UNUSUAL PATTERN DETECTION IN DNA DATABASE USING KMP ALGORITHM

S.RAJESH
LECTURER, DEPT.OF CSE
V.R.SIDDHARTHA ENGG.COLLEGE
VIJAYAWADA, AP, INDIA

S.PRATHIMA
LECTURER, DEPT.OF CSE
DJR COLLEGE OF ENGG&TECH
VIJAYAWADA, AP, INDIA

Dr.L.S.S.REDDY
DIRECTOR&PROF IN CSE
LBR COLLEGE OF ENGG
MYLAVARAM

ABSTRACT

Bioinformatics is the application of computer technology to the management and analysis of biological data. The result is that computers are being used to gather, store, analyze and merge biological data. The goal of bio-informatics is to uncover the wealth of biological information hidden in the mass of data and obtains a clearer insight into the fundamental biology of organisms. The most well known application of bioinformatics is sequence analysis. In sequence analysis, DNA sequences of various diseases are stored in databases for easy retrieval and comparison.

When we know a particular sequence is the cause for a disease, the trace of the sequence in the DNA and the number of occurrences of the sequence defines the intensity of the disease. As the DNA is a large database, I propose ***String and Pattern matching algorithms*** to find out a particular sequence in the given DNA. This paper entirely focuses on a novel approach for detecting the unusual patterns present in the gene database. Also, this paper emphasizes on how the disease can be transformed from parents to their children and efficient method for identifying the presence of the disease on hereditary basis and its impact.

Key Words

Bio informatics, pattern matching, sequence analysis, disease identification, KMP algorithm, DNA, failure function

1. Introduction

Bioinformatics is an interdisciplinary research area that is the interface between the biological and computational sciences. The Ultimate goal of bioinformatics is to uncover the wealth of biological information hidden in the mass of data and obtains a clearer insight into the fundamental biology of organisms. This new knowledge could have profound impacts on fields as varied as human health,

agriculture, the environment, energy and biotechnology. There are many other applications of bioinformatics, including predicting entire protein strands, learning how genes express themselves in various species, and building complex models of entire cells. As computing power increases and our databases of genetic and molecular information expand, the realm of bioinformatics is sure to grow and change drastically, allowing us to build models of incredible complexity and utility.

DNA is an acronym for the molecule deoxyribonucleic acid. Deoxyribonucleic acid (DNA) is called a double helix because it is a double-stranded molecule. DNA contains three components: deoxyribose (a five-carbon sugar), a series of phosphate groups, and four nitrogenous bases, (nitrogen compounds that contain bases). The four bases in DNA are adenine (A), thymine (T), guanine (G), and cytosine (C). The deoxyribose combines with phosphates to form the backbone of DNA. Thymine and adenine always come in pairs. Likewise, guanine and cytosine bases come together too.

Every human has his/her unique genes. Genes are made up of DNA; therefore the DNA sequence of each human is unique. However, surprisingly, the DNA sequences of all humans are 99.9% identical, which means there is only 0.1% difference. DNA is contained in each living cell of an organism, and it is the carrier of that organism's genetic code.The genetic code is a set of sequences, which define what proteins to build within the organism. Since organisms must replicate and/or reproduce tissue for continued life, there must be some means of encoding the unique genetic code for the proteins used in making that tissue. The genetic code is information, which will be needed for biological growth and reproductive inheritance.

A DNA sequence is a representation of the genetic code contained within an organism. Molecular biology researchers have great need to compare portions of DNA sequences. DNA sequence alignment is a representation of the similarity between two or more sections of genetic code. It is used to compare these sections in a quantitative way.Biologists use the comparisons to discover evolutionary divergence, the origins of disease, and ways to apply genetic codes from one organism into another. A DNA sequence is our representation of a string of nucleotides contained in a strand of DNA. For example: **ATGCGATACAAGTTGTGA**

Represents a string of the nucleotides A, G, C, and T.

The term **DNA sequencing** encompasses biochemical methods for determining the order of the nucleotide bases, **adenine, guanine, cytosine, and thymine**, in a DNA oligonucleotide. The sequence of DNA constitutes the heritable genetic information in nuclei, plasmids, mitochondria, and chloroplasts that forms the basis for the developmental programs of all living organisms. Determining the DNA sequence is therefore useful in basic research studying fundamental biological processes, as well as in applied fields such as diagnostic or forensic research. Because DNA is key to all living organisms, knowledge of the DNA sequence may be useful in almost any biological subject area. For example, in medicine it can be used to identify, diagnose and potentially develop treatments for genetic diseases. Similarly, genetic research into plant or animal pathogens may lead to treatments of various diseases caused by these pathogens.

Most biological databases consist of long strings of nucleotides (guanine, adenine, thymine, cytosine and Uralic) and/or amino acids (threonine, serine, glycine, etc.). Each sequence of nucleotides or amino acids represents a particular gene or protein (or section thereof), respectively. Sequences are represented in shorthand, using single letter designations. This decreases the space necessary to store information and increases processing speed for analysis. While most biological databases contain nucleotide and protein sequence information, there are also databases, which include taxonomic information such as the structural and biochemical characteristics of organisms. The power and ease of using sequence information has however, made it the method of choice in modern analysis.

2. Disease

An unhealthy symptoms or a specific illness in the body is termed as a disease. The term 'disease' refers to any abnormal condition of an organism that impairs function. In human beings, "disease" is often used more broadly to refer to any condition that causes discomfort, dysfunction, distress, social problems, and/or death to the person affected, or similar problems for those in contact with the person. In this broader sense, it sometimes includes injuries, disabilities, disorders, syndromes, infections, symptoms, deviant behaviors, and typical variations of structure and function, while in other contexts these may be considered distinguishable categories.

2.1 Disease in terms of DNA

The four nucleotides are grouped to form words and these words make sentences. These sentences are called as genes. Genes tell the cell to make other molecules called proteins. Proteins enable a cell to perform special functions.

The occurrence of unwanted or mutated words causes the disease. Every disease will have its own words or sequences and **the intensity of the disease depends on the occurrence of the mutated sequence in the DNA**. Especially cancer is caused due to the uncontrolled growth of cells is the result of alterations or mutations in the genetic material. More precisely, the emergence of cancer may require the accumulation of multiple mutations which allow cells to break out of the regulatory networks which ensure cooperation. This concept is referred to as multi-stage carcinogenesis. Once a cancerous cell has been created it can undergo a process known as clonal expansion. That is, it gives rise to descendants by cell division, and the population of cells grows to higher numbers. During this process, cells can acquire a variety of further mutations which leads to more advanced progression. A cancer is typically comprised of a variety of different genotypes

and represents a "mosaic" of cell lineages. The growth of a single, or primary, cancer does not usually lead to the death of the organism. Some cancer cells can, however, acquire the ability to enter the blood supply, travel to a different site, and start growing in a different organ. This process is referred to as metastasis. It is usually the metastatic growth which kills the organism.

2.2 Genetic Instability

Many cancer cells show a large variety of genetic alterations which range from small scale mutations to large chromosomal aberrations. While this is an intriguing observation, this does not prove that the cells are genetically unstable. The alterations could come about through a variety of factors, such as the exposure to extensive damage at some point in time, or specific selective conditions. Genetic instability is defined by an increased *rate* at which cells acquire genetic abnormalities [Lengauer *et* al. (1998)l. That is, cells have a defect in specific repair genes which results in higher variability. Indeed, studies have shown that many cancer cells are characterized by an increased rate at which genetic alterations are accumulated and are truly genetically unstable. Different types of genetic instabilities can be distinguished. They can be broadly divided into two categories. mall sequence instabilities involve subtle genetic changes which can dramatically speed up the process of cancer progression. Defects in mismatch repair mechanisms give rise to microsatellite instability or MSI. This involves copying errors in repeat sequences. MSI is most common in colon cancer. It is observed in about **13%** of sporadic cases and is the mechanism of cancer initiation in the hereditary non – polyposis colorectal cancer (HNPCC). Another type of small scale instability comes about through defects in nucleotide excision repair genes. These are responsible for the repair of DNA damage caused by exogenous mutagens, most importantly ultraviolet light. It is thus most important in the development of skin cancers. A defect in such repair mechanisms has been found in a disease called *xeroderma* pigmentosum, which is characterized by the development of many skin tumors in sun exposed areas. Instabilities which involve gross chromosomal alterations are called chromosomal instability or CIN. Cells which are characterized by CIN show a variety of chromosomal abnormalities. There can be alterations in chromosome numbers which involve losses and gains of whole chromosomes. This results in aneuploidy. Alternatively, parts of chromosomes may be lost, or we can observe chromosome translocations, gene amplifications, and mitotic recombinations. Many cancers show evidence of chromosomal instability. For example, 87% of sporadic colon cancers show CIN. The reason why CIN is observed in so many cancers is unclear. CIN can be advantageous because it helps to inactivate tumor suppressor genes where both functional copies have to be lost. Assume that one copy of a tumor suppressor gene becomes inactivated by a point mutation which occurs with a rate of per cell division. The second copy can then be lost much faster by a CIN event. For example, CIN could speed up the generation of an APC deficient cell in the colon. On the other hand, CIN is very destructive to the genome. Therefore, even though a cell with an inactivated tumor suppressor gene can be created with a faster rate, clonal expansion of this cell can be compromised because of elevated cell death as a consequence of chromosome loss.

2.3 Detection of Disease

When we know a particular sequence is the cause for a disease, the trace of the sequence in the DNA and the number of occurrences of the sequence defines the intensity of the disease. As the DNA is a large database we need to go for efficient algorithms to find out a particular sequence in the given DNA. We have to find the number of repetitions and the start index and end index of the sequence, which can be used for the diagnosis of the disease and also the intensity of the disease by counting the number of pattern matching strings, occurred in a gene database.

3. Hereditary Transformation

Since children inherit their genes from their parents, their can also inherit any genetic defects. Children and siblings of a patient generally have a 50% chance of also being affected with the same disease. Genetic testing can identify those family members who carry the familial unusual mutation and should undergo annual tumor screening from an early age. Conversely, genetic testing can also identify family members who do not carry the familial unusual mutation and do not need to undergo the increased tumor surveillance recom-mended for patients with unusual mutations.

The unusual pattern in the strand reflects in the split strand and hence increases in the unusual mutations increase in the cells. All familial cancer syndromes are caused by a defect in a gene that is important for preventing development of certain tumors (a so-called tumor suppressor gene). Everybody carries two copies of this gene in each cell, and tumor development only

occurs if both gene copies become defective in certain susceptible cells. In the general population, impairment of both gene copies in susceptible cells requires two independent events affecting the

same gene (ie, one target out of a million of such targets) in the same susceptible cell (ie, one cell out of trillions of cells) – a very unlikely event. Patients with unusual mutation, however, already carry a defect in one of theircorresponding gene copies in every cell of their body. In these patients, only one additional event affecting the intact gene copy in certain susceptible cells is needed to allow tumor development from these cells – a much more likely event. Therefore, patients with unusual mutation are at a greatly increased risk for developing the tumors associated with unusual mutation. Genetic testing can help to diagnose by detecting defects in the unusual mutated gene

4. Unusual Pattern Detection

Text documents are ubiquitous in modern computing, as they are used to communicate and publish information. From the perspective of algorithm design, such documents can be viewed as simple character strings. That is, they can be abstracted as a sequence of the characters that make up their content. Performing interesting searching and processing operations on such data, therefore, requires that we have efficient methods for dealing with character strings.

4.1 String Operations

At the heart of algorithms for processing text are methods for dealing with character strings. Character strings can come from a wide variety of sources, including scientific, linguistic, and Internet applications. Indeed, the following are examples of such strings:

***P*="CGTAAACTGCTTTAATCAAACGC"**

***R* = "u.s. Men Win Soccer World Cup!"**

S = http://www.wiley.com/college/ good rich/"...

The first string, *P*, comes from DNA applications, the last string, S, is the Internet address (URL) for the Web site that accompanies this book, and the middle string, *R*, is a fictional news headline. In this section, we present some of the useful operations that are supported by the string ADT for processing strings such as these.

Several of the typical string processing operations involves breaking large strings into smaller strings. In order to be able to speak about the pieces that result from such operations, we use the term *substring* of an m-character string *P* to refer to a string of the form *P[i] P [i*+ I] P [i+2] L .P[j], for some 0:::; *i.:::;* j:::; m-l, that is, the string formed by the characters in *P* from index *i* to index j, inclusive. Technically, this means that a string is actually a substring of itself (taking *i* = 0 and j = m - 1).

So if we want to rule this out as a possibility, we must restrict the definition to *proper* substrings, which require that either *i* > 0 or j < m - 1. To simplify the notation for referring to substrings, let us use *P [i....j]* to denote the substring of *P* from index *i* to index j, inclusive. That is,

P[i…j]=p[i] p[i+1]….p[j].

We use the convention that if *i* > j, then *P [i...j]* is equal to the *null string*, which has length 0. In addition, in order to distinguish some special kinds of substrings, let us refer to any substring of the form P [0...i], for 0 :::; i :::; m - 1, as a prefix of P, and any substring of the form P[i..m - 1], for 0: i: m - 1, as a suffix of P. For example, if we again take *P* to be the string of DNA given above, then "CGTAA" is a prefix of *P*, "CGC" is a suffix of *P*, and "TTAATC" is a (proper) substring of *P*. Note that the null string is a prefix and a suffix of any other string.

4.2 The Pattern Matching Problem

In the classic *pattern-matching* problem on strings, we are given a *text* string *T* of length *n* and a *pattern* string *P* of length *m*, and want to find whether *P* is a substring of *T*. The notion of a "match" is that there is a substring of *T* starting at some index *i* that matches *P*, character by character, so that

T[i]=P[0], T[i+I]=P[I], ..., T[i+m-I]=P[m-I].

That is,

P= T [i...i+m-l].

Thus, the output from a pattern-matching algorithm is either an indication that the pattern *P* does not exist in *T* or the starting index in *T* of a substring matching *P*.

To allow for fairly general notions of a character string, we typically do not restrict the characters in *T* and *P* to come explicitly from a well-known character set, like the ASCII or Unicode character sets. Instead, we typically use the general symbol ∑ to denote the character set, or *alphabet*, from which the characters of *T* and *P* can come. This alphabet ∑ can, of course, be a subset of the ASCII or Unicode character sets, but it could also be more general and is even allowed to be infinite. Nevertheless, since most document processing algorithms are used in applications where the underlying character set is finite, we usually assume that the size of the alphabet ∑, denoted with |∑|, is a fixed constant.

Example: *Suppose* we are *given the text string*

***T* =" abacaabaccabacabaabb "**

and *the pattern string*

***P* = "abacab".**

Then P is a *substring of T. Namely, P = T [10...15].*

There are various pattern-matching algorithms. These efficient algorithms can be used to trace the sequence of DNA in the gene database. They are

- Brute-Force
- Boyer-Moore
- Knuth-Morris –Pratt

Brute Force Pattern Matching

The *brute force* algorithmic design pattern is a powerful technique, when we have something that we wish to search for or to optimize some function. In applying we derive *brute-force pattern matching* algorithm, as probably the first algorithm for solving the pattern matching problem-we simply test all the possible placements of *P* relative to *T.* The brute-force pattern matching algorithm could not be simpler. It consists of two nested loops, with the outer loop indexing through all possible starting indices of the pattern in the text, and the inner loop indexing through each character of the pattern, comparing it to its potentially corresponding character in the text. Thus, the correctness of the brute- force pattern matching algorithm follows immediately.

The running time of brute-force pattern matching in the worst case is not good. However, because, for each candidate index in *T,* we can perform up to *m* character comparisons to discover that *P* does not match *T* at the current index. The running time of the brute force method is *O((n -m + l)m),* which is *O(nm).* Thus, in the worst case, when *n* and *m* are roughly equal, this algorithm has a quadratic running time.

The Boyer-Moore Algorithm

The *Boyer-Moore (BM)* pattern matching algorithm, sometimes avoid comparisons between *P* and a sizable fraction of the characters in *T unlike Brute-force in which* it is always necessary to examine every character in *T* in order to locate a pattern *P* as a substring . The only caveat is that, whereas the brute-force algorithm can work even with a potentially unbounded alphabet, the BM algorithm assumes the alphabet is of fixed, finite size. It works the fastest when the alphabet is moderately sized and the pattern is relatively long. The worst-case running time complexity of the BM algorithm is $O(nm + |\sum|)$. Namely, the computation of the last function takes time $O(m+|\sum|)$ and the actual search for the pattern takes $O(nm)$ time in the worst case, the same as the brute-force algorithm.

Pattern Matching Technique	Time Complexity
Brute-Force	**O ((n-m+1)*m)**
Boyer-Moore(BM)	**O (n*m+| ∑|)**
Knuth-Morris-Pratt(KMP)	**O (n+m)**

By comparing the time-complexities of the three patterns matching algorithms the Knuth Morris–Pratt Algorithm has been implemented in the first module to detect the pattern of DNA sequence in a gene database.

4.3 Pattern matching using KMP algorithm

In studying the worst-case performance of the *brute-force* and *boyer-moore* pattern matching algorithms on specific instances of the problem, we should notice a major inefficiency. Specifically, we may perform many comparisons while testing a potential placement of the pattern against the text, yet if we discover a pattern character that does not match in the text, then we throwaway all the information gained by these comparisons and start over again from scratch with the next incremental placement of the pattern. The Knuth-Morris-Pratt (or "KMP") algorithm avoids this waste of information and, in so doing, it achieves a running time of *O(n + m),* which is optimal in the worst case. That is, in the worst case any pattern matching algorithm will have to examine all the characters of the text and all the characters of the pattern at least once.

The Failure Function

The main idea of the KMP algorithm is to preprocess the pattern string *P* so as to compute *a failure function f* that indicates the proper shift of *P* so that, to the largest extent possible, we can reuse previously performed comparisons. Specifically, the failure function *l (j)* is defined as the length of the longest prefix of *P* that is a suffix of *P [1...j]* (note that we did *not* put *P[0...j]* here). We also use the convention that *1(0)* =0. The importance of this failure function is that it "encodes" repeated substrings inside the pattern itself.

Algorithm ***failure Function*** (***P***)

```
F[0] ← 0
i ← 1
j ← 0
while i < m
if P[i] = P[j]
        {we have matched j + 1 chars}
        F[i] ← j + 1
        i ← i + 1
        j ← j + 1
else if j > 0 then
        {use failure function to shift P}
        j ← F[j - 1]
else
        F[i] ← 0 { no match }
        i ← i + 1
```

```
i ← 0
j ← 0
while i < n
if T[i] = P[j]
        if j = m - 1
                return i - j { match }
        else
                i ← i + 1
                j ← j + 1
else
        if j > 0
                j ← F[j - 1]
        else
                i ← i + 1
```

Return "there is no substring of T matching P"

The KMP pattern matching algorithm incrementally processes the text string *T* comparing it to the pattern string *P.* Each time there is a match, we increment the current indices. On the other hand, if there is a mismatch and we have previously made progress in *P,* then we consult the failure function to determine the new index in *P* where we need to continue checking *P* against *T.* Otherwise (there was a mismatch and we are at the beginning of *P),* we simply increment the index for *T* (and keep the index variable for *P* at its beginning). We repeat this process until we find a match of *P* in *T* or the index for *T* reaches *n,* the length of *T* (indicating that we did not find the pattern *P in T).*

The main part of the KMP algorithm is the while-loop, which performs a comparison between a character in *T* and a character in *P for* each iteration. Depending upon the outcome of this comparison, the algorithm either moves on to the next characters in *T* and *P,* consults the failure function for a new candidate character in *P,* or starts over with the next index in *T.* The correctness of this algorithm follows from the definition of the failure function. The skipped comparisons are actually unnecessary, for the failure function guarantees that all the ignored comparisons are redundant-they would involve comparing characters that are already known to match.

Algorithm ***KMPMatch*****(*T, P*)**

```
F ← failureFunction(P)
```

5. Conclusion

The field of Bioinformatics has paved a path in diagnosing the disease by the identification of presence of unusual patterns in DNA database and the repetitive presence of unusual patterns in the gene, which represents the intensity of the disease in the effected person. Also, helps in detection of presence of the unusual pattern to the children of infected person, which hereditarily transferred. This project will be a fruitful application for the cancer research centers and hospitals in detecting the disease easily through the DNA database.

6. Reference:

- *Fast Pattern matching in strings, SIAM Journal of computer science, pp323 – 350, 1977,* Knuth D., Morris J. and Pratt V.
- *String matching with k differences by finite automata. In Proceedings of the International Congress on Pattern Recognition (ICPR'96). IEEE CS Press, Silver Spring, MD, 1996. 256–260,* Melichar.B.
- *A Minimum Cost Process in Searching for a Set of similar DNA Sequence, International conference*

on Telecommunications and Informatics, May 2006, pp348 – 353, Saman, Rahman, Ahmad, Osman.

- *Fast practical Exact and Approximate Pattern Matching in Protein Sequences,* C. S. Iliopoulos, Inuka Jayasekera1, and L. Mouchard.
- *Whole – Genome DNA Sequencing, IEEE Computer society, 1999, pp33 – 43,* Gene Myers.
- *A fast string-searching algorithm, Comm. Assoc. Comput. Mach., pp762 – 772, 1977,* R S Boyer & J S Moore.
- *Occurrences Algorithm for string searching based on Brute-force Algorithm,* Journal of Computer Science, 82 – 86, 2006.
- *Brute Force Algorithm,* Christian Charras.