cDNA gene expression	- dbEST (Expression Sequence Tag Database): http://www.ncbi.nlm.nih.gov/dbEST/index.html -GEO (Gene Expression Omnibus): http://www.ncbi.nlm.nih.gov/geo/
Structure: three-dimensional structures of small molecules, proteins, nucleic acids (both RNA and DNA) folding predictions	PDB (Protein Data Bank): http://www.rcsb.org/pdb/index.html -NDB (Nucleic Acid Database): http://ndbserver.irutgers.edu/NDB/ndb.html -CSD (Cambridge Structural Database): http://www.ccdc.cam.ac.uk/prods/csd/csd.html
Protein pathway, Protein-protein interactions and metabolic pathway	-KEGG (Kyoto Encyclopedia of Genes and Genomes): http://www.genome.ad.jp/kegg/kegg2.html#pathway -DIP (Database of Interacting Proteins): http:Hdip.doe-mbi.ucla.edu/
Proteomics: proteins, protein family	-JCSG (Joint Center for Structure Genomics): http://www.jcsg.org/scripts/prod/home.html -PKR (Protein Kinase Resource): http://pkr.sdsc.edu/html/index.shtml

2.4 DNA

In this work, we use DNA database as example of biological data but not individual sequences. More details about history of DNA research and properties of DNA will be provided in this section.

DNA is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms (with the exception of RNA viruses). The main role of DNA molecules is the long-term storage of information. DNA is often compared to a set of blueprints, like a recipe or a code, since it contains the instructions needed to construct other components of cells, such as proteins and RNA molecules. The DNA segments that carry this genetic information are called genes, but other DNA sequences have structural purposes, or are involved in regulating the use of this genetic information. Along with RNA and proteins, DNA is one of the three major macromolecules that are essential for all known forms of life [2].

DNA consists of two long polymers of simple units called nucleotides, with backbones made of sugars and phosphate groups joined by ester bonds. These two strands run in opposite directions to each other and are therefore anti-parallel. Attached to each sugar is one of four types of molecules called nucleobases (informally, *bases*). It is the sequence of these four nucleobases along the backbone that encodes information. This information is read using the genetic code, which specifies the sequence of the amino acids within proteins. The code is read by copying stretches of DNA into the related nucleic acid RNA, in a process called transcription.

Within cells, DNA is organized into long structures called chromosomes. These chromosomes are duplicated before cells divide, in a process called DNA replication. Eukaryotic organisms (animals, plants, fungi, and protists) store most of their DNA inside the cell nucleus and some of their DNA in organelles, such as mitochondria or chloroplasts. In contrast, prokaryotes (bacteria and archaea) store their DNA only in the cytoplasm. Within the chromosomes, chromatin proteins such as histones compact and

organize DNA. These compact structures guide the interactions between DNA and other proteins, helping control which parts of the DNA are transcribed.

2.4.1 History of DNA Research

DNA was first isolated by the Swiss physician Friedrich Miescher who, in 1869, discovered a microscopic substance in the pus of discarded surgical bandages. As it resided in the nuclei of cells, he called it "nuclein"[59]. In 1919, Phoebus Levene identified the base, sugar and phosphate nucleotide unit [60]. Levene suggested that DNA consisted of a string of nucleotide units linked together through the phosphate groups. However, Levene thought the chain was short and the bases repeated in a fixed order. In 1937 William Astbury produced the first X-ray diffraction patterns that showed that DNA had a regular structure [61].

In 1927 Nikolai Koltsov proposed that inherited traits would be inherited via a "giant hereditary molecule" which would be made up of "two mirror strands that would replicate in a semi-conservative fashion using each strand as a template"[62]. In 1928, Frederick Griffith discovered that traits of the "smooth" form of the *Pneumococcus* could be transferred to the "rough" form of the same bacteria by mixing killed "smooth" bacteria with the live "rough" form [63]. This system provided the first clear suggestion that DNA carries genetic information—the Avery– MacLeod –McCarty experiment—when Oswald Avery, along with coworkers Colin MacLeod and Maclyn McCarty, identified DNA as the transforming principle in 1943 [64]. DNA's role in heredity was confirmed in 1952, when Alfred Hershey and Martha Chase in the Hershey–Chase experiment showed that DNA is the genetic material of the T2 phage [65].

In 1953, James D. Watson and Francis Crick suggested what is now accepted as the first correct double-helix model of DNA structure in the journal *Nature* [66]. Their double-helix, molecular model of DNA was then based on a single X-ray diffraction image (labeled as "Photo 51")[67] taken by Rosalind Franklin and Raymond Gosling in May 1952, as well as the information that the DNA bases are paired — also obtained through private communications from Erwin Chargaff in the previous years. Chargaff's

rules played a very important role in establishing double-helix configurations for B-DNA as well as A-DNA.

Experimental evidence supporting the Watson and Crick model were published in a series of five articles in the same issue of *Nature* [68].Of these, Franklin and Gosling's paper was the first publication of their own X-ray diffraction data and original analysis method that partially supported the Watson and Crick model [69]; this issue also contained an article on DNA structure by Maurice Wilkins and two of his colleagues, whose analysis and *in vivo* B-DNA X-ray patterns also supported the presence *in vivo* of the double-helical DNA configurations as proposed by Crick and Watson for their double-helix molecular model of DNA in the previous two pages of *Nature*[70]. In 1962, after Franklin's death, Watson, Crick, and Wilkins jointly received the Nobel Prize in Physiology or Medicine [71]. However, Nobel rules of the time allowed only living recipients, but a vigorous debate continue on who should receive credit for the discovery [72].

In an influential presentation in 1957, Crick laid out the central dogma of molecular biology, which foretold the relationship between DNA, RNA, and proteins, and articulated the "adaptor hypothesis"[73]. Final confirmation of the replication mechanism that was implied by the double-helical structure followed in 1958 through the Meselson–Stahl experiment [74]. Further work by Crick and coworkers showed that the genetic code was based on non-overlapping triplets of bases, called codons, allowing Har Gobind Khorana, Robert W. Holley and Marshall Warren Nirenberg to decipher the genetic code [75]. These findings represent the birth of molecular biology.

2.4.2 Properties of DNA

DNA (deoxyribonucleic acid) is an acid that contains biological information that is carried from one generation to the next. It is composed of four bases, or nucleotides: adenine, thymine, cytosine and guanine [76]. These are typically referred to by their initials: A, T, C and G. DNA is arranged in two strands that are entwined in a double-helix such that A and T bases are bound together, and likewise for C and G. These pairings are called "complementary". The two strands are conventionally read in opposite directions. To illustrate, if one strand contained the sequence "GATACCA", the other strand (when read backwards) would contain "TGGTATC". Figure 2.1 illustrates the structure of DNA.

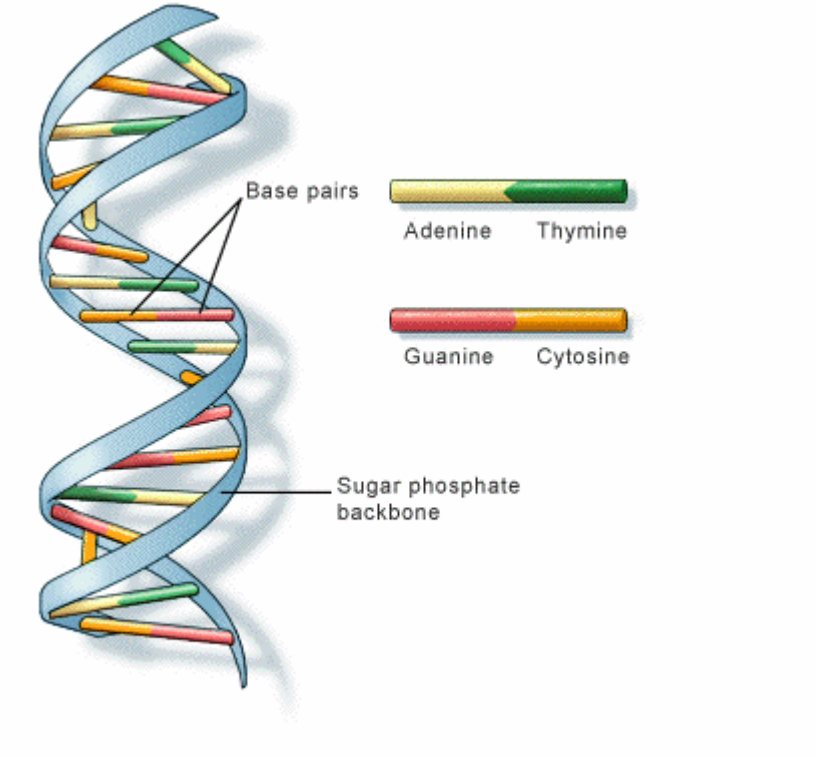


Figure 2.1: Structure of DNA

The backbone of the DNA strand is made from alternating phosphate and sugar residues [77]. The sugar in DNA is 2-deoxyribose, which is a pentose (five-carbon) sugar. The sugars are joined together by phosphate groups that form phosphodiester bonds between the third and fifth carbon atoms of adjacent sugar rings. These asymmetric bonds mean a strand of DNA has a direction. In a double helix the direction

of the nucleotides in one strand is opposite to their direction in the other strand: the strands are *antiparallel*. The asymmetric ends of DNA strands are called the 5′ (*five prime*) and 3′ (*three prime*) ends, with the 5' end having a terminal phosphate group and the 3' end a terminal hydroxyl group. One major difference between DNA and RNA is the sugar, with the 2-deoxyribose in DNA being replaced by the alternative pentose sugar ribose in RNA. The DNA double helix is stabilized primarily by two forces: hydrogen bonds between nucleotides and base-stacking interactions among the aromatic nucleobases [78]. Hydrogen bonds shown as dotted lines in Figure 2.2.

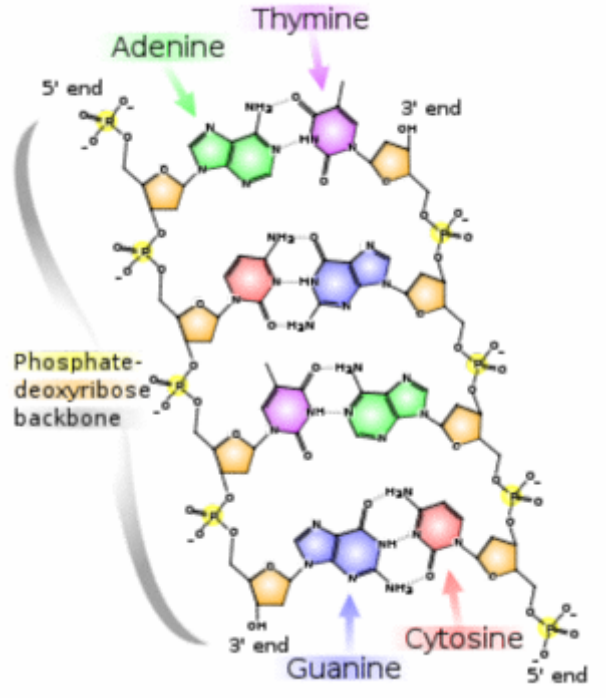


Figure 2.2: Chemical structure of DNA

Because both strands can be uniquely determined by examining only one strand, DNA can be represented as a one-dimensional sequence composed of A, C, G and T characters. Some parts of an organism's DNA comprise genes. These genes contain DNA that will eventually be translated into proteins. During this translation, the four DNA bases are read in triplets called codons. Each codon (there are $4^3 = 64$ in total) corresponds to one of the twenty amino acids. For example, the codon "ATA" will be translated to the amino acid "I" (isoleucine). Some amino acids have as many as six source codons, resulting in a bias in the amino acid distribution. Genomic regions are

often divided into separate regions (exons) which are combined to form the protein. The exons can be combined in different ways to form slightly different proteins. Consequently, a certain number of genes can create a greater number of proteins.

DNA is replicated by splitting the helix in two, separating the two strands. Because a new complementary strand can be formed from a single strand, each separate strand can be formed into a new helix that is identical to the original. However, copy errors cause random mutations and the new helices diverge slightly from the original. When DNA is copied, the new stretches may not necessarily be formed in the same direction as the original. This leads to reverse complementary repeats in the sequence, of which general text compressors cannot take advantage.

DNA and protein sequences manifest different properties from other kinds of data. While compression algorithms for, say, text and image files, exploit small repeated patterns and contextual similarities to achieve compression, these methods cannot be applied successfully to DNA and protein. The properties particular to biological sequences are outlined below.

In the process of biological replication, large chunks of DNA are often copied from one place to another. It is not uncommon for a single organism to contain the same stretch of DNA more than once. Additionally, different organisms can share segments of DNA due to common ancestry and, roughly speaking, more similar organisms are more likely to share common DNA sequences. However, as mentioned above, the copying process is not exact. During duplication as well as afterwards, random mutations occur. Some nucleotides may be transformed or deleted, or extra bases may be inserted into the sequence. Consequently, DNA sequences (and hence, protein sequences) contain stretches that are approximately the same as regions from earlier in the sequence. Distinct proteins can also share subsequences because they have been formed using some of the same exons. These near-repeats can be used to achieve compression.

Another difference between biological sequences and other data is the nature of repeated regions. Aside from the inexact matching described above, repeats in DNA and protein are often quite long in comparison to those in, say, English text. While certain short patterns in English such as “the” and “ing” are likely to show up with high frequency, the repeated patterns in biological sequences are typically much longer and less frequent. It is for this reason that traditional compression algorithm — for example, the algorithm in [25] — performs poorly on DNA and protein.

One of protein’s features is its no uniform composition of source symbols. Whereas in DNA each base can in general be considered equally likely, some amino acids are more common than others. In addition to this, when mutations occur in the underlying DNA certain amino acids are inclined to change into “similar” amino acids.

2.4.3 Chemical Modifications

The expression of genes is influenced by how the DNA is packaged in chromosomes, in a structure called chromatin. Base modifications can be involved in packaging, with regions that have low or no gene expression usually containing high levels of methylation of cytosine bases. For example, cytosine methylation, produces 5-methylcytosine, which is important for X-chromosome inactivation [79]. The average level of methylation varies between organisms – the worm *Caenorhabditis elegans* lacks cytosine methylation, while vertebrates have higher levels, with up to 1% of their DNA containing 5-methylcytosine [80]. Despite the importance of 5-methylcytosine, it can deaminate to leave a thymine base, so methylated cytosines are particularly prone to mutations. Other base modifications include adenine methylation in bacteria, the presence of 5-hydroxymethylcytosine in the brain [81], and the glycosylation of uracil to produce the "J-base" in kinetoplastids [82].

DNA can be damaged by many sorts of mutagens, which change the DNA sequence. Mutagens include oxidizing agents, alkylating agents and also high-energy electromagnetic radiation such as ultraviolet light and X-rays. The type of DNA damage produced depends on the type of mutagen. For example, UV light can damage

DNA by producing thymine dimers, which are cross-links between pyrimidine bases. On the other hand, oxidants such as free radicals or hydrogen peroxide produce multiple forms of damage, including base modifications, particularly of guanosine, and double-strand breaks. A typical human cell contains about 150,000 bases that have suffered oxidative damage. Of these oxidative lesions, the most dangerous are double-strand breaks, as these are difficult to repair and can produce point mutations, insertions and deletions from the DNA sequence, as well as chromosomal translocations [83].

Many mutagens fit into the space between two adjacent base pairs, this is called *intercalation*. Most intercalators are aromatic and planar molecules; examples include ethidium bromide, daunomycin, and doxorubicin. In order for an intercalator to fit between base pairs, the bases must separate, distorting the DNA strands by unwinding of the double helix. This inhibits both transcription and DNA replication, causing toxicity and mutations. As a result, DNA intercalators are often carcinogens, and benzo[*a*] pyrene diol epoxide, acridines, aflatoxin and ethidium bromide are well-known examples [84]. Nevertheless, due to their ability to inhibit DNA transcription and replication, other similar toxins are also used in chemotherapy to inhibit rapidly growing cancer cells [85].

2.4.4 Biological Functions

DNA usually occurs as linear chromosomes in eukaryotes, and circular chromosomes in prokaryotes. The set of chromosomes in a cell makes up its genome; the human genome has approximately 3 billion base pairs of DNA arranged into 46 chromosomes [86]. The information carried by DNA is held in the sequence of pieces of DNA called genes. Transmission of genetic information in genes is achieved via complementary base pairing. For example, in transcription, when a cell uses the information in a gene, the DNA sequence is copied into a complementary RNA sequence through the attraction between the DNA and the correct RNA nucleotides. Usually, this RNA copy is then used to make a matching protein sequence in a process called translation, which depends on the same interaction between RNA nucleotides. In alternative fashion, a cell may simply copy its genetic information in a process called

DNA replication. Here we focus on the interactions between DNA and other molecules that mediate the function of the genome.

2.4.4.1 Genes and Genomes

Genomic DNA is tightly and orderly packed in the process called DNA condensation to fit the small available volumes of the cell. In eukaryotes, DNA is located in the cell nucleus, as well as small amounts in mitochondria and chloroplasts. In prokaryotes, the DNA is held within an irregularly shaped body in the cytoplasm called the nucleoid [87]. The genetic information in a genome is held within genes, and the complete set of this information in an organism is called its genotype. A gene is a unit of heredity and is a region of DNA that influences a particular characteristic in an organism. Genes contain an open reading frame that can be transcribed, as well as regulatory sequences such as promoters and enhancers, which control the transcription of the open reading frame.

In many species, only a small fraction of the total sequence of the genome encodes protein. For example, only about 1.5% of the human genome consists of protein-coding exons, with over 50% of human DNA consisting of non-coding repetitive sequences. The reasons for the presence of so much noncoding DNA in eukaryotic genomes and the extraordinary differences in genome size, or *C-value*, among species represent a long-standing puzzle known as the "C-value enigma"[88]. However, DNA sequences that do not code protein may still encode functional non-coding RNA molecules, which are involved in the regulation of gene expression.

Some noncoding DNA sequences play structural roles in chromosomes. Telomeres and centromeres typically contain few genes, but are important for the function and stability of chromosomes [89]. Abundant forms of noncoding DNA in humans are pseudogenes, which are copies of genes that have been disabled by mutation [90]. These sequences are usually just molecular fossils, although they can occasionally serve as raw genetic material for the creation of new genes through the process of gene duplication and divergence.

2.4.4.2 Transcription and Translation

A gene is a sequence of DNA that contains genetic information and can influence the phenotype of an organism. Within a gene, the sequence of bases along a DNA strand defines a messenger RNA sequence, which then defines one or more protein sequences. The relationship between the nucleotide sequences of genes and the amino-acid sequences of proteins is determined by the rules of translation, known collectively as the genetic code. The genetic code consists of three-letter 'words' called codons formed from a sequence of three nucleotides (e.g. ACT, CAG, TTT).

In transcription, the codons of a gene are copied into messenger RNA by RNA polymerase. This RNA copy is then decoded by a ribosome that reads the RNA sequence by base-pairing the messenger RNA to transfer RNA, which carries amino acids. Since there are 4 bases in 3-letter combinations, there are 64 possible codons. These encode the twenty standard amino acids, giving most amino acids more than one possible codon. There are also three 'stop' or 'nonsense' codons signifying the end of the coding region; these are the TAA, TGA and TAG codons.

2.4.4.3 Replication

Cell division is essential for an organism to grow, but, when a cell divides, it must replicate the DNA in its genome so that the two daughter cells have the same genetic information as their parent. The double-stranded structure of DNA provides a simple mechanism for DNA replication. Here, the two strands are separated and then each strand's complementary DNA sequence is recreated by an enzyme called DNA polymerase. This enzyme makes the complementary strand by finding the correct base through complementary base pairing, and bonding it onto the original strand. As DNA polymerases can only extend a DNA strand in a 5′ to 3′ direction, different mechanisms are used to copy the anti parallel strands of the double helix [91]. In this way, the base on the old strand dictates which base appears on the new strand, and the cell ends up with a perfect copy of its DNA.

As in Figure 2.3, the double helix is unwound by a helicase and topoisomerase. Next, one DNA polymerase produces the leading strand copy. Another DNA polymerase binds to the lagging strand. This enzyme makes discontinuous segments (called Okazaki fragments) before DNA ligase joins them together. Figure 2.4 describes the central dogma of molecular biology.

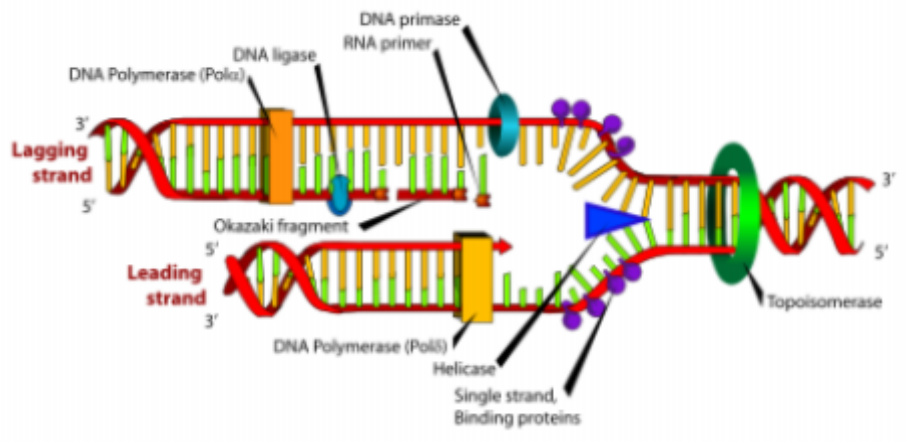


Figure 2.3: DNA replication.

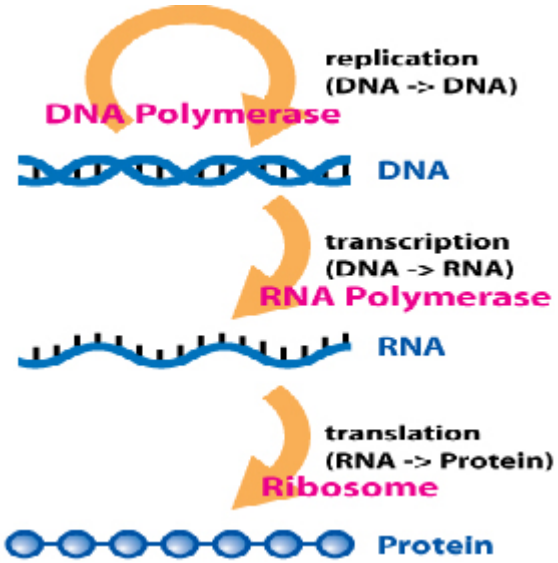


Figure 2.4: Central dogma of molecular biology

2.4.4.4 Interactions with proteins

All the functions of DNA depend on interactions with proteins. These protein interactions can be non-specific, or the protein can bind specifically to a single DNA sequence. Enzymes can also bind to DNA and of these, the polymerases that copy the DNA base sequence in transcription and DNA replication are particularly important.

Structural proteins that bind DNA are well-understood examples of non-specific DNA-protein interactions. Within chromosomes, DNA is held in complexes with structural proteins. These proteins organize the DNA into a compact structure called chromatin. In eukaryotes this structure involves DNA binding to a complex of small basic proteins called histones, while in prokaryotes multiple types of proteins are involved [92]. The histones form a disk-shaped complex called a nucleosome, which contains two complete turns of double-stranded DNA wrapped around its surface. These non-specific interactions are formed through basic residues in the histones making ionic bonds to the acidic sugar-phosphate backbone of the DNA, and are therefore largely independent of the base sequence. Chemical modifications of these basic amino acid residues include methylation, phosphorylation and acetylation [93]. These chemical changes alter the strength of the interaction between the DNA and the histones, making the DNA more or less accessible to transcription factors and changing the rate of transcription [94]. Other non-specific DNA-binding proteins in chromatin include the high-mobility group proteins, which bind to bent or distorted DNA. These proteins are important in bending arrays of nucleosomes and arranging them into the larger structures that make up chromosomes.

A distinct group of DNA-binding proteins are the DNA-binding proteins that specifically bind single-stranded DNA. In humans, replication protein A is the best-understood member of this family and is used in processes where the double helix is separated, including DNA replication, recombination and DNA repair [95]. These binding proteins seem to stabilize single-stranded DNA and protect it from forming stem-loops or being degraded by nucleases.

In contrast, other proteins have evolved to bind to particular DNA sequences. The most intensively studied of these are the various transcription factors, which are proteins that regulate transcription. Each transcription factor binds to one particular set of DNA sequences and activates or inhibits the transcription of genes that have these sequences close to their promoters. The transcription factors do this in two ways. Firstly, they can bind the RNA polymerase responsible for transcription, either directly or through other mediator proteins; this locates the polymerase at the promoter and allows it to begin transcription. Alternatively, transcription factors can bind enzymes that modify the histones at the promoter; this will change the accessibility of the DNA template to the polymerase [96]. As these DNA targets can occur throughout an organism's genome, changes in the activity of one type of transcription factor can affect thousands of genes. Consequently, these proteins are often the targets of the signal transduction processes that control responses to environmental changes or cellular differentiation and development. The specificity of these transcription factors' interactions with DNA come from the proteins making multiple contacts to the edges of the DNA bases, allowing them to "read" the DNA sequence. Most of these base-interactions are made in the major groove, where the bases are most accessible [97].

Nucleases are enzymes that cut DNA strands by catalyzing the hydrolysis of the phosphodiester bonds. Nucleases that hydrolyse nucleotides from the ends of DNA strands are called exonucleases, while endonucleases cut within strands. The most frequently used nucleases in molecular biology are the restriction endonucleases, which cut DNA at specific sequences. For instance, the EcoRV enzyme shown to the left recognizes the 6-base sequence 5′-GAT|ATC-3′ and makes a cut at the vertical line. In nature, these enzymes protect bacteria against phage infection by digesting the phage DNA when it enters the bacterial cell, acting as part of the restriction modification system [98]. In technology, these sequence-specific nucleases are used in molecular cloning and DNA fingerprinting. Enzymes called DNA ligases can rejoin cut or broken DNA strands [99]. Ligases are particularly important in lagging strand DNA replication, as they join together the short segments of DNA produced at the replication fork into a complete copy of the DNA template. They are also used in DNA repair and genetic recombination [99].

2.4.5 Uses in Technology

DNA has many applications in technology such as genetic engineering, forensics, bioinformatics, DNA nanotechnology and anthropology as the following.

2.4.5.1 Genetic Engineering

Methods have been developed to purify DNA from organisms, such as phenol-chloroform extraction, and to manipulate it in the laboratory, such as restriction digests and the polymerase chain reaction. Modern biology and biochemistry make intensive use of these techniques in recombinant DNA technology. Recombinant DNA is a man-made DNA sequence that has been assembled from other DNA sequences. They can be transformed into organisms in the form of plasmids or in the appropriate format, by using a viral vector [100]. The genetically modified organisms produced can be used to produce products such as recombinant proteins, used in medical research [101], or be grown in agriculture [102].

2.4.5.2 Forensics

Forensic scientists can use DNA in blood, semen, skin, saliva or hair found at a crime scene to identify a matching DNA of an individual, such as a perpetrator. This process is formally termed DNA profiling, but may also be called "genetic fingerprinting". In DNA profiling, the lengths of variable sections of repetitive DNA, such as short tandem repeats and minisatellites, are compared between people. This method is usually an extremely reliable technique for identifying a matching DNA. However, identification can be complicated if the scene is contaminated with DNA from several people. DNA profiling was developed in 1984 by British geneticist Sir Alec Jeffreys [103], and first used in forensic science to convict Colin Pitchfork in the 1988 Enderby murders case [104].

People convicted of certain types of crimes may be required to provide a sample of DNA for a database. This has helped investigators solve old cases where only a DNA sample was obtained from the scene. DNA profiling can also be used to identify victims of mass casualty incidents. On the other hand, many convicted people have been released

from prison on the basis of DNA techniques, which were not available when a crime had originally been committed.

2.1.5.3 DNA Nanotechnology

DNA nanotechnology uses the unique molecular recognition properties of DNA and other nucleic acids to create self-assembling branched DNA complexes with useful properties [105]. DNA is thus used as a structural material rather than as a carrier of biological information. This has led to the creation of two-dimensional periodic lattices (both tile-based as well as using the "DNA origami" method) as well as three-dimensional structures in the shapes of polyhedra [106]. Nanomechanical devices and algorithmic self-assembly have also been demonstrated [107], and these DNA structures have been used to template the arrangement of other molecules such as gold nanoparticles and streptavidin proteins [108].

2.4.5.4 Anthropology

Because DNA collects mutations over time, which is then inherited, it contains historical information, and, by comparing DNA sequences, geneticists can infer the evolutionary history of organisms, their phylogeny [109]. This field of phylogenetics is a powerful tool in evolutionary biology. If DNA sequences within a species are compared, population geneticists can learn the history of particular populations. This can be used in studies ranging from ecological genetics to anthropology; For example, DNA evidence is being used to try to identify the Ten Lost Tribes of Israel [110]. DNA has also been used to look at modern family relationships, such as establishing family relationships between the descendants of Sally Hemings and Thomas Jefferson. This usage is closely related to the use of DNA in criminal investigations detailed above. Indeed, some criminal investigations have been solved when DNA from crime scenes has matched relatives of the guilty individual [111].

2.5 Bioinformatics Software and Tools

Software tools for bioinformatics range from simple command-line tools, to more complex graphical programs and standalone web-services available from various bioinformatics companies or public institutions.

2.5.1 Open Source Bioinformatics Software

Many free and open source software tools existed and continued to grow since the1980s [112]. The combination of a continued need for new algorithms for the analysis of emerging types of biological readouts, the potential for innovative in silico experiments, and freely available open code bases have helped to create opportunities for all research groups to contribute to both bioinformatics and the range of open source software available, regardless of their funding arrangements. The open source tools often act as incubators of ideas, or community-supported plug-ins in commercial applications. They may also provide de facto standards and shared object models for assisting with the challenge of bioinformatics integration.

The range of open source software packages includes titles such as Bioconductor, BioPerl, BioJava, BioRuby, Bioclipse, EMBOSS, Taverna workbench, and UGENE. In order to maintain this tradition and create further opportunities, the non-profit Open Bioinformatics Foundation [112] have supported the annual Bioinformatics Open Source Conference (BOSC) since 2000 [113].

2.5.2 Web Services in Bioinformatics

SOAP [114] and REST-based interfaces[115] have been developed for a wide variety of bioinformatics applications allowing an application running on one computer in one part of the world to use algorithms, data and computing resources on servers in other parts of the world. The main advantages derive from the fact that end users do not have to deal with software and database maintenance overheads.

Basic bioinformatics services are classified by the EBI [116] into three categories: SSS (Sequence Search Services) [5], MSA (Multiple Sequence Alignment) [117] and

BSA (Biological Sequence Analysis) [118]. The availability of these service-oriented bioinformatics resources demonstrate the applicability of web based bioinformatics solutions, and range from a collection of standalone tools with a common data format under a single, standalone or web-based interface, to integrative, distributed and extensible bioinformatics workflow management systems.

2.6 Bioinformatics Algorithms and Applications

Biological databanks can be less informative without any applications to analyze data or search for new information. Researchers have developed a large collection of bioinformatics applications and algorithms over the last few decades. These applications and algorithms are being used for various sequence analysis range from homologous sequence comparisons to protein folding prediction [41].

2.6.1 Bioinformatics Algorithms

The following gives a brief description of some of the available bioinformatics algorithms.

1- Global Alignment Algorithm

Global alignments of two sequences can be used to identify the common regions between the two sequences, hence allowing users to find out how two sequences are related to each other over periods of biological evolution. A general global alignment technique is the Needleman-Wunsch algorithm [119], which is based on dynamic programming. It can use this method to identify homologous sequences. Figure 2.5 gives an example of global alignment.

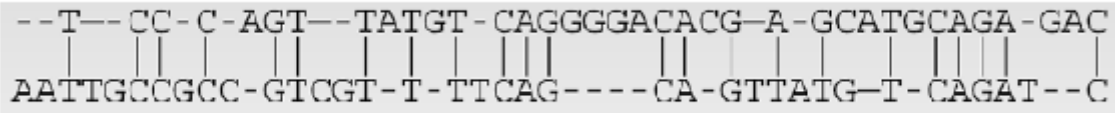


Figure 2.5: Global Alignment

2- Local Alignment Algorithm

Local alignments of two sequences allow users to determine whether two sequences are distantly related. This is because global alignment may show that two sequences do not have any homologous regions over their entire length of sequences. However, local alignment compares segments of the two sequences and aligns multiple local regions of two sequences. We know that as the evolutionary distance between two sequences increases, the length of a local alignment required to achieve a statistically significant score also increases[4]. Therefore, if we have long regions of two sequences that can be successfully aligned, then we can say that the two sequences are related. The Smith-Waterman algorithm [120] is a general local alignment method also based on dynamic programming. Figure 2.6 gives an example of local alignment.

Figure 2.6: local Alignment

Illustration of global and local alignments demonstrating the 'gappy' quality of global alignments that can occur if sequences are insufficiently similar in Figure 2.7.

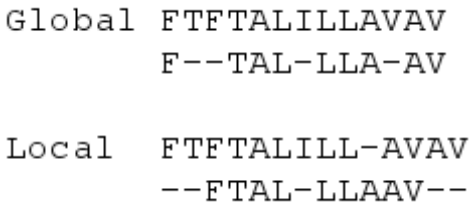


Figure 2.7: Comparison between global and local sequence alignments

[4]http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/rules.html

3- Multiple Sequence Alignment Algorithm

Program such as Clustal W [121] carries out alignments of multiple sequences. It is used to align a set of related protein sequences. The alignment result should show the common regions of the set of protein sequences. Therefore, we can determine which regions of the set of related sequences do not change over the evolution. In Figure 2.8 shows the first 90 positions of a protein multiple sequence alignment of instances of the acidic ribosomal protein P0 (L10E) from several organisms. Some operations are produced from multiple sequence alignment such as inserstion, deletion and subistitions as in Figure 2.9.

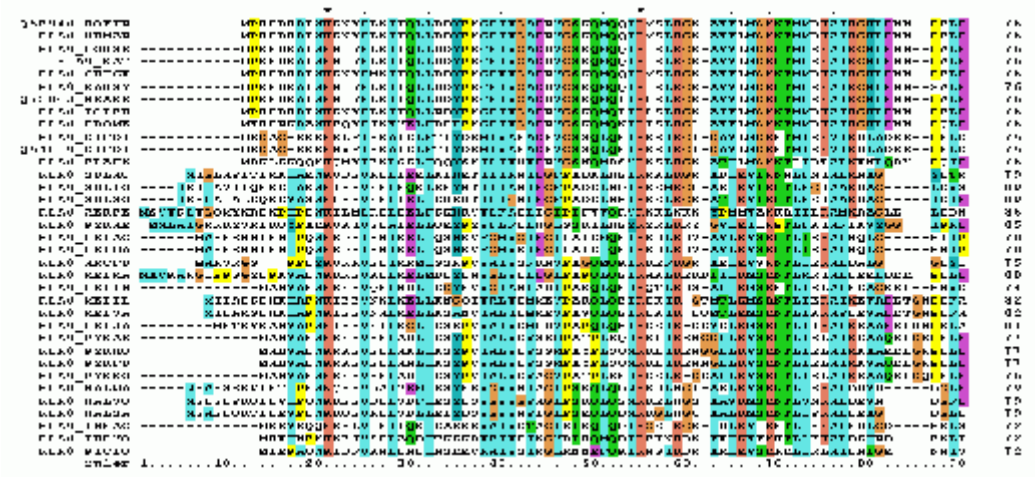


Figure 2.8: Multiple Sequence Alignment generated with ClustalX

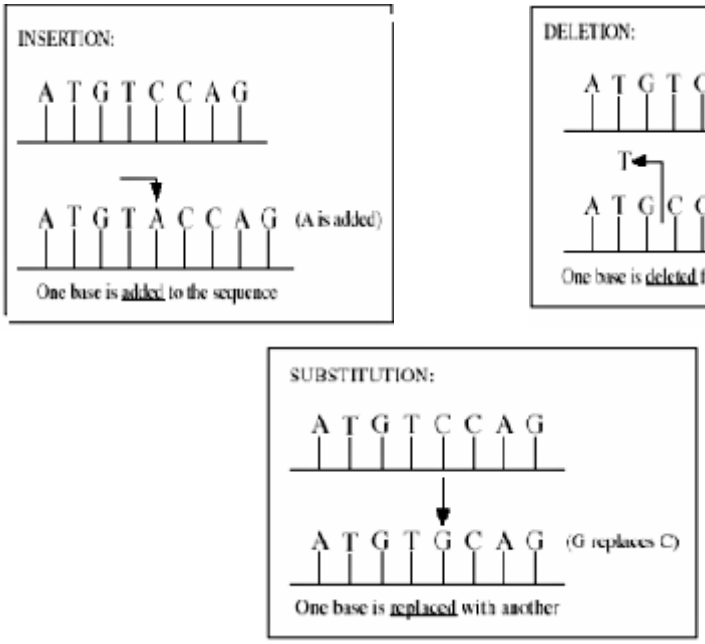


Figure 2.9: Various types of operations

4- Similarity Search Algorithm

Similarity search algorithm is mainly used to identify related or similar sequences from databanks. Package such as BLAST and FastA specialize in this following:

– Basic Local Alignment Search Tools (BLAST) [122] is a collection of different programs which allow different combinations of nucleic acid and protein query sequences and databank to be used. It carries out local alignments between an unknown sequence and the sequence databank. This results in a large set of locally aligned sequences. The major strength of BLAST is the statistical significance evaluation that it can perform on the results and its fast speed. The statistical expectation value (E-value) is evaluated according to the scores of each local alignment. BLAST then ranks the results based on the expectation value. It tells users the probability of a particular alignment being a chance occurrence. So if we have a result that has an E-value of 0, it means that the probability for this alignment to occur by chance is 0.

– FastA algorithm [123] is a similarity search program which can be used to search a nucleotide sequence databank with a nucleotide input query, or search a protein databank with a protein input query. FastA speeds up search by using several passes over the data source and only retaining a ”best matching” subset for further analysis at each pass. The latest version of FastA performs a statistical evaluation of its results similar to that of BLAST. However, the statistical model used is less rigorous. It returns a score called Z-score, which is then converted into E-value.

2.6.2 Bioinformatics Applications

The following gives a brief description of some of the available bioinformatics applications.

1- 2D and 3D protein prediction

We know that proteins are formed by joining amino acids by peptide bonds into a chain. They differ in length and in the arrangement of the amino acids [124]. In water, the chain folds up into a unique three-dimensional (3D) structure. The 3D structure of a protein determines its biological function. A detailed analysis of the underlying chemistry shows that the folding of protein sequence into 3D structure is only possible if the protein forms regular patterns of a macroscopic substructure called ”secondary structure”. Therefore, predicting the secondary structure of a protein sequence helps researchers to predict its final 3D structure and subsequently, its biological function.

2- Phylogenetics

Phylogenetics is the study of phylogeny, which is the study of the evolutionary development and history of a species or higher taxonomic grouping of organisms. Program such as PHYLIP[5] helps create evolutionary trees for sets of related sequences. The tree shows the evolutionary distance between the set of sequences, and if the sequences are from different organisms, then we can see the evolutionary relationships of this sequence between the organisms.

3- Compression

Compression is useful because it helps to reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. On the downside, compressed data must be decompressed to be used, and this extra processing may be detrimental to some applications. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed (the option of decompressing the video in full before watching it may be

[5]http://evolution.genetics.washington.edu/phylip.html

inconvenient, and requires storage space for the decompressed video). The design of data compression schemes therefore involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (if using a lossy compression scheme), and the computational resources required to compress and uncompress the data.

In computer science and information theory, data compression, source coding or bit-rate reduction is the process of encoding information using fewer bits than the original representation would use [18]. Many kinds of data are compressed, from linguistic and programming texts to digital images, audio and movies. Modern biological science produces vast amounts of DNA and protein sequence data. This is fuelling the need for efficient algorithms to estimate the information content of sequence data for sequence compression. In next chapter, various methods of compression for DNA are described.

2.7 Statistical Analysis of DNA Sequences

Statistical analysis of DNA sequences is essential immediately after the procedure more structured assembly sequencing. Analyze the frequencies of nucleotides, the chemical families - the purines (A and G) and pyrimidines (C and T) -, or codons of known motifs are among the first to perform characterizations [125]. Follow one another more thorough analysis with respect to distributions, statistical comparison with other sequences [126], characterization of correlations [127] and frequencies [128], GC richness and statistical modeling [129], statistical differentiation between coding and non-coding regions [130], among others. Therefore, the statistical analysis of DNA [131] is a contemporary of the first genome sequencing and has accompanied its evolution. It is so important that the (bio) statistics is a cornerstone of bioinformatics, molecular biology along with and computer science.

Statistical analysis is essential in the accumulation of knowledge of the peculiarities that characterize the sequences, both at intra-genomic, from prokaryotes to eukaryotes, since the coding regions conserved regions. Such knowledge is critical when building, for example, models probabilistic prediction of sequences for the design of compression of information. Statistical analysis of the occurrences of the sequence of

symbols that make up a gene or a genome can be arranged to build probabilistic models of prediction. In such cases, the statistical analysis is performed on-line and prediction of symbols is based on history, which corresponds to the knowledge base of the probabilistic model prediction. As the study developed methodologies based on cooperation to achieve greater efficiency in the compression of DNA sequences, this approach is of course part and the methodologies developed to generate op-code table in this area will be properly explained further in Chapter 4.

2.8. Information Theory: Biological Information

The DNA molecule encodes information that, by convention, is represented as a sequence of bases, represented by characters belonging to the alphabet {A, C, G, T}.

Assuming that each nucleotide, shown here as a character, occurs with equal probability and all the other symbols in the sequence are independent. We can use the basic information theory [24,132] to assert that the coding sequence of a computer would need 2 bits per character. This would be the case of maximum entropy.

Maximum Entropy = $\log_2$ (n) where n is the number of possible states.

Examples:

-Representation of results of flipping a coin in the air → $\log_2$ (2) = 1 bit

-Representation of a nucleotide in a chain → $\log_2$ (4) = 2 bits

-Representation of an amino acid in a chain → $\log_2$ (20) = 4:32 bits

In fact the DNA molecule must be understood as highly ordered, objectively drawn, and the result of a clearance by evolution. Thus, it is expected that there legitimately statistical properties, regularities, similarities, interdependencies, etc.. That allows us to relieve the entropy in its representation and to describe the natural DNA sequences as nonrandom.

A common but wrong assumption based on the assumption that the coding regions DNA is inherently more compressible, that they are formed by codons, therefore 64 combinations of three nucleotides, which are designed to encode only 20 amino acids, given that the addition of start and stop codons, three in total. Thus, we are 61 codons that give content to exons, and calculating the maximum of entropy per nucleotide for this reality we have $\log_2 (61 / 3) \approx 1.977$ bits. In fact, the coding regions are less compressible than the non-coding regions (Introns), justified largely by abundant repetitions or repetitions introns present in the rough, so the case in several studies [133,134].

The complexity of a sequence can be defined as the richness of its vocabulary. It is important to determine how many different words of length appear in the sequence. This approach to linguistic complexity was introduced by Trifonov in [135].

The low-complexity regions of DNA sequences [136,137] are of high importance to the issue of compression [138], that contains more redundancy because qualitative or quantitative, which features simple repetitions or reverse, exact or approximate.

Compression of DNA sequences is an interesting challenge but great difficulty in recent years, the most successful algorithms surpassed their predecessors in less than 0.01 bits per base, which is demonstrative of the difficulty inherent in this field.

2.9 Entropy coding in Genomic Sequences

It was in 1972 that Gatlin published the seminal book [139] for the introduction of entropy analysis in DNA sequences. This work explored the relationship between information theory and biological information as well as the applicability of the concepts of entropy in the analysis of DNA sequences. This was followed by several studies that explored the applications and potential of this new avenue of research, since the prediction of genes, segmentation, and compression of information or the study of biological importance of repetitive DNA regions in the estimation of entropy [140, 141]. In short, all these attempts to characterize the complexity of the genetic code, but given

the multiplicity of factors contributing to this complexity, as is the case of synonymous codons, replications, deletions, the palindromes, etc...

The entropy, redundancy, differentiation probabilistic predictability, complexity and compression are related concepts. Studying these concepts having as scope of DNA sequences is a challenge, just think that the sequences of peptides have on average 99% of unpredictability [142,143], which is not clearly understood practically nothing of the complexity of its language, or that the DNA of less complex organisms such as viruses and bacteria present levels of entropy very close to 2 bits per base [144]. This subtlety of the code of life is a real lesson, subject to millions of years of evolution, how to draw simple code, easy to replicate, with lots of information and ability to reasonably resistant to errors, is a virtuous balance between fidelity and error.

One way to compress the natural uses of DNA information to obtain an estimate of the entropy with the prediction that provides a high degree of accuracy of a particular sequence based on the portion known in advance [133]. The DNA repeats include natural rough, a fact well known and documented as well as the statistical properties of these events [145].

2.9.1 Entropy According to Shannon's Theory

The information theory [132] is used for the importance of understanding and estimation of entropy. Let S be a discrete source emitting symbols X_i. The symbols X_i are modeled as random variables with realizations infinite alphabet X of length | X | = L. The output of the source is a sequence of random variables $X_1, X_2, ..., X_n$, and can be modeled as a stochastic process with entropy

$$H(S) = \lim_{n \to \infty} \frac{1}{n} H(X_1, ... X_n) \qquad (2.1)$$

An upper bound for the entropy estimate can be defined by assuming a stationary source, without memory, with an alphabet of symbols uniformly distributed. In the case of DNA, this upper limit may correspond to two bits per symbol, and given that we are dealing with the code of life, which is expected in some order, it is presumed that the actual

entropy may be lower [146]. The fundamental theorem of Shannon [147] compresses the information, and establishes that the information from any source can be compressed without any loss in proportion to their rate of entropy. Therefore, the compression rate in bits per symbol achieved by an optimal compression algorithm for data *s* generated by a source *S*, is a good approximation to its true entropy rate, defined by Equation 2.2, where |. | represents the set size in bits or symbols.

$$H(S) \approx \frac{|comp(s)|}{|s|} \qquad (2.2)$$

Entropy and data compression concepts are necessarily intertwined. To estimate the entropy of DNA sequence [133], we allow to accurately suggesting the degree of compression in the recoding of compact sequence.

If the DNA code was a purely random string , the maximum entropy would be the best way to represent it using two bits for each of the four different symbols, but there are regularities and specific properties of the sequence statistically verifiable, proving the existence of reducible entropy [148]. The degree of the entropy of DNA is still to be discovered for paving the way for research that will provide a better understanding of the reasons which nature uses peculiar code, and leading to genomic discoveries / compression and necessarily phylogenetic information.

The compression of the nucleotide sequences can be performed by some simulations, and taking into account the entropy analysis provided. However, it is known that entropy and compression are directly related, but actually as great as the compressor is not always able to reach the compression theoretically predicted by the analysis of entropy.

Recalling, the formula of Shannon's information entropy [147] has been in Equation 2.3:

$$H = -\sum_{i}^{n} p_i \log_2 p_i \qquad (2.3)$$

Analyzing a sequence of n symbols, the entropy (H) is the negative sum of the probabilities of occurrence of a symbol p_i multiplied by the binomial ($\log_2 p_i$). To clarify the concept here there are some illustrative examples. Considering the tosses of a coin "fair", the odds for heads and tails would be the same (50%) and by applying the Shannon formula, we found Equation 2.3 would have:

$$- ((0.5) (-1) + (0.5) (-1)) = 1 \text{ bit}$$

Given a coin in which the occurrences of face were characterized by a probability of 75% of the crown and face occurred with 25% probability, then the entropy would be reduced:

$$- ((0.75) (-0.415) + (0.25) (-2)) = 0.81 \text{ bits}$$

In the case of DNA, considering the quaternary alphabet $\Sigma = \{a, c, t, g\}$, and given a random sequence, with no differences between the four bases probability we would have:

$$- ((0.25) (-2) + (0.25) (-2) + (0.25) (-2) + (0.25) (-2)) = 2 \text{ bits}$$

Whereas the probability of A or T is 90%, compared with only 10%probability of C or G, then the entropy would be reduced to:

$$- (2(0.45) (-1.15) + 2(0.05) (-4.32)) = 1.47 \text{ bits}$$

Well, in natural DNA sequences, the probability of occurrence of the bases are very similar, in fact, the differences between them usually do not exceed 5%, so we understand the high degree of entropy and therefore the associated high degree of difficult to perform compression.

2.9.2 Algorithmic Complexity, Kolmogorov Entropy

Entropy, Kolmogorov's vision, understood as the algorithmic entropy of a finite object is defined as the size of the smallest computer program that will generate completely that object [149]. Thus, given a string x of length l and a universal Turing machine U, using the program p, the complexity K is given by:

$$K(x) = \min\{l(p) : U(p) = x\} \qquad (2.4)$$

This methodology for the evaluation of entropy is necessarily dependent on the evolution of computers and programming languages of the same. *K* is incomputable and its determination can only be approximated. It is standard to evaluate this approach by using algorithmic entropy contributions of the best compression tools available. Only the upper limit is possible to determine the value of *K*, using the following expression for this:

$$K(n) \underset{-k}{\overset{+}{\leq}} \log_2 n + 2\log_2 \log n \qquad (2.5)$$

The DNA sequences are capable of analysis using the theory of Kolmogorov complexity [150]. Studies of similarity analysis for the determination of ortólogas1 sequences or phylogenetic correlations showed that the measurement and comparison of entropy is a factor to determine valid and robust similarity [151,152].

In short, analysis of DNA sequences depend on either from the perspective of information theory by Shannon or algorithmic information theory of Kolmogorov, which basically points to the same reality, only viewed from different angles. The first is the particular to the general and second in the reverse direction. It should be noted that the Kolmogorov theory still needs development and formalism than Shannon's theory, because of their greater antiquity has already achieved.

2.9.3 Estimation of entropy in DNA sequences

Although the concepts are often confused in the literature, there is conceptually a subtle difference between the program entropy estimator and the compressor. It is also common to estimate the entropy based on compressors but, as mentioned before, in the particular case of DNA sequences, some compressors exceed their representation in the maximum entropy of DNA code, so the estimate made in this way is misleading. The entropy estimator has no ultimate goal to produce a compact file. Thus, one can neglect the component coding of redundant elements, focusing on the discovery of these same elements redundant. Obviously, in scientific terms the estimation of entropy is the most important phase because of the compression provides the "raw material" needed

downstream. Upon completion of this phase, it uses a series of heuristics along with codes to recode the information in compact form.

We present the following references to some of the most representative studies that were conducted to evaluate and estimate the entropy of DNA sequences:

- In 1995, Faraj, M. et al. [134] proposed a new method to estimate the entropy of the DNA called the match length entropy estimator. This method was used to compare the differences between exons and introns, and contrary to expectations, faced with the facts that the entropy of exons was 73% of times greater than that of introns and that the variability of introns was 80% of times greater than that of exons.
- In 1999, Lowenstern, D. and Yianilos, P. [133] created the program to estimate CDNA the entropy of a DNA sequence. The basic observation used by them was to DNA sequences that contain more repetitions approximate (near repeats) of that would normally be expected. The CDNA uses two parameters to capture the approximate repetitions: (w) the length of the substring and (h) the Hamming distance [158].
- In 2000, Lanctot, K., Li, M., and Yang, EH [148] based on the ideas of Kieffer and Yang [154], published shortly before, based on codes in grammars recognizing repetitions and reversals complementary DNA sequences, developed a program to estimate the entropy of DNA sequences called GTAC (Grammar Transform Analysis and Compression).

In this work, greater emphasis is placed on the estimation of entropy to identify the reference of data set of DNA sequences. Chapter 4 describes our methodology developed for estimation and evaluation of the entropy for coding of genomic sequences that based on the minimum entropy. One of our goals is to calculate entropy in a useful way regarding two sequences and their corresponding species to serve in reduction compression ratio and to be used as criterion for identifying the reference sequence to other sequences. By using reference, we can compress the differences between reference and each sequence in data set instead of compress the individual sequence in data set.

Chapter 3

Analysis of Compression Methods for Genomic Sequence

In this chapter, the methodology of data compression will be presented as in general form and it discusses the methods of data compression that refer to the identification and reduction redundancy. The first part, the identification is as important as the second, which corresponding to the reduced coding. It also discusses the compression methods most suited to deal with the specificity of the genomic alphabet of DNA.

3.1 Introduction to Data Compression

Compression is used just about everywhere. All the images that get on the web are compressed, typically in the JPEG or GIF formats, most modems use compression, HDTV will be compressed using MPEG-2, and several file systems automatically compress files when stored, and the rest of us do it by hand. The neat thing about compression, as with the other topics we will cover in this thesis, is that the algorithms used in the real world make heavy use of a wide set of algorithmic tools, including

sorting, hash tables, tries, and probability coding. Furthermore, algorithms with strong theoretical foundations play a critical role in real-world applications.

The task of compression consists of two components, an *encoding* algorithm that takes input data and generates a "compressed" representation (hopefully with fewer bits), and a *decoding* algorithm that reconstructs the original data or some approximation of it from the compressed representation [18]. These two components are typically intricately tied together since they both have to understand the shared compressed representation as in Figure 3.1.

Figure 3.1: A general data compression scheme

In generally, compression algorithms divided into two algorithms: *lossless algorithms*, which can reconstruct the original data exactly from the compressed data, and *lossy algorithms*, which can only reconstruct an approximation of the original data. Lossless algorithms are typically used for text, and lossy for images and sound where a little bit of loss in resolution is often undetectable, or at least acceptable. Lossy is used in an abstract sense, however, and does not mean random lost pixels, but instead means loss of a quantity such as a frequency component, or perhaps loss of noise. For example, one might think that lossy text compression would be unacceptable because they are imagining missing or switched characters. Consider instead a system that reworded sentences into a more standard form, or replaced words with synonyms so that the file can be better compressed. Technically the compression would be lossy since the text has changed, but the "meaning" and clarity of the data might be fully maintained, or even improved. In this work, we deal with lossless algorithms for DNA compression in order to the importance of genomic information.

Some examples of lossless data compression include:

- Burrows-Wheeler Transform [155],

- Prediction by Partial Matching (also known as PPM)[27],
- Dictionary Coders (LZ77 & LZ78 and LZW) [25,156],
- Dynamic Markov Compression (DMC) [157],
- Run-length encoding and context mixing [158].

Examples of lossy data compression include:

- Discrete Cosine Transform[159],
- Fractal compression [160],
- Wavelet compression [161].

Because one can't hope to compress everything, all compression algorithms must assume that there is some bias on the input data so that some inputs are more likely than others, i.e. that there is some unbalanced probability distribution over the possible data. Most compression algorithms base this "bias" on the structure of the data – i.e., an assumption that repeated characters are more likely than random characters, or that large white patches occur in "typical" images. Compression is therefore all about probability.

When discussing compression algorithms it is important to make a distinction between two components: the model and the coder. The *model* component somehow captures the probability distribution of the data by knowing or discovering something about the structure of the input. The *coder* component then takes advantage of the probability biases generated in the model to generate codes. It does this by effectively lengthening low probability data and shortening high-probability data. A model, for example, might have a generic "understanding" of human faces knowing that some "faces" are more likely than others (*e.g.*, a teapot would not be a very likely face). The coder would then be able to send shorter data for objects that look like faces. This could work well for compressing teleconference calls. The models in most current real-world compression algorithms, however, are not so sophisticated, and use more mundane measures such as repeated patterns in text. Although there are many different ways to design the model component of compression algorithms and a huge range of levels of sophistication, the coder components tend to be quite generic—in current algorithms are almost exclusively based on either Huffman [19] or arithmetic codes [20]. It known that

information theory [132] is the glue that ties the model and coder components together. In particular it gives a very nice theory about how probabilities are related to information content and code length.

3.1.1 Measuring Compression

An important question about compression algorithms is how one judges the quality of one versus another. In the case of lossless compression there are several criteria which are the time to compress, the time to reconstruct, and the size of the compressed data. In the case of lossy compression the judgment is further complicated since we also have to worry about how good the lossy approximation is. There are typically tradeoffs between the amount of compression, the run time, and the quality of the reconstruction. Depending on application one might be more important than another and one would want to pick algorithm appropriately.

Also, there are different ways of knowing the compression achieved by a given algorithm between data and a compressed version of data such as:

1- Compression ratio that equals (compressed_length / original_length) * 100, for example if we had a file with a size of 400 bytes and compress it down to 100 bytes, the ratio will be (100/400)*100 = 25% so the output compressed file is only 25% of the original. However there's other method: (1-(compressed_length / original_length)) * 100. In this case the ratio is 75% meaning that we have subtracted 75% of the original file.

2- Bits per Byte (bpb): the formula is (compressed_length / original_length) * 8. In the previous example (100/400)*8 = 2bpb, meaning that we need only 2 bits to represent one byte. It's also accurate enough: (47/134)*8 = 2.805970149254 bpb, usually we only get three digits, like 2.806 bpb. Note that when we know the expected compression of a given algorithm we can know the supposed length of the output: (input_length / 8) * bpb = output length.

3- Bit per Char, bpc, in some cases is the same as bpb, and when specified, it refers to a given alphabet, which usually only contains characters that appear in text data. (but then this is not suitable for compressing binary data). In compression we code a file, and if the coded file is smaller than the original file we have achieved compression, otherwise

we have expanded it. For image files one can use the term of bpp, which means Bits per Pixel.

4- There are also some measures for speed. Like kbyps, Kilo Bytes per Second, it computes with: file_length / seconds. Obviously the file length must be in kilobytes (divide its size in bytes by 1024) and seconds should have more precision than a second, using hundredths of seconds is enough. Unfortunately this has a big problem; a program runs faster in a faster computer. Obviously we can't use the same computer for testing all the programs.

3.1.2 Prediction

Compression can be achieved via effective prediction of the future. While it is often impossible to tell with certainty what the next symbol in a data will be, an encoder can assign probabilities to each symbol in the source alphabet based on the earlier symbols in data. Once the probabilities have been calculated, the encoder then looks at the actual symbol and encodes it using $-\log_2 p$ bits, where p is the previously calculated probability of the symbol's occurrence.

This raises the question of how to calculate the probabilities. Assuming that there is no a *priori* knowledge of the composition of data, the only information that can be used is the previously encoded symbols. This restriction is made with the decoding process in mind: at any given point in decoding, the only information that the decoder has about the data is that which it has already decoded. It is possible to give the decoder additional information about data by transmitting it before data proper as a form of preamble, but this extra information bears its own cost and often does not yield enough benefit to pay for itself. In that case, providing the extra information leads to *worse* compression and is an indication that it is a suboptimal description of the data.

The simplest prediction model assigns a uniform probability to all source symbols under all circumstances. Since it makes no use of any information in data (or a *priori* information) to make predictions, it cannot achieve any compression. A small improvement over this model is to use the frequency of the previous occurrences of each

symbol to calculate the probabilities. It is necessary to use only the previous occurrences so that the decoder has enough to knowledge to repeat the calculations.

For example, let the previous part of the data be "DATA COMP". The symbol "A" has occurred twice, and the symbols "D", "T", *space*, "C", "O", "M" and "P" have each occurred once. Based on these counts, the predictor would assign a probability for the next symbol of 2/9 to "A" and 1/9 to the other seven symbols. The problem of assigning some probability to other symbols that have not yet occurred can be solved in several ways. It can be avoided altogether by starting each frequency count at one instead of at zero, or a special escape symbol (which requires its own code) can be used to indicate a novel symbol.

A Markov model [157] is a slightly more complex prediction model. It is based on the premise that the future can be predicted using only the recent context. At its simplest level, the Markov model uses the previous symbol to predict the next one. In other words, the model asks "Given that the previous symbol was X, what is the probability that the next symbol will be Y?" To calculate the probability requires a two-dimensional table storing a frequency count for every pair of symbols in data. The table can be recomputed by the encoder and transmitted before the data or computed adaptively as the data is sent. The concept of the Markov model can be extended to more than one previous symbol. In the case of an order-2 Markov model, the model asks "Given that the previous *two* symbols were W and X, what is the probability that the next symbol will be Y?". This can be extended to any order N.

For a non-random data of sufficient length, a higher-order Markov model will generally achieve greater compression than a lower-order one. A higher-order model can take advantage of structure in the data over longer sections — in the case of English text, a higher-order model might be able to see patterns in words that are commonly juxtaposed rather as well as the patterns in consecutive letters that a lower-order model would capture. However, the benefits of higher orders carry limitations.

Firstly, a higher-order model requires more temporary storage. For an order-N model with a source alphabet of size a, the storage requirement for the model is $O\ (a^{N+1})$. Secondly, with any adaptive model there is an initial "stabilizing" period during which the model accumulates enough information to be useful. During this time, most of the N + 1 length strings will not have been encountered and the predictions the model gives are likely to be very coarse and inefficient to code. With higher-order models, this stabilizing period is increased exponentially.

Linguistic text data inherently contains patterns and lends itself to compression by Markov-style prediction. In order to gain some of the advantages of higher-order Markov models while avoiding the initial time when most of the contexts have not been seen, Cleary and Witten [27] proposed a form of variable-order model called PPM. The model is variable-order in the sense that the length of the context used is determined by how much data is available for the current context. In the PPM scheme a maximum order N is specified rather than a fixed order. When a symbol is to be predicted, the maximum order (a context of N + 1 symbols) is tried first. If no such context has been encountered yet, an escape code is transmitted and the next shorter context (N symbols) is used. If this string has also not occurred, again an escape symbol is sent and the order decreased. This process continues until the context is a single symbol, whereupon the marginal probability of the symbol is used. PPM [27] is an improvement over basic Markov models, but it still bears some of the Markov model's drawbacks. Its storage requirement remains $O\ (a^{N+1})$ and would perform poorly on data where many distinct long contexts occur and few of them are repeated. Its characteristics are suited to more repetitive data.

3.1.3 Copying

Another mechanism for obtaining compression is by taking advantage of duplication within the data. If the encoder is midway through data and can recognize that the section of the data it is about to process is a replication of a section it has previously transmitted, it can inform the decoder that the next few characters are merely a copy of those occurring earlier. The contents of the sequence need not be transmitted at all as the decoder, given the starting position and length of the repeated section, can derive them.

One such compression scheme is the algorithm presented in [25]. This algorithm reduces data into a series of blocks, where every block is the concatenation of a copied section from earlier in the data and an additional character. To find earlier copies, a fixed size buffer is used. This makes the algorithm practical in running time as its search for a repeat is bounded. It also gives some guarantee of efficiency by avoiding a large position value, the encoding of which may outweigh the gain obtained by not stating explicitly the block's contents. However, the fixed size buffer precludes the recognition of repeats from much earlier in the data and also disallows the efficient handling of repeats that are longer than the buffer size. Like the PPM scheme described above, this algorithm is suited to data that contains frequently occurring short patterns as are found in text documents. This algorithm is also unsuitable for biological data because it looks for exact repeats.

The algorithm presented by Hategan and Tabus [162] employs a form of copying to compress protein sequences. It divides a sequence into fixed size blocks and attempts to match a block to an earlier block that is similar enough to be of use. To be deemed similar, two blocks must have a certain number of matching amino acids — that is, the same amino acid occurring in the same position relative to the beginning of the block. During the encoding process, each block is encoded as either a match of a previous block or a non-match. For the case of the non-match, the contents of the block are encoded using arithmetic encoding [20]. In the case of a match, the block is encoded by the number of the block that it matches and the relationship between the two blocks. That relationship is stated character-by-character, with every amino acid having a probability

of transforming into another amino acid or remaining unchanged. The probability of remaining unchanged will be quite high and consequently blocks with many matches will be encoded cheaply.

3.1.4 Finding Repeats

To take full advantage of repeated sections in a sequence one requires a fast way to recognize repeats no matter how distant they are. Much work has been done in the area of string matching. Performing a naive character-by-character comparison over a data of length n is at worst $O(n^3)$, but with some preprocessing of the data this can be improved. Suffix trees can reduce the time required for a single lookup to $O(|w|)$ (where w is the word to be searched for) with a pre-processing cost proportional to the length of the text [163]. The on-line construction algorithm presented in by Ukkonen [163] could be adapted to finding a repeat of a substring from earlier in the string. Suffix arrays [164] can also be constructed in linear time [165] and offer similar advantages to suffix trees, but are not as suited to incremental construction due to cost of inserting a new element into an array.

A simple and practical method of detecting repeated substrings is fingerprinting. The text is processed by applying a hash function to every substring of a fixed length and the results stored in a table. Subsequently, when the encoder is to encode some section of the data, it can consult the table in constant time to determine whether the following few characters have occurred previously. This method has been used for DNA compression [167] and a similar approach applied to finding homologies in large databases [167]. Although this method is fast and is able to find distant repeats, it cannot detect repeats that are shorter in length than the fixed hashing size. Conversely, if the hashing size is too low, too many false positives will be found. Approximate repeats may not be detected, depending on the length of the repeat, the hashing size and the amount of corruption in the replication.

3.2 Methodologies of similarity within and between genomic sequences

In the context of bioinformatics, homology and similarity are not synonymous. The analysis of homology occupies essentially functional structures for different genomes. On the other hand, the similarity between two sequences does not contain only the functional or evolutionary nature, is a broader perspective, going beyond the motifs or genes, comparing completely, an informational point of view, regions of a sequence or several sequences. In this section, it is primarily to study the intra-similarity sequences in order to comprehensively identify any replication either exact or approximate, and the repetitive segments of DNA without restrictions or limitations imposed by the functional logic of molecular biology. The major objective is considering the desirability of biological information by compression of dictionary, more advantageous to build the dictionary as possible to maximize compression, abstracting from the specificity of information and given only to the theory of abstract information. Sure, some of the patterns which will appear in the dictionary may in the light of current genomic knowledge does not receive a functional reading, but in future, with future discoveries, the percentage of random patterns will be considered small.

Although genome sequences are replicated and passed on to subsequent generations with high fidelity, errors and mutations are occurred which prevent the accurate copies for the good (evolution) and bad (diseases). Thus, the similarity between the sequences should be understood by the idea of proximity, which follows the three types:

- Positivist: the distance between two sequences is always greater than or equal to zero (will be zero if they are equal);
- Symmetry: the distance from sequence x to sequence y is the same as that between y and x;
- Triangle inequality: if two sequences are similar to a third, so can not be dissimilar to each other.

The analysis of similarity and alignment within and between genomic is important in bioinformatics [168]. Also, it corresponds to a complex problem that motivates a growing community of researchers for finding solutions with better performance, and space time. It can be inferred with greater speed and accuracy in DNA homologies of the post-genomic era.

Dynamic programming, including the Smith-Waterman algorithm [120] provides an exact algorithmic solution to the problem of determining optimal local alignments. However, the quadratic complexity ($O(n^2)$) associated with the creation of heuristics quickly, and concessions at the level of sensitivity is right, but also guaranteeing the approximate solutions in most cases. The key idea to reduce the information to rationally analyze these sequences is filtering; that consists of refining the regions that show signs of similarity, or hits, which correspond to repetitions of a primary standard. These primary standards, usually of small size (up to 25 nucleotides), commonly called seeds, repeated inter-and intra-genomic. The recollection of these seeds should be performed with selection criteria so as to not incurring redundant hits. At a later stage these seeds can be confirmed or not as revealing more or less extensive regions of similarity. The various methods of research and selection of seeds [169], as well as configuration templates and the seeds of their own, are distinctive elements of the analysis algorithms that use sequence alignments filtering. The methods implemented in BLAST [121], PatternHunter [170,171] and LAGAN [172] are relevant examples of sequence analysis algorithms with heuristic-based filtering.

The term that includes the evaluation sensitivity of the alignments detected by heuristics and compared to optimal. If a particular method achieves a sensitivity of 70% corresponds to 70% of which identifies similar regions actually exist, it backed by a comprehensive or optimal algorithm (which identifies 100%) based on comprehensive analysis of the sequence using dynamic programming.

3.3 Revision of Suitable Methodologies for Bioinformatics Compression

In this section we review the ways in which ideas and approaches fundamental to the theory and practice of data compression have been used in the area of bioinformatics [173]. Shannon information theory and biology have a long and, at times, controversial relationship [174,139]. As well presented by Godfrey-Smith and Sterelny[175], that seems to be mostly due to the expectation one has about the word 'information' in a biological system as opposed to in a signal transmitting one. For many years, the most successful use of information theory methods has been for sequence analysis and, in particular, to measure the 'deviation from randomness' of nucleotide and amino acid sequences (Konopka [176]). As Konopka points out, information theory does not seem to be essential for that task, because it could be performed by other means. Yet, given the actual deluge of data and the shift towards system wide views of biology, it is argued that information theory may offer some advantages for data analysis over traditional statistical methods (Rissanen *et al.*[177]). While those issues are being debated, it is worth recalling that textual data compression, one of the quintessential contributions of information theory to science and technology, has been found to be fundamentally connected to classification, statistics and various notions of sequence complexity [132,25,149] all crucial for bioinformatics. Despite those connections, the applicability of data compression to tasks in the computational biology sciences has been somewhat underestimated, probably due to the dispersion of results over conferences and journals catering to different scientific communities. One of the major conclusions stemming from this effort is the pervasiveness of data compression in computational biology and the associated techniques.

In fact, we found 10 areas of relevance for computational biology, in which data compression techniques have either resulted in the development of top-ranking methods or have been the fulcrum for major theoretic break-through, which need to be duly followed by additional work to result in valuable tools.

1) Compression for storage of biological sequences (e.g. substitutional–statistical methods, transformational methods, and grammar-based methods)[21,22,23]
2) Entropy Estimators [133,134]

3) String matching primitives [27,163]
4) Probabilistic suffix trees as optimal classification devices[164]
5) Towards parameter-free classification and data mining in biological sequences [178].
6) Clustering and indexing of microarrys [179]
7) Speed-ups of dynamic programming recurrences: Alignments [121] and HMMS [50].
8) Segmentation of biological sequences [180].
9) Pattern discovery [181]
10) Compression and inference of biological networks[182]

We will explain some basic theoretical ideas from data compression, such as the notions of entropy, mutual information, and grammar have been more used for analyzing biological sequences in order to infer phylogenetic relationships between organisms and study viral populations.

3.3.1 DNA Logo

Given that DNA is a means of transmitting information between generations it was natural that the concepts of entropy and information be applied to understanding DNA. One of the first contributions to the use of the concept of information in bioinformatics was the work of Lila Gatlin [139] in the 1960's. Gatlin proposed a definition for the information content of DNA which was essentially a measure of the divergence of the DNA sequence from an *iid* sequence. Given an alphabet of size N, where N is four for DNA sequences and 20 for amino acid sequences, Gatlin defined two quantities D_1 and D_2 which measured divergence from the probable state and divergence from independence, respectively,

$$D_1 = \log N - H_1(X)$$
$$D_2 = H_1(X) - H(X|Y)$$

where $H_1(X)$ is the first order entropy of the sequence and $H(X|Y)$ is the conditional entropy where X and Y are neighboring elements in the sequence. The information content of a DNA sequence is then defined as the sum of these two measures of

divergence which can be shown to be the difference between the maximal entropy *log N* and the conditional entropy $H(X|Y)$. Gatlin connects this definition of information to redundancy by noting that defining redundancy as [152].

$$R = 1 - \frac{H(X|Y)}{\log N}$$

results in $$R \log N = D_1 + D_2$$

Based on the limited data available at that time, Gatlin showed empirically that DNA from vertebrates, bacteria, and phage (viruses that prey on bacteria) can be distinguished by looking at their information content and that there is increasing redundancy in the DNA of organisms as we move from lower complexity organisms like bacteria to higher complexity organisms such as vertebrates. Plotting the data available to her as shown in Figure 3.2 it is easy to see why she would come to that conclusion.

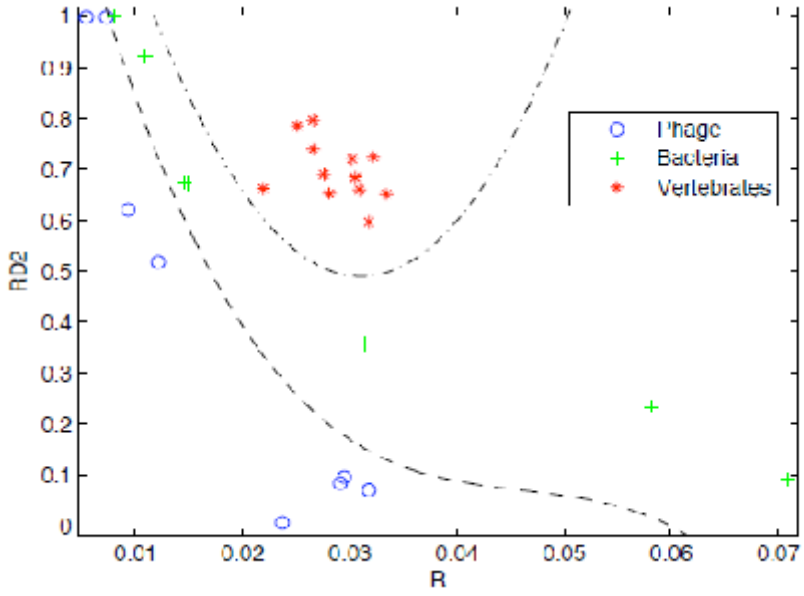


Figure 3.2: Plot of the redundancy rate versus $\frac{D_2}{D_1 + D_2}$ using the genes available at the time shows a clear segregation of Phage, Bacteria, and Vertebrate sequences.

However, if we add more sequences from these groups the clear segregation breaks down as shown in Figure 3.3. Clearly first order entropy is not sufficient to

capture the complexity of DNA and provide a differentiation between simpler and more highly complex life forms.

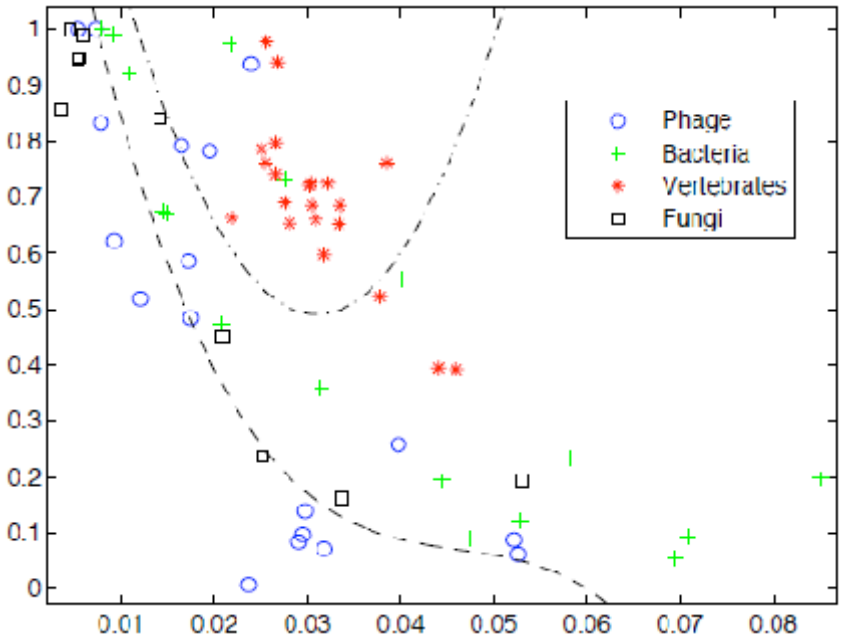


Figure 3.3: Inclusion of additional sequences breaks down the segregation observed by Gatlin

While Gatlin's work seems to have had more impact in the philosophical realm [183] the work of Stormo and colleagues [184] and Schneider and colleagues [185–187] has had more direct impact on the practice of bioinformatics. Gatlin assumed that the DNA sequence was a realization of an ergodic process and estimated probabilities along a sequence. Schneider et al. [185] align multiple DNA sequences and treat them as realizations of a random process. They then compute the information content per base (or residue for amino acid sequences). The definition of information content by Schneider and Stephens [186] is only slightly different from that of Gatlin's D_l parameter. Schneider and Stephens define the information at a location l by:

$$R_{sequence}(l) = \log_2 N - (H(l) + e(n))$$

where H(l) is the estimate of the first order entropy given by

$$H(l) = -\sum f(x,l)\log_2 f(x,l)$$

e(n) is a correction term employed to account for the small number of sequences used to estimate the entropy, and $f(x, l)$ is the frequency of occurrence of base (or residue) x at location l. Based on Schneider and Stephens, this value creates a *logo* for the sequences. At each position the elements of the aligned sequences that appear at that location are represented by a letter the height of which is proportional to its frequency of occurrence at that location multiplied by the information at that location $R_{sequence}(l)$. The various letters are stacked with the order of the stacking dictated by the frequency of occurrence. The total height of the stack at each position l is given by $R_{sequence}(l)$.

The *logo* as created by Schneider and Stephens is widely used in various bioinformatics applications [188,189] as they provide information useful to biologists. A tall stack of letters at a particular location implies that the corresponding site is important for some reason. Typically, the location is a binding site for proteins involved in the regulation of expression of genes. These sites are thus connection points of gene regulation networks. The height of individual letters in the stack shows the frequency of occurrence of the letter and so can inform a biologist of the level of mutation at this particular site of the DNA molecule. An example of a *logo* created by aligning a number of different sequences from E. coli at the beginning of a gene is shown in Figure 3.4. The sequence ATG pops out immediately in the logo alerting the user to its importance. Notice that while T and G are present in all sequences, in some of the sequences aligned here the first element of the start codon is a G or, more rarely, T rather than an A. Looking upstream (left) from the start codon we can see another region of consensus. This corresponds to the regulatory site known as the -10 promoter region. This region helps direct the RNA polymerase enzyme which makes RNA copies of the gene.

Schneider and colleagues in other work have used logos for studying DNA protein interaction [190], to investigate variants [191] and look for novel genes [181]. The utility of the entropy concept in such a wide range of application suggests that perhaps these concepts are natural to biological sequences and a more wide-ranging and deeper analysis would be a fruitful endeavor.

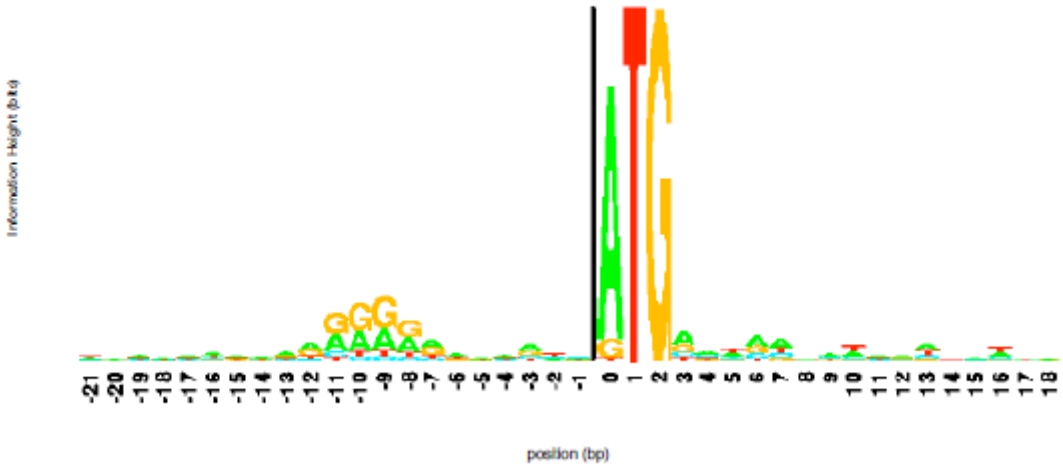


Figure 3.4: The logo of a number of sequences at the beginning of a gene. The start codon ATG is immediately apparent. The logo was constructed using the software at http://weblogo.threeplusone.com/.

3.3.2 Application of Average Mutual Information

What entropy is to lossless compression, the rate distortion function is to lossy compression. Both provide a bound on the efficacy of compression. The rate distortion function depends on two things, the distortion measure and average mutual information between the source coder input and the source coder output. The average mutual information between random variables X and Y is given by

$$I(X;Y) = H(X) - H(X|Y) = I(Y;X)$$

It is a measure of the information contained in the random variable X about the random variable Y and vice versa, and is a powerful tool for exploring the relationship between random processes. In some ways it can be thought of as a measure of correlation. This ability of the average mutual information to expose relatedness has been used in a number of ways in bioinformatics.

When dealing with viruses we often end up with populations that differ in small ways from one another. We can treat each individual clone from the population as the realization of a random process. If we do so we can treat each position in an aligned collection of these clones as a random variable. The average mutual information can then be used to identify bases in DNA sequences, or residues in amino acid sequences that in some way depend upon each other. This dependence can help us understand something about the two and three dimensional structure of the virus, or protein. Korber et al. [192] in a groundbreaking study identified correlated mutations in the envelope protein of the HIV-1 virus. The envelope protein of a virus is the face the virus presents to the host and the target for which the host fashions its assault. By mutating the envelope protein the virus tries to keep the host's defense off-balance. This effect is clearly seen in AMI charts developed by Sayood et al. [193] when studying the differences between infected infants who succumbed to HIV and those who did not [194,195]. The AMI chart is simply a representation of the average mutual information values in the form of a matrix where the $(i,j)^{th}$ pixel in the chart represents the average mutual information between the i^{th} residue and the j^{th} residue of the envelope protein. Figure 3.5 contains AMI charts representing the HIV population from infants who remained asymptomatic, while Figure 3.6 shows AMI charts representing viral populations for infants who succumbed to the disease.

The difference in the charts shows the differing characteristics of the populations with substantial levels of mutation continuing in those patients who remained asymptomatic an indication perhaps of continuing efforts of the viral population to overcome the host defenses.

There is great uniformity as well as divergence in life. Many organisms use the same proteins to perform the same functions. These proteins have very similar primary sequences though there might be slight variations. The study of these variations in the context of which organism the protein came from can help us understand something about the role of the proteins. These studies have used the average mutual information in order to study correlated mutations in proteins [196,197]. Correlations among residues

can be evidence of secondary structure. Adami [174] has used this fact to explore secondary structure in RNA and proteins.

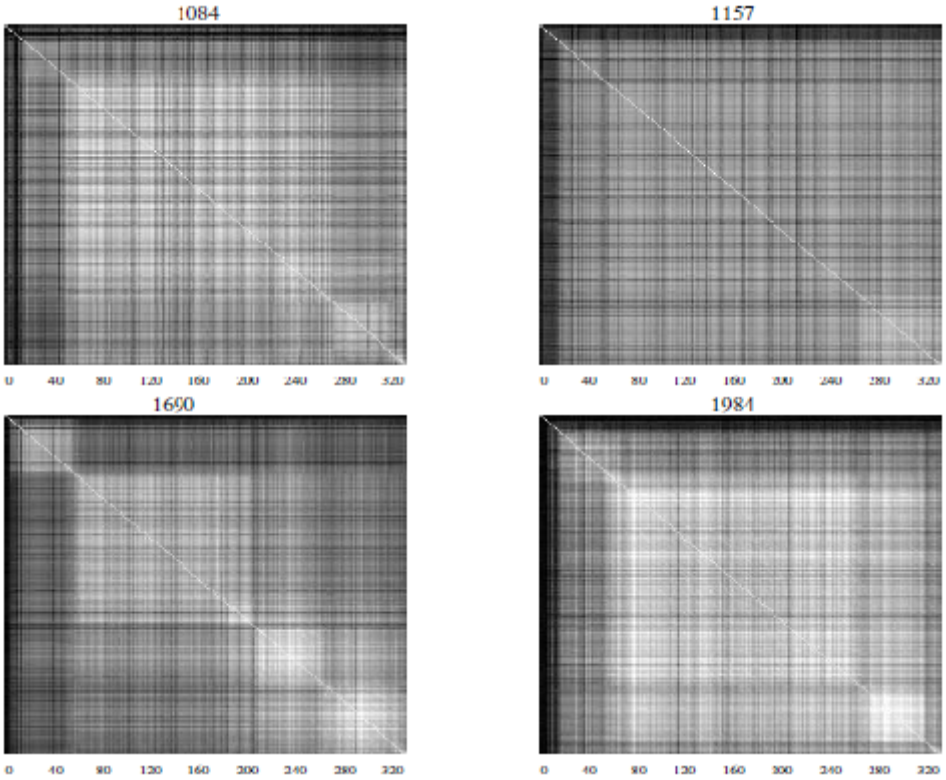


Figure 3.5: AMI charts for HIV-1 populations isolated from patients who remained asymptomatic. The large number of white pixels indicates generally a high degree of co-variation while "checkerboard" regions indicate specific segments of the envelope protein with correlated mutations [193].

The DNA of organisms contains regions which code for proteins and regions which do not. In eukaryotes these regions are interspersed in genes. Because the protein coding regions are interpreted in terms of triplets, one would expect a periodicity of three in the dependence between bases in coding regions, and no such periodicity in non-coding regions. This is indeed the case and the existence or non-existence of such periodicity in the average mutual information can be used to distinguish between coding and non-coding regions [198].

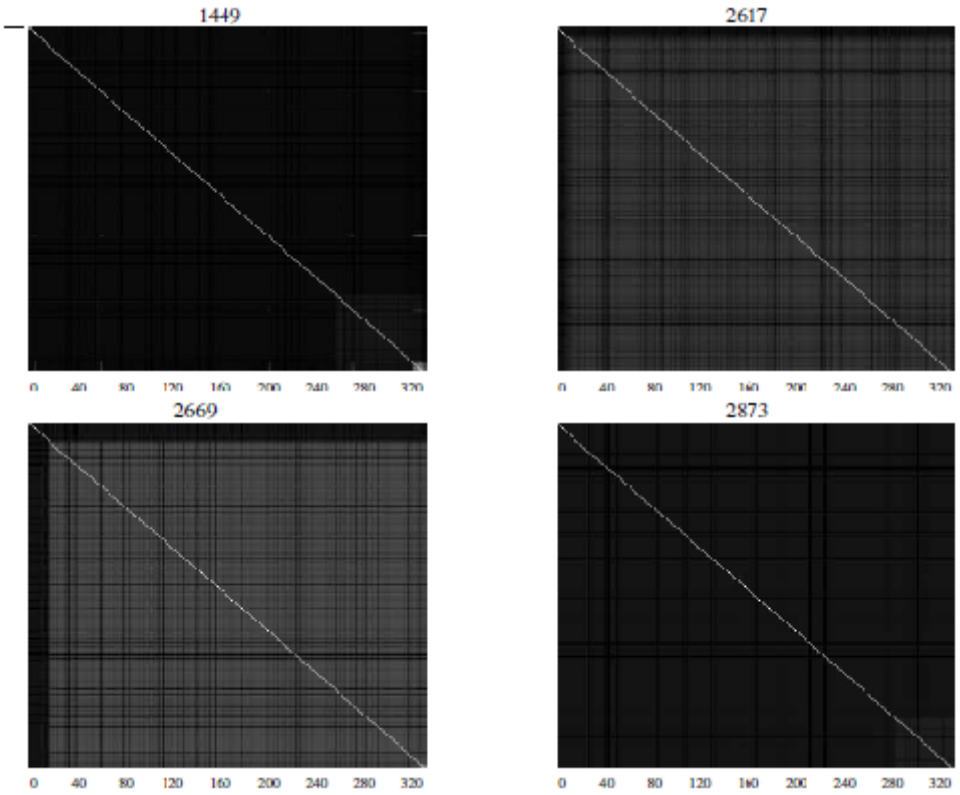


Figure 3.6: AMI charts for HIV-1 populations isolated from patients who succumbed to AIDS. The preponderance of black pixels indicates a relatively homogeneous population [193].

Instead of treating a large number of sequences as multiple realizations of a random process, we can also analyze long single sequences assuming ergodicity. By computing the average mutual information between bases that are k apart, $k = 1, 2, \ldots, N$ under the ergodic assumption, we can create an average mutual information profile for a given genomic sequence. Bauer and colleagues [199,200] have shown that these profiles can be used as a signature for the species from which the DNA was obtained. Thus, a mouse will have a different average mutual information profile than a human. Even for closely related organisms the average mutual information profile can be used to detect evolutionary relationships. Bauer and colleagues show that the average mutual information profile can be used to cluster subtypes of the HIV-1 virus. The fact that average mutual information profiles reflect evolutionary relationships has been demonstrated by Berryman et al. [201] who show how the average mutual information profile for a chromosome reflects events in the organisms evolutionary history. Holste et

al. [202] demonstrate the effect of evolutionary history on the average mutual information profile by showing the effect of evolutionary events on particular characteristics of the profile for various human chromosomes.

Much of the work cited above used relatively long sequences of DNA on the order of many thousands of bases. The average mutual information profile has also been shown to be useful when used with relatively short fragments of DNA. When obtaining the sequences of DNA one is faced with a conundrum. The DNA molecules we wish to sequence are quite long, consisting of millions of bases. The technology for obtaining sequences works only for fragments whose length is measured in the hundreds. The solution for this conundrum is often a method, called shotgun sequencing, in which multiple copies of the target DNA molecule are broken into random fragments that are small enough to be sequenced. Because multiple copies of the DNA molecule have been fragmented randomly many of these fragments will tend to overlap. The sequences of overlapping fragments can be put together to form much longer sequences called contigs. The process of finding overlapping fragments is a computationally expensive one. However, it can be simplified if we note that the average mutual information profiles of neighboring fragments are more similar than those of fragments further away. Otu et al. [3have used this fact to divide the fragments into clusters based on their average mutual information profile, using the Linde-Buzo-Gray [204] algorithm for clustering, thus considerably reducing the complexity, and increasing the accuracy of reassembly.

The concept of average mutual information has also been used to understand relationships between expression patterns of genes. A popular method for observing a cell in action is through the use of microarrays. By measuring the amount of messenger RNA present under different conditions these arrays indicate which genes are active in each condition and the level of activity. A gene seldom acts alone and in order to understand the causal relationships between genes it is important to find which genes behave similarly. A number of studies use average mutual information to obtain this clustering [205-207].

3.3.3 Grammar and Biology

The Lempel-Ziv algorithms, though usually not thought of that way, are examples of the use of a grammar for compression. The original concept behind abstract grammars is that a grammar G is meant to completely describe the underlying structure of a corpus of sequences. Because most naturally occurring sequences contain repetition and redundancy, grammars are often able to describe sequences efficiently. Hence, the usage of grammars can be thought of as a means to provide compression.

A block diagram describing how grammars can be used for compression is shown in Figure 3.7. The input to the encoder is a sequence w. The encoder first infers a grammar G specific to w. It should be noted that an orthodox linguist may not approve of the term "grammar" in the sense provided here, as G will derive the single string w and nothing else. However, time and engineering often find ways of modifying and applying existing ideas to new applications. Once the grammar is estimated, it is encoded, first into symbols then into bits, followed by storage or transmission. Upon reception, the bits are decoded into symbols and then into the inferred grammar G. Given this kind of grammar, it is a simple matter to recover w from G by beginning with the special "start symbol" S, which is part of G. In a seminal paper Kieffer and Yang [155] showed that a grammar based source code is a universal code with respect to finite state sources over a finite alphabet.

Among many other linguistic innovations, Noam Chomsky defined four categories for types of grammars in [208] and elaborated upon in [209]. The language levels are contained in terms of complexity as $3 \subset 2 \subset 1 \subset 0$, where type-3 or regular grammars generate regular languages, type-2 or context-free grammars (CFGs) generate context-free languages, type-1 or context-sensitive grammars (CSGs) generate context-sensitive languages, and type-0 or unrestricted grammars generate recursively enumerable languages. All other languages can be classified between type-3 and type-0. Knowing the type-containment of a certain grammar is important in understanding the computational complexity necessary in solving linguistic problems.

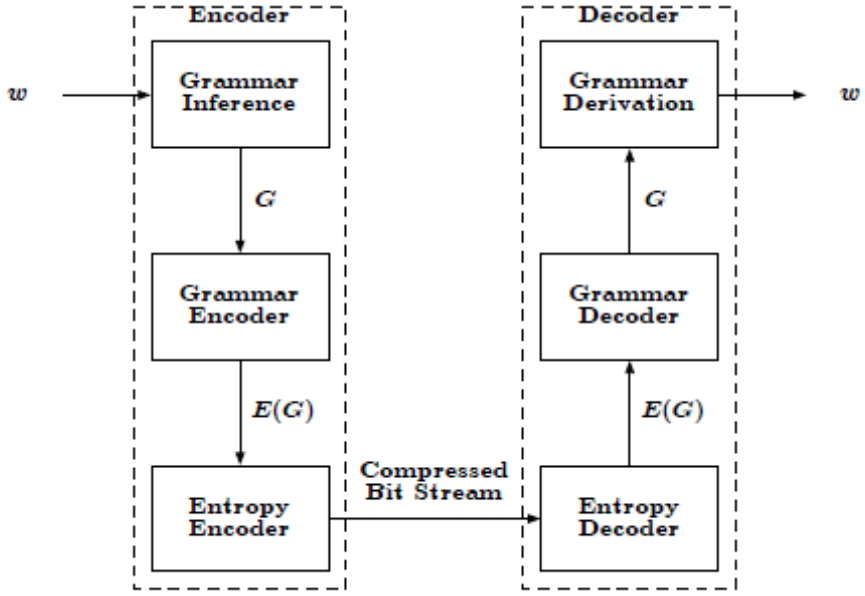


Figure 3.7: A block diagram depicting the basic steps involved with a grammar-based compression scheme.

Identification of function and/or meaning of segments of biological sequences remains an ongoing and active area of research. This means studying primary structure, or the sequential ordering, of sequences and the secondary structure, or the three-dimensional shapes that form due to attractions that occur among separated segments within the sequences. A somewhat uncommon method for predicting RNA secondary structure focuses only on the information contained within the sequences. For example, [210] reviews many ways in which linguistics, specifically abstract grammars, may be used to model and analyze secondary structures found in RNA and protein sequences. Another example [211] includes RNA secondary structure prediction using stochastic context-free grammars (SCFGs).

Abstract grammars have been shown to be useful models of biological sequences at various levels of detail. Surveys presented in [212] and [213] describe correlations between linguistic structures and biological function. In particular, linguistic models of macromolecules [214, 215], have been used to model nucleic acid structure [216], protein linguistics [217], and gene regulation [218]. Much of the work available in the literature

assumes the underlying grammar is known a priori. Hence, there is a need for general methods to infer grammars efficiently from biological structures.

In [219] a general algorithm is presented for inferring sequential structure in the form of CFGs for generic inputs including biological data. Two other algorithms in which sets of arbitrary sequential data are categorized to generate a CFG are presented in [220] and [221]. One drawback with these algorithms is the inability to make use of domain knowledge, although [219] discusses the improvement available when domain knowledge is applied. In fact, the algorithm was modified in [222] to operate specifically on DNA and makes use of the Chargaff base pairing rules to generate a more compact model.

The most commonly known and recognized application of grammars to computational biology are in the form of SCFGs used to search for the most likely secondary structures in RNA leading to the identification of mechanistic elements that control various aspects of regulation [211]. Another application has found use in multiple sequence alignment [223], where a simple regular grammar was inferred and used as an information-theoretic metric in determining distance between organisms. The remaining primary usage of grammars is in a data-mining paradigm, where grammars are used to efficiently scan databases full of experimental data from the literature (e.g., RegulonDB).

More interestingly, some work has briefly been done in regards to modeling Genetic Regulation Networks (GRNs) using a subclass of context-sensitive grammars [212] called definite clause grammars (DCGs) developed in the efficient language, Prolog. This was further developed into Basic Gene Grammars in [224]. The end result is a very high-level model description with a database approach to determining the classification of sequences of data in *silico*.

3.4 Developments of Genomic Sequence Compression

The advent of the sequencing method enabled genomic sequence data to be collected and manipulated on computers, paving the way for explosive growth in the new field of bioinformatics. Publicly available DNA sequence databases such as GenBank play a crucial role in collecting and disseminating the raw data needed by researchers in the field. This database currently contains 168 Gb of sequence data[1], and is expected to continue to grow at an exponential rate, doubling in size roughly every 14 months[2]. The volume of data being dealt with now presents serious storage and data communications problems. Currently, sequence data is usually kept in large "flat files," which are then compressed using standard Lempel-Ziv compression [25] (e.g. with gzip[3]). Unfortunately this approach rarely achieves good compression ratios: typically, gzip fails to match the "compression" afforded by simply encoding each base using 2 bits [26].

Previous work concerning the compression of genomic (DNA or protein) sequences can be divided into two categories: techniques developed for efficiently compressing sequence data for the sake of reduced resource consumption (disk space or network usage) [21,28,133,225-227]; and investigations of the usefulness of compressibility as a measure of information content, for the purpose of making inferences about sequences (such as the relatedness of two sequences) [228,229]. Examining this former category of work reveals two distinct approaches:

- Compressing individual genomic sequences
- Compressing databases of genomic sequences

3.4.1 Compressing Individual Genomic Sequences

It is now widely recognized that DNA data is inherently difficult to compress below the level of 2 bits per base achievable through direct encoding [26, 223,224]. Much research has gone into developing algorithms for more effectively compressing individual DNA sequences.

---

[1] ftp://ftp.ncbi.nih.gov/genbank/release.notes/gb159.release.notes.
[2] http://www.ncbi.nlm.nih.gov/Web/Newsltr/V14N2/ 100gig.html.
[3] J. Gailly, M. Adler, “gzip (GNU zip) compression utility”, [http:// www.gnu.org/software/gzip/].

These include BioCompress [225], BC [21], GenCompress (GC) [230], the CTW+LZ algorithm [26], and DC [226]. BC detects exact repeats and complementary palindromes located in the target sequence and then encode them by repeat length and the position of a previous repeat occurrence. BC also uses arithmetic coding of order 2 if no significant repetition is found. BioCompress also uses the same methodology as in BC except that it does not use order-2 arithmetic coding. GC is a one-pass algorithm that search for approximate matches. This algorithm carefully finds the optimal prefix and uses order-2 arithmetic encoding whenever needed. GC also detects the approximate complemented palindrome in DNA sequences.

An efficient encoding scheme, *Cfact* [22] is brought forward by Rivals. It had a two pass technique, where the first pass, is used to detect exact repeats, and the second pass is used to encode the repeats. Regions with no repeats are coded at two bits per base. Though the encoding scheme is useful, the algorithm, which used this scheme, did lossy compression, which is of little use for DNA sequences.

CTW+LZ algorithm came about as a very efficient compression algorithm, but it has very slow execution time. This algorithm used CTW [231] method and local heuristics for resolving the Greedy Selection problem. Though this algorithm is not practically used because of the high time consumption, Lempel-Ziv algorithms have formed the crux of most Gene Compression algorithms. Perhaps the best of these is DC, which employs the PatternHunter [175] sequence search algorithm to discover patterns of approximate repeats or approximate palindromic repeats in sequence data. DC achieved compression averaging 13.7% on a sample set of DNA sequences and is substantially faster than earlier algorithms. Grumbach and Tahi [21] allude to a "vertical" mode of compression for compressing multiple sequences in a database; however they do not elaborate on how this might be accomplished.

The Sequitur used Digram Uniqueness and Rule Utility. Digram Uniqueness says no pair of adjacent symbols appear more than once in the grammar, while Rule Utility says that each rule is used at east twice(except for the start rule). The DNASequitur is an

improvement on the previous Grammar-based compression algorithm [222], and it used Reverse Compliments. The usual approach used by most algorithms is to find exact repeats and approximate repeats. The largest subset of compatible repeats is generated and encoded using algorithm-specific methods. Behzadi and Fessant [232] proposed algorithm that called DNAPack (DP). They find repeats to the cost of a dynamic programming search and select from a second order markov model, a context tree and two-bit coding for the non-repeated parts.

Compressing genomic sequences can be broadly classified as substitutional or statistical and transformational methods as the following:

3.4.1.1 Substitutional–Statistical methods

These class of methods combines the two most well-known and successful compression techniques: substitutional [233], and statistical [132]. Basically, the sequence to be compressed is partitioned into substrings, some of which are compressed well via substitutional methods, while the remaining ones are compressed well via statistical methods. A suitably defined gain function is used to establish the division of the substrings in the two groups. A substitutional model uses some form of pointer back to an earlier instance of a repeated subsequence to encode a later instance. On the other hand, a statistical model encodes the sequence element by element using a probability distribution over the possible values of the next element in the sequence. The distribution can be formed as a blend of opinions derived from the base distribution and from the length and fidelity of matches between recent history and earlier parts of the sequence. A statistical method can directly yield a per element information sequence, in addition to deriving a compressed encoded sequence. However, there is no simple natural way to derive a per element information sequence for a substitutional model. This paradigm has been initiated by Biocompress 1 and 2 [225,21] and offers a wide variety of methods with a range of appealing choices in terms of the trade-offs between compression and speed, e.g, Cfact [22], GenCompress [230] and its improved version DNACompress[226], DNAPack [232], and NMLComp [227]. Most of those methods use the peculiar nature of redundancy in biological sequences that presents itself under the form of reverse

complement matches and approximate repeats. To the best of our knowledge, XM [28] is the first pure statistical compression method for biological sequences. Following that general scheme, XM compresses each symbol in a sequence using arithmetic coding [20] and an adaptive model for symbol probability distribution. This distribution is computed and updated via a combination of 'expert' models, where each model specializes for a particular type of statistical information in the sequence and has been carefully designed on a sound biological hypothesis. To date based on experiments on benchmark data sets; XM seems to be the compression method of choice, both on DNA and proteins, guaranteeing improvements in both compression and running time. For instance, on a DNA corpus of sequences, the average compression ratio (bits per symbol) is 1.6940 as opposed to 1.7148 achieved by DNAPack, the best performing of the methods against which XM has been compared. Moreover, its performance compares favorably with the highly specialized method ProtComp for protein sequences, i.e. 3.9434 bits per symbol. In addition to its versatility in compressing biological sequences, XM offers the advantage of computing the information content of a sequence *per base*. In turn, that can be used to identify areas of interest, e.g. repeated subsequences or low complexity regions, as the authors demonstrate on the HUMHBB human gene. We anticipate that the identification of repetitions, 'unusual' subsequences and low complexity regions are recurring themes in the application of data compression techniques to the analysis of biological sequences.

3.4.1.2 Transformational methods

The Burrows–Wheeler transform [155] is the most well-known example in this class, where the sequence is subject to transformations before the actual compression takes place. Based on that transform, there are only two methods, variants of each other, which specialize in biological sequences (Adjeroh and Nan [234], Adjeroh et al., [235]).

The latest of the two has been a big step forward in protein sequence compression, yielding, also, novel insights into protein sequence structure on a genomic scale. In fact, applying their technique to several proteomes, Adjeroh and Nan [234] provide experimental evidence that redundancy in protein sequences is in the form of

repeated subsequences that are separated by thousands of symbols, e.g. 350 000 in one case for *Homo Sapiens*. This scale of redundancy has not been observed before, even with the use of computational methods. Although multiple gene copies and repeated histone clusters are known to be present in most eukaryotic genomes, their number and their sizes do not seem to be enough to explain such 'long range' correlations in protein sequences. Probably, lack of knowledge about sequence structure is the reason for the apparent incompressibility of protein sequences.

While these single-sequence algorithms are interesting from a theoretical point of view, and are certainly becoming increasingly practical in the modern world of genome scale analysis, a great deal of everyday bioinformatics work continues to entail the communication and storage of multiple sequences, and the modest compression gains afforded by these algorithms are ultimately not sufficient to justify their adoption for large databases.

3.4.2 Compressing Databases of Genomic Sequences

Strelets and Lim [236] describe a program, SAGITTARIUS, for compressing PIR-format [237] protein sequence databases. Their system uses standard dictionary-style compression of sequence entry metadata, and a novel alignment-based compression strategy for the protein sequence data itself. A small number of sequences are maintained in memory as the *reference sequence accumulator* and each sequence in the database is aligned with each sequence in this list. If any alignment produces a strong match, the input sequence is recoded using symbols describing insertions and deletions to enable recovery from its close match in the accumulator; otherwise, the sequence is output verbatim and added to the accumulator, overwriting the oldest incumbent sequence if the accumulator is full. Sequences to be output are compressed using run-length encoding and Huffman encoding, and the shorter of the two encodings is chosen. Thus the accumulator represents a window of recently encountered interesting sequences. The authors set the size of the accumulator at three sequences, and were able to achieve 2.50:1 compression, significantly better than PKZIP [238] at 2.13:1.

Strelets and Lim [236] were interested in producing a compressed database that could be used interactively in much the same way as the original database. This was facilitated in part by the fact that their approach never requires recursive decoding of sequences – each sequence is encoded in terms of at most one other sequence, which is itself available "as-is," (i.e. not compressed in terms of another sequence). While useful for interactive operations, it is clear that avoiding recursive encoding must limit the overall level of compression obtained. J White and Hendy [6] were targeting maximum compression coil differs from that of [236] in this respect. Another difficulty arises in the assumption that similar sequences are likely to appear near each other in the input file. This is crucial in order to be able to limit the size of the accumulator and thereby the runtime. The authors found that increasing the size of the accumulator past three sequences increased the runtime but made no substantial improvement in compression, which appeared to justify their assumption. Unfortunately, while this neat localization of similar sequences may have been true of the PIR database in 1995, it is certainly not true of the large nucleotide databases of today, and [6] chose not to make this assumption.

Li, Jaroszewski and Godzik have taken a similar approach to the related problem of producing non-redundant protein databases with their CD-HI [179] and CD-HIT [239] packages. More recently, Li and Godzik have extended this approach to DNA sequences with the cd-hit-est program [240]. Their main advance over [236] is in employing short word filters to rapidly determine that two sequences cannot be similar, which significantly reduces the number of full alignments necessary. Despite impressive speed on small-to-medium data sets, they report that clustering 6 billion ESTs at 95% similarity takes 139 hours [240]. The program nrdb[4] locates and removes exact duplicate sequences from a DNA database in FASTA format. While this program is clearly a step in the right direction, many sequences in a typical database are almost but not quite exact duplicates of other sequences (perhaps differing at one or two positions), highlighting opportunities for further improvements.

[4] http://blast.wustl.edu/pub/nrdb/

3.4.3 DNA Compression

To cite some specific examples of technologies, methodologies and algorithms used in compression of the DNA [26], presents the following list:

- Dictionary-based scheme, which compresses data by replacing long sequences by short pointer information to the same sequences in a dictionary such as dictionary based on repetitions, repetitions and quasi-palindromes by methods heuristics [226, 230]
- Based on probabilistic models of language, using probabilistic prediction with support or by using the figures of the experts 'experts' [28];
- Based on probabilistic Markov models [241];
- Using the Burrows-Wheeler transform [235];
- Using off line-greedy textual substitution [242];
- Based on exhaustive dictionary of patterns discovered by dynamic programming [240];
- Based on the model NML (Normalized Maximum Likelihood) [227, 243];
- Based on statistical methods, which compresses data by replacing a more popular symbol to a shorter code such as arithmetic coding and CTW [26];
- Based on grammars [154].

By way of review and refer matters to characterize more carefully the methods that are employed in work, more specifically, the dictionary compression, and the probabilistic prediction compression depend on arithmetic coding statistics.

In statistical methods, arithmetic coding [20] and CTW [26] are known to compress the DNA data well and Huffman coding [19] is known to compress not very efficiently. The Burrows-Wheeler transform (BWT, also called block-sorting compression) [155] is an algorithm used in data compression techniques such as bzip2. Both the LZ77 and LZ78 algorithms work on this principle. But these methods achieve a limited compression of DNA because of using directly to DNA (without modeling of DNA). Therefore, Specific compression algorithms have been proposed for DNA compression.

Specific compression algorithms have been proposed for the compression of DNA sequences by using particular characteristics in DNA sequences. One of these characteristics is the appearance of "reverse complements" In DNA, A and T are complements of each other; G and C are also complements of each other. The complement of the DNA sequence AAACGT would be TTTGCA. If we then reversed TTTCGA we would get ACGTTT. ACGTTT is defined as being the reverse complement of AAACGT. One method that takes advantage of this characteristic would store ACGTTT as an index to the position where AAACGT occurs in the string. Presumably, you would also need to store a bit indicating that it is a reverse complement.

Another characteristic that is employed is the occurrence of "approximate repeats" within a given DNA sequence. This means that there are many repeats within a given DNA sequence. For example, AAACGT may occur more than once in a given DNA sequence. However, there exists an "approximate repeat" of AAACGT in the DNA sequence as well. For example, AAACGT may have AAACGG as an approximate repeat because it differs by the last character in the sequence. Storing AAACGG would require us to index AAACGT. One method of indexing AAACGG would involve the following: if S = AAACGT then AAACGG = Substring (S, 0, 5) + G. Better results would appear for long strings that are very close approximate repeats.

The earliest special purpose DNA compression algorithm found in the literature is BioCompress [225] developed by (Grumbach et al., 1993). BioCompress detects an exact repeats and complementary palindromes in DNA, and then encodes them by the repeat length and the position of a previous repeat occurrence. If a subsequence is not a repeat, it is encoded by the naive 2 bits per symbol technique. The improved version, BC [21], improved by (Grumbach et al., 1994) and uses order-2 arithmetic coding (Arith-2) to encode non-repeat regions. Biocompress, and its second version, BC are similar to the Ziv-Lampel data compression method.

The Cfact [22] DNA compressor developed by (Rivals et al., 1996) also searches for the longest exact repeats but is a two-pass algorithm. It builds the suffix tree in the first pass. In the encoding phase, the repetitions are coded with guaranteed gain; otherwise, encoded by two-bit per base encoding will be used. This is similar to the codeword encoding condition in BC except that the order-2 arithmetic coding is not used in Cfact.

A substitution approach is used in GC [225] use approximate (and not exact) repetitions. It exists in two variants: GC-1 uses the Hamming distance (only substitutions) for the repeats while GC-2 uses the edition distance (deletion, insertion and substitution) for the encoding of the repeats. The GC achieves significantly higher compression ratios than both BC and Cfact. Most other compression algorithms employ similar techniques to GC to encode approximate repeats. They differ only in the encoding of non-repeat regions and in detecting repeats. In this algorithm, LZ77 scheme with reverse complement is introduced as a dictionary-based scheme.

Since almost all repeats in DNA are approximate, GC obtains better compression ratios than BC and Cfact. The same compression technique Ziv- Lampel [25] is used in the DC [226] algorithm by (Chen et al., 2002), which finds approximate repeats including complemented palindromes in one pass; using a specific software, PatternHunter [170] and encodes approximate repeat regions and non-repeat regions in another pass. In practice, the execution time of DC is much less than GC.

At the cost of time complexity, DP by (Behzadi et al., 2005) [232], uses Hamming distance for the repeats and complementary palindromes. Non-repeat regions are encoded by the best choice from an Arith-2, context tree weighting, and naive 2 bits per symbol methods. Unlike the above algorithms, DP does not find the repeats by a greedy algorithm, but uses a dynamic programming approach instead. It has good sensitivity in detecting patterns conferred by the use of dynamic programming, but this research makes it more comprehensive slower. To counter this deficit heuristics were introduced using seeds to accelerate the exploration of similarities

In terms of methodology, it is clear that the compression dictionary compression standards and probabilistic prediction share preferences and credits. The arithmetic coding of order 2 (Arith-2) is almost unanimously chosen as technical support for the portion leftover, whose entropy is too high to be used by techniques previously highlighted.

Sadeh [244] has proposed lossy data compression schemes based on approximate string matching and proved some asymptotic properties with respect to stationary sources. In spite of the good compression ratio, arithmetic coding and CTW have disadvantages such as low decompression speed.

The good compression ratios have been reported by Cao et al, first pure statistical compression method for biological sequences [28]. It used an expert model with Bayesian averaging over a second order Markov model, a first order Markov model estimated on short term data (last 512 symbols) and a repeat model. Weighting of probabilities for each model is based on the minimum description length of the corresponding model in short time history (previous 20 symbols). One advantage of this method is that it assigns probabilities to each symbol to be encoded; therefore we can evaluate the information content of each region spatially. Expressed per element complexity can provide information about the structure of the regions and local properties of a genome, or proteome. Table 3.1 summarizes some specific encoders of DNA and Table 3.2 gives the comparative analysis with other DNA-specific compression algorithms.

Table 3.1: DNA-specific encoders

Program	Authors
BioCompress	Grumbach and F. Tahi
Biocompress-2	S. Grumbach and F. Tahi
Cfact	É. Rivals, J.P. Delahaye, M. Dauchet and O. Delgrange
CTW-LZ	T. Matsumoto, K. Sadakane and H. Imai
GenCompres	X. Chen, S. Kwong and M. Li
DNACompress	X. Chen, M. Li, B. Ma, and J. Tromp
Expert Model	Minh Duc Cao, Trevor I. Dix ,Lloyd Allison
DNAPack	Behshad Behzadi and Fabrice Le Fessant

Table 3.2: Comparison of DNA-Specific Compression Algorithms

standard sequences	BioCompress	BioCompress-2	GenCompress	CTW-LZ	DNACompress	DNA Sequitur	DNAPack	Expert Model
CHMPXX	1.8312	1.6848	1.6730	1.6690	1.6716	2.13	1.6602	1.6577
CHNTXX	1.6748	1.6172	1.6146	1.6120	1.6127	---	1.6103	1.6068
HEHCMVCG	1.882	1.8480	1.8470	1.8414	1.8492	---	1.8346	1.8426
HUMDYSTROP	2.000	1.9262	1.9231	1.9175	1.9116	2.34	1.9088	1.9031
HUMGHCSA	1.345	1.3074	1.0969	1.0972	1.0272	1.86	1.039	0.9828
HUMHBB	1.9356	1.8800	1.8204	1.8082	1.7897	---	1.7771	1.7513
HUMHDAB	1.9228	1.8770	1.8192	1.8218	1.7951	---	1.7394	1.6671
HUMHPRTB	1.9546	1.9066	1.8466	1.8433	1.8165	---	1.7886	1.7361
MPOMTCG	1.9722	1.9378	1.9058	1.9000	1.8920	---	1.8932	1.8768
MIPACGA	1.9904	1.8752	1.8624	1.8555	1.8556	2.16	1.8535	1.8447
VACCG	1.8418	1.7614	1.7614	1.7616	1.7580	2.11	1.7583	1.7649
Average	**1.850**	**1.7837**	**1.7428**	**1.7389**	**1.7254**	**2.12**	**1.7148**	**1.6940**

The SFC, statistical methods applied to a single file, includes BC, GC, DC, DP, and XM. Sequence data compresses poorly using single file compression (SFC) methods (1.6-2.0bits/base). On the other hand, Delta compression is method for lossy compression that make good compression rather than pervious method. Delta compression is used for sizes 10^4, 10^5,10^6 and 10^7. The "db" plot for each size class is for deltas stored in the optimal database schema and the "file" plot is for deltas stored as files. Size 10^2 cannot be stored efficiently as a delta in the database. The storage requirement for the delta file approaches that of the text file. Beginning with size 10^4 the storage efficiency exceeds that of either text files or previously reported SFC approaches. Delta compression [245] can compress DNA sequence data to less than 0.2 bits per nucleotide. Figure 3.8 compares the Delta compression with different sizes to the SFC methods.

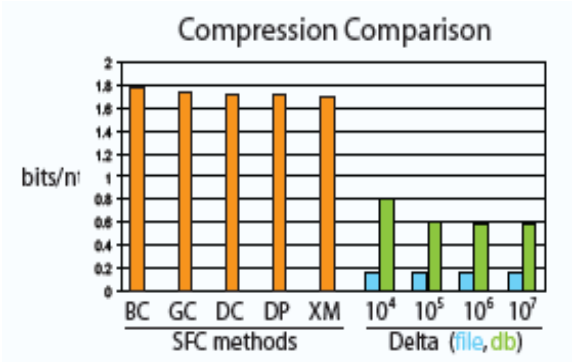


Figure 3.8: Comparison between different compression algorithms

Their average compression rate is about 1.74 bits per base. Recently, P. Rajarajeswari, and A. Apparao [29] presented a new compression algorithm named "DNABIT Compress" whose compression rate is below 1.56 bits per base (for Best case) even for larger genome (nearly 2,00,000 characters)as in Figure 3.9.

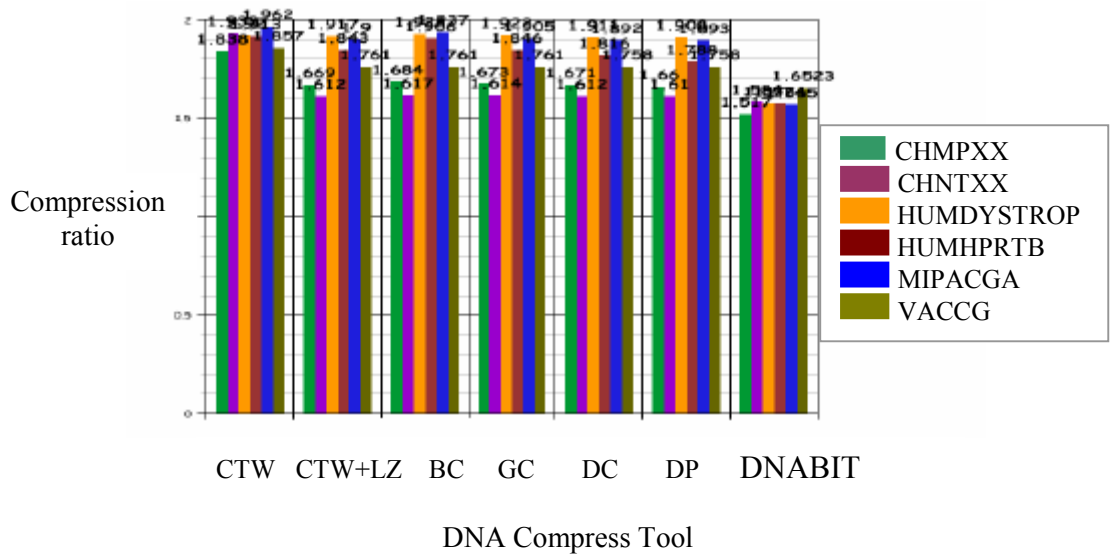


Figure 3.9: Comparison of ratios of DNABIT Compress with existing algorithms

3.4.4 Protein Compression

Less work has been done in the area of protein compression. Nevill-Manning and Witten [246] presented an algorithm for compressing protein based on the PPM algorithm of [27]. The proposed algorithm, Compress Protein (CP), differs from PPM in its ability to cope with amino acid mutations. Where PPM would look for previous occurrences that match exactly the prefix of some region of text, CP attempts to consider similar prefixes as well as exact ones. In doing so, CP must make a prediction taking into account several different contexts and must combine these by assigning each of them a weight. The weights are calculated with respect to the similarity of the contexts to the actual text: those that have few mutations or more common mutations (such as leucine to isoleucine) are given greater weights.

While the above algorithm did not achieve significant compression and was deemed a negative result, an algorithm involving copying by Hategan and Tabus [162] obtained some significant compression. However, this latter algorithm uses pre-determined fixed length blocks and could almost certainly be refined to achieve greater rates of compression.

That the basic principle behind the Lempel-Ziv compression algorithms have been so successful in identifying evolutionary relationships may mean that the differences uncovered through the use of compression are somehow natural to the evolutionary process. This speculation is further supported by the exploitation of distance metrics based on compression for protein classification [247-248] and genome segmentation [248]. Kocsor et al. [229] showed that using compression based approaches can be more accurate for protein classification than the commonly used Smith-Waterman alignment algorithm or Hidden Markov Models. Pelta [248] showed that the compression of protein contact maps can be used for protein classification and [250] showed that the UNIX compress algorithm can be used for protein domain identification.

3.4.5 Difference Compression

An additional major concern is the feasibility of storing or reacquiring samples. Many clinical samples have low DNA content, meaning that the samples cannot be stored and freely distributed in the future. Where samples are non-renewable, as for many cancer and other human samples, future availability of sequence data may only be guaranteed through permanent electronic archives. Even when samples are renewable, the cost of acquiring them can be considerable, especially for clinical samples. Long term storage and distribution of DNA samples worldwide is a complex operation, with considerable costs in physical storage, shipping and end-point sequencing.

Another option for dealing with increasing sequencing data is difference compression (*reference-based compression*). It based on reference genome and then encodes the differences between sequence and the reference genome for storage. With this new compression method, it observe exponential efficiency gains as read lengths increase, and the magnitude of this efficiency gain can be controlled by changing the amount of quality information stored. This compression method is tunable: the storage of quality scores and unaligned sequences may be adjusted for different experiments to conserve information or to minimize storage costs, and provides one opportunity to address the threat that increasing DNA sequence volumes will overcome our ability to store the sequences. This is closest to the “don’t store the raw data, store only the analysis

output, such as the differences" intuition cited above, but changes the challenge from appropriate scientific analysis to appropriate compression. It does not aim to perform a certain type of analysis under this method, nor does it constrain itself to current resources, such as any one current reference genome.

A key feature of difference compression is that its performance improves exponentially with respect to areas of growth in DNA sequencing technology, so it can mitigate the mismatch between DNA sequencing capacity and disk technology growth rates as well as reducing the absolute quantity of stored data. Efficient reference-based compression requires a controlled loss of precision in terms of the sequencing information stored. Usefully, the trade-off between loss of precision and storage efficiency cost can be quantified and presented to the scientific community for discussion of where it is most appropriate to maintain higher precision. In other words, the key insight is to consider that any two human genomes are more than 99% identical, so it is much more efficient to store genomes as variations from a common reference genome; therefore only that 1% of variation needs to be stored. This technique diverges from the research attempting the challenging problem to compress a single genome [226] or recent research on compressing a large database of unrelated sequences [245].

For difference compression scheme to be appropriate, it should be suitable for the way the set is being used. When an entire set is archived or transmitted, the concern will be focused on the compression rate. In this situation, a reference sequence is stored, and the next sequence is generated from this reference and the appropriate difference sequence. Newly generated sequences may be used as references for subsequent sequences. When the set is being actively used, in addition to the compression rate, the speed of decompression is of comparable importance, where any sequence may be repeatedly fetched at any moment. To speed up sequence decompression, it is preferable to have a single reference for the whole set.

Interestingly, difference compression schemes targets the efficient compression of entire databases of sequences. DNAzip [251] was first algorithm introducing the

important idea of only storing differences to a reference sequence, but in this case for storing an entire, assembled genome as a series of difference. This algorithm [251] did not consider the process of generating the variations, which can be a challenging problem in itself, but assume that the variation data have been provided. Variation data are comprised of single nucleotide polymorphisms (SNPs) and insertions/deletions of multiple nucleotides (indels). A SNP was stored as a position value on a chromosome and a single nucleotide letter of the polymorphism, an insertion was stored as a position value and the sequence of nucleotide letters to be inserted, and a deletion was stored as a position value and the length of the deletion. It currently treats inversions as indels. Given the variation data for a genome relative to a reference genome, it considered the compression techniques to reduce the variation data even smaller.

Difference compression schemes, which compress entire sets of homologous sequences by encoding only the differences between a genomic sequences and a reference sequence, are also suggested in [252]. Brandon et al [252] found that selecting the reference sequence is important for having an effective compression of data set. With only a partial level of optimization, 3615 genome sequences occupying 56MB in GenBank are compressed down to only 167 KB, achieving a 345-fold compression rate, using the revised Cambridge Reference Sequence as the reference sequence. Using the consensus sequence as the reference sequence, the data can be stored using only 133 KB, corresponding to a 433-fold level of compression, roughly a 23% improvement.

On the other hand, C. Wang [253] implemented a generic tool, GRS, for de novo compression of genome resequencing data which did not need the reference SNPs map, thus can be widely used for many genomes. When its performance was tested on the first Korean personal genome sequence data set, GRS was able to achieve 159-fold compression, reducing the size of the data from 2986.8 to 18.8 MB. While being tested against the sequencing data from rice and Arabidopsis thaliana, GRS compressed the 361.0 MB rice genome data to 4.4 MB, and the A. thaliana genome data from 115.1MB to 6.5 KB. The architecture of GRS is shown in Figure 3.10. When an individual genome sequence data needs to be compressed, GRS first evaluates each chromosome varied

sequence percentage (δ) based on the reference chromosome. Then it filters the longest identical nucleotide sequence and extracts the different sequence (δ≤0.03), and recodes the different sequence file that has been generated to reduce the file size. Then, GRS uses the Huffman coding strategy to compress the reduced different sequence file to the bz2 type file and generates the command file to decompress the compressed file.

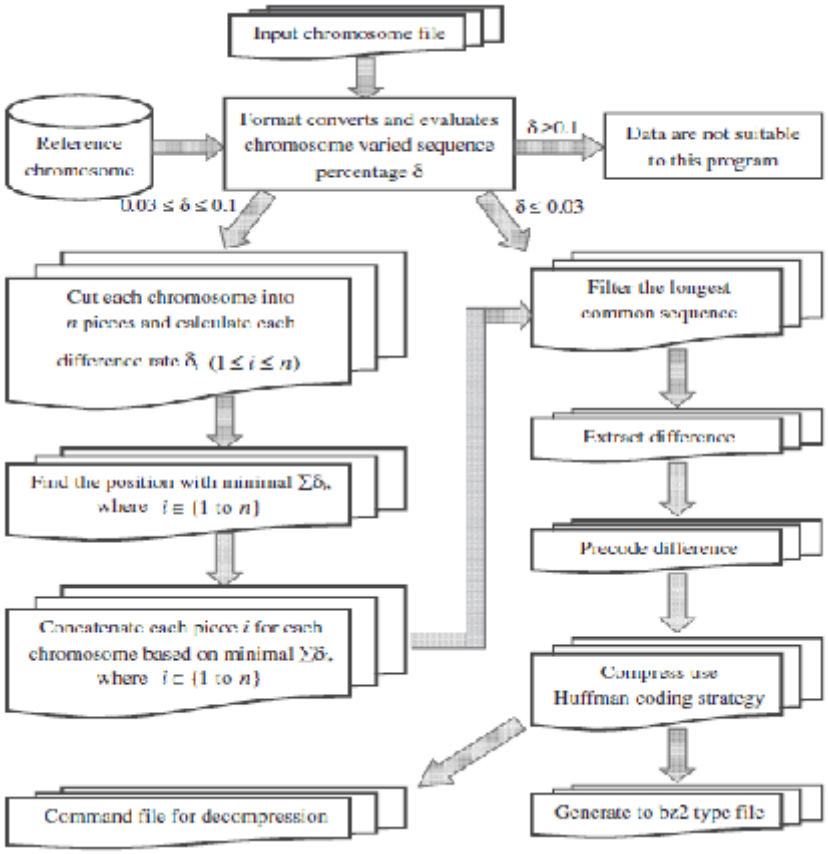


Figure 3.10: Architecture of the GRS Tool. The main modules in GRS connect the input chromosome file, the intermediate data and the final compressed file.

DNAEncodeWG [254] also presented how to compress DNA sequence data using the whole genome sequence of an organism to identify differences between DNA sequences if a repository of the whole genome sequence of the organism is accessible through the Web. It encoded the sequences 10-fold better than the other standard

algorithms. Kozanitis [255] focused on fragment compression as opposed to sequence compression by using SLIMGENE.

Markus Hsi-Yang Fritz [256] describes more efficient method for raw DNA sequence data storage, based on reference sequence. They take a mapping of the reads against the reference sequence (Figure 3.11b) and summarize both mapping properties and deviations from the reference in an efficient manner (Figure 3.11c). Much of the efficiency of this method relies on appropriate use of Golomb codes [257] that are a standard compression technique for storing integer values. This method works as follows:

1. Store the lookup position of each read on the reference sequence as the integer position on this compression framework. The lengths of reads are compressed using Huffman coding [19]. For constant read length across a file, we store the length once.
2. Assume that read order does not have any meaning for any given data set and reorder the reads with respect to the integer position determined in (1), allowing efficient relative encoding of the positions by storing the differences between successive values instead of the absolute values. Given coverage and read length, a Golomb code [257] is chosen that is parameterized on the expected distance between reads.
3. Any variation from the reference is stored as an offset relative to the integer position from (1), along with the base identities (for substitutions and insertions) or the length (for deletions). Offset values, again, are assigned a Golomb code parameterized on the distance between successive variation positions. .
4. The read pair information is stored as an unsigned rank offset from the positionally lower read to the positionally higher read, with three bits present to indicate relative orientation of reads and the strands from which they were sequenced. Again, Golomb encoding, parameterized on expected rank offsets, is used. Figure 3.11 shows schematic of this compression technique.

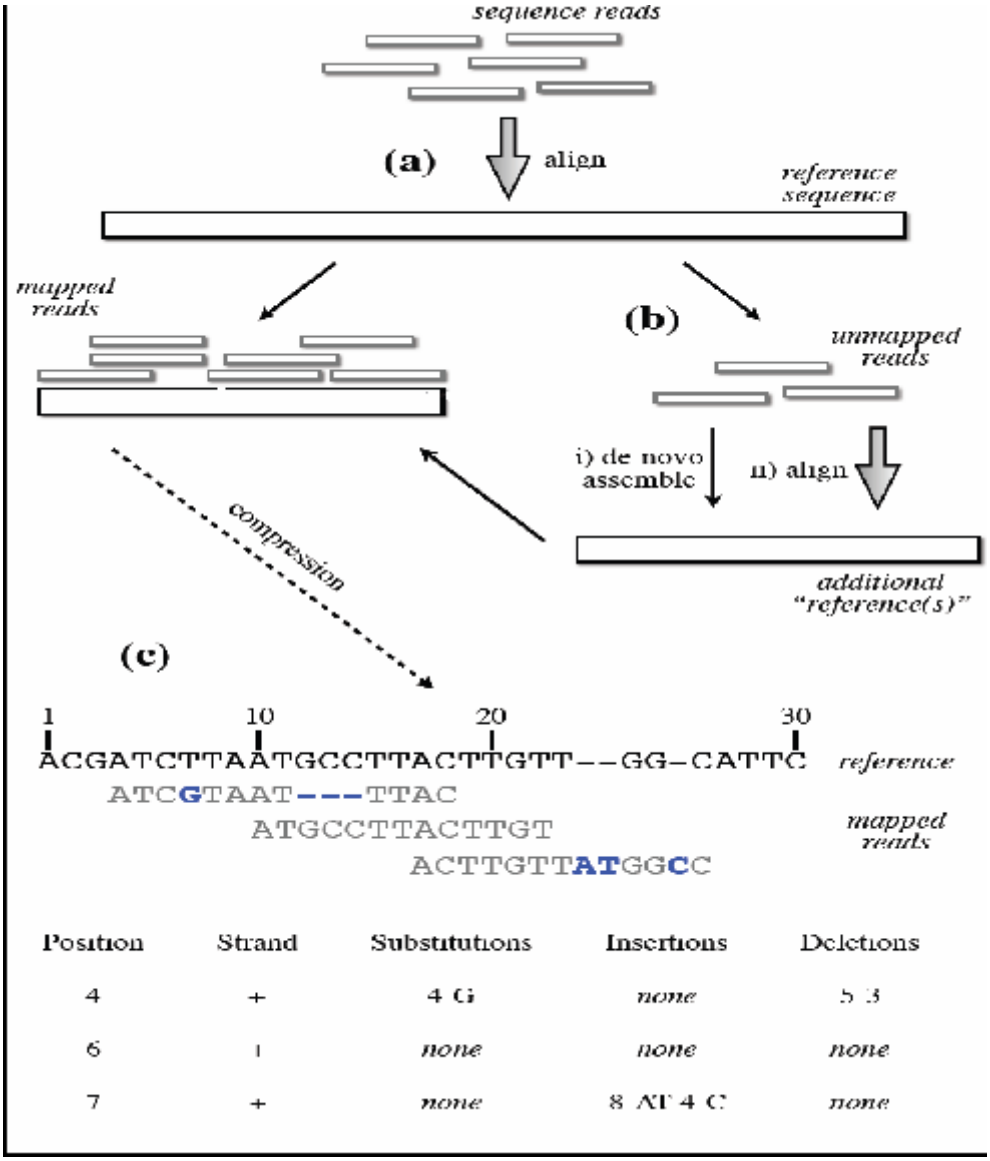


Figure 3.11: Schematic of the compression technique. (a) Reads are first aligned to an established reference. (b) Unaligned reads are then pooled to create a specific "compression framework" for this data set. (c) The base pair information is then stored using specific offsets of reads on the reference, with substitutions, insertions or deletions encoded in separate data structures.

3.5 Applications of Compression

The results of compressing biological sequences can be applied to the problem of evolution derivation [162]. For a pair of sequences, if one sequence is used as training data for compressing the other sequence, the rate of compression achieved is some measure of the relatedness between them. In other words, it is possible to measure how much information one sequence gives about the other.

For example, if two organisms have common ancestry, it is likely that they have some regions of DNA in common. Consider if the DNA sequences for those two organisms are concatenated and subsequently compressed. If a compression algorithm uses repeated regions to compress data, it will detect the similarities between the sequences as repeats and the second sequence will be compressed well. If the compression achieved in the second sequence is better when concatenated with the first than if it was compressed alone, there is some evidence of similarity between the two sequences.

Similarly, compression rates can be used to classify new sequences. Firstly, sequences that belong to the same "class" can be expected to obtain roughly reference sequence under the model. Secondly, applying the information from one sequence to the task of compressing another can show the similarity between sequences.

The suitability of a model can also be measured by compression. A model that is able to predict accurately the remainder of a sequence given its beginning — that is, a good model of the sequence — would achieve excellent compression. A model that made poor predictions would not do as well. Thus, the compression rate can be used to evaluate a new model. This is of prime importance for biological sequences; a model of the composition of such sequences would help in their understanding.

The concepts behind data compression have been very useful in understanding how information is organized in a number of signals used in multimedia communications. It is a natural step to go from analyzing signals such as speech, audio and video to

analyzing biological "signals" such as DNA, RNA, and proteins. The results have been somewhat counterintuitive. Instead of these techniques being useful in the development of compression algorithms for biological signals, these concepts have been most useful in illuminating various biological relationships. These range from providing a species signature to providing tools for analyzing the behavior of gene regulation networks. Reviewing the wide variety of places where these concepts have been useful it is difficult to escape the feeling that information theory, in particular those aspects of it that relate to data compression, are somehow organic to the area of bioinformatics. We believe this will be a fascinating field of study for many years to come and a productive area in which people with an understanding of these concepts can make valuable contributions.

Data compression, and the related information-theoretic techniques, find a wide use for investigation in computational biology. Such a pervasive use has grounds in some outstanding notions that deeply characterizes data compression, in particular universality and quantification of statistical dependence via information measures. Those notions give raise to methods that need very few assumptions on the data models and, as a consequence, very minor parameter estimations for the application of those tools. That seems to be a major advantage for computational biology applications, where the statistical modeling of the data is a highly non-trivial task. In addition, the low-computational demand of those tools allows them to scale well with data set size, even on a genomic scale.

In conclusion, versatility, 'parameter-free' data and association mining, and speed are the main advantages for the use of data compression in biological investigations. However, a non-trivial organizational effort is required in order for this area to collect, in a homogenous way, the set of ideas and tools that would constitute the critical mass required to be recognized as one of the pillars in Bioinformatics. Moreover, the connection of data compression to machine learning is also receiving attention [258] and hopefully it will result in further unifying principles and methodologies, with impact on many disciplines, including the ones connected to the Life Sciences.

Chapter 4

Algorithm Design and Implementation

This chapter will first give a brief overview of the proposed algorithm. It will then state the core requirements for the algorithm, that is, the functional requirements for the algorithm to solve the problem mentioned in the previous chapter. The detailed design and implementation of the algorithm and its components will then be discussed.

4.1 Overview of the algorithm

In order to address the problems mentioned earlier, we propose a new algorithm for compressing different genomic data sources. The new algorithm is designed to develop the difference compression that already discussed in chapter 3. This thesis presents a differential compression algorithm that is based on production of difference sequences according to op-code table in order to optimize the compression of homologous sequences in data set. Therefore, the stored data are composed of representative sequence, the set of differences, and differences locations, instead of storing each sequence individually.

The new algorithm is designed to achieve the less storage and less compression - decompression time to accommodate the huge genomic data. It also designed to produce phylogenic trees based on the representative sequence from set of sequences in order to compression and updating new data. Such that the algorithm can be truly extensible as more bioinformatics tools are being developed.

We address the issues of reference sequence selection, when differentially compressing a set of similar sequences, and the order in what the set should be differentially compressed. The second issue is how to update the differentially compressed set of similar sequences, when a new similar sequence is available. The third issue is how to reduce the exaction time of compression and decompression to achieve excellent compression.

For difference compression scheme to be appropriate, it should be suitable for the way the set is being used. First, when an entire set is archived or transmitted, the concern will be focused on the compression rate. In this situation, a reference sequence is stored, and the next sequence is generated from this reference and the appropriate difference sequence. Newly generated sequences may be used as references for subsequent sequences. The second situation is when the set is being actively used, in addition to the compression rate; the speed of decompression is of comparable importance, where any sequence may be repeatedly fetched at any moment. To speed up sequence decompression, it is preferable to have a single reference for the whole set. Therefore, any generated sequence immediately generated in terms of the single reference. The third situation is when a new similar sequence is added to the set. Here, instead of using the signal fixed reference, any of the already available set sequences can be used to improve the compression efficiency. In this situation, variable references are used. For each situation, a method to select the most appropriate reference sequence is proposed.

In this study, we describe new algorithm for genomic sequence data set which compresses the data set based on comparing it with a reference sequence. We focus our study on large sets of sequences that belong to the same class. If two genomes are, e.g., more than 99% identical, it is much more efficient to store one genome as a variation from the other; in which case, only that 1% representing the variation needs to be stored. A differentially compressed set is a set where a single reference sequence is stored, along

with information about the difference between this sequence and the rest of the set. Our algorithm will be useful to compress the DNA sequence before transmitting it. Also, this algorithm will be used to decompress the DNA data after receiving it without any data loss. To evaluate the suggested algorithm, the compression of the differences, compression of difference locations and the size of the compressed set are examined, as is explained in the next chapter.

This work presents the development and analysis of a family of algorithms for generating differentially compressed output from binary sources. The algorithms all perform the same fundamental task: given two versions of the same data as input streams, generate and output a compact encoding of one of the input streams by representing it as a set of changes with respect to the other input stream. Differential compression provides a computationally efficient compression technique for applications that generate versioned data and we often expect differencing to produce a significantly more compact file than more traditional compression techniques. There are complex practical aspects for using this scheme, such as who is responsible for providing the reference alignment, how reference sequences are stored and verified in different locations, and how additional assembled references would be computed. Again, a practical implementation would have to fit into the current data flow from sequencing groups into the archive sites.

4.2 Algorithm Requirements

The following are the core requirements of the proposed algorithm.

- Free open source to download genomic database (NCBI)[1]
- High similarity between genomic sequences is essential to improve the compression.

In this section, we describe the data that used in our algorithm, compression inductors to evaluate the proposed algorithm and compression methods.

4.2.1 Data Extraction

The approach is demonstrated primarily using a benchmark data set comprising a few thousand individual mitochondrial genomes sequences. Human mitochondrial sequences provide an excellent testbed for developing and testing efficient data structures and algorithms because, unlike nuclear genome sequences, many thousands of fully sequenced mitochondrial genomes are already available, from a diverse population of individuals [259]. In addition, mitochondrial genome sequences pose unique challenges due to their greater variability, as compared with single nucleotide polymorphism (SNP) data. The data used in proposed work consists of:

1- Huge data consists of three different data sets of genomic sequences including 3615 human mitochondrial genomic sequences, 500 human virus sequences H1N1, and 100 mouse sequences Mus Musculus Dmesticus. Mitochondrial data set takes 56MB size in GenBank, and is downloadable from the GenBank database, HapMap web site and the MITOMAO database [259]. Among the sequences, 2671 correspond to complete genome, while the remaining 944 correspond only to the coding region sequence, which is about 1100bp shorter than the full genome sequence. Average sequence length of human is 15446bp.Virus data set takes 601KB size in GenBank, and is downloaded from Influenza Research Database[2]. Average sequence length of virus is 2296bp. Mouse data sets takes 106KB in GenBank, and is downloaded from Mouse Genome Database (MGD)[3]. Average sequence length of mouse is 1500bp.

[1]http://www.ncbi.nlm.nih.gov/
[2]http://www.fludb.org/brc/home.do?decorator=influenza
[3]http://www.informatics.jax.org

References were selected as follows: Cambridge sequence NC_012920 sequence for human mitochondrial data set which is about 16569bp long, HM17663 for virus data set which is about 1714bp long and AJ843867 for mouse data set which is about 1009bp long.

2- We divided huge data set into N separate small data sets randomly which are combined to form huge data set to select the reference sequence based on entropy estimation. We choose the number of small data set after many tries that based on memory of our computer. It consists of 22 genomic sequences from two different classes: the human and mouse genome, downloaded from the GenBank database according to Table 4.1. S1-S10 are human data set, S11-S20 are mouse data set. SH, and SM are external sequences for human and for mouse that need to determine which sequence from data set close to external sequence to refer to reference sequence, then compress it related to reference.

Table 4.1: Small data for Human and Mouse

Complete human mitochondrion genomes		Complete mouse genomes	
Sequence	Accession number	Sequence	Accession number
S1	EU849002	S11	FJ374658
S2	EU849091	S12	FJ374656
S3	EU825949	S13	FJ374665
S4	EU828774	S14	FJ374655
S5	EU828638	S15	FJ374657
S6	EU828637	S16	FJ374659
S7	EU742163	S17	FJ374600
S8	EU742162	S18	FJ374661
S9	EU742161	S19	FJ374662
S10	EU742160	S20	FJ374663
SH	NC_001807	SM	AB042432

4.2.2 Compression Indicators

Three are some parameters as measures for the compression process and acts as indicator of efficient compression such as compression ratio, compression rate, space saving and compression time.

Data compression ratio, also known as compression power, is a computer-science term used to quantify the reduction in data-representation size produced by a data compression algorithm. The data compression ratio is analogous to the physical compression ratio used to measure physical compression of substances, and is defined in the same way, as the ratio between the *compressed size* and the *uncompressed size* [260].

$$\text{Compression Ratio} = \frac{\text{Compressed Size}}{\text{Uncompressed Size}}$$

Space savings is defined as the reduction in size relative to the uncompressed size. And instead of space savings, one speaks of data-rate savings, which is defined as the data-rate reduction relative to the uncompressed data rate:

$$\text{Space Savings} = 1 - \frac{\text{Compressed Size}}{\text{Uncompressed Size}}$$

$$\text{Data Rate Savings} = 1 - \frac{\text{Compressed Data Rate}}{\text{Uncompressed Data Rate}}$$

Historically, there are two main types of applications of data compression: transmission and storage. The term “compression rate” comes from the transmission camp, while “compression ratio” comes from the storage camp. Compression rate is the rate of the compressed data (which we imagined to be transmitted in “real-time”). Typically, it is in units of bits/sample, bits/character, bits/pixels, bits/second or bit/base for genomic data. Compression rate is the ratio of the size or rate of the original data to the size or rate of the compressed data. For example, if a gray-scale image is originally represented by 8 bits/pixel (bpp) and it is compressed to 2 bpp, we say that the

compression rate is 4-to-1. Sometimes, it is said that the compression rate is 75%. Compression rate must be larger than 1.0, the higher the compression rate and lower the compression ratio produces the better the lossless compression scheme. Compression rate is an absolute term, while compression ratio is a relative term. Sometimes, compression rate is called compression ratio.

$$Compression\_Rate = \frac{1}{Compression\_Ratio}$$

When the uncompressed data rate is known, the compression ratio can be inferred from the compressed data rate.

Note: There is some confusion about the term 'compression ratio', particularly outside academia and commerce. In particular, some authors use the term 'compression ratio' to mean 'space savings', even though the latter is not a ratio; and others use the term 'compression ratio' to mean its inverse, even though that equates higher compression ratio with lower compression.

Lossless compression of digitized data such as video, digitized film, and audio preserves all the information, but can rarely do much better than 1:2 compression because of the intrinsic entropy of the data. In contrast, lossy compression (for example JPEG, or MP3) can achieve much higher compression ratios at the cost of a decrease in quality, as visual or audio compression artifacts from loss of important information are introduced.

Compression efficiency, compression time, and decompression time are also obvious measures that should be optimized in the challenge. A more advanced objective is to provide access to compressed file, that is, to allow local decompression. In the case of short sequences, the typical operation would be extract to the original sequence. Finally, the grade of any method depends on how well this method evaluates against its competitors and on how novel are the methods used.

4.2.3 Compression Methods

Generally, compression methods are used after modeling stage to reduce the data storage. In first part, we used Huffman [19] and ZLIB Deflator algorithm[4] for huge data set. In second part, we used BWT [155] and arithmetic coding [20] for small data set.

4.2.3.1 ZLIB Deflator Algorithm: Rapid Lossless Compression Tool

ZLIB Deflator algorithm is divided into two techniques: Zlib, and Deflator. Zlib [261] is a software library used for data compression. Zlib was written by Jean-Loup Gailly and Mark Adler and is an abstraction of the DEFLATE compression algorithm used in their gzip file compression program. Zlib is also a crucial component of many software platforms including Linux, Mac OS X and the iOS. It has been also used in gaming consoles such as the Playstation 3 and Wii. The first public version of Zlib, 0.9, was released on 1 May 1995 and was originally intended for use with image library. It is free software, distributed under the Zlib license. On the other hand, Deflate is a lossless data compression algorithm that uses a combination of the LZ77 algorithm [25] and Huffman coding [19]

4.2.3.1.1 Zlib Technique

Zlib compressed data is typically written with a gzip wrapper or a Zlib wrapper. The wrapper encapsulates the raw DEFLATE data by adding a header and trailer. This provides stream identification and error detection which are not provided by the raw DEFLATE data. The gzip header is larger than the Zlib header as it stores a file name and other file system information. This is the header format used in the ubiquitous gzip file format.

As of February 2010 Zlib only supports one algorithm called DEFLATE which is a variation of LZ77 [25]. This algorithm provides good compression on a wide variety of data with minimal use of system resources. This is also the algorithm used in the ZIP archive format.

---

[4] http://www.mathworks.com/matlabcentral/fileexchange/8899

It is unlikely that the Zlib format will ever be extended to use any other algorithms, though the header makes allowance for this possibility.

The library provides facilities for control of processor and memory use. A compression level value may be supplied which trades-off speed with compression. There are also facilities for conserving memory. These are probably only useful in restricted memory environments such as some embedded systems. There is no limit to the length of data that can be compressed or decompressed. Repeated calls to the library allow an unlimited numbers of blocks of data to be handled. Some ancillary code (counters) may suffer from overflow for long data streams but this does not affect the actual compression or decompression.

Today, Zlib and DEFLATE are often used interchangeably in standards documents. Thousands of applications rely on it for compression, directly or indirectly including:

- The Linux kernel, where it is used to implement compressed network protocols, compressed file systems and to decompress the kernel image itself at boot time.
- libpng, the reference implementation for the PNG image format, which specifies DEFLATE as the stream compression for its bitmap data.
- Libwww, an API for web applications like web browser
- The Apache HTTP server, which uses Zlib to implement HTTP/1.1.
- The OpenSSH client and server, which rely on Zlib to perform the optional compression offered by the Secure Shell protocol.
- The OpenSSL and GnuTLS security libraries, which can optionally use Zlib to compress TLS connections.
- The FFmpeg multimedia library, which uses Zlib to read and write the DEFLATE-compressed parts of stream formats such as Matroska.
- The rsync remote file synchronizer, which uses zlib to implement optional protocol compression.
- The dpkg and RPM package managers, which use Zlib to unpack files from compressed software packages.

- The Subversion and CVS version control systems, which use Zlib to compress traffic to and from remote repositories.
- The Git version control system uses Zlib to store the contents of its data objects (blobs, trees, commits and tags).
- The PostgreSQL RDBMS uses Zlib with custom dump format (pg_dump -Fc) for database backups.

Zlib is also used in many embedded devices such as the Apple Inc. iPhone and Sony Playstation 3 because the code is portable, liberally-licensed and has a relatively small memory footprint.

4.2.3.1.2 Deflate Technique

On the other hand, Deflate is a lossless data compression algorithm that uses a combination of the LZ77 algorithm [25] and Huffman coding [19]. It was originally defined by Phil Katz for version 2 of his PKZIP archiving tool, and was later specified in RFC 1951. Deflate is widely thought to be free of any subsisting patents, and was so at a time before the patent on LZW (which is used in the GIF file format) expired [262]; this has led to its use in gzip compressed files, in addition to the ZIP file format for which Katz originally designed it. A Deflate stream consists of a series of blocks. Each block is preceded by a 3-bit header:

- 1-bit: Last block in stream marker:
  - 1: this is the last-block in the stream.
  - 0: there are more blocks to process after this one.
- 2-bits: Encoding method used for this block type:
  - 00: a stored/raw/literal section follows, between 0 and 65,535 bytes in length.
  - 01: a *static Huffman* compressed block, using a pre-agreed Huffman tree.
  - 10: a compressed block complete with the Huffman table supplied.
  - 11: reserved, don't use.

Most blocks will end up being encoded using method 10, the *dynamic Huffman* encoding, which produces an optimized Huffman tree customized for each

block of data individually. Instructions to generate the necessary Huffman tree immediately follow the block header. Compression is achieved through two steps

- The matching and replacement of duplicate strings with pointers.
- Replacing symbols with new, weighted symbols based on frequency of use.

1- Duplicate string elimination

Within compressed blocks, if a duplicate series of bytes is spotted (a repeated string), then a back-reference is inserted, linking to the previous location of that identical string instead. An encoded match to an earlier string consists of a length (3–258 bytes) and a distance (1–32,768 bytes). Relative back-references can be made across any number of blocks, as long as the distance appears within the last 32 kB of uncompressed data decoded (termed the *sliding window*).

2- Bit reduction

The second compression stage consists of replacing commonly used symbols with shorter representations and less commonly used symbols with longer representations. The method used is Huffman coding which creates an un-prefixed tree of non-overlapping intervals, where the length of each sequence is inversely proportional to the probability of that symbol needing to be encoded. The more likely a symbol has to be encoded, the shorter its bit-sequence will be. A tree is created which contains space for 288 symbols:

- 0–255: represent the literal bytes/symbols 0–255.
- 256: end of block – stop processing if last block, otherwise start processing next block.
- 257–285: combined with extra-bits, a match length of 3–258 bytes.
- 286, 287: not used, reserved and illegal but still part of the tree.

A match length code will always be followed by a distance code. Based on the distance code read, further "extra" bits may be read in order to produce the final distance. The distance tree contains space for 32 symbols:

- 0–3: distances 1–4
- 4–5: distances 5–8, 1 extra bit
- 6–7: distances 9–16, 2 extra bits
- 8–9: distances 17–32, 3 extra bits
- ...
- 26–27: distances 8,193–16,384, 12 extra bits

- 28–29: distances 16,385–32,768, 13 extra bits
- 30–31: not used, reserved and illegal but still part of the tree.

Note that for the match distance symbols 2–29, the number of extra bits can be calculated as $\frac{n}{2}-1$.

During the compression stage, it is the *encoder* that chooses the amount of time spent looking for matching strings. The Zlib/gzip reference implementation allows the user to select from a sliding scale of likely resulting compression-level vs. speed of encoding. Options range from `-0` (do not attempt compression, just store uncompressed) to `-9` representing the maximum capability of the reference implementation in Zlib/gzip. Other Deflate encoders have been produced, all of which will also produce a compatible bitstream capable of being decompressed by any existing Deflate decoder. Differing implementations will likely produce variations on the final encoded bit-stream produced. The focus with non-Zlib versions of an encoder has normally been to produce a more efficiently compressed and small encoded stream.

Deflate64[5], specified by PKWare, is a proprietary variant of the Deflate procedure. The fundamental mechanisms remain the same. What has changed is the increase in dictionary size from 32kB to 64kB, an addition of 14 bits to the distance codes so that they may address a range of 64kB, and the length code has been extended by 16 bits so that it may define lengths of 3 to 65538 bytes[5]. This leads to Deflate64 having a slightly higher compression ratio and a slightly lower compression time than Deflate. Several free and/or open source projects support Deflate64, such as 7-Zip[6], while others, such as Zlib, do not, as a result of the proprietary nature of the procedure and the very modest performance increase over Deflate. Inflate is the decoding process that takes a Deflate bit stream for decompression and correctly produces the original full-size data or file.

---

[5] http://www.binaryessence.com/dct/imp/en000225.htm
[6] http://docs.bugaco.com/7zip/MANUAL/switches/method.htm

Deflation is a means of compressing an octet sequence that combines the LZ77 algorithm for marking common substrings and Huffman coding to take advantage of the different frequencies of occurrence of byte sequences in the file. This algorithm may not be as easy to understand or as efficient as the LZW compression algorithm [261] but Deflate does have the important advantage in that it is not patented. Thus Deflate widely used. Presently it's the most common compression method used by Windows Zip programs (e.g. Winzip) and in the Unix gzip program. Java jar files, being just zip files, also use this compression method.

Both techniques union to be ZLIB Deflator algorithm as encoding step; rapid lossless data compression, it consists of two functions (DZIP and DUNZIP) losslessly compress or decompress (available in MATLAB) most data types so that they occupy less space, and less time. There are some notes about this algorithm:

(1) The input variable can be a scalar, vector, matrix, or n-dimensional matrix

(2) The input variable must be a non-complex and full (meaning matrices declared as type "sparse" are not allowed).

(3) In testing, DZIP compresses several megabytes of data per second.

(4) In testing, sparsely populated matrices or matrices with regular structure can compress to less than 10% of their original size. The realized compression ratio is heavily dependent on the data. (For example, a large stream of truly random data is theoretically impossible to compress.)

(5) Variables originally occupying very little memory (less than about half of one kilobyte) are handled correctly, but the compression requires some overhead and may actually increase the storage size of such small data sets.

(6) LOGICAL variables are compressed to a small fraction of their original sizes.

(7) The DUNZIP function decompresses the output of this function and restores the original data, including size and class type. .

4.2.3.2 Burrows-Wheeler Algorithm

Block-sorting or "Burrows-Wheeler" algorithm [155] is a relatively recent lossless algorithm. An implementation of the algorithm called bzip, is currently one of the best overall compression algorithms for text. It gets compression ratios that are within 10% of the best algorithms such as PPM [27], but runs significantly faster. During the encoding process, the entire input sequence of symbols is permutated and the new sequence contains hopefully some favorable features for compression.

Burrows-Wheeler algorithm processes data in the three stages as shown in Figure4.1.

1. The heart of the algorithm is the Burrows-Wheeler transformation (BWT) on the incoming data, which permutes or reorders data into a form which is especially suited to processing by the following two stages, with clusters of similar symbols and many symbol runs.

2. The permutation step is followed by a recoding of the permuted data by a Move-To-Front (MTF) (For a given symbol x, if n different symbols have been seen since the last occurrence of x, then the symbol x is recoded to the integer n). The essence of this step is that recent symbols recode into small integers and symbol runs recode into runs of zeros; the MTF output is dominated by small values and especially zeros.

3. The final step is some form of statistical compressor. An adaptive Huffman [19] or similar bit-oriented coder is appropriate where speed is more important, but an arithmetic coder [20] may be better if good compression is emphasized.

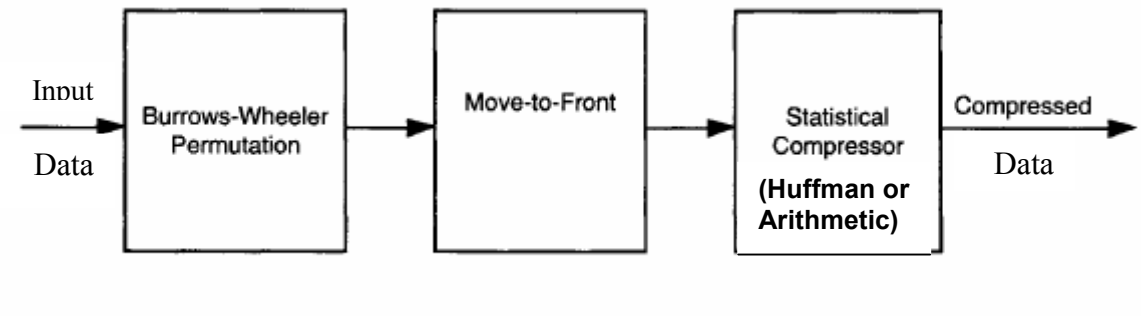


Figure 4.1: Stages of Burrows-Wheeler Algorithm.

There are some following steps for BWT as in Figure 4.2:

1- Forms all possible rotations of input genomic sequence to store in a matrix form.

We first shift the one symbol to the left in a circular way. By circular, we mean that the leftmost symbol in the sequence is shifted out of sequence, and then added back from the right and become the rightmost element in sequence. Repeating the circular shift *n - 1* times, we can generate the *n x n* matrix below where *n* is the number of symbols in sequence, and each row and the column is a particular permutation of sequence.

2- We sort the rows of the matrix in lexicographic order.

3- Output of transform is the last column in sorting matrix.

The repetition of the same character in block might slow the sorting process; to avoid this, using Move-To-Front [18] encoding after BWT.

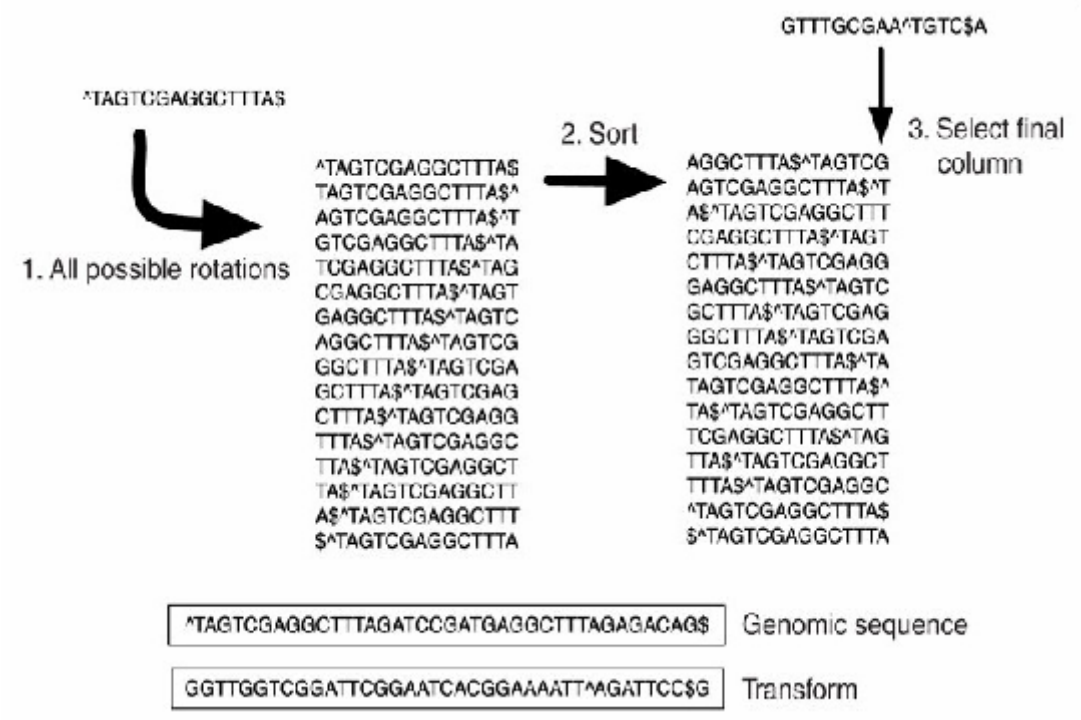


Figure 4.2: Example of BWT Algorithm

4.2.3.3 Move-To-Front (MTF)

Move-To-Front recoding was first used in data compression by Bentley *et al.* [263], who used a word-based compressor with quite large movements of large objects through the MTF list or table. This is used as a sub-step in several other algorithms including the BWT algorithm [155]. The idea of move-to-front coding is to preprocess the input sequence by converting it into a sequence of integers, which hopefully is biases toward integers with low values. The algorithm then uses some form of probability coding to code these values. In practice the conversion and coding are interleaved, but we will describe them as separate passes. The algorithm assumes that each data comes from the same alphabet, and starts with a total order on the alphabet (*e.g.*, [*a,b,c,d,...*]). For each data, the first pass of the algorithm outputs the position of the character in the current order of the alphabet, and then updates the order so that the character is at the head. For example, coding the character *c* with an order [*a,b,c,d,...*] would output a 3 and change the order to [*c,a,b,d*,...]. This is repeated for the full data sequence. The second pass converts the sequence of integers into a bit sequence using Huffman [19] or Arithmetic coding [20]. The hope is that equal characters often appear close to each other in the input sequence so that the integers will be biased to have low values. This will give a skewed probability distribution and good compression.

Suppose that the following sequence of symbols is to be compressed: DDCBEEEFGGAA from a source alphabet (A, B, C, D, E, F,G). MTF works as the following:

1- Initially, the alphabet is stored in an array:

0	1	2	3	4	5	6
A	B	C	D	E	F	G

Read D, the first symbol of the input sequence. Encode D by index 3 of the array, and then move D to the front of the array:

0	1	2	3	4	5	6
D	A	B	C	E	F	G

2- Read D. Encode D by its index 0, and leave the array unchanged because D is already at the front position of the array.

0	1	2	3	4	5	6
D	A	B	C	E	F	G

3- Read C. Encode C by its index 3, and move C to the front:

0	1	2	3	4	5	6
C	D	A	B	E	F	G

4- Read B. Encode it by 3 and move B to the front:

0	1	2	3	4	5	6
B	C	D	A	E	F	G

5- Read E. Encode it by 4 and move E to the front:

0	1	2	3	4	5	6
E	B	C	D	A	F	G

⋮

and so on.

This process continues until the entire string is processed. Hence the encoding is 3, 0, 3, 3, 4 , . - . . In this way, the more frequently occurring symbols are encoded by 0 or small decimal numbers.

4.2.3.4 Statistical Coding

Most of the work on high-performance Burrows-Wheeler compressors has used arithmetic coder [20], in the final stage. Initial work used a combination of Witten's "CACM" arithmetic coder [264], in conjunction with a new data structure for holding the frequency counts [265] which allowed greater choice of the coding range and per-symbol increment. When Moffat *et al.*[266] released their improved coder, that was substituted and immediately gave much poorer compression! Solving this problem led to a much better understanding of Burrows-Wheeler compression. An arithmetic coding model for Burrows-Wheeler algorithm must combine the two attributes:

1. It must handle a wide range of symbol probabilities, from 10,000:1 for a typical text file to 65,000:1 for some binaries. This requires a significant difference between the limit for the accumulated count and the per-symbol increment. The ratio need not be as high as 10,000 or 50,000 to 1, but certainly should be high.
2. It must be able to adapt quickly. Fast adaptation usually implies a small ratio of limit: increment to force frequent overflows and scaling of the counts.

Arithmetic coding [20] is a technique for coding that allows the information from the data in input sequence to be combined to share the same bits. The technique allows the total number of bits sent to asymptotically approach the sum of the self information of the individual data (recall that the self information of data is defined as $\log_2 1/p_i$). To see the significance of this, consider sending a thousand symbols each having probability 0. 999. Using a Huffman code [19], each symbol has to take at least 1 bit, requiring 1000 bits to be sent. On the other hand the self information of each symbol is $\log_2 1/p_i$ = 0. 00144 bits, so the sum of this self-information over 1000 symbols is only 1.4 bits. It turns out that arithmetic coding will send all the symbols using only 3 bits, a factor of hundreds less than a Huffman coder. Of course this is an extreme case, and when all the probabilities are small, the gain will be less significant. Arithmetic coders are therefore most useful when there are large probabilities in the probability distribution.

As mentioned in the introduction, coding is the job of taking probabilities for data and generating bit strings based on these probabilities. How the probabilities are

generated is part of the model component of the algorithm. In practice we typically use probabilities for parts of a larger data rather than for the complete data, *e.g.*, each character or word in sequence. Our model based on op-code files to determine difference sequences. We distinguish between coding algorithms that assign a unique code (bit-string) for each data, and ones that "blend" the codes together from more than one data in a row. In the first class we will consider Huffman codes [19] for huge data, which are a type of prefix code. A prefix code is a special kind of uniquely decodable code in which no bit-string is a prefix of another one, for example {(A, 1), (C, 01), (G, 000), (T, 001)}. All prefix codes are uniquely decodable since once we get a match, there is no longer code that can also match. Prefix codes actually have an advantage over other uniquely decodable codes in that we can decipher each data without having to see the start of the next data. This is important when sending data of different types (e.g., from different probability distributions). In fact in certain applications one data can specify the type of the next data, so it might be necessary to fully decode the current data before the next one can be interpreted. A prefix code can be viewed as a binary tree as follows:

• Each symbol is a leaf in the tree.

• The code for each symbol is given by following a path from the root to the leaf, and appending a 0 each time a left branch is taken, and a 1 each time a right branch is taken. We will call this tree a prefix-code tree. Such a tree can also be useful in decoding prefix codes. As the bits come in, the decoder can follow a path down to the tree until it reaches a leaf, at which point it outputs the symbol and returns to the root for the next bit (or possibly the root of a different tree for a different symbol type). In the later category we consider arithmetic codes [20] and BWT [155] for encoding small data. The arithmetic codes can achieve better compression, but can require the encoder to delay sending data since the data need to be combined before they can be sent.

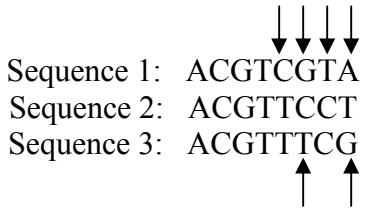
4.3 Differential Compression Algorithm

Previously, differencing has it origins in both longest common subsequence (LCS) algorithms [267] and the string-to-string correction problem [268]. Some of the first applications of differencing updated the screens of slow terminals by sending a set of edits to be applied locally rather than retransmitting a screen full of data. Another early application was the UNIX diff utility which used the LCS method to find and output the changes to data file. LCS algorithms find the longest common sequence between two strings by optimally removing symbols in both files leaving identical and sequential symbols. While the LCS indicates the sequential commonality between strings, it does not necessarily detect the minimum set of changes. More generally, it has been asserted that string metrics that examine symbols sequentially fail to emphasize the global similarity of two strings [269]. Miller and Myers [270] established the limitations of LCS when they produced a new file compare program that executes at four times the speed of the diff program while producing significantly smaller differences.

The edit distance [271] proved to be a better metric for the difference of files and techniques based on this method enhanced the utility and speed of file differencing. The edit distance assigns a cost to edit operations such as “delete a symbol”, “insert a symbol”, and “copy a symbol”. For example, one longest common subsequence between strings xyz and xzy is xy, which neglects the common symbol z. Using the edit distance metric, z may be copied between the two strings producing a smaller change cost than LCS. In the string-to-string correction problem [272], an algorithm minimizes the edit distance to minimize the cost of a given string transformation**.**

Tichy [271] adapted the string-to-string correction problem to file differencing using the concept of block move. Block move allows an algorithm to copy a string of symbols rather than an individual symbol. He then applied the algorithm to source code revision control package and created RCS [272]. RCS detects the modified lines in a file and encodes a difference file by adding these lines and indicating lines to be copied from the base version.

Compression –based distance measures (CBMs) that depend on probabilistic mismatching [273] with neglect the locations of differences, are not distinct enough among different classes. For example, three sequences construct table listing all differences (distance matrix) as the following.



	Seq1	Seq2	Seq3
Seq1	-	4	4
Seq2		-	2
Seq3			-

However, results of pervious work to characterize the difference files are achieve little or no compression as compared to storing the versions uncompressed. Our algorithm is called “Differential compression” that is developing of difference algorithm especially for DNA data sets. A key feature of differential compression is that its performance improves exponentially with respect to areas of growth in DNA sequencing technology, so it can mitigate the mismatch between DNA sequencing capacity and disk technology growth rates as well as reducing the absolute quantity of stored data. Efficient reference-based compression requires a controlled loss of precision in terms of the sequencing information stored whatever the differences and the locations. Usefully, the trade-off between loss of precision and storage efficiency cost can be quantified and presented to the scientific community for discussion of where it is most appropriate to maintain higher precision. Our goal is to bring about a scenario where, for any reasonable sustained improvement in DNA sequencing technology, a constant cost storage solution is available with a well understood loss of precision appropriate for future reuse.

The basis of our algorithm is the efficient storage of information that is identical or near identical to input "reference" sequences. The key feature of our algorithm is that some sequences identical to the reference have minimal impact on storage regardless of their length or depth of sequencing coverage. Additionally, we use the reference genome strictly as a compression framework, and do not require any biological correctness for the reference. Our strategy is to store data set from a variety of "similar" sequences, e.g., those using the same individual or species. These sequences then serve as a secondary compression framework upon which we can provide efficient storage. This is similar but not identical to the task of finding the difference sequences and their locations from whole data set.

Biologically reference sequence that already exits in GenBank (Cambridge reference sequence (CRS)) is not always appropriate for the process of compression. For the highest efficiency of compression, we aim to develop a compression framework based on good selection of reference. Here the distinction between a biologically correct reference sequence and a "useful" compression framework becomes more pronounced. Critically, a compression framework need not be biologically correct or interpretable; it simply needs to provide efficient compression.

4.3.1 Algorithm Components

a) We start first by aligning between each sequence in data set and reference using Sequence Alignment. The sole purpose of sequence alignments is to place homologous positions of homologous sequences into the same column by inserting gapes. Gaps reflect the occurrence of insertions/deletions or other rearrangements during the process. Also, the alignment of similar sequences can help in discovering patterns and relationships between sequences**,** consequently improve their compression ratio.

b) Then, the operation table is used to produce op-code files or differences between each two aligned sequences. Op-code files consist of symbols that ranging from 1 to 8 by using the nine operations. We suppose that there are three relations between each two of DNA sequences according to Figure 4.3: complement, diagonal and vertical. In complement relation, A and T are complements of each other, C and G are also complements of each other, while diagonal relation is the appearance of A and C, or G and T. Vertical is the relation between A and G, or C and T within each two sequences. Similar operation means the repeated base between each two aligned sequences. If two sequences contain numerous of identical bases, the run will be encoded as small difference sequence. Replace operation means replace one base with another base between each two base at the same position. On the other hand, deletion means replace any base with '-' that produced from alignment sequences and insertion operation is the reverse of deletion operation. It means that insertion means replace '-' with any base.

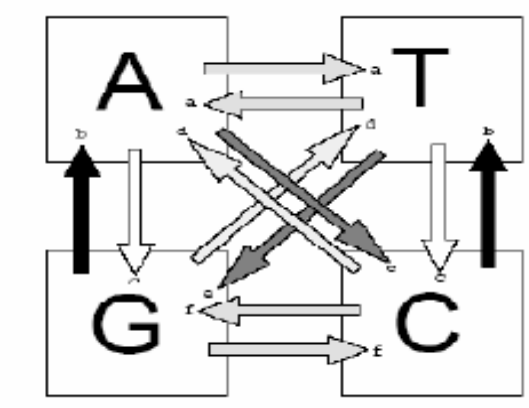


Figure 4.3: Possible transitions between DNA nucleotides

c) Finally, Difference codes and difference locations were compressed using different lossless compression algorithms. The overall flow chart of our algorithm is illustrated in Figure 4.4. We divided our algorithm into two parts according to size of data (huge data in part I and small data in part II).

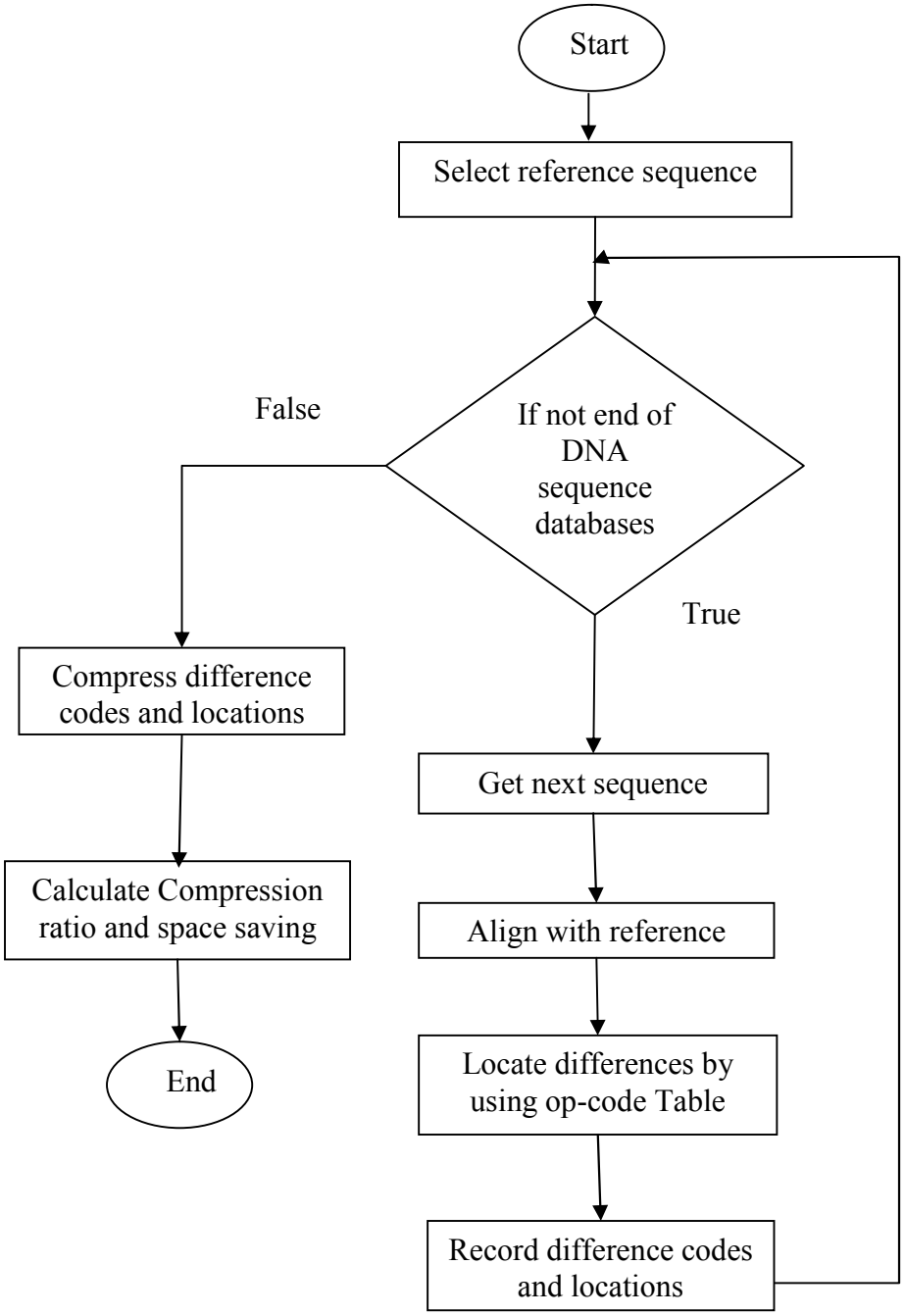


Figure 4.4: Flow chart of the differential compression algorithm

4.3.2 Differential Compression Algorithm of huge data (part I)

The data used in part I consists of huge data sets of genomic sequences. In this part, we select reference sequence for each data set with random method. So, we not focus on how to select the reference sequence. To generate an operation code of differences between a target sequence and its reference, the base at every location of the target is compared with the corresponding bases at the reference. Some efforts have gone towards the construction of the optimal description of differences. In first, we tried to calculate operation code based on distances between two sequences (similar or dissimilar only) e.g. op-code is close to 1 for poorly related sequences, and it is close to 0 for similar sequences only. However, this classification at which the two sequences is not enough for describing difference sequences. We found that increasing of operations gives good model with more details about data. In other word, researches that have been suffering from the poor information to characterize the relatedness between sequences provide a little compression. To fill this gape, a difference between the two bases can be due to one of three modifications to the target: a base insertion, a base deletion, or a replacement. Also, there is three type of replacement operation: complement, vertical, and diagonal. Our proposed operation has been shown to be effective for compression of DNA sequences. Table 4.2 summarizes the operation codes.

Table 4.2: Operation code generation.

Corresponding Bases	Operation	Op-Codes
The same	Similarity	"0"
A ←→T G ←→C	Replacement	"1"
A ←→G C ←→T		"2"
A ←→C G ←→T		"3"
A or T or G or C → '-'	Deletion	"4"
'-' →G	Insertion	"5"
'-' → A		"6"
'-' → C		"7"
'-' → T		"8"

The proposed differential compression algorithm consists of three main steps: Alignment, Differences recording and Differences compression.

1- Alignment

One of the most important tools for sequence analysis, if not the most important one, is sequence alignment which attempts to arrange biological sequences to identify regions of similarity. Similarities between sequences can provide clues to discover the evolutionary relationship between species, to annotate new sequences and to compare an un-known sequence against existing sequences in a large database. There are two broad kinds of sequence alignment, namely global alignment and local alignment. Global alignment attempts to match entire sequences from end to end and thus is suitable for comparing sequences that are expected to have similar structures and functions such as proteins or genes. One the other hand, local alignment searches used for conserved regions, possibly reordered, between two sequences. So, global alignment is thus more suitable for analyzing long sequences, such as chromosomes or genomes, especially from distantly related species where significant insertions, deletions and large rearrangements may have occurred.

Therefore, we start by aligning each sequence in the data set with the reference sequence using global Sequence Alignment that based on Needleman-Wunsch algorithm [119] to make each two sequences in same length and discriminate between similarity parts and un-similarity parts.

2- Differences Recording

In this step, differences and differences locations vectors are recorded. Difference vector is achieved by storing only the op-code of the different bases between the aligned reference sequence and the aligned genomic sequence; these will be according to Table 4.3 ranging from 1 to 8. Differences locations vector is achieved by storing the locations of the unaligned reference sequence where a change of base has happened. It is to be noted out that an insertion in the aligned reference sequence will not change the recorded unaligned reference sequence base location, while any other operation in the aligned

genomic sequence will increment to the next recorded unaligned reference sequence base location.

Table 4.3: Op-code of differences.

		Aligned Reference Sequence Bases				
		G	A	C	T	-
Aligned Genomic Sequence Bases	G	0	2	1	3	5
	A	2	0	3	1	6
	C	1	3	0	2	7
	T	3	1	2	0	8
	-	4	4	4	4	0

As illustrated in Table 4.4, given sequence is aligned with the reference to identify the variants. The variants have an insertion at position 311, a deletion at position 263, and a replacement at positions 73,195,303. We supposed that if there was the repeated operation, it should occupy the same position.

Table 4.4: An Example of using Operation Codes

Reference	**T G T.....T…..G.....AA……._**
Given sequence	**T G C ....A....._ ….GG..... …C**
Difference	2 1 4 2 2 7
Difference locations	73 195 263 303 303 311

3- Differences Compression

The proposed compression algorithm can be divided into two phases, the first phase is differences compression and the second phase is differences locations compression as in Figure 4.5. Each phase consists of encoding and decoding. To improve the compression of differences locations, the distance between successive locations are stored.

Differences compression is implemented by the following procedure:

a) Collect all the differences vectors obtained in the previous step to form one vector *Dvector*.
b) Compress the *Dvector* using two lossless compression algorithms namely, Huffman [19] and ZLIB Deflator algorithm to choose the best compression.
c) Decoding step is concerned with applying DUNZIP and DHuffman to back to difference sequences.

Differences locations compression is implemented by the following procedure:

a) For each difference locations vector a distance vector is calculated by storing the distance *d* that refer to the difference between each two consecutive locations. Note that there will be no change in the first recorded location.
b) Collect all the distance vectors obtained in (a) to form one vector *D.Locvector*.
c) Compress the *D.Locvector* using ZLIB Deflator algorithm.
d) Decoding step is reverse of pervious step by using DUNZIP to back to difference locations.

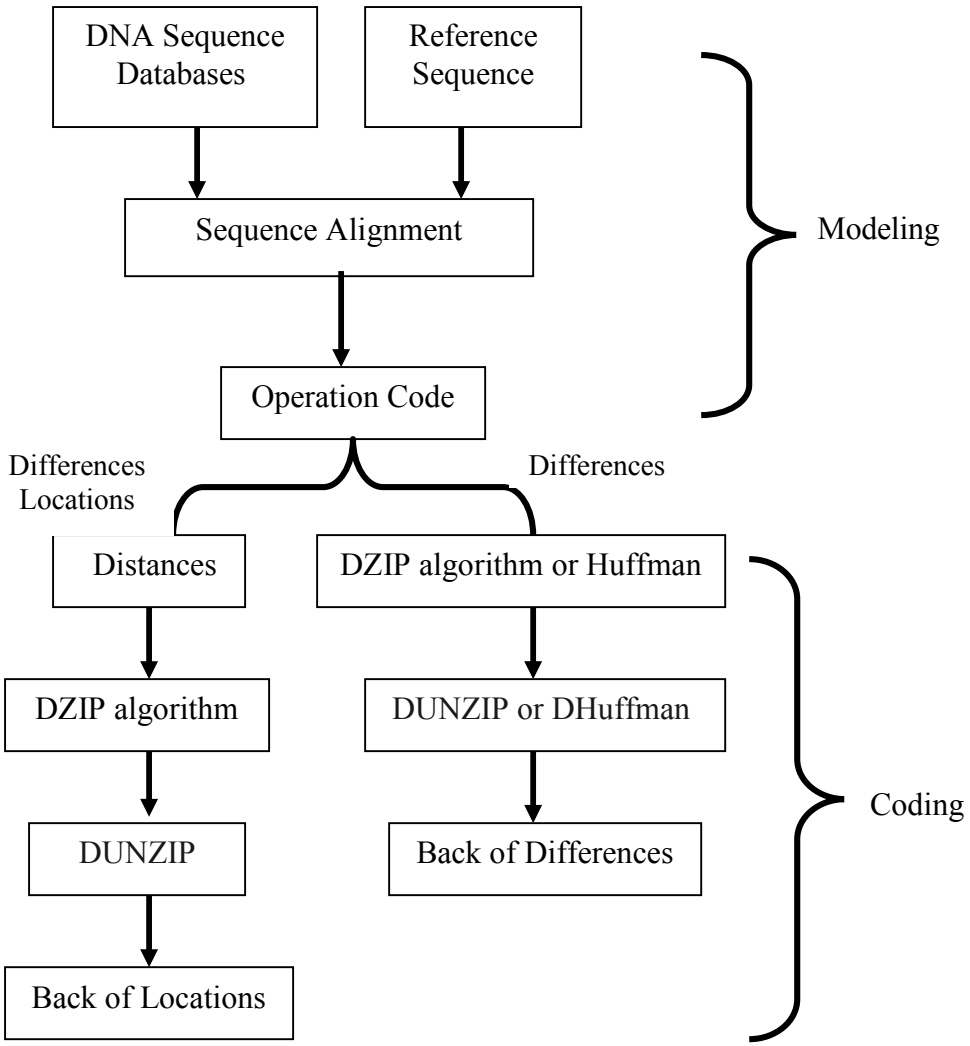


Figure 4.5: A block diagram of the proposed compression algorithm

4.3.3 Differential Compression Algorithm of small data (part II)

We presented another solution to compress huge data set by dividing data set into N of small data sets, which are combined to form hugs data set to select the best reference sequence. We found the number of N depends on performance of our computer memory. We experimentally show that this algorithm take a prohibitive amount of time on data set as large as 1 MB to select the reference sequence based on entropy estimation. The data used in Part II consists of small data sets after partition huge data set that used in Part I. First, we determine the representative sequence from set of sequences based on entropy [147] that described in chapter 2 (equation 2.3). Second, we exploit the representative sequence for best compression of data sets.

In information theory, entropy is a measure that allows for the evaluation of the level of 'randomness' in a string of symbols. Because of the duality of randomness/structure, it seems important to estimate the information content of genomic sequences in order to acquire information on the 'model' generating them that may, in turn, shed light on structure and function, e.g. characterization/identification of coding regions, exons, introns and so on. To give an example of differing entropies in data, consider the following two binary data:

data1=000001000001000001000001

data2=111000100111010010011111

There is much order and repetition in the first data, indicating very little randomness or information content. Therefore, the first data has a low entropy value. On the other hand, the second data seems to have less structure and order and appears more random. This gives the second data higher entropy. For the purposes of compression, a high value of entropy for data indicates that it is difficult to compress. Intuitively, this makes sense: it is easier to describe concisely the first data given above (five zeroes followed by a one and repeat four times $(0^5 1)^4$) than it is to fully describe the second data. The difficulty of compressing high-entropy data is also natural from an informational point of view: data with a lot of information cannot be condensed to a great degree without losing some of the information contained therein. Therefore, the question arises

whether it is possible to used estimators to refer to reference sequence in data sets which reduce either the bias or the variance of the estimate.

One of our goals is to calculate entropy in a useful way regarding two or more sequences and their corresponding species to serve in reduction compression ratio and to be used as criterion for identifying the representative reference sequence to other sequences. Minimum entropy was computed on op-codes files as an indicator to good compression. Difference sequences can help in building phylogenic trees, while the entropy can help in selecting appropriate compression reference for the sequence set, or the order in which a group of similar sequences is differentially compressed (Figure 4.6).

We have applied some following applications to the encoded sequences in order to obtain their entropy estimates and compression ratio.

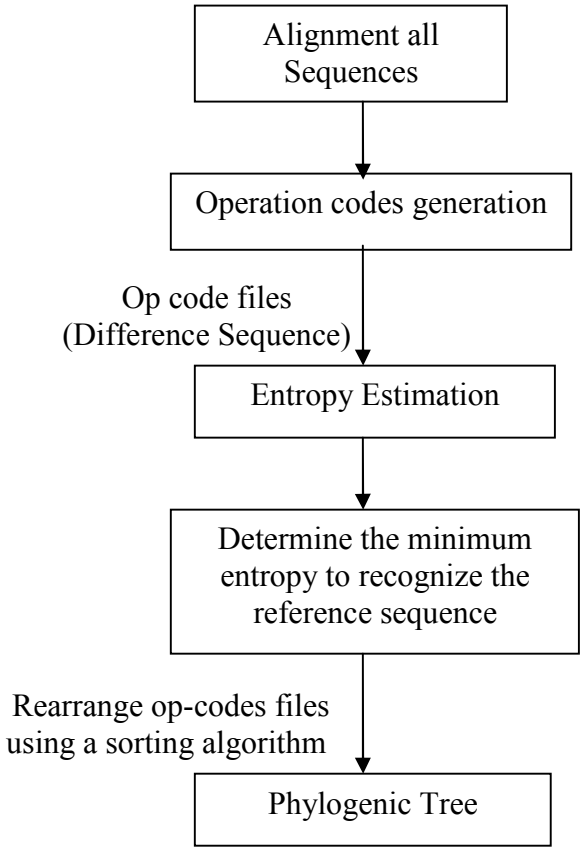


Figure 4.6: A block diagram of selection of reference sequence

4.3.3.1 Selection the representative sequence of a set of sequences

In this section, we briefly demonstrate basic steps to determine the representative sequence of a set of sequences on the condition of the same class.

1- We start first by aligning all the sequences using Sequence Alignment MATLAB tool to align between each two sequences of each class. Some gaps may be inserted to a sequence to make all the sequences with same length.

2- Then produce op-codes files that refer to difference sequences by using the operation presented in Table 4.2, Table 4.3.

3- A good choice for a reference sequence is yielded by calculating the entropy of op-codes files that consist of symbols and determine the minimum entropy to recognize the reference sequence. The sequence which has minimum entropy must be close to the reference sequence and produces a good estimation of compression. Compression and entropy are inseparable.

Based on selection the reference sequence, we can compress data set by calculating the differences between reference sequence and each sequence in data set instead of compress individual sequence in data set as shown in next section.

4.3.3.2 Compression of data set based on selection of the representation sequence

The implementation of the proposed algorithm is outlined as follows:

1. Calculate the entropy of each sequence related to other sequence based on op-codes files, then determine the summation entropy of each sequence and select the minimum one that refers to the representative sequence.

2. Determine the nearest sequence to the representative sequence by using the method presented in section 4.3.3.1 that makes the other reference. The representative sequence can be changed through the algorithm; therefore, we need to arrange the reference by using the threading. Reconstructed phylogenic tree based on entropy of each class under study are produced by rearrange sequences.

3. By using reference sequence for data set, we can use this reference to considerably compress the remaining sequences of this algorithm organism.

4. We have applied BWT [155] that based on a permutation of the input sequence so the ordering could be saved in a few bits. Attempted improvements on BWT that followed by MTF coding [18] have shown success in our algorithm.
5. Finally, we have applied the lossless compression method, arithmetic coding [20], which converts a string into another representation that represents frequently used characters using fewer bits and infrequently used characters using more bits, with the goal of using fewer bits in total.

4.3.4 Update the Differentially Compressed set of similar sequences

The small data set was divided into two types of data; learning data and testing data. Testing data sequences were presented to our algorithm as new sequences that require adding to the learning data set. We choose 5 sequences as testing data and 6 sequences as learning data for both types of data set as in Table 4.1. We used this application, when a new similar sequence is available. We used two methods to update the compressed data set as the following.
1- First method: by using Cambridge sequence (single) as the reference of learning data set, then we calculated difference sequences between reference sequence and each sequence in the testing data set.
2- Second method: by selecting variable reference based on minimum entropy in section 4.3.3.2, and then we compared average entropy and number of differences for each method to determine which method is more efficient to update the compressed data set.

It more difficult to used this method for update the huge data because it takes more time; in turn this new sequence need to determine which sequence from testing data is valid to be reference for this new sequence. Note that the reference that is valid for original data is not necessary to be valid for any external sequence that adds to the original data in the future. In summary, changing of references will occur for any new addition of sequences.

4.3.5 Compression of data set from two different species

We found that there is DNA similarity degree between different organisms e.g. Human and mouse. The overall distribution of local (G+C) content is significantly similar between the mouse and human genomes as in Figure 4.7. Mouse has a higher mean (G+C) content than human (42% compared with 41%), but human has a larger fraction of windows with either high or low (G+C) content. The distribution was determined using the unmasked genomes in 20-kb non-overlapping windows, with the fraction of windows (y axis) in each percentage bin (x axis) plotted for both human and mouse.

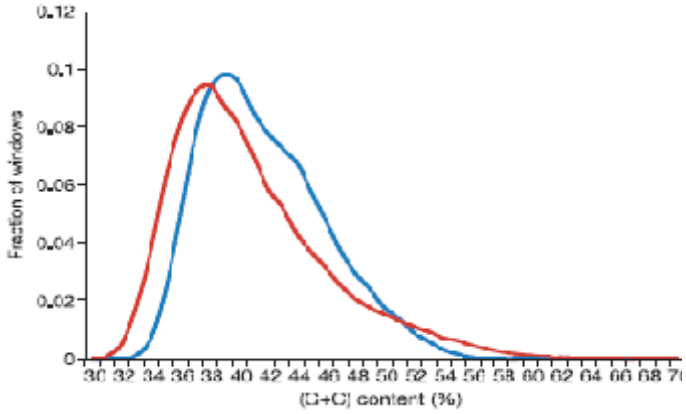


Figure 4.7: Distribution of (G+C) content in the mouse (blue) and human (red) genomes

In Figure 4.8, (a) DNA fragments are sequenced across a number of different organisms; (b) regions of sequence similarity are identified and compared, and (c) Sequence alignments – mouse and human DNA sequence identical is noted with "." and different sequence is shown as the corresponding DNA nucleotide (A, T, C or G). Within these alignments, areas of highest similarity are flagged as identifying possible genes or regulatory regions that require additional analysis. Figure modified from Marguiles and Birney Nature Reviews Genetics 9:304 (2008). Therefore, similarity exists between humans and mice DNA.

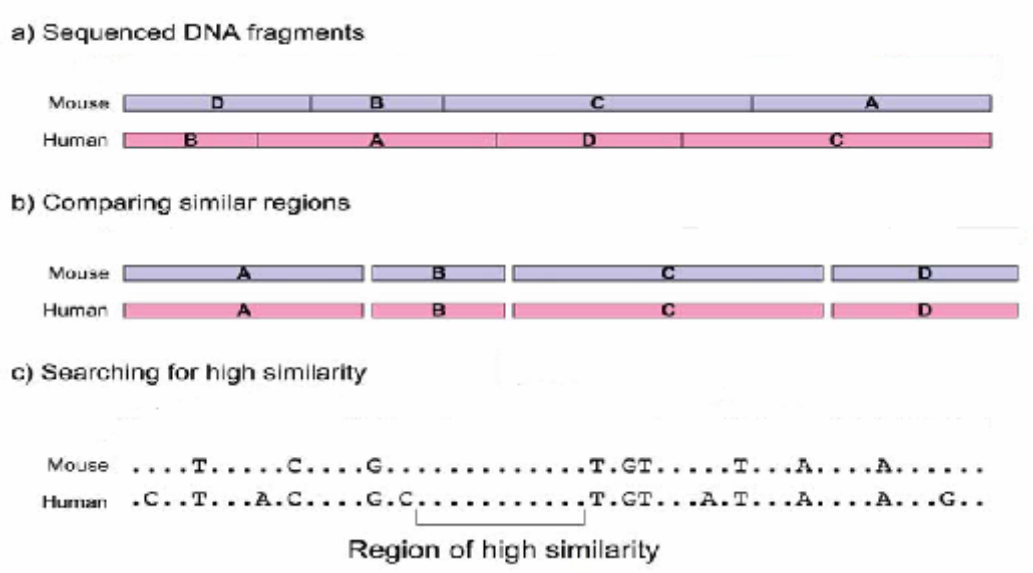


Figure 4.8: Sequence homology among organisms

In Figure 4.9, this evolutionary tree shows the position of mammals whose genomes have been sequenced and analysed (red) or are being sequenced (blue). Humans, mice and rats come from the same major clade (Euarchontoglires), dogs and cows from a further clade (Laurasiatheria). But two major clades of placental mammals remain unsampled (Xenarthra and Afrotheria). Note also the fast evolutionary rate (represented by a longer branch length) of rats and mice compared with humans [274]. Also, Table 4.5 shows the evidence of similarity percent between mouse with Human.

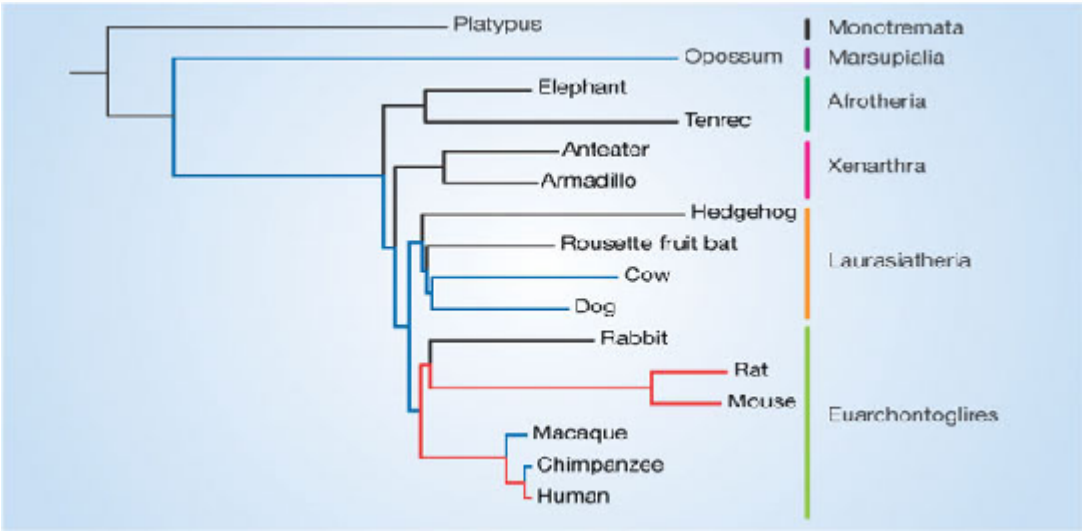


Figure 4.9:Mammalian evolution and genome sequencing.

Table 4.5: What percent of their genes match with Human?

Another human?	99% - All humans have the same genes, but some of these genes contain sequence differences that make each person unique.
A mouse?	94% - All mammals are quite similar genetically.
A chimpanzee?	93% - Chimpanzees are close to humans.
A fruit fly?	44% - Studies of fruit flies have shown how shared genes govern the growth and structure of both insects and mammals.
Yeast?	26% - Yeasts are single-celled organisms, but they have many housekeeping genes that are the same as the genes in humans, such as those that enable energy to be derived from the breakdown of sugars.
A weed?	18% - Plants have many metabolic differences from humans.

Moreover, as scientists begin to understand the common elements shared among species, it may also become possible to approach the even harder challenge of difference compression to identify and understand the differences that make each species unique.

In this part, the small data set consists of the collection between two species to determine which species can compress related to another species and which reference is appropriate for data set. We used this application, when differentially compressing one set of similar sequences in terms of another set of similar sequences. It helps to study the evolution history between various organisms. We used two methods to compress the data set as the following.

1-First method; by using Cambridge sequence (single) as the reference of data set, then we calculated difference sequences between reference sequence and each sequence according to part I.

2-Second method, by selecting reference based on minimum entropy as in section 4.3.3.2. Each species has two references based on two methods. So, we have applied four references to data set for comparing the compression ratio of each method.

We summarize the comparison between two parts as the following table:

Table 4.6: Comparison between part I and part II

	Part I	**Part II**
Objective	This method is used for compression of huge data set with less storage and less time. It treats the whole data set as one unit. (Not easy to select reference by using part I)	This method is used for selecting reference sequence based on minimum entropy and then ordering the sequences by using phylogenic tree to help in updating of new data. It divides whole data set into N small sets to compress each small set individually. It also shows how various organisms are closely related.
Reference	- Single Cambridge sequence. - External reference (outside the original data set)	- May be variable references based on entropy estimation. - Inside the original data set
Applications	It valid for huge data set and small data set	It valid only for small data set (not valid for huge data set)
Execution Time	It achieves the rapid compression.	It takes more time, when ordering the huge sequences, and updating new data.
Similarity	The most important contribution is the idea of high similarity between sequences to compress data set from the same class.	It used for data set from the same class or different species to compress one set of similar sequences in terms of another set of similar sequences e.g. mouse in terms of human.

Chapter 5

Results and Discussions

A prototype is developed to demonstrate the proposed architecture. The prototype is implemented as a plug-in module for the database compression. The plug-in embeds bioinformatics tools as part of the core algorithm. The main objective of implementing the prototype is to include a collection of differences and difference location of database to compress them instead of storing each sequence individually. The prototype is optimized for encoding and decoding speed.

Compression algorithms were run against data sets in order to establish the viability of differential compression for operating system applications such as updating new data, file system backup and restore, selection of best reference for compression, and compress unknown data set from different organisms.

5.1 Experimental Results

To experiment our algorithm, we tried to compress huge data set of DNA sequences by our algorithm, and we compare with results published for other efficient DNA compressors. Results are obtained on platform; a desktop workstation (PC) with 2.1 GHZ Intel Core 2 processor with 3MB cache and 4 GB of RAM under 64 bits window XP. A test prototype is implemented to assess the capability of our framework. The data consists of 3 data sets for huge data (Human, Virus, and Mouse) and 2 data sets for small data (Human, Mouse). We discuss the results of two parts as the following.

5.1.1 Differential Compression of huge data (part I)

The data used in this part consists of three different data sets of genomic sequences including 3615 human mitochondrial genomic sequences, 500 human virus sequences H1N1, and 100 mouse sequences Mus Musculus Dmesticus. We have tested the proposed differential compression algorithm on the three genomic data sets. The size of each data set can be compressed to differences sequences as well as locations of differences. Differences are formed from operation code that shown very success in our model to improve the compression performance. Locations of differences are also compressed in form of distances between the locations.

To compare the results obtained after compression of both differences and differences locations, we used compression ratio which is the ratio between the compressed size and the uncompressed size, and space saving which is defined as the reduction in size relative to the uncompressed size (1-compression ratio), as measures for the compression process.

Difference codes were compressed using two lossless compression algorithms namely, Huffman [19] and ZLIB Deflator algorithm. Table 5.1 shows the generated Huffman codes of difference sequences that produced from op-code table (see Table 4.2). Table 5.2 shows the size of the compressed difference codes. It also shows that ZLIB Deflator algorithm achieved better compression ratio than Huffman algorithm in human

data, while Huffman algorithm achieved slightly better compression ratio in virus and mouse data.

On the other hand, Huffman coding cannot be applied successfully to compress the distances that refer to difference locations. So difference locations are converted to distance and compressed using ZLIB Deflator algorithm. Table 5.3 summarizes the compression ratio and space saving of our algorithm on the three data sets. The results confirm that mitochondria data set that occupies 56MB (58829292 bits) size in GenBank can be stored using only 294.3KB, corresponding to 195-fold compression rate. While virus data set that occupies 601KB (4927072 bits) size in GenBank can be stored using only 212.9KB, corresponding to 3-fold level of compression. Mouse data set that occupies 106KB (870448 bits) size in GenBank can be stored using only 9.6KB, corresponding to 11-fold level of compression. It should be noted that compression of mitochondria data set is better than virus and mouse data sets because it has high similarity between genomic sequences.

Table 5.1: Huffman codes for difference sequences.

	Human data		**Virus data**		**Mouse data**	
Relevant number	Probability	Encoding value	Probability	Encoding value	Probability	Encoding value
1	0.0011	11101	0.0641	110	0.0718	0011
2	0.0942	10	0.0653	101	0.0906	0010
3	0.0021	1111	0.0550	111	0.0483	0111
4	0.8957	0	0.7218	0	0.0696	0110
5	0.0002	1110001	0.0237	10001	0.191	10
6	0.0007	111001	0.025	10000	0.1849	11
7	0.005	110	0.0221	10011	0.1595	010
8	0.0005	1110000	0.0225	10010	0.1839	000

Table 5.2: Size of compressed differences for different compression algorithms

Data set	Uncompressed size	Huffman algorithm	ZLIB Deflator algorithm
Human	1.08MB	156.6KB	**35.6KB**
Virus	374.7KB	**81.6KB**	84.3KB
Mouse	11.8KB	**4.3KB**	5.2KB

Table 5.3: Comparison of compression ratios and space saving

Data set	Uncompressed size	Differences	Locations	Compression ratio	Space saving
Human	56MB	35.6KB	258.7KB	**0.005**	**99.4%**
Virus	601KB	81.6KB	131.3KB	0.354	**64.6%**
Mouse	106KB	4.3KB	5.3KB	0.090	**91.0%**

The most popular algorithms, including specific compression algorithms: DC [226], XM [28], and difference compression scheme: DNAEncodeWG [263], are not suitable for compression of entire databases. We believe that DNAEncodeWG is superior for a single sequence. It should be mentioned that Brandon et al. algorithm [259], was done by using extra reference sequence and based on statistics of the sequence set that are not valid for other sets. The compression result of our algorithm is better than C. Wang [260] that achieved 159-fold compression on Korean personal genome. Our results confirm that the proposed algorithm compresses a 56MB human data file to 294.4KB, with a space savings of 99.4%. For virus data, a 601KB file was compressed to 212.9KB, with a space savings of 64.6%. For mouse data, a 106KB file was compressed to 9.6KB, with a space savings of 91%.

Some notes about part I

- Op code generation acts as good model to describe genomic data set to help in compression process.
- Distance between successive locations is stored rather than locations as the absolute value.
- Huffman coding is more suitable for encoding difference sequence of virus and mouse data set only to produce bit sequences, generally using a binary alphabet to reduce data size.
- Human mitochondrial data set achieves the best compression rather than virus and mouse data sets in both encoding of differences and locations that used ZLIB Deflator algorithm.
- Virus sequences achieve worst compression rather than the others due to their less similarity. Therefore, virus data sets are not valid for our algorithm.

5.1.2 Differential Compression Algorithm of small data (part II)

We have applied differential algorithm to 22 genome sequences encoded op-codes files from alignment method to determine minimum entropy that refers to the representative sequence for given sequence. This part consists of compression one sequence related to another sequence, and compression of small data set based on selection the representative sequence. It also extends to using the pervious algorithm in part I for small data and compare the results.

When the op-code files are ordered by BWT [155] which groups symbols with a similar context close together, the sorting is done in standard lexicographical order. BWT based on a permutation of the input sequence so the ordering could be saved in a few bits. Attempted improvements on BWT that followed by MTF coding have shown success in our model. The most prominent improvements come from the extensive study done by Peter Fenwick [275] to improve the compression performance. Table 5.4 and Table 5.5 show the entropy of op-codes files to recognize the minimum one. The sequence which has minimum entropy must be close to the given sequence and produces a good estimation of compression and minimum compression ratio (Bites/Base) after applying

BWT and arithmetic technique [20]. The results confirm that S5 is reference to SH for human data and S13 is references to SM for mouse data. Compression ratio of SH depending on S5 equals 0.1527 Bites/Base and compression ratio of SM depending on S13 equals 0.0417 Bites/Base.

We also applied the differential model to two classes (10 Human Mitochondria sequences and 10 Mouse sequences) to find the reference sequence of each class. The results of human sequences confirm that S10 is reference where the minimum summation entropy of S10 is 0.1969. We describe these sequences (S8, S9, S1) by S10 because the entropy of S8-S10 equals 0.0017, S9-S10 equals 0.0009, and entropy of S1-S10 equals 0.0295. Sequences (S2, S3) are close to sequence S1 and can be deduced using S1. Similarly sequences (S5, S6) can be deduced using S4. Moreover, S4 close to S8 and S7 close to S9. On the other hand, the results of mouse sequences confirm that S20, S13, S17 are references. So we can select any one from these references to describe S11, S12, S14, S15but we can use S14 as reference to S16 and S18. We also use S11 or S15 as reference to S19. Results of minimum entropy and compression ratio for human and mouse sequences are shown in Table 5.6 and Table 5.7 respectively. Reconstructed phylogenic trees based on entropy of each class under study are shown in Figure 5.1 and Figure 5.2.

By using reference sequence for each model organism where sufficient data is available, we can use this reference sequence to considerably compress the remaining DNA sequences of this model organism.

In this part, we compressed each of ten different DNA sequences of complete human mitochondria genomes by average compression ratio equals to 0.1194 Bites/Base and for mouse genomes by average compression ratio equals to 0.0427 Bites/Base without sending the reference sequence.

Table 5.4: Entropy and compression ratio for human sequences

Differential sequences	Entropy	Compression ratio
SH-S1	0.0272	0.1719
SH-S2	0.0278	0.1784
SH-S3	0.0257	0.1675
SH-S4	0.0159	0.1619
SH-S5	**0.0154**	**0.1527**
SH-S6	0.0231	0.1659
SH-S7	0.0278	0.1674
SH-S8	0.0252	0.1652
SH-S9	0.0243	0.1672
SH-S10	0.0249	0.1644

Table 5.5: Entropy and compression ratio for mouse sequences

Differential sequences	Entropy	Compression ratio
SM-S11	0.0009	0.0421
SM-S12	0.0019	0.0453
SM-S13	**0.00052**	**0.0417**
SM-S14	0.0009	0.0421
SM-S15	0.0009	0.0421
SM-S16	0.0009	0.0421
SM-S17	0.00054	0.0453
SM-S18	0.0019	0.0451
SM-S19	0.0018	0.0451
SM-S20	0.00055	0.0418

Table 5.6: Minimum Entropy and compression ratio for human sequences

Differential sequences	Minimum Entropy	Compression ratio
S9-S10	0.0009	0.0421
S8-S10	0.0017	0.0449
S1-S10	0.0295	0.0797
S2-S1	0.0135	0.1534
S3-S1	0.0145	0.1558
S4-S8	0.0266	0.1717
S5-S4	0.0257	0.1752
S6-S4	0.0270	0.1759
S7-S9	0.0050	0.0762
Average of compression ratio:**0.1194**		

Table 5.7: Minimum entropy and compression ratio for mouse sequences

Differential sequences	Minimum Entropy	Compression ratio
S13-S20	0.0009	0.0421
S17-S20	0.0009	0.0421
S11-S20	0.0017	0.0449
S12-S20	0.0017	0.0449
S14-S20	0.0009	0.0421
S15-S20	0.0009	0.0421
S16-S14	0.0009	0.0421
S18-S14	0.0009	0.0421
S19-S11	0.0009	0.0421
Average of compression ratio: **0.0427**		

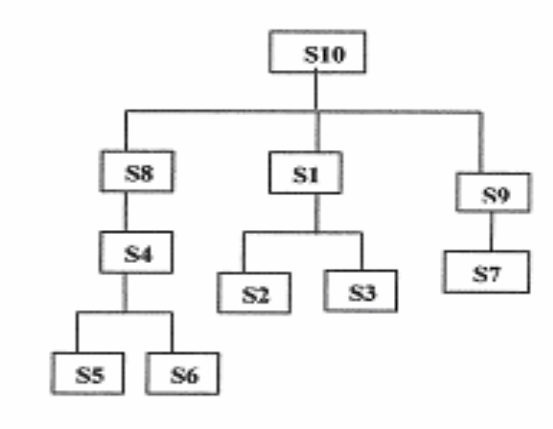


Figure 5.1: The phylogenic tree of human genomic sequences

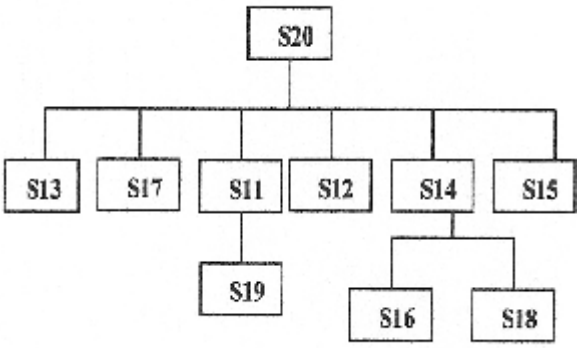


Figure 5.2: The phylogenic tree of mouse genomic sequences

Some notes about part II

- Difference sequence that has minimum entropy refers to reference sequence that will use to produce the minimum compression ratio with all sequences in data set.
- When we select reference sequence based on minimum entropy, we need to reorder all sequences in data set related to selection reference to provide the efficient compression.
- This part is valid for small data set only because it takes a long time to determine reference for huge data set.
- We experimentally show that this algorithm take a prohibitive amount of time on data sets as large as 1 MB to select the reference sequence based on minimum entropy. It depends on specifications of executed computer to determine the limitation of used memory. Improvement our computer memory may be decrease numbers of divided small data set from huge data set.

5.1.3 Update the compressed data set

Updating the data set with a group of sequences, similar to the differentially compressed sequences, is considered. In this application, the original differentially encoded data set was build using 6 sequences of the above set of small genomic set, while we randomly selected 5 sequences as testing data, were presented as new sequences that require adding to the original data set for updating the set. Results show that the entropy, which is an indicator of the compression efficiency, and a measure of relatedness, is much lower with variable reference that based on minimum entropy than that with the single fixed Cambridge reference sequence. Results of average entropy and number of differences between the compressed and reference sequences are shown in Table 5.8.

Table 5.8: Comparison between two methods for updating new data set

Methods	Average Entropy	No. of Differences	Compression ratio
By using entropy theory to select reference	**0.0039**	**236**	**0.0031**
By using Cambridge reference sequence	0.0057	276	0.0039

Some notes about this application.

- This application is used to update the set with new similar sequences whose differences do not obey the same statistical model as that of differences of the original set. When we update the compressed data set, we need to reorder all sequences based on reference selection.
- For updating, we found that method based on entropy theory to select reference is more efficient than method based on Cambridge reference. On the other hand, this application takes a long time for huge data by using the method based on entropy theory rather than method based on Cambridge reference.
- By using entropy theory for reference selection, we search for good reference that closes to each sequence. So, reference that suitable for compressed data set is not always appropriate for updating the compressed data set.
- For the same data set, there are variable references for each new updating.

5.1.4 Comparison between part I and part II

On the other hand, we applied the algorithm of part I to compress small data set by two methods. First, we select the reference based on entropy theory as in part II. Second, we used the Cambridge sequence 'NC_012920' as reference for human data set and 'AJ843867' as reference for mouse data. Then compare the results of each data set as in Table 5.9 and Table 5.10.

Some notes about small Human data set:

- Size of human data before compression =161KByte (1325648 bits)
- Reference is different according to two methods. By using entropy theory, we found this sequence 'DQ779930' as reference according to phylogenic tree. By using Cambridge reference, we select 'NC_012920' as reference sequence.
- By using entropy theory, we found that best compression method for differences is Huffman coding and for locations is ZLIB Deflator algorithm.
- By using Cambridge reference, we found that best Compression method for differences and locations is ZLIB Deflator algorithm.
- Compression of locations achieves good results than compression distances of locations in two methods.
- We found that method based on reference selection by entropy theory is slightly better than method by using Cambridge reference.

Table 5.9: Comparison between two methods for Human data set

Methods	Differences	Difference Locations	Compression ratio	Space saving
By using entropy theory to select reference	61Byte	465Byte	**0.0031**	**99.7%**
By using 'NC_012920' as reference sequence	114Byte	536Byte	0.0039	99.6%

Some notes about small Mouse data set

- Size of mouse data before compression =10KByte(83160 bits)
- Reference is different according to two methods. By using entropy theory, we found this sequence 'MMU47467' as reference according to phylogenic tree. By using Cambridge reference, we select 'AJ843867' as reference sequence.
- By using entropy theory, we found that best Compression method for differences and locations is ZLIB Deflator algorithm.
- By using Cambridge reference, we found that best Compression method for differences and location distances is ZLIB Deflator algorithm.
- Compression of locations achieves good results than compression of location distances by using entropy theory. But, compression of location distances is the best by using Cambridge reference.
- We found that method based on reference selection by entropy theory is significant improvement than method by using Cambridge reference.

Table 5.10: Comparison between two methods for Mouse data set

Methods	Differences	Difference Locations	Compression ratio	Space saving
By using entropy theory to select reference	27Byte	100Byte	**0.0122**	**98.7%**
By using ‘ AJ843867’ as reference sequence	289Byte	339Byte	0.0600	93.3%

For selection the reference by entropy theory, using Huffman and ZLIB Deflator algorithm for small data set compression produces obvious improvement than using BWT and arithmetic coding that used in part II. We proved that the suitability of our model can be measured by compression. A model that is able to predict accurately the sequence given the representative would achieve excellent compression. Thus, a model that made poor predictions would not do as well.

5.1.5 Compression data set from two different species

Another application, the data set used consists of collection between Human &Mouse (20 Sequences). We used two methods to compress the data set by changing the reference sequence as in Table 5.11. It means that compression of sequence related to another sequence from different class is available, on condition that there is similarity between different classes or sequence close to another sequence according to evolution tree.

Some notes about this application

- Size of data set before compression = 171.9KB(1408808 bits)
- By using 'NC_012920' (Cambridge reference of human data) or 'DQ779930' (selected reference of human data based on entropy) as reference for data set, we found that compression of differences by ZLIB Deflator algorithm is better than Huffman. We also found that compression of location distances is better than locations only by ZLIB Deflator algorithm.
- By using 'AJ843867' (Cambridge reference of mouse data) or 'MMU47467' (selected reference of mouse data based on entropy) as reference for data set, we found that compression of differences by Huffman is better than ZLIB Deflator algorithm. We also found that compression of location distances is better than locations only by ZLIB Deflator algorithm.
- By using 'DQ779930' as reference for data set, we found that this reference achieve the best of compression ratio than the others. Therefore, space saving of data set by using this reference is 91.6%. It means that ’DQ779930’ as reference is suitable for data compression that consists of collection between human and mouse rather than the others.
- We noted that method of reference selection for human data set by entropy theory achieves good compression. Human sequence is used to compress data set of collection between human and mouse, but mouse sequence is used only to compress data set of the same class. This meaning confirms the idea of evolution history.
- From above results, we found that compression of sequence related to another sequence from the same class is more efficient rather than this application because

of high similarity between sequences from the same class. However, this application may be used when we need to compress unknown data set from different organisms.

Table 5.11: Comparison between two methods for data set

References	Differences	Locations	Compression ratio	Space saving
'NC_012920'	1.90KB	12.9KB	0.086	91.3%
'AJ843867'	40.9KB	12.5KB	0.311	68.8%
'DQ779930'	1.87KB	12.7KB	**0.084**	**91.6%**
'MMU47467'	40.4KB	12.3KB	0. 307	69.3%

5.1.6 Execution Time

The computational complexity of the new algorithm is related to computation time to obtain the solution for genomic compression. This computational complexity is dividing into modeling phase of sequences based on similarity between them and coding phase to reduce bit size. Modeling phase of DNA data set that based on operation code generation takes many hours. Speed has been part of DNA compression; therefore, we focus on the computation time for data coding. It should be noted that difference encoding take less time rather than difference location encoding. The ability of the user to control the reconstruction time is a unique feature in this algorithm by two methods. First, we compute the execution time for encoding both differences and difference locations of three huge data sets based on Cambridge reference sequence. The time taken for compression is approximately in seconds as shown in Table 5.12. Second, another solution to improve the compression efficiency by dividing data set into N small sets, through changing the reference sequence, was also presented. The reference sequence was selected based on entropy of the differences of each of the candidate sequences. We compute the execution time for small data set as shown in Table 5.13 and then calculate the total time for huge data set that equals N* time for small data set. Although second solution obtains good compression ratios, its execution time is too high to be used for

huge data sets due to rearrangement sequences in each small divided data set according to variable selected references. It noted that execution time for encoding huge data set by using Cambridge reference less rather execution time for small data set by using entropy to select reference. Furthermore, it allows the users to increase the accuracy of the compression of huge data sets by allowing selection of reference based on entropy with a long time. Given the Cambridge reference sequence, compression can be performed with virtually no noticeable delay.

Table 5.12: Execution Time for huge data set

Data set	Uncompressed size	Execution Time
Human	56MB	80seconds
Virus	601KB	50seconds
Mouse	106KB	35seconds

Table 5.12: Execution Time for small data set

Data set	Uncompressed size	Execution Time
Human	161KByte	15Min
Mouse	10KByte	10Min

Figure 5.2 shows the required time to compress huge data set by using three approaches. First, ZLIB deflator algorithm was used to store the differences and locations. Second, Huffman coding was applied to the same data set. Third, Arithmetic coding was used to compress the same data set. As expected, Huffman coding performs well for small size of sequences. With increasing size, the ZLIB deflator algorithm performs better than either the Arithmetic coding or Huffman coding.

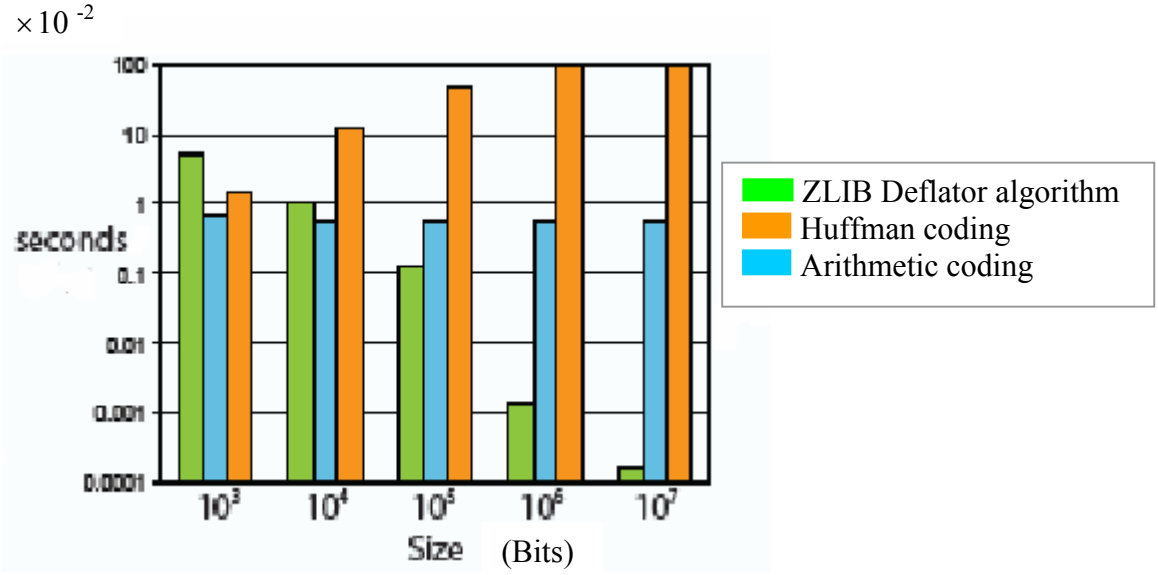


Figure 5.2: Time comparison of three coding algorithm

5.2 Discussions

Although the challenge of storing DNA sequencing information is manageable in the near term, it is important for the biological community to consider the implications of sustained technology improvements in DNA sequencing and to plan for better data compression methods—potentially with controlled loss of precision—before there is a critical mismatch between data generation and storage.

High-throughput sequencing (HTS) is revolutionizing the way molecular biology is researched [276]. The advances in the technology allow cost-effective ways to compress the huge data set of DNA from a given sample. Different enrichment techniques make it possible to prepare samples that contain DNA from targeted areas of genome, such as from the vicinity of transcription factor binding site. Whole genome assembly is also feasible with the new technology [37]. In addition to the obvious challenges to understand the structure, function and evolution of genomes, HTS methods also raise questions about how to efficiently represent, store, transmit, query and protect the privacy of sequence information. These questions are further reinforced if one takes into account also progress in synthetic biology and ability of bioengineer to add new sequences.

We present here a novel compression algorithm that creates a more explicit balance between storage cost and the precision at which data is stored. This compression framework enables us to evaluate the impact of discarding different components of the sequencing information and thereby to make informed choices that choose precision at the expense of storage costs, loss of precision to reduce storage costs, or some value in between. This efficient algorithm is used for encoding huge data by two solutions to reduce storage time. One solution is deal with whole huge data as one unit by using the standard reference. It achieved up to 195-fold compression rate corresponding to 99.4% space saving. Another option for dealing with huge data is dividing into small sets. We noted that selection of reference is very important for compression process but it take a long time for huge data. So, we used the method of reference selection for small data only. We also found that lossless compression methods such as Huffman and ZLIB

Deflator algorithm in part I are better for huge DNA compression than BWT and arithmetic coding in part II.

Our algorithm takes advantage of a number of standard compression techniques on a specific data structure, namely alignment of each sequence to a reference sequence, modeling of differences by operation code generation and relatively standard compression techniques. So far we have focused on the development of lossless differential compression. Also important for a number of applications of sequencing data is per-base continuous value information. While previously this was commonly raw intensity data, the recent trend amongst large projects has been to recommend not preserving these data due to the lack of observed reuse of this information, but rather to compress the whole data. Although previous algorithms scores are reasonably compressible due to their limited integer range and very biased composition (in our hands, real data sets have about 3.5 bits/quality score after Huffman based compression), this is still far higher than the lossless base pair compression described above. Instead we have implemented a different scheme in which all variations and locations of differences are stored. In addition, a user defined method for selection of the right reference.

As can be observed, high similarity between genomic sequences is essential to improve the compression. For each data set, the compression ratio achieved is a measure of the relatedness between its sequences. In other words, it is possible to measure how much information one sequence gives about the others. An important point should be mentioned here is that the proposed algorithm does not require a priori knowledge about that statistics of the sequence set. This will be a big advantage when the updated sequences have different statistics.

The execution time of the differential compression is depending on the length of the reference sequence and difference sequence that needed to be compressed. As the difference length or reference length increases, the execution time increases. On the other hand, decoding speed is similar since both encoder and decoder do essentially the same

computation. Therefore, our current implementation has been optimized for encoding and decoding speed, when the sequence set is under heavy usage.

The differential compression algorithm has been used for lossless compression. Through our investigation, we have found that the use of this algorithm could open new frontiers in quickly identifying unknown sequence related to the set of sequences. So, this algorithm will be used for comparative analysis to classify genomic data .This algorithm can potentially be modified to include variable references instead of using a single reference to improve the difference compression. Further investigation of the new algorithm is needed to further assess its practical value. Moreover, when individuals have complete genome sequences available as part of their personal health records, the focus will shift to difference sequence-level compression. It seems likely that improvement of difference compression will continue to be important to advance knowledge of human genetic variation, and is the pressing problem faced by researchers today.

We have focused our efforts on compressing whole genome shotgun information. This is because such data will be the largest component of sequence archive growth for the next decade, mainly because of the high sample numbers per cancer (~500), high coverage requirements (30x on tumor and normal samples) and high scientific interest (at least 10 declared projects so far) in cancer resequencing [16]. In addition, whole genome sequencing for medical genetics purposes is likely to be the next largest segment of growth. Although there may be a large number of RNA-seq, Chip-seq or other more basic biology focused experiments; they currently occupy less than 20% of the archive by bases, and are likely to occupy progressively less due to the growth in cancer and medical sequencing. Our method can also be extend to apply this algorithm to RNA-seq and Chip-seq data, though careful attention must be paid to such aspects as unaligned data. Other data types, such as shotgun data for *de novo* genome assembly in previously unstudied organisms and environmental sequencing projects aimed at understanding genomics at the community level, will bring further challenges in relation to unaligned reads.

Our differential compression algorithm provides efficient lossless compression for data set that aligns to the reference sequence to locate the differences. This corresponds to most people's intuition that one needs to "store only the analysis output, such as the differences" of a sequencing run. However, there are still critical decisions about what information to discard. One is to what extent unaligned data should be discarded. A critical scientific question is whether the remaining unaligned reads should be stored or discarded, and if they should be stored, can some other compression approach be used to decrease the storage cost. It seems likely that this decision would be different for different data sets; for example cancer genomes compared to their normal controls. A further critical decision is how large the "quality" budget should be, and how this budget should be distributed. A practical approach is to adjust the quality budget to balance the disk improvement rate with improvements in DNA sequencing technologies. As with unaligned reads, such decisions should ideally be made at the level of the data set and, as we need to control the overall storage cost, we might increase the quality budget on some samples (such as tumors) at the expense of other samples (such as their matched normal). Importantly, this framework is flexible with regard to quality budgets, so early implementations can be generous in their quality budget while discussions with analyst and user communities about the impact of different quality budgets in production pipelines are undertaken.

In this scheme, the quality budget is bounded by the number of differences between data set and the reference genome to ensure the correct base sequence is returned; in turn this difference rate is likely to be dominated by sequencing error rates rather than biologically interesting phenomena. In scenarios when there is a large systematic biological difference (e.g., a very high density of SNPs), one can imagine providing a condensed edit structure for the reference sequence to improve compressibility. Achieving quality budgets lower than the sequencing error rate will be more challenging. Potentially one could imagine pre-processing data sets to recognize "clear" sequencing errors, by a combination of the weight of evidence from other data and quality values, and "fix" such errors to be more consistent. However such manipulations are a more fundamental change of the experimental data, and would

require considerable discussion of the consequence for downstream analysis before implementation.

Practical implementation of these ideas will be more complex. There are some complex practical aspects for using this scheme, such as who is responsible for providing the reference alignment, how reference sequences are stored and verified in different locations, and how additional assembled references would be computed. Again, a practical implementation would have to fit into the current data flow from sequencing groups into the archive sites. However many of these questions are relevant to current discussions on how to handle alignment submission in general. By asking for submission as an alignment, itself a very common part of local analysis, some of the complex compute stages would be distributed to the submitters rather than the central resources. In summary, although there are many individual technical details to resolve before this compression routine becomes part of production processes, there is no critical piece which is not solvable with appropriate effort.

DNA sequence has become the first molecular data for which the cost of storage has become a significant proportion of the overall cost of generation and analysis. In common with other technologies with a high storage to generation cost, in particular imaging, we must consider carefully what information, and at what level of precision, is worth storing. As with imaging, intelligent decisions about the precision of the information stored allow efficient, lossy compression algorithms which retain key components of the information for future analysis. By creating efficient compression algorithms we can at least postpone any harder decision about whether specific data sets should be deleted in their entirety, and at best not limit future analysis of DNA data based on constraints on current storage budgets. It seems likely that similar decisions will eventually be necessary for other biological data, from proteomics to metabolomics, as well as for the compression algorithms used for biological and clinical images.

Finally, compression is still largely an art of science and to gain proficiency in an art we need to get a feel for the process. It not depends on fixed or static rules; each data has a special treatment for compression. It depends on trial-and-error solving to close to best compression. Therefore, DNA Compression is still open issue to how conceptual tools in the data compression repertoire have been used in the field of bioinformatics. In the process we hope to convey to the reader the idea that bioinformatics is not simply one more area of application for concepts from data compression—though it is a fertile area for application of these concepts. Rather, it is an area in which the concepts used for the development of data compression have a natural home. If we think of DNA as the original data and evolution as a process of programming where the fitness functions are specified by natural selection, bioinformatics becomes unlike most biomedical applications where the biomedical data is mined for information. The biological sequences themselves become the messages. On the other hand, Information theory and the data compression concepts become natural concepts for understanding the structures in these messages.

Having presented an efficient and general differential algorithm, we establish this algorithm as a viable data compression method for any data sets. We envision differencing as an enabling technology that will amplify the performance of DNA compression and help to mitigate limitations of DNA therapeutic applications.

Chapter 6

Concluding Remarks

6.1 Conclusions

There are three approaches to dealing with the growth in sequences submitted to the public databases: 1) add storage, 2) throw away some data ("triage"), and 3) compress the stored data. These three are not mutually exclusive. A number of arguments for not storing all sequence data have been put forward. These include "don't store the data, just store the physical DNA sample", "discard older data", "discard data from samples that can be regenerated" and "don't store the raw data, store only the analysis output, such as the differences between sequences". The last parallels the approaches taken by the differential compression in that it saves only informative data.

What is common to most of these suggestions is the implied ability to re-sequence any sample at any given point of time, thus obviating the need to store the available sequence data electronically. Given the large scale of international projects, it is clear that at least in the short term the large volumes of data generated by distributed production centers will have to be kept available and ready for aggregation and analysis. One might argue that after analysis, the raw data should no longer be stored electronically. However,

many of these projects run for several years and, even after they have been completed, further follow-up projects arise that require easy access to the data. In addition, the exponential decrease in storage costs means that the cost for storing any particular data set is heavily weighted to the early years of its storage. Given the large worldwide investment in variation and cancer sequencing, it seems inappropriate to limit the re-analysis of sequencing data over the forthcoming decade for the sake of data storage costs.

Data storage costs have become an appreciable proportion of total cost in the creation and analysis of DNA sequence data. Of particular concern is that the rate of increase in DNA sequencing is significantly outstripping the rate of increase in disk storage capacity. In this thesis, we present a differential compression algorithm that efficiently compresses DNA sequences for storage. The output of the algorithm shows that it would be useful for finding repeated regions within sequences and similarities between different sequences to compress only the variations. Every variation is stored as its op-code and its position on the read. The variation type (substitution, insertion, deletion) is produced from the possible transitions between DNA nucleotides (complement, diagonal, vertical) to illustrate additional information (the base change in the case of a substitution, the inserted bases or deleted base).

The proposed challenge is to develop a tailored compression scheme for huge data by using two solutions. First, the compression mechanism can exploit the fixed reference genome that is free on GenBank (one can assume that both the compressor and decompressor have the same reference sequence available). Second, the compression strategy that based on selection the variable reference can be a good approach rather than the first solution to compress each sequence related to the reference and store somehow the occurrence positions and the list of edit differences. In second solution, the reference sequence can be changed through the model; therefore, we need to arrange the references and hence both the compressor and decompressor take more time. It is observed that there is no doubt that selection of reference is very important to improve compression however, it costs more time for huge data. A disadvantage of our algorithm, the

relationship between selection of reference to achieve the good compression for huge data and execution time is inverse proportion.

The lossless differential compression algorithm that is based on the differences and difference locations between each sequence in a genomic data set and its reference sequence is presented. This algorithm is simple, universal which does not depend on the statistics of the data set and could achieve up to 195-fold compression. This algorithm is also effective to achieve higher compression rations, less compression timings, and fast enough to compress DNA huge data sets. Sometime, DNA compression algorithms that used for small data set are not fit for huge data set or it needs more optimization to be valid for huge data set because every data set has different statistics. However, our algorithm is valid for both data whatever the length of sequences. This is considered a great benefit compared to previous works.

In conclusion, the differential compression as presented leaves some avenues explored. Firstly, sequences that belong to the same "class" can be expected to obtain roughly the reference sequence based on the minimum entropy. Secondly, applying the information from the reference sequence can achieve the efficient compression for other sequences and can predict the updating by adding new sequences under our model. Thirdly, modeling of differences is of prime importance for genomic sequences; a model of the composition of such sequences would help in their understanding to achieve the efficient compression. Compression methods are undergoing rapid development making it tempting to store sequencing data for long periods of time so that the data can be re-analyzed with the latest techniques. The challenging open research problems, huge influx of data, and rapidly improving analysis techniques have created the need to store and transfer very large volumes of data. More work is needed to select the prefect reference for huge data, speed up the method of reference selection for huge data to improve difference compression generation, and to further investigate the use of the new method in practical genomic applications.

6.2 Future Developments

It is believed that the set of ideas and tools presented here will contribute innovative to the advancement of science in the field of genomic compression. Soon this work will be made available to the scientific community and all users who need this type of applications. It will certainly arise at this stage that evidence of suggestions and improvements to be introduced. Still, we suggest some considerations in terms of future developments as the following.

(1) The part of data modeling can be better exploited by multiple sequence alignment of various DNA data sets. So just continue their development to alignment all sequences with the reference.

(2) The differential compression model would be an interesting area of research to optimize DNA compression. So it can be extend to compress different classes such as protein or RNA sequences. Are compression algorithms that used for DNA valid for protein or RNA? , especially protein consists of 20 symbols instead of 4 symbols for DNA. Because the distribution of amino acids is not uniform, and non-repeated regions of protein should compress to a little over 4 bits per character rather than $\log_2(20) = 4.3219$.

(3) This would offer reliable data for transferring DNA sequences in order to serve many biological problems mentioning one is to understand a certain disease and be able to develop rational therapeutic strategies.

(4) Developing the difference operations (such as insert or delete more than one base) that may be improving compression ratio.

(5) Developing the differential compression to select the reference based on entropy theory for huge data with less execution time, improving their adaptability could be able to improve the compression performance. In future work, reference selection and time will merge into a unified approach. In any case, tools to compare the successes and failures of approach will be crucial.

(6) The compression algorithm can be improved by incorporating dynamic programming technique to the differential compression.

(7) Algorithm can be extended to check if it finds such data in which same base is being repeated for multiple times, it can be compressed as base{n} where n is the frequency of repetition.

(8) In order to facilitate the use of differential model would be interesting to develop a graphical interface which offers more usable application, in addition would also be useful to assist future users of the work. It also can then carry out more accurate benchmarking on how it scales only the number of variations against other commercial bioinformatics application platforms.

(9) Other development can include the implementation of algorithm, which was developed for searching sequence databases in time proportional to the logarithm of the database size [277].

(10) One of the future goals is to design of the system allows integration of unlimited number of bioinformatics applications including different compression techniques, beside all existing databanks. Also, it is can be extend to design a website about DNA compression, and running multiple database systems with built-in bioinformatics plug-in. This can potentially increase the efficiency of data analysis within a community by off-loading data to other computers on the network and wait for the final result. This allows every host on the network that runs this system has access to all bioinformatics applications.

(11) It extend to using 2_order difference compression algorithm to achieve best compression

(12) Finally, the reuse of test methods developed in other application areas of bioinformatics (Image compression e.g. DNA Microarray).

With so many interesting and important challenges in the data compression field, it is a pity that most work is now concentrated on straightforward applications and minor modifications of existing algorithms, instead of working on more fundamental problems. Obviously, tackling the underlying problems and finding a satisfying, useful, answer will take a lot more time and effort but in the end it will not only be more rewarding, it will turn out to be a dire necessity.

References:

[1] U. Ozkan Nalbantoglu, J. David Russell and Khalid Sayood, "Data Compression Concepts and Algorithms and Their Applications to Bioinformatics", OPEN ACCESS, Entropy, vol.12, pp.34-52; 2010.

[2] E.Schrodinger,"What is Life",Cambridge University Press: Cambridge, UK, 1944.

[3] J. Cohen, "Computer Science and Bioinformatics," Communications of the ACM, vol. 48, no.3, 2005.

[4] B.A.Gata,"Database Similarity Searching Using BLAST and FastA", Australasian Biotechnology, vol. 5, pp.282-290, 1995.

[5] D.G. Korn, K.P. Vo, Vdelta, "Differencing and Compression", Practical Reusable Unix Software, Editor B. Krishnamurthy, John Wiley & Sons, Inc., 1995.

[6] W.Timothy J White, D.Michael Hendy,"Compressing DNA sequence databases with coil, BMC Bioinformatics, 9:242, 2008.

[7] V. D. Gusev, L. A. Nemytikova, and N. A. Chuzhanova, "On the complexity measures of genetic sequences", Bioinformatics, vol. 15(12), pp. 994-999, 1999.

[8] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, "An information-based sequence distance and its application to whole mitochondrial genome phylogeny", Bioinformatics, vol. 17(2), pp.149-154, 2001.

[9] L.Allison, L.Stern, T.Edgoose, and T.I. Dix, "Sequence complexity for biological sequence analysis", Computers and Chemistry, vol. 24(1), pp.43-55, 2000.

[10] E. Rivals, J. Delahaye, M. Dauchet, and O. Delgrange, "A first step toward chromosome analysis by compression algorithms", presented at First International Symposium on Intelligence in Neural and Biological Systems (INBS'95), pp.233, 1995.

[11] N.Goodman, "Biological data becomes computer literate: new advances in bioinformatics", Curr. Op. Biotech., vol. 13, pp.66-71, 2002.

[12] F .Sanger, S. Nicklen, AR.Coulson, "DNA sequencing with chain-terminating inhibitors", Proc. Natl. Acad. Sci. USA 74:5463-5467, 1977.

[13] M. Margulies, M.Egholm, WE.Altman, S.Attiya, JS.Bader, LA.Bemben, J .Berka, MS.Braverman, Y-J.Chen, Z.Chen, et al.,"Genome sequencing in micro fabricated high-density picolitre reactors", Nature 437:376-80, 2005.

[14] LD. Stein, "The case for cloud computing in genome informatics", Genome Biology, 11:207, 2010.

[15] The 1000 Genomes Project Consortium. "A map of human genome variation from population scale sequencing". Nature 467:1061-1073, 2010.

[16] ICGC (The International Cancer Genome Consortium). "International network of cancer genome projects". Nature 464:993-998, 2010.

[17] ENCODE Project Consortium,"The ENCODE(ENCyclopedia Of DNA Elements) \Project", Science (New York, N.Y.) 306:636-40, 2004.

[18] Khalid Sayood, "Introduction to Data Compression, 3rd ed.", University of Nebraska, 2006.

[19] D. A. Huffman. "A method for the construction of minimum-r*edundancy codes*", In Proceedings of the Institute of Electrical and Electronic Engineers, vol. 40, pp.1098–1101, September 1952.

[20] H.Witten, R.M. Neal, and J.G. Cleary, "Arithmetic coding for data compression", Communications of the ACM, vol.30(6),pp. 520–540, 1987.

[21] S. Grumbach, F. Tahi, "A new challenge for compression algorithms: Genetic sequences", J. of Information Proc. and Management, 30(6), pp.875-866, 1994.

[22] E. Rivals, J.-P.Delahaye, M.Dauchet, O. Delgrange, "A Guaranteed Compression Scheme for Repetitive DNA Sequences", Data Compression Conference, 1996.

[23] X. Chen et al. "A compression algorithm for DNA sequences and its applications in Genome comparison", In Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, Tokyo, Japan, April 8-11, 2000.

[24] T. C. Bell, J. G. Cleary, and I. H. Witten, Text Compression: Prentice Hall, 1990.

[25] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," IEEE Trans. Information Theory, vol. 23, pp. 337-343, May 1977.

[26] T. Matsumoto, K. Sadakane, H. Imai, "Biological sequence compression algorithms", Genome Informatics, vol.11, pp.43-52, 2000.

[27] J.G. Cleary, and I.H. Witten, "Data compression using adaptive coding and partial string matching", IEEE Transactions on Communications, Vol. 32 (4), pp.396–402, April 1984.

[28] M. D. Cao, T. I. Dix, L. Allison, and C. Mears, "A Simple Statistical Algorithm for Biological Sequence Compression," presented at IEEE Data Compression Conference (DCC), 2007.

[29] P. Rajarajeswari1, Allam Apparao, "DNABIT Compress–Genome compression algorithm", Bioinformation, Vol. 5, 2011.

[30] RB. Altman, "The interactions between clinical informatics and bioinformatics: a case study", J Am Med Inform Assoc; vol.7:pp.439–443, 2000.

[31] J. Cohen, "Bioinformatics - An Introduction for Computer Scientists", ACM Computer Surveys, vol. 36, no.2, pp. 122-158, 2004.

[32] H. P. Yockey, "Information Theory on Molecular Biology", Cambridge - UK: Cambridge University Press, 1992.

[33] A. Danchin and S. Noria, " Genome structures, operating systems and the image of the machine", 2004.

[34] D.W.Mount,"Bioinformatics: Sequence and Genome Analysis, 2nd Edition", NY: Cold Springer Harbor Press, 2004

[35] C. A. Orengo, D. T. Jones, and J. M. Thornton, "Bioinformatics: Genes, Proteins and Computers", BIOS Scientific Publishers, Oxford, UK, 2003.

[36] I. Korf., "Gene finding in novel genomes", BMC Bioinformatics vol.5,pp.59–67,2004.

[37] Li, R.; Zhu, H.; Ruan, J.; Qian, W.; Fang, X.; Shi, Z.; Li, Y.; Li, S. et al., "De novo assembly of human genomes with massively parallel short read sequencing", Genome Res 20 (2): 265–72 , Feb 2010.

[38] Cohen, N. Claude, "Guidebook on Molecular Modeling in Drug Design", Boston: Academic Press,1996.

[39] Gad, C.Shayne, "Drug discovery handbook. Hoboken", N.J: Wiley-Interscience/J. Wiley, 2005

[40] J.Jung, B.Lee ,"Protein structure alignment using environmental profiles", Protein Eng vol.13,pp.535-543, 2000.

[41] Y. Zhang, "Progress and challenges in protein structure prediction", Curr Opin Struct Biol 18 (3): 342–8, 2008.

[42] F. Brueckner, KJ. Armache, A. Cheung, et al, "Structure–function studies of the RNA polymerase II elongation complex", Acta Crystallogr. D Biol. Crystallogr.65 (Pt 2): 112–20, Feb. 2009.

[43] EM. Phizicky, S. Fields, "Protein-protein interactions: methods for detection and analysis", Microbiol. Rev.: 94–123, 1995.

[44] TA. Manolio; Guttmacher, E.Alan; Manolio, A.Teri, "Genome wide association studies and assessment of the risk of disease", N. Engl. J. Med. 363 (2): 166–76, July 2010.

[45] Maynard Smith, J. , "The Theory of Evolution: Canto Edition", Cambridge University Press, 1993.

[46] P. Hogeweg, "Simulating the growth of cellular forms", Simulation 31 (3), pp.90–96, 1978.

[47] P. Hogeweg, Searls, David B.. ed., "The Roots of Bioinformatics in Theoretical Biology ", PLoS Computational Biology 7 (3): e1002021, 2011.

[48] H.V. Jagadish and F. Olken, "Database Management for Life Science Research," OMICS: A Journal of Integrative Biology vol. 7(1),pp.131-137, 2003.

[49] J.C. Venter, K. Remington, J.F. Heidleberg, A.L. Halpern, D. Rusch, J.A. Eisen, D. Wu, et al., "Environmental Genome Shotgun Sequencing of the Sargasso Sea," Science 304(5667),pp.66-74, 2004.

[50] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP, " Markov Models and HIdden Markov Modeling", Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press, 2007.

[51] V.L. Junker, R. Apweiler, and Bairoch,. "A. Representation of functional information in the SWISS-PROT data bank", Bioinformatics, vol.15, pp.1066-1067, 1999.

[52] V. Junker., S.Contrino, W. Fleischmann., et al., "The role SWISS-PROT and TrEMBL play in the genome research environment", Journal of Biotechnology, vol.78, pp.221- 234, 2000.

[53] D.A. Benson., Karsch-mizrachi. I., Lipman. D.J., Ostell. J., Rapp. B.A., Wheeler. D.L. "Genbank", Nucleic Acids Research., 28(1):15-8, 2000.

[54] H.M.Berman., J.Westbrook, Z.Feng., G.Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov and P.E.Bourne, "The Protein Data Bank", Nucleic Acids Research, vol.28(1), pp.235-242, 2000.

[55] K.Hofman., P.Bucher, L. Falquet and A.Bairoch, "The Prosite Datbase, its status in 1999", Nucleic Acids Research, vol.27(1), pp.215-219, 1999.

[56] T.K. Attwood, Croning. M.D.R., Flower. D.R., et al. "PRINTS-S: The Database Formerly Known As Prints", Nucleic Acids Research., vol.28(1), pp.225- 227, 2000.

[57] F. Corpet., Gouzy. J. and Kahn. D. "The ProDom database of domain families", Nucleic Acids Research, vol.26(1), pp.323-326, 1998.

[58] E.L.L. Sonnhammer., Eddy. S.R, Birney. E., Bateman. A. and Durbin. R. "Pfam: multiple sequence alignments and hmm-profiles of protein domains", Nucleic Acids Research., vol.7(1), pp.260-262, 1999.

[59] R.Dahm, "Discovering DNA: Friedrich Miescher and the early years of nucleic acid research", Hum. Genet.,vol.122 (6), pp.565–81, January 2008.

[60] P. Levene, "The structure of yeast nucleic acid" , J. Biol Chem,vol. 40 (2): pp.415–24, December 1919.

[61] W. Astbury, "Nucleic acid". Symp. SOC. Exp. Bbl 1 (66), 1947.

[62] N. Valery Soyfer, "The consequences of political dictatorship for Russian science", Nature Reviews Genetics , vol.2, pp.723-729, 2001.

[63] MG. Lorenz, Wackernagel W.,"Bacterial gene transfer by natural genetic transformation in the environment", Microbiol. Rev., vol.58 (3), pp.563–602, PMC 372978. PMID 7968924, September 1994.

[64] O. Avery, C. MacLeod, M. McCarty, "Studies on the chemical nature of the substance inducing transformation of pneumococcal types. Inductions of

transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III", J. Exp Med 79 (2), pp.137 158, 1944.

[65] A. Hershey, M.Chase, "Independent functions of viral protein and nucleic acid in growth of bacteriophage" , J Gen Physiol ,vol.36 (1), pp. 39 -56 , 1952.

[66] Watson, J.D., and Crick, F.H.C., "Molecular Structure of Nucleic Acids. A Structure for Deoxyribose Nucleic Acid", Nature 171, pp.737-738, 1953.

[67] Rosalind Franklin and Raymond Gosling ,"The B-DNA X-ray pattern on the right of this linked image", was obtained in at high hydration levels of DNA and it has been labeled as "Photo 51", May 1952.

[68] Nature Archives "Double Helix of DNA": 50 Years".

[69] "Original X-ray diffraction image",Osulibrary.oregonstate.edu, Retrieved 06-02-2011.

[70] Wilkins M.H.F., A.R. Stokes A.R. & Wilson, H.R.."Molecular Structure of Deoxypentose Nucleic Acids", Nature 171 (4356), pp. 738–740, 1953.

[71] "The Nobel Prize in Physiology or Medicine 1962", Nobelprize .org Accessed 22 December 06

[72] Brenda Maddox, "The double helix and the wronged heroine" ,Nature 421 (6921), pp.407–408, 23 January 2003.

[73] Crick,F.H.C.,"On degenerate templates and the adaptor hypothesis", genome.wellcome.ac.uk (Lecture, 1955), Retrieved 22, December 2006.

[74] M. Meselson, F.Stahl ,"The replication of DNA in Escherichia coli", Proc Natl Acad Sci USA 44 (7),pp. 671–82, 1958.

[75] "The Nobel Prize in Physiology or Medicine 1968", Nobelprize.org Accessed 22 December 06

[76] Saenger, Wolfram , "Principles of Nucleic Acid Structure", New York: Springer-Verlag. ISBN 0-387-90762-9, 1984.

[77] A. Ghosh, M. Bansal, "A glossary of DNA structures from A to Z", Acta Crystallogr D Biol Crystallogr 59 (Pt 4): 620–6., 2003.

[78] P.Yakovchuk, E. Protozanova, MD. Frank-Kamenetskii, "Base-stacking and base-pairing contributions into thermal stability of the DNA double helix", Nucleic Acids Res. 34 (2): 564–74, 2006.

[79] A. Bird," DNA methylation patterns and epigenetic memory", Genes Dev 16 (1): 6–21, 2002.

[80] C. Walsh, G. Xu ,"Cytosine methylation and DNA repair", Curr Top Microbiol Immunol 301, pp.283–315, 2006.

[81] S. Kriaucionis, N.Heintz, "The nuclear DNA base 5-hydroxymethylcytosine is present in Purkinje neurons and the brain", Science 324 (5929): 929–30, May 2009.

[82] D.Ratel, J. Ravanat, F. Berger, D.Wion, "N6-methyladenine: the other methylated base of DNA", Bioessays 28 (3): 309–15, 2006.

[83] K. Valerie, L.Povirk, "Regulation and mechanisms of mammalian double-strand break repair", Oncogene 22 (37): 5792–812, 2003.

[84] L. Ferguson, Denny W.,"The genetic toxicology of acridines", Mutat Res 258 (2): 123–60, 1991.

[85] M. Braña, Cacho M, Gradillas A, de Pascual-Teresa B, A.Ramos, "Intercalators as anticancer drugs", Curr Pharm Des 7 (17): 1745–80, 2001.

[86] J. Venter; Adams, MD; Myers, EW; Li, PW; Mural, RJ; Sutton, GG; Smith, HO; Yandell, M et al., "The sequence of the human genome", Science 291 (5507): 1304–51, 2001.

[87] M. Hanbichler, S. Wang, L .Shapiro, "The bacterial nucleoid: a highly organized and dynamic structure", J Cell Biochem 96 (3): 506–21, 2005.

[88] T.Gregory, "The C-value enigma in plants and animals: a review of parallels and an appeal for partnership". Ann Bot (Lond) 95 (1): 133–46, 2005.

[89] A.Pidoux, R.Allshire,"The role of heterochromatin in centromere function",Philos Trans R Soc Lond B Biol Sci 360(1455): 569–79, 2005.

[90] P. Harrison, Hegyi H, Balasubramanian S, Luscombe N, Bertone P, Echols N, Johnson T, Gerstein M., "Molecular fossils in the human genome: identification and analysis of the pseudogenes in chromosomes 21 and 22", Genome Res 12(2): 272 80, 2002.

[91] M. Albà, "Replicative DNA polymerases", Genome Biol 2(1): REVIEWS3002, 2001.

[92] RT.Dame,"The role of nucleoid-associated proteins in the organization and compaction of bacterial chromatin", Mol. Microbiol. 56 (4): 858–70, 2005.

[93] T. Jenuwein, C.Allis,"Translating the histone code",Science 293 (5532): 1074–80, 2001.

[94] T.Ito "Nucleosome assembly and remodeling", Curr Top Microbiol Immunol 274: 1–22, 2003.

[95] C. Iftode, Y. Daniely, J.Borowiec, "Replication protein A (RPA): the eukaryotic SSB",Crit Rev Biochem Mol Biol 34 (3): 141–80, 1999.

[96] B. Spiegelman, R.Heinrich, "Biological control through regulated transcriptional coactivators", Cell 119 (2): 157–67, 2004.

[97] C. Pabo, R.Sauer, "Protein-DNA recognition", Annu Rev Biochem 53: 293–321, 1984.

[98] T. Bickle, D.Krüger, "Biology of DNA restriction", Microbiol Rev 57 (2): 434–50, 1993.

[99] A. Doherty, S.Suh, "Structural and mechanistic conservation in DNA ligases", Nucleic Acids Res 28 (21): 4051–8, 2000.

[100] SP. Goff, P.Berg, "Construction of hybrid viruses containing SV40 and lambda phage DNA segments and their propagation in cultured monkey cells", Cell 9 (4 PT 2): 695–705., 1976.

[101] L. Houdebine, "Transgenic animal models in biomedical research", Methods Mol Biol 360: 163–202, 2007.

[102] H. Daniell, A.Dhingra, "Multigene engineering: dawn of an exciting new era in biotechnology", Curr Opin Biotechnol 13 (2): 136–41, 2002.

[103] A. Jeffreys, Wilson V, Thein S.,"Individual-specific 'fingerprints' of human DNA", Nature 316 (6023): 76–9.1985.

[104] Colin Pitchfork, "first murder conviction on DNA evidence also clears the prime suspect", Forensic Science Service Accessed 23 , December 2006.

[105] PW. Rothemund, "Folding DNA to create nanoscale shapes and patterns", Nature 440 (7082): 297–302, March 2006.

[106] ES. Andersen, M. Dong, MM .Nielsen, "Self-assembly of a nanoscale DNA box with a controllable lid", Nature 459 (7243): 73–6, May 2009.

[107] Y. Ishitsuka, T.Ha, "DNA nanotechnology: a nanomachine goes live", Nat Nanotechnol 4 (5): 281–2, May 2009.

[108] FA. Aldaye, AL Palmer, Sleiman HF, "Assembling materials with DNA as the guide", Science 321(5897): 1795–9, September 2008.

[109] G.Wray, Martindale, Q.Mark, "Dating branches on the tree of life using DNA", Genome Biol 3 (1): REVIEWS0001, 2002.

[110] Lost Tribes of Israel, NOVA, PBS airdate: 22 February 2000. Transcript available from PBS.org, Retrieved 4 March 2006.

[111] Bhattacharya, Shaoni, "Killer convicted thanks to relative's DNA", newscientist.com, Retrieved 22 December 06, 20 April 2004.

[112] Joseph Feller, Brian Fitzgerald, Scott A. Hissam, Karim R. Lakahani, "Perspectives on Free and Open Source Software", MIT Press, 2005.

[113] " Open Bioinformatics Foundation: BOSC", Official website. Open Bioinformatics Foundation, Retrieved 10 May 2011.

[114] "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) ", W3C. April 27, 2007, Retrieved 2011.

[115] Fielding, Roy T.; Taylor, Richard N., "Principled Design of the Modern Web Architecture", ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery) 2 (2): 115–150,2002.

[116] C. Brooksbank, G.Cameron, J.Thornton, "The European Bioinformatics Institute's data resources", Nucleic Acids Research 38(Database issue): D17–D25, 2009.

[117] H.Carrillo, DJ.Lipman, "The Multiple Sequence Alignment Problem in Biology", SIAM Journal of Applied Mathematics, Vol.48, No. 5, 1073-1082,1988.

[118] Durbin, Richard , "Biological sequence analysis: probalistic models of proteins and nucleic acids", Cambridge, UK: Cambridge University Press,1998.

[119] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," J. Mol. Biol., vol. 48, pp. 443-453, 1970.

[120] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," J. Mol. Biol., vol. 147, pp. 195-197, 1981.

[121] J.D.Thompson., D.G.Higgins, and T.J.Gibson, "Clustal W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice", Nucleic Acids Research, vol.22, pp.4673-4680, 1994.

[122] S.F. Altschul, W.Gish, W.Miller, E.W.Myers and D.j.Lipman, "Basic Local Alignment Search Tool", Journal of Molecular Biology.,vol. 215,pp.403-410, 1990.

[123] W.R. Pearson, and D.J.Lipman,"Improved tools for biological sequence analysis", in Proceedings of the National Academy of Sciences, vol.85, pp.2444-2448, 1988.

[124] B. Rost. "Protein structure prediction in 1D, 2D, and 3D" , in The Encyclopaedia of Computational Chemistry (eds. PvR Schleyer, NL Allinger, T Clark, J Gasteiger, PA Kollman, HF Schaefer III and PR Schreiner), vol.3, pp.2242-2255, 1998.

[125] M. S. Vieira, "Statistics of DNA sequences: a low-frequency analysis," Phys. Rev. E, vol. 60(5), pp. 5932–5937, 1999.

[126] F. Piazza and P. Liò, "Statistical analysis of simple repeats in the human genome," Physica A, vol. 347, pp. 472-488, 2005.

[127] S.Karlin and V.Brendel,"Patchiness and correlations in DNA sequences," Science, vol. 259 (5095), pp. 677-680, 1993.

[128] A. Fukushima, M. Kinouchi, Y. Kudo, S. Kanaya, H. Mori, and T. Ikemura, "Statistical Analysis of Genomic Information: Various Periodicities in DNA Sequence,"Genome Informatics, vol. 12, pp. 435–436, 2001.

[129] A. Provata and T. Oikonomou, "Power law exponents characterizing human DNA", Phys. Rev. E, vol. 75, pp. 6, 2007.

[130] D. Kugiumtzis and A. Provata, "Statistical analysis of gene and intergenic DNA sequences", PHISICA A, vol. 342(3-4), pp. 623-638, 2004.

[131] S. Karlin and G. Ghandour, "Comparative Statistics for DNA and Protein Sequences: Multiple Sequence Analysis," Proc. Natl. Acad. Sci. USA, vol. 82, pp. 6186-6190, 1985.

[132] T. M. Cover and J. A. Thomas, "Elements of Information Theory", NY: Wiley & Sons, 1991.

[133] D. Loewenstern and P. N. Yanilos, "Significantly Lower Entropy Estimates for Natural DNA Sequences," Journal of Computational Biology, vol. 6, no.1, 1997.

[134] M. Farach, M. Noordewier, S. Savari, L. Shepp, A. Wyner, and J. Ziv, "On the entropy of DNA: algorithms and measurements based on memory and rapid convergence", presented at Sixth annual ACM-SIAM symposium on Discrete algorithms, San Francisco, California, 1995.

[135] E. N. Trifonov, "Making sense of the human genome," Structure & Methods Adenine Press, vol. 1, pp. 69-77, 1990.

[136] H. Wan, L. Li, S. Federhen, and J. C. Wootton, "Discovering simple regions in biological sequences associated with scoring schemes," Journal of Computational Biology, vol. 10, pp. 171-185, 2003.

[137] J. M. Hancock, "Genome size and the accumulation of simple sequence repeats: implications of new data from genome sequencing projects, " Genetica, vol. 115, pp. 93-103, 2002.

[138] Y. L. Orlov and V. N. Potapov, "Complexity: an internet resource for analysis of DNA sequence complexity," Nucleic Acids Research, vol. 32, pp. 628-633, 2004.

[139] L. L. Gatlin, "Information theory and the living systems", New York: Columbia University Press, 1972.

[140] H. Herzel, A. Schmitt, and W. Ebeling, "Finite sample effects in sequence analysis", Chaos, Solitons & Fractals, vol. 4(1), pp. 97-113, 1994.

[141] M. A. Jimenez-Montano, W. Ebeling, T. Pohl, and P. E. Rapp, "Entropy and complexity of finite sequences as fluctuating quantities", Biosystems, vol. 64(1-3), pp. 23–32, 2002.

[142] Y. Orlov, V. Filippov, V. Potapov, and N. Kolchanov, "Complexity software tools for analysis of information measures of genetic texts, " presented at Workshop on Genomic Signal Processing and Statistics (GENSIPS), Raleigh, North Carolina, USA, 2002.

[143] O. Weiss, M.A.Jimenez-Montano, and H. Herzel, "Information content of protein sequences, "J. Theor. Biol., vol. 206(3), pp. 379-386, 2000.

[144] A. Hariri, B. Weber, and J. Olmsted, “On the validity of Shannon information calculations for molecular biological sequences,” J. Theor. Biol., vol. 147(2), pp. 235–54, 1990.

[145] H. Herzel, “Complexity of Symbol Sequences, ” Systems Analysis Modelling Simulation, vol. 5(5), pp. 435-444, 1988.

[146] A. O. Schmitt and H. Herzel, “Estimating the entropy of DNA sequences,” J. Theor. Biol., vol. 188, pp. 369-377, 1997.

[147] C. E. Shannon and W. Weaver, “The Mathematical Theory of Communication, ” Univeristy of Illinois Press, 1949.

[148] J. K. Lanctot, M. Li, E. Yang, and, “Estimating DNA sequence entropy, ” presented at Symposium on Discrete Algorithms, 2000.

[149] M. Li and P. Vitányi, “An Introduction to Kolmogorov Complexity and Its Applications ”, 2nd Ed. New York: Springer, 1997.

[150] V. K. Balasubrahmanyan and S. Naranan, “Information Theory and Algorithmic Complexity: Applications to Language Discourses and DNA Sequences as Complex Systems Part II: Complexity of DNa Sequences, Analogy with Linguistic Discourses," Journal of Quantitative Linguistics, vol. 7(2), pp. 153-183, 2000.

[151] H.-K. Pao and J. Case, “Computing Entropy for Ortholog Detection, ” presented at Proceedings of World Academy of Science, Engineering and Technology, vol. 1, pp. 89-92, 2005.

[152] J. Liu and D. Li, “Conditional LZ Complexity of DNA Sequences Analysis and its Application in Phylogenetic Tree Reconstruction, ” presented at International Conference on BioMedical Engineering and Informatics, pp. 111-116, 2008.

[153] R. W. Hamming, “Error Detecting and Error Correcting Codes, ” Bell System Technical Journal, vol. 26(2), pp. 147-160, 1950.

[154] J. Kieffer and E. Yang, “Grammar Based Codes: A New Class of Universal Lossless Source Codes, ” IEEE Transactions on Information Theory, vol. 46(3), pp. 737-754, 2000.

[155] M. Burroes and D. J. Wheeler, “A Block Sorting Lossless Data Compression Algorithm,” Technical Report, Digital System Research Center, 1994.

[156] J. Ziv and A. Lempel, “Compression of individual sequences via variable-rate coding,” IEEE Transactions on Information Theory, vol. IT-24, pp. 530-536, 1978.

[157] G. Cormack and N. Horspool, “Data Compression using Dynamic Markov Modelling”, Computer Journal 30:6 ,December 1987.

[158] M. Mahoney, “Adaptive Weighing of Context Models for Lossless Data Compression”, Florida Tech. Technical Report CS-2005-16, 2005.

[159] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete Cosine Transform”, IEEE Trans. Computers, pp.90-93, Jan 1974.

[160] Y.Fischer, “Fractal Image Compression,” SIGGRAPH'92 course notes,1992.

[161] N. Malmurugan, A. Shanmugam, S. Jayaraman and V. Dinesh Chander, “ A New and Novel Image Compression Algorithm Using Wavelet Footprints”,2000.

[162] A. Hategan and I. Tabus, “Protein is compressible”, In Proc 6th Nordic Signal Processing Symposium, pages 192–195, 2004.

[163] E. Ukkonen, “On-line construction of suffix trees”, Algorithmica, vol.14 (3), pp.249–260, 1995.

[164] Udi Manber and Gene Myers. “Suffix arrays: a new method for on-line string searches”, SIAM Journal on Computing, vol. 22, Issue 5, pp. 935–948, October 1993.

[165] P. Ko and S. Aluru. “Space efficient linear time construction of suffix arrays”, Lecture Notes in Computer Science, 2676:200–210, Jan 2003.

[166] G. Manzini and M. Rastero, “A simple and fast DNA compressor”, Software Practice and Experience, vol.34,pp.1397–1411, August 2004.

[167] A. Califano and I. Rigoutsos. “FLASH: A fast look-up algorithm for string homology”, In Computer Vision and Pattern Recognition, Proceedings CVPR ’93, IEEE Computer Society Conference on, pages 353–359, New York, June 1993.

[168] S. Kumar and A.Filipski, “Multiple sequence alignment: In pursuit of homologous DNA positions,” Genome Research, vol. 17, pp. 127-135, 2007.

[169] L. Noé and G. Kucherov, “Improved hit criteria for DNA local alignment,” BMC Bioinformatics, vol. 5(149), 2004.

[170] B. Ma, J. Tromp, and M. Li, "Pattern Hunter: fast and more sensitive homology search," Bioinformatics, vol. 18, pp. 440-445, 2002.

[171] M. Li, B. Ma, D. Kisman, and J. Tromp, "PatternHunter II: Highly Sensitive and Fast Homology Search," J Bioinform Comput Biol, vol. 2(3), pp. 417-439, 2004.

[172] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, and S. Batzoglou, "LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA," Genome Research, vol. 13(4), pp. 721-731, 2003.

[173] Giancarlo, R.; Scaturro, D.; Utro, F. "Textual data compression in computational biology: A synopsis", Bioinformatics, 25, 1575–1586, 2009.

[174] C. Adami, "Information Theory in Molecular Biology," Phys. Life Rev., 1, 3–22, 2004 .

[175] Godfrey-Smith,P. and K. Sterelny, "Biological information", In The Stanford Encyclopedia of Philosophy. Stanford University Press, 2008.

[176] A.K.Konopka, "Information theories in molecular biology and genomics", Nat. Encyclopedia Hum. Genome, 3, 464–469, 2005.

[177] J. Rissanen, et al., "Editorial: information theoretic methods in bioinformatics", EURASIP J. Bioinform. Syst. Biol., 7, 1–4, 2007.

[178] P. Ferragina, R.Giancarlo, V.Greco, G.Manzini, G.Valiente, "Compression-based classification of biological sequences and structures via the universal similarity metric: Experimental assessment", BMC Bioinf., accessed on December 29, 2009. http://www.biomedcentral.com/1471-2105/8/252 .

[179] Li WZ, Jaroszewski L, Godzik A., "Clustering of highly homologous sequences to reduce the size of large protein databases", Bioinformatics, 17:282-283, 2001.

[180] N. Haiminen, et al. ,"Comparing segmentations by applying randomization techniques", BMC Bioinformatics, 7, 171,2007.

[181] I. Lyakhov, Krishnamachari, A.; Schneider, T., "Discovery of novel tumor suppressor p53 response elements using information theory", Nucleic Acids Res., vol.36, pp.3828–3833, 2008.

[182] P.E. Meyer, et al., "Information-Theoretic inference of large transcriptional regulatory networks", EURASIP J. Bioinform. Syst. Biol., 2007.

[183] L. Kovac, “Information and knowledge in biology: Time for reappraisal”, Plant Signal. Behav., vol.2, pp.65–73, 2007.

[184] G. Stormo, G.W. Hartzell, “Identifying protein-binding sites from unaligned DNA fragments”, PNAS, 86, pp.1183–1187, 1989.

[185] T. Schneider, G. Stormo, L.Gold, Ehrefeucht, “A. Information content of binding sites on nucleotide sequences”. J. Mol. Biol., pp.188, 415–431, 1986.

[186] T. Schneider, R. Stephens, “Sequence logos: A new way to display consensus sequences”. Nucleic Acids Res., vol.18, pp.6097–6100, 1990.

[187] T. Schneider, D. Mastronade,“Fast Multiple alignment of ungapped DNA sequences using information theory and a relaxation method”, Discrete Appl. Math.,vol.71, pp.259–268, 1996 .

[188] T. Bailey, C. Elkan, “Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Biopolymers”, In Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, Stanford, CA, USA, August 14–17, 1994.

[189] S. Mahony, P. Benos, “STAMP: a web tool for exploring DNA-binding motif similarities”, Nucleic Acids Res., 35, W253–W258, 2007.

[190] R. Shultzaberger, T. Schneider, “Using sequence logos and information analysis of Lrp DNA binding sites to investigate discrepancies between natural selection and SELEX”, Nucleic Acids Res., 27, 882–887, 1999.

[191] T. Schneider, “Strong minor groove base conservation in sequence logos implies DNA distortion or base flipping during replication and transcription initiation”, Nucleic Acids Res., 27, pp. 882–887, 2001.

[192] B. Korber, R. Farber, D.Wolpert, A.Lapedes, “Covariation of mutations in the V3 loop of human immunodeficiency virus Type I envelope protein: An information theoretic analysis”, PNAS, 90, pp.7176–7180, 1993.

[193] K. Sayood, F. Hoffman, C. Wood, “Use of Average Mutual Information for Studying Changes in HIV Populations”, In Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Minneapolis, MN, USA, September 3–6, 2009.

[194] H. Zhang, G. Orti, Q. Du, J. He, C. Kankasa, G. Bhat, C. Wood, "Phylogenetic and phenotypic analysis of HIV Type 1 Env gp120 in cases of Subtype C mother-to-child transmission", AIDS Res. Hum. Retrov., 18, pp.1415–1423, 2002.

[195] F. Hoffman, X. He, J. West, P. Lemey, C. Kankasa, C. Wood, "Genetic variation in mother-child acute seroconverter pairs from Zambia.",AIDS, 22, pp.817–824, 2008.

[196] B. Giraud, A. Lapedes, L.Liu, "Analysis of correlations between sites in models of protein sequences", Phys. Rev. E, 58, pp. 6312–6322,1998, .

[197] L. Martin, G. Gloor, S. Dunn, L. Wahl, "Using information theory to search for co-evolving residues in proteins", Bioinformatics, vol.21, pp.4116–4124, 2005 .

[198] I. Grosse, H.Herzel, S.Buldyrev, H.Stanley, "Species independence of mutual information in coding and noncoding regions", Phys. Rev. E, 61, pp.5624–5629, 2000.

[199] M. Bauer, "A Distance Measure for DNA Sequences", PhD thesis, University of Nebraska-Lincoln, Lincoln, NE, USA, January 1, 2001.

[200] M. Bauer, S. Schuster, K. Sayood, "The average mutual information profile as a genomic signature", BMC Bioinf., http://www.biomedcentral.com/1471-2105/9/48, accessed on December 29, 2009.

[201] M. Berryman, A. Allison, D. Abbot, "Mutual information for examining correlations in DNA",Fluct. Noise Lett., 4, pp.237–246, 2004 .

[202] D. Holste, I. Grosse, S. Beirer, P. Schieg, H. Herzel, "Repeats and correlations in human DNA sequences", Phys. Rev. E, 67, 061913:1–061913:7, 2003.

[203] H. Otu, K. Sayood, "A divide and conquer approach to sequence assembly", Bioinformatics, vol.19, pp.22–29, 2003.

[204] Y. Linde, A. Buzo, R.M. Gray, "An algorithm for vector quantization design", IEEE Trans. Commun., COM-28, pp.84–95, 1980.

[205] A. Butte, I. Kohane, "Mutual Information Relevance Networks: Functional Genomic Clustering Using Pairwise Entropy Measurements", In Proceedings Pacific Symposium on Biocomputing 2000, Oahu, HI, USA, January 4-9, 2000.

[206] R. Steur, J. Kurths, C. Daub, J. Wiese, J. Selbig, "The mututal information: Detecting and evaluating dependencies between variables", Bioinformatics, vol.18, S231–S240, 2002.

[207] J. Quackenbush, "Computational analysis of microarray data", Nat. Rev. Genet., vol.2, pp.418–427, 2001.

[208] N. Chomsky, "Logical Structure of Linguistic Theory", PhD thesis, University of Pennsylvania,Philadelphia, PA, USA, 1955.

[209] N. Chomsky, "On certain formal properties of grammars", Inform. Control, vol.2, pp.137–167.66, 1959.

[210] D. Chiang, A.K. Joshi, D.B. Searls, "Grammatical representations of macromolecular structure", J.Comput. Biol., vol.13, pp.1077–1100, 2006.

[211] Y. Sakakibara, M.Brown, R.Hughey, I.S.Mian, K.Sj¨olander, Underwood, R.C.; Haussler, D. ," Stochastic context-free grammars for tRNA modeling", Nucleic Acids Res., 22, pp.5112–5120, 1994.

[212] D.B. Searls, "The language of genes", Nature, 420, 211–217, 2002.

[213] M.Gheorghe, V.Mitrana, "A formal language-based approach in biology". Comp. Funct. Genom., vol.5,pp. 91–94, 2004.

[214] V. Brendel, H.G. Busse, "Genome structure described by formal languages", Nucleic Acids Res., vol.12,pp.2561–2568, 1984.

[215] T. Head, "Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors", Bull. Math. Biol., vol.49, pp.737–759, 1987.

[216] D.B.Searls, "The linguistics of DNA", Am. Sci., 80, pp.579–591, 1992.

[217] N. Abe, H. Mamitsuka, "Predicting protein secondary structure using stochastic tree grammars", Mach. Learn., 29, pp.275–301, 1997.

[218] J. Collado-Vides, "A transformational-grammar approach to the study of the regulation of gene expression", J. Theor. Biol., vol.136, pp.403–425, 1989.

[219] C.G. Nevill-Manning, "Inferring Sequential Structure", PhD thesis, University of Waikato, Waikato, New Zealand, 1996.

[220] Y. Sakakibara, "Learning context-free grammars using tabular representations", Pattern Recognit., 38, pp.1372–1383, 2005.

[221] K. Nakamura, M. Matsumoto, “Incremental learning of context free grammars based on bottom-up parsing and search”, Pattern Recognit., 38, pp.1384–1392, 2005.

[222] N.Cherniavsky, R.E. Ladner,” Grammar-based Compression of DNA Sequences”, Presented at the DIMACS Working Group on the Burrows-Wheeler Transform, DIMACS Center, Rutgers University, Piscataway, NJ, USA, August 19–20, 2004, http://www.cs.washington.edu/homes/nchernia/dnasequitur/dnasequitur.pdf, accessed on December 18, 2009.

[223] D.Russell, H.Otu, K.Sayood, “Grammar-based distance in progressive multiple sequence alignment”, BMC Bioinf., http://www.biomedcentral.com/1471-2105/9/306, accessed on December 29, 2009.

[224] S. Leung, C. Mellish, D. Robertson, “Basic gene grammars and DNA-chart parser for language processing of Escherichia coli promotor DNA sequences”, Bioinformatics, 17, 226236, 2001.

[225] S. Grumbach and F. Tahi, “Compression of DNA Sequences,” presented at Data Compression Conference, DCC'93, pp. 340-350, Snowbird, UT, USA, 1993.

[226] X. Chen, M. Li, B. Ma, and J. Tromp. “DNACompress: fast and effective DNA sequence compression”. Bioinformatics, vol.18, pp.1696–1698, 2002.

[227] I. Tabus, G. Korodi, and J. Rissanen, “DNA sequence compression using the normalized maximum likelihood model for discrete regression,” in Proc. Data Compression Conf. (DCC), 2003.

[228] M. Li, JH. Badger, X. Chen, S. Kwong, P. Kearney, HY. Zhang, “An information-based sequence distance and its application to whole mitochondrial genome phylogeny”, Bioinformatics, 17:149-154, 2001.

[229] A. Kocsor, A. Kertesz-Farkas, L. Kajan, S. Pongor, “Application of compression-based distance measures to protein sequence classification: a methodological study”. Bioinformatics, vol.22, pp.407-412, 2006.

[230] X. Chen, Kwong S, Li M: “A compression algorithm for DNA sequences”. IEEE Engineering in Medicine and Biology Magazine, vol.20, pp.61-66, 2001.

[231] F. M. J.Willems, Y. M. Shtrakov and T. J.Tjalkens, "The Context Tree Weighting Method: Basic Properties", IEEE Trans. Inform. Theory, IT-41(3), pp 653-664, 1995.

[232] B. Behzadi, F. L. Fessant, "DNA compression challenge revisited: A dynamic programming approach", In Lect. Notes Comput. SC; Springer: Berlin/Heidelberg, Germany, CPM, pp.190-200, 2005.

[233] J.A. Storer, and T.G. Szymanski, " Data compression via textual substitution," J. ACM, 29, 928–951,1982.

[234] D. Adjeroh, and F. Nan, , "On compressibility of protein sequences", In Proceedings of the IEEE Data Compression Conference (DCC), IEEE Computer Society, pp. 422–434, 2006.

[235] D. Adjeroh, Y. Zhang, A. Mukherjee, M. Powell, and T. Bell, "DNA Sequence Compression Using the Burrows-Wheeler Transform," presented at IEEE Computer Society Bioinformatics Conference (CSB'02), 2002.

[236] VB. Strelets, HA. Lim, "Compression of Protein-Sequence Databases, Comput Appl Biosci, 11:557-561, 1995.

[237] CH. Wu, LSL. Yeh, HZ. Huang, L. Arminski, J Castro-Alvear, YX .Chen, ZZ. Hu, P. Kourtesis, RS. Ledley, BE. Suzek, CR. Vinayaka, J. Zhang, WC. Barker, "The Protein Information Resource", Nucleic Acids Res, 31, pp.345-347, 2003.

[238] P. Katz, "PKZIP. 1.1th edition." [http://www.pkware.com/]. Milwaukee, WI, USA, PKWARE, Inc, 1990.

[239] WZ. Li, L. Jaroszewski, A. Godzik, "Tolerating some redundancy significantly speeds up clustering of large protein databases", Bioinformatics, vol.18,pp.:77-82, 2002.

[240] WZ .Li, A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences", Bioinformatics, vol.22, pp.1658-1659 , 2006.

[241] P. J. S. G. Ferreira, A. J. R. Neves, V. Afreixo, and A. J. Pinho, "Exploring Three-Base Periodicity for DNA Compression and Modeling," presented at ICASSP 2006, vol. 5, Toulouse, France, 2006.

[242] A. Apostolico and S. Leonardi, “Compression of biological sequences by greedy off-line textual substitution,” presented at Data Compression Conference, pp. 143-152, 2000.

[243] G. Korodi and I. Tabus, “An efficient normalized maximum likelihood algorithm for DNA sequence compression," ACM Trasactions on Information Systems, vol. 23, pp. 3-34, 2005.

[244] I. Sadeh, “Universal data compression algorithm based on approximate string matching”, Probability in the Engineering and Informational Sciences, 10:465{486, 1996.

[245] M. Joe Gatewood and J. Callum Bell, ”Efficient Delta Compression and Comparisonof DNA Sequence Data”, patent 505 438 1881 , 2007.

[246] C. G. Nevill-Manning and I. H. Witten, “Protein is incompressible”, In Proc. Data Compression Conference, pp. 257–266, 1999.

[247] N. Krasnogor, D.A.Pelta, “Measuring the similarity of protein structures by means of the universal similarity metric”, Bioinformatics, 20, pp.1015–1021, 2004.

[248] D. Pelta, J.R.Gonzales, N.Krasnogor, “Protein Structure Comparison Through Fuzzy Contact Maps and the Universal Similarity Metric”, In Proceedings of the Joint 4thConference of the European Society for Fuzzy Logic and Technology (EUSFLAT) and the 11th Rencontres Francophones sur la Logique Floue et ses Applications (LFA), Barcelona, Spain, September 7–9, 2005.

[249] D. Loewenstern, H. Hirsh, P. Yianilos, M. Noordewier, “DNA Sequence Classification Using Compression-Based Induction”, DIMACS Technical Report 95-04; Rutgers University: New Brunswick, NJ 08903, USA, 1995.

[250] J. Rocha, F. Rossello, J. Segura, “Compression ratios based on the universal similarity metric still yield protein distances far from CATH distances”, 2006, arXiv:q-bio/0603007. arXiv.org e-Print archive, http://arxiv.org/abs/q-bio/0603007, accessed on December 18, 2009.

[251] S. Christley, Y. Lu, C. Li, X. Xie “Human genomes as email attachments”. Bioinformatics , 25,pp.274-5, 2009.

[252] C. Brandon, D. C. Wallace and P. Baldi, "Data Structures and Compression algorithms for Genomic Sequence data," Bioinformatics, vol. 25, no. 14, pp. 1731-1738, 2009.

[253] C. Wang and D.Zhang, "A novel compression tool for efficient storage of genome resequencing data," Nucleic Acids Research, January 2011.

[254] H. Do Kim, Ju-Han Kim, "DNA Data Compression Based on the Whole Genome Sequence," Journal of Convergence Information Technology, vol. 4, no. 3, 2009.

[255] C. Kozanitis, C. Saunders, S. Kruglyak, V. Bafna, and G.Varghese, "Compressing Genomic Sequence Fragments Using SLIMGENE," RECCOMB, 2010.

[256] Markus Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane, et al., **"**Efficient storage of high throughput sequencing data using reference-based compression", Genome Res. published online January 18, 2011.

[257] SW.Golomb "Run-length encodings", IEEE Transactions on Information Theory 12(3):399-401, 1966.

[258] D.Sculley, and C.Brodley, "Compression and machine learning: a new perspective on feature space vectors", In Proceedings of the IEEE Data Compression Conference (DCC). IEEE Computer Society, pp. 332–332, 2006.

[259] M. Brandon, et al., "MITOMAP: a human mitochondrial genome database," Nucleic Acids Res., vol. 33, pp. D611–D613, 2005.

[260] David Salomon ,Giovanni Motta and David Bryant, "Data Compression: The Complete Reference. 4th Edition". Published by Springer, Dec 2006.

[261] Gailly, Jean-loup, Adler, Mark, "zlib Applications", 18-04-2002.

[262] Terry Welch, "A Technique for High-Performance Data Compression", IEEE Computer, June, p. 8–19, 1984.

[263] B. Jung, , and W.P. Burleson, " Efficient VLSI for Lempel-Ziv compression in wireless data communication networks", IEEE Transactions on Very Large Scale lntegration(VLSI) Systems, Vol. 6, No. 3, pp. 475-483, September 1998.

[264] B.D. Tseng, and W.C. Miller, "On computing discrete cosine transform", IEEE Transactions on Computers, Vol. C-27, No. 10, pp. 966-968, 1978.

[265] B.G. Lee, “A new algorithm to compute the discrete cosine transform”, In Proceedings of International Conference on Acoustics, Speech and Signal Process, Vol. ASSP, pp. 1243-1245, December 1984.

[266] C.Cafforio, and F. Rocca, “Methods for measuring small displacements of television images”, IEEE Transactions on Information Theory, Vol. IT-22, pp. 573-579, September 1976.

[267] C. RICK, “A new flexible algorithm for the longest common subsequence problem”, In Proceeding o f the 6th Annual Symposium on Combinatorial Patterm Matching Espoo, Finland, 5-7 July 1995.

[268] R. WAGNER, and M.FISCHER, “The string-to-string correction problem”, Journal of the ACM 21, 1, pp.168–173, Jan. 1973.

[269] A. EHRENFEUCHT, and D.HAUSSLER, “A new distance metric on strings computable in linear time”, Discrete Applied Mathematics 20, pp.191–203, 1988.

[270] W. MILLER, and E. W. MYERS, “A file comparison program”, Software – Practice and Experience 15, 11), pp.1025–1040, Nov. 1985.

[271] W. F. TICHY, “The string-to-string correction problem with block move”, ACM Transactions on Computer Systems 2, 4, Nov. 1984.

[272] W. F. TICHY, “RCS – A system for version control”, Software – Practice and Experience 15, 7, 637–654, July 1985.

[273] R. Cilibrasi and P. M. B. Vitanyi, “ Clustring by Compression, ” IEEE Trans. Information Theory, vol. 51, no.4, pp.1523-1545, April 2005 .

[274] Kerstin Lindblad-Toh, “Genome sequencing”, Nature 428, 475-476, April 2004.

[275] P. Fenwick, “Block Sorting Text Compression | Final Report,” Technical Report 130, The University of Auckland Department of Computer Science, March 1996.

[276] M. L. Metzker. “Sequencing technologies — the next generation”, Nat. Rev. Genet., 11(1):31–46, Jan. 2010.

[277] E.Giladi, M.G.Walker, J.Z. Wang, and W.Volkmuth, “SST: An Algorithm for Searching Sequence Databases in Time Proportional to the Logarithm of the Database Size”, in RECOMB2000 The Fourth Annual International Conference on Computational Molecular Biology Tokyo, Japan, April 8 - 11, 2000.

Publications:

1-"GENOMIC SEQUENCES DIFFERENTIAL COMPREESION MODEL", Conference paper in National Radio Science conference, Academy of Scientific Research and Technology Faculty of Electronic Engineering, Menoufia Univ., Menouf. Egypt, 27th National Radio Science Conference 16-18 March 2010.

2-"DNA LOSSLESS DIFFERENTIAL COMPRESSION ALGORITHM BASED ON SIMILARITY OF GENOMIC SEQUENCE DATABASE", International Journal of Computer Science & Information Technology (IJCSIT) Vol. 3, No. 4, August 2011.

الفصل الخامس: يشرح هذا الفصل النتائج لكل طريقة و مناقشة المميزات و العيوب من حيث معدل الضغط و مساحة التوفير والوقت المنفذ لكل طريقة. النتائج توضح بأن اختيار المرجع المتغير اعتمادا على أقل انتروبى يحسن الضغط عن استخدام المرجع الثابت لكل مجموعة.

الفصل السادس: يقدم هذا الفصل موجز عن البحث وكذلك الاستنتاجات الكلية من الرسالة و تصورنا للعمل المستقبلى فى مجال هذا البحث.

يقسم مجموعة كاملة من البيانات إلى مجموعات صغيرة لضغط كل مجموعة صغيرة باستخدام مراجع متغيرة تم اختيارهم على أساس الانتروبى . تم تطبيق الطريقة على ثلاث مجموعات بيانات مختلفة للتسلسل الجيني ،وحققت معدل ضغط بصل إلى 195 فى مقابل توفير مساحة 99.4 ٪. هذه الطريقة هى ناجحة فى بعض التطبيقات مثل اختيار المرجع ، وتحديث مجموعة بيانات جديدة ، وضغط بيانات غير معروفة لمجموعة من الأنواع المختلفة فى حالة استخدام مجموعة بيانات صغيرة . ونتيجة لذلك ، فإنه لا يقبل استخدام مجموعات البيانات الضخمة في هذه التطبيقات. نبين تجريبيا أن هذه التطبيقات لهذه الطريقة تأخذ كمية باهظة من الوقت لمجموعات بيانات كبيرة مثل أكبر من 1 ميغا بايت ونخلص إلى أن أداء الطريقة غير مقبول لبعض التطبيقات.

باستخدام تقنيات الترميز سابقا ، طريقتنا توفر حلا بسيطة لتخزين مجموعات البيانات المتشابه من تسلسل الجينى من أي حجم ،وتوفر سرعة مناسبة للضغط وفك الضغط. وأظهر تحليل الطريقة الأداء الجيد على ترميز العملية المقترحة التي تولد الفرق بين التسلسل الجينى ، وسلبا على عمليات أخرى. أيضا ، البساطة والمرونة في طريقة الضغط الفرقى جعله أداة لا تقدر بثمن لضغط الحمض النووي في الابحاث الطبية. كلا من تقدير الانتروبى وعملية التكويد هما اتحاد مناسب لكفاءة الضغط ولتكون اختيار للمرجع فى عديد من التطبيقات لمجموعات بيانات صغيرة. وبالتالي ، فإننا ما زلنا بحاجة لبعض التعديلات لتوفير حل لضغط فائق لمدخلات ضخمة مع أقل وقت بواسطة مزيد من التجارب.

وتحتوى الرسالة على ستة فصول وهى:

الفصل الاول: يعطى وصفا موجزا لتعريف المشكلة و موضوعية الطرح و يقدم ملخص لمحتوى الرسالة.

الفصل الثانى: يعطى وصفا لصفات الحمض النووى و استخدامة فى مجال الابحاث الطبية الحيوية. كما يعرض طرق تحليل و توصيف الحمض النووى ليساعد فى اختيار أفضل الطرق لضغط مجموعة كبيرة من التسلسل الجينى.

الفصل الثالث: يعرض هذا الفصل بداية خطة سير العمل لضغط الحمض النووى و يناقش مكونات عملية الضغط لكل طريقة سابقة مستخدمة سواء ضغط كل تسلسل بمفردة أوضغط مجموعة من التسلسل الجيني. و يركز على طرق لضغط قواعد البيانات الضخمة المتشابة للتسلسل الجيني اعتمادا على وجود المرجع.

الفصل الرابع: يشرح هذا الفصل الطريقة المقترحة لضغط قواعد البيانات الضخمة للتسلسل الجيني التى تعتمد على تخزين الاختلافات وأمكان الاختلافات اعتمادا على وجود مرجع بين التسلسلات الجينية. ويتناول طريقة الضغط الفرقى من خلال اثنين من الحلول المختلفة و اسخدامة فى بعض التطبيقات مثل اختيار المرجع المناسب وتحديث المعلومات و ضغط مجموعة من الاصناف المختلفة عن طريق تعريف سلسلة جديدة وتحديد علاقتها مع المجموعة.

ملخـص الرسـالـة

تحليل كميات ضخمة من البيانات الجينومية التي تم توليدها باستمرار يضع عددا من المشاكل الصعبة فى مجالات الابحاث، وبخاصة في مجالات البيولوجيا الحاسوبية والمعلوماتية الحيوية الطبية. واحدة من هذه التحديات هى مشكلة ضغط التسلسل للحمض النووي بطريقة فعالة ، وذلك لأن هذه التسلسلات قد تكون كبيرة مثل الجينوم بأكمله (على سبيل المثال ، يتكون الجينوم البشري من حوالي 3000 مليون قاعدة). الوضع المعقد لتسلسل الحمض النووي هو تسلسل رقمي لمختلف الاطوال التي تدل على القدرة على التميز لكل قاعدة من الحمض النووي. ملامح الحمض النووي هى مهمة لأنها تسمح ، على سبيل المثال ، بالبحث عن هياكل المتكررة داخل الكروموسوم أو عبر عدة كروموسومات. غالبا ما ترتبط هذه الهياكل مع المهام الوظيفية للحمض النووي. وعلاوة على ذلك ، يمكن أيضا استخدامها في تحديد المسافات بين الاطوار ، وبالتالي في بناء أشجار التطور.

ومن المثير للاهتمام ، أن حساب الضغط الفرقى يحقق كفاءة ضغط لقواعد بيانات كاملة لسلاسل بدلا من الطرق السابقة للضغط. مخططات الضغط الفرقى هى ضغط مجموعات كاملة من سلاسل متشابة من نفس النوع بواسطة دراسة العلاقات بين السلاسل لتخزين الاختلافات بين تسلسل الجينوم والتسلسل المرجعي الذى تم استخدامة في هذا العمل. لتحسين طرق ضغط البيانات عن طريق محاولة للعثورعلى تقنيات لتوصيف فعال التسلسل الفرقى. ونحن نركز دراستنا على مجموعات كبيرة من تسلسلاث التي تنتمي إلى نفس الفئة. على سبيل المثال ، إذا كان اثنان من التسلسلات الوراثية بينهم أكثر من 99 ٪ تشابة، ولذلك الأكثر كفاءة هو تخزين الاختلافات بينهم ، وفي هذه الحالة هى 1 ٪ فقط يجب أن يخزن بدل من تخزين كل تسلسل على حدة. عملية فك الضغط هى مشابهه لعملية الضغط ، ولكن عكس عملية الضغط لتحقبق الضغط بدون الفقد فى البيانات والرجوع الى أصل البيانات. النتائج التجريبية وضعت الاختلافات بين السلاسل وأماكن الاختلاف كمكون جديد في فن ضغط تسلسل الحمض النووي ، وتحقيق مساحة أقل ووقت أقل ، من سابقاتها.

في هذا العمل ، وقد استعرضنا طرق الضغط السابقة ، وقدمنا طريقة جديدة لترميزالتسلسل الفرقى بدون الاعتماد على احصائيات معبنة للتسلسل. والطرق السابقة تشمل العثور على محاذاة لسلاسل مطابقة ، وتقدير الإنتروبي (مقياس درجة الفوضى) ، وأساليب التكويد. بالاضافة إلى هذه الطرق السابقة، اقتراحنا بعض الميزات مثل النمذجة الجديدة لتوصيف مجموعات البيانات الضخمة كرمز للعملية للتقليل من المعلومات ، واستخدام الانتروبى كمؤشر التسلسل المرجعي لتحسين الضغط ، وتحديث التقنية لتسمح بإضافة مجموعة تسلسل جديدة.

طريقة الضغط الفرقى المقترحة تبين أدائها فى الضغط لتكون الأمثل في ظل بعض الافتراضات البسطية. استخدمنا هذه الطريقة لضغط البيانات الضخمة بواسطة اثنين من الحلول كنموذج للمقارنة بينهم وبين الطرق الأخرى السابقة للضغط. الحل الأول يعامل كل مجموعة بيانات كوحدة واحدة باستخدام تسلسل قياسى مراجعى واحد. الحل الثاني

مهنـــــــــــدس: هبه محمود محمد عفيفى
تاريـخ الميـــلاد: 1979/7 /21
الجنسيـــــــــة: مصرية
تاريخ التسجيل: 2008 / 6 /17
تـــاريخ المنـــح: / / (يكتب بمعرفه إدارة الدراسات العليا)
القســـــــــــم: الهندسه الحيويه الطبيه و المنظومات
الدرجـــــــــة: دكتوراه

المشرفون : ا.د ياسر مصطفى قدح
ا.م.د منال عبد الواحد
د. محمد اسلام

الممتحنـــــون : ا.د عبد الله سيد احمد
ا.د محمد إبراهيم العدوى
ا.د ياسر مصطفى قدح
ا.م.د منال عبد الواحد

عنـــوان الرسالــة : **(نظام الضغط الفرقى بدون الفقد فى قواعد البيانات للتسلسل الجيني)**

الكلمات الدالة : **البيانات الجينومية، ضغط التسلسل للحمض النووي ، أشجار التطور.**

ملخـــص البحـــث :

تحليل كميات ضخمة من البيانات الجينومية التي تم توليدها باستمرار يضع عددا من المشاكل الصعبة فى مجالات الابحاث، وبخاصة في مجالات البيولوجيا الحاسوبية والمعلوماتية الحيوية الطبية. واحدة من هذه التحديات هى مشكلة ضغط التسلسل للحمض النووي بطريقة فعالة ، وذلك لان هذه التسلسلات قد تكون كبيرة مثل الجينوم باكمله (على سبيل المثال ، يتكون الجينوم البشري من حوالي 3000 مليون قاعدة). الوضع المعقد لتسلسل الحمض النووي هو تسلسل رقمي لمختلف الاطوال التي تدل على القدرة على التميز لكل قاعدة من الحمض النووي. ملامح الحمض النووي هى مهمة لأنها تسمح ، على سبيل المثال ، بالبحث عن هياكل المتكررة داخل الكروموسوم أو عبر عدة كروموسومات. غالبا ما ترتبط هذه الهياكل مع المهام الوظيفية لحمض النووي. وعلاوة على ذلك ، يمكن أيضا استخدامها في تحديد المسافات بين الاطوار ، وبالتالي في بناء أشجار التطور.

نظام الضغط الفرقى بدون الفقد فى قواعد البيانات للتسلسل الجيني

اعداد

هبه محمود محمد عفيفى

قسم الهندسة الحيوية الطبية و المنظومات

كلية الهندسة - جامعة القاهرة

رسالة مقدمة الى كلية الهندسة – جامعة القاهرة

كجزء من متطلبات الحصول على درجه الدكتوراة

فى الهندسة الحيوية الطبية و المنظومات

يعتمد من لجنة الممتحنين:

الأستاذ الدكتور : عبد الله سيد احمد محمد قسم الهندسة الحيوية الطبية و المنظومات- جامعة القاهرة	**ممتحن**
الأستاذ الدكتور : محمد ابراهيم العدوى وكيل كلية الهندسة لشئون الطلاب –جامعة حلوان	**ممتحن**
الأستاذ الدكتور : ياسر مصطفى قدح قسم الهندسة الحيوية الطبية و المنظومات- جامعة القاهرة	**المشرف الرئيسى**
الأستاذ المساعد الدكتور : منال عبد الواحد قسم الهندسة الحيوية الطبية و المنظومات- جامعة القاهرة	**المشرف**

كلية الهندسة - جامعة القاهرة

الجيزة – جمهورية مصر العربية

2012

نظام الضغط الفرقى بدون الفقد فى قواعد البيانات للتسلسل الجينـي

اعداد

هبه محمود محمد عفيفى

قسم الهندسة الحيوية الطبية و المنظومات

كلية الهندسة - جامعة القاهرة

رسالــة مقــدمــة الى كليــة الهندســة – جـامعــة القـاهــرة

كجـزء من متطـلبـات الحصــول على درجه الدكتوراة

فى الهندســة الحيــوية الـطبيــة و المنظومــات

تحت اشـراف

أ. د/ ياسر مصطفى قدح
قسم الهندسة الطبية والمنظومات
كلية الهندسة - جامعة القاهرة

أ .م.د/ منال عبد الواحد
قسم الهندسة الطبية والمنظومات
كلية الهندسة - جامعة القاهرة

د. / محمد اسلام
قسم الهندس الطبية والمنظومات
كلية الهندسة - جامعة القاهرة

كلية الهندسة - جامعة القاهرة
الجيزة – جمهورية مصر العربية
2012

نظام الضغط الفرقى بدون الفقد فى قواعد البيانات للتسلسل الجيني

اعداد

هبه محمود محمد عفيفى

قسم الهندسة الحيوية الطبية و المنظومات

كلية الهندسة - جامعة القاهرة

رسالة مقدمة الى كلية الهندسة – جامعة القاهرة

كجزء من متطلبات الحصول على درجه الدكتوراة

فى الهندسة الحيوية الطبية و المنظومات

كلية الهندسة - جامعة القاهرة

الجيزة – جمهورية مصر العربية

2012