The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

An Approach to Multiple DNA Sequences Compression

Wu Choi Ping Paula

A thesis submitted in partial fulfillment of the requirements for the

Degree of Master of Philosophy

February 2009

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

__WU CHOI PING PAULA_____ (Name of student)

Summary of contribution

The original contributions reported in this thesis are as follows:

1. A similarity study of the sixteen chromosome sequences of *Saccharomyces cerevisiae.*

   We have performed an extensive study of the sixteen chromosome sequences of *Saccharomyces cerevisiae* (*S. cerevisiae*). In particular, similar subsequences have been searched throughout these sixteen sequences. Their location and length have been investigated so as to quantify the potential gain in cross-sequence compression. Our study indicates that there are significant cross-sequence similarities among these sixteen sequences. Also, the similar subsequences from different chromosome sequences do not overlap in position, hence, two types of cross-chromosomal predictions are proposed to improve the overall sequences compressibility.

2. A new algorithm for multiple sequence compression

   All state-of-the-art compression algorithms are based on finding similar subsequences within the current sequence only. The average bits per base (bpb) is reduced by 0.08 in *S. cerevisiae* as compared with the no-compression case. We have proposed a multiple sequence compression algorithm that compresses a number of sequences together to take into account both self-sequence similarity and cross-sequence similarity. The experimental results show that our proposed algorithm can consistently reduce the average bpb used while maintaining a low computational complexity.

3. The implementation of a software tool for multiple sequence compression

   A software tool written using Matlab has been developed that allows users to compress a number of sequences together. Information, such as the execution time and the bpb of compressing with and without a reference sequence, can also be shown by the software tool.

Abstract

Deoxyribonucleic acid (DNA) technologies have been widely used in genetic engineering, forensics and anthropology applications. A DNA sequence is a long sequence consisting of four types of nucleotides called bases. The number of bases of the 24 chromosomes in humans ranges from 50 to 250 million. Without any compression, two bits per base are required for storage which results in a large number of bits for encoding DNA sequences. An effective way to compress these sequences is thus desirable in order to reduce the storage space required.

General-purpose compression tools such as gzip use more than two bits to encode a base. This is because these tools do not make use of the special characteristics of a DNA sequence. For example, it is well known that a DNA sequence has long-term correlation in that subsequences in different regions of a DNA sequence are similar to each other. State-of-the-art DNA compression schemes all rely on exploiting this long-term correlation. In particular, repetitions within the DNA sequence are searched so that similar subsequences can be encoded with reference to each other. For these DNA compression schemes, the reduced average bits per base (bpb) are around 0.25 for the benchmark DNA sequences. Thus, the compression gain is not large.

It is well known that there are similarities among different chromosome sequences. All state-of-the-art compression algorithms exploit only self-sequence similarities, and specifically ignore the cross-sequence similarities. We have performed a thorough study of similarities within the same chromosome sequence as well as similarities among different chromosome sequences. These similarities are characterized by the existence of similar subsequences in different chromosome sequences. Our study indicates that these cross-sequence similarities are often significant when compared to self-sequence similarities. In the experimental results from the sixteen chromosome sequences in *S. cerevisiae*, the average repetitive length from cross-sequence prediction was almost fourteen times of that from self-sequence prediction.

To make use of both self-sequence and cross-sequence similarities in DNA compression, we have proposed a multi-sequence compression algorithm. Our scheme compresses two or more sequences together so that similar subsequences found among multiple sequences can be encoded together. In this scheme, we first create a list of similar subsequences, from either the reference sequence or the current sequence which are ordered according to their importance. The list is then modified by removing the overlapping

similar subsequences. After reordering the list according to their position and removing similar subsequences from the current sequence, Arith-2 coder is used to further compress the non-repetitive regions.

Our experimental results show that compressing a sequence with reference to another sequence achieves an average of 5.5% saving in bpb as compared with that without reference to another sequence, hence the bpb of compressing two chromosome sequences together is consistently better than that of compressing them separately. This shows the importance of cross-sequence similarities. We have also extended the cross-sequence predictions to more than two chromosome sequences. We found that there is always additional savings in bpb by compressing a number of chromosome sequences together. Since by making use of the cross-sequence similarities, our proposed multiple sequence compression algorithm can outperform other single sequence-based compression algorithms.

Table of Contents

List of Figures

List of Tables

1 Introduction

Due to the recent progress in human genome sequencing, there is a great surge in demand for storing and transmitting deoxyribonucleic acid (also known as DNA) sequences. Compression thus becomes essential in order to reduce the size of DNA sequence to save storage space and transmission time.

DNA sequences contain four kinds of nucleotides or bases: adenine (A), cytosine (C), guanine (G) and thymine (T). Without compression, two bits are required for representing each base. If general compression tools such as gzip are used, it has been found that more than two bits are often needed to represent a base [1]. Consequently, these general compression tools cannot compress, but rather expand the DNA sequences [1-2]. Compression algorithms that are designed specifically for DNA sequences thus need to be developed.

DNA sequences are not purely random sequences. If these sequences were totally random, the most efficient and logical way to store them would be using two bits per base (bpb). Because DNA contains genetic information and codes for protein in living organisms, it must contain a logical organization and some redundancies could be exploited through the compression strategies. The main source of redundancy in DNA sequences lies in the long-term repetitions,

in the form of either approximate repeats or complementary palindromes; as a result, current DNA compression algorithms focus heavily on the exploitation of repetitions within the DNA sequence. However, even though these DNA compression algorithms obtain better results than general purpose compression algorithms, the compression ratios are not high [2]. For example, the context tree weighting method [21] achieved an average 0.05 bpb reduction for the sixteen chromosome sequences of *Saccharomyces cerevisiae* (*S. cerevisiae*) when compared to the no-compression case.

To achieve further compression in encoding DNA sequences, it is important to understand the special properties associated with DNA sequences and to fully utilize these properties in compressing them [3]. The main objectives of this thesis are to study the similarity among different DNA sequences and to investigate how the similarity can be used effectively for DNA sequence compression.

1.1 Problems

Current DNA compression schemes are based on searching repetitions within the DNA sequence. These repetitions can be in the form of exact matches or approximate matches. Exact matches mean that the repetitive sequence is

exactly the same as the original one, while approximate matches mean the two sequences look identical only if substitutions, insertions and/or deletion of certain bases are performed during matching [3-7]. Note that DNA sequences are very long. For example, the lengths of the 24 chromosomes in humans range from 50 to 250 million base pairs. Thus, the time to find exact or approximate matches can also be very long and searching for all repeats in a DNA sequence is not a trivial task [3]. Some researchers have modified the searching strategy from greedy algorithms [2, 3, 8-11] to dynamic programming in order to reduce the searching time [2]. Tools for finding these exact and approximate matches are available freely [27-28].

In the field of video compression, images can be compressed either as an I-frame or as a P-frame [12-13]. The I-frame means that an image is intra-coded and redundancy is exploited within itself only. By contrast, the P-frame means that an image is inter-coded and redundancy is exploited between two consecutive images. The P-frame consistently has a better compression ratio than the I-frame. As for DNA compression, current algorithms are analogous to intra-coded image compression as redundant information is exploited only within the current DNA sequence. Therefore, besides self-sequence redundancy, cross-sequence redundancy can also be exploited in DNA sequence compression.

Although cross-sequence similarity is well known and is the basis of sequence analysis algorithms, such as multiple sequence alignment or phylogenetic analysis, the idea of exploiting this information specifically for DNA sequence compression is novel. These similarities are characterized by the existence of similar subsequences among different DNA sequences. The longer the similar subsequences are, the higher the cross-sequence similarities are. While only a modest compression ratio might be achieved for one DNA sequence, we hypothesized that a higher compression ratio can be achieved for multiple sequence compression since multiple sequence compression can benefit from both self-sequence similarity and cross-sequence similarities.

1.2 Objectives

This thesis attempts to give a quantitative analysis of sequence similarities and develop an effective compression algorithm for DNA sequences. In particular, we will 1) study the self-sequence similarities as well as cross-sequence similarities among different chromosome sequences of *S. cerevisiae*; 2) investigate the lengths and locations of similar subsequences found at various chromosome sequences; 3) develop a strategy to combine both self-sequence and cross-sequence similarities to achieve compression; and finally, 4) perform a comparative study with some existing DNA compression schemes such as CTW

[21] and GenCompress [4-6] to see the effectiveness of incorporating cross-sequence similarities in compression.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter Two reviews the literature on DNA sequences and compression schemes. This includes reviews on the characteristics of DNA sequences, some existing DNA compression algorithms that take into account DNA sequence characteristics, and some software tools for finding repetitions in DNA sequences. Chapter Three provides a detailed study of the similarities among DNA sequences. We discuss the importance of cross-sequence similarities in terms of their repetitive length and repeated location as compared with self-sequence similarity through some experiments on the sixteen chromosome sequences of *S. cerevisiae*. Chapter Four presents the structure of our proposed multiple sequence compression algorithm. Our proposed algorithm first searches for all the repeats among the different DNA sequences; these repeats are then sorted according to their significance in bits reduction. After removing all the repeats in different sequences, an entropy coder is used to compress the remaining non-repeating regions. In Chapter Five, we evaluate our proposed algorithms by experimenting on real datasets. The results are also compared with some

existing algorithms such as CTW and GenCompress. Chapter Six concludes our work with suggestions on the future development of our proposed multiple sequence compression algorithm.

2 Literature Review

Deoxyribonucleic acid (DNA) is a molecule composed of deoxyribonucleotides connected by phosphodiester linkages [8]. The genome is the complete DNA sequence of a living organism. Regions in a genome that code for proteins are known as genes. The largest publicly accessible DNA data are maintained in: GenBank (National Center for Biotechnology Information Genetic Databank) [14], EMBL (European Molecular Biology Laboratory) [15], and DDJB (DNA Database of Japan) [16]. Each of these databases shares their information with the others. In February 2008, GenBank reported that there were approximately 85,759,586,764 bases in 82,853,685 DNA sequence records in the traditional GenBank database, and 108,635,736,141 bases in 27,439,206 DNA sequence records in the WGS (Whole Genome Shotgun) division [14].

A large number of DNA sequences have been stored in these databases, and the size of the databases is expected to increase exponentially. Compression is hence desirable to reduce the storage requirements as well as the transmission time. This chapter first gives a brief review about the characteristics of DNA sequences, and then some existing DNA compression algorithms that exploit these DNA characteristics are presented.

2.1 Characteristics of DNA Sequences

DNA is a long sequence consisting of four kinds of nucleotides: adenine (A), cytosine (C), guanine (G) and thymine (T). It is in the form of a double helix held together by hydrogen bonds. The nucleotides (A, T) and (C, G) are complement pairs, as shown in Table 1. Each nucleotide in one DNA strand always binds to its complementary nucleotide in another strand. The two strands are biologically similar to each other. Thus, only one strand needs to be encoded while another strand can be deduced from this strand.

Figure 1. An example of a DNA sequence. [18]

Tab1e 1. Four types of nucleotides, Adenine (A), Guanine (G), Thymine (T) and Cytosine (C), and their complements.

Bases	Notation	Complement
Adenine	A	T
Cytosine	C	G
Guanine	G	C
Thymine	T	A

DNA sequences are not random sequences. They contain long-term repetitions in which subsequences that are thousands of bases apart could be similar to each other. There are two forms of long-term repetitions, namely repeats and complementary palindromes. These are often exploited by DNA sequence-oriented compression algorithms.

2.1.1 Repeats

Repeats include exact matches and approximate matches with some operations such as substitution, deletion and insertion. An exact match means two subsequences consist of identical nucleotides along the whole subsequence. Approximate matches with substitution, deletion and insertion, are illustrated in Figures 2(a), 2(b) and 2(c), respectively. In Figure 2(a), the DNA sequence is "ACGCTTACGCAT". The first six nucleotides, i.e., "ACGCTT", form the first subsequence and the last six nucleotides, i.e., "ACGCAT" form the second subsequence. The second subsequence can be obtained from the first subsequence if the 5^{th} base "T" in the first subsequence is substituted by "A". This is called an approximate match with substitution. In Figure 2(b), the DNA sequence is "ACGCTACGCAT", the first subsequence is formed from the first five nucleotides, i.e., "ACGCT" and the second subsequence is formed from the last six nucleotides, i.e., "ACGCAT". The second subsequence can be obtained

if the base "A" is inserted between the 4th and the 5th bases of the first subsequence. The two subsequences can thus be matched with an insertion operation. In Figure 2(c), the DNA sequence is "ACGCTTACGCT", the first subsequence is formed from the first six nucleotides, i.e., "ACGCTT" and the second subsequence is formed from the last five nucleotides, i.e., "ACGCT". The second subsequence is obtained if the 5th base "T" in the first subsequence is deleted. This is called an approximate match with deletion.

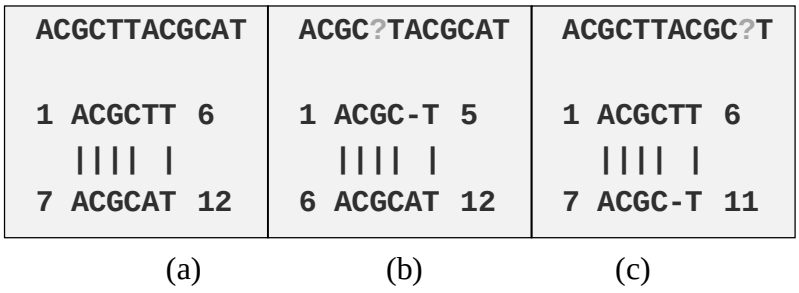


Figure 2. Examples of (a) substitution, (b) insertion and (c) deletion in approximate matches. The sequence is a part of a DNA sequence.

2.1.2 Complementary Palindromes

The complementary palindrome is also referred to in the literature as reversed repeat, palindrome, complemented palindrome, complemented inverted repeat, or reverse complement repeat [2, 7]. A repeat is said to be complementary palindrome if nucleotides in a subsequence are the reverse ordering of nucleotides in another subsequence with each nucleotide replaced by

its complement. For instance, the subsequences “AAACGT” and “ACGTTT” are complementary palindrome since (A,T) and (C,G) are complement pairs.

In Figure 3, the 12 bases sequence “ACGCTTAAGCGT” is a part of a DNA sequence. If we consider a subsequence “AAGCGT” formed from the last six bases, the complement of “AAGCGT” is “TTCGCA”. Its reverse ordering is “ACGCTT”, which is the same as the first six bases of the original 12 bases sequence. Thus, the repeat is called a complementary palindrome.

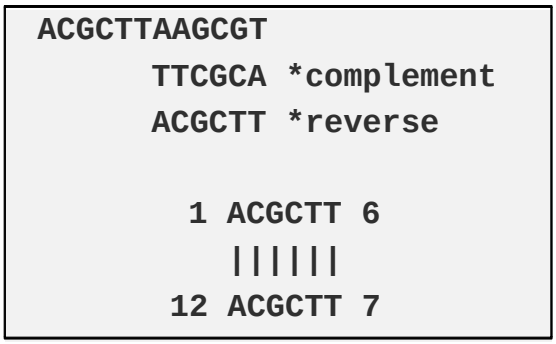


Figure 3. An example of a complementary palindrome. The sequence is a part of a DNA sequence.

2.1.3 Three-base periodicity

Functionally, a DNA sequence consists of two types of regions, namely protein-coding and non-protein-coding regions [26]. In organisms with a distinct cellular nucleus, called Eukaryotes, the coding regions are usually divided into several disconnected fragments known as exons. The non-coding regions in-

between the exons are known as introns. Before producing proteins, introns are removed so that exons are joined together to form an "uninterrupted" gene. Codons combined by three consecutive bases represent a protein unit, i.e., an amino acid. As listed in Table 2, sixty-four possible codons can be formed by four bases but there are only twenty amino acids. For example, GCT, GCC, GCA and GCG represent the amino acid "Alanine" as in the first row of Table 2. One of the characteristics of the protein-coding region is its three-base periodicity [41]. This periodicity might be due to the codon structure. The three-based periodicity implies that the power spectrum of the subsequence in a protein-coding region would have a strong component at the period-3 frequency, i.e., $2\pi/3$.

Tab1e 2. Amino acids formed by triplets of bases (codons).

Amino acids	Codon
Ala	GCT, GCC, GCA, GCG
Arg	CGT, CGC, CGA, CGG, AGA, AGG
Asn	AAT, AAC
Asp	GAT, GAC
Cys	TGT, TGC
Gln	CAA, CAG
Glu	GAA, GAG
Gly	GGT, GGC, GGA, GGG
His	CAT, CAC
Ile	ATT, ATC, ATA
Leu	TTA, TTG, CTT, CTC, CTA, CTG
Lys	AAA, AAG
Met	ATG
Phe	TTT, TTC
Pro	CCT, CCC, CCA, CCG
Ser	TCT, TCC, TCA, TCG, AGT, AGC
Thr	ACT, ACC, ACA, ACG
Trp	TGG
Tyr	TAT, TAC
Val	GTT, GTC, GTA, GTG
START	ATG
STOP	TAG, TGA, TAA

2.2 Compression Methods

There are two kinds of compression methods, namely lossless compression and lossy compression. Retrieving from compressed data without loss is defined as lossless, while that with data loss is defined as lossy. Since it is not possible to sacrifice any of the data in a DNA sequence, only lossless compression is considered for DNA sequence compression. As there are only four bases in a DNA sequence, two bits are required to code each nucleotide without any compression. An example is shown in Figure 4. The four bases {A, C, G, T} are represented by {00, 01, 10, 11}. Thus, two bits per base is the minimum requirement for compressing a DNA sequence.

Figure 4. Two bits per base. [18]

Some researchers have examined the use of a general-purpose compression algorithm for compressing DNA sequences. The two main compression approaches are statistical and substitutional [8] methods. In the statistical approach, blocks of fixed length subsequences are encoded with respect to their occurrence probabilities. For example, in Huffman coding [19], short codewords are used for encoding frequent patterns while long codewords

are used for encoding non-frequent patterns. In the substitutional approach such as LZ77 [20], pointers are used to locate previous occurrences. So, the lengths of subsequences to be encoded are not fixed, in contrast to the statistical approach.

General-purpose compression algorithms often do not perform well for DNA sequences. Some of these algorithms, such as gzip, use more than 2 bits per base [1]. The reason behind is that these algorithms do not consider the special characteristics in a DNA sequence, such as ignoring the long term repetitions in a DNA sequence. Some of them, such as context tree weighting (CTW) [21], are very slow and use a great deal of memory in finding characteristic patterns in the long DNA sequences. To sum up, lossless compression algorithms for DNA sequences need to be developed. These algorithms should exploit DNA sequence characteristics, such as the long-term repetitions, effectively and efficiently. Consequently, DNA-oriented compression methods such as Biocompress-2, GenCompress, context tree weighting (CTW)+LZ and DNACompress, have been developed [3-8].

2.2.1 Compression Methods for DNA Sequences

Biocompress [22] was the first algorithm designed by Grumbach *et al.* in 1993 specifically for compressing DNA sequences. Biocompress-2 [8] is its

second version. These two algorithms are based on a sliding window algorithm, known as LZ77, proposed by Ziv and Lempel [23]. A subsequence is encoded by reference to an identical subsequence occurring in the past, i.e., only the position of the previously occurred similar subsequence and the repetition length are encoded. Biocompress detects exact matches and complementary palindromes, while Biocompress-2 introduces an additional order-2 arithmetic coding (Arith-2). Biocompress-2 uses Arith-2 if no significant repetition is found. For both Biocompress and Biocompress-2, the compression ratio is higher when the length of similar subsequences is longer.

Cfact proposed by Rivals *et al*. [9-10] is another compression technique based solely on exact matches. A two-pass strategy is used. In the first pass, the whole sequence is parsed by using a suffix tree. A list of repetitive subsequences sorted according to their lengths is produced. In the second pass, the subsequences are encoded with reference to previously occurred similar subsequences. Two bits per base are then used to encode the remaining non-repetitive regions.

GenCompress-2 [4-6] proposed by Chen *et al*. achieves a significantly better compression ratio than the previously presented algorithms. Contrary to

Biocompress and Cfact, GenCompress utilizes approximate matches instead of exact matches. GenCompress-1 considers only substitutions for the repeats, while GenCompress-2 considers deletion, insertion and substitution for finding the repeats. As with Biocompress, GenCompress considers whether it is worthwhile to encode a subsequence. If not, Arith-2 is used for encoding.

DNACompress [3] employs the Ziv-Lempel compression scheme as Biocompress-2 and GenCompress. It consists of two phases. The first phase is to find all approximate repeats, including complementary palindromes. The second phase is to encode approximate repeat regions by referring to the previous regions in the sequence and non-repeat regions by Arith-2. To identify all the similar subsequences, a software tool called PatternHunter [27] is used, which is a fast and sensitive homology search engine. Besides providing additional compression gains, DNACompress is considerably faster than GenCompress.

CTW+LZ proposed by Matsumoto *et al.* [7] is a technique based on the context tree weighting (CTW) method and LZ-based compression. Basically, long exact or approximate repeating subsequences including complementary palindromes are encoded by a LZ-based algorithm, whereas short subsequences

are compressed using CTW. Though CTW+LZ obtains good compression ratios, its execution time is too long, especially for long sequences.

DNAC [11] is a DNA compression scheme consisting of four phases. In the first phase, a suffix tree is built to locate exact matches. In the second phase, all the exact repeats are extended to approximate repeats by dynamic programming. In the third phase, the non-overlapping repeats with the highest scores are extracted from the overlapping regions. In the last phase, all the repeats are encoded.

GeNML, proposed by Tabus *et al.* [24], is based on normalized maximum likelihood discrete regression or approximate block matching. The compression performance and speed are both improved in comparison with Biocompress-2, GenCompress-2, CTW+LZ and DNACompress [25]. A DNA sequence is divided into fixed-size blocks, and GeNML encodes the fixed-size blocks by reference to a previously encoded subsequence with minimum substitution operations.

DNAPack, proposed by Behzadi *et al.* [2], considers only substitutions for the repeats and complementary palindromes, and uses either CTW or Arith-2

for encoding non-repeating regions. In identifying repeats, it uses dynamic programming approaches instead of greedy techniques. DNAPack provides a better compression gain, on average, when compared with DNACompress for a number of short DNA sequences [26].

All of the above compression algorithms exploit the long-term repetitions in a DNA sequence. Recently, Pinho *et al.* proposed a Three-state model for compressing the DNA protein-coding regions [26, 42]. As the protein-coding regions contain three-base periodicity, three finite-context models are used to characterize the periodicity statistically. It is reported that the compression ratio for the protein coding regions is better than DNACompress for some DNA sequences. However, the proposed algorithm is not a complete compression algorithm as the three-state model can be used in the protein-coding regions only, not for the whole DNA sequence.

2.2.2 Experimental Results

Most of the DNA-oriented compression methods have been tested on a set of standard benchmark DNA sequences downloadable from [43]. These sequences together with their lengths [5] are listed in Table 3.

Tab1e 3. Information on the standard benchmark DNA sequences. Note KB stands for kilo bytes.

Sequence name	Length	Sources	File size (KB)
CHMPXX (or MPOCPCG)	121024	chloroplasts	29.55
CHNTXX	155844		38.05
HEHCMVCG (or HS5HCMVCG)	229354	complete genome from viruses	55.99
HUMDYSTROP	38770	sequences from humans	9.47
HUMGHCSA	66495		16.23
HUMHBB	73323		17.90
HUMHDABCD	58864		14.37
HUMHPRTB	56737		13.85
MPOMTCG	186608	complete genomes of mitochondria	45.56
MTPACGA (or PANMTPACGA)	100314		24.49
VACCG	191737	complete genome from viruses	46.81

The eleven sequences shown in Table 3 come from various sources such as chloroplasts, mitochondria, human and virus. The length refers to the number of bases in the sequence. The file size in KBytes is obtained by using 2 bits for each base. This is then the file size required without any compression.

The experimental results of compressing these eleven sequences by various compression algorithms are summarized in Table 4 [2-11]. The compression ratio is calculated by the bits per base (bpb) used. Without compression, 2 bpb is required. Thus, a good compression ratio implies that we have a bpb considerably smaller than 2. Biocompress-2, GenCompress and DNACompress are represented by ‘BioComp-2’, ‘GenComp-2’ and ‘DNAComp’

respectively. We can see that the average bpb ranges from 1.70 to 1.78 for most of the DNA-oriented compression methods.

Table 5 shows the compression gain of the various compression methods. The compression gain is defined as 1-(|O|/2|I|)x100%, where |O| is the number of bits required for storing the compressed sequence, and |I| is the number of bases of a particular DNA sequence. The compression gain ranges from 11% to 15%. This means that most algorithms can reduce the DNA sequence size by less than 20%. Note that there is no compression result for GeNML for the sequence HUMHBB. According to other compression methods, the result for compressing HUMHBB is always above the average value, therefore, the average bpb and the average compression gain without considering HUMHBB are shown in the last row of Table 4 and Table 5 respectively. From the average values, we can see that GeNML preformed slightly better than other compression schemes for these eleven sequences.

As for the compression of the protein-coding regions in the sixteen chromosome sequences of the yeast named *S. cerevisiae*, Table 6 shows the bpb used by the three-state model and the DNACompress [26]. The bolded value indicates the better compression ratio for a particular sequence. Within the

sixteen chromosome sequences of *S. cerevisiae*, DNACompress performs better in six chromosome sequences while the three-state model performs better in ten chromosome sequences (i.e., Chr II, Chr III, Chr VI, Chr VII, Chr VIII, Chr IX, Chr X, Chr XI, Chr XIV and Chr XV). The average bpb of the three-state model is slightly lower than that of DNACompress, as shown in the last row of Table 6. Although the three-state model performs comparably to DNACompress, the algorithm is applicable to protein-coding region only. Thus, the sequence is first required to be divided into two regions: protein-coding region and non-protein-coding region. Then, the three-state model is applied to the protein-coding region only while other compression algorithm is required for the non-protein-coding region. The three-state model is not yet a complete compression scheme as DNACompress.

Tab1e 4. The bits per base (bpb) used by various DNA-oriented compression methods

Sequence name	BioComp-2	GenComp-2	DNAComp	CTW+LZ	DNAC	GeNML	DNAPack	DNAMem
CHMPXX	1.6848	1.6730	1.6716	1.6690	1.6716	1.6617	1.6602	1.6601
CHNTXX	1.6172	1.6146	1.6127	1.6129	1.6127	1.6101	1.6103	1.6101
HEHCMVCG	1.8480	1.8470	1.8492	1.8414	1.8492	1.842	1.8346	1.8349
HUMDYSTROP	1.9262	1.9231	1.9116	1.9175	1.9116	1.9085	1.9088	1.9084
HUMGHCSA	1.3074	1.0969	1.0272	1.0972	1.0272	1.0089	1.0390	1.0311
HUMHBB	1.8800	1.8204	1.7897	1.8082	1.7897	-	1.7771	1.7765
HUMHDABCD	1.8770	1.8192	1.7951	1.8218	1.7951	1.7059	1.7394	1.7395
HUMHPRTB	1.9066	1.8466	1.8165	1.8433	1.8165	1.7639	1.7886	1.7884
MPOMTCG	1.9378	1.9058	1.8920	1.9000	1.8920	1.8822	1.8932	1.8925
PANMTPACGA	1.8752	1.8624	1.8556	1.8555	1.8556	1.8440	1.8535	1.8533
VACCG	1.7614	1.7614	1.7580	1.7616	1.7580	1.7644	1.7583	1.7582
average bpb	**1.7838**	**1.7428**	**1.7254**	**1.7389**	**1.7254**	**1.6992**	**1.7148**	**1.7139**
average bpb without HUMHBB	**1.7742**	**1.7350**	**1.7190**	**1.7320**	**1.7190**	**1.6992**	**1.7086**	**1.7077**

Tab1e 5. The compression gains of various DNA-oriented compression methods.

Sequence name	BioComp-2	GenComp-2	DNAComp	CTW+LZ	DNAC	GeNML	DNAPack	DNAMem
CHMPXX	15.76%	16.35%	16.42%	16.55%	16.42%	16.92%	16.99%	17.00%
CHNTXX	19.14%	19.27%	19.37%	19.36%	19.37%	19.50%	19.49%	19.50%
HEHCMVCG	7.60%	7.65%	7.54%	7.93%	7.54%	7.90%	8.27%	8.26%
HUMDYSTROP	3.69%	3.85%	4.42%	4.13%	4.42%	4.58%	4.56%	4.58%
HUMGHCSA	34.63%	45.16%	48.64%	45.14%	48.64%	49.56%	48.05%	48.45%
HUMHBB	6.00%	8.98%	10.52%	9.59%	10.52%	-	11.15%	11.18%
HUMHDABCD	6.15%	9.04%	10.25%	8.91%	10.25%	14.71%	13.03%	13.03%
HUMHPRTB	4.67%	7.67%	9.18%	7.84%	9.18%	11.81%	10.57%	10.58%
MPOMTCG	3.11%	4.71%	5.40%	5.00%	5.40%	5.89%	5.34%	5.38%
PANMTPACGA	6.24%	6.88%	7.22%	7.23%	7.22%	7.80%	7.33%	7.34%
VACCG	11.93%	11.93%	12.10%	11.92%	12.10%	11.78%	12.09%	12.09%
average	**10.81%**	**12.86%**	**13.73%**	**13.05%**	**13.73%**	**15.04%**	**14.26%**	**14.30%**
average without HUMHBB	**11.29%**	**13.25%**	**14.05%**	**13.40%**	**14.05%**	**15.04%**	**14.57%**	**14.62%**

Tab1e 6. The bits per base (bpb) used for Three-state model and DNACompress in *S.cerevisiae.*

Chr	Reference	No. of bases	Three-state model	DNA Compress
I	GI:50593113	143157	1.911	**1.884**
II	GI:50593115	605184	**1.897**	1.912
III	GI:42759850	217332	**1.911**	1.918
IV	GI:50593138	1129605	1.890	**1.846**
V	GI:7276232	391086	1.901	**1.883**
VI	GI:42742172	183702	**1.904**	1.932
VII	GI:50593213	784707	**1.893**	1.897
VIII	GI:50882583	402792	**1.903**	1.907
IX	GI:6322016	310041	**1.903**	1.933
X	GI:42742252	557103	**1.899**	1.907
XI	GI:50593424	478620	**1.895**	1.938
XII	GI:42742286	784695	1.898	**1.863**
XIII	GI:44829554	693291	1.894	**1.886**
XIV	GI:50593505	576585	**1.900**	1.930
XV	GI:42742309	785568	**1.897**	1.901
XVI	GI:50593503	687666	1.896	**1.889**
	average	545696	**1.8995**	1.9016

2.3 Repetitions in DNA sequences

Basically, all DNA-oriented compression methods make use of the ideas of finding repeats and complementary palindromes in DNA sequence. In fact, most of the time needed to run these compression programs are in searching of the repeats. Thus, searching for repetitions in DNA sequences effectively and efficiently becomes a very important problem. It is often not a trivial task to search for all approximate repeats in a very long DNA sequence. Algorithms such as CTW+LZ algorithm [7] take a long time to find approximate repeats that are optimal for compression [3]. Recently, several homology search engines have been developed for searching approximate repeats and complementary

palindromes. Examples include PatternHunter [27] and Blastn [28]. DNACompress is a well-known example of utilizing PatternHunter for searching repetitions.

2.3.1 Blastn

The Basic Local Alignment Search Tool (BLAST) [29] is a utility that is maintained by the National Center for Biotechnology Information (NCBI). BLAST is a set of similarity search programs designed to explore all the available sequence databases, regardless of whether the query is a protein or DNA sequence [28]. The BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. The scores assigned in a BLAST search have a well-defined statistical interpretation, in order to make real matches easier to distinguish from random background hits. BLAST uses a heuristic algorithm that seeks local as opposed to global alignments, and is therefore able to detect relationships among sequences that share only isolated regions of similarity.

Blastn is a type of BLAST search for a DNA sequence in which similar nucleotide sequences are searched throughout the contents of a nucleotide sequence database. It is freely available on the Web [30].

2.3.2 PatternHunter

Pattern Hunter is a homology search engine like Blastn, but with modifications aimed at improving sensitivity, alignments, memory use and speed [3]. It is more sensitive; is two orders of magnitude faster than Blastn when processing long sequences; and requires only a fraction of the memory. By using a patented spaced seed technology and algorithm for handling hit generation, hit extension and gap extension, all approximate repeats, including complementary palindromes, are produced and arranged in the order of a score [31]. The score indicates the similarities in the repeats. Thus, a large score implies highly similar repeats.

3 Similarity Study

How to minimize the value of bits per base (bpb) in DNA sequence compression is always a question for researchers. As we can see from the experimental results in Table 4, the average bpb of the first compression algorithm is 1.78 and that of the recent algorithm is 1.71. Thus, over the twelve years of research in DNA compression, the improvement of the average bpb is only about 0.06. How to further reduce the value of bpb is the main focus of this thesis.

As discussed in Chapter 2, state-of-the-art DNA compression methods are always based on searching repetitions within the DNA sequence. Compression is achieved only if there are similar subsequences along the current DNA sequence. In fact, similarities in DNA sequences could exist among different species that are close in terms of evolutionary distance [33-34]. Similarities could also exist among different chromosome sequences of one species [35]. These similarities imply that similar subsequences can be found among different DNA sequences which can thus be used beneficially for compression.

To verify the cross-sequence similarities, we studied similarities in DNA sequences among different chromosome sequences of *Saccharomyces cerevisiae*

(*S. cerevisiae*). The 16 chromosome sequences can be downloaded from ftp://ftp.ncbi.nlm.nih.gov/genomes/. Firstly, the self-sequence similarity and the cross-sequence similarities of these 16 chromosome sequences are studied. Secondly, the location and length of similar subsequences will be discussed. Finally, the result of cross-chromosomal prediction will be analyzed. To facilitate the discussion, similar subsequences located within the chromosome sequence are called self-(chromosomal) similarity/ self-referencing while those located in another chromosome sequence are called cross-(chromosomal) similarity/ cross-referencing.

3.1 Existence of Similar Subsequences Among Chromosomes

In this section, the search engine PatternHunter (Section 2.3.2) is employed to search for all approximate repeats and approximate complementary palindrome repeats in one DNA sequence or between a pair of DNA sequences [27]. The approximate repeats are repeats that contains errors, i.e., with certain unmatched nucleotides between two subsequences (Section 2.1.1). The complementary palindrome repeats mean nucleotides in a sequence is the reverse ordering of nucleotides in another sequence with each nucleotide replaced by its complement.

All approximate repeats obtained from PatternHunter are ranked by a score. Besides, details of the repeats, including the location of the repetitive regions and the length of the repetitive regions, are output to an "aln" file.

3.1.1 About *S. cerevisiae*

In *S. cerevisiae*, the 16 chromosome sequences are denoted as Chr I to Chr XVI. The longest chromosome sequence is Chr IV which has around 1540k bases while the shortest chromosome sequence is Chr I which has around 230k bases. Figure 5 shows the number of bases in each chromosome sequence of *S. cerevisiae*.

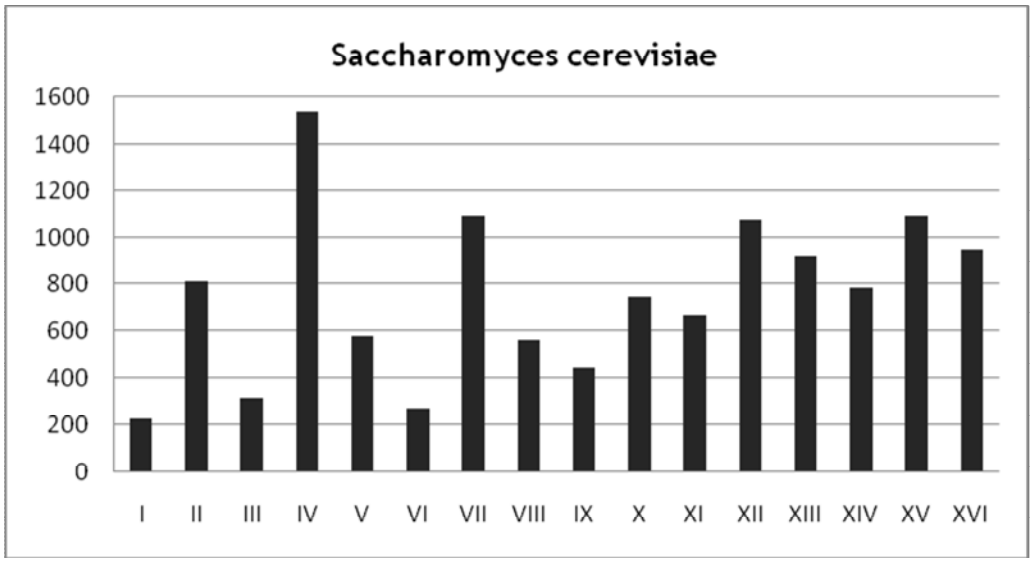


Figure 5. The lengths of the sixteen chromosome sequences in *S. cerevisiae*. The y-axis is the number of bases in 1000 units.

3.1.2 Self-referencing

Self-referencing is defined as finding repetitions within the current DNA sequence. All state-of-the-art DNA compression algorithms consider self-referencing only. PatternHunter is used to search for repetitions within one chromosome sequence. The following shows part of the outputs from the PatternHunter in finding the self-similarity within Chr I.

```
Identities = 13159/14613 (90%)
Identities = 2434/2588 (94%)
Identities = 2071/2298 (90%)
Identities = 1610/1759 (91%)
Identities = 1573/1759 (89%)
```

The self-similarities are sorted according to the score which is obtained from the repetitive lengths. In the first record, the total length and the number of identical nucleotides of the repetitive regions are ‘14613’ and ‘13159’, respectively. The length of the repetitive regions is of special interest in our study. It is because the repetitive regions can be encoded with respect to similar regions that have been encoded in the past. Thus the longer the matching sequences are, the higher the compression ratios attained. The above shows that the longest repetitive region found within Chr I contains about 13000 bases. Figure 6a shows the lengths of the top four score repetitive regions found inside Chr I, Chr III, Chr IV, Chr V, Chr VII, Chr VIII, Chr XI, Chr XII, Chr XIII, Chr XIV, Chr XV and Chr XVI. Most repetitive regions have a length of around

6000 bases. Chr I is one special case as the length of the longest repetitive region is around 13000 but that of the second longest drops to around 2000. Besides, the lengths of the top four longest repetitive regions of Chr III and Chr XI are very short, they are around 1000 only.

Figure 6b shows the length of self-referencing repetitive regions and non-repetitive regions in *S. cerevisiae*. The grey parts in the bars are the length of self-similar subsequences while the black parts are the length of non-repetitive regions. We can see that the lengths of grey parts for all the chromosome sequences are very short. The self-similar subsequence parts in Chr III and Chr XI can even be unseen since the portions are too small when comparing with the lengths of non-repetitive regions. This means that the self-referencing repetitive regions are often not very significant.

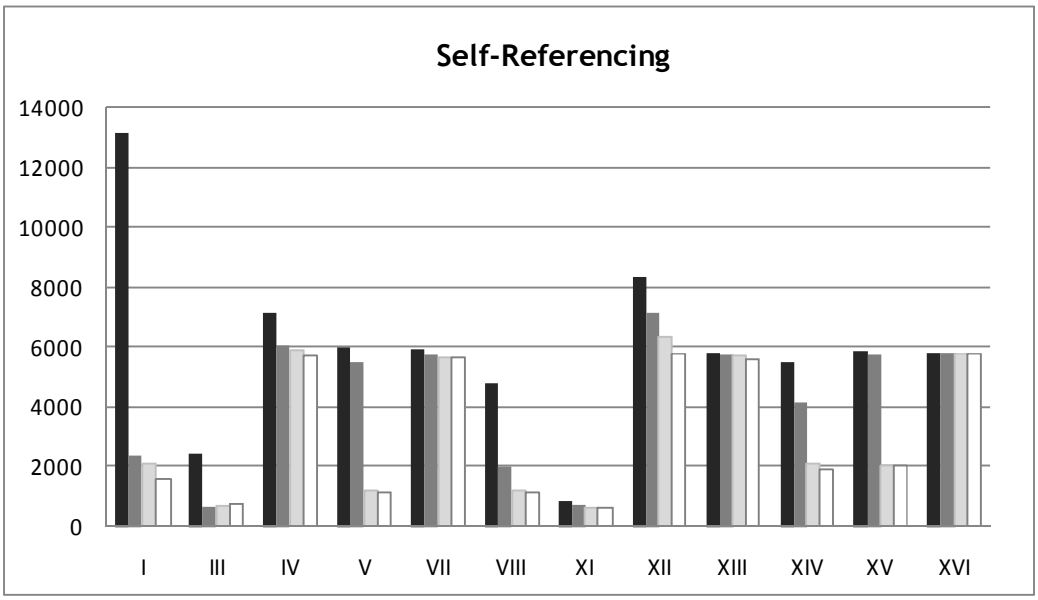


Figure 6a. The lengths of the top four longest repetitive regions found within the current chromosome sequence in *S. cerevisiae*. The first, second, third and fourth scores are illustrated by black, grey, light grey and white color bars respectively. The y-axis denotes the lengths of the repetitive regions.

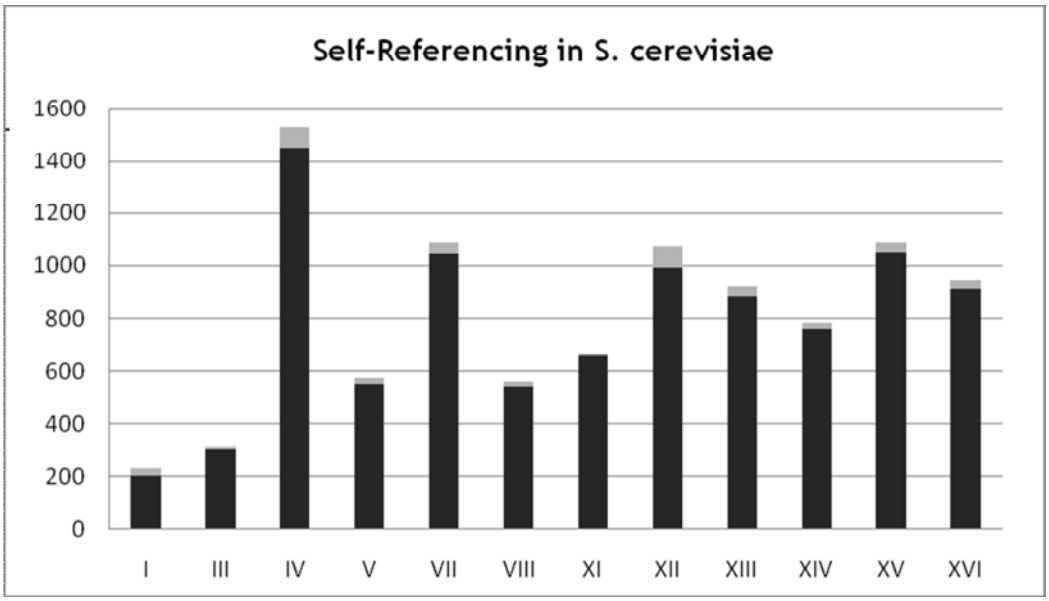


Figure 6b. A study of the lengths of the repetitive regions and non-repetitive regions in *S. cerevisiae*. The light grey parts indicate the length of self-referencing repetitive regions. The black parts indicate the length of non-repetitive regions. The y-axis is the number of bases in 1000 units.

3.1.3 Cross-referencing

Cross-referencing is defined as finding repetitions from different chromosome sequences. It attempts to find similarities among different chromosome sequences. To show the similarities between different chromosome sequences in *S. cerevisiae,* cross-referencing between Chr I and Chr VIII are explored. Using PatternHunter, the top five score repetitive regions between Chr I and Chr VIII are obtained as follows,

```
Identities = 17034/17466 (97%)
Identities = 12502/13765 (90%)
Identities = 6407/6790 (94%)
Identities = 5677/6041 (93%)
Identities = 1518/1904 (79%)
```

We can see that the two longest similar regions found between Chr I and Chr VIII are about 17000 and 12500 bases long. In fact, if we compare the top four results, the lengths of similar regions between Chr I and Chr VIII are greater than that of similar subsequences found within Chr I. To have a clear picture, Figure 7 is provided which depicts the lengths of the top three score repetitive regions between a particular chromosome sequence and the other fifteen chromosome sequences of *S. cerevisiae*.

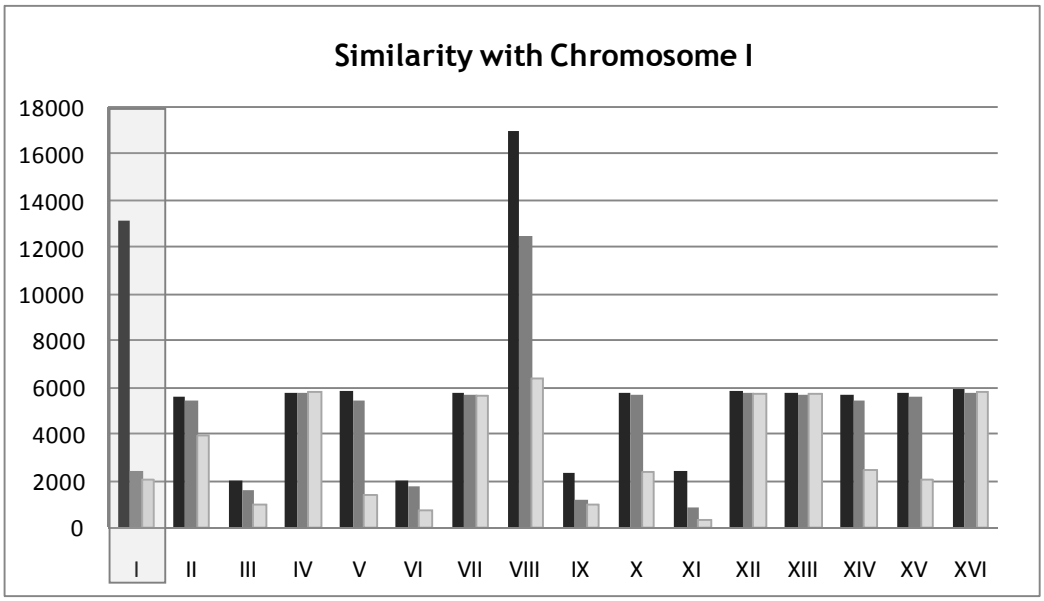
Figure 7(a) summarizes the lengths of the top three score repetitive

regions between Chr I and the other fifteen chromosome sequences of *S. cerevisiae*. We can see that the lengths of the repetitive regions found between Chr I and Chr VIII are always larger than those found within Chr I alone. In addition, the lengths of the repetitive regions found between Chr I and other chromosome sequences such as Chr II, Chr IV, Chr VII, Chr XII, Chr XIII and Chr XVI are significant too.

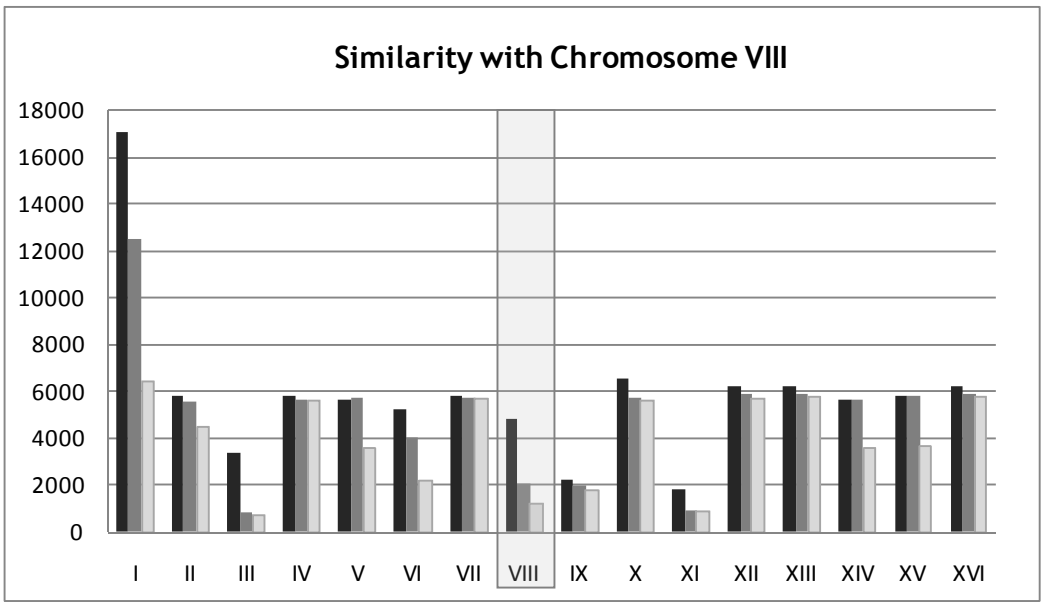
Figure 7(b) shows the lengths of the top three score repetitive regions between Chr VIII and the other fifteen chromosome sequences of *S. cerevisiae*. Compared with Figure 7(a), the lengths of the repetitive regions found between Chr I and Chr VIII are always larger than those found within Chr VIII alone. At the same time, the lengths of the repetitive regions found between Chr I and other chromosome sequences (with an exception of III, IX and XI) are noteworthy too. The interesting point is that the lengths of the repetitive regions found between Chr I and Chr VIII are always larger than those found within Chr VIII alone or those found within Chr I alone. Besides, the lengths of the repetitive regions found between Chr I and other chromosome sequences (except VIII), shown in Figure 7(a), and that between Chr VIII and other chromosome sequences (except I), shown in Figure 7(b), are nearly the same.

Figure 7(c) illustrates the lengths of the top three score repetitive regions between Chr III and the other fifteen chromosome sequences of *S. cerevisiae*. In this case, we can see that the self-similarity inside Chr III is small, as compared with the cross-similarities between regions in other chromosome sequences.

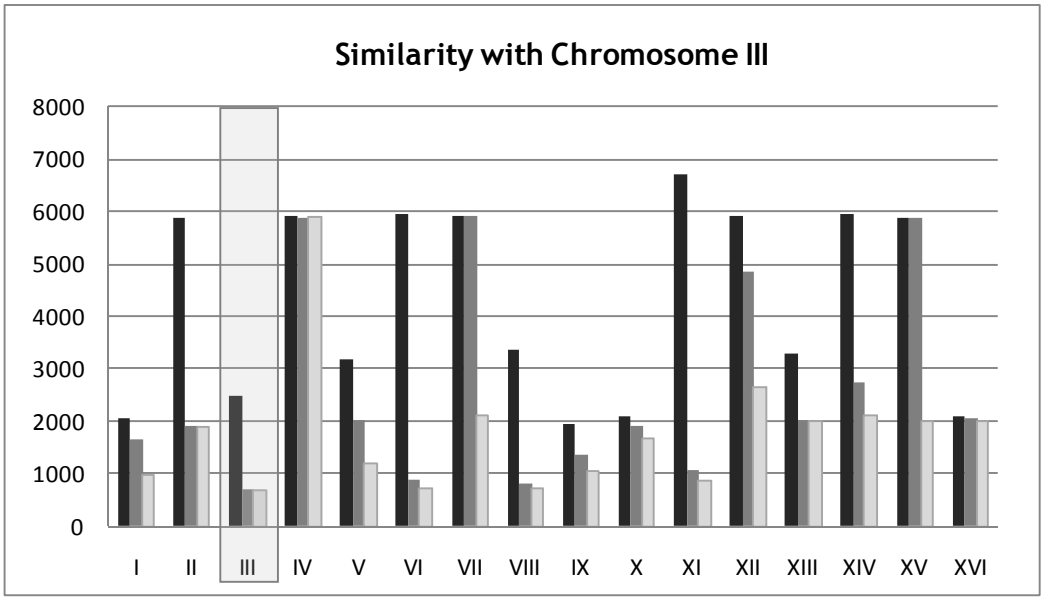
The same is true for Chr XI as shown in Figure 7(d). In fact, similar observation is obtained from other chromosome sequences of *S. cerevisiae*. This shows that besides self-similarity within the chromosome sequence itself, cross-similarities with other chromosome sequences are often significant and should not be ignored. These cross-similarities can be exploited which should be beneficial for compression applications.


Similarity with Chromosome I
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
I
II
III
IV
V
VI
VII
VIII
IX
X
XI
XII
XIII
XIV
XV
XVI

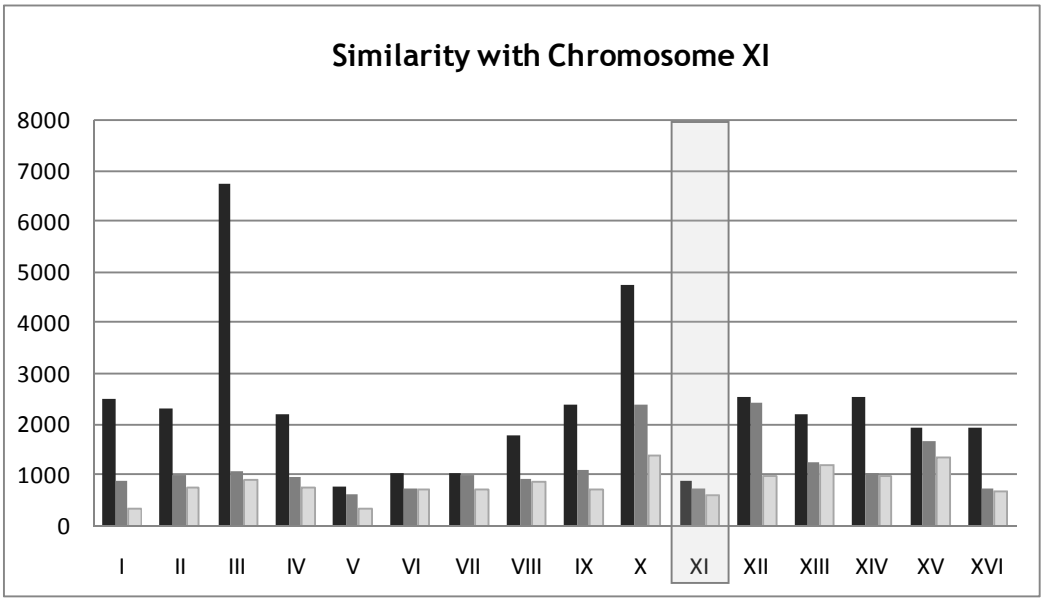
(a)


Similarity with Chromosome VIII
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
I
II
III
IV
V
VI
VII
VIII
IX
X
XI
XII
XIII
XIV
XV
XVI

(b)



(c)



(d)

Figure 7. The lengths of the top three score repetitive regions between (a) Chr I, (b) Chr VIII, (c) Chr III and (d) Chr XI and the other fifteen chromosome sequences of S. cerevisiae. The first, second and third scores are illustrated by black, grey and light grey color bars respectively. The highlighted boxes indicate self-chromosomal similarity. Y-axis denotes the length of the repetitive regions.

3.2 Analysis with Similar Sequences between Chromosomes

In the previous section, we have shown that similarities among different chromosome sequences are often significant as compared with self-similarity. To quantify the potential gain in cross-sequence compression, we need to find out whether any subsequence in the current sequence can be predicted from regions in another sequence. If so, there will be gain if these two sequences are compressed together by reference to each other. We termed this as cross-sequence compression. The length of these cross-reference subsequences determines the potential compression gains that would result by considering multiple DNA sequences in compression. The longer the length is, the higher the potential compression gain will be. In this section, we studied the length and the location of these similar subsequences.

3.2.1 Length of Similar Sub-sequences

Table 7 shows the total lengths of subsequences that can be predicted either from the current chromosome sequence or from other chromosome sequences. Each column entry in the table represents the total lengths of subsequences in Chr *a* that can be predicted from certain regions in Chr *b*. So the first entry for Chr I, "24807", represents the total length of similar subsequences that can be found within Chr I. In other words, a total of 24807

nucleotides can be predicted by reference to itself. Similarly, the second entry "15253" represents the total lengths of similar subsequences in Chr I that can be predicted from Chr II. In other words, a total of 15253 nucleotides in Chr I can be encoded with reference to similar subsequences in Chr II. Furthermore, the first entry for Chr III "12411" is highlighted since that is greater than "11361" which is the total length of similar subsequences that can be found within itself.

Table 7. Total lengths of subsequences in Chr *a* that can be predicted from certain regions in Chr *b*. The bolded value represents self-similarity (i.e., self-prediction) while the highlighted boxes represent those entries that have greater values than the self-predicted one.

b \ *a*	**I**	**III**	**IV**	**V**	**VII**	**VIII**	**XI**	**XII**	**XIII**	**XIV**	**XV**	**XVI**
Length of Chr *a*	230208	316617	1531918	576869	1090946	562643	666454	1078175	924429	784333	1091289	948062
Class of Chr *a*	3	1	3	1	2	1	1	3	2	1	2	2
I	**24807**	12411	31766	13354	23469	36809	8459	22818	18084	19422	33736	15894
II	15253	17228	58017	35443	56365	22205	9926	36714	29897	40236	43754	40400
III	9964	**11361**	29904	12292	26207	13925	15414	26790	11836	32780	22574	13006
IV	16241	22604	**82152**	47444	55529	35110	12097	87680	41181	46787	70059	43633
V	10988	11933	56508	**25003**	37456	20144	7095	42686	32899	37707	29723	26308
VI	9634	12218	33910	16000	34056	23460	6975	30481	19089	30273	26885	22531
VII	16149	14952	67605	39911	**43212**	26373	11571	79301	45231	35342	41149	67663
VIII	50536	14030	48346	27718	29262	**20263**	19659	32142	24432	35704	25680	30953
IX	7623	9438	19237	21098	27053	16160	12521	17685	16718	34194	32314	14307
X	14274	20753	61192	37469	37470	28774	35014	41511	37283	41269	38576	34794
XI	7467	17228	13789	8715	15015	19735	**7169**	12743	19559	11450	17025	8671
XII	7623	17316	77913	37045	62116	29828	9765	**84170**	51846	41057	48221	40916
XIII	13193	14127	46460	29155	44821	31372	21718	46768	**37573**	35588	46740	40699
XIV	13049	28820	53655	39883	39941	27743	13460	55506	31969	**22881**	49117	24580
XV	25981	16711	73035	35748	46149	24951	13033	64470	51032	51085	**37964**	55019
XVI	10455	14598	55231	34973	66621	33132	9145	58181	43181	26549	58936	**34648**

The self-referencing values are bolded in Table 7. All entries that have a greater number of nucleotides predicted from other chromosome sequences than the self-referencing value are highlighted. Results can be grouped into three classes. The first class, consisting of Chr III, Chr XI, Chr XIV, Chr VIII and Chr V, has high similarities with chromosome sequences other than itself. We can see that more than half of chromosome sequences have larger cross-referencing values than the self-referencing value. This implies that a potentially high compression gain can be obtained if these chromosome sequences employ a cross-referencing strategy with subsequences obtained from other chromosome sequences in addition to self-referencing.

The second class consists of Chr XV, Chr XVI, Chr VII and Chr XIII. The numbers of highlighted entries for Chr XV, Chr XVI, Chr VII and Chr XIII are 8, 7, 6 and 5 respectively. Although its numbers are not as high as that in the first class, a potential compression gain is also expected since the cross-referencing values are still big. As self-referencing is still considered in compression, an effective cross-referencing strategy should improve the overall compression ratio.

The last class consists of Chr I, Chr XII and Chr IV. The numbers of

highlighted entries for Chr I and Chr XII are 2 and 1 respectively. There is no highlighted entry for Chr IV. For Chr I, a total of 50536 nucleotides can be predicted from Chr VIII. In contrast, only 24807 nucleotides can be self-referenced within Chr I itself. The number is almost doubled if a reference is made to Chr VIII. This is consistent with the findings in Figure 7(a). For Chr XII, a total of 87680 nucleotides can be predicted from Chr IV. This is comparable to the self-referencing value which is 84170. As the length of Chr XII is 1078175, these self- and cross-referencing numbers are indeed significant. For Chr IV, the self-similarity consists of 82152 nucleotides. In contrast, the largest cross-similarity is 77913 with Chr XII. While this is smaller than the self-referencing value, the combination of self-referencing and cross-referencing values should contribute to a better compression.

3.2.1 Location of Similar Subsequences

Besides considering the total length of subsequences that can be referenced from other chromosome sequences, the distribution of these repetitive regions within a chromosome sequence is also important. Let the subsequence in a sequence S that is similar to a subsequence in sequence i be S(i) and the subsequence in S that is similar to a subsequence in sequence j be S(j). The total length of subsequences within S that can be referenced from i and j is given by

$T=|S(i)|+|S(j)|-|S(i) \cap S(j)|$. Obviously if these subsequences are well spread out such that $|S(i) \cap S(j)|$ is zero, i.e., they do not overlap in position, T is maximized. This implies that a high proportion of the nucleotides within S can be predicted by cross-referencing among different chromosome sequences, which can result in a high compression gain.

Figure 8 provides a detailed analysis on the locations of similar subsequences. The similar subsequences are well spread out. This shows the potential benefits of encoding multiple DNA sequences together. In order to present the locations of similar subsequences clearly, we only considered those repeats with scores above 100. Also, the illustration only shows the repeat lengths which are above 20. Figure 8(a), 8(b) and 8(c) demonstrate the locations of similar subsequences for the first, the second and the third class respectively. For self-similarity, only the repetitive regions are marked. Note that the symbol * next to the chromosome sequences represent those sequences without significant self-similar subsequence.

In Figure 8(a), we can see that the portions of self-referencing regions in all the five chromosome sequences are very small, as compared with that of cross-referencing regions with other chromosome sequences. In the case of Chr

XI, Chr XIV, Chr VIII and Chr V, we cannot even see the self-referencing subsequences in the figure. Besides, similar subsequences predicted from other chromosome sequences contribute to different locations. For example, the four similar subsequences found from Chr X, Chr XIII, Chr VIII and Chr II contribute to four different areas in Chr XI. Similar observations can be seen from Figure 8(b) about the second class.

Figure 8(c) shows locations of similar subsequences for the third class. For Chr I, we can see that the portions of cross-referencing regions with either Chr VIII or Chr XV are much larger than that of self-referencing regions. For Chr XII, the portions of cross-referencing regions with Chr XIII or Chr IV are comparable to that of self-referencing regions. For Chr IV, the portions of cross-referencing regions with Chr XII are comparable to that of self-referencing regions too.

Figure 8 shows that the cross-referencing regions with other chromosome sequences are often significant when compared with self-referencing regions within itself. Also, similar subsequences from different chromosome sequences contribute to different locations in the current sequence. As a result, our study shows that it would be advantageous to compress different chromosome

sequences together to be beneficial from both self-chromosomal and cross-chromosomal similarities.

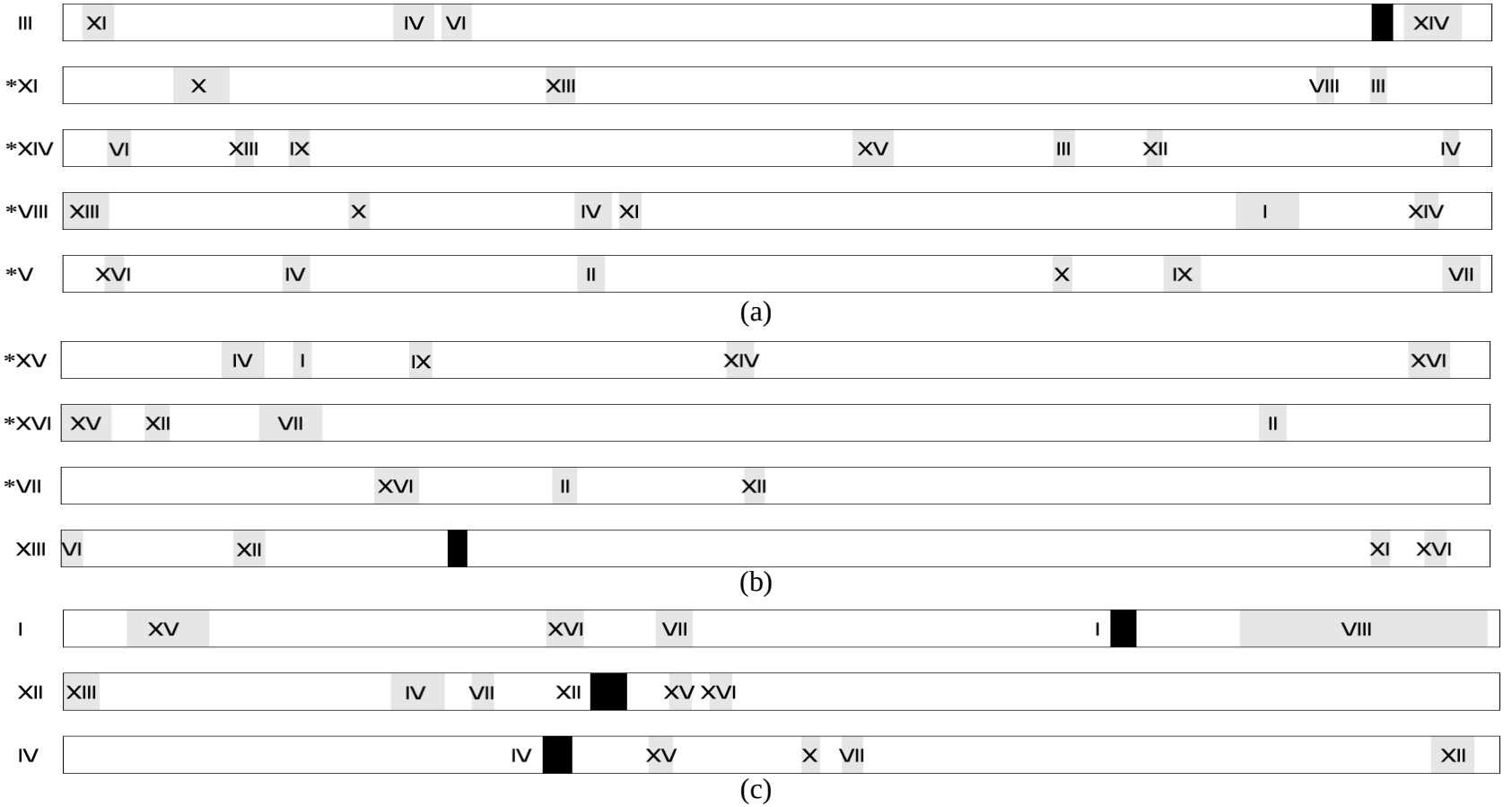


Figure 8. Locations of similar subsequences for (a) the first class, (b) the second class and (c) the third class. Self-similarity is shown in black color while cross-similarities with other chromosome sequences are in other colors. The sequence number of the chromosome sequence is marked inside the colored region. Only significant regions are presented and are drawn on scale with the chromosome sequence.

3.3 Analysis with Cross-chromosomal Predictions

We considered two cases for cross-chromosomal prediction. In the first case named prediction-2, the prediction is restricted to only two chromosome sequences including the current chromosome sequence. In the second case named prediction-16, the prediction is from the current chromosome sequence and the other 15 chromosome sequences. The self-prediction and cross-predictions are examined to remove all those overlapping regions and are sorted to produce a combined list. This combined list is then used to show all the repetitive regions including both self-chromosomal and cross-chromosomal repetitions.

Table 8 shows the experimental results. Column 2 and 3 give the class and the number of bases for a particular chromosome sequence denoted as Chr *b* respectively. Chr *a* in Column 4 is the most similar chromosome sequence with Chr *b* in Column 1. In Column 5, the sub-columns (a)(b) and (c)(d) provide the length of repetitive regions in cross-chromosomal prediction from Chr *a* (i.e. prediction-2) and from the other 15 chromosome sequences (i.e. prediction-16) respectively. The sub-columns (a)(c) and (b)(d) refer to cross-chromosomal and self-chromosomal predictions respectively.

In prediction-2, the cross-predictions come from another chromosome sequence that gives the longest similar subsequences. In Column 5(a) and (b), it is clear that the cross-predictions are always significant, as compared with the self-predictions. In particular, the cross-predictions are in the range of 5% to 22%. In contrast, the self-predictions are always less than 3.5%. In prediction-16, the cross-predictions from the other 15 chromosome sequences are in the range of 12.5% to 32% as listed in Column 5(c), whereas the self-predictions are always less than 3%. As a result, our study indicates that different chromosome sequences should be compressed together to take into account both self-similarity and cross-similarities.

We consider two different ways of using GenCompress in compressing two chromosome sequences. In the first way, these two chromosome sequences are compressed separately, i.e., only self-chromosomal similarities are considered. In the second way, the two chromosome sequences are compressed together, i.e., both self-chromosomal and cross-chromosomal similarities are considered. In Table 8, the total number of bits required for storing Chr *a* and Chr *b* without any compression is listed in Column 6(a). For the first case that considers self-chromosomal similarities only, the total number of bits required by GenCompress is shown in Column 6(b). In considering both self-chromosomal

and cross-chromosomal similarities, the total number of bits required is shown in Column 6(c). Column 7(a) shows the number of bits saved in self-chromosomal similarity case, it is obtained by calculating the difference between Column 6(a) and Column 6(b). It is obvious that less number of bits is required for compressing Chr *a* and Chr *b* together than that for compressing them separately; but the time taken for the former case is more than double of the latter case. Column 7(b) shows the additional saving in bits from cross-chromosomal repetition which is obtained by comparing Column 6(b) and Column 6(c). In other words, Column 7(a) is the savings resulting from self-chromosomal predictions as compared with the no compression case while Column 7(b) is the savings resulting from cross-chromosomal predictions as compared with no compression case. Column 7(c) is the savings from cross-chromosomal predictions as compared with the self-chromosomal predictions.

We can see that there is always extra savings by considering cross-chromosomal predictions in addition to self-chromosomal predictions. Since the cross-prediction found between Chr I and Chr VIII is the highest as shown in Column 5(a), the saving from cross-chromosome predictions is the largest. While the size of repetitive regions in cross-predictions ranged from 5% to 22%, their savings in bits are between 9% and 60%.

Table 8. Lengths of cross-chromosomal and self-chromosomal repetitions and the number of bits required/saved in compressing two chromosome sequences

1. Chr *b*	2. Class of Chr *b*	3. Length of Chr *b*	4. Chr *a*	5. Repetitive length in terms of the no. of bases (%)				6. Total no. of bits required for Chr *a* and Chr *b*			7. Total no. of bits saved (%) from		
				Prediction-2		Prediction-16							
				a. Cross-predictions	b. Self-predictions	c. Cross-predictions	d. Self-predictions	a. Without compression	b. Compressing separately	c. Compressing together	a. Self-predictions	b. Cross-predictions	c. % improvement of (b) over (a)
I	3	230208	VIII	50536 (22.0%)	5526 (2.4%)	74058 (32.2%)	4209 (1.8%)	1585702	1499256	1447264	86446 (5.8%)	138438 (9.6%)	51992 (60.1%)
III	1	316617	XIV	28818 (9.1%)	6416 (2.0%)	54714 (17.3%)	4737 (1.5%)	2201900	2112392	2096936	89508 (4.2%)	104964 (5.0%)	15456 (17.3%)
IV	3	1531918	XII	79909 (5.2%)	44897 (2.9%)	197093 (12.9%)	31532 (2.1%)	5220186	4855360	4815592	364826 (7.5%)	404594 (8.4%)	39768 (11.9%)
V	1	576869	VII	39909 (6.9%)	6859 (1.2%)	94421 (16.4%)	4094 (0.7%)	3335630	3177920	3149392	157710 (5.0%)	186238 (5.9%)	28528 (18.1%)
VII	2	1090946	XVI	66619 (6.1%)	17936 (1.6%)	156422 (14.3%)	5812 (0.5%)	4078016	3881368	3841968	196648 (5.1%)	236048 (6.1%)	39400 (20.0%)
VIII	1	562643	I	36808 (6.5%)	15086 (2.7%)	104628 (18.6%)	6129 (1.1%)	1585702	1499256	1447432	86446 (5.8%)	138270 (9.6%)	51824 (59.9%)
XI	1	666454	X	35013 (5.3%)	3930 (0.6%)	85186 (12.8%)	2655 (0.4%)	2824398	2729104	2720464	95294 (3.5%)	103934 (3.8%)	8640 (9.1%)
XII	3	1078175	IV	87678 (8.1%)	36310 (3.4%)	164488 (15.3%)	27744 (2.6%)	5220186	4855360	4816424	364826 (7.5%)	403762 (8.4%)	38936 (10.7%)
XIII	2	924429	XII	51845 (5.6%)	17079 (1.9%)	117607 (12.7%)	12670 (1.4%)	4005208	3742920	3713616	262288 (7.0%)	291592 (7.9%)	29304 (11.2%)
XIV	1	784333	XV	51084 (6.5%)	8952 (1.1%)	122687 (15.6%)	6396 (0.8%)	3751244	3604120	3566944	147124 (4.1%)	184300 (5.2%)	37176 (25.3%)
XV	2	1091289	IV	70056 (6.5%)	14168 (1.3%)	183165 (16.8%)	7434 (0.7%)	5246414	4973664	4931832	272750 (5.5%)	314582 (6.4%)	41832 (15.3%)
XVI	2	948062	VII	67662 (7.1%)	8658 (0.91%)	145116 (15.3%)	4860 (0.5%)	4078016	3881368	3845376	196648 (5.1%)	232640 (6.0%)	35992 (18.3%)
			Average	55495 (7.9%)	15485 (1.8%)	124965 (16.7%)	9856 (1.2%)	3594384	3401007	3366103	193376 (5.7%)	228280 (6.8%)	34904 (23.0%)

We find that cross-chromosomal similarities are always significant as compared with self-chromosomal similarities. For example, the average percentage of similar subsequences between two chromosome sequences is about 10% in which 8% comes from cross-chromosomal prediction and 2% from self-chromosomal prediction. For the 16 chromosome sequences of *S. cerevisiae*, the average percentage is about 18% in which 16.8% comes from cross-chromosomal prediction and 1.2% from self-chromosomal prediction. Therefore, it would be advantages to compress different chromosome sequences together to take advantage of cross-chromosomal similarities.

Our experimental results in Table 8 demonstrate that on average an additional 23% of storage is reduced in cross-chromosomal predictions as compared with self-chromosomal predictions. Therefore, a high compression ratio could be obtained by considering both self-prediction and cross-predictions for the entire set of chromosome sequences.

3.4 Chapter Summary

In this chapter, a detailed study of the sixteen chromosome sequences of *S. cerevisiae* has been described. Our study indicated that the length of similar repeated regions within one chromosome sequence is about 4.5% of the total

sequence length. In contrast, the average percentage of similar subsequences between two chromosome sequences is about 10%, in which only 2% comes from the self-chromosome sequence. In characterizing similarities of the sixteen chromosome sequences, the percentage of similar subsequences is about 18%, in which only 1.2% comes from the self-chromosome sequence, while the rest is from the other fifteen sequences. This indicates that it would be highly advantageous to consider cross-chromosomal similarities in addition to self-chromosomal similarities in DNA sequence compression.

4 Our proposed multiple sequence compression algorithm

Cross-chromosomal similarities have found to be as important as self-chromosomal similarity. However, state-of-the-art compression algorithms work by finding self-similar subsequences within the current sequence only. In particular, identical subsequences are encoded by reference to their previous occurrences to achieve compression. These algorithms thus ignore the cross-chromosomal similarities completely. We proposed a multiple sequence compression algorithm that takes both the self-chromosomal and cross-chromosomal similarities into account. Identical subsequences within a number of sequences are identified and encoded together to achieve data compression. One may argue that more sequences would have to be sent if multiple sequence compression is considered. In fact, researchers always download a complete set of chromosome sequences to study their characteristics. Thus, multiple sequence compression can be used.

This chapter is organized as follows. First, an overview about the proposed compression method will be presented. Second, two components of our proposed algorithm, namely the DNAComp coder and Arith-2 coder, are

discussed. Finally, we will discuss the performance measures for our proposed algorithm.

4.1 Overview

Our proposed compression algorithm adopts a similar approach as DNACompress [3]. In particular, PatternHunter [27] is used for finding repetitive subsequences from a number of chromosome sequences. A strategy is then developed to remove the overlapping regions in these repetitive subsequences. Similar subsequences are then sorted according to their importance and encoded together. Finally, Arith-2 coder is used to further minimize the size of the sequences. In the following, the structure of the encoding and the decoding processes are briefly discussed.

4.1.1 Encoding Processes

The encoding process as shown in Figure 9 includes two encoders called DNAComp and Arith-2 encoders. Firstly, the original file containing the uncompressed chromosome sequence

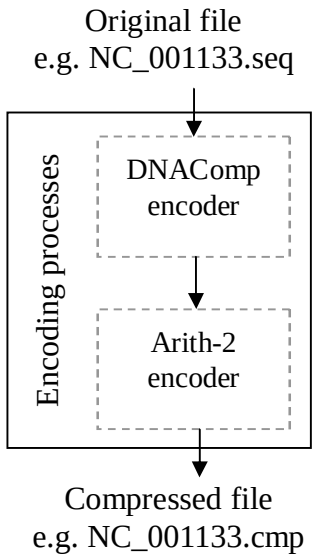


Figure 9. The encoding processes.

(such as NC_001133.seq) is inputted into the DNAComp encoder. An example of the input sequence "NC_001133.seq" is shown in Figure 10. Secondly, the output of the DNAComp encoder is passed to Arith-2 encoder to make a compressed file. The file size of the compressed file is expected to be smaller than that of the original file.

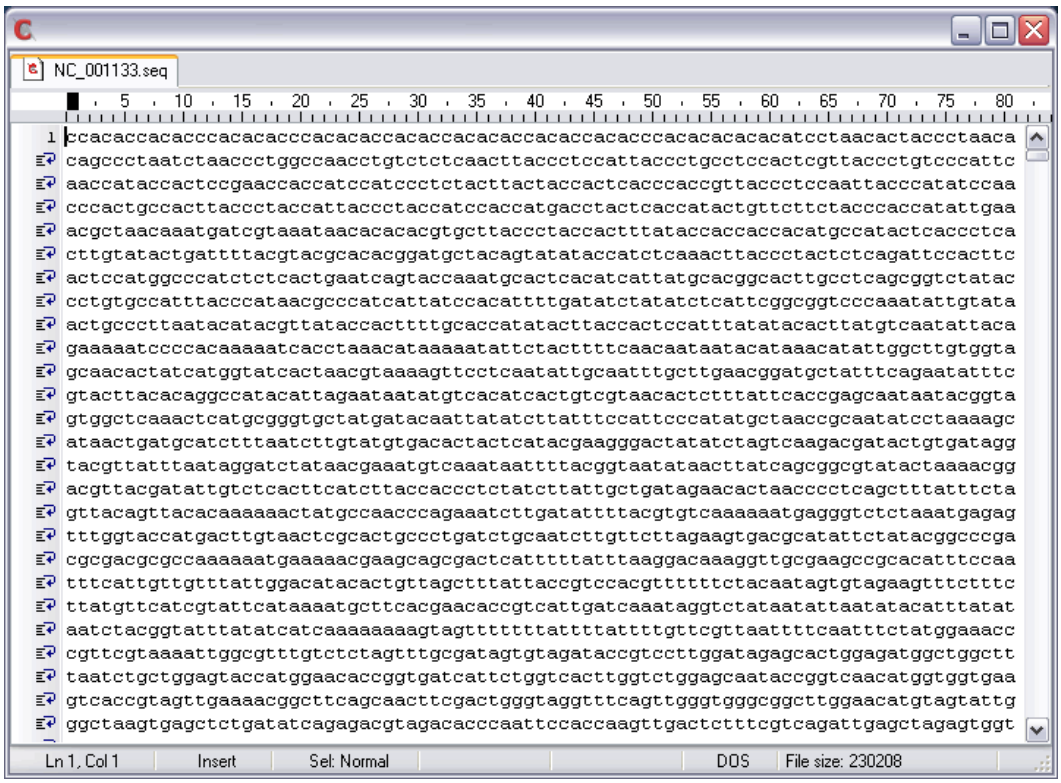


Figure 10. An example of a sequence called NC_001133.seq.

4.1.2 Decoding Processes

Figure 11 illustrates details of the decoding process which reverses the order of the encoding process in Figure 9. After the compressed file is passed through the two decoders (Arith-2 and DNAComp decoders), the reconstructed

file is formed. Due to the use of lossless mechanism, it is expected that the reconstructed file in Figure 11 is exactly the same as the original file in Figure 9, i.e., NC_001133de.seq is the same as NC_001133.seq.

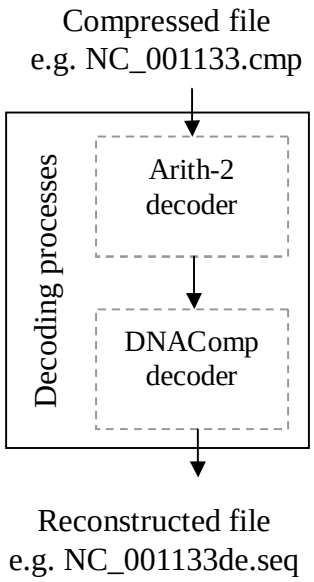


Figure 11. The decoding processes.

4.2 DNAComp coder

DNAComp coder is a core component of our proposed compression algorithm. Its main function is to identify similar subsequences among a set of DNA sequences and then encodes these similar subsequences together to achieve bits savings. The skeleton of the encoder and the decoder of DNAComp will be introduced below.

4.2.1 Encoder

There are altogether five steps in the DNAComp encoder. These steps are as follows,

1. Extraction of similar subsequences within a DNA sequence or from a number of DNA sequences;
2. Ordering of similar subsequences according to their importance;
3. Removal of overlapping similar subsequences;
4. Reordering of non-overlapping similar subsequences according to their position; and
5. Preparation of final sequences for further compression using Arith-2 coder.

In the first step, similar subsequences among a number of chromosome sequences are extracted using PatternHunter. Here similar subsequences mean either approximate repeats or approximate reverse repeats. Also, similar subsequences can refer to those subsequences found from the current sequence or from any one of the other chromosome sequences. For example, if there are two chromosome sequences, we will have results stored at two "aln" files:

- "self.aln": stores the self-similar subsequences;
- "reference.aln": stores the cross-similar subsequences found from another sequence.

In each aln file, information about similar subsequences including the scores, the direction, the starting and the ending positions of query and subject sequences is recorded. An example record is shown in Figure 12. The query and the subject sequences refer to the similar subsequences in the current sequence in the case of self-referencing (i.e., self.aln). They refer to the reference sequence and the current sequence respectively in the case of cross-referencing (i.e., reference.aln). The scores relate to the similarity between the query and the subject sequences. High score indicates that they are similar to each other. The direction can be either 'plus' or 'minus'. The 'plus' direction means an approximate repeat, so the sequence should be read in ascending order. The 'minus' direction means a reverse complement repeat, hence the sequence should be read in reverse direction. The starting and ending position of the subsequence marks its location in the original sequence. A list is created for storing information about repetitive records with score over a threshold. The threshold is set so that only significant similar subsequences are considered by our proposed compression algorithm.

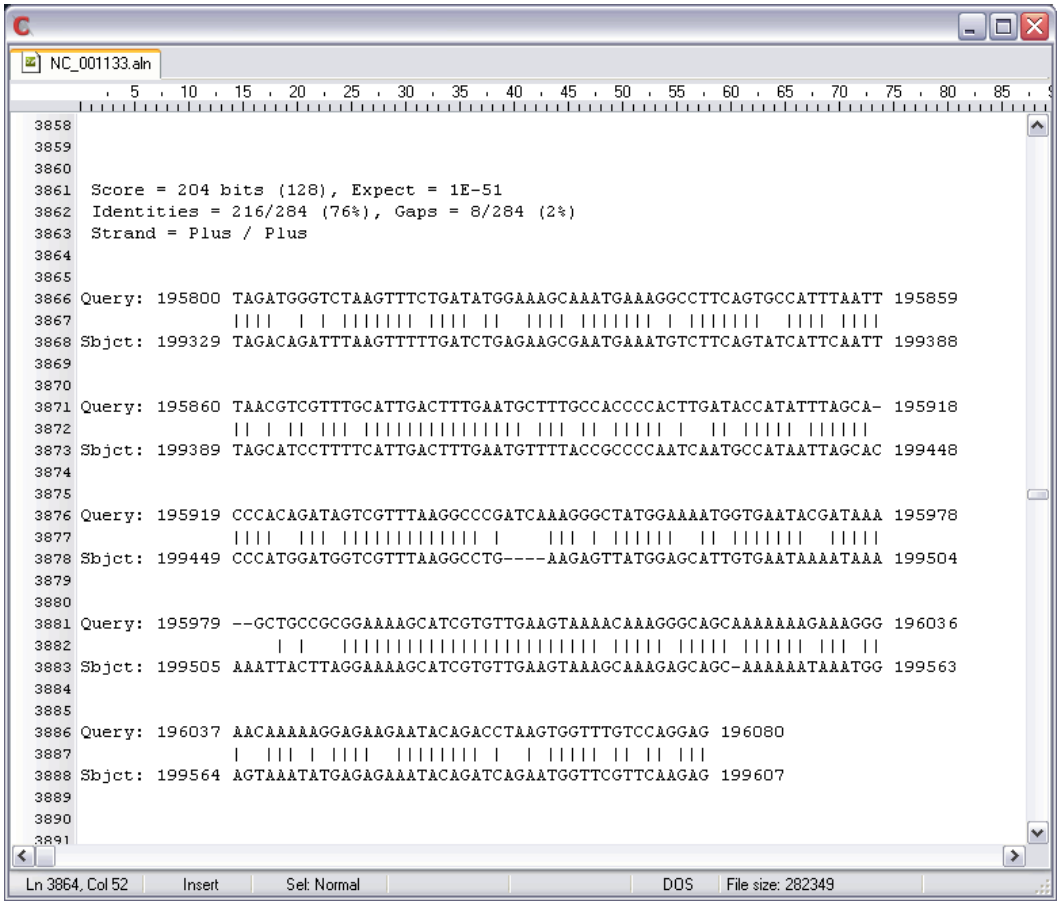


Figure 12. An example record of NC_001133.aln.

In the third step, each repetitive record in the combined list from the second step is examined. In particular, overlapping similar subsequences from two records are trimmed. If the subject sequences in two repetitive records are overlapped with each other in position, the overlapping part will be kept in the record with a higher score and be removed in the other record with a lower score. The rationale behind is to keep a long repetitive length rather than a short one. If the length of the trimmed repetitive record is less than the threshold after removing the overlapping part, the repetitive record will be removed from the list.

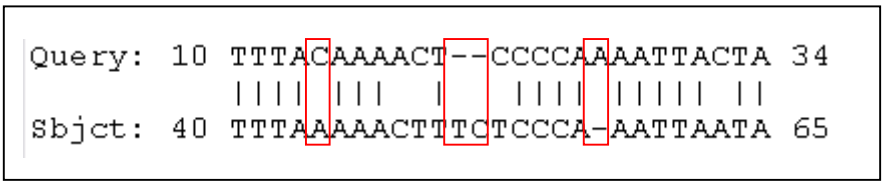
After removing all the overlapping parts in the similar subsequences, DNAComp saves the differences between the two similar subsequences. This step is essential as approximate repeats, rather than exact repeats, are considered in DNAComp. Thus, operations such as deletion, insertion and substitution might be required to match the two subsequences (see Section 2.1.1 for details).

Each repetitive record contains two lists, namely an offset list and a base list, for storing information about the operations. The offset list essentially marks the relative positions at where the bases are different in the two similar subsequences. Besides, the symbol '0' in the offset list is used to indicate an insertion operation. The base list stores the replacement base for a substitution operation and the inserted base for an insertion operation. The symbol "-" in the base list is used to indicate a deletion operation. It is not necessary to store the bases which are deleted in the query sequence, but the relative position must be stored in the offset list.

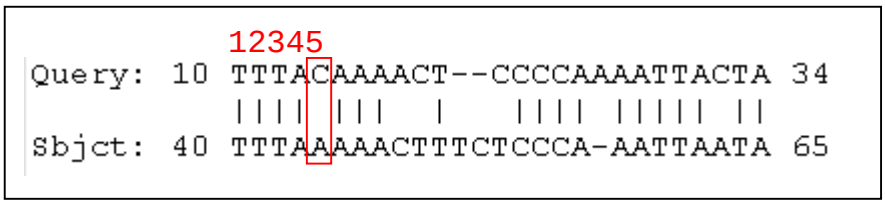
```
Query:  10 TTTACAAAACT--CCCCAAAATTACTA 34
           |||| |||   |    |||| ||||| ||
Sbjct:  40 TTTAAAAACTTTCTCCCA-AATTAATA 65
```

Consider a simple example shown above. The query subsequence from the 10^{th} base to the 34^{th} base and the subject subsequence from the 40^{th} base to

the 65th base are found to be an approximate repeat. This example includes all the operations - substitution, deletion and insertion.

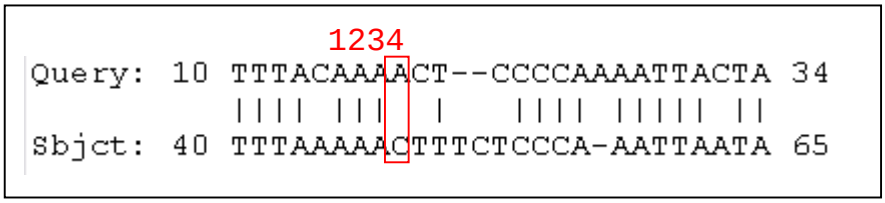


For example, the base "C" in the fifth position of the query subsequence is replaced by "A" in the corresponding position of the subject subsequence. There are two more bases added in the subject subsequence in between the twelve and the thirteen bases of the query subsequence. The twenty-sixth base "A" in the query subsequence is deleted in matching to the subject subsequence.



Initially, the base list and the offset list are empty. In matching the query and the subject subsequences, the first operation is substitution where the base 'C' is replaced by the base 'A' in the 5th position. So '5' is added to the offset list for indicating the location of the change and 'A' is stored in the base list, i.e.,

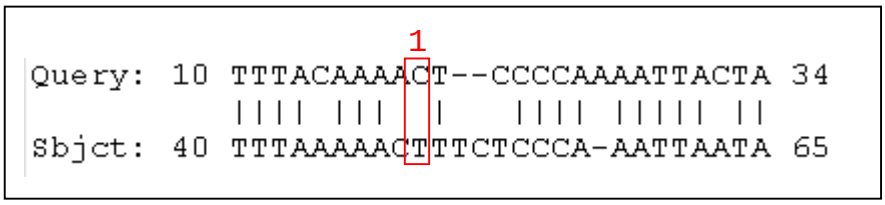
Offset list = {5}
Base list = {A}



As relative positioning is used, the 5th position of the query subsequence is reset so the next position becomes the 1st position now. The second substitution operation is at the 4th position, so '4' is added to the offset list and 'C' is stored in the base list, i.e.,

Offset list = {5, 4}

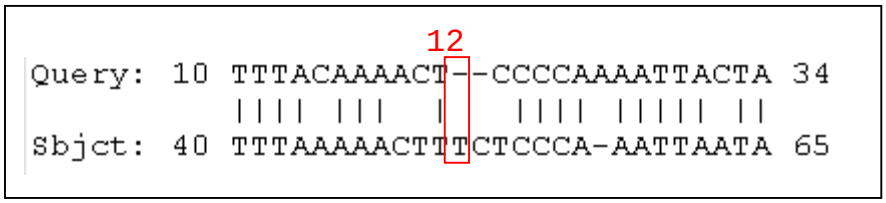
Base list = {A, C}



Similarly, the third operation is substitution, so the new lists become,

Offset list = {5, 4, 1}

Base list = {A, C, T}

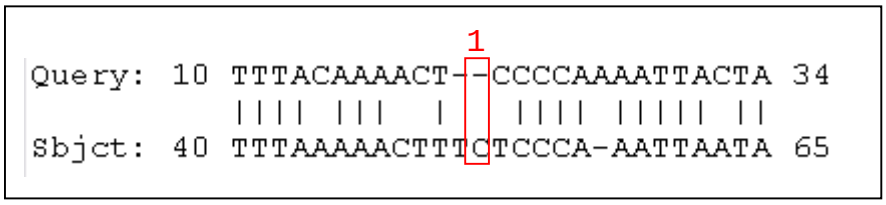


The fourth operation is an insertion. The symbol '0' is first inserted to the offset list to indicate that a base will be inserted in the subject subsequence. The

added base is ‘T’ which is to be added at the 2nd position, so ‘2’ is added to the offset list and ‘T’ is stored in the base list. Thus, the new lists become,

Offset list = {5, 4, 1, 0, 2}

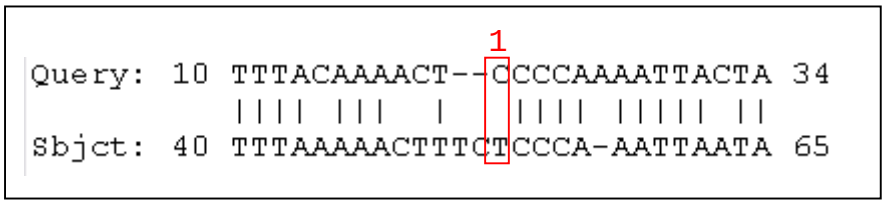
Base list = {A, C, T, T}



The next operation is in insertion where the base to be added is ‘C’ just next to the previous operation. Hence the two lists are,

Offset list = {5, 4, 1, 0, 2, 0, 1}

Base list = {A, C, T, T, C}



The next operation is a substitution, where the base ‘C’ is replaced by ‘T’, i.e.,

Offset list = {5, 4, 1, 0, 2, 0, 1, 1}

Base list = {A, C, T, T, C, T}

```
                                12345
Query:  10 TTTACAAAACT--CCCCAAAATTACTA 34
           |||| |||   |    |||| ||||| ||
Sbjct:  40 TTTAAAAACTTTCTCCCA-AATTAATA 65
```

Deletion is then required in matching the two subsequences. In particular, the base 'A' is deleted from the query sequence. '5' is added to the offset list since this is the 5th position after the previous operation, and the symbol '-' is stored in the base list to indicate a deletion. Hence we have,

Offset list = {5, 4, 1, 0, 2, 0, 1, 1, 5}

Base list = {A, C, T, T, C, T, - }

```
                                   123456
Query:  10 TTTACAAAACT--CCCCAAAATTACTA 34
           |||| |||   |    |||| ||||| ||
Sbjct:  40 TTTAAAAACTTTCTCCCA-AATTAATA 65
```

The final operation is a substitution where the base 'C' is based by 'A', i.e.,

Offset list = {5, 4, 1, 0, 2, 0, 1, 1, 5, 6}

Base list = {A, C, T, T, C, T, -, A}

```
                                      12
Query:  10 TTTACAAAACT--CCCCAAAATTACTA 34
           |||| |||   |    |||| ||||| ||
Sbjct:  40 TTTAAAAACTTTCTCCCA-AATTAATA 65
```

Since the end positions of both query subsequence and subject subsequence are not marked, it is important to include the offset from the final operation to the end of the subsequence. The resultant offset list and base list are,

Offset list	5	4	1	0	2	0	1	1	5	6	2
Base list	A	C	T	T	C	T	-	A			

Afterwards, the non-overlapping repetitive records are sorted according to their starting positions. Therefore, redundancy can be removed sequentially from one end of the sequence to the other end. Finally, similar subsequences are removed from the sequence to form another sequence that contains non-repetitive subsequences only. This non-repetitive sequence will be sent to Arith-2 for further compression. Besides the non-repetitive sequence, Arith-2 will also be used to encode the offset and the base lists, the starting positions and the direction of the query and the subject sequences.

4.2.2 Decoder

Basically, DNAComp decoder reverses the operations done in the encoder. However, the structure of the decoder is much simpler than the DNAComp encoder. In particular, there is no need to identify similar subsequences among a number of chromosome sequences. Also, no overlapping detection or sorting is required to be performed in the decoder.

First, the Arith-2 decoder will send out a sequence containing non-repetitive records only. Similar subsequences are then needed to be added back to the non-repetitive sequence sequentially. In the encoder, operations such as deletion, insertion and substitution to match two subsequences have been

recorded. Thus, this information is used to construct the similar subsequences which are then added back to the non-repetitive sequence. Note that the addition of the similar subsequences is done sequentially according to their starting positions. The original sequence should be reconstructed losslessly after all the repetitive records have been added.

4.3 Arith-2 coder

Arith-2 stands for two-order arithmetic coder or second order finite-context arithmetic coder. It is used to further compress the shortened non-repetitive sequence and the related information. Arithmetic coding [44-47] replaces a stream of symbols with a single floating-point output number, which is less than 1 and greater than or equal to 0. It is a lossless compression scheme so that the original stream of symbols can be uniquely reconstructed. Initially, the probability of each symbol is assigned. The output number, represented by a sub-interval of the cumulative probability of the symbol sequence, is formed by recursively sub-dividing the interval between 0 and 1. Arith-2 is always involved in DNA-based compression, such as Biocompress-2 and GenCompress, because it can compress DNA sequence efficiently. The compression gain is higher when the current base and the base that are two bases apart from the current base are the same. Indeed, order-2 means the context to be used are the last two symbols.

This seems to correspond to the codons structure of amino-acid in a protein [2, 26]. An adaptive arithmetic coder is used so that the occurrence probabilities of A, C, G and T in a sequence can be calculated and updated inside the coder. The source code and document of the arithmetic coder utilized is available from [36-39].

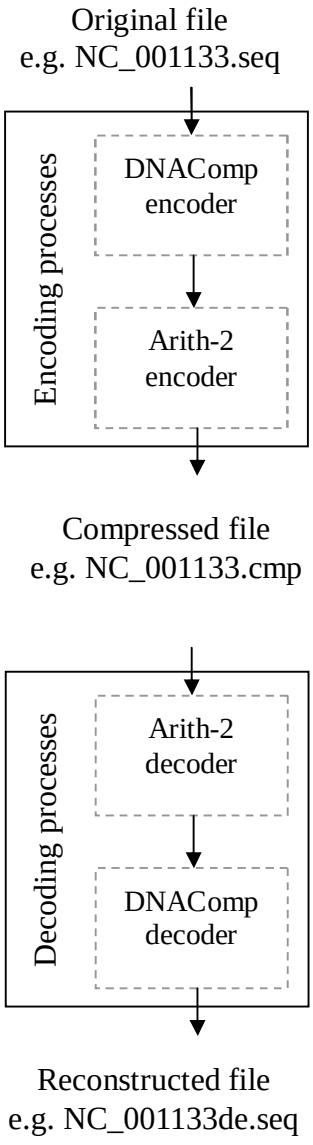


Figure 13. The coder.

4.4 Performance Measurement

Due to the use of a lossless compression scheme, there is no difference between the original and the reconstructed DNA sequences. The performance of our proposed compression algorithm is measured by the execution time, bits per base (bpb) used and compression gain.

The execution time is counted by adding a timer at the beginning of the encoding process and the end of the decoding process as shown in Figure 13. The value of bpb can be calculated by |O|/|I| where |O| is the number of bits for the compressed file and |I| is the number of bases of the original sequence. The compression gain is obtained by 1-|O|/2|I| as two bits are required for each base without compression.

4.5 Chapter Summary

In this chapter, a new multiple sequence compression algorithm has been proposed to take into account both self-sequence and cross-sequence similarities. Our proposed algorithm contains the following steps: 1) to search for all the similar subsequences among a number of sequences that are to be compressed together; 2) to identify and remove the overlapping subsequences; 3) to form two lists containing operations and the relative positions that are required in matching

two similar subsequences; 4) to identify the non-repetitive sequences; and 5) to use an Arith-2 coder to compress the non-repeating sequences, the offset and the base lists, the starting positions and the direction of the query and the subject sequences. Then the decoder and the principle of Arith-2 coder have been introduced briefly. Finally, the performance measures for our proposed algorithm, including the execution time, bpb used and compression gain have been discussed

5 Simulation Results

To evaluate the effectiveness of our proposed algorithm, two real datasets are considered. They are the sixteen chromosome sequences of *S. cerevisiae* and the three chromosome sequences of *S. pombe*. The average length of *S. cerevisiae* and *S. pombe* are 800k and 4200k respectively. Therefore, we can study the performance of our proposed algorithm on cases where there is a large number of chromosome sequences and where there is large variation of average chromosome length. We first present the results on single sequence compression, which is followed by multiple sequence compression.

5.1 Simulation Results on *S. cerevisiae*

In this experiment, we consider compressing the sixteen chromosome sequences of *Saccharomyces cerevisiae* (*S. cerevisiae*). These sequences can be downloaded from ftp://ftp.ncbi.nlm.nih.gov/genomes/. The average length of *S. cerevisiae* is 800k.

5.1.1 Single sequence compression

Table 9 shows the resultant bits per base (bpb) used of our proposed compression algorithm in compressing each chromosome sequence separately. A comparative study with gzip [40], Arith [36-39], CTW [7], and GenComp [4-6] is

also performed. Column 1 in Table 9 shows the current chromosome sequence to be compressed. Column 2 specifies the number of bases in the chromosome sequence. Column 3 specifies the bpb for gzip. Without any compression, 2 bpb is required. Thus, the average bpb of 2.32 achieved by gzip is not satisfactory. It cannot compress, but expand, the size of the sequence. In fact, the result is expected as gzip is a general purpose compression scheme and does not design to capture characteristics in a DNA sequence.

Column 4 of Table 9 shows the bpb for Arith. The average bpb is slightly less than 2. Note that for cases like Chr I and Chr III, Arith uses more than 2 bpb. Column 5 shows the results for CTW. Its performance is rather stable as the bpb for all the chromosome sequences are between 1.94 and 1.95. Column 6 shows the bpb for the GenCompress. Its performance is the best as the average bpb is 1.91 only. Column 7 shows the bpb for our proposed algorithm. It is slightly higher than that for the GenCompress.

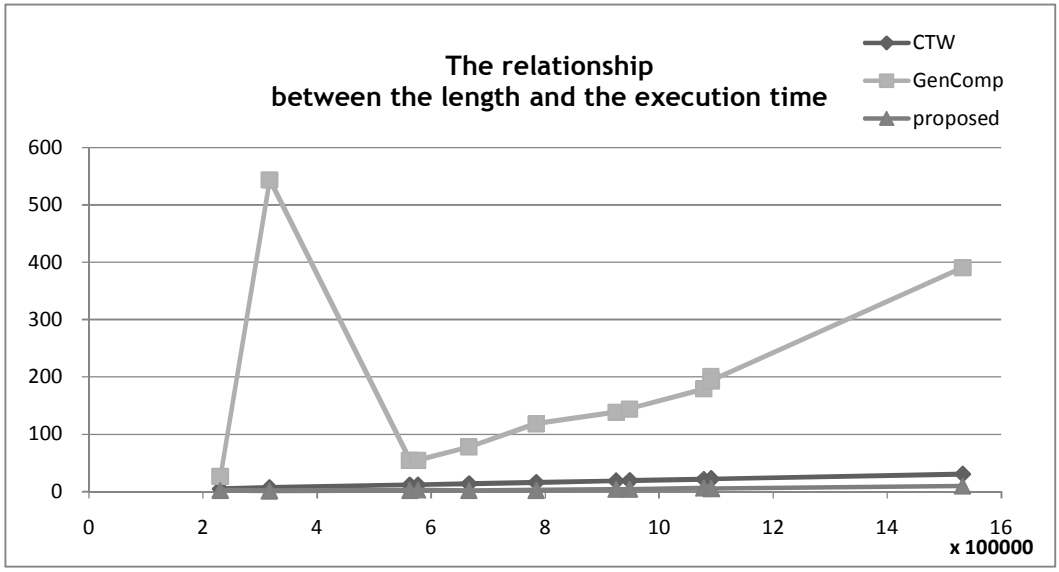


Figure 14. The relationship between the length and the execution time of the sixteen chromosomes in *S. cerevisiae*. The performance of CTW, GenCompress and the proposed algorithm are repseneted by the line with rhombus, square and triangle respectively. X-axis marks the length of the chromosome sequences while y-axis marks the execution time in seconds(s).

Besides the compressibility indicated by bpb, another important consideration is the execution time. As DNA is a long sequence, finding repetitive subsequences is often time-consuming. Table 10 shows the execution time used by various compression algorithms. We can see that on average, the execution times of gzip and Arith are very short. They use less than 1 second to compress the long DNA sequences. In contrast, GenCompress has the longest executive time. It spends long time in searching for repetitive records, especially for long sequences such as Chr III, Chr IV, Chr VII and Chr XV. We can see that the time increases non-linearly for sequences with an average length more than 800k. Our proposed algorithm uses about 4 seconds, which compares

favorably with GenComp and CTW. Figure 14 shows the relationship between the length and the execution time of the sixteen chromosome sequences in *S. cerevisiae*. The line with rhombus, square and triangle represent the performance of CTW, GenComp and the proposed algorithm respectively. In general, the execution time increases exponentially with the length of the sequences for GenComp. However, for both CTW and our proposed algorithm, the execution time increases linearly with the length of the sequences. There is one exception case for GenComp, the execution time is about 550s for Chr III which contains about 300K bases. Although Chr III is not a long sequence, GenComp takes over ten hours to search for similar sequences since less similar sequences can be found within Chr III (see Figure 6b).

Table 9. The bpb (bits pre base) of compressing the 16 chromosome sequences in *S. cerevisiae.*

cur	Length	gzip	Arith	CTW	GenComp	proposed
I	230208	2.30	2.02	1.95	1.83	1.88
II	813178	2.33	1.98	1.94	1.93	-- --
III	316617	2.33	2.00	1.94	1.91	1.94
IV	1531918	2.31	1.97	1.94	1.88	1.90
V	576869	2.33	1.99	1.95	1.91	1.93
VI	270148	2.34	2.01	1.95	1.95	-- --
VII	1090946	2.33	1.97	1.94	1.91	1.92
VIII	562643	2.32	1.99	1.95	1.91	1.94
IX	439885	2.32	1.99	1.95	1.93	-- --
X	745745	2.32	1.98	1.95	1.93	-- --
XI	666454	2.33	1.98	1.94	1.94	1.96
XII	1078175	2.27	1.98	1.94	1.84	1.86
XIII	924429	2.31	1.98	1.94	1.91	1.93
XIV	784333	2.33	1.98	1.95	1.92	1.94
XV	1091289	2.33	1.97	1.95	1.92	1.94
XVI	948062	2.33	1.97	1.94	1.90	1.92
	average	**2.32**	**1.98**	**1.95**	**1.91**	**1.92**

Table 10. The execution time (seconds) of compressing 16 chromosome sequences in *S. cerevisiae.*

cur	Length	gzip	Arith	CTW	GenComp	proposed
I	230208	0.07	0.15	4.90	26.67	2.05
II	813178	0.24	0.37	16.45	115.68	-- --
III	316617	0.09	0.17	6.69	544.27	1.25
IV	1531918	0.45	0.64	30.75	390.70	9.89
V	576869	0.17	0.27	11.78	54.72	2.46
VI	270148	0.08	0.15	5.57	42.48	-- --
VII	1090946	0.31	0.48	22.00	200.93	5.78
VIII	562643	0.16	0.27	11.57	54.35	2.34
IX	439885	0.13	0.22	9.12	106.53	-- --
X	745745	0.24	0.34	15.12	89.59	-- --
XI	666454	0.19	0.31	13.56	77.88	1.87
XII	1078175	0.31	0.47	21.77	179.18	6.08
XIII	924429	0.27	0.41	18.66	138.80	3.93
XIV	784333	0.23	0.35	16.01	118.27	2.34
XV	1091289	0.31	0.47	22.11	193.48	5.14
XVI	948062	0.27	0.42	19.11	144.46	4.24
	average	**0.22**	**0.34**	**15.32**	**154.87**	**3.95**

In summary, our proposed algorithm maintains a good balance in compressibility and execution time. Despite that, single-sequence compression considers only self-repetition. Although GenComp has the best result of 1.91, it achieves a saving of 0.09 only as compared with the no-compression case. In the next section, cross-sequence repetition is considered in a hope to reduce the bpb.

5.1.2 Two-sequence compression

Since the academic version of PatternHunter does not work properly for Chr II, Chr VI, Chr IX and Chr X, the remaining twelve chromosome sequences of *S. cerevisiae* are considered in this part. The first case we considered is the potential advantage of compressing one chromosome sequence if one more chromosome sequence is given. Table 11 shows the experimental results of compressing a sequence by reference to its most similar sequence obtained from Table 8.

Column 1 of Table 11 shows the chromosome sequence to be compressed (denoted as "cur") while Column 2 shows the reference sequence (denoted as "ref"). In other words, "cur" is compressed by considering both self-repetition and cross-sequences repetition with "ref". Column 3 shows the number of bases in "cur". Column 4 is the bpb used for single-sequence compression, thus, it

considers self-repetition only. Column 5 is the bpb used for two-sequence compression in which similar subsequences from the “cur” and the “ref” are considered. The sixth and the last column list the additional saving in bpb and the saving percentage (%) respectively. For Chr I, the bpb of compressing itself alone is 1.8838 while that of compressing Chr I with reference to itself and Chr VIII is 1.4656. Hence, the saving is around 0.42 bpb (i.e., 22%). In the second row, the bpb of compressing Chr III with reference to itself and Chr XIV is 1.7853 while that of compressing itself alone is 1.9430, so the saved bpb is around 0.16 and saving percentage is around 8.1%.

For each chromosome sequence shown in Table 11, we can see that the bpb is always lower when the current sequence is compressed by considering the reference sequence in addition to itself. In fact, the additional savings depend on the similarity between the current and the reference chromosome sequences. An interesting observation is that the additional savings for the long sequences such as Chr IV, Chr VII, Chr XII and Chr XV are not too high as compared with that for the short sequences. Further investigation on the biological significance is required.

Table 11. The experimental results of compressing a sequence in *S. cerevisiae* provided with a reference sequence.

cur	ref	Length	Without given ref sequence (bpb)	With given ref sequence (bpb)	Additional saving (bpb)	Additional saving (%)
I	VIII	230208	1.8838	1.4656	0.4182	22.2%
III	XIV	316617	1.9430	1.7853	0.1577	8.1%
IV	XII	1531918	1.8964	1.8473	0.0491	2.6%
V	IV	576869	1.9286	1.8312	0.0974	5.0%
VII	XV	1090946	1.9244	1.8649	0.0595	3.1%
VIII	I	562643	1.9397	1.7686	0.1711	8.8%
XI	XIII	666454	1.9614	1.9135	0.0479	2.4%
XII	IV	1078175	1.8557	1.7859	0.0698	3.8%
XIII	XII	924429	1.9270	1.8550	0.0720	3.7%
XIV	XV	784333	1.9402	1.8526	0.0876	4.5%
XV	IV	1091289	1.9378	1.8666	0.0712	3.7%
XVI	VII	948062	1.9204	1.8519	0.0685	3.6%
		average	**1.9215**	**1.8074**	**0.1142**	**5.9%**

In the last row of Table 11, the average bpb of compressing a sequence alone is 0.11 bpb larger than that of compressing a sequence with reference to another sequence. Thus, from 1.92 to 1.81, an additional 5.9% savings in bpb can be achieved.

Figure 15 shows the distribution of self-similarity and cross-similarities for various chromosome sequences in *S. cerevisiae*. The dark grey color bar displays the proportion of similar repetitive subsequence found from the sequence itself while the light grey color bar shows the proportion of similar repetitive subsequence found from the reference sequence where the reference

sequence is defined as the most similar sequence as shown in Column 2 of Table 11. For example, in Chr I, the self-similarity is about 18% when the cross-similarity is about 82%. We can see the repetition found from reference sequence is always more than 60% of the whole repetitive part.

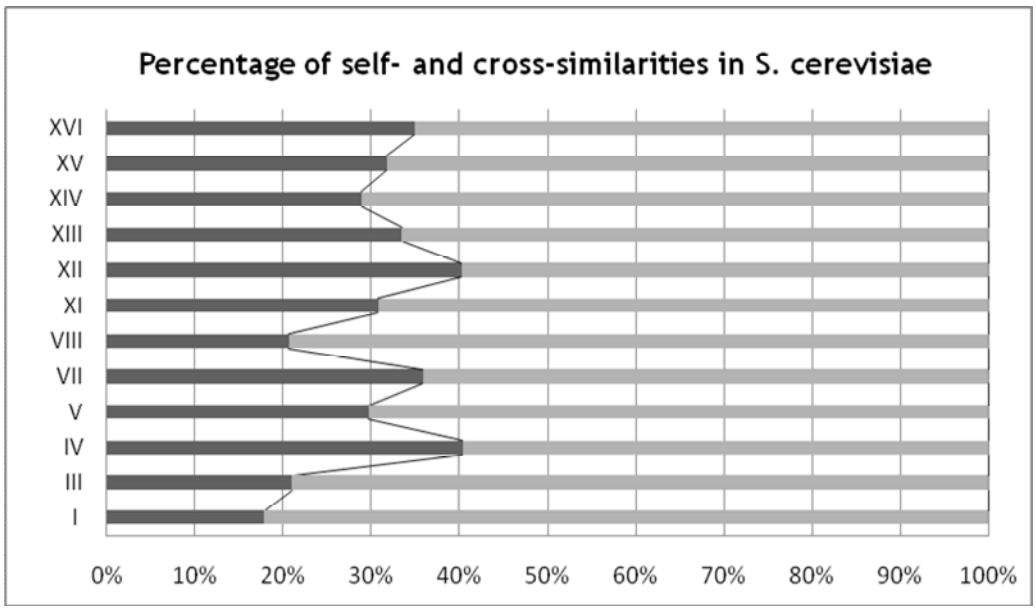


Figure 15. The percentage of self-sequence and corss-sequence similarities in *S. cerevisiae*. The dark grey and the light grey color bars indicate the proportion of the self-similarity and the cross-similarities respectively.

There are four steps in compressing two sequences, namely finding self-repetitions, finding cross-repetitions, performing compression in DNAComp encoder and further compression by Arith-2 encoder. Table 12a and Table 12b list the execution times (in second) for each individual steps in compressing 2 sequences together in *S. cerevisiae*. In Table 12a, Column 3 and Column 4 represent the time required for finding repetitions in self sequence and reference

sequence respectively. The time required for the DNAComp encoder and Arith-2 encoder are shown in Column 5 and Column 6 respectively. The total encoding time, which is the sum of the time required for each of the four steps, is listed in the last column of Table 12a. The average encoding time is 8.82 seconds. In Table 12b, the time required for encoding and the decoding are shown in Column 4 and Column 5 respectively. The total execution time is listed in Column 6. The average encoding time is 2s more than the decoding time due to the need to find repetitive sequences.

Next, we consider the compression results in compressing two chromosome sequences together. In other words, both current and reference sequences are compressed by cross-referencing. Table 13 lists the results for *S. cerevisiae*. Column 1 lists the two chromosome sequences in a group which will be compressed together. Column 2 shows the total number of bases of a particular chromosome sequence denoted as Chr. Column 3 and Column 4 list the number of bases of non-repetitive sequence and the bpb, respectively, when compressing the two chromosome sequences separately. Thus, 1.8838 bpb and 1.9387 bpb are required respectively for compressing Chr I and Chr VIII alone and the average bpb is 1.9236. Column 5, Column 6 and Column 7 show respectively the number of bases of non-repetitive sequence, bpb and execution

time in seconds when the two chromosome sequences are compressed together. Thus, 1.8020 bpb is required if Chr I and Chr VIII are compressed together. It takes only 7.5 seconds to compress Chr I and Chr VIII together.

Table 12a. The encoding time(sec) of compressing 2 sequences together in *S. cerevisiae.*

cur	ref	Self-repetitions	Cross-repetitions	DNAComp	Arith-2	Total Encoding Time
I	VIII	0.86	1.34	1.98	0.15	4.33
III	XIV	1.11	1.66	2.02	0.15	4.94
IV	XII	3.52	3.77	9.33	0.19	16.80
V	IV	1.33	3.42	4.05	0.15	8.95
VII	XV	2.31	2.97	5.95	0.17	11.40
VIII	I	1.11	1.22	2.09	0.15	4.57
XI	XIII	1.33	2.12	2.16	0.16	5.77
XII	IV	2.20	3.92	8.32	0.17	14.61
XIII	XII	1.78	2.86	5.02	0.16	9.83
XIV	XV	1.34	2.22	3.34	0.15	7.05
XV	IV	2.09	1.78	3.48	0.17	7.52
XVI	VII	1.81	3.08	5.05	0.16	10.10
average		**1.73**	**2.53**	**4.40**	**0.16**	**8.82**

Table 12b. The execution time(sec) including encoding and decoding time of compressing 2 sequences together in *S. cerevisiae.*

cur	ref	Length	Encoding Time	Decoding Time	Total Time
I	VIII	230208	4.33	3.18	7.50
III	XIV	316617	4.94	6.73	11.67
IV	XII	1531918	16.80	10.38	27.18
V	IV	576869	8.95	4.10	13.05
VII	XV	1090946	11.40	14.28	25.68
VIII	I	562643	4.57	3.16	7.73
XI	XIII	666454	5.77	5.68	11.45
XII	IV	1078175	14.61	6.01	20.62
XIII	XII	924429	9.83	7.01	16.84
XIV	XV	784333	7.05	7.01	14.06
XV	IV	1091289	7.52	9.06	16.57
XVI	VII	948062	10.10	5.49	15.59
		average	**8.82**	**6.84**	**15.66**

Table 13. The experimental results of compressing 2 sequences in *S. cerevisiae* together.

Chr	Total no. of bases	Compressing separately		Compressed together		
		The no. of bases of non-repetitive sequence	bpb	The no. of bases of non-repetitive sequence	bpb	Time(s)
I	230208	205382	1.8838	716107	1.8020	7.5
VIII	562643	542351	1.9387			
	Total: 792851	Total: 747733	Average: 1.9236			
III	316617	305230	1.9430	1042785	1.8957	11.7
XIV	784333	761426	1.9402			
	Total: 1100950	Total: 1066656	Average: 1.9410			
IV	1531918	1449733	1.8964	2403013	1.8508	27.2
XII	1078175	993958	1.8557			
	Total: 2610093	Total: 2443691	Average: 1.8798			
V	576869	551827	1.9286	1973051	1.8786	13.1
IV	1531918	1449733	1.8964			
	Total: 2108787	Total: 2001560	Average: 1.9053			
VII	1090946	1047675	1.9244	1919695	1.8907	25.7
XV	1091289	913386	1.9378			
	Total: 2182235	Total: 1961061	Average: 1.9306			
VIII	562643	542351	1.9387	716107	1.8020	7.8
I	230208	205382	1.8838			
	Total: 792851	Total: 747733	Average: 1.9236			

Chr	Total no. of bases	Compressing separately		Compressed together		
		The no. of bases of non-repetitive sequence	bpb	The no. of bases of non-repetitive sequence	bpb	Time(s)
XI	666454	659274	1.9614	1527006	1.9214	11.5
XIII	924429	886815	1.9270			
	Total: 1590883	Total: 1546089	Average: 1.9417			
XII	1078175	993958	1.8557	2403013	1.8508	20.6
IV	1531918	1449733	1.8964			
	Total: 2610093	Total: 2443691	Average: 1.8798			
XIII	924429	886815	1.9270	1849416	1.8554	16.8
XII	1078175	993958	1.8557			
	Total: 2002604	Total: 1880773	Average: 1.8893			
XIV	784333	761426	1.9402	1777731	1.9022	14.1
XV	1091289	1053289	1.9204			
	Total: 1875622	Total: 1814715	Average: 1.9287			
XV	1091289	1053289	1.9378	1346647	1.8840	16.6
IV	1531918	305230	1.8964			
	Total: 2623207	Total: 1358519	Average: 1.9285			
XVI	948062	913386	1.9204	1919379	1.8907	15.6
VII	1090946	1047675	1.9244			
	Total: 2039008	Total: 1961061	Average: 1.9225			
average	**1860765**	**1664440**	**1.9162**	**1632937**	**1.8687**	**15.7**

For all the cases shown in Table 13, it is always beneficial to compress two sequences together. In the last row of Table 13, an average of 1.8687 bpb is achieved if two chromosome sequences are compressed together. It is smaller than that of 1.9162 bpb of separately compression case. The average number of bases of non-repetitive sequence in the separately compression case is about 1.66×10^6. However, it is about 1.63×10^6 when the two sequences are compressed together. Thus there is a reduction of about 32k bases. By comparing the average bpb of compressing two chromosome sequences separately in the Column 4 with that of compressing those sequences together, the performance of the compression together case is consistently better than that of compressing separately case. Moreover, the execution time is only 15.7s on average for the twelve chromosome sequences with an average of 800k bases. The execution time of more than 10s is for the case when the total lengths in the two most similar chromosome sequences have more than 1800k bases. Thus the reduction in bpb does not result in a significantly long execution time.

5.1.3 Multiple sequence compression

From Table 13, we can see the bpb of compressing two chromosome sequences together is consistently better than that of compressing them separately. The case of compressing a number of chromosome sequences

together is considered. Table 14 shows the experimental results. Column 1 indicates the number of chromosome sequences that are compressed together while Column 2 shows the respective chromosome sequence. Column 3 lists the total number of bases of the particular chromosome sequence. Column 4 and Column 5 indicate the number of non-repetitive bases and the bpb used respectively if each of the chromosome sequence is compressed separately. Column 6 and Column 7 show respectively the number of non-repetitive bases and the bpb if those sequences are compressed together. Column 8 and Column 9 list respectively the additional savings and the execution time in seconds. We can see that the bpb of compressing various chromosome sequences always is smaller than that of compressing them separately. On average, there is an additional 4.5% savings in bpb. The execution time is around 37 seconds. Thus, the savings can be achieved without significantly increasing the execution time.

The grouping strategy for choosing the chromosomes sequences that are to be compressed together are as follows. In the first step, we choose a similar chromosome sequences pair from Table 13. In the second step, we find another similar chromosomes sequences pair where one of them is the already selected chromosome sequence in the first step. For example, the pair consisting Chr IV and Chr XII is selected first in Table 13. To include more chromosome

sequences, we search chromosome sequences that are similar to either Chr IV or Chr XII. As a result, Chr V and Chr XV are founded. To include more chromosome sequences in a group, the second step can be repeated.

Table 14. The experimental results of compressing 3 to 6 chromosome sequences in *S. cerevisiae* together.

	Chr	Total no. of bases	Compressing separately		Compressed together			
			The no. of bases of non-repetitive sequence	bpb	The no. of bases of non-repetitive sequence	bpb	Additional Saving (%)	Time(s)
3	IV	1531918	1449733	1.8964	2926331	1.8378	4.40%	27.9
	V	576869	551827	1.9286				
	XII	1078175	993958	1.8557				
		Total: 3186962	Total: 2995518	Average: 1.8885				
3	IV	1531918	1449733	1.8964	3444430	1.8361	4.75%	32.4
	XII	1078175	993958	1.8557				
	XV	1091289	1053289	1.9378				
		Total: 3701382	Total: 3496980	Average: 1.8967				
4	IV	1531918	1449733	1.8964	3967748	1.8450	4.16%	38.3
	V	576869	551827	1.9286				
	XII	1078175	993958	1.8557				
	XV	1091289	1053289	1.9378				
		Total: 4278251	Total: 4048807	Average: 1.9010				
4	IV	1531918	1449733	1.8964	4168872	1.8556	3.59%	37.4
	XII	1078175	993958	1.8557				
	XIV	784333	761426	1.9402				
	XV	1091289	1053289	1.9378				
		Total: 4485715	Total: 4258406	Average: 1.9044				

5	III	316617	305230	1.9430	4450231	1.8415	4.72%	40.5
	IV	1531918	1449733	1.8964				
	XII	1078175	993958	1.8557				
	XIV	784333	761426	1.9402				
	XV	1091289	1053289	1.9378				
		Total: 4802332	Total: 4563636	Average: 1.9069				
5	III	316617	305230	1.9430	4020269	1.8464	5.34%	34.6
	IV	1531918	1449733	1.8964				
	V	576869	551827	1.9286				
	XIV	784333	761426	1.9402				
	XV	1091289	1053289	1.9378				
		Total: 4301026	Total: 4121505	Average: 1.9226				
6	III	316617	305230	1.9430	4973549	1.8398	4.79%	46.4
	IV	1531918	1449733	1.8964				
	V	576869	551827	1.9286				
	XII	1078175	993958	1.8557				
	XIV	784333	761426	1.9402				
	XV	1091289	1053289	1.9378				
		Total: 5379201	Total: 5115463	Average: 1.9092				
	average	**4304981**	**4085759**	**1.9113**	**3993061**	**1.8432**	**4.5%**	**36.8**

5.2 Simulation Results on *S. pombe*

In this experiment, we consider testing our proposed algorithm on another real dataset. The three chromosome sequences of *Schizosaccharomyces pombe* (*S. pombe*) are tested. *S. pombe* is a species of yeast. It is often used as a model organism in molecular and cell biology. These sequences can be downloaded from ftp://ftp.ncbi.nlm.nih.gov/genomes/. Note that the average length of *S. pombe* is 4200k, which is significantly longer than that of *S. cerevisiae*. Thus, the long length would increase the compression time considerably as it would take much longer time to search for the repetitive records in the sequences.

5.2.1 Single sequence compression

Table 15 lists the bits per bases (bpb) of the three chromosome sequences of *S. pombe*. Column 1 and Column 2 denote the chromosome sequence and its length respectively. Column 3, Column 4, Column 5 and Column 6 show respectively the bpb for the gzip, Arith, CTW, and our proposed algorithm. Similar to the case of *S. cerevisiae*, gzip uses an average of 2.3 bpb that expands the sequences. The Arith and CTW have stable performances which use 1.95 and 1.93 bpb respectively for all the three chromosome sequences. Our proposed algorithm uses an average of 1.90 bpb which is the lowest among the four algorithms. Note that the execution time of GenCompress is over 10 hours.

Since it is not practical to compress and decompress a file over 1 hour, the experiment for GenCompress is terminated manually.

To sum up, the proposed scheme performs comparably with other DNA oriented compression schemes. In single sequence compression case, only self-repetition is considered. From the results, we can see that if only long-term correlation in the current sequence is considered, the compression gain is not high. From 2 to 1.9 bpb, the compression gain is only about 5%. Therefore, an effective way for characterizing DNA sequences is much desirable to achieve further savings.

Table 15. The bpb (bits pre base) of compressing the 3 chromosome sequences in *S. pombe*.

cur	Length	gzip	Arith	CTW	proposed
I	5570797	2.31	1.95	1.93	1.92
II	4468099	2.32	1.95	1.93	1.91
III	2456786	2.28	1.95	1.93	1.89
	average	**2.30**	**1.95**	**1.93**	**1.90**

Table 16. The execution time (seconds) of compressing the 3 chromosome sequences in *S. pombe*.

cur	Length	gzip	Arith	CTW	proposed
I	5570797	1.61	2.36	115.99	53.16
II	4468099	1.30	1.87	94.51	44.13
III	2456786	0.70	1.10	52.23	27.95
	average	**1.20**	**1.78**	**87.57**	**41.75**

5.2.2 Two-sequence compression

Table 17 shows the experimental results of compressing a sequence by reference to its most similar sequence. Note that "cur" represents the chromosome sequence to be compressed. Its most similar sequence is represented as "ref". Column 3 shows the number of bases in the current sequence. Column 4 is the bpb of compressing the current sequence alone, i.e. without considering similar subsequences from the reference sequence. This is the same as the last column in Table 15. Column 5 gives the bpb of compressing a current sequence by considering similar subsequences from the current sequence as well as reference sequence. Column 6 and Column 7 list the additional saving in bpb and percentage (%) respectively.

Table 17. The experimental results of compressing a sequence in *S. pombe* provided with reference sequence.

cur	ref	Length	Without given ref sequence (bpb)	With given ref sequence (bpb)	Additional saving (bpb)	Additional saving (%)
I	II	5570797	1.9176	1.8857	0.0318	1.66%
II	I	4468099	1.9074	1.8678	0.0397	2.08%
III	II	2456786	1.8894	1.8069	0.0825	4.37%
average		**4165227**	**1.9048**	**1.8535**	**0.0513**	**2.69%**

From Table 17, we can see that the bpb is always smaller when the current sequence is compressed by considering one more sequence in addition to itself. The actual savings depend on the length of similar subsequences between the current sequence and the reference sequence. On average there is 0.05 bpb saving in using one more reference sequence. Although the additional saving in *S. pombe* is not as high as that in *S. cerevisiae*, the actual file size reduced in the chromosome sequences of *S. pombe* is larger than that of *S. cerevisiae* due to the long sequence length.

Figure 16 illustrates the distribution of self-similarity and cross-similarities for the three chromosome sequences of *S. pombe*. The dark grey color bar displays the proportion of similar repetitive subsequence found from itself while the light grey color bar shows the proportion of similar repetitive subsequence found from the reference sequence. We can see the repetition found from the reference sequence is always more than 50% of the whole repetitive part. Thus, the combination of self-similarity and cross-similarities should lower the bpb used.

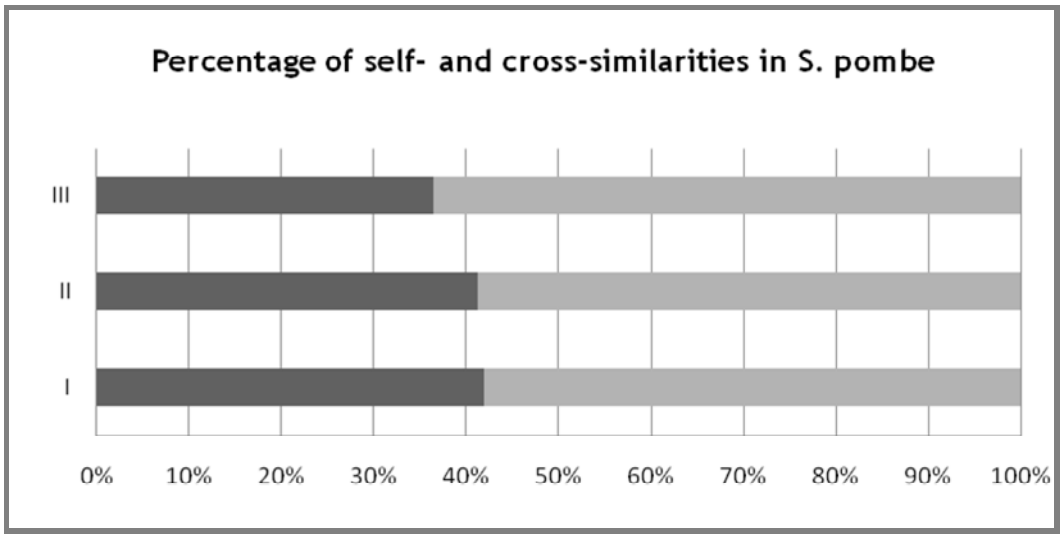


Figure 16. The percentage of self-sequence and corss-sequence similarities in *S. pombe*. The dark grey and the light grey color bars indicate the proportion of the self-similarity and the cross-similarities respectively.

As can be seen from Table 17 and Figure 16, the combination of self-repetition and cross-repetition should have positive effect on compression gain. Table 18 lists the results of compressing two chromosome sequences together. Column 1 lists the two chromosome sequences in a group which will be compressed together. Column 2 shows the length of the particular chromosome sequences and Column 3 shows the bpb of compressing the chromosome sequence in Column 1 alone. Thus, 1.9176 bpb and 1.9074 bpb are required for compressing Chr I and Chr II alone respectively. Column 4 shows the bpb when the two chromosome sequences are compressed together, so 1.8954 bpb is required if Chr I and Chr II are compressed together. The final column lists the execution time for compressing the two chromosome sequences in seconds. It takes about 92 seconds to compress Chr I and Chr II together.

Table 18. The experimental results of compressing 2 sequences in *S. pombe* together.

Chr	Total no. of bases	Compressing separately	Compressed together	
		bpb	bpb	Time(s)
I	2456786	1.9176	1.8954	92.0
II	4468099	1.9074		
	Total: 6924885	Average: 1.9110		
II	4468099	1.9074	1.8954	75.8
I	2456786	1.9176		
	Total: 6924885	Average: 1.9110		
III	5570797	1.8894	1.8718	156.0
II	4468099	1.9074		
	Total: 10038896	Average: 1.8974		
average	**7962889**	**1.9078**	**1.8875**	**107.9**

For all the cases in Table 18, it is always beneficial to compress two sequences together. An average of 1.8875 bpb is required to compress two chromosome sequences together. It is smaller than the average bpb in separated compression case. Although a reduction of 0.0203 bpb seems to be small, the average chromosome sequence length is 8000k bases in *S. pombe*. Thus, a small drop in bpb can significantly decrease the resultant file size. For example, with reduction of 0.01 bpb, 40,000 bits can be reduced for a sequence length with 4000k bases while only 8000 bits can be reduced for a sequence length with 800k bases. Moreover, the execution time is about 108s on average for these long chromosome sequences. The execution time of compressing Chr I and Chr II is around 92s because the total number of bases in these two sequences is 10,000k. Therefore, the execution time is in direct proportion to the number of bases.

5.2.3 Multiple sequence compression

The experimental results of compressing three chromosome sequences of *S. pombe* together are listed in Table 19. The average bpb in compressing separately is 1.9048 while that in compressing together is 1.8780. Thus an additional saving of 3.22% is achieved. The time taken for compression is 166.3 seconds as the total number of bases involved is 12,500k.

Table 19. The experimental results of compressing 3 sequences in *S. pombe* together.

	Chr	Total no. of bases	Compressing separately	Compressed together		
			bpb	Additional Saving (%)	bpb	Time(s)
3	I	2456786	1.9176	3.22%	1.8780	166.3
	II	4468099	1.9074			
	III	5570797	1.8894			
		Total: 12495682	Average: 1.9084			

5.3 Chapter Summary

The performance of our proposed multiple sequence compression algorithm has been evaluated using two real datasets. They are the sixteen chromosome sequences of *S. cerevisiae* and the three chromosome sequences of *S. pombe*. The results indicate that the multiple sequence compression strategy can always outperform the single sequence compression strategy as the bit per base used can be reduced by taking into account both self-sequence and cross-sequence similarities. The execution time does not increase significantly when

compressing a number of sequences together. From the experiment results, we can see the average additional saving in bpb is about 6% in *S. cerevisiae* if a chromosome sequence is compressed with reference to the most similar chromosome sequence. Also, the percentage of cross-sequence similarities is always more than 60% while that of the self-sequence similarity is less than 40%. The bpb of compressing three or more chromosome sequences is always lower than that of compressing each chromosome sequence separately. Therefore, we can see the benefit of multiple sequence compression.

6 Conclusions & Future Work

The compression gain of state-of-the-art DNA compression schemes is not large due to the fact that they search only similar subsequences within the current sequence. In other words, the cross-chromosomal similarities are completely ignored. The objective of this work was to exploit the uses of cross-sequence similarities in compressing a number of sequences together.

We have investigated similarities between the sixteen chromosome sequences in *S. cerevisiae* and the three chromosome sequences in *S. pombe*. Although cross-sequence similarities have been recognized and exploited in many applications, we have quantified it here for the first time with a view to an efficient multiple DNA sequence compression. A detailed similarity analysis including the length and location of similar subsequences between chromosome sequences has been performed. In particular, the percentage of cross-sequence similarities is always over 55% and that of self-sequence similarity always under 45% in both *S. cerevisiae* and *S. pombe*. While current DNA compression considers only repetitions found within the sequence itself, our study implies that it would be highly advantageous to compress different chromosome sequences together.

A multiple sequence compression algorithm has been proposed to take into account the self-sequence and the cross-sequence similarities. The proposed algorithm first searches for all the similar subsequences among a number of sequences. Then the overlapping regions in these similar subsequences are removed to form a list containing non-overlapping similar subsequences. Afterwards, the similar subsequences are removed to form a non-repetitive sequence for further compression by an arithmetic coder. Information such as the operations needed for matching two similar subsequences is also compressed by the arithmetic coder.

Our proposed multiple sequence compression algorithm was tested on two real datasets: *S. cerevisiae* and *S. pombe*. Our experimental results showed that the bpb of compressing two or more chromosome sequences together is lower than that of compressing each chromosome sequence separately. Therefore, it is always advantageous to compress a number of sequences together to benefit from both self-sequence similarity and cross-sequence similarities. Our proposed algorithm is also efficient as the execution time does not increase significantly, even for long sequences. Over the past twelve years of research into DNA compression, the improvement of the average bpb has been only about 0.06. However, the experimental results on average for our proposed algorithm

shows an improvement of 0.11bpb in *S. cerevisiae* when two chromosome sequences are compressed together.

Our future work is to extend this study to the DNA sequences of other species. For example, it is well known that monkeys and humans are closely related species. In fact, the monkey genome is more than 95% similar to the human genome [33-34], so similarity in DNA sequences among different species should be explored for an efficient multiple sequence compression. As shown in Table 4, the average bpb that can be saved for the five sequences in humans is 0.32. Since similarities exist between multi-species sequences, additional bpb can be saved with multiple sequence compression. To implement multiple sequence compression with different species, the first step is to conduct a similarity study between the sequences from different species, such as that of chimpanzees and humans. The lengths and locations of the repetitive subsequences can be explored. The second step would be to group similar sequences pairs and the final step would be to compress with those similar pairs together.

Although both *S. cerevisiae* and *S. pombe* are yeasts, their chromosome sequences have similarities and dissimilarities. In particular, both species share

genes with humans that they do not share with each other. For example, *S. pombe* contains the same heterochromatin genes as humans, while *S. cerevisiae* does not. Our future direction is to investigate how this information can be incorporated effectively into our multiple sequence compression algorithm.

References

[1] T. Matsumoto, K. Sadakane, H. Imai and T. Okazaki, “Can General-Purpose Compression Schemes Really Compress DNA Sequences?”, Currents in Computational Molecular Biology, pp.76-77, 2000.

[2] B.Behzadi and F. Le Fessant, “DNA Compression Challenge Revisited”, Symposium on Combinatorial Pattern Matching (CPM'2005), pp. 190–200, June 2005.

[3] X. Chen, M. Li, B. Ma and J. Tromp, “DNACompress: Fast and Effective DNA Sequence Compression”, Bioinformatics, vol. 18, no. 12, pp. 1696-1698, 2002.

[4] X. Chen, S. Kwong and M. Li, “A Compression Algorithm for DNA Sequences and Its Applications in Genome Comparison”, The 10th Workshop on Genome Informatics (GIW’99), pp 51-61, Tokyo, Japan, 1999.

[5] X. Chen, S. Kwong and M. Li, “A Compression Algorithm for DNA Sequences”, IEEE Engineering in Medicine and Biology Magazine, vol. 20(4), pp. 61-66, Jul/Aug 2001.

[6] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney and H. Zhang, “An Information-Based Sequences Distance and Its Application to Whole Mitochondrial Genome Phylogeny”, Bioinformatics, vol. 17(2), pp. 149-154, 2001.

[7] T. Matsumoto, K. Sadakane and H.Imai, “Biological Sequence Compression Algorithms”, Genome Informatics Workshop, Universal Academy Press, vol. 11, pp. 43-52, 2000.

[8] S. Grumbach and F. Tahi, “A New Challenge for Compression Algorithms: Genetic Sequences”, Journal of Information Processing and Management, vol. 30, pp.857-866, 1994.

[9] E. Rivls, J,P. Delahayem M. Dauchet and O. Delgrange, “A Guaranteed Compression Scheme for Repetitive DNA Sequences LIFL”, Université des Science et Technologies de Lille, Tech, Rep. IT-95-285, Nov 1995.

[10] E. Rivls, J,P. Delahayem M. Dauchet and O. Delgrange, “A Guaranteed Compression Scheme for Repetitive DNA Sequences”, in Proc. Data Compression Conf. (DCC-96), Snowbird, UT, pp. 453, 1996.

[11] Chang C. H., “DNAC: A Compression Algorithm for DNA Sequences by Non-overlapping Approximate Repeats”, Master Thesis, 2004.

[12] I. E. G. Richardson, “H.264 and MPEG-4 Video Compression – Video Coding for Next-generation Multimedia”, John Wiley & Sons, ISBN 0-470-84837-5, 2003.

[13] Y. Wang, J. Ostermann and Y. Q. Zhang, “Video Processing and Communications”, Prentice Hall, ISBN 0-13-017547-1, 2002.

[14] GenBank Overview: http://www.ncbi.nlm.nih.gov/Genbank/index.html

[15] The EMBL Nucleotide Sequence Database: http://www.ebi.ac.uk/embl/

[16] DDBJ Homepage: http://www.ddbj.nig.ac.jp/Welcome-e.html

[17] DNA Sequence Alignment: http://michael.dipperstein.com/dna

[18] "The Inquiry into BSE and Variant CJD in the United Kingdom", copyright held by Crown Copyright.

[19] D.A. Huffman, “A Method for the Construction of Minimum-redundancy Codes”, In Proc. IRE, vol.40, pp. 1098-1101, Sept 1952.

[20] A. Lempel and J. Ziv, “On the Complexity of Finite Sequences”, IEEE Trans. Inform. Theory, vol. 22(1), pp. 75-81, 1976.

[21] K. Sadakane, T. Okazaki and H. Imai, “Implementing the Context Tree Weighting Method for Text Compression”, Proc. Of IEEE Data Compression Conference, 2000.

[22] S. Grumbach and F. Tahi, “Compression of DNA Sequences”, In Data compression conference, pp. 340-350. IEEE Computer Society Press, 1993.

[23] J. Ziv and A. Lempel, “A Universal Algorithm for Sequential Data Compression”, IEEE Trans. Inform. Theory, vol. IT-23, pp.337-343, May 1977.

[24] 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), Snowbird, UT, pp.253-262, 2003.

[25] G. Korodi and I. Tabus, “An Efficient Normalized Maximum Likelihood Algorithm for DNA Sequence Compression”, ACM Trans. Inf. Syst., vol. 23, no. 1, pp. 3-34, Jan 2005.

[26] A. J. Pinho, A. J. R. Neves, C. A. C. Bastos and P. J. S. G. Ferreira, “A Three-State Model for DNA Protein-Coding Regions”, IEEE Tran. on Biomed. Eng. vol. 53, no. 11, Nov 2006.

[27] B. Ma, J. Tromp and M. Li, “PatternHunter - Faster and More Sensitive Homology Search”, Bioinformatics. vol. 18, pp. 440-445, 2002.

[28] BLAST Overview: http://www.ncbi.nlm.nih.gov/blast/blast_overview.shtml

[29] DB2 Universal Database: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.ii.doc/opt/c0007271.htm

[30] Sequence Similarity Search – BLAST: http://blast.genome.jp/

[31] Bioinformatics Solutions Inc.: http://www.bioinformaticssolutions.com/products/ph

[32] M. Ruvolo, "Molecular phylogeny of the hominoids: inferences from multiple independent DNA sequence data sets", Mol Biol Evol 14 (3), pp.248-65.

[33] J. W. Thomas et al., "Comparative analyses of multi-species sequences from targeted genomic regions." Nature 424, pp. 788-793, 2003.

[34] Tarjei S. Mikkelsen et al., "Initial sequence of the chimpanzee genome and comparison with the human genome." Nature, vol. 437, pp.69-87, 2005.

[35] Wentian Li, Gustavo Stolovitzky, Pedro Bernaola-Galvan and Jose L. Oliver, "Compositional Heterogeneity within, and Uniformity between, DNA Sequences of Yeast Chromosomes", *Genome Research* 8, pp. 916-928, 1998.

[36] FastAC: Arithmetic Coding Implementation
http://www.cipr.rpi.edu/~said/FastAC.html

[37] Amir Said, "Arithmetic Coding" in Lossless Compression Handbook, (K. Sayood, Ed.), Academic Press, San Diego, CA, 2003.

[38] Amir Said, "Introduction to Arithmetic Coding Theory and Practice", Hewlett-Packard Laboratories Report, HPL-2004-76, Palo Alto, CA, April 2004.

[39] Amir Said, "Comparative Analysis of Arithmetic Coding Computational Complexity", Hewlett-Packard Laboratories Report, HPL-2004-75, Palo Alto, CA, April 2004.

[40] The gzip home page: http://www.gzip.org/

[41] E. N. Trifonov and J. L. Sussman, "The pitch of chromatin DNA is reflected in its nucleotide sequence," Proc. Natl. Acad. Sci. USA, vol. 77, no. 7, pp. 3816 – 3820, July 1980.

[42] Paulo J. S. G. Ferreira, Ant´onio J. R. Neves, Vera Afreixo and Armando J. Pinho, "Exploring Three-Based Periodicity for DNA Compression and Modeling", IEEE ICASSP, vol. V, pp. 877-880, 2006.

[43] The GeNML homepage: http://www.cs.tut.fi/~tabus/genml/results.html

[44] Ida M. Pu., "Fundamental Data Compression", Butterworth-Heinemann, Chapter 6: Arithmetic coding, 2006.

[45] Mark N. and Jean-Loup G., "The Data Compression Book", 2nd Ed., M&T Books, Chapter 5: Huffman One Better: Arithmetic Coding, 1996.

[46] David S., "Data Compression", Springer, Section 2.14: Arithmetic Coding, 1997.

[47] James A. S., "Data Compression methods and theory", Computer science press, Section 2.9 Arithmetic Codes, 1988.

[48] Manzini G. and Rastero M., "A simple and fast DNA compressor. Software: Practice and Experience", vol. 34(14), pp. 1397-1411, 2004.

[49] Willems F. M. J., Shtrakov Y. M. and Tjalkens T. J., "The Context Tree Weighting Method: Basic Properties", IEEE Trans. Inform. Theory, IT-41(3), pp 653-664, 1995.

[50] Powell et al., D.R. Powell, D.L. Dowe, L. Allison and T.I. Dix, "Discovering simple DNA sequences by compression", In: Hawaii, Pacific Symposium on Biocomputing (1998b), pp. 597–608. 1998.

[51] Milosavljevic and Jurka, A. Milosavljevic and J. Jurka, “Discovering simple DNA sequences by the algorithmic significance method”, Comp. Appl. BioSci., vol. 94, pp. 407-411, 1993.

[52] Waterman, M.S., “Mathematical Methods for DNA Sequences”, CRC Press.

[53] Huffman, D.A., “A method for the construction of minimum-redundancy codes”, In: Proc. IRE volume 40, pp. 1098–1101.

[54] Arquès, D.G. and Michel, C.J., “Periodicities in coding and noncoding regions in genes”, Journal of Theoretical Biology 143, pp. 307–318.

[55] Oliver, S.G. et al., “The complete DNA sequence of yeast chromosome III”, Nature, vol. 357, pp. 38–46.

[56] R. Curnow and T. Kirkwood, “Statistical analysis of deoxyribonucleic acid sequence data-a review,” J. Royal Statistical Soc., vol. 152, pp. 199-220, 1989.

[57] E.J. Gardner, M.J. Sinnoms, and D.P. Snustad, Principles of Genetics, 8th ed. New York: Wiley, 1991.

[58] K. Lanctot, M. Li, and E.H. Yang, “Estimating DNA sequence entropy”, in Proc. SODA 2000.

[59] A. Lempel and J. Ziv, “Compression of individual sequences via variable-rate coding,” IEEE Trans. Inform. Theory, vol. 24, pp. 530-536, 1978.

[60] D. Loewenstern and P. N. Yianilos. Significantly lower entropy estimates for natural DNA sequences. Computational Biology, 6(1):125–142, 1999.

[61] A. Milosavjevic, “Discovery by minimal length encoding: A case study in molecular evolution,” Mach. Learn., vol. 12, pp. 68-87, 1993.

[62] I.H. Witten, R. Neal, and J.G. Cleary, “Arithmetic coding for data compression,” Commun. ACM, vol. 30, pp. 52-541, Jun. 1987.

[63] H.E. Williams and J. Zobel, “Compression of Nucleotide Databases for Fast Searching,” Computer Applications in the Biosciences, vol. 13, no. 5, pp. 549-554, 1997.

[64] L. Stern, L. Allison, R. L. Coppel, and T. I. Dix., “Discovering patterns in plasmodium falciparum genomic DNA”, Molecular & Biochemical Parasitology, vol. 118, pp. 175–186, 2001.

[65] D. R. Powell, L. Allison, and T. I. Dix., “Modelling alignment for non-random sequences”, Advances in Artificial Intelligence, pp. 203–214, 2004.

[66] A. Hategan and I. Tabus., “Protein is compressible”, NORSIG, pp. 192–195, 2004.

[67] J. G. Cleary and I. H. Witten., “Data compression using adaptive coding and partial string matching”, IEEE Trans. Comm., COM-32(4), pp. 396–402, April 1984.

[68] L. Allison, T. Edgoose, and T. I. Dix., “Compression of strings with approximate repeats”, ISMB, pp. 8–16, 1998.

[69] A. Apostolico and S. Lonardi. “Compression of biological sequences by greedy off-line textual substitution”, DCC, pp. 143–152, 2000.

[70] D. Adjeroh and F. Nan., “On compressibility of protein sequences”, DCC, pp. 422-434, 2006.

[71] Galvan P.B., Carpena P., Roldan R.R., and Oliver J.L., “Study of Statistical Correlations in DNA Sequences”, Gene, vol. 300, no. 1-2, pp. 105–115, 2002.

[72] Buldyrev S.V.,Goldberger A.L.,Havlin S., et al., “Long-Range Correlation Properties of Coding and Noncoding DNA Sequences: GenBank Analysis,” Phys.Rev.E, vol. 51, no.5, pp. 5084–5091, 1995.

[73] Chakravarthy N, Spanias A, Iasemidis LD, et al., “Autoregressive Modeling and Feature Analysis of DNA Sequences”, EURASIP Journal on Applied Signal Processing, Jan. 2003.

[74] Zhenqiang Tan, Xia Cao, Beng Chin Ooi, and Anthony K. H. Tung., “The ed-tree: An index for large dna sequence databases”, ssdbm, 00:151, 2003.

[75] Yong Zhang, Rahul Parthe, and Don Adjeroh. “Lossless compression of dna microarray images”, csbw, vol. 0, pp. 128-132, 2005.

[76] Toshio Modegi., “Development of lossless compression techniques for biology information and its application for bioinformatics database retrieval”, Genome Informatics, vol. 14, pp. 695-696, 2003.

[77] I. Sadel. “Universal data compression algorithm based on approximate string matching”, In Probability in the Engineering and Informational Sciences, pp. 465-486, 1996.

[78] Hisahiko Sata, Takashi Yoshioka, Akihiko Konagaya, and Tetsuro Toyoda., “Dna compression in the post genomic era”, Genome Informatics, vol. 12, pp. 512-514, 2001.

[79] E. Rivals and M. Dauchet., “Fast discerning repeats in DNA sequences with a compression algorithm”, In Proc. Genome Informatics Workshop, pages 215-226. Universal Academy Press, Tokyo, 1997.

[80] D. Kotlar and Y. Lavner, “Gene prediction by spectral rotation measure: A new method for identifying protein-coding regions,” Genome Res., vol. 13, pp. 1930-1937, 2003.

[81] V. R. Chechetkin and A. Y. Turygin, “Size-dependence of three-periodicity and long-range correlations in DNA sequences,” Phys. Lett., A, vol. 199, pp. 75–80, 1995.

[82] S. Tiwari, S. Ramachandran, A. Bhattacharya, S. Bhattacharya, and R. Ramaswamy, “Prediction of probable genes by Fourier analysis of genomic sequences,” Bioinformatics, vol. 13, pp. 263–270, 1997.

[83] B. Issac, H. Singh, H. Kaur, and G. P. S. Raghava, “Locating probable genes using Fourier transform approach,” Bioinformatics, vol. 18, no. 1, pp. 196–197, 2002.

[84] D. Kotlar and Y. Lavner, L. Rowen, G. Mahairas, and L. Hood, “Sequencing the human genome,” Science, vol. 278, pp. 605–607, Oct. 1997.

[85] M. Z. Ludwig, “Functional evolution of noncoding DNA,” Curr. Opin. Genetics Develop., vol. 12, no. 6, pp. 634–639, 2002.

[86] L. Allison, L. Stern, T. Edgoose, Dix TI, “Sequence complexity for biological sequence analysis”, Computers and Chemistry, vol. 24, pp. 43-55, 2000.

[87] G. Korodi and I. Tabus, “An Improved Pruning Condition for Tree Machines and Applications to Random-Access Coding,” in Proc. ISCCSP 2006.

[88] G. Korodi, J. Rissanen and I. Tabus, “Lossless data compression using optimal tree machines,” Proc. IEEE Data Compression Conference (DCC 2005), pp. 348–357, 2005.

[89] Kannan, SK and Myers, EW., “An algorithm for locating non-overlapping regions of maximal alignment score”, SIAM J Comput. vol.25, pp.648–662, 1996.

[90] Benson, G., “Tandem repeats finder: a program to analyze DNA sequences”, Nucleic Acids Res., vol. 27, pp. 573–580, 1999.

[91] Kurtz, S and Schleiermacher, C. “REPuter - fast computation of maximal repeats in complete genomes”, Bioinformatics, vol. 15, pp. 426–427, 1999.

[92] Gusfield, D., “Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology”, New York: Cambridge University Press, 1997.

[93] Volfovsky N, Haas B J and Salzberg S L., “A clustering method for repeat analysis in DNA sequences”, Genome biology, vol. 2(8), 2001.

[94] Lehman, E., “Approximation Algorithms for Grammar-based Data Compression”, PhD thesis, Massachusetts Institute of Technology, 2002.

[95] Lehman, E., and Shelat, A., “Approximation algorithms for grammar-based compression”, In Proceedings of the 13th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA- 02), ACM Press, pp. 205–212, Jan 2002.

[96] Cherniavski N. and Lander R., “Grammar-based Compression of DNA sequences”, 2004.

[97] T. I. Dix, D. R. Powell, L. Allison, S. Jaeger, J. Bernal, and L. Stern., “Exploring long DNA sequences by information content”, Probabilistic Modeling and Machine Learning in Structural and Systems Biology, Workshop Proc, pp. 97-102, 2006.

Appendix – Accepted and submitted papers

Published papers – Conference

1. Paula Wu, N. F. Law, and W. C. Siu, "Study of Inter-sequence Similarity for Multiple DNA Sequence Compression", International Symposium on Computational Models for Life Sciences, Australia, pp.167-176 Dec 2007.

Accepted/published papers – Journal

1. Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, "Cross chromosomal similarity for DNA sequence compression", Bioinformation 2(9): 412-416, 2008.
   http://www.bioinformation.net/002/008900022008.htm

2. Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, "Analysis of cross sequence similarities for multiple DNA sequences compression", International Journal of Computer Aided Engineering and Technology (accepted), 2009.