RESEARCH PAPER | Engineering | Volume : 3 | Issue : 3 | March 2013 | ISSN - 2249-555X

DNA Compression & Security Techniques based on Palindrome searching

KEYWORDS: Data Compression

Syed Mahamud Hossein
District Officer, Regional Office, Kolaghat, DVET, Govt. of West Bengal

P. K. Das Mohapatra
Assistant Professor, Deptt. of Microbiology, Vidyasagar University,W.B

ABSTRACT *A lossless compression algorithm, for genetic sequences, based on searching for exact palindromes is reported. The compression results obtained in the algorithm show that the exact palindromes are one of the main hidden regularities in DNA sequences. The proposed DNA sequence compression algorithm is based on genetic palindrome substring and creates online Library file acting as a Look Up Table. The genetic palindrome substring is replaced by corresponding ASCII character starting from 33(!). This substring length depends on user. Information security is the most challenging question to protect the data from unauthorized user. It can provide the data security, by using ASCII code and on line Library file acting as a signature. This algorithm is tested on benchmark DNA sequences, also on the reverse, the complement and the reverse complement benchmark DNA sequences, and on artificial DNA sequences. The algorithm can approach a compression rate of 3.851273 bit/base.*

1. Introduction

Biological sequence compression is a useful tool to recover information from biological sequences. I study one such basic question: the compressibility of DNA sequences. Life represents order. It is not chaotic or random [1]. Thus, we expect the DNA sequences that encode Life as nonrandom. Naturally they should be very compressible. There are also strong biological evidences in supporting this claim: It is well-known that DNA sequences, especially in higher eukaryotes, contain many genetic palindromes. It is also established that many essential genes (like rRNAs) have many copies. It is believed that there are only about a thousand basic protein folding patterns. Further it has been conjectured that genes duplicate themselves sometimes for evolutionary or simply for "selfish" purposes. These all concretly support that the DNA sequences should be reasonably compressible. It is well recognized that the compression of DNA sequences is a very difficult task [2,3,4,5]. The DNA sequences only consist of 4 nucleotide bases {a, t, g, c},8 bits are enough to store each base. While the regularities in DNA sequences are much subtler. It is our purpose to study such subtleties in DNA sequences. We will present a DNA compression algorithm, based on exact matching that gives the best compression results on standard benchmark DNA sequences. However, searching for all exact palindromes in a very long DNA sequence is not a trivial task. This algorithms take a long time (essentially a quadratic time search or even more) in order to find approximate palindromes that are optimal for compression. Simultaneously achieving high speed and best compression ratio remains to be a challenging task. Proposed DNA sequences Compression achieves a better compression ratio and runs significantly faster than any existing compression program for benchmark DNA sequences, simultaneously. Proposed algorithm consists of two phases: i) find all exact genetic palindromes; and ii) encode exact genetic palindrome regions and non-genetic palindrome regions. We have developed for fast and sensitive homology search[6], as our exact genetic palindrome search engine. Compression of DNA sequences is a very challenging task. We will present a DNA compression algorithm, based on genetic palindrome substring and corresponding genetic palindrome substring is place in Library file , this genetic palindrome substring create a dynamic Look Up Table and place ASCII character in appropriate places on source file and that gives the best compression results on standard benchmark DNA sequences. We will discuss details of the algorithm, provide experimental results and compare the results with the one most effective compression algorithm for DNA sequence (gzip-9).

Also we can find the compression rate, compression ratio of randomly generated of equivalent length of artificial DNA sequence. Compare all result to each other.

In this paper, if not otherwise mentioned, we will use lower case letters u, v, to denote finite strings over the alphabet {a, t, g, c},|u| denotes the length of u, the number of characters in u. u_i is the i-th character of u. $u_{i:j}$ is the substring of u from position i to position j. The first character of u is u_1. Thus u = $u_{1:|u|-1}$. and |v| denotes the length of v, the number of characters in v. v_i is the i-th character of v. $v_{i:j}$ is the another substring of v from position i to position j. $u_{i:j}$ match with $v_{i:j}$. The first character of v is v_1. Thus v = $v_{1:|v|-1}$. The minimum different between u-v is of substring length. The palindrome found if $u_{i:j=}$ $v_{i:j}$ and count exact maximum palindrome of $u_{i:j..}$ We use e to denote empty string and e=0.

2. Methods

2.1: All DNA sequence is stored on text File with extention .txt.

2.2 : Searching for exact palindromes

Consider a finite sequence s over the DNA alphabet {a,t,g,c}. An exact palindrome is a substring in s that can be transformed from another substring in s with edit operations (palindrome , insertion). We only encode those exact palindromes that provide profits on overall compression.

This methods of compression is as below

1. Run the program and output all exact palindromes into a list s in the order of descending scores;
2. Extract a palindrome r with highest score from list s, then replace all r by corresponding ASCII code into another list o and place r in library file.
3. Process each palindrome in s so that there's no overlap with the extracted palindrome r ;
4. Goto step 2 if the highest score of palindromes in s is still higher than a pre-defined threshold; otherwise exit.

2.3 Encoding palindromes

An exact palindrome can be presented as two kinds of triples. first is (l, m, p), where l means the palindrome substring length, m and p show the starting positions of two substrings in a palindrome, respectively, second Replace. This operation is expressed as (r; p; char) which means replacing the exact palindrome substring at position p by ASCII character char.

In order to recover an exact genetic palindrome correctly the following information must be encoded in the output data stream:

2.4 : Each substring match with all other substring for finding the exact maximum genetic palindrome substring.
Match condition occur if S[m]=S[p] p=l+l

Step-I
S[1] match with S[p] to S[n-l+1] and count S[1] {As per example S[1]=atg where substring size=3 and S[4]=gat,S[5]=gat….S[19]=ata So, S[1] substring genetic palindrome at 3 places Then m and p incremented by one

Step-2
Match S[2] match with S[p] to S[n-L+1] and count S[2] [As per example S[2]=gat and S[5]=tag ,S[6]=gat So, S[2] substring genetic palindrome at one places Then m and p incremented by one

Step-3
This method will continue to S[n-l+1] So S[n-2*l+1] match with S[p] to S[n-2*l+1] and count S[n-2*l+1] So, S[n-2*L+1] genetic palindrome only one place if mach occur.

Step-4 : Store all genetic palindrome count in descending order and find all exact maximum genetic palindrome count

Step-5 : Replace exact maximum palindromes substring by corresponding ASCII code and place genetic palindrome substring in library file, and create a on line look up Table.

Step- 6: Genetic palindrome Step-1 to step-5 excluding ASCII code

Step-7 : if the highest score of palindromes in s is still higher than a pre-defined threshold; otherwise exit.

2.5 : Decoding
Decoding time, first require on line Library file, which was created at the time of encoding the input file.

On this particular value , the encoded input string is decoded and produce the output original file.

At the time of decoding each ASCII character is replaced by corresponding base pair i,e O[M]=L[k] where O[M] is define by output sequence and L[k] is define by library file substring. If match occure in between L[33] to L[256] with O[M], place ASCII equivalent substring in ith places in output file. The value of m is incremented by one. If unmatch found in between L[33] to L[256] with O[M], place base pair in ith position in output file. The value of M is incremented by one. This process will continue until $M=n_1$ position will appear.

The Decoding process mentioned this rule and produce original output string.

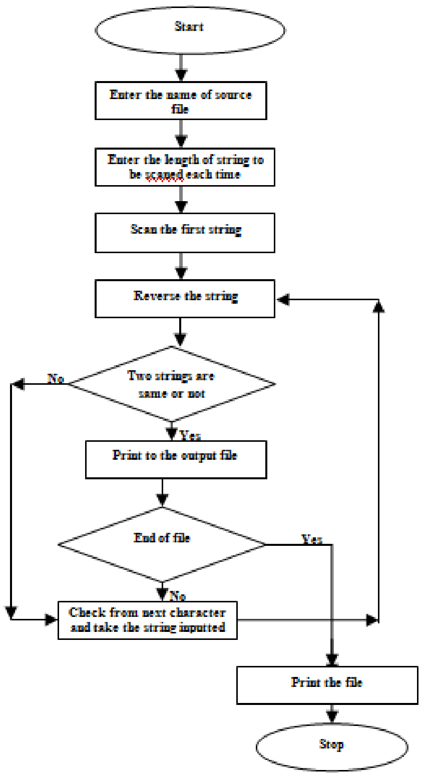
Match found if o[m]=L[33] to L[256] place ASCII character equivalent substring in i-th position. If match found, the value of m is incremented by one.

Otherwise o[m]≠L[33] to L[256] place base pair in i-th position in output file. If unmatch occure , the value of m is incremented by one.

For easy implementation, characters a,t,g,c will no longer appear in pre-coded file and A,T,G,C will appear in pre-coded file.

2.5 : ALGORITHM
1. Enter the name of the source file.
2. Enter the name of the destination file where the palindrome will be printed.
3. Enter the length of the string be taken input each time from the source file.
4. Take the first string of the specified length.
5. Reverse the string.
6. Check whether the source and reverse string are same or not. If same write it to output file specifying the position.
7. If palindrome found or not take the second string of specified length starting from second character of the source file.Continue steps 5, 6 & 7 till the end of the file.
8. If the file is ended stop.

Flowchart


Start
Enter the name of source file
Enter the length of string to be scaned each time
Scan the first string
Reverse the string
Two strings are same or not
No
Yes
Print to the output file
End of file
Yes
No
Check from next character and take the string inputted
Print the file
Stop

3. Algorithm evaluation
3.1: Accuracy
As to the DNA sequence storage, accuracy must be taken firstly in that even a single base mutation, insertion, deletion would result in huge change of phenotype as we see in the sicklemia. It is not tolerable that any mistake exists either in compression or in decompression. It is prove by generating one to one mapping of string.

3.2: Efficiency
I can see that the internal palindrome algorithm can compress original file from substring length (l) into 1 characters for any DNA segment, and destination file uses less ASCII character to represent successive DNA bases than source file.

3.4: Space Occupation
This algorithm reads characters from source file and writes them immediately into destination file. It costs very small memory space to store only a few characters. The space occupation is in constant level. In our experiments, the OS has no swap partition. All performance can be done in main memory which is only 512 MB on our PC.

4. Experimental Results
This algorithm tested palindrome techniques on standard benchmark data used in [7]. For testing purpose we use eight types of data.

These tests are performed on a computer whose CPU is Intel P-IV 3.0 GHz core 2 duo(1024FSB), Intel 946 original mother board, IGB DDR2 Hynix, 160GB SATA HDD Segate. Since the program to implement the technique have been written originally in the C++ language, (Windows XP platform, and TC compiler) .

The definition of the compression ratio [8]; 1− (|O|/2| I|), where |I| is number of bases in the input DNA sequence and

Volume : 3 | Issue : 3 | March 2013 | ISSN - 2249-555X

Table-I

Sequence Size	Sequence Name	Base pair/ File size	Cellular DNA Sequences								Artificial sequences							
			Normal Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences		Normal Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences	
			Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)
Sub string Size 3	tatsgs	9647	-0.92184	3.843682	-0.95418	3.908365	-0.92184	3.843682	-0.95418	3.908365	-0.95916	3.918317	-0.95501	3.910024	-0.95916	3.918317	-0.95501	3.910024
	atef1a23	6022	-0.93557	3.871139	-0.93823	3.876453	-0.93557	3.871139	-0.93823	3.876453	-0.94487	3.889738	-0.94753	3.895051	-0.94487	3.889738	-0.94753	3.895051
	atrdnaf	10014	-0.90533	3.810665	-0.90613	3.812263	-0.90533	3.810665	-0.90613	3.812263	-0.93329	3.866587	-0.96125	3.922508	-0.93329	3.866587	-0.96125	3.922508
	atrdnai	5287	-0.88311	3.766219	-0.86646	3.73293	-0.88311	3.766219	-0.86646	3.73293	-0.95423	3.908455	-0.95423	3.908455	-0.95423	3.908455	-0.95423	3.908455
	celk07e12	58949	-0.89852	3.797045	-0.89554	3.791074	-0.89852	3.797045	-0.89554	3.791074	-0.95009	3.900185	-0.95267	3.905342	-0.95009	3.900185	-0.95267	3.905342
	hsg6pdgen	52173	-0.92812	3.856247	-0.93456	3.869128	-0.92812	3.856247	-0.93456	3.869128	-0.96477	3.929542	-0.96216	3.924329	-0.96477	3.929542	-0.96216	3.924329
	mmzp3g	10833	-0.92043	3.840857	-0.916	3.831995	-0.92043	3.840857	-0.916	3.831995	-0.94923	3.898458	-2.49894	6.997877	-0.94923	3.898458	-0.94628	3.892551
	xlxfg512	19338	-0.88396	3.767918	-0.89306	3.786121	-0.88396	3.767918	-0.89306	3.786121	-0.95925	3.918502	-0.09029	2.180577	-0.95925	3.918502	-0.96008	3.920157
Sub string Size 4	atatsgs	9647	-0.7734	3.546802	-0.77589	3.551778	-0.7734	3.546802	-0.77589	3.551778	-0.8016	3.603193	-0.78833	3.576656	-0.8016	3.603193	-0.78833	3.576656
	atef1a23	6022	-0.78811	3.576221	-0.78412	3.56825	-0.78811	3.576221	-0.78412	3.56825	-0.80007	3.600133	-0.80804	3.616074	-0.80007	3.600133	-0.80804	3.616074
	atrdnaf	10014	-0.77671	3.553425	-0.78071	3.561414	-0.77671	3.553425	-0.78071	3.561414	-0.79748	3.594967	-0.81786	3.63571	-0.79748	3.594967	-0.81786	3.63571
	atrdnai	5287	-0.85739	3.714772	-0.78929	3.578589	-0.85739	3.714772	-0.78929	3.578589	-0.82788	3.655759	-0.82334	3.646681	-0.82788	3.655759	-0.82334	3.646681
	celk07e12	58949	-0.72563	3.451254	-0.73784	3.475682	-0.73621	3.472425	-0.71335	3.426691	-0.77957	3.559144	-0.77896	3.557923	-0.77957	3.559144	-0.77896	3.557923
	hsg6pdgen	52173	-0.72863	3.457267	-0.7363	3.472601	-0.72863	3.457267	-0.7363	3.472601	-0.76888	3.537769	-0.73676	3.473521	-0.76888	3.537769	-0.77379	3.547582
	mmzp3g	10833	-0.74319	3.486384	-0.7683	3.536601	-0.74319	3.486384	-0.7683	3.536601	-0.79415	3.588295	-0.79858	3.597157	-0.77569	3.551371	-0.79858	3.597157
	xlxfg512	19338	-0.70773	3.415451	-0.6889	3.377805	-0.70773	3.415451	-0.6889	3.377805	-0.95925	3.918502	-0.78426	3.568518	-0.78819	3.576378	-0.78426	3.568518

Volume : 3 | Issue : 3 | March 2013 | ISSN - 2249-555X

|O| is the length (number of bits) of the output sequence, The compression rate, which is defined as (|O|/| I|), where |I| is number of bases in the input DNA sequence and |O| is the length (number of bits) of the output sequence. The compression ratio and compression rate are presented in Table.

5. Result Discussion:

This algorithm is very useful in database storing. You can keep sequences as records in database instead of maintaining them as files. By just using the exact palindrome, users can obtain original sequences in a time that can't be felt. Additionally, our algorithm can be easily implemented.

From these experiments, we conclude that internal palindrome matching patter are same in all type of sources and Look up Table plays a key role in finding similarities or regularities in DNA sequences. Output file contain ASCII character with unmatch a,u,g and c so, it can provide information security which is very important for data protection over transmission point of view. This techniques provide the high security to protect nucleotide sequence in a particular source. Here we can get better security than static LUT.

6.Conclusion

In this article, we discuss a new DNA compression algorithm whose key idea is internal genetic palindrome. This compression algorithm gives a good model for compressing DNA sequences that reveals the true characteristics of DNA sequences. The compression results of genetic palindrome DNA sequences also indicate that our method is more effective than many others. This method is able to detect more regularities in DNA sequences, such as mutation and crossover, and achieve the best compression results by using this observation. This method is fails to achieve higher compression ratio than others standard method, but it has provide very high information security.

Important observation are :

a) Genetic palindrome substring length vary from 2 to 5 and no match found in case the substring length becoming six or more.

b) The substring length is three of highly genetic palindromeed than substring length of four and five. That is why substring length of three is highly compressible over substring length of four, five or more.

c) Normal sequence is highly compressible than reveres, complement and reverse complement sequences. Find the compression ratio, compression rate result in other orientation such as the reverse, the complement and the reverse complement the input sequences. But experimental result showing no meaningful changes are found using other orientation taking as a input sequences.

d) Cellular DNA sequences compression rate and compression ratio are distinguesable different due each sequence that come into different sources where as artificial DNA sequences compression rate and compression ratio are same in all time in all data sets.

7. Future work

We are developed to further research on as combination of palindrome, reverse ,repeat and genetic palindrome, this technique are also use to compression method. Also we try to reduce the time complexity.

8. Acknowledgement

Authors are grateful to all their our colleagues/friends for their interest and constructive criticism of this study. The authors offer special thanks to Dr. S.Basu, West Bengal University of Technology, Kolkata.

REFERENCE [1] M. Li and P. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, 2nd ed. New York: Springer-Verlag, 1997. | [2] Curnow, R. and Kirkwood, T., Statistical analysis of deoxyribonucleic acid sequence data { a review, J. | Royal Statistical Society, 152:199{220, 1989. | [3] Grumbach, S. and Tahi, F., A new challenge for compression algorithms: genetic sequences, J. Information | Processing and Management, 30(6):875-866, 1994. | [4] Lanctot, K., Li, M., and Yang, E.H., Estimating DNA sequence entropy, to appear in SODA '2000. | [5] Rivals, _E., Delahaye, J.-P., Dauchet, M., and Delgrange, O., A Guaranteed Compression Scheme for | Repetitive DNA Sequences, LIFL Lille I University, technical report IT-285, 1995. | [6] Ma,B., Tromp,J. and Li,M. (2002) PatternHunter—faster and more sensitive homology search. Bioinformatics, 18, 440–445. | 1698 | [7] S. Grumbach and F. Tahi, "A new challenge for compression algorithms: Genetic sequences," J. Inform. Process. Manage., vol. 30, no. 6, pp. 875-866, 1994. | [8] Xin Chen, San Kwong and Mine Li, "A Compression Algorithm for DNA Sequences Using Approximate Matching for Better Compression Ratio to Reveal the True Characteristics of DNA", IEEE Engineering in Medicine and Biology,pp 61-66,July/August 2001. | [9] ASCII code. [Online]. Available: http://www.asciitable.com | [10] National Center for Biotechnology Information,http://www.ncbi.nlm.nih.gov