An Algorithm for Detecting Approximate Tandem Repeats in Genomic Sequences

by
Nevzat Onur Domaniç
B. S., Middle East Technical University, 2004

Thesis

Submitted in partial fulfillment of the requirements for the
Degree of Master of Science in the Department of Computer Science
at Brown University

PROVIDENCE, RHODE ISLAND
MAY 2006

Vita

Name:	Nevzat Onur Domaniç
Date of Birth:	17 December 1982
Place of Birth:	Ankara, Turkey

Undergraduate Education

2000 - 2004	B.S. in Computer Engineering with High Honors Middle East Technical University Department of Computer Engineering Ankara, Turkey

Professional Experience

2005	Software Development Engineer Intern with Speech & Natural Language Group Microsoft Corporation Redmond, Washington

Contests and Awards

2000	Silver Medal 12th International Olympiad in Informatics Beijing, China
1999	Gold Medal 7th Balkan Olympiad in Informatics Ioannina, Greece

Preface and Acknowledgements

This thesis would not have been possible without the guidance, insight and support of my advisor Franco P. Preparata. I would like to express my deepest gratitude for his support and insight during this exciting endeavor. I would also like to thank my mother, Serpil, my father, Yüksel, and my brother, Arman, for their continuous love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Genomic sequences contain regions which are recurrences of some shorter sequences called *patterns*. These regions can be classified as either *interspersed* or *tandem* repeats according to the locations of the recurrences. Interspersed repeats consist of recurrences of some pattern which are not necessarily adjacent, whereas tandem repeats consist of consecutive recurrences of a pattern.

Another criterion for classifying the recurrences is the form in which the patterns are repeated. If the pattern is repeated without any variations, then the repeat is called a *perfect* or *exact repeat*. However this is not usually the case because of the mutations throughout the evolution history. In the case where the recurrences are slightly varied by the mutations (insertions, deletions and substitutions) the repeat is called an *approximate repeat*.

The functions and origins of approximate tandem repeats are not well understood even though they occur frequently in genomes, approximately 10% in mammalian genomes and and up to 50% in some arthropods [30]. However it's known that they play regulatory roles in genes and they may cause some diseases.

Tandem repeats may participate in protein binding [36] and alter the structure of chromatin [28]. They also play a role in the immunization system by affecting the recombination intensity in humans [33].

Several diseases including fragile-X mental retardation [44], myotonic dystrophy [18], Huntington's disease [20], Parkinson's disease in the Korean population [46], spinal and bulbar muscular atrophy [26] and Friedreich's ataxia [10] are known to be caused by some tandem repeat *polymorphisms* (variation within a population), usually abnormal increase in the number of copies. Some correlations between the structure of certain tandem repeats and some other genetic diseases (multiple sclerosis [19], Alzheimer's [31], Autism [11], and androgen insensitivity syndrome [17]) are being investigated as well.

Since the copy numbers of some specific tandem repeats often exhibit *polymorphism* due to replication slippage, unequal crossing-over and evolution history [8] [14] [41], tandem repeat polymorphism is useful in DNA fingerprinting [23] [22] [35], pedigree analysis, investigating the phylogenic relationships between species, evolution studies [4] and forensic DNA analysis. [21] [9].

The functional and structural roles of approximate tandem repeats which are mentioned above attracted researchers to developing powerful algorithms and tools to detect tandem repeats (briefly mentioned in Section 2.2.3), and maintaining the collected information about tandem repeats in databases. The Tandem Repeats Database (TRDB) [1] is a public repository of information on tandem repeats in genomes and contains a variety of tools for their analysis. The main tool is the Tandem Repeats Finder [6] which can query and filter for particular repeats of interest. The Minisatellite Database [2] and Short Tandem Repeat DNA Internet DataBase [3] are two databases that are focused on short repeats. TRbase, A Database Of Tandem Repeats In The Human Genome [4], is another database which is focused on human

[1]http://tandem.bu.edu/cgi-bin/trdb/trdb.exe

[2]http://minisatellites.u-psud.fr/

[3]http://www.cstl.nist.gov/biotech/strbase/

[4]http://trbase.ex.ac.uk/

genome. Some other tandem repeats databases include PlantSat [5], MICAS - Microsatellite Analysis Server [6], Repetitive Sequence DataBases (RSDB) [7] and MRD - A Microsatellite Repeats Database for genomes [8].

This thesis presents a new algorithm for detecting Approximate Tandem Repeats in genomic sequences without the need of any prior knowledge about the *pattern* (the repeated subsequence) or *period* (the length of the pattern). An implementation of the algorithm is compared with two state-of-the-art tandem repeats detection tools, Tandem Repeats Finder [6] and ATRHunter [45]. More formal definitions of tandem repeats and some background including the related work are presented in the next chapter. Chapter 3 describes this work in detail. Preliminary results of the comparisons are presented in Chapter 4.

[5]http://w3lamc.umbr.cas.cz/PlantSat/

[6]http://210.212.212.7/MIC/index.html

[7]http://binfo.ym.edu.tw/rsdb/

[8]http://www.ccmb.res.in/mrd/

Chapter 2

Background

A genomic sequence can be represented computationally as a string S of characters in the alphabet $\Sigma = \{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$. A substring of S denoted by $S_{i,j}$ or $S[i : j]$ is the sequence of the characters starting at position i and ending at position j where $a \leq i \leq j \leq n$ and n is the length of S. If $S_{i,i+l}$ is equal to a string Y where l is the length of Y, then we say that S contains Y at position i, or Y occurs in S at position i.

A Tandem repeat in a genomic sequence S is a substring $Y = S_{i,i+l}$ of S where l is the length of the repeat (length of the string Y) and Y is composed of multiple adjacent concatenations of some pattern repeated either perfectly or with slight variations. Firstly perfect tandem repeats will be defined and some related work will be mentioned.

2.1 Perfect Tandem Repeats

The notation $Y = X_1 X_2 \ldots X_n$ will be used throughout this thesis to denote that Y is the concatenation of the strings X_1, X_2, $\ldots$ and X_n. Similarly $Y = X^c$ is the notation for the concatenation of c copies of a string X.

Definition A *perfect* or *exact single tandem repeat* with period t is a string $Y = XX$ which is a concatenation of two copies of a string X of length t where $t \geq 1$. The string X is called the repeating pattern of Y.

Definition A *perfect* or *exact multiple tandem repeat* with period t and copy number $c+f$ is a string $Y = X^c X'$ which is a concatenation of c copies of a string X of length t and, if $f \neq 0$, a prefix X' of X where $t \geq 1$ and $c \geq 2$ and $f = \dfrac{length(X')}{t}$. The string X is called the repeating pattern of Y.

For instance the sequence

$$S = \textbf{ATCGT}\underline{\textbf{AGCGAGCG}}\textbf{TA}\underline{\textbf{TCCGCTCCGCTCCG}}\textbf{ATC}$$

contains a perfect single tandem repeat with period 4 at position 6 and a perfect multiple tandem repeat with period 5 and copy number 2.8 at position 16.

The problem of detecting perfect tandem repeats which consists of only two repeating units (perfect single tandem repeats) is also called as “detecting squares in strings” in computer science literature and several $O(n \log n)$ algorithms have been proposed (Apostolico and Preparata [3], Main and Lorentz [32], Crochemore [12], Stoye and Gusfield [43]). Apostolico [2] presents an optimal speed-up parallel algorithm. The algorithm by Stoye and Gusfield [43] also detects the perfect multiple repeats in $O(n \log n)$ time and $O(n)$ space using suffix trees. Crochemore [12] showed that the maximum number of occurrences of perfect single tandem repeats is $\Omega(n \log n)$ thus that the $O(n \log n)$ time bound is asymptotically optimal.

2.2 Approximate Tandem Repeats

Tandem repeats in genomes usually don’t consist of perfectly repeated patterns. Each repeating unit may be different from each other. Since there is more than one way to measure the difference between repeated units, several definitions of approximate tandem repeats exist.

2.2.1 Compression Based Definitions

There are some approximate tandem repeat detection algorithms which use data compression techniques (Milosavljevic and Jurka [34], Delgrange and Rivals [13]). They define a tandem repeat as a region which may be represented as a repeat of some pattern plus a set of mutations such that this representation can be coded using less space than coding the same region directly. Unfortunately these algorithms have some usage limitations: for instance the algorithm by Delgrange and Rivals [13] only finds tandem repeats consisting of repetitions of a given pattern.

2.2.2 Distance Based Definitions

An approximate tandem repeat is simply a concatenation of several repeating units which are slight variations of some pattern. Each of these repeating units is a substring of the actual repeat. In other words an approximate tandem repeat $Y = X_1X_2 \dots X_c$ is a concatenation of c strings X_1, X_2,... and X_c where each X_i is called a repeating unit and where the repeating units are similar according to some distance metric.

There are three main distance metrics used in tandem repeats detection algorithms: *Hamming distance*, *Levenshtein* or *Edit distance*, and *Alignment score*. Another item which causes the diversity when defining approximate tandem repeats besides the variety of distance metrics is the existence of different ways to choose the repeating units whose distances will be calculated. The most common ones are:

Pairwise repeats A string $Y = X_1X_2 \dots X_c$ is a *pairwise* approximate tandem repeat according to some distance criterion D if and only if every possible pair of repeating units passes the given distance criterion. In other words criterion

$D(X_i, X_j)$ must be *true* for all $1 \leq i \leq c$ and $1 \leq j \leq c$.

Consensus repeats A string $Y = X_1X_2 \ldots X_c$ is a *consensus* approximate tandem repeat according to some distance criterion D if and only if each repeating unit passes the given distance criterion with some consensus pattern C. In other words criterion $D(X_i, C)$ must be *true* for all $1 \leq i \leq c$ and for some C.

Neighboring repeats A string $Y = X_1X_2 \ldots X_c$ is a *neighboring* approximate tandem repeat according to some distance criterion D if and only if each repeating unit passes the given distance criterion with the adjacent repeating unit. In other words criterion $D(X_i, X_{i+1})$ must be *true* for all $1 \leq i < c$.

One or more of the above constraints can be combined with any of the distance metrics which will be described soon to define approximate tandem repeats.

Hamming Distance

Hamming Distance between two strings is only defined when the lengths of the two strings are identical. The Hamming distance is the number of mismatches when the two strings are aligned character by character.

Kolpakov and Kucherov [25]; and Landau *et al.* [27] proposed some algorithms to find tandem repeats where each unit differs by k-mismatches. The algorithm by Landau *et al.* has a time complexity of $O(nka \log(n/k))$ where a is the maximum copy number in any reported repeats. Unfortunately it's not possible to express other mutations like insertions and deletions using Hamming distance measure so that other kinds of similarity metrics, like edit distance or alignment scores, are needed.

Edit Distance and Alignment Score

Levenshtein or *edit distance* [29] between two strings is simply the minimum number of operations to transform the first string into the second one. The operations allowed are substitution, insertion or a deletion of a single character. The edit distance can be computed in $O(nm)$ time using dynamic programming where n and m are the lengths of the strings. If the number of insertions and deletions is bounded by d, then the computation can be done in $O(nd)$ time by dynamic programming.

Landau *et al.* [27] proposed an algorithm to detect approximate single repeats where each repeating unit differs by at most k edit operations (substitutions, insertions or deletions). The algorithm has a time complexity of $O(nk \log k \log(n/k))$.

Alignments are more general cases of edit distance because the weights of the operations can be specified with alignment scores.

Definition A *global alignment* of two strings S_1 and S_2 from alphabet Σ is a pair of strings (S_1', S_2') such that:

- S_1' and S_2' only contain characters from the alphabet $\Sigma' = \Sigma \cup \{\_\}$.
- $length(S_1') = length(S_2')$.
- removing all occurrences of the character $'\_'$ from S_1' yields S_1.
- removing all occurrences of the character $'\_'$ from S_2' yields S_2.

Then the alignment score of an alignment (S_1', S_2') is $F(S_1', S_2')$ where only additive score functions will be considered throughout this thesis, namely $F(S_1', S_2') = \sum_{i=1}^{l} f(S_1'[i], S_2'[i])$, where l is the common length of S_1 and S_2, and $S_1'[i]$ and $S_2'[i]$ denote the i^{th} character of the strings S_1' and S_2' respectively $(1 \leq i \leq l)$. The function $f(S_1'[i], S_2'[i])$ is called the *score function.*

When the alignment strings S_1' and S_2' are aligned character by character, namely when the two characters $(S_1'[i], S_2'[i])$ are compared:

- a pair (x, x) where $x \in \Sigma$ is called a match.
- a pair (x, y) where $x \neq y$ and $x, y \in \Sigma$ is called a mismatch or substitution.

- a pair $(x, '\_')$ where $x \in \Sigma$ is conventionally [1] called a deletion.
- a pair $('\_', y)$ where $y \in \Sigma$ is conventionally [1] called an insertion.

The number of errors $E(S'_1, S'_2)$ of an alignment (S'_1, S'_2) is the sum of substitutions, insertions and deletions in the alignment.

Definition An *optimal global alignment* of two strings S_1 and S_2 with respect to a score function f is the global alignment (S'_1, S'_2) of S_1 and S_2 such that the score $F(S'_1, S'_2) = \sum_{i=1}^{length(S'_1)} f(S'_1[i], S'_2[i])$ is maximal.

Definition A *local alignment* of two strings S_1 and S_2 from alphabet Σ is a pair of strings (S'_1, S'_2) such that:

- S'_1 and S'_2 only contain characters from the alphabet $\Sigma' = \Sigma \cup \{\_\}$.
- $length(S'_1) = length(S'_2)$.
- removing all occurrences of the character $'\_'$ from S'_1 yields a substring of S_1.
- removing all occurrences of the character $'\_'$ from S'_2 yields a substring of S_2.

In other words a local alignment of two strings is a global alignment of two respective substrings.

Definition An *optimal local alignment* of two strings S_1 and S_2 with respect to a score function f is the local alignment (S'_1, S'_2) of S_1 and S_2 such that the score $F(S'_1, S'_2) = \sum_{i=1}^{length(S'_1)} f(S'_1[i], S'_2[i])$ is maximal.

In this thesis and in all the tandem repeats detection algorithms which will be mentioned later it's assumed that the the score function $f(x, y)$ where $x, y \in \Sigma \cup \{'\_'\}$ obeys the following:

[1]During the replication of DNA, mutations like substitution, deletion or insertion of bases may take place.

- $f(x, x) = m_f$ for all $x \in \Sigma$ where $m_f \geq 0$. m_f is called the match score of the score function f.
- $f(x, y) = -s_f$ for all $x \neq y$ and $x, y \in \Sigma$ where $-s_f \leq 0$. s_f is called the mismatch penalty of the score function f.
- $f(x,' \_') = f('\_', x) = -i_f$ for all $x \in \Sigma$ where $-i_f \leq 0$. i_f is called the indel penalty of the score function f.
- $f('\_','\_') = -\infty$ so that the pair $('-','-')$ never appears in any optimal alignment.

Then any alignment score function can be identified by the triple (m_f, s_f, i_f), namely by its match score, mismatch penalty and indel penalty.

Another way to express the edit distance between two strings is to negate the score of the optimum global alignment between those two strings with the score function of the type $(0, 1, 1)$. Analogous to the computation of edit distance, the optimum global and local alignments between two strings X and Y can be computed using dynamic programming in $O(mn)$ time where m is the length of the first string and n is the length of the second. The computation is done by filling a $m \times n$ matrix such that each entry at i^{th} row and j^{th} column is the optimum global (or local) alignment score of the two strings $X_{1,i}$ and $Y_{1,j}$. The solution is then found by backtracing on the matrix in linear time. If only solutions which contain at most d insertions and deletions are in interest then the optimum global alignment can be computed in $O(nd)$ time. Only the diagonal band with width $2d$ of the matrix needs to be computed.

2.2.3 Related Work

The edit distance and/or alignment based algorithms may be classified into two groups. The algorithms in the first group try to compute the full alignment matrices exhaustively (Kannan and Myers [24], Benson [5], Schmidt [40]) to detect the tandem repeats. However even the most efficient one (by Schmidt [40]) is not applicable to sequences with over a thousand bases because of its $O(n^2 \operatorname{polylog}(n))$ time complexity.

Algorithms of the second group (Sagot and Myers [39], Benson [6], Stolovitzky *et al.* [42], Wexler *et al.* [45]) have two phases: the first phase filters some regions as candidate tandem repeat regions using statistical heuristics and the second phase verifies these candidates by computing alignments. The algorithm by Sagot and Myers [39] has a limitation of only accurately detecting the repeats with period size in the range between 30 and 40.

Benson's algorithm Tandem Repeats Finder [6] looks for short substrings (length between 3 and 7 depending on the period of the repeat) which are repeated in a neighborhood. Sufficiently high number of short substrings which are exactly replicated at some distances around d is an indication of an approximate tandem repeat with period d in that region. Then the actual alignment for verification of the region starts. This heuristic of identifying similarities of short windows (the *seeds*) and then extending the seeds for detecting the similarity of larger portions is called *filtration* and is used in many well-known homology search algorithms like BLAST [1].

The algorithm by Stolovitzky *et al.* [42] tries to incorporate the idea of using patterns with *don't care characters* (gaps) instead of using exact matches of short substrings. The pattern discovery algorithm TEIRESIAS [37] [38] is used to find a sufficiently high number of patterns occurring at positions shifted by distance d to detect candidate tandem repeat regions.

ATRHunter [45] by Wexler *et al.* uses the matches of longer seeds in its filtering based verification approach. However these matches are within some k-Hamming distance instead of exact matches or pattern matches with gaps (k is determined statistically).

2.2.4 Tandem Repeats Finder

Tandem Repeats Finder [6] is designed to overcome many of the limitations of the previous algorithms which make them not practically usable for detecting repeats with a wide range of period in long sequences. It looks for repetitions of short substrings to identify candidate regions and thus avoids the need for full scale alignment matrix computations. Also it does not require a priori knowledge of the pattern, pattern size or number of copies.

A probabilistic model of tandem repeats is assumed in the algorithm based on Bernoulli trials. The alignment of two tandem copies of a pattern of length n is modeled by a sequence of n-independent Bernoulli trials (coin tosses). The probability of success p_M, which is also called the *matching probability*, represents the average percent identity between two copies. Each head in the Bernoulli sequence is interpreted as a match between aligned nucleotides. Each tail is a mismatch, insertion or deletion. A second probability p_I, or *indel probability*, specifies the average percentage of insertions and deletions between adjacent copies. These parameters are considered as the *conservation parameters* and the pair (p_M, p_I) is a quantitative description of the most divergent copies that the algorithm detects.

The algorithm consists of two phases, *detection* and *analysis*. The detection phase uses a set of statistically based criteria to find *candidate* tandem repeats. These statistical criteria are derived according to the parameters mentioned above. The analysis phase attempts to produce an alignment for each candidate and in case of success it reports the repeat with some information like the percentage of identity, percentage of indels and alignment of copies with consensus pattern.

Detection Phase

The algorithm assumes that adjacent copies of any pattern should contain some matching characters in corresponding positions but the number of matches and how the distances between those matches vary is unknown. Let's first describe how these corresponding matches are detected.

The algorithm looks for matching nucleotides separated by a common distance d, which is not specified in advance. For reasons of efficiency it looks for runs of *k-tuple matches*. A k-tuple is described as a window of k consecutive characters from the nucleotide sequence and matching k-tuples are two windows with identical contents.

A list (called *the history list*) for all possible 4^k k-tuples (they are called probes) are constructed and these lists are processed by sliding a k-length window across the entire sequence. For each probe p, the *history list* H_p maintains the positions where p occurs.

When a position i is added to H_p, the algorithm scans for all earlier occurrences of p, and for each earlier occurrence j, the distance $d = i - j$ is considered as a *possible pattern size* for a tandem repeat. Before reporting this occurrence as a candidate tandem repeat, more evidence of occurrences of more k-tuples with the same (or closer) distance d starting at position between j and i is needed. A list called the *distance list* D_d stores this information.

The list D_d is updated every time a match at distance d is detected. The position i of the match is stored on the list. The right end of the window is set to i and matches that occurred before $j = i - d$ are dropped from the list. The lists for other nearby distances to d are also updated as follows: their right ends are set to i and the matches which occurred before $j = i - d$ are removed. The information stored in these lists is used to test the distance d and position i according to the statistical criteria and if the tests are successful this occurrence is reported as a candidate tandem repeat and the analysis phase takes place.

Statistical Criteria

Four probability distributions are defined depending on the pattern length d, the matching probability p_M, the indel probability p_I, and the tuple size k. These distributions are either calculated by some formulae or computed by simulation and some cut-off value is determined for each distribution. They are:

Sum of heads distribution This distribution indicates how many matches are required. Let's define the random variable R_{d,k,p_M} as the total number of heads in head runs of length k or longer in an independent and identically distributed Bernoulli sequence of length d with success probability p_M. This distribution is well approximated by the normal distribution and Benson and Su [7] showed that its exact mean and variance can be calculated in constant time. The largest number x such that 95% of the time $R_{d,k,p_M} \geq x$ is determined and this number is used as the *sum of heads criterion* for the test. For example if $p_M = 0.75$, $k = 5$ and $d = 100$, then the criterion is 26. In other words for a pattern with length 100 where aligned copies are expected to match with probability at least 75%, we expect to count at least 26 5-tuple matches 95% of the time.

Random walk distribution This distribution is used to analyze how distances between matches vary due to insertions and deletions. Remember that the algorithm looks at the distance lists $D_{d\pm1}$, $D_{d\pm2}$, ...,$D_{d\pm\Delta d_{max}}$ as well as D_d. To determine this Δd_{max} it's assumed that the insertions and deletions are equally likely. Let's define the random variable W_{d,p_I} as the maximum displacement from the origin of a one-dimensional random walk with expected number of steps equal to $p_I \cdot d$. Feller [15] showed that 95% of the time, W_{d,p_I} ranges within $\pm 2.3\sqrt{p_I \cdot d}$. Therefore Δd_{max} is set to $\lfloor 2.3\sqrt{p_I \cdot d} \rfloor$. For instance if $p_I = 0.1$ and $d = 100$, then $\Delta d_{max} = 7$. An analogous criterion is used in the algorithm in this thesis.

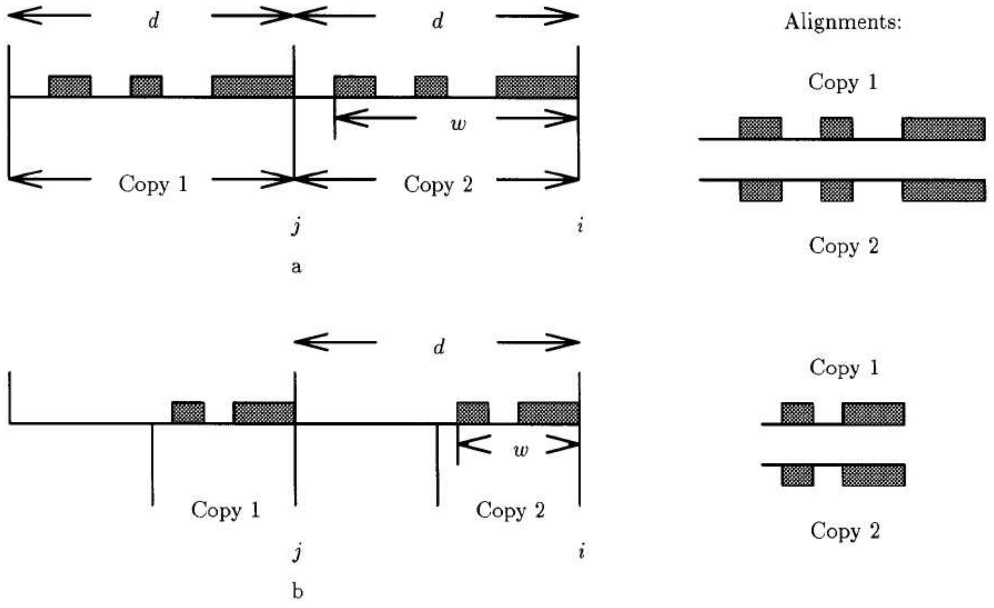


Figure 2.1: Apparent size distribution criterion is used in Tandem Repeats Finder to distinguish between (a) tandem repeats (matching k-tuples are distributed throughout the interval from j to i) and (b) non-tandem repeats (matching k-tuples are concentrated on the right side of the interval).

Apparent size distribution This distribution and criterion is used to distinguish between tandem repeats from interspersed repeats. The matching k-tuples are distributed throughout the interval from j to i for tandem repeats, whereas they should be concentrated on the right side of the interval for non-tandem repeats (illustrated in Figure 2.1). The criterion is determined by simulation. If the distance between the first and the last tuple on list D_d is smaller than the criterion than the repeat is considered as non-tandem repeat.

Waiting time distribution This distribution is used to determine the tuple size k. As the tuple size increases the running time of the algorithm decreases because the probability of a long tuple to be appearing is lower than a shorter one, thus the history lists become shorter. On the other hand the probability of missing some approximate tandem repeats increases as k increases because approximate repeats may not contain so long exact matches. The tuple sizes for some period

Tuple	Pattern Sizes	
Sizes	p_M **= .75**	p_M **= .8**
3	1 – 29	–
4	30 – 43	1 – 29
5	44 – 159	30 – 159
7	160 – 500	160 – 500

Table 2.1: Calculated tuple sizes in Tandem Repeats Finder for some range of periods.

ranges which are determined according to this criterion are shown in Table 2.1.

Analysis Component

The analysis component of the algorithm verifies whether the candidate tandem repeats (those passing the statistical criteria) are actual tandem repeats. A candidate pattern consisting of positions $j+1, j+2, \ldots, i$ is selected and it's aligned with the surrounding sequence using a specialized version of wraparound dynamic programming [16]. If at least two copies of the candidate pattern is aligned then it's considered as a tandem repeat. But this candidate pattern is usually not the best *consensus pattern*. A new consensus pattern is determined using the majority rule from the alignments of the original candidate pattern and then this new consensus pattern is realigned to find the final alignment. The tandem repeats in Tandem Repeats Finder are defined as a sequence which has an optimal alignment score larger than a given threshold when aligned with a periodic repetition of a consensus pattern.

The most time consuming process in the algorithm is this wraparound dynamic programming alignment. To decrease the running time the diagonal band is narrowed to a radius of Δd_{max} in the alignment matrix for patterns larger than 20 characters.

Also the same repeat may be reported several times with different pattern sizes. For instance a repeat with pattern size 25 can be reported multiple times with pattern sizes 25, 50, 75 and so on (smallest period size isn't always the best alignment).

Complexity Analysis

In the detection phase, for each occurrence of a k-tuple, up to $2 \cdot \Delta d_{max}$ distance lists are updated. Assuming that it takes constant time to update a distance list, the time required for each occurrence of a k-tuple is $O(\Delta d_{max})$. Let t_{max} be the maximum period length that is of interest. Then only the occurrences of k-tuples up to t_{max} positions are needed to be observed. Since each k-tuple is expected to occur $t_{max}\frac{1}{4^k}$ times in t_{max} positions, the overall complexity of the detection phase is $O(n \cdot \Delta d_{max} \cdot t_{max} \cdot 4^{-k})$ where n is the sequence length. Since Δd_{max} is $O(\sqrt{t_{max}})$, the overall complexity is $O(n \cdot (t_{max})^{1.5} \cdot 4^{-k})$ where t_{max} is the maximum period length and k is the minimum tuple size used.

The analysis phase takes $O(l \cdot \Delta d_{max})$ time for a repeat with length l. Under the assumption that the ratio of failed alignments (false alarms) to the successful alignments (reported repeats) is constant, the overall detection phase takes $O(L \cdot \Delta d_{max}) = O(L \cdot \sqrt{t_{max}})$ time where L is the total length of the reported repeats and t_{max} is the maximum period.

2.2.5 ATRHunter

ATRHunter [45] consists of two phases analogously to the Tandem Repeats Finder, a screening phase which generates a list of candidate tandem repeat regions based on a statistical model, and a verification phase which verifies the candidate tandem repeat regions.

Screening Phase

The regions which have a high probability of being a tandem repeat are detected in this phase based on some similarity criteria. For a substring of length t to qualify as a pattern of a tandem repeat, there should be some similarity with the subsequent (adjacent) substring of length t. To test the similarity between two consecutive substrings of length t, segments of length l of these two substrings are compared where $l < t$. However instead of exact matches, approximate matches are considered under the Hamming distance. Every segment of length l (called an l-window) of the first substring is compared with an appropriate l-window of the second substring. The outcome of a comparison of two l-windows is called a q-quality vector ($0 \leq q \leq 1$) if there are at least $q \cdot l$ matching characters when these two l-windows are compared character by character (if the Hamming distance of these two windows is at most $(1 - q)l$). Given l and q, let's define two quantities: score and gap, for every position i in the sequence. The score $S_t(i)$ is the number of q-quality vectors produced by the comparison of the l-windows in the substring of length t starting at position i. Since there are $t-l+1$ l-windows for a substring of length t, the maximum value that $S_t(i)$ can take is $t - l + 1$. The gap $\Delta_t(i)$ is the maximal number of consecutive l-windows in the substrings of length t starting at position i that produce vectors which are not q-quality.

There are three similarity criteria in the screening phase which qualifies a region as a candidate region. These are:

Score criterion $S_t(i)$ should be greater than or equal to the threshold σ_t

Continuity criterion $\Delta_t(i)$ should be less than or equal to the threshold δ_t

Distance criterion For every comparison which resulted in a q-quality vector, the position difference of the two l-windows that are compared should be in the range $[t - d^t_{max}, t + d^t_{max}]$.

The thresholds σ_t, δ_t and d^t_{max} in these criteria depend on the pattern length t and the distribution of gap and score values. The process of determining the threshold d^t_{max} for distance criterion is identical to the Random Walk Distribution of Tandem

Repeats Finder. The other thresholds are determined based on random walks on some specially defined graphs [45] which will not be explained here.

The screening phase of the algorithm consists of a loop of t_{max} iterations where candidate tandem repeats with pattern length t are detected in each iteration t. For each t, the parameters l and q are determined and two l-windows are placed at positions 1 and $t+1$ initially. These two l-windows are slided towards the end of the sequence and they are compared at each step to produce q-quality vectors. The first l-window is slided 1 position at each step whereas the the second l-window is slided by $0, 1$ or 2 positions greedily to maximize the number of q-quality vectors produced. The default choice is to advance the second l-window by 1 position if it produces a q-quality vector. If it doesn't produce a q-quality vector then the other choices are tried as soon as the distance criterion described above is met and a q-quality vector is produced. If none of the tree choices produces a q-quality vector, then the default choice, which is advancing it by 1 position, is selected. At the end of the each i^{th} step where $i > t-1$ the algorithm counts the number of q-quality vectors and the maximum number of consecutive non q-quality vectors within the last $t-l+1$ vectors and sets these quantities as $S_t(i-t+l)$ and $\Delta_t(i-t+l)$ respectively. These computations can be done in $O(1)$ time by maintaining a doubly linked list. The position i's that pass the above three criteria are reported as candidate tandem repeats and the verification phase takes place for these regions. When both windows are slided by one position, it takes $O(1)$ time to check whether they produce a q-quality vector or not, because only the first and last character pairs are changed in the alignment of the windows. However if the second window is slided by 0 or 2 positions then the alignment of windows completely change and it takes $O(l)$ time to check whether they produce a q-quality vector. The parameters l and q are determined so that the probability of sliding the second window 0 or 2 positions is $1/l$. Then the amortized time for comparing two l-windows become $O(1)$ for each position. Since all the pairs at distance up to t_{max} starting at every position of the sequence are compared, the screening phase takes $O(n \cdot t_{max})$ time, where n is the length of the input sequence.

Verification Phase

Candidates are verified using dynamic programming alignment. For the alignments of pattern length greater than 20, only the diagonal with radius d^t_{max} is computed in the alignment matrix to save some computation time analogously to Tandem Repeats Finder.

Instead of using wraparound dynamic programming, ATRHunter combines single repeats to produce multiple repeats. The second repeating unit of a single repeat is aligned with the first unit of some following alignment, and they are combined if the alignments are similar. Thus the computation takes $O(l \cdot d^t_{max}) = O(l \cdot \sqrt{t_{max}})$. Again under the assumption that the ratio of number of false alarms to the number of detected repeats is constant, the verification phase takes $O(L \cdot \sqrt{t_{max}})$ time, where L is the total length of the reported repeats.

Chapter 3

Algorithm

This chapter will discuss the algorithm that is developed to detect the tandem repeats. The algorithm can be conceived as two phases even if these phases are not sequentially executed during the run of the algorithm. Before explaining the algorithm, the definition of Approximate Tandem Repeats in this work and the goal of the algorithm will be explained Section 3.1. Then the first phase which detects the candidate tandem repeat regions is presented in the Section 3.2. Section 3.3 discusses the second phase of the algorithm which verifies the candidate repeat regions. Finally various criteria that are used in the algorithm are explained in Section 3.4.

3.1 Definition of Approximate Tandem Repeats

3.1.1 Model of Formation of Tandem Repeats

The probabilistic assumption in the stochastic process of the formation of approximate tandem repeats in this work is very similar to the assumption of Tandem Repeats Finder [6]. It's assumed that there's a substitution probability of p_S and an insertion or deletion probability of p_I (insertions and deletion are assumed equally likely) at each position when a unit is repeated; and it's also assumed that these errors are independent from each other. Since the total error probability is $p_S + p_I$, the probability of a character being copied without errors is called p_M or probability of match where $p_M = 1 - (p_S + p_I)$. Therefore roughly $p_M \times t$ matches, $p_S \times t$ substitutions and $p_I \times t$ indels are expected when aligning two units of a repeat with period t.

3.1.2 Terminology

Since insertions and deletions are allowed in the approximate repeats, the best way to define an approximate repeat is with a distance based approach (explained in Chapter 2.2.2) with an edit distance or alignment score metric. First approximate single tandem repeats will be defined.

Definition A string $Y = XX'$ is an *approximate single tandem repeat* if and only if the number of errors (substitutions plus indels) in the optimum global alignment of its two repeating units X and X' with respect to a score function f is less than or equal to some similarity criteria $\theta_{max}(t)$ where t (called the *period*) is the length of the first repeating unit X.

The alignment score function f is an input of the algorithm and is universal for all tandem repeats. The threshold $\theta_{max}(t)$ is a function of the parameter p_M of the probabilistic assumption described above and the period of the repeat t. It serves as a threshold of similarity between the repeating units. The calculation of this threshold is explained in section 3.4.

Since the algorithm presented in this thesis is capable of detecting both single and multiple repeats, a new definition of tandem repeats will be introduced later to express both single and multiple repeats. However because of the similarity between the processes of detecting single and multiple repeats, all the examples and explanations in the algorithm will be about single repeats in order to be clearer. The only difference between the detection of single and multiple repeats is in the last step of the verification phase and necessary remarks will be made there (Section 3.3.2).

To extend the definition of single tandem repeats to multiple repeats, a combination of the consensus and the neighboring approximate tandem repeat definitions according to an alignment score criteria is used because of its similarity to the definitions of tandem repeats in Tandem Repeats Finder [6] and ATRHunter [45] which allows a comparison of this work with them. Here is a formal definition of an approximate tandem repeat [1] [2]:

Definition A string $Y = X_1X_2 \ldots X_cX_{c+1}$ is defined as a *Tandem Repeat* if and only if the following conditions hold for some string C (which is called the consensus pattern) with length t_C:

- For all i such that $1 \leq i \leq c$; $e_i = E(X'_i, C')$ should be less than or equal to $\theta_{max}(t_C)$ where $E(X'_i, C')$ is the number of errors in the optimal global alignment (X'_i, C') (with respect to some score function f) of the repeating unit X_i and consensus pattern C.

- For all i such that $1 \leq i < c$; $E(X'_i, X'_{i+1})$ should be less than or equal to $\theta_{max}(t)$ where $E(X'_i, X'_{i+1})$ is the number of errors in the optimal global alignment (X'_i, X'_{i+1}) (with respect to score function f) of the two adjacent repeating units X_i and X_{i+1} and t is the total number of matches, mismatches, insertions and deletions in that alignment.

- if X_{c+1} is a full string then $e_{c+1} = E(X'_{c+1}, C'_p)$ should be less than or equal to $\theta_{max}(length(C_p))$ where $E(X_{c+1}, C'_p)$ is the number of errors in the optimal global alignment (X'_{c+1}, C'_p) (with respect to score function f) of the *partial*

[1]For both single and multiple repeats

[2]The term *tandem repeat* will refer to both perfect and approximate tandem repeats from now on

repeating unit X_{c+1} with some nonempty prefix C_p of the consensus C such that $C_p \neq C$.

Again the alignment score function f is an input the algorithm and is universal for all tandem repeats. The threshold $\theta_{max}(t)$ is the same threshold used in the above definition of single repeats, however here it is also used to express the lower bound of similarity between the repeating units and the consensus pattern.

Now let's extend the above definition of the Tandem Repeat $Y = X_1X_2 \ldots X_cX_{c+1}$ with some additional properties:

Consensus pattern The string C is called the *consensus pattern* of the tandem repeat Y.

Consensus period The length t_C of the consensus pattern is called the *consensus period* of the tandem repeat Y.

Period The period t_i of a repeating unit X_i is defined as the length of the string X_i for $1 \leq i \leq c$. Then the *period* of the tandem repeat Y is the most common period among the periods of the repeating units. If there is more than one candidate then the one closest to the consensus period is chosen.

Copy number The sum $c + \dfrac{length(C_p)}{t_C}$ is called the *copy number* of the tandem repeat Y. In other words it is the sum of the number of full repeating units plus the fraction of the partial repeating unit.

Score of a repeating unit $s_i = F(X'_i, C')$, which is the score of the optimal global alignment (with respect to some score function f) of the repeating unit X_i with consensus pattern C, is called the score of the repeating unit X_i.

Score of the partial repeating unit $s_{c+1} = F(X'_{c+1}, C'_p)$, which is the score of the optimal global alignment (with respect to some score function f) of the partial repeating unit X_{c+1} with some nonempty prefix C_p of the consensus C such that $C_p \neq C$, is called the score of the partial repeating unit. It's 0 if X_{c+1} is empty string.

Total score The sum $S(Y) = \sum_{i=1}^{c+1} s_i$ is called the *total score* of the tandem repeat Y where s_i's are the scores of each repeating unit defined above.

It can be seen that all the above properties depend on the consensus pattern C. Since there may be more than one consensus pattern that satisfies the conditions in the definition, none of the properties is unique. Note that even if there's a unique consensus pattern, the repeating units can be decomposed in different ways.

3.1.3 Goal

The goal of the algorithm is to report the tandem repeats in a given sequence that satisfies the given criteria. The inputs to the algorithm are as follows:

Sequence A sequence S over the alphabet $\Sigma = \{\mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{T}\}$ in which the tandem repeats will be searched for.

Error probabilities The pair (p_M, p_I) which defines the error probabilities in the assumption of the stochastic repeat replication process described in Section 3.1.1.

Minimum period The minimum period t_{min} of the repeats which will be reported.

Maximum period The maximum period t_{max} of the repeats which will be reported.

Total score threshold θ_{score} such that only the tandem repeats with total score equal to or greater than this threshold will be reported.

Alignment score function The parameters (m_f, s_f, i_f) of the alignment score function f which will be used in alignments for determining the scores of repeats.

These parameters are the score for a match, penalty for a mismatch, and penalty for an indel, respectively.

The sequence S is searched for tandem repeats with period t in the range $t_{min} \leq t \leq t_{max}$ and with total score equal to or greater than the θ_{score} according to the score function f. The following properties of the found repeats are reported in the output:

Start - End positions The integers i and j where $S_{i,j}$ is the repeat.

Consensus The consensus pattern of the repeat.

Period The period of the repeat. Note that it may be different than the consensus period.

Total score The total score of the repeat with respect to score function f (Section 3.1.2).

Alignments The details of the alignments of each repeating unit with the consensus.

The details of output are discussed in Section 4.1.3.

3.2 Detection Phase

The algorithm uses the *filtration* technique and it's similar to the other algorithms mentioned before (in Chapter 2.2.3) in the sense that it consists of two phases. The first phase, which finds the regions exhibiting evidence of being a tandem repeat, is explained in this section.

3.2.1 Observing the Recurrences of Short Substrings

Remember that the Tandem Repeats Finder [6] looks for exact matches of short substrings to gather clues about repeat regions. For each substring (of the whole sequence S) of length w it looks for all the previous occurrences of that substring.

The algorithm in this thesis looks only for the immediately preceding occurrence of substrings (windows) of length w starting at every position. We claim that observing only the immediately preceding occurrences of short substrings is as powerful as (also more efficient than) observing all the previous occurrences when the window size w is properly chosen for different ranges of repeat periods. Presence of significant number of substrings occurring t positions before in a region is a strong evidence of the argument that the corresponding region is repeated t positions before. To illustrate the idea let's consider a perfect single tandem repeat of period $t = 20$ (the underlined substring is the second repeating unit, which is equal to the non-underlined part):

$$Y = \textbf{CGCAAGTTCATGAAAGAACC}\underline{\textbf{CGCAAGTTCATGAAAGAACC}}$$

Since it's a perfect tandem repeat, it's obvious that $Y_{i,i+w} = Y_{i+t,i+t+w}$ for all $1 \leq w \leq t$ and $1 \leq i \leq t - w + 1$. In other words if we look for all occurrences of substrings of Y with length w starting at positions between $t + 1$ and $2t - w + 1$, we observe that those substrings also occur t positions before. However if we look for only the immediately preceding occurrence of those substrings we may observe that not all of the are observable at distance 20. For instance when $w = 1$ none of the 20 substrings are observable at distance 20, for $w = 2$ only 8 of the 19 substrings are observable, for $w = 3$ we observe 14 of the 18 substrings and for $w = 4$ all of them are observable. Therefore the window size of 3 or above is a good choice for detecting the repeats of period 20 since most of the windows are observable. The substrings which are observable are the ones which occur exactly once in a single repeating unit.

Definition Let $u(S, w)$ be a function of some string S and an integer w such that it is the number of substrings of length w which occurs exactly once in S.

Since there are at most $n = L - w + 1$ substrings of length w in S, it's clear that $0 \leq u(S, w) \leq n = L - w + 1$ where L is the length of the string S.

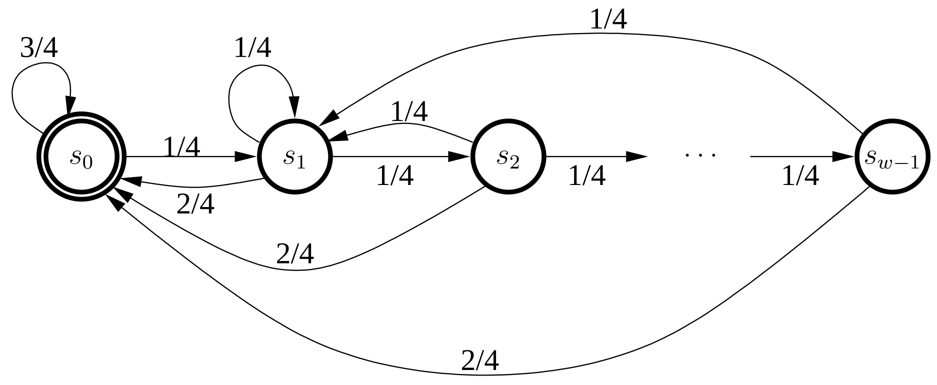


Figure 3.1: State diagram of automaton $\mathcal{A}$.

In the previous example $u(X,1) = 0$, $u(X,2) = 8$, $u(X,3) = 14$ and $u(X,4) = 17$ where $X =$ **CGCAAGTTCATGAAAGAACC**.

Let's now analyze the number of substrings of length w which occurs exactly once in a random string of length L.

Definition Let $U_{L,w}$ be a random variable which is the number of unique substrings of length w in a random string S of length L where each character of S is uniformly and independently chosen from a 4 symbol alphabet. Then it's clear that $U_{L,w} = u(S,w)$. From the previous definition we know that $0 \leq U_{L,w} \leq n = L - w + 1$.

Before studying the random variable $U_{L,w}$, we'll first analyze the probability that a fixed string v of length w *does not occur* in S. The detection of v in S may be modeled as follows. Consider an automaton $\mathcal{A}$ with with w states as illustrated in Figure 3.1. $\mathcal{A}$ has a reset state s_0 and a sequence of $w-1$ states associated with the recognition of v; the last state s_{w-1} has only "return" arcs. Here we have made the simplifying assumption that return arcs exist only to s_0 and s_1. (This simplifying assumption negligibly overestimates the probabilities.)

Denoting P_j the probability that a sequence of length $j \geq w$ does not contain v, by standard analysis (signal flow-graphs) we obtain the recurrence relation

$$P_j = P_{j-1} - \frac{1}{4^w} P_{j-w}$$

whose characteristic polynomial is $p(x) = x^w - x^{w-1} + 1/4^w$. The largest real root of $p(x)$ determines the behavior of P_n for large n. We note first that $p(1) = 4^{-w}$; moreover, $p'(1) = 1$, so that, approximating $p(x)$ around $x = 1$ with its tangent line at $x = 1$, we obtain

$$P(1 - 4^{-w}) \approx 0$$

i.e., $1 - 4^{-w}$ is the sought root. It follows that $P_j \approx (1 - 4^{-w})^{(j+H)}$, where H is a constant we now determine. We can directly determine that $P_j = 1$ for $j = 1, \ldots, w-1$ and $P_w = 1 - 4^{-w}$, so that we obtain $(1 - 4^{-w})^{w+H} = 1 - 4^{-w}$, i.e., $H = -w + 1$. Ignoring the approximation, we conclude:

$$P_j = \left(1 - 4^{-w}\right)^{j-w+1}$$

Let $n = L - w + 1$, so that $P_{n+w-1} = P_L = (1 - 1/4^w)^n$. Now consider the following analogy: We have 4^w bins (numbered $1, 2, \ldots, 4^w$) and n balls and each ball has probability $1/4^w$ to fall into any bin. Let θ_j be a binary variable which is 1 if and only if bin j remains empty after all the n balls have been thrown:

$$\Pr(\theta_j = 1) = (1 - 1/4^w)^n$$

The last equation can be interpreted as stating that the probability P_L that a sequence of length L does not contain a specific string of length w is the same as the probability $\Pr(\theta_j = 1)$ that none of the n thrown balls will fall into a specific bin in a collection of equally likely 4^w bins.

Now the number ν_0 of empty bins is

$$\nu_0 = \theta_1 + \theta_2 + \ldots \theta_{4^w}$$

Since these variables are identically distributed

$$\mathrm{E}[\nu_0] = \sum_{i=1}^{4^w} (1 - 1/4^w)^n = 4^w \,(1 - 1/4^w)^n$$

This illustrates a significant analogy between the expected number of never occurring strings of length w in a sequence of length n and the number of empty bins after throwing n balls into 4^w bins. Due to its inherent simplicity, we shall analyze the latter model to shed light on our original problem.

Again consider the process of throwing $n = L - w + 1$ balls into 4^w bins, where the balls can be thought as the n substrings of length w in a random string of length L and the bins can be seen as all the possible 4^w strings of length w. Let the binary random variable ϕ_i be 1 if and only if the number of balls in the bin i is 1 ($1 \leq i \leq 4^w$). Then

$$\Pr(\phi_i = 1) = \frac{n}{4^w}\left(1 - \frac{1}{4^w}\right)^{n-1}$$

since ϕ_i is 1 if and only if only one ball is thrown into bin i and the other $n-1$ balls are thrown into other $4^w - 1$ bins.

Now let's define the random variable $\nu_1 = \sum_{i=1}^{4^w} \phi_i$ as the number of bins with only one ball after throwing n balls to 4^w bins. Then the expectation of ν_1 is:

$$\mathrm{E}[\nu_1] = \mathrm{E}\left[\sum_{i=1}^{4^w} \phi_i\right] = \sum_{i=1}^{4^w} \mathrm{E}[\phi_i] = 4^w\,\mathrm{E}[\phi_i] = 4^w \frac{n}{4^w}\left(1 - \frac{1}{4^w}\right)^{n-1} = n(1 - 4^{-w})^{n-1}$$

since the ϕ_i's are identically distributed random variables.

The variance of ν_1 is:

$$\begin{aligned}
\mathrm{Var}[\nu_1] = \mathrm{E}\left[(\nu_1 - \mathrm{E}[\nu_1])^2\right] &= \mathrm{E}[\nu_1^2] - 2\,\mathrm{E}[\nu_1]\,\mathrm{E}[\nu_1] + \mathrm{E}[\nu_1]^2 \\
&= \mathrm{E}[\nu_1^2] - \mathrm{E}[\nu_1]^2 \\
&= \mathrm{E}\left[\left(\sum_{i=1}^{4^w} \phi_i\right)^2\right] - \mathrm{E}[\nu_1]^2 \\
&= \mathrm{E}\left[\sum_{i=1}^{4^w} \phi_i^2 + \sum_{i \neq j} \phi_i \phi_j\right] - \mathrm{E}[\nu_1]^2 \\
&= \mathrm{E}\left[\sum_{i=1}^{4^w} \phi_i^2\right] + \mathrm{E}\left[\sum_{i \neq j} \phi_i \phi_j\right] - \mathrm{E}[\nu_1]^2 \\
&= \mathrm{E}\left[\sum_{i=1}^{4^w} \phi_i\right] + \mathrm{E}\left[\sum_{i \neq j} \phi_i \phi_j\right] - \mathrm{E}[\nu_1]^2 \qquad (\text{since } \phi_i^2 = \phi_i) \\
&= \mathrm{E}[\nu_1] - \mathrm{E}[\nu_1]^2 + \mathrm{E}\left[\sum_{i \neq j} \phi_i \phi_j\right]
\end{aligned}$$

Next let's analyze the last term $\mathrm{E}\left[\sum_{i \neq j} \phi_i \phi_j\right]$. Note that $\phi_i \phi_j = 1$ if and only if $\phi_i = 1$ and $\phi_j = 1$. In other words, it's the event that only one ball is in bin i and one in bin j ($i \neq j$). So the probability of this event is:

$$\Pr(\phi_i \phi_j = 1) = \frac{n}{4^w} \frac{n-1}{4^w - 1} \left(1 - \frac{2}{4^w}\right)^{n-2}$$

since one ball will be thrown into bin i and one ball will be thrown into bin j and the remaining $n-2$ balls will be thrown into the remaining $4^w - 2$ bins. Now returning to the term that is being analyzed, since there are $4^w(4^w - 1)$ events ($i \neq j$):

$$\begin{aligned}
\mathrm{E}\left[\sum_{i \neq j} \phi_i \phi_j\right] &= \sum_{i \neq j} \left(\frac{n}{4^w} \frac{n-1}{4^w - 1} \left(1 - \frac{2}{4^w}\right)^{n-2}\right) \\
&= 4^w(4^w - 1) \frac{n}{4^w} \frac{n-1}{4^w - 1} \left(1 - \frac{2}{4^w}\right)^{n-2} \\
&= n(n-1) \left(1 - \frac{2}{4^w}\right)^{n-2}
\end{aligned}$$

since $\phi_i\phi_j$'s are identically distributed random variables where $i \neq j$. Now the variance becomes:

$$\begin{aligned}\mathrm{Var}[\nu_1] &= \mathrm{E}[\nu_1] - \mathrm{E}[\nu_1]^2 + \mathrm{E}\left[\sum_{i\neq j} \phi_i\phi_j\right] \\ &= \mathrm{E}[\nu_1] - \mathrm{E}[\nu_1]^2 + n(n-1)\left(1 - \frac{2}{4^w}\right)^{n-2}\end{aligned}$$

Several simulation results suggest that the expectation of random variable ν_1 is an almost perfect approximation to the expectation of $U_{L,w}$ where $n = L - w + 1$ (Figure 3.2(a), 3.3(a) and 3.4(a)). Similarly, the variance of ν_1 approximates the variance of $U_{L,w}$ well enough (Figure 3.2(b), 3.3(b) and 3.4(b)).

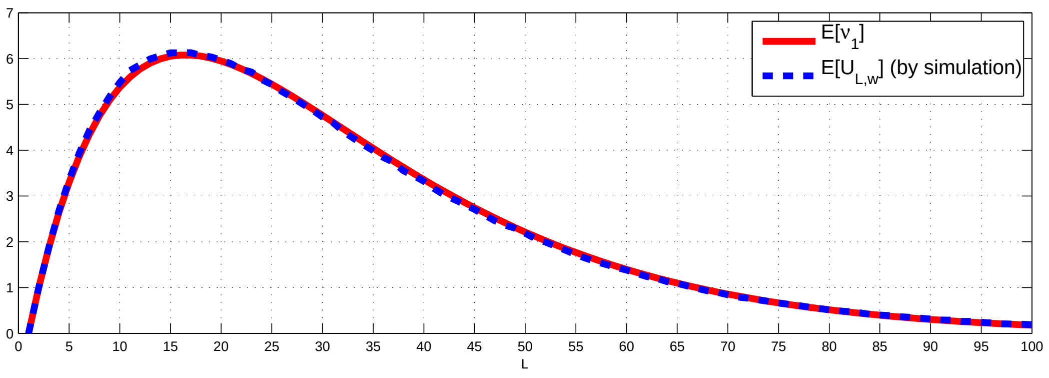
$$\mathrm{E}[U_{L,w}] \approx \mathrm{E}[\nu_1]$$

$$\mathrm{Var}[U_{L,w}] \approx \mathrm{Var}[\nu_1]$$

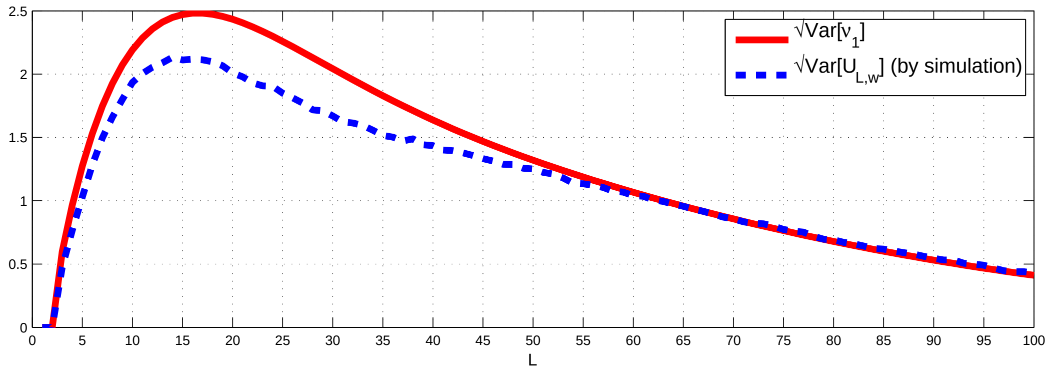
Under the assumption that a genomic sequence is a random sequence, the random variables $U_{L,w}$ and ν_1 hint that a properly chosen substring length w allows the algorithm to detect repeating regions without having the need of observing all the occurrences of the substrings. Only observing the last occurrence of a substring of length w (called an *w-window scan*) is powerful enough to detect repeating regions and is much more efficient than observing all occurrences.

For instance, for repeats with period around $30 - 40$, almost all of the substring of length 4 are observable according to the Figure 3.4(a). Therefore the window size 4 is a good choice for such repeats.

For perfect tandem repeats, these distributions may be used to adjust the threshold on the number of substrings which are needed to be observed for considering the region as a tandem repeat. However substitutions, insertions and deletions *block out* some of these substrings. This situation will be discussed in Section 3.4.3.

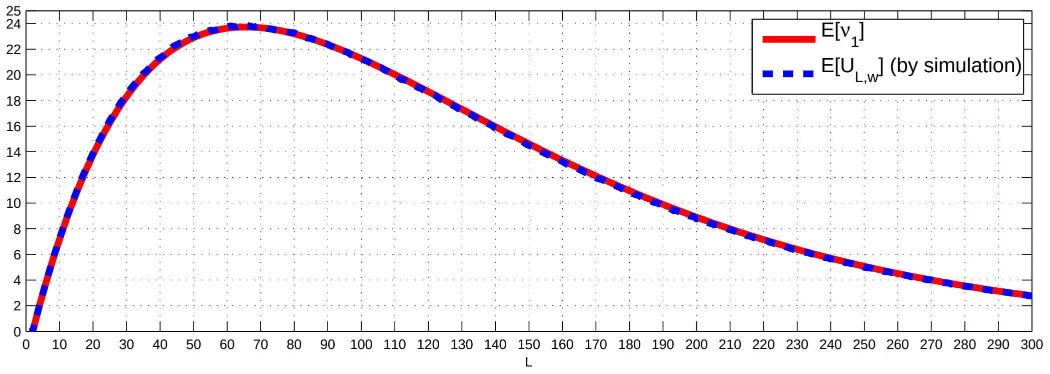


(a) $\mathrm{E}[\nu_1]$ versus the simulation results of $\mathrm{E}[U_{L,w}]$.

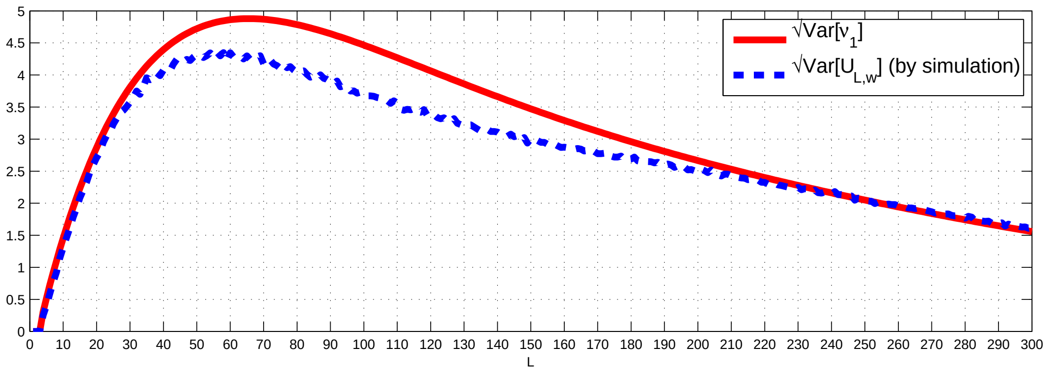


(b) $\sqrt{\mathrm{Var}[\nu_1]}$ versus the simulation results of $\sqrt{\mathrm{Var}[U_{L,w}]}$.

Figure 3.2: Comparison of ν_1 and simulation of $U_{L,w}$ for $w = 2$.

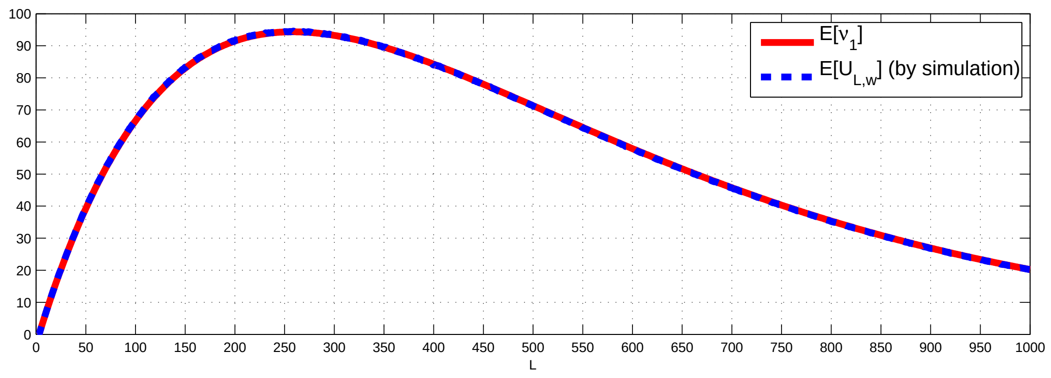


(a) $\mathrm{E}[\nu_1]$ versus the simulation results of $\mathrm{E}[U_{L,w}]$.

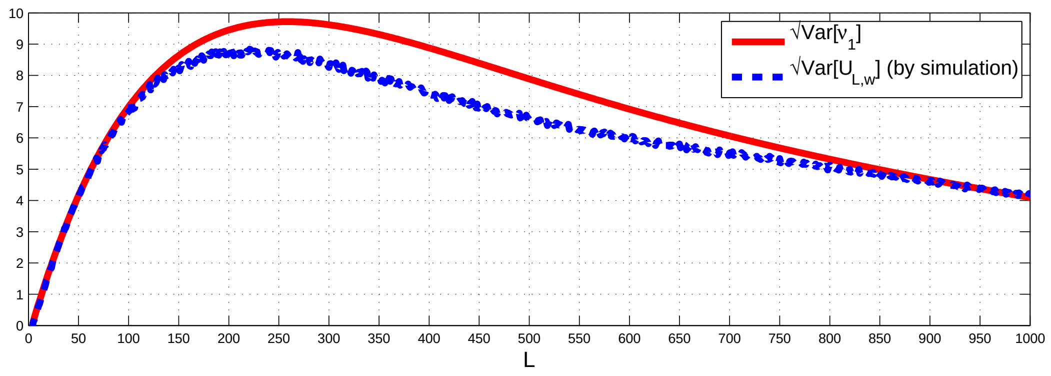


(b) $\sqrt{\mathrm{Var}[\nu_1]}$ versus the simulation results of $\sqrt{\mathrm{Var}[U_{L,w}]}$.

Figure 3.3: Comparison of ν_1 and simulation of $U_{L,w}$ for $w = 3$.



(a) $\mathrm{E}[\nu_1]$ versus the simulation results of $\mathrm{E}[U_{L,w}]$.



(b) $\sqrt{\mathrm{Var}[\nu_1]}$ versus the simulation results of $\sqrt{\mathrm{Var}[U_{L,w}]}$.

Figure 3.4: Comparison of ν_1 and simulation of $U_{L,w}$ for $w = 4$.

3.2.2 Distance Arrays

Since we are interested in observing only the immediately preceding occurrences of short substrings, the only information that will be needed is the distances between the occurrences of these substrings at each position. The array of these distances for each position is called the *distance array.* More formally:

Definition For a string S of length L, the *distance array* $\delta_{S,w}$ is an $(L - w + 1)$-component vector. $\delta_{S,w}[i]$ denotes the i^{th} element of this vector. Let W_i (called the w-string at position i) denote the substring $S_{i,i+w-1}$. Then:

$$\delta_{S,w}[i] = \begin{cases} 0 & \text{if string } W_i \text{ never occurs before position } i \\ h & \text{if } h = i - j \text{ where } j \text{ is the largest } j < i \text{ such that } W_j = W_i \end{cases}$$

In other words $\delta_{S,w}[i]$ is the distance between the current occurrence and the last occurrence of W_i.

Definition Let $\delta_{S,w}[i : j]$ denote the vector $\langle \delta_{S,w}[i] \quad \delta_{S,w}[i+1] \quad \cdots \quad \delta_{S,w}[j] \rangle$.

Let's return to example in Section 3.2.1 where $Y = XX$ is a perfect single tandem repeat with period 20:

$X =$ **CGCAAGTTCATGAAAGAACC**

$Y =$ **CGCAAGTTCATGAAAGAACC**<u>**CGCAAGTTCATGAAAGAACC**</u>

Then:

$$\delta_{Y,1} = \langle 0 \quad 0 \quad 2 \quad 0 \quad 1 \quad 4 \quad 0 \quad 1 \quad 6 \quad 5 \quad 3 \quad 6 \quad 3 \quad 1 \quad 1 \quad 4 \quad 2 \quad 1 \quad 10 \quad 1$$
$$1 \quad 6 \quad 2 \quad 6 \quad 1 \quad 4 \quad 16 \quad 1 \quad 6 \quad 5 \quad 3 \quad 6 \quad 3 \quad 1 \quad 1 \quad 4 \quad 2 \quad 1 \quad 10 \quad 1\rangle$$
$$\delta_{Y,2} = \langle 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 6 \quad 0 \quad 0 \quad 0 \quad 9 \quad 1 \quad 10 \quad 4 \quad 3 \quad 0 \quad 0 \quad 1$$
$$20 \quad 20 \quad 14 \quad 7 \quad 10 \quad 20 \quad 20 \quad 20 \quad 6 \quad 20 \quad 20 \quad 16 \quad 9 \quad 1 \quad 10 \quad 4 \quad 3 \quad 20 \quad 19\rangle$$
$$\delta_{Y,3} = \langle 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 10 \quad 0 \quad 4 \quad 0 \quad 0 \quad 0 \quad 0$$
$$20 \quad 20 \quad 20 \quad 10 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 16 \quad 20 \quad 10 \quad 20 \quad 4 \quad 20 \quad 20\rangle$$
$$\delta_{Y,4} = \langle 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$
$$20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20\rangle$$

And,

$$\delta_{Y,1}[21:40] = \langle 1 \quad 6 \quad 2 \quad 6 \quad 1 \quad 4 \quad 16 \quad 1 \quad 6 \quad 5 \quad 3 \quad 6 \quad 3 \quad 1 \quad 1 \quad 4 \quad 2 \quad 1 \quad 10 \quad 1\rangle$$
$$\delta_{Y,2}[21:39] = \langle 20 \quad 20 \quad 14 \quad 7 \quad 10 \quad 20 \quad 20 \quad 20 \quad 6 \quad 20 \quad 20 \quad 16 \quad 9 \quad 1 \quad 10 \quad 4 \quad 3 \quad 20 \quad 19\rangle$$
$$\delta_{Y,3}[21:38] = \langle 20 \quad 20 \quad 20 \quad 10 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 16 \quad 20 \quad 10 \quad 20 \quad 4 \quad 20 \quad 20\rangle$$
$$\delta_{Y,4}[21:37] = \langle 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20 \quad 20\rangle$$

This simple example on perfect repeats illustrates the use of w-window scanning and the use of distance arrays when detecting the tandem repeat regions. The distance arrays of $w = 2, 3$ and 4 have a significant number of 20's after position 21 therefore the portion of the string after position 21 provides strong evidence of being a repeating unit of a tandem repeat.

Now consider the following example of approximate tandem repeat $Y = XX'$ with period 20 where the first repeating unit X is the same as the one in the previous example but the second repeating unit X' has an insertion, two deletions and a substitution.

$$Y = \mathbf{CGCAAGTTCATGAAAGAACC}\underline{\mathbf{CGTCAAGTCCATGAGAACC}}$$

Here's an alignment of the two repeating units:
```
CG_CAAGTTCATGAAAGAACC
CGTCAAGTCCATG_A_GAACC
```

Only the distance array with $w = 3$ will be shown in this case:

$$\delta_{Y,3} = \langle 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 10 \quad 0 \quad 4 \quad 0 \quad 0 \quad 0 \quad 0$$
$$0 \quad 0 \quad 15 \quad 21 \quad 11 \quad 21 \quad 5 \quad 0 \quad 0 \quad 21 \quad 21 \quad 21 \quad 0 \quad 19 \quad 19 \quad 19 \quad 19\rangle$$

And,

$$\delta_{Y,3}[21:37] = \langle 0 \quad 0 \quad 15 \quad 21 \quad 11 \quad 21 \quad 5 \quad 0 \quad 0 \quad 21 \quad 21 \quad 21 \quad 0 \quad 19 \quad 19 \quad 19 \quad 19\rangle$$

Here the distance array is not as simple as the previous perfect repeat case. The mutations cause the w-strings around them to be hidden (*blocked out*) in the distance array; and the insertions (deletions) cause the values in the distance array to be increased (decreased) by 1.

The insertion at the 3^{rd} position in the second repeating unit blocks out the first three 3-strings of $\delta_{Y,3}[21 : 37]$ and increases all the other values by 1. Similarly the two deletions near the end decreases the 21's by 2. Also the substitution near the center of the second repeating unit blocks out three strings.

Even if this distance list is not as perfect as the previous example, it still provides a significant clue of a tandem repeat because it contains five 21's and four 19's which are in the neighborhood of 20.

As these two examples clarifies, the presence of some subsequence (of the distance array) of length close to t containing a significant number of values which are in the neighborhood of t is evidence of a candidate tandem repeat with period t. The goal of the detection phase of the algorithm is to find these subsequences (or *t-chains*) in the distance array. Some criteria relating to the minimum number of values, the length of the chain and the allowed neighborhood will be explained in Section 3.4.

3.2.3 Chains

A chain, to be formally defined below, can be thought as a subsequence of a distance array with some restrictions. The algorithm constructs the chains according to the distance array of the sequence. The chains which pass the acceptance criteria are then considered as candidate tandem repeat regions and corresponding portions of sequence S are passed in to the verification phase. The process of constructing the chains will be explained in Section 3.2.4.

Definition A *t-chain* or *chain* $\Gamma_{t,s}$ associated with a distance array $\delta_{S,w}$ is an l-component vector of non-negative integers where:

- s is called the *start position* of the chain.
- $\Gamma_{t,s}[i]$ denotes the $(i-s+1)^{th}$ element of the vector where $s \leq i \leq s+l-1$.
- $\Gamma_{t,s}[i:j]$ denotes the vector $\langle \Gamma_{t,s}[i] \quad \Gamma_{t,s}[i+1] \quad \cdots \quad \Gamma_{t,s}[j] \rangle$
- $h_{last}(i)$ in a chain $\Gamma_{t,s}$ denotes the last non-zero value in the vector $\Gamma_{t,s}[s:i-1]$
- $\Delta d(i)$ in a chain $\Gamma_{t,s}$ denotes the difference $i-j$ such that j is the position of the occurrence of the last non-zero value ($h_{last}(i)$) in the vector $\Gamma_{t,s}[s:i-1]$

and where the following two conditions hold:

1. $\Gamma_{t,s}[s] = \delta_{S,w}[s] = t$. In other words the first element of the chain is always t (and also the s^{th} element of the distance array is t). Therefore $\Gamma_{t,s}$ is called the *t-chain* starting at position s in the distance array $\delta_{S,w}$.

2. $$\Gamma_{t,s}[i] = \begin{cases} h & \begin{array}{l} \text{if } h = \delta_{S,w}[i] \neq 0 \text{ and} \\ t - \Delta t_{max}(t) \leq h \leq t + \Delta t_{max}(t) \text{ and} \\ del_{max}(\Delta d(i)) \leq h - h_{last}(i) \leq ins_{max}(\Delta d(i)) \end{array} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } 1 < i < l$$

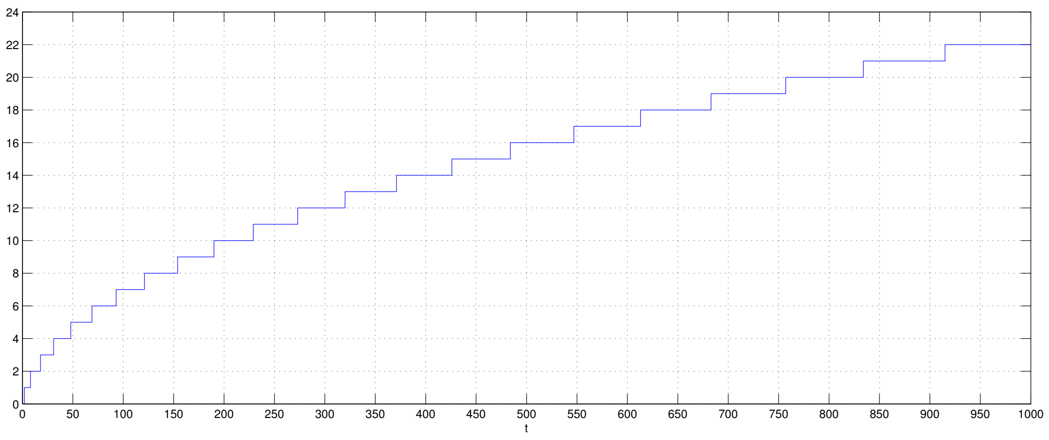


Figure 3.5: $\Delta t_{max}(t)$ for $p_I = 0.1$.

3. $\Gamma_{t,s}[s+l-1] = \delta_{S,w}[s+l-1] \neq 0$. The chains always end with a non-zero value which is also the value of the distance array at position $s+l-1$.

The conditions in the definition above says that that each t-chain $\Gamma_{t,s}$ is an array of 0's or positive integers within the $\Delta t_{max}(t)$ neighborhood of t. The value $\Delta t_{max}(t)$ basically depends on the probability of indels p_I which is mentioned in Section 3.1.1. It allows the insertions and deletions to be sensed by the detection phase. The calculation of this threshold is explained in the Section 3.4. Figure 3.5 shows the values of $\Delta t_{max}(t)$ for $p_I = 0.1$.

Also each t-chain starts with t and each value $\Gamma_{t,s}[i]$ is either 0 or $\delta_{S,w}[i]$. In other words a chain is a subarray of the distance array where some values are changed to 0. Remember the distance array from the previous example:

$$\delta_{Y,3} = \langle 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 10 \quad 0 \quad 4 \quad 0 \quad 0 \quad 0 \quad 0$$
$$0 \quad 0 \quad 15 \quad 21 \quad 11 \quad 21 \quad 5 \quad 0 \quad 0 \quad 21 \quad 21 \quad 21 \quad 0 \quad 19 \quad 19 \quad 19 \quad 19\rangle$$

Then $\Gamma_{21,24} = \langle 21 \quad 0 \quad 21 \quad 0 \quad 0 \quad 0 \quad 21 \quad 21 \quad 21 \quad 0 \quad 19 \quad 19 \quad 19 \quad 19\rangle$ is a 21-chain and it starts at position 24 of the distance array $\delta_{Y,3}$ (if we assume that $\Delta t_{max}(21) \geq 2$ and we relax the conditions about $del_{max}(d)$ and $ins_{max}(d)$).

Now let's define some properties of chains:

Hit Any non-zero element in the chain is called a *hit* and it's considered as a contribution to the significance of the chain in increasing the evidence of the region being a tandem repeat.

Gap The 0's in the chain are called a *gaps*. They are the indication of *missed* w-strings if the region associated with the chain is really a tandem repeat. A w-string may be missed either because of mutations or because of the fact that all w-strings are not observable when only the last occurrences of them are of interest (Section 3.2.1).

Score The *score* of a chain is simply the number of hits and it is represented by $\text{score}(\Gamma_{t,s})$. This score is compared with a threshold $score_{min}(t)$ to decide whether the region as a candidate or not.

Size or Length The number of elements in the chain is called the *size* or *length* and it is represented by $\text{size}(\Gamma_{t,s})$. The chains are not permitted to be longer than some threshold $size_{max}(t)$.

Average Hit This is simply the average of the all hit values in the chain and it is represented by $\mu_t(\Gamma_{t,s})$.

Now consider another example where a string $X =$ **AATAGCTTCGATCGG** is tandem-repeated with two insertions, forming the following approximate repeat:

$$\mathbf{AATAGCTTCGATCGG\underline{AATTAGGCTTCGATCGG}}$$

Here is an alignment of these two repeating units:
```
AA_TAG_CTTCGATCGG
AATTAGGCTTCGATCGG
```

The associated distance array of the sequence Y with window size $w = 3$ is (the first 15 values are omitted):

$$\delta_{Y,3}[16,30] = \langle 15 \quad 0 \quad 0 \quad 16 \quad 0 \quad 0 \quad 17 \quad 17 \quad 17 \quad 13 \quad 17 \quad 17 \quad 17 \quad 4 \quad 17 \rangle$$

Now assume that the threshold $\Delta t_{max}(15) \geq 2$ (we allow 2 indels) and the conditions about $del_{max}(h)$ and $ins_{max}(h)$ are ignored. Then the following is a valid 15 chain starting at position 16:

$$\Gamma_{15,16} = \langle 15 \quad 0 \quad 0 \quad 16 \quad 0 \quad 0 \quad 17 \quad 17 \quad 17 \quad 13 \quad 17 \quad 17 \quad 17 \quad 0 \quad 17 \rangle$$

Since this chain is associated with a tandem repeat of period 15, the first hit in the chain is expected. The first insertion in the second repeating unit renders the hit 16 reasonable. 17's are also accepted because of the second insertion. But the value 13 which is between 17's cannot be justified as a hit and it is more likely a *noise* or *peak* instead of a hit. These noise terms should not contribute to the score of the chain so they should be filtered out.

The thresholds $del_{max}(h)$ and $ins_{max}(h)$ are basically the criteria which prevent those noise terms from being accepted into the chains. They are the maximum allowed differences of any pairs of hits which satisfy the condition that the elements (if there are any) between those hits are only gaps (0's).

The second condition of the definition of chains says that if a value h is to be accepted to a chain at position i, then it must be in the range

$$[h_{last}(i) + del_{max}(d), h_{last}(i) + ins_{max}(d)]$$

where d is the distance of position i and the position where $h_{last}(i)$ occurs. Remember that $h_{last}(i)$ was the last hit which occurs before position i.

$del_{max}(d)$ is a negative valued nonincreasing function whereas $ins_{max}(d)$ is a positive valued nondecreasing function. Their calculations are shown in Section 3.4. Figure 3.6 shows the graph of $del_{max}(d)$ and $ins_{max}(d)$ used for $w = 4$ and $p_I = 0.1$.

Returning to the previous example, there is a value 13 in the distance array right after a 17 which may be interpreted erroneously as a representation of 4 deletions in a small space. The threshold $del_{max}(d)$ prevents this 13 from being accepted in the chain. In this case $d = 1$ because there is a non-zero value (which is 17) just before

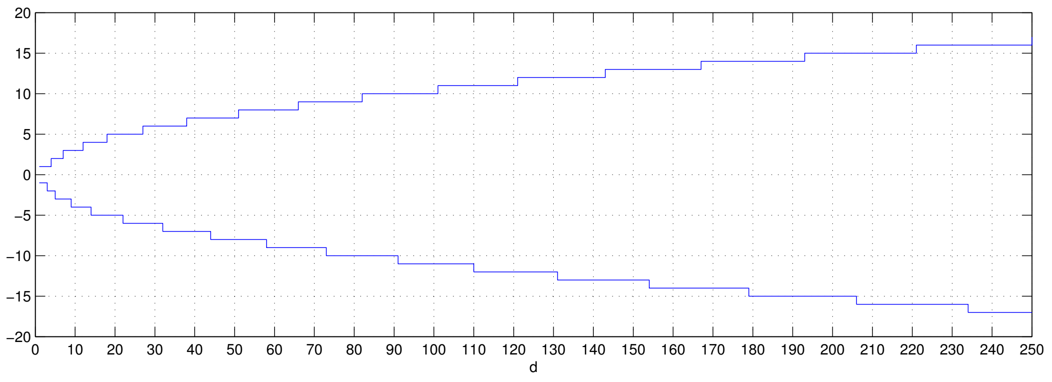


Figure 3.6: $del_{max}(d)$ and $ins_{max}(d)$ for $w = 4$ and $p_I = 0.1$.

the place where the acceptance of 13 is being considered. But $13 - 17 = -4$ is smaller than $del_{max}(1)$, so 13 is not accepted to the chain. Here the $del_{max}(d)$ in Figure 3.6 is used.

Briefly a t-chain $\Gamma_{t,s}$ with length l may be thought of as a copy of the $\delta_{S,w}[s : s + l - 1]$ (subsequence of the distance array) where the values which are believed to be noise (determined by the criteria $del_{max}(d)$ and $ins_{max}(d)$) and the values which are not close to t (determined by the criteria $\Delta t_{max}(t)$) are filtered out (set to 0's). Subsequently the chains having a score above a threshold are interpreted as candidate tandem repeats and verified by the verification phase of the algorithm.

Representing the Chains

As it will be clearer later, the chains are created with size 1 (containing only one hit) and then expanded by appending some values to the end or shrunk by trimming from the beginning. Chains are represented as doubly linked lists of runs of hits or gaps to make the implementation of these operations easy. For instance, the chain $\Gamma_{15,70} = \langle 15 \quad 15 \quad 0 \quad 0 \quad 16 \quad 16 \quad 16 \quad 15 \quad 15 \rangle$ is represented as the linked list in Figure 3.7.

The two basic operations on the chains are appending a value from the distance array and trimming the head of the chain. In more detail these operations are:

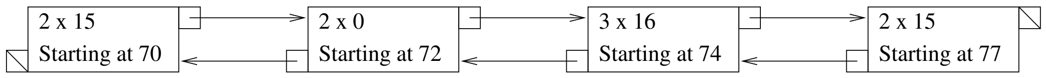


Figure 3.7: Representation of the chain $\Gamma_{15,70}$.

Append $\delta_{S,w}[i]$: When an element meets the criteria to be accepted into the chain, it needs to be appended to the linked list which represents the chain. Suppose that the element to be appended is $\delta_{S,w}[i]$ and the chain $\Gamma_{t,s}$ ends at position $e = s + l - 1$ where l is the length or size of the chain. There are two cases where:

1. $e = i - 1$: In this case the element is to be appended right after the last hit in the chain. If the element $\delta_{S,w}[i]$ is equal to the last hit $h_{last}(e + 1)$ then the last node in the linked list is updated such that it includes the position i as well. Otherwise (if $\delta_{S,w}[i] \neq h_{last}(e + 1)$) a new node with value $\delta_{S,w}[i]$ and position i is created and linked after the last node in the list.

2. $e < i - 1$: In this case the element is to be appended to the chain after some space. First the space between the last element in the chain and the new element should be filled with gaps. In order to do it a new node representing $i - e - 1$ gaps starting at position $e + 1$ is inserted to the linked list and then the node representing the new element $\delta_{S,w}[i]$ starting at position i is inserted to the list.

Trim to s': During the process of constructing the chains, the start position of a chain needs to be advanced so that it shouldn't be less than some given position s'. This operation trims the head of a chain $\Gamma_{t,s}$ so that the new start position s_{new} is the minimum over all s_{new}'s such that $\Gamma_{t,s}[s_{new}] = t$ and $s' \leq s_{new} \leq e$ where $e = s + l - 1$ is the end position of the chain. If no such s_{new} exists then the chain is destroyed.

The nodes in the linked list are scanned from the head to the tail until a node representing a run of hits with value t including a position which is not less than s' is found. All the previous nodes are deleted and the found node is adjusted so that it starts at position s_{new}.

For example our previous chain $\Gamma_{15,70} = \langle 15 \quad 15 \quad 0 \quad 0 \quad 16 \quad 16 \quad 16 \quad 15 \quad 15 \rangle$ becomes a new chain $\Gamma_{15,77} = \langle 15 \quad 15 \quad 0 \quad 0 \quad 16 \rangle$ after the operations **Trim to** 73 and **Append** $\delta_{S,w}[81] = 16$. Note that the operation **Trim to** 73 shifts the start position from 70 to 77 because 77 is the first position after 73 where the hit value is equal to $t = 15$.

3.2.4 Constructing the Chains

After the introduction of chains and their functions in the previous section, the process of constructing and using them are described in this section.

As described earlier, chains are constructed from distance arrays. The distance array of a specific window size can be constructed in $O(n)$ time by a single pass over the sequence, where n is the sequence length. The chains are constructed on-the-fly as the distance array is constructed and each element of the distance array is processed only once. One or both of the following happens for each element of the distance array:

1. That element may be added to one or more of the existing chains
2. A new chain is created starting with that element. This case always happens if the first one didn't happen (It may also happen with the first case together).

After that element is processed, it's not needed anymore so only the current element is stored in memory instead of the whole distance array. It was mentioned that the distance array can be computed in $O(n)$ time. The space complexity is $O(4^w)$ because there are 4^w possible w-strings. When a w-string is detected at position i, the position (call j) of its previous occurrence is fetched from a list of size 4^w and i

is stored in the list. Then the i^{th} value of the distance array is simply the difference $i - j$.

Chain Lists

Assume that tandem repeats with period in a range $[t_{min}, t_{max}]$ are searched for. This implies that we're interested in all t-chains where, $t_{min} \leq t \leq t_{max}$. When processing (only once) the element at position i of a distance array at step i, there may be more than one chain at that step (practically almost always more than one). There may even exist several t-chains having the same t at that step. This requires that chains be stored in a way that inserting a new chain, deleting an existing chain, and searching chains for specific t's, can be performed efficiently. Therefore the t-chains with same t are stored in a *chain set* of t, and these chain sets (at most $t_{size} = t_{max} - t_{min} + 1$ of them exist) are indexed by a *red-black tree*. This data structure is called a *chain list*. In addition to being accessible in $O(\log t_{size})$ time by searching in the red-black tree, the chain sets are also accessible in $O(1)$ time directly by their index. The search with red-black tree method will be used when a range of chains will be processed.

The chain sets are simple linked lists which hold only the chains with same t values. Whenever a chain set becomes empty (when all the chains with a specific t are destroyed) that set is deleted from memory and removed from the red-black tree. Therefore no empty chain sets exist in the chain list data structure. Each chain set also maintains a link to the next chain set. Figure 3.8 illustrates a chain list data structure where the interested t range is $[t_{min} = 10, t_{max} = 20]$ and there are 2 14-chains, 3 19-chains, a 12-chain, an 11-chain and a 17-chain.

There are some operations on the chain lists which are used during the algorithm. These are:

Inserting a t-chain to the chain list: If there's already a chain in the chain list with same t, then the chain set of t exists and it can be directly accessed in

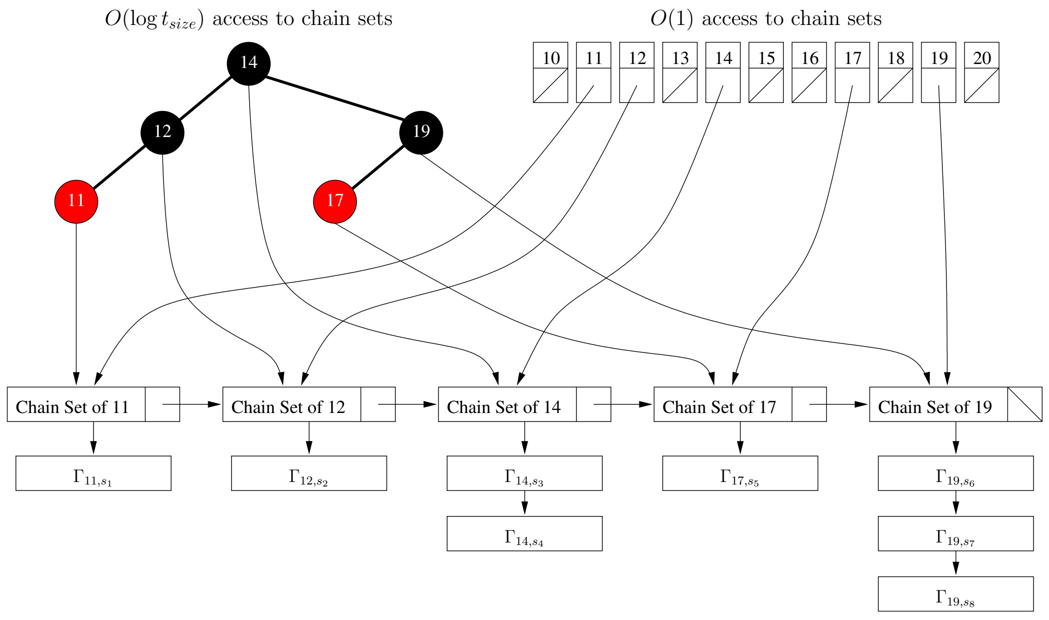


Figure 3.8: The *chain list* data structure.

constant time. Since inserting an element to a chain set (which is a linked list) takes $O(1)$ time, the insertion is a constant time operation when there's already a chain with same t.

If no chain with same t exists then a new chain set of t has to be created. This operation takes $O(\log t_{size})$ time since inserting an element and updating the pointers to the previous and next elements in a red-black tree takes logarithmic time ($t_{size} = t_{max} - t_{min} + 1$ is the maximum number of chain sets in a chain list). After creating the set it takes constant time to add the chain to the set. Therefore the overall time needed is $O(\log t_{size})$ in this case.

Accessing a chain set of minimum t where $t >= t_{low}$: This operation is used when the chains having a start value within a range $[t_{low}, t_{high}]$ are needed to be processed. This operation corresponds to a binary search in the red-black tree which has a time complexity of $O(\log t_{size})$ since there may be at most t_{size} sets.

Deleting a chain set of t: When a chain set becomes empty it's removed from the memory and from the red-black tree. Since deleting an element from the red-black tree takes logarithmic time, this operation takes $O(\log t_{size})$ time.

The Construction Process

After introducing the *chain list* data structure and the operations on it, it's now time to explain the process of constructing chains.

As mentioned earlier a window with size w is placed to the first position and slided one position at each step during the process of computing the elements of the distance array. Briefly, at each step i of this process, all chains which may potentially include the position i are kept in a *chain list* and the promising ones (passing acceptance criteria) are verified by the verification phase which will be explained later. If the verification phase succeeds for some chain, then that chain and all the related chains are removed from the list. In more detail, at each step i (where $1 \leq i \leq L$ and L is the length of the input sequence S):

1. The element $\delta_{S,w}[i]$ is computed in constant time (explained previously). If $\delta_{S,w}[i] = 0$ then the following steps are skipped and step 5 is executed.

2. The range $[t_{low}, t_{high}]$ of t-chains which may be interested in accepting the element $\delta_{S,w}[i]$ is computed. In other words the values t_{low} and t_{high} are determined such that the minimum t_{low} satisfying the inequality $t_{low} + \Delta t_{max}(t_{low}) \geq \delta_{S,w}[i]$; and the maximum t_{high} satisfying the inequality $t_{high} - \Delta t_{max}(t_{high}) \leq \delta_{S,w}[i]$ are found. These pairs for each possible value are calculated and stored in a table beforehand when preparing the criteria (Section 3.4), thus this step takes constant time.

3. For each t-chain $\Gamma_{t,s}$ such that $t_{low} \leq t \leq t_{high}$:

   (a) If $i - s > size_{max}(t)$ where $size_{max}(t)$ is a threshold (Section 3.4) limiting the maximum size of chains, then the start position of the chain is advanced from s to s_{new} such that s_{new} is the minimum value satisfying $s \leq s_{new} < s + size_{max}(t)$ and $\Gamma_{t,s}[s_{new}] = t$ (remember the trimming operation on the chains). If no such s_{new} exists then the chain is deleted and all the following steps (3b and 3c) are omitted.

(b) The value $\delta_{S,w}[i]$ is proposed to the chain. In other words the criteria of $del_{max}(d)$ and $ins_{max}(d)$ (Section 3.2.3) are checked where d is the distance between the current position and the position where the last non-zero element occurs in the chain. If $h + del_{max}(d) \leq \delta_{S,w}[i] \leq h + ins_{max}(d)$, then the value $\delta_{S,w}[i]$ is accepted and appended to the chain (where h was the last non-zero element in the chain before the current position).

(c) The chain is checked for two criteria. These are:

   i. score$(\Gamma_{t,s}) \geq score_{min}(t)$: The score of the chain should be greater than or equal to the threshold $score_{min}(t)$.

   ii. $|t - \mu_t(\Gamma_{t,s})| \leq 2$: The difference between t and the average hit values should be less than or equal to 2.

If these conditions are met then this chain is considered as a candidate tandem repeat region and the verification phase (Section 3.3) takes place. If it's the case that the verification phase returns with a successful tandem repeat then this chain and all the chains

$$\{\Gamma_{t',s'} \quad | \quad t_r - \Delta t_{max}(t_r) \leq t' \leq t_r + \Delta t_{max}(t_r) \quad \wedge \quad s_r \leq s' \leq e_r\}$$

are destroyed where t_r is the period, s_r is the start position and e_r is the end position of the detected tandem repeat in the verification phase. In other words all the t-chains where t is close to the period of the detected tandem repeat and where the start of the chain is inside the tandem repeat are destroyed in the assumption that they were related to the already detected tandem repeat.

If the verification phase fails, then the chain is kept in the list with the expectation that appending some additional terms to the right and trimming the head may result in a successful chain later.

4. If $\delta_{S,w}[i]$ is not accepted to any t-chain where $t = \delta_{S,w}[i]$, then a new chain $\Gamma_{t,s}$ is created and added to the chain list with $t = \delta_{S,w}[i]$ and $s = i$.

5. The window which was at position i is slided one position to the right so i is increased by 1. If $i > L - w + 1$ then the process is complete because the whole sequence is scanned, else execution returns to the first step.

In the steps 3 and 3c, the logarithmic-time tree search operation, which was explained before, is used in order to access the chains in a specific range of t's.

To summarize the above algorithm, at each step i of the process of scanning the sequence, all chains which may potentially include the position i are kept in the *chain list* and the ones which pass the criteria are verified by the verification phase which will be explained next. If the verification phase succeeds for some chain, then that chain and all the related chains are removed from the list.

Usually the sequence may be scanned for more than one window sizes. Since each distance array will be entirely different for each w, the chains which result from these distance arrays will be different. The algorithm which was defined above is only for one window size, however it's not practically different when several window sizes are in consideration. The same steps take place for each w simultaneously. The only trick is that each execution (each process for different w's) keeps its own separate chain list, but whenever a tandem repeat is verified then the related chains in all the chain lists (for all w's) are destroyed in step 3c.

3.3 Verification Phase

Whenever a chain passes the criteria for being a candidate tandem repeat region, then the surrounding portion is verified for being an actual tandem repeat via two types of alignments. The first type of alignment is computed (only once) to detect several potential start positions for the repeat (described in Section 3.3.1) and the second type is computed to verify these start positions (once for each pair of start positions until a verification succeeds; described in Section 3.3.2).

3.3.1 Detecting the Start Positions

Let's assume that a chain $\Gamma_{t,s}$ related to a window size w passes the criteria and qualifies to be verified for a tandem repeat. According to the definition of chains the first element of $\Gamma_{t,s}$ is t and the last element is $h_{last}(s+l)$ where l is the length of the chain. Then being qualified is an indication of the similarity between the substrings $S_{c,d}$ (which we call the *span of the chain*) and $S_{a,b}$ (which we call the *matching span of the chain*) where:

$c = s$ is the start position of the chain (*start of the span*).

$d = s + l + w - 2$ is the position where the last w-string of the chain ends (*end of the span*).

$a = c - t = s - t$ is the start of the span minus the first hit value (*start of the matching span*).

$b = d - h_{last}(s + l) = s + l + w - 2 - h_{last}(s + l)$ is the end of the span minus the last hit value (*end of the matching span*).

To illustrate these definitions, consider the following sequence S where there is an approximate repeat $Y = XX'$ of period 12 starting at position 6 and the two

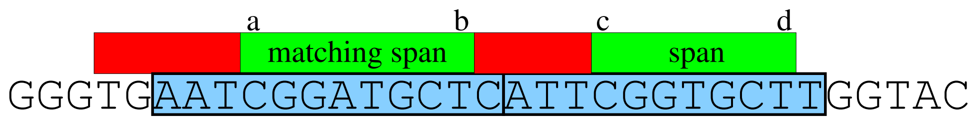


Figure 3.9: The *matching span* and *span* of the chain $\Gamma_{12,21}$.

repeating units X and X' differ by two substitutions and a deletion:

$$S = \mathbf{GGGTG\underline{AATCGGATGCTC}\overline{ATTCGGTGCTT}GGTAC}$$

The first repeating unit $X = \underline{\mathbf{AATCGGATGCTC}}$ is underlined and the second repeating unit $X' = \overline{\mathbf{ATTCGGTGCTT}}$ is overlined above. The alignment of these two units is:

```
AATCGGATGCTC
ATTCGG_TGCTT
```

The distance array $\delta_{S,2}$ of this sequence with 2-windows is:

$$\delta_{S,2} = \langle 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 8 \quad 6 \quad 5 \quad 9 \quad 0 \quad 0 \quad 8$$
$$0 \quad 6 \quad 0 \quad 4 \quad 12 \quad 12 \quad 20 \quad 11 \quad 11 \quad 11 \quad 8 \quad 4 \quad 7 \quad 7 \quad 0 \quad 0\rangle$$

Now assume that the following 12-chain starting at position 21 passed the criteria (which is really a strong chain):

$$\Gamma_{12,21} = \langle 12 \quad 12 \quad 0 \quad 11 \quad 11 \quad 11\rangle$$

Then the span of this chain is $S_{21,27} =$ **CGGTGCT** and the matching span is $S_{9,16} =$ **CGGATGCT**. Notice how the chain reflects the similarity of these two substrings. Figure 3.9 shows the span and the matching span in green and the repeating units in blue.

The chain $\Gamma_{12,21}$ shows us that the green portions in the figure are sufficiently similar that it may be assumed these portions (*matching* span and *span*) will be included in the repeat in separate units.

Now the problem to be solved is to find what portion of the first red region will be included in the first repeating unit and what portion of the second red region will be included in the second unit. In other words it is to find the pair (s_1, s_2) such that a proposed tandem repeat with the first repeating unit starting at position s_1 and the second unit starting at position s_2 yields a successful repeat where $s_1 \leq a$ and $b < s_2 \leq c$.

The first obvious pair worth trying is $(s - t, s)$ (start of the matching span and start of the span). However several empirical results teach us that the first repeating unit may start before the matching span and the second unit may start before the span because of some mutations in the beginning of the repeat (which is also the case in the above example). Therefore two more candidate start pairs will be proposed by an alignment which is called the *suffix alignment*.

Definition A *suffix alignment* of two strings X and Y from alphabet Σ is a pair of strings (X', Y') such that:

- X' and Y' only contain characters from the alphabet $\Sigma' = \Sigma \cup \{\_\}$.
- $length(X') = length(Y')$.
- removing all occurrences of the character $'\_'$ from X' yields a *suffix* of X.
- removing all occurrences of the character $'\_'$ from Y' yields a *suffix* of Y.

In other words a suffix alignment of two strings is a global alignment of two respective suffixes.

Analogous to the computation of other alignments, the computation of suffix alignment is done by filling a $n \times m$ matrix. However in this case each entry at row i and column j represents the optimum score of the global alignment of suffixes $X_{i,n}$ and $Y_{j,m}$ where n is the length of X and m is the length of Y. Obviously the matrix is filled from bottom to top and right to left. Again if the number of insertions and deletions is bounded by d then the computation takes $O(nd)$ time by only computing the elements in the diagonal band of width $2d$.

For the problem of finding candidate start pairs, the suffix alignment is computed for the two red substrings (let's call R_1 and R_2). As it can be seen in Figure 3.9 the second red substring R_2 is simply $S_{b+1,c-1}$ but the start position of the first red substring is unknown. It's adjusted so that R_1 is a bit longer than R_2 in order to allow space for insertions and deletions. More specifically the start of R_1 is set to the position which makes length(R_1) equal to length(R_2) + Δt_{max}(length(R_2)). And the width of the diagonal band in the alignment computation is set to the value $2\Delta t_{max}$(length(R_2)). Then among all the pairs (s_1, s_2) where the number of errors in the optimum global alignment of $S_{s_1,a-1}$ (which is a suffix of R_1) and $S_{s_2,c-1}$ (which is a suffix of R_2) is less than or equal to the threshold $\theta_{max}(a - s_1)$ ($a - s_1$ is the length of the first suffix $S_{s_1,a-1}$):

- The pair (l_1, l_2) is chosen so that l_1 and l_2 is minimum. This pair represents the start positions of the longest pair of suffixes of R_1 and R_2 respectively such that these suffixes are similar enough according to the criteria $\theta_{max}(a - l_1)$.

- The pair (m_1, m_2) is chosen so that the score of the alignment of $S_{m_1,a-1}$ and $S_{m_2,c-1}$ is maximum. This pair represents the start positions of suffixes of R_1 and R_2 respectively such that the alignment score of these suffixes are maximal, and they are similar enough according to the criteria $\theta_{max}(a - m_1)$.

For instance $(l_1, l_2) = (5, 17)$ and $(m_1, m_2) = (6, 18)$ for a score function $(2, 1, 2)$ according to the above example of Figure 3.9.

To put together, whenever a chain $\Gamma_{t,s}$ is qualified, the pairs (l_1, l_2) and (m_1, m_2) are computed as explained above.

- If the pair (l_1, l_2) hasn't been tried for verification alignment before, then a verification alignment (Section 3.3.2) starts from (l_1, l_2). If the alignment succeeds, the resulting repeat is reported.

- Otherwise (when verification fails) the pair (m_1, m_2) is tried (if hasn't been tried before) and the repeat is reported if found.
- Otherwise the pair $(s - t, s)$ is tried lastly for verification and the repeat is reported if found.

As seen above, a chain may cause up to three verification alignments to take place, but only one successful repeat is reported. The next section will describe how a start pair (s_1, s_2) is verified by the alignment called verification alignment. A pair (s_1, s_2) may be taken into consideration more than once because two different chains may have the same output for suffix alignments. Even the suffix alignment for a specific chain may end up with the pairs $(l_1, l_2) = (m_1, m_2)$. To prevent the redundant computation every start pair is stored in a list, and a second attempt for verification is not allowed for same pair.

3.3.2 Verifying the Tandem Repeats

After obtaining a possible pair of candidate start points (s_1, s_2) where s_1 is the start of the first unit and s_2 is the start of the second unit, the repeat has to be verified before reporting it as a tandem repeat.

When only single repeats are of interest, it's easy to verify a pair of start points (s_1, s_2). Let's denote the first candidate repeating unit as X_1 which is simply the substring S_{s_1,s_2-1}, and the second repeating unit as X_2, where X_2 is a substring of S starting at position s_2. To accurately detect the end of the second repeating unit, the string X_1 is aligned with a string X'_2 starting from s_2 in S and having the length $\text{length}(X'_2) = \text{length}(X_1) + \Delta t_{max}(\text{length}(X_1))$ to allow space for insertions. Then the second repeating unit X_2 is the prefix with the maximum score of global alignment with X_1 among all the prefixes of X'_2. If the number of errors in the optimal global alignment of X_1 and X_2 is less than or equal to the threshold $\theta_{max}(t)$ where

$t = \text{length}(X_1)$ is the period then the repeat then $Y = X_1X_2$ is reorted as a repeat. The alignment can be computed in $O(td)$ time where t is the period and $d = \Delta t_{max}(t)$ is the bound on insertions and deletions.

For example, consider the sequence

$$S = \textbf{AACTGTTAACTGTAACTTTAAGGGGGGG}$$

and a pair of start positions $(1, 8)$ where it's assumed that $\Delta t_{max}(7) = 1$ and a score function of $(2, 1, 2)$ is used. Then $X_1 = S_{1,7} =$ **AACTGTT** and $X_2' = S_{8,15} =$ **AACTGTAA**. And the the best scoring prefix of X_2' is $X_2 = S_{8,14} =$ **AACTGTA** with a score of 11. Therefore the repeat is detected as starting at position 1 and ending at position 14 with a period of 7. The repeat is shown below with the first unit underlined and the second overlined:

$$S = \underline{\textbf{AACTGTT}}\overline{\textbf{AACTGTA}}\textbf{ACTTTAAGGGGGGG}$$

Now let's consider the problem of detecting multiple repeats. The easiest method that can be considered is to extend the previous process for single repeats. That is, when a repeat with c units ending at position e is found then a substring starting at position $e + 1$ is aligned with one or more of the previous c units and is added to the repeat as the $(c + 1)^{th}$ unit if the alignments are successful.

For the previous example with the sequence:

$$S = \underline{\textbf{AACTGTT}}\overline{\textbf{AACTGTA}}\textbf{ACTTTAAGGGGGGG}$$

a repeat with two units (underlined and overlined) was found before. The repeat ends at position 14 so a third unit will start at position 15 according to the current method. The best substring starting at position 15 which can be considered as the third unit is $X_3 = S_{15,20} =$ **ACTTTA** because it's the most similar substring to both X_1 and X_2 among the other substrings starting at position 15. According to the best alignment between X_1 and X_3 $\substack{\texttt{AACTGTT}\\ \texttt{A\_CTTTA}}$ the score is 4, there's one deletion and two substitutions. The best alignment between X_2 and X_3 $\substack{\texttt{AACTGTA}\\\texttt{A\_CTTTA}}$ has score 7 and there's one deletion and one substitution. The repeat is shown below with the first unit underlined, the second one overlined and the third one underlined again:

$$S = \mathbf{\underline{AACTGTT}\overline{AACTGTA}\underline{ACTTTA}AGGGGGGG}$$

As an alternative to these three units, consider the following three units over the same sequence:

$$S = \mathbf{\underline{AACTGTT}\overline{AACTGT}\underline{AACTTT}AAGGGGGGG}$$

where $X_1' = \mathbf{AACTGTT}$, $X_2' = \mathbf{AACTGT}$ and $X_3' = \mathbf{AACTTT}$. Here the second repeating unit is ended at position 13 instead of 14, and therefore the third repeating unit starts at position 14 instead of 15. Now the best alignment between X_1' and X_3' $\substack{\texttt{AACTGTT}\\\texttt{AACT\_TT}}$ has score 10 and there is only one deletion. The best alignment between X_2' and X_3' $\substack{\texttt{AACTGT}\\\texttt{AACTTT}}$ has score 9 and there is only one substitution. As the alignments show, the second set of units is a better choice than the first set. Therefore the method of extending the number of units at each step without changing the previous units is not powerful enough for detecting multiple repeats.

One approach that is usually successful in decomposing a repeat Y into units is simply to compute the optimum global alignment of Y with $(X_1)^c$ where c is the copy number and $(X_1)^c$ is the repetition of the first unit c times (remember that we know the first repeating unit from the initial pair of start positions). Again for the previous example, let's assume that we know the copy number c is 4. Then the best alignment between a prefix of S and $(X_1)^4$ is:

```
AACTGTTAACTGTTAACTGTTAACTGTT
AACTGTTAACTGT_AACT_TTAAGGGGG
```

If we decompose this alignment from the positions where the repetitions of X_1 starts, we obtain

```
AACTGTT AACTGTT AACTGTT AACTGTT
AACTGTT AACTGT_ AACT_TT AAGGGGG
```

the following set of units:

$$\{X_1 = \mathbf{AACTGTT}, X_2 = \mathbf{AACTGT}, X_3 = \mathbf{AACTTT}, X_4 = \mathbf{AAGGGGG}\}$$

Notice that the first three units are the same as in the above example of the alternative and better decomposition.

Unfortunately the copy number c is one of the unknowns that's being searched for, therefore this method is not applicable directly. However another variation of local alignment called *wraparound dynamic programming* [16] is introduced to compute the optimum local alignment of a string S of size n and a periodic repeat P^n of a pattern P of size m in $O(nm)$ time. This algorithm can be used with a slight modification which restricts that only prefixes will be considered instead of all substrings during the local alignment. The computation is done by filling a $O(nm)$ matrix as in the other alignments, however the bound d on the number of insertions and deletions doesn't reduce the computation time to $O(nd)$ in this case because the whole row must be computed in order to allow the connections between the end and start of each row which allows the recognition of repetitions of the pattern.

Instead, a modified version of wraparound alignment is used which combines the idea of repeating pattern alignment like the wraparound dynamic programming and the idea of only computing a diagonal band of the alignment matrix to reduce the computation time. First a substring of the sequence with length $p + d$ is aligned with the pattern P where the pattern has length p and the diagonal band width is $2d$. The pattern P is the first repeating unit inititally. The cell with the minimum number of errors in the last column is marked after the computation of the alignment matrix. If there are several such cells then the one with the highest score is chosen. Then a second alignment starts with a substring of length $p + 2d$ with the pattern P where the diagonal band is recentered at the marked cell where the minimum number of errors is observed in the last column of the previous alignment matrix. However the last column of the previous alignment matrix is copied to the first column of this new alignment matrix before the computation starts. This process goes on and on until the difference between the lowest number of errors in the last column and the lowest number of errors in the first column in any alignment matrix happens to be above the threshold $\theta_{max}(p)$.

Now let's illustrate this alignment on the example that was used before where the sequence is

$$S = \mathbf{AACTGTTAACTGTAACTTTAAGGGGGGG}$$

and the pair of start positions is $(1, 8)$. The first repeating unit is set as $X_1 = S_{1,7}$ according to the start positions and we're interested in finding the remaining repeating units where we don't know the copy number. Therefore we'll start the alignment from position 8 of the sequence S and the pattern P will be $X_1 = \mathbf{AACTGTT}$. Let's assume that the diagonal width $2d$ is 2.

Then the first alignment matrix is shown at Table 3.1. In each cell the first entry is the score and the second entry (in parenthesis) is the number of errors in the corresponding alignment. The cell with the minimum number of errors in the last column is at 14^{th} row with 1 error and score of 11 (the cell at 13^{th} row has also 1 error but its score is less than 11) and the minimum number of errors in the first column is 0. Since $1 - 0 \leq \theta_{max}(7)$ the alignment process continues.

Now the last column will be copied to the first column of a new alignment matrix where the diagonal will be recentered at row 14 (the same diagonal with the previous alignment) and the alignment is computed. Table 3.2 shows this alignment matrix. The cell with the minimum number of errors in the last column is at 19^{th} row with 2 errors and score of 20 and the minimum number of errors in the first column is 1. Since $2 - 1 \leq \theta_{max}(7)$ the alignment process continues.

Again the last column will be copied to the first column of a new alignment matrix where the diagonal will be recentered at row 19 (two rows above the previous diagonal) and the alignment is computed. Table 3.3 shows this alignment matrix. The cell with

			A	A	C	T	G	T	T
		0 (0)	-2 (1)	-4 (2)					
8	A	-2 (1)	2 (0)	0 (1)	-2 (2)				
9	A	-4 (2)	0 (1)	4 (0)	2 (1)	0 (2)			
10	C		-2 (2)	2 (1)	6 (0)	4 (1)	2 (2)		
11	T			0 (2)	4 (1)	8 (0)	6 (1)	4 (2)	
12	G				2 (2)	6 (1)	10 (0)	8 (1)	6 (2)
13	T					4 (2)	8 (1)	12 (0)	10 (1)
14	A						6 (2)	10 (1)	11 (1)
15	A							8 (2)	9 (2)
16	C								7 (3)

Table 3.1: Alignment of $S_{8,16}$ and X_1.

			A	A	C	T	G	T	T
12	G	6 (2)							
13	T	10 (1)	8 (2)						
14	A	11 (1)	12 (1)	10 (2)					
15	A	9 (2)	13 (1)	14 (1)	12 (2)				
16	C	7 (3)	11 (2)	12 (2)	16 (1)	14 (2)			
17	T		9 (3)	10 (3)	14 (2)	18 (1)	16 (2)		
18	T			8 (4)	12 (3)	16 (2)	17 (2)	18 (2)	
19	T				10 (4)	14 (3)	15 (3)	19 (2)	20 (2)
20	A					12 (4)	13 (4)	17 (3)	18 (3)
21	A						11 (5)	15 (4)	16 (4)
22	G							13 (5)	14 (5)
23	G								12 (6)

Table 3.2: Alignment of $S_{12,23}$ and X_1.

			A	A	C	T	G	T	T
19	T	20 (2)	18 (3)	16 (4)					
20	A	18 (3)	22 (2)	20 (3)	18 (4)				
21	A	16 (4)	20 (3)	24 (2)	22 (3)	20 (4)			
22	G		18 (4)	22 (3)	23 (3)	21 (4)	22 (4)		
23	G			20 (4)	21 (4)	22 (4)	23 (4)	21 (5)	
24	G				19 (5)	20 (5)	24 (4)	22 (5)	20 (6)
25	G					18 (6)	22 (5)	23 (5)	21 (6)
26	G						20 (6)	21 (6)	22 (6)
27	G							19 (7)	20 (7)
28	G								18 (8)

Table 3.3: Alignment of $S_{19,28}$ and X_1.

the minimum number of errors in the last column is at 26^{th} row with 6 errors and the minimum number of errors in the first column is 2. Since $6 - 2 > \theta_{max}(7)$ no more alignments take place.

The end point of the repeat is the cell at row i and column j such that the number of errors in that cell minus the minimum number of errors in the first column is not greater than $\theta_{max}(j)$ and the score of the cell is maximal. The cell with score 24 and errors 2 in the 21^{th} row is the one satisfying this condition so the alignment is considered to be ended at this cell.

After finding the position where the alignment ended, now it's the time to back-trace from that position and find the whole alignment. If we combine all the three alignment matrices virtually into a bigger alignment matrix, we obtain the alignment of the sequence with the periodic repeat of the pattern where the diagonal is recentered at each start position of the pattern in the repeat accordingly (Table 3.4).

			A	A	C	T	G	T	T	A	A	C	T	G	T	T	A	A	C	T	G	T	T
		0 (0)	−2 (1)	−4 (2)																			
8	A	−2 (1)	2 (0)	0 (1)	−2 (2)																		
9	A	−4 (2)	0 (1)	4 (0)	2 (1)	0 (2)																	
10	C		−2 (2)	2 (1)	6 (0)	4 (1)	2 (2)																
11	T			0 (2)	4 (1)	8 (0)	6 (1)	4 (2)															
12	G				2 (2)	6 (1)	10 (0)	8 (1)	6 (2)														
13	T					4 (2)	8 (1)	12 (0)	10 (1)	8 (2)													
14	A						6 (2)	10 (1)	11 (1)	12 (1)	10 (2)												
15	A							8 (2)	9 (2)	13 (1)	14 (1)	12 (2)											
16	C								7 (3)	11 (2)	12 (2)	16 (1)	14 (2)										
17	T									9 (3)	10 (3)	14 (2)	18 (1)	16 (2)									
18	T										8 (4)	12 (3)	16 (2)	17 (2)	18 (2)								
19	T											10 (4)	14 (3)	15 (3)	19 (2)	20 (2)	18 (3)	16 (4)					
20	A												12 (4)	13 (4)	17 (3)	18 (3)	22 (2)	20 (3)	18 (4)				
21	A													11 (5)	15 (4)	16 (4)	20 (3)	24 (2)	22 (3)	20 (4)			
22	G														13 (5)	14 (5)	18 (4)	22 (3)	23 (3)	21 (4)	22 (4)		
23	G															12 (6)		20 (4)	21 (4)	22 (4)	23 (4)	21 (5)	
24	G																		19 (5)	20 (5)	24 (4)	22 (5)	20 (6)
25	G																			18 (6)	22 (5)	23 (5)	21 (6)
26	G																				20 (6)	21 (6)	22 (6)
27	G																					19 (7)	20 (7)
28	G																						18 (8)

Table 3.4: Alignment of $S_{8,28}$ and $(X_1)^3$.

Backtracing from the position where we ended the alignment (second character of the third repeat of the pattern which is the cell with the score 24 in the 21th row) we obtain the following alignment:

```
AACTGTTAACTGTTAA
AACTGT_AACT_TTAA
```

Now if we append the alignment of the pattern with itself to the beginning of this alignment and decompose it from the positions where the pattern repeats start:

```
AACTGTT AACTGTT AACTGTT AA
AACTGTT AACTGT_ AACT_TT AA
```

We obtain the set of repeating units $\{X_1 = \textbf{AACTGTT}, X_2 = \textbf{AACTGT}, X_3 = \textbf{AACTTT}, X_4 = \textbf{AA}\}$. A final test is performed for each repeating unit to check whether the number of errors in that unit is less than or equal to $\theta_{max}(t_C)$ where t_C is the length of the consensus which is 7 in this case. If a unit fails this test then the part of the alignment after the position where the test fails is discarded. A similar test is also performed to check the similarity of adjacent repeating units according to the same threshold $\theta_{max}(t)$. Since each repeating unit passes these tests in this case, we have a repeat with copy number $3 + \frac{2}{7}$. The last step is to determine the consensus pattern by the majority rule among the alignments of repeating units with the pattern. If we align only the bottom lines of the alignments of the repeating units we obtain the consensus:

$$\begin{array}{l} \texttt{AACTGTT} \\ \texttt{AACTGT\_} \\ \texttt{AACT\_TT} \\ \texttt{AA} \end{array} \xrightarrow{\text{by the majority rule}} \texttt{AACTGTT}$$

Since the resulting consensus pattern is identical to the first repeating unit there's no need to do any extra work and the repeat is reported as is. However if the consensus pattern happens to be different than the first repeating the unit, then all the alignment that were mentioned before are computed again but this time between the sequence (including the first repeating unit) and the consensus pattern.

The length of the first repeating unit is 7 whereas the lengths of the second and the third are 6. Since the most common unit length is 6 in this example the period of the repeat is determined as 6. Notice that it is not identical to the consensus period which is 7.

3.4 Criteria

Various criteria are used during both the detection and verification phase of the algorithm. The thresholds based on these criteria were briefly introduced during the explanation of the algorithm but the calculations of these thresholds are left to this section. Here is a review of the uses of these thresholds:

$\Delta t_{max}(t)$**:** This threshold is mainly used to determine the range of values that can be accepted to a t-chain. The range is $[t - \Delta t_{max}(t), t + \Delta t_{max}(t)]$. The other use of this threshold is to set the width of the diagonal band when computing the alignments. If two strings of length t are to be computed, then only the diagonal of width $2\Delta t_{max}(t)$ is computed.

$del_{max}(d)$ **and** $ins_{max}(d)$**:** These thresholds are used to prevent some noise from being accepted to chains (Section 3.2.3).

$score_{min}(t)$**:** This is the threshold which qualifies a t-chain as candidate and thus triggers the verification phase (Section 3.2.4).

$size_{max}(t)$**:** This threshold is used to limit the size of a chain.

$\theta_{max}(t)$**:** This is the threshold which determines whether two strings of length t are similar enough to be considered as repeating units of a tandem repeat.

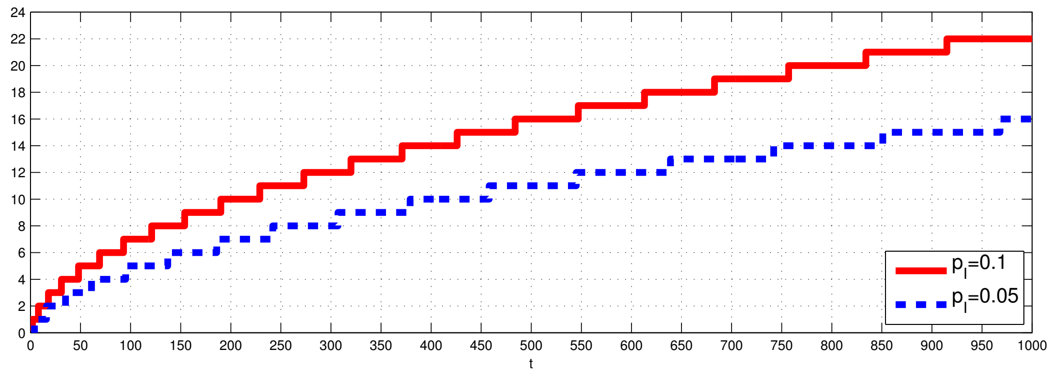


Figure 3.10: $\Delta t_{max}(t)$ for $p_I = 0.1$ and $p_I = 0.05$.

3.4.1 $\Delta t_{max}(t)$

When observing the occurrences of short substrings in a perfect tandem repeat with period t, it's expected to see these occurrences at distance t. However if insertions and deletions are allowed, these distances increase after each insertion and decrease after each deletion. Therefore the distances in some neighborhood of t are acceptable to the chains.

In the model of formation of the tandem repeats (Section 3.1.1) it is assumed that the probability of having an indel (insertions and deletions are equally likely) at each position is p_I. Let's consider a one-dimensional random walk of t steps where the probability of staying at the same node is $1 - p_I$, the probability of going left is $p_I/2$ and the probability of going right is $p_I/2$ at each step. Then it's known [15] that 95% of the time the maximum displacement from the start node is not greater than $2.3\sqrt{p_I \cdot t}$. Therefore the threshold $\Delta t_{max}(t)$ is set to $\lfloor 2.3\sqrt{p_I \cdot t} \rfloor$ and only the differences in the range $[t - \Delta t_{max}(t), t + \Delta t_{max}(t)]$ are accepted to a t-chain. Since only the differences in this range are allowed in a chain, it's reasonable to only fill the diagonal band of width $2\Delta t_{max}(t)$ when computing alignments of strings of length t. This idea of accepting only the distance range according to the same criterion and limiting the width of the alignment matrix is also used in Tandem Repeats Finder [6] and ATRHunter [45]. Figure 3.10 shows the $\Delta t_{max}(t)$ for $p_I = 0.1$ and $p_I = 0.05$.

3.4.2 $del_{max}(d)$ and $ins_{max}(d)$

The role of these two thresholds were explained in Section 3.2.3. Briefly they prevent the acceptance of distance values into the chains which can be considered noises (the values which cause high increase or decrease between two hits within an unexpectedly small distance).

To describe the calculation of these two thresholds let's first introduce the random variable X_t which is the number of insertions minus the number of deletions in a random sequence of length t. Again remember the assumption of formation of the tandem repeats which says that there's an insertion with probability $p_I/2$ and a deletion with probability $p_I/2$ at each position of the sequence (and no insertions or deletions with probability $1 - p_I$). Then it's obvious that

$$\begin{aligned} f_{X_t}(k) = \Pr(X_t = k) = (1-p_I)\Pr(X_{t-1}=k) & \qquad \text{no indel at position } t \\ + \frac{p_I}{2}\Pr(X_{t-1}=k-1) & \qquad \text{insertion at position } t \\ + \frac{p_I}{2}\Pr(X_{t-1}=k+1) & \qquad \text{deletion at position } t \end{aligned}$$

and

$$f_{X_0}(k) = \Pr(X_0 = k) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

Then the probability distribution of all X_t's for $t \leq n$ can be computed in $O(n^2)$ time using dynamic programming. Figure 3.11 and Figure 3.12 shows the distribution of the random variables X_{20} and X_{100} respectively. The random variables X_t's will be used to determine the thresholds $del_{max}(d)$ and $ins_{max}(d)$.

Let's first study the case where the number of deletions is larger than the number of insertions. Consider a chain where a hit value of h_2 occurs g position after a hit value h_1 such that $h_2 < h_1$ and all the values between these two are gaps:

$$\Gamma_{t,s} = \langle \quad \cdots \quad h_1 \quad \underbrace{0 \quad 0 \quad \cdots \quad 0}_{g-1 \text{ gaps}} \quad h_2 \quad \cdots \quad \rangle$$

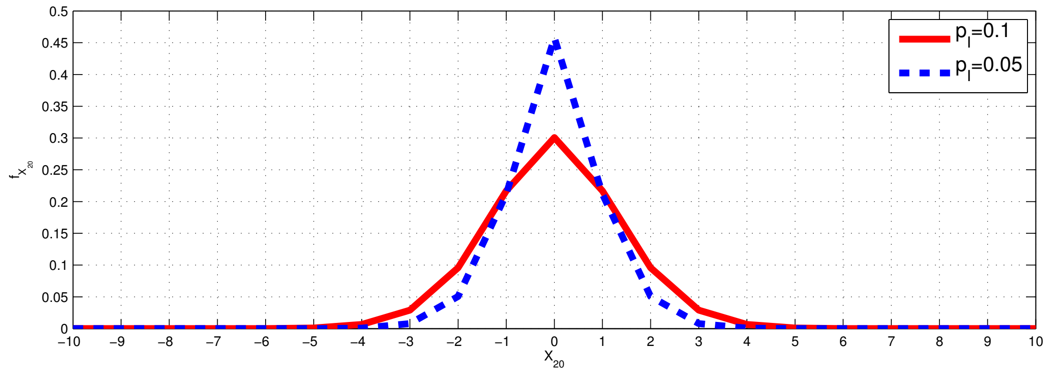


Figure 3.11: Probability mass function of X_{20} for $p_I = 0.1$ and $p_I = 0.05$.

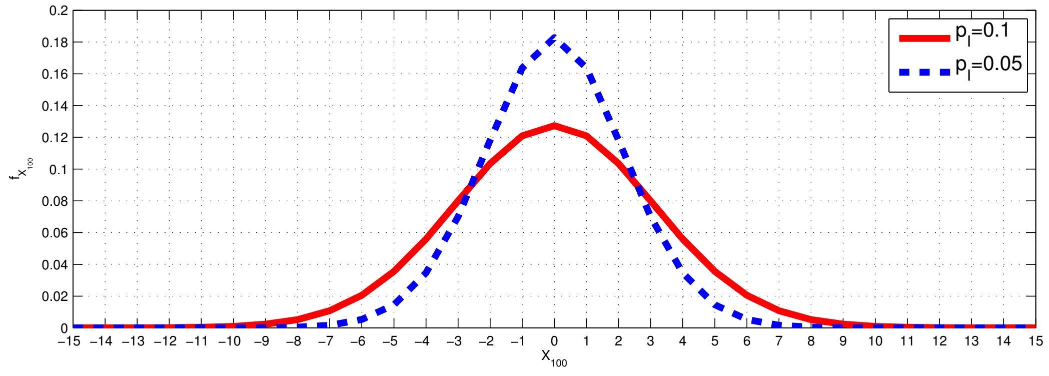


Figure 3.12: Probability mass function of X_{100} for $p_I = 0.1$ and $p_I = 0.05$.

This chain indicates that there are $d = h_1 - h_2$ deletions (or $d+i$ deletions and i insertions) in approximately $d + g$ positions. For instance, consider a repeating unit $X_1 =$ **. . . ACTGGCAT . . .** with length t and a second unit $X_2 =$ **. . . ACGAAT . . .** such that their alignment is $\genfrac{}{}{0pt}{}{\texttt{...ACTGGCAT...}}{\texttt{...AC\_GA\_AT...}}$. Then the chain of window size 2 corresponding to the unit X_2 would be:

$$\Gamma_{t,s} = \langle \ldots \quad t \quad 0 \quad 0 \quad 0 \quad t-2 \quad \ldots \rangle$$

The hit value t indicates that the substring **AC** has also occurred t positions before and the following hit value $t - 2$ indicates that the substring **AT** has also occurred $t - 2$ positions before. Note that $t - 2$ appears $g = 4$ positions after t in the chain. The gaps between these two hits hide the information about the alignment of the substring of length $g - w = 4 - 2 = 2$ which is between **AC** and **AT** in the second repeating unit X_2. The best we can say is that when we align this substring of length 2 with the corresponding substring of length $2 + d$ in the first repeating unit we expect a difference d between the number of deletions and number of insertions ($d = t - (t - 2) = 2$).

Based on this observation, the upper limit of the difference between two hits is the smallest d where the probability of the difference between the number of deletions minus the number of insertions being more than d in approximately $g + d$ characters is less than $\epsilon = 0.001$. The probability of this difference being more than d in p characters is simply:

$$\sum_{x<-d} \Pr(X_p = x)$$

where the random variable X_p, which was described before, is the number of insertions minus the number of deletions in a random sequence of length p. The "approximate length $g + d$" of the sequence where the deletions and insertions occur is set to $g + d - w + 2$ empirically.

Thus the threshold $del_{max}(g)$ is set to the largest negative θ (since the second hit is smaller than the first one) satisfying

$$\sum_{x<\theta} \Pr(X_{g-x-w+2} = x) < 0.001$$

The method followed in the case where the number of insertions is larger than the number of deletions is similar. Consider a chain where a hit value of h_2 occurs g position after a hit value h_1 such that $h_2 > h_1$ and all the values between these two are gaps:

$$\Gamma_{t,s} = \langle \quad \cdots \quad h_1 \quad \underbrace{0 \quad 0 \quad \cdots \quad 0}_{g-1 \text{ gaps}} \quad h_2 \quad \cdots \quad \rangle$$

This chain indicates that there are $i = h_2 - h_1$ insertions (or $i + d$ insertions and d deletions) in approximately g positions.

Then the upper limit of the difference between these two hits is the smallest i where the probability of the difference between the number of insertions minus the number of deletions being more than i in approximately g characters is less than $\epsilon = 0.001$. The probability of this difference being more than i in p characters is simply:

$$\sum_{x>i} \Pr(X_p = x)$$

The "approximate length g" of the sequence where the insertions and deletions occur is set to $g - w + 2$ empirically.

Thus the threshold $ins_{max}(g)$ is set to the smallest positive θ satisfying

$$\sum_{x>\theta} \Pr(X_{g-w+2} = x) < 0.001$$

Figure 3.13 shows the thresholds $del_{max}(g)$ and $ins_{max}(g)$ for $p_I = 0.1$ and $p_I = 0.05$.

3.4.3 $score_{min}(t)$

During the detection phase, only the chains containing a significant number of hit values are considered as the evidence of a tandem repeat. The threshold of the number

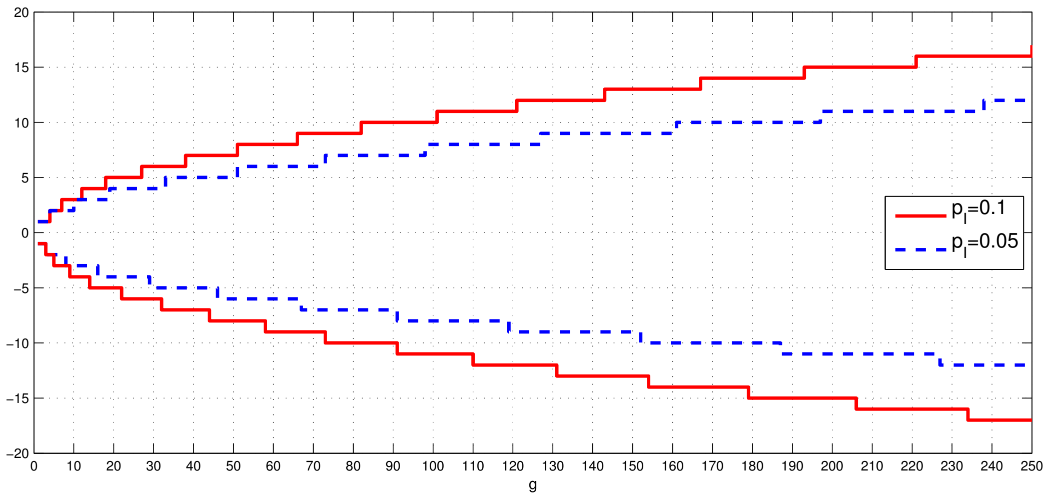


Figure 3.13: $del_{max}(g)$ and $ins_{max}(g)$ for $w = 4$.

of hits is $score_{min}(t)$ for a t-chain.

Determination of $score_{min}(t)$ is very crucial both for the sensitivity and runtime of the algorithm. Low values for $score_{min}(t)$ will qualify many chains as candidates whereas they cannot be classified as tandem repeats. These chains are called *false alarms* and the verification phase fails for these false alarms. On the other hand, setting high values for $score_{min}(t)$ may cause some valid candidate chains to be missed.

It was mentioned before (Section 3.2.1) that for a perfect tandem repeat $Y = XX$ with period t, only the substrings of X which occurs exactly once in X are observable in a t-chain. We call these substrings *matches* because they match with the corresponding substrings t positions before. However a substitution at a position i causes the substrings starting at positions $i - w + 1, i - w + 2 \ldots i$ not to match anymore with the substrings t positions before. These substrings (windows) are called the *blocked-out windows* by the substitution. Similarly an insertion or a deletion may also block out up to w windows. Insertions and deletions also shift the distance t up or down. Since the values in a range $[t - \Delta t_{max}(t), t + \Delta t_{max}(t)]$ are accepted in a t-chain, the shift effect of insertions and deletions does not concern the score of the chain. However since the block-out windows will not match the substrings at distances around t, they will not be detected in the chains. We will generalize all the substitutions, deletions and insertions as *edit events* since only the block-out effect of these mutations are here of interest.

The generation of block-out windows by edit events suppresses some of the matches within their coverage. We wish to analyze this phenomenon. However a detailed formal analysis has a complexity that does not completely justify the effort. More important is the elucidation of the mechanism, since actual operational data can be obtained through simulations.

Edit events take place at each character with probability $p_E = 1 - p_M$ and they are independent from each other. Each edit event blocks out w matches; however, these block-outs may overlap. If an edit event occurs within the block-out of its predecessor, e.g., at the i^{th} position of it ($i = 2, 3, \ldots, w$), then the block-out of the latter is effectively shortened to length $i - 1$. However the shortening effect is not independent of the actual number of edit events. The smaller is the number of events, the larger is the average length of the block-out stretch of each event (since two consecutive events are less likely to interfere). In any case the edit events can be modeled as a binary Bernoulli sequence e where each element is 1 with probability p_E. Then the binary sequence $stretch(e)$ is simply the representation of block-outs corresponding to the edit events e. In other words i^{th} element of $stretch(e)$ is 1 if and only if i^{th} or $(i+1)^{th}$ $\ldots$ or $(i+w-1)^{th}$ element of e is 1. Let's represent the windows which occur only once in a sequence X as a binary sequence m where i^{th} element of m is 1 if and only if the substring $X_{i,i+w-1}$ occurs only once in X. Then the observable matches after the edit events are:

$$obs(m) = m \wedge \overline{strech(e)}$$

The number of observable matches is then $O = weight(obs(m))$. If we knew the probability distribution of O we would set the threshold $score_{min}(t)$ to the largest θ

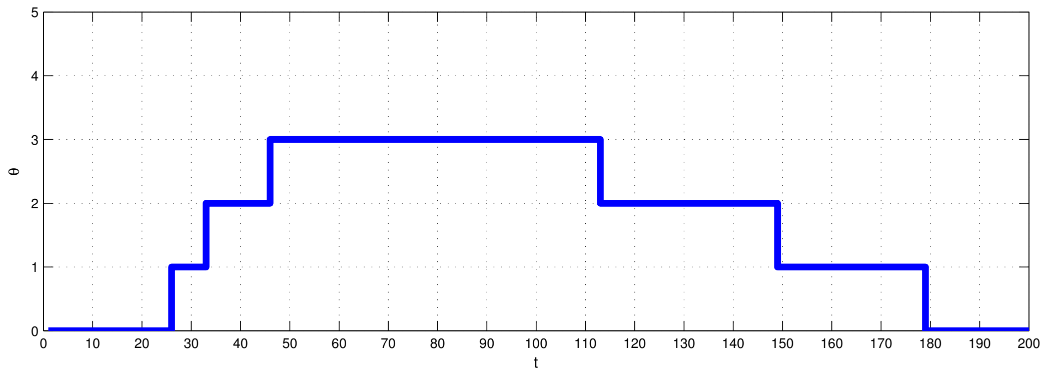


Figure 3.14: θ for $w = 3$.

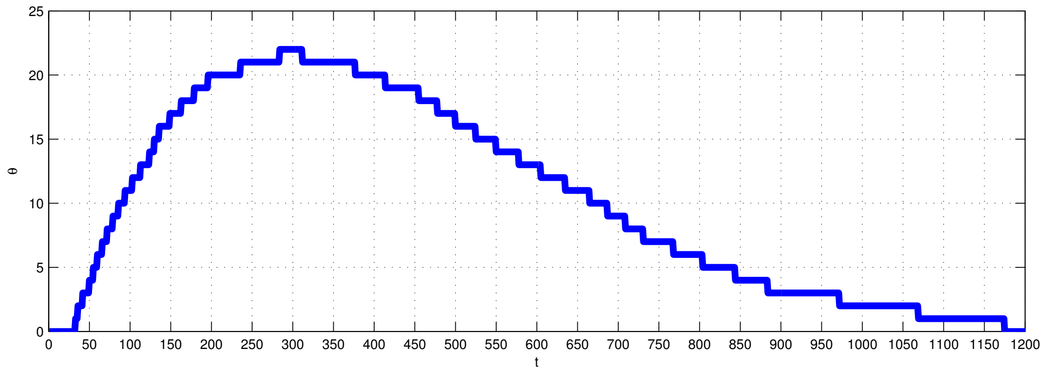


Figure 3.15: θ for $w = 4$.

such that

$$\sum_{x<\theta} \Pr(O = x) < \epsilon$$

where ϵ is a small constant (0.01).

Unfortunately, the analytical form of $\Pr(O = x)$ is unknown and presumably not worth pursuing. Instead, a suitable approximation will be obtained by simulation. For each sequence length t and window size w we generate a large sample of 4-valued strings of length t. Then for each string in the sample we create a sequence e of edit events and compute the value

$$O = weight\left(m \wedge \overline{strech(e)}\right)$$

and record them in a histogram. Then this histogram is finally used to evaluate θ for $\epsilon = 0.01$. The values of θ are plotted in Figures 3.14, 3.15, 3.16 and 3.17.

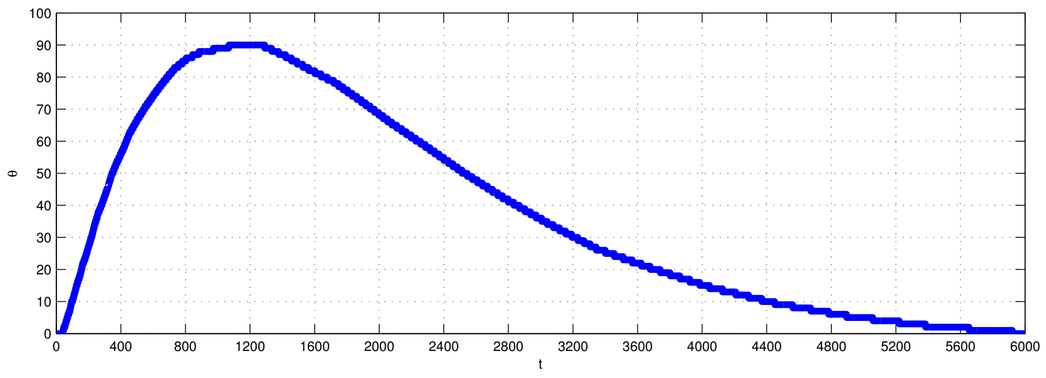


Figure 3.16: θ for $w = 5$.

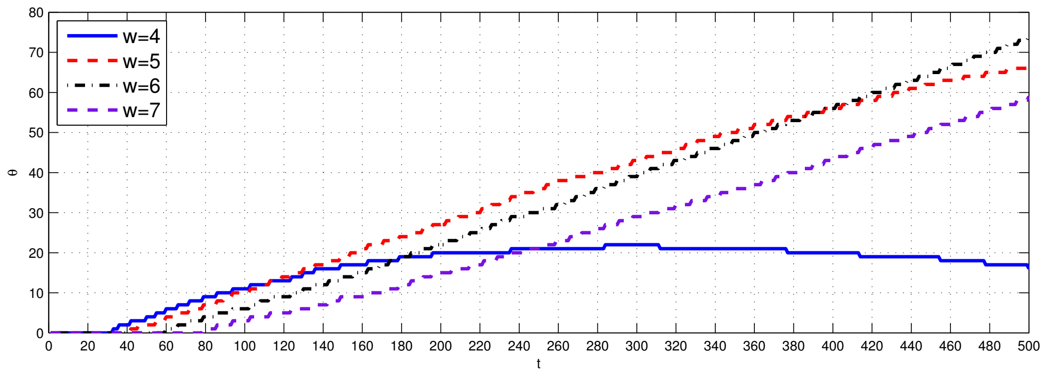


Figure 3.17: θ's for $w = 4, 5, 6, 7$ and $t \leq 500$.

Remember the random variable $U_{L,w}$ which was the number of unique substrings of length w in a random string of length L (Section 3.2.1). As it can be anticipated from the high ratio of variance over the expectation of the random variable $U_{L,w}$ (for $w = 1, 2, 3$), the θ's are all zero for window sizes 1 and 2 and they are very small for window size 3 (Figure 3.14). Similarly θ's begin to increase after t greater than 40s and 50s for larger window sizes because of the high variance of $U_{L,w}$ for low L's.

Let's now consider the case where the error rate p_E is 0.2. Assume that we're scanning the sequence with window sizes $4, 5$ and 6 and we're interested in repeats with period range of $[50, 200]$. We claim that setting the thresholds $score_{min}(t)$'s according to the θ's (Figure 3.17) will significantly reduce the false alarms. A null hypothesis would be, on a random sequence the expected number of false alarms $\mathrm{E}[F_1]$ when all the $score_{min}(t)$'s are identical to 1 is equal to the expected number of false alarms $\mathrm{E}[F_\theta]$ when we set the $score_{min}(t)$'s as the θ's. Then the alternative hypothesis would be that $\mathrm{E}[F_1]$ will be much bigger than $\mathrm{E}[F_\theta]$. To support the alternative hypothesis we created several random sequences of length 100000 and observed the number of false alarms for both cases. The observed results $\mathrm{E}[F_1] \approx 50000$ and $\mathrm{E}[F_\theta] \approx 700$ shows that setting the thresholds according to θ's significantly reduces the false alarms.

In addition we also claim that we do not miss a significant ratio of tandem repeats when we set the thresholds according to the θ's. A null hypothesis would be that the expected number of detected repeats would be significantly lower than the actual number of repeats in a sequence. Again several random sequences are generated including repeats with period in the range $[55, 195]$ and 99% of the generated repeats are detected by the chains where the thresholds $score_{min}(t)$'s are set to θ's. These results show that the thresholds are very effective in the specified period range.

However as the Figure 3.14 shows, the values of θ for window size $w = 3$ are too low to be considered as thresholds. Therefore the thresholds $score_{min}(t)$'s for $t < 50$

Window size	3				4			
$score_{min}(t)$	1	2	3	4	1	2	3	4
Detected repeats	844	784	714	545	833	759	685	529
False alarms	1835769	1161812	657794	383646	663072	291315	141741	87695
Ratio	0.00045	0.00067	0.00108	0.00142	0.00125	0.00260	0.00483	0.00603

Table 3.5: Detected repeats and false alarms for period up to 32 in E. coli O157:H7.

are set to 3 and only the window size 4 is used for periods that are smaller than 32 (determined empirically to optimize the ratio of detected repeats and running time). To determine this threshold and window size, the combinations of window sizes 3 and 4, and the thresholds $1, 2, 3$ and 4 are tried to detect tandem repeats with period up to 32 in E. coli O157:H7 genome (5498450 base pairs). Table 3.5 shows the number of tandem repeats detected and the false alarms for these combinations. The ratio in the table is simply the number of detected repeats over false alarms. The window size 4 with a threshold of 3 is chosen as the optimal pair for repeats with period up to 32 according to the table since it catches most of the repeats without producing many false alarms compared to the others.

For the period range between 33 and 50, we simply scanned the same sequence with several combinations of window sizes and $score_{min}(t)$'s (Table 3.6). The window size 5 with a threshold of 3 is chosen as the optimal pair for the period range between 33 and 50.

For repeats with periods that are greater than 50 we mentioned before that setting the $score_{min}(t)$'s according to the θ's catches almost all of the repeats with negligible false alarms. We empirically observed that using the window size 5 for periods up to 250 and window size 7 for periods larger than 250 does a good job in detecting the

Window size	4		5	
$score_{min}(t)$	3	4	3	4
Detected repeats	56	52	50	42
False alarms	104560	48542	18317	6827
Ratio	0.00053	0.00107	0.00272	0.006152

Table 3.6: Detected repeats and false alarms for period between 33 and 50 in E. coli O157:H7.

repeats with a very small number of false alarms.

To summarize, for detecting the repeats with period up to 32, window size 4 is used. For the period range between 33 and 250 we use the window size 5 and for the periods that are greater than 250 we use the window size 7. The thresholds ($score_{min}(t)$) for each of them is set to the θ's in the Figure 3.17. If θ is less than 3, then the thresholds is set to 3.

3.4.4 $size_{max}(t)$

Since the purpose of constructing a chain $\Gamma_{t,s}$ is to provide evidence that the substring of length t of S starting at position s is similar to the previous substring t positions before it, we do not permit a chain to be longer than t. Therefore we set the threshold $size_{max}(t)$ to t.

However, since we are interested in tandem repeats which has a score of at least θ_{score} when aligned with a consensus pattern, the repeats with small period have to have a copy number greater than 2 in order satisfy this score criterion. For instance if the the score of a match is 2 in the score function and we are interested in repeats with a minimum score of 50, then a repeat with period 3 has to have a copy number of at least $\frac{50}{2 \cdot 3} = 8.3$. Then we allow the chains for $t = 3$ to grow to the size of at most $(8.3 - 1) \cdot 3 = 47$.

3.4.5 $\theta_{max}(t)$

This threshold determines whether two strings of length (approximately) t are similar enough to be considered as repeating units of a tandem repeat. For the multiple tandem repeats, according to our definition each repeating unit should be similar to the consensus and each pair of adjacent units should be similar to each other according to this threshold.

Assume that two repeating units (or a unit and the consensus pattern) is aligned.

Then the probability of observing a match is p_M and the probability of observing a substitution, a deletion or an insertion is $1 - p_M$ for each position in the alignment according to the assumed error model. Now let's define the random variable E_t as the total number of substitutions and indels in an alignment of size t according to this error model. Then E_t is a binomial random variable with parameters t and $1 - p_M$. We know that the expectation $\mu_{E_t} = \mathrm{E}[E_t]$ is $t(1 - p_M)$ and the standard deviation σ_{E_t} is $\sqrt{t(1 - p_M)p_M}$.

Then we allow a number of errors of at most $\lfloor \mu_{E_t} + \sigma_{E_t} \rfloor$ for an alignment of size t. All the comparisons in the next chapter are performed according to this setting, since most of the tandem repeats that are reported by Tandem Repeats Finder and ATRHunter are observed to be within this similarity range. However there are some exceptions like having 28 errors in an alignment of size 99 (reported by Tandem Repeats Finder in the sequence of E. coli O157:H7 starting at position 923619) and having 11 errors in alignment of size 36 (reported by ATRHunter in the same sequence starting at position 923830) which exceeds this threshold that our algorithm uses, but these are only a few exceptions that are observed in the outputs of these two algorithms. The user can adjust this threshold if desired, for a more lenient definition of tandem repeats.

Chapter 4

Results

To test the quality of the algorithm, the results on both natural and synthetic data are compared with the results of Tandem Repeats Finder [6] and ATRHunter [45]. The quality is measured both by the total number of repeats found and the running time. The natural data include the yeast chromosome I (230203 base pairs) and the complete genome of E. coli O157:H7 (5498450 base pairs).

4.1 Input and Output

All the sequences are scanned with the following input parameters for all the three algorithms:

$p_M = 0.8$ **and** $p_I = 0.1$**:** An error model of 80% expected matches and 10% of expected indels is chosen since it is the default model that was used in the original papers of Tandem Repeats Finder [6] and ATRHunter [45]. Also it's not possible to change this error model for ATRHunter.

Score function $(2, 5, 7)$**:** Again these parameters for score function are the default in the papers for comparison with each other.

Score threshold 50**:** All the searches in the original papers are performed with repeats having at least a score of 50 when aligned with the periodic repetition of the consensus pattern,

Maximum period 500**:** ATRHunter does not allow to search for repeats with period larger than 500. Tandem Repeats Finder can search for periods up to 2000 and our algorithm is virtually capable of searching for any period length. Again the maximum period of 500 is the default in the papers mentioned above.

4.1.1 ATRHunter

The information about the detected repeats on the output on all the three programs is similar. For each detected repeat, ATRHunter displays the following information:

Starting Position: This is simply the position where the repeat starts. Position 0 denotes the first nucleotide in the sequence.

Motif Length: The period of the repeat.

Number of units: This is simply the copy number. Repeats having a minimum copy number of 1.9 are allowed instead of a minimum copy number of 2.

Score: This is the score of the optimal alignment between the repeat and a periodic repetition of the consensus pattern.

For all the repeats that are detected, ATRHunter also prints the alignment of each repeating unit with the consensus pattern.

4.1.2 Tandem Repeats Finder

Tandem Repeats Finder outputs more detailed information about the repeats:

Indices: These are the start and end positions of the repeats. The first nucleotide in the sequence is the one at position 1.

Period Size: This *period size* is the most common length among the length of the repeating units and it may differ from the length of the consensus pattern. The same definition of *period size* is used in our algorithm.

Copy number: Again repeats having a copy number of at least 1.9 are detected.

Consensus Size: The length of the consensus pattern.

Percent Matches: This is the percentage of the total matches in the alignments of adjacent repeating units. If m is the total number of matches, s is the total number of mismatches, and i is the total number of indels in all the $c-1$ alignments of adjacent copies (c is the copy number), then percent matches is simply $\frac{m}{m+s+i}100$.

Percent Indels: This is the percentage of the total indels in the alignments of adjacent repeating units. It is simply $\frac{i}{m+s+i}100$.

Score: The score of the optimal alignment between the repeat and a periodic repetition of the consensus pattern.

Besides the alignments of each repeating unit with the consensus pattern, Tandem Repeats Finder also prints some statistics like the entropy of the repeat and count of **A**'s, **C**'s, **G**'s and **T**'s.

4.1.3 Our Algorithm

The output of our algorithm is similar to the output of Tandem Repeats Finder:

Start Position: The position where the repeat starts. The first nucleotide in the sequence is the one at position 1.

End Position: The position where the repeat ends.

Period Size: The same definition of period size of Tandem Repeats Finder is used.

Copy number: Again repeats having a copy number of at least 1.9 are detected.

Consensus Size: The length of the consensus pattern.

Percent Matches: The percentage of the total matches in the alignments of adjacent repeating units (same information with Tandem Repeats Finder). If m is the total number of matches, s is the total number of mismatches, and i is the total number of indels in all the $c - 1$ alignments of adjacent copies (c is the copy number), then percent matches is simply $\frac{m}{m+s+i}100$.

Percent Indels: The percentage of the total indels in the alignments of adjacent repeating units. It is simply $\frac{i}{m+s+i}100$.

Score: The score of the optimal alignment between the repeat and a periodic repetition of the consensus pattern.

Percent Consensus Errors: This is the percentage of the total number of errors in the alignmet of the whole repeat with the periodic repetition of the consensus pattern. Remember that this alignment is totally different than the alignments of the adjacent copies. If m_c is the total number of matches, s_c is the total number of mismatches, and i_c is the total number of indels in the optimal global alignment of the tandem repeat with the c times repeated consensus pattern (c is the copy number), then percent consensus errors is simply $\frac{s_c+i_c}{m_c+s_c+i_c}100$.

In addition to the alignments of the each repeating unit with the consensus pattern, the alignments of adjacent units are also printed in the output.

4.1.4 Redundancy in the output

In the output of all the three programs, a particular tandem repeat may be reported multiple times. For instance a repeat of a period 15 and copy number 6 can also be reported as a repeat with period 30 and copy number 3 or as a repeat with period 45 and copy number 2 starting at the same or close by positions. For instance the output of Tandem Repeats Finder while scanning the sequence E. coli O157:H7 includes:

Indices	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score
198–254	12	4.8	12	86	0	79
207–253	6	7.8	6	82	0	59
207–254	18	2.7	18	76	0	68

These three reported repeats are actually three different interpretations of the same repeat. A similar example from the output of the ATRHunter for the same sequence is:

Starting position	Motif Length	Number of units	Score
197	12	4.8	79
203	18	2.8	60
1066517	39	6.9	358
1066544	21	4.7	63
1786528	324	1.9	916
1786617	162	3.3	947

Notice that in the second and the third group of repeats, the start positions are not very close but the repeats overlap. Our algorithm may also report a similar kind of redundancies such as:

Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
198	254	12	4.75	12	86.7	0	79	8.8
208	255	6	8.17	6	81.4	2.3	54	12.2
411014	411759	94	8.02	95	92.3	1.7	1163	6.2
411071	411748	282	2.41	282	96.7	0.3	1265	1.9

Unlike to the cases above, Tandem Repeats Finder rarely includes another kind of interesting redundancy in the output such as (for instance on the sequence E. coli O157:H7):

Indices	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score
1757508–1757542	11	3	11	84	15	52
1757508–1757542	13	2.9	12	88	12	54
1618166–1618202	12	3	12	84	7	58
1618165–1618201	13	2.9	13	88	4	60

In this case a repeat is reported more than once with very similar consensus patterns which are not multiples of each other.

These kind of multiple reports are eliminated by hand as much as possible and are not included in the outputs that are presented in this thesis.

4.2 Natural Data

Chromosome I of yeast was analyzed by Tandem Repats Finder in the original paper by Benson [6]. Also ATRHunter used the same sequence to compare itself with Tandem Repeats Finder in the paper [45]. However since the length of the yeast chromosome I is short (230203 base pairs), the complete genome of E. coli O157:H7 (5498450 base pairs) was also used to compare the two algorithms in the same paper. It was reported [45] that ATRHunter outperformed Tandem Repeats Finder (Version 2.02) in the measures of both the number of detected repeats and the time spent per repeat. The comparisons in this thesis are performed between the latest version Tandem Repeats Finder (version 4.00 which is significantly improved over the older version), the latest version of ATRHunter and our algorithm. We used the same sequences, yeast chromosome I and E. coli O157:H7 and for both of these real world sequences our algorithm significantly outperforms the others in the measure of both the detected total number of repeats and the running time.

4.2.1 Yeast Chromosome I

Table 4.1 shows the reported number of repeats, the number of repeats after eliminating the redundancies, and the running times for all the three algorithms on the sequence of yeast chromosome I with the parameters mentioned above (repeats with period up to 500 and having a consensus score of at least 50 according to the score function of $(2, 5, 7)$ under an error model of $(p_M = 0.8, p_I = 0.1)$).

	Tandem Repeats Finder	ATRHunter	Our Algorithm
# of repeats reported	56	62	108
# of repeats after elim.	48	54	92
Running time (seconds)	1.00	8.00	0.79

Table 4.1: Comparison of algorithms on yeast chromosome I.

The repeats that are reported by our algorithm after eliminating the redundancies are listed in Table A.1. The repeats which are also detected by Tandem Repeats Finder are marked with "#". Similarly the repeats which are also detected by ATRHunter are marked with "+" in the table.

Comparison with Tandem Repeats Finder

While the running times of our algorithm and Tandem Repeats Finder is similar for yeast chromosome I, our algorithm is approximately two times faster per reported repeat. Among the 48 repeats that are detected by Tandem Repeats Finder, our algorithm only misses 5. Those missed tandem repeats are listed in Table 4.2 (repeats marked with "+" are also detected by ATRHunter).

Let's inspect the first one of these repeats, the one between the indices 23709-23759. This repeat has a consensus of **AAATAAAAA**. As only the unique substrings are observable according to our algorithm, missing this repeat is predictable since the consensus pattern is almost a repetition of a single character.

Another observation about the repeats missed by our algorithm is that all of them have scores which are very close to the θ_{score} which was set to 50. Although most of

	Indices	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score
+	23709–23759	9	5.8	9	79	4	60
+	31118–31147	2	15	2	92	0	53
+	93834–93884	15	3.4	15	75	0	60
	112740–112791	12	4.6	12	74	13	55
+	116146–116196	21	2.4	21	73	0	53

Table 4.2: Unique tandem repeats found by Tandem Repeats Finder in yeast chromosome I.

Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
222559	222776	108	2.02	108	82	1.8	294	9.1
206990	207086	48	2.02	48	75.5	0	110	12.4
24919	25012	47	1.94	48	78.7	6.4	114	10.5
87687	87726	20	1.95	21	80	5	52	9.8
133378	133418	20	2	20	81	4.8	52	9.8
151499	151534	20	1.9	20	83.3	11.1	51	7.9
217349	217388	20	2	20	80	0	52	10
18042	18081	19	2.16	19	81.8	4.5	52	9.8
129282	129322	19	2.16	19	83.3	16.7	50	9.3
202638	202670	18	1.94	17	88.2	11.8	50	5.9

Table 4.3: Some repeats missed by Tandem Repeats Finder and ATRHunter in yeast chromosome I.

these missed repeats are detected by the chains as candidates, the alignments could not succeed with a score over the θ_{score} since even one position shift in the start of the repeat may introduce one or two substitutions or indels which may reduce the score significantly.

On the other hand, among the 92 repeats detected by our algorithm, Tandem Repeats Finder catches only 43 of those (marked with "#" in Table A.1). Table 4.3 lists just 10 of the missed repeats (these are also missed by ATRHunter) sorted according to the periods.

Let's inspect two of these repeats which are missed by Tandem Repeats Finder (also missed by ATRHunter):

222559-222776 This repeat has a period of 108 and the alignments of the two repeating units has only 20 errors (18 substitutions, 1 insertion and 1 deletion) which is less than 20%. Each repeating unit has 10 errors when aligned with the consensus pattern.

24919-25012 The repeat between these indices has a consensus pattern of length 48 and the alignment of the two repeating units has 10 errors (7 substitutions and 3 insertions) where it's legitimate for a 20% error model on a repeat with consensus period 48. Also the alignments of each of the two units with the consensus pattern have only 5 errors.

	Starting position	Motif Length	Number of units	Score
#	23713	1	46	50
#	31117	2	15	53
	48199	15	2.6	50
#	93828	29	2.3	60
	110691	13	3.5	53
#	116145	21	1.9	52
	169202	20	2.1	56

Table 4.4: Unique tandem repeats found by ATRHunter in yeast chromosome I.

If all the 49 repeats missed by Tandem Repeats Finder are inspected carefully, it can be seen that all of them are valid repeats within the assumed error model. Also our algorithm detected several unique repeats which score significantly higher than $\theta_{score} = 50$.

Comparison with ATRHunter

Among the 54 repeats that are detected by ATRHunter, our algorithm only misses 7 while being much faster. Those missed tandem repeats are listed in Table 4.4 (repeats marked with "#" are also detected by Tandem Repeats Finder).

Similar to the unique repeats found by Tandem Repeats Finder, these unique repeats by ATRHunter have score very close to the θ_{score}.

Again, let's inspect one of these repeats missed by our algorithm. The one starting at position 110691 has a consensus **AAAAGAGAGAAAA** with low entropy which is not easily detectable by our algorithm.

We'll look at one more missed repeat, the one starting at position 93828. The alignment of the first repeating unit with the second one has 9 errors (5 substitutions and 4 indels) in 32 positions which is out of the similarity thresholds that our algorithm uses. The alignment of the consensus pattern and the first unit has also 9 errors for this repeat. Remember that we allowed at most $\lfloor \mu_{E_t} + \sigma_{E_t} \rfloor$ errors when setting the threshold $\theta_{max}(t)$. The threshold $\theta_{max}(t)$ for $t = 32$ is 8, thus these two units are not similar enough to be considered as a tandem repeat according to our algorithm.

If we employ a threshold of $\lfloor \mu_{E_t} + 2\sigma_{E_t} \rfloor$ for $\theta_{max}(t)$, the running time increases from 0.79 to 0.98 seconds where the number of reported repeats increase from 108 to 177. However all the results mentioned in these comparisons are for the case where the $\theta_{max}(t)$ is set to $\lfloor \mu_{E_t} + \sigma_{E_t} \rfloor$ as described in Section 3.4.5 (it can be adjusted manually if desired).

ATRHunter reports only 47 repeats (marked with "+" in Table A.1), among the 92 repeats detected by our algorithm. Table 4.3 lists some of the missed repeats by ATRHunter. All the 45 missed repeats are legitimate within the assumed error model and some of them have scores significantly higher than $\theta_{score} = 50$.

4.2.2 Complete Genome of E. Coli O157:H7

Table 4.5 shows the reported number of repeats, the number of repeats after eliminating the redundancies, and the running times for all the three algorithms on the sequence of E. coli O157:H7 with the parameters mentioned above (repeats with period up to 500 and having a consensus score of at least 50 according to the score function of $(2, 5, 7)$ under an error model of $(p_M = 0.8, p_I = 0.1)$).

The repeats that are reported by our algorithm after eliminating the redundancies are listed in Table A.2. The repeats which are also detected by Tandem Repeats Finder are marked with "#". Similarly the repeats which are also detected by ATRHunter are marked with "+" in the table.

	Tandem Repeats Finder	ATRHunter	Our Algorithm
# of repeats reported	222	236	791
# of repeats after elim.	190	227	751
Running time (seconds)	13.68	175.22	13.10

Table 4.5: Comparison of algorithms on E. coli O157:H7.

	Indices	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score
+	169132–169179	18	2.7	18	76	0	54
	923619–923833	99	2.2	99	71	0	227
	2143394–2143445	21	2.5	21	71	3	55
	2412879–2412910	16	2.1	16	88	5	50
	4360094–4360138	13	3.4	13	78	6	53
+	4399204–4399246	18	2.4	18	80	0	51
	4510580–4510630	22	2.4	22	66	3	53
+	4600399–4600427	15	1.9	15	92	0	51
+	4952191–4952240	25	2	25	80	7	63

Table 4.6: Unique tandem repeats found by Tandem Repeats Finder in E. coli O157:H7.

Comparison with Tandem Repeats Finder

While the running times of our algorithm and Tandem Repeats Finder is similar for E. coli O157:H7, our algorithm is almost four times faster per detected repeat. Among the 190 repeats that are detected by Tandem Repeats Finder, our algorithm only misses 9. Those missed tandem repeats are listed in Table 4.6 (repeats marked with "+" are also detected by ATRHunter).

Let's inspect the repeat between indices 923619-923833 with a period of 99. If we align the two repeating units of this repeat, we observe 28 substitutions which is over the limit of our assumption of number of errors. Remember that we allowed at most $\lfloor \mu_{E_t} + \sigma_{E_t} \rfloor$ errors when setting the threshold $\theta_{max}(t)$. The threshold $\theta_{max}(t)$ for $t = 99$ is 23, thus these two units are not similar enough to be considered as a tandem repeat according to our algorithm.

If we employ a threshold of $\lfloor \mu_{E_t} + 2\sigma_{E_t} \rfloor$ for $\theta_{max}(t)$, the running time increases from 13.7 to 17.5 seconds where the number of reported repeats increase from 791 to 2129 (almost 10 times the number of repeats reported by the other two algorithms). Then the repeat mentioned above is also detected by our algorithm. However all the results mentioned in these comparisons are for the case where the $\theta_{max}(t)$ is set to $\lfloor \mu_{E_t} + \sigma_{E_t} \rfloor$ as described in Section 3.4.5 (it can be adjusted manually if desired).

Again all the other repeats missed by our algorithm have scores very close to the

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	3793916	3794702	393	2.01	394	81.7	5.2	1045	9.1
+	1732109	1732677	297	1.92	298	81.4	4.3	766	9
+	4520601	4521111	255	2	256	80.2	0.8	665	9.9
+	865062	865476	208	2.01	208	90.5	2.4	688	4.8
+	3541923	3542285	182	2.01	181	88.2	3.8	566	6
	2971505	2971739	116	2.03	117	78.3	1.7	288	11
	3244819	3245036	115	1.9	115	87.6	2.9	343	5.9
	4009295	4009525	113	2.02	114	79.2	5.8	286	10.3
	3823291	3823507	113	1.94	112	78.7	5.6	267	10.5
+	3890931	3891162	111	2.07	112	83.1	6.5	309	8.9

Table 4.7: Some repeats missed by Tandem Repeats Finder in E. coli O157:H7.

θ_{score}.

Among the 751 repeats detected by our algorithm, Tandem Repeats Finder only detects 181 of those (marked with "#" in Table A.2). Table 4.7 lists just 10 of the missed repeats by Tandem Repeats Finder sorted according to the periods.

Again all of the 570 repeats missed by Tandem Repeats Finder are legitimate within the assumed 20% percent error model and a significant number of them have very high scores.

Comparison with ATRHunter

Our algorithm misses 34 of the repeats among the 227 repeats that are detected by ATRHunter. Those missed tandem repeats are listed in Table 4.8 (repeats marked with "#" are also detected by Tandem Repeats Finder).

If we inspect the repeat starting at position 923830 with period 36, we see that there are 11 substitutions both in the alignment of the second repeating unit with the consensus and in the alignment of the first unit with the second which is over our threshold.

A run of our algorithm with a threshold $\theta_{max}(t)$ of $\lfloor \mu_{E_t} + 2\sigma_{E_t} \rfloor$, detects this repeat as well as 7 more of the missed repeats that are listed in Table 4.8.

Again all the other repeats missed by our algorithm have scores very close to the θ_{score}.

	Starting position	Motif Length	Number of units	Score
#	169131	18	2.7	54
	246640	25	1.9	50
	296649	21	1.9	52
	314050	27	1.9	55
	379277	23	2.1	56
	607646	24	1.9	60
	923830	36	2.2	79
	1026837	20	1.9	51
	1168004	29	2	61
	1428498	22	2	55
	1500714	24	1.9	50
	1546474	20	2.4	55
	1562565	27	2	66
	1781039	27	2	66
	1948337	27	2	66
	2101534	19	2.1	50
	2419335	18	1.9	54
	2425067	21	1.9	52
	2639069	23	1.9	58
	2647093	21	1.9	50
	2795584	17	1.9	50
	3372687	23	2	53
	3781067	20	1.9	53
	3801087	21	2	51
	3953147	15	2.7	54
	4299717	20	1.9	53
#	4399203	18	2.4	51
	4521076	21	2	56
#	4600396	15	2.3	54
	4762541	21	2	54
#	4952192	25	1.9	52
	5027738	22	1.9	52
	5030604	24	1.9	50
	5209777	21	2.2	50

Table 4.8: Unique tandem repeats found by ATRHunter in E. coli O157:H7.

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	3793916	3794702	393	2.01	394	81.7	5.2	1045	9.1
#	4520856	4521322	234	2	233	78	1.7	566	11.1
#	3249429	3249780	119	2.92	120	81.4	4.6	416	11.3
	2971505	2971739	116	2.03	117	78.3	1.7	288	11
	3244819	3245036	115	1.9	115	87.6	2.9	343	5.9
	4009295	4009525	113	2.02	114	79.2	5.8	286	10.3
	3823291	3823507	113	1.94	112	78.7	5.6	267	10.5
	776587	776808	113	2.03	110	81.9	6	291	9.3
#	5472324	5472519	101	1.94	101	88.4	0	315	5.6
	5180652	5180846	94	2.02	95	83.5	8.7	266	8.1

Table 4.9: Some repeats missed by ATRHunter in E. coli O157:H7.

ATRHunter reports 193 repeats (marked with "+" in Table A.2) among the 751 repeats detected by our algorithm. Table 4.9 lists 10 of the missed repeats by ATRHunter sorted according to the periods.

Again all of the 558 repeats that are missed by ATRHunter are legitimate within the assumed 20% percent error model and a significant number of them have scores much higher than θ_{score}.

These results show that our algorithm detects significantly more (between $2\times$ and $3.3\times$) tandem repeats that are legitimate within the assumed error model than the Tandem Repeats Finder and ATRHunter in less time. One may suspect that the repeats missed by Tandem Repeats Finder and ATRHunter contain too many errors according to their definition of tandem repeats, however if we inspect the outputs of theirs, we see the number of errors in these missed repeats are no more than the number of errors in the repeats that they report with similar periods (also for the number of indels).

In addition, we see that the repeats that could not be detected by our algorithm are missed because of the very critical range of the score of those repeats around the threshold θ_{score}. This does not mean that the other algorithms do a better job when determining the start positions and aligning the repeating units because they miss many more repeats having this critical range of score than our algorithm misses. A full list of these repeats are included in Appendix A (Tables A.1 and A.2).

	Tandem Repeats Finder	ATRHunter	Our Algorithm
Sequence 1	91	87	99
Sequence 2	95	92	100
Sequence 3	92	90	100
Sequence 4	91	86	99
Sequence 5	97	91	100
Overall	93.2%	89.2%	99.6%

Table 4.10: Comparison of algorithms on synthetic sequences.

4.3 Synthetic Data

The quality of our algorithm is also demonstrated on synthetic data. In the paper by Wexler *et al.* [45], experiments were performed on 10 synthetic sequences of length 100000 to compare ATRHunter with Tandem Repeats Finder. Each of these sequences includes 100 approximate tandem repeats with location, period, and level of similarity randomly chosen, and the number of copies geometrically distributed with parameter $p = 0.5$. The average score of an ATR over all the sequences was 238 with a standard deviation 116. We could obtain the first 5 of these 10 sequences from the website [1] of ATRHunter, and we scanned them by our algorithm, Tandem Repeats Finder (Version 4.00) and ATRHunter. Table 4.10 shows the numbers of tandem repeats recovered by the three algorithms for each of these five synthetic sequences.

Our algorithm recovered almost all (99.6%) of the tandem repeats and was the fastest (≈ 0.36 seconds on each sequence). Tandem Repeats Finder recovered 93.2% of the repeats in a time comparable to our algorithm (≈ 0.43 seconds on each sequence). ATRHunter recovered 89.2% of the repeats while running approximately 1.5 times slower than Tandem Repeats Finder.

[1]http://bioinfo.cs.technion.ac.il/atrhunter/ATRexperiments.htm

	Tandem Repeats Finder	ATRHunter	Our Algorithm
Sequence for period 50	46	45	100
Sequence for period 100	51	50	99
Sequence for period 200	38	70	100
Sequence for period 400	37	91	100

Table 4.11: Comparison of algorithms on repeats with significant number of errors.

4.3.1 Repeats with Significant Errors

To demonstrate our algorithm's ability to detect repeats with significantly high number of errors, we created several synthetic sequences containing repeats with 10% substitutions and 10% indels. First we created a random sequence of length 100000 and then we planted 100 single repeats with period 50 where on the second repeating unit, 5 substitutions (10%) are applied to randomly chosen positions and 5 indels (10%) are applied again to randomly chosen positions. We repeated the same process for period sizes of 100, 200 and 400 such that each repeat has 10% substitutions and 10% indels. Then we ran all the three algorithms on these sequences. The numbers of recovered repeats are shown in Table 4.11.

Again our algorithm recovered all of them (with only one exception) while being fastest. ATRHunter has also a good performance for repeats with large periods. However as the experiments on real data also confirms, Tandem Repeats Finder is not as good as the others for long repeats with errors.

4.3.2 Repeats with Long Periods

Our algorithm is virtually capable of detecting repeats with any period range. While ATRHunter is limited by only detecting repeats with period up to 500, Tandem Repeats Finder can detect periods up to 2000. To compare the effectiveness of our algorithm on repeats with long periods, we created two sets of sequences of length

	Tandem Repeats Finder	Our Algorithm
Sequence for period 1000	19	100
Sequence for period 1800	18	100

Table 4.12: Comparison of algorithms on repeats with long periods.

1000000 containing 100 approximate single repeats with periods 1000 and 1800 respectively. The repeats are created and planted as described above to have 10% substitutions and 10% indels. We ran Tandem Repeats Finder and our algorithm on these sequences (Table 4.12).

Our algorithm recovered all of the repeats running in approximately 4 seconds. On the other hand Tandem Repeats Finder only recovered 20% of the repeats running in 3.3 seconds.

Chapter 5

Conclusions

An efficient algorithm for detecting approximate tandem repeats in genomic sequences is presented in this thesis. The algorithm is based on statistical criteria to detect candidate regions which may include tandem repeats and then these regions are subsequently verified by alignments using dynamic programming. No prior information about the period size or pattern which is repeated is needed. Also the algorithm is virtually capable of detecting repeats with any period.

The parameters (p_M, p_I) for the error model of formation of tandem repeats as well as the similarity thresholds $\theta_{max}(t)$ between the repeating units are adjustable, therefore our algorithm can be used for detecting tandem repeats with various range of similarity measures.

An implementation of the algorithm is compared with the two state-of-the-art tandem repeats detection tools, Tandem Repeats Finder [6] and ATRHunter [45], and as the results show that our algorithm performs significantly better both in detecting more repeats and spending less time.

5.1 Future Work

5.1.1 Accuracy for Short Repeats

While our algorithm performs better than the others for repeats with periods smaller than 40, most of the false alarms are produced for those repeats as explained in Section 3.4.3. Different approaches can be incorporated to reduce the false alarms for short repeats such as combining the idea of producing chains with looking all the occurrences (not only the immediately preceding ones) of shorter windows (of size 2 or 3), or using some suffix tree based approaches.

5.1.2 Speeding up the Alignments

The chains are only used to detect the positions and the periods of candidate tandem repeat regions in our algorithm. However they also include a lot of information about the matching substrings of these regions. When computing the alignments in the verification phase, this information could be effectively used to reduce the computation effort.

5.1.3 Detecting Hierarchical Tandem Repeats

Some tandem repeats exhibit a hierarchical structure, namely the repeated pattern may include several other tandem repeats. Our algorithm may be modified to detect such kind of hierarchies in tandem repeats.

Appendix A

Detected Tandem Repeats in Yeast Chromosome I and Complete Genome of E. Coli O157:H7

The approximate tandem repeats that are detected by our algorithm are listed in the following tables for the sequences of yeast chromosome I and E. coli O157:H7. The redundancies are removed in these tables as described earlier. The mark "#" at a repeat indicates that that repeat is detected by Tandem Repeats Finder as well as our algorithm. Similarly the mark "+" indicates that the repeat is detected by ATRHunter. Repeats marked by both "#" and "+" are detected by all the three algorithms. Unmarked repeats are only detected by our algorithm.

Table A.1: Tandem repeats found by our algorithm in yeast chromosome I.

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
#+	15	61	6	7.83	6	83.7	9.3	55	10.2
	219	296	36	2.14	36	75	11.4	73	13.8
	5763	5815	25	2.04	26	75	7.1	55	13
#+	11864	11935	27	2.7	27	84.8	2.2	88	11
#+	12249	12327	21	3.64	22	79.3	5.2	77	13.4
#+	12466	12855	48	8.08	48	89.5	0.6	573	7.4

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
#+	13001	13128	15	8.53	15	81.9	5.2	130	14.1
#+	14791	14821	13	2.38	13	94.4	0	55	3.2
+	14892	14927	18	2	18	83.3	0	51	8.3
	18042	18081	19	2.16	19	81.8	4.5	52	9.8
	18186	18230	24	1.91	23	78.3	13	51	10.9
	23551	23612	33	1.94	31	74.2	6.5	64	12.9
#+	24296	24367	27	2.7	27	84.8	2.2	88	11
	24373	24417	24	1.92	24	78.3	13	53	10.6
#+	24688	24780	21	4.43	21	80.6	0	109	11.8
	24919	25012	47	1.94	48	78.7	6.4	114	10.5
#+	25166	25293	15	8.53	15	81.9	5.2	130	14.1
#+	25403	26598	135	8.86	135	92.5	0.2	987	16.6
#+	26464	27148	135	5.07	135	92.4	0	1069	6.3
	32850	32907	30	2.03	29	74.2	9.7	58	13.3
	39130	39180	24	2.04	25	74.1	7.4	51	13.5
	40241	40281	18	2.29	17	80	12	50	9.8
	49399	49451	29	1.96	27	74.1	11.1	53	12.7
#+	55048	55093	21	2.14	21	80	4	55	10.9
	69661	69711	26	1.93	27	76	4	60	11.5
	73168	73217	25	1.93	27	76.9	15.4	56	11.3
#+	76702	76746	21	2.14	21	87.5	0	69	6.7
#+	76831	76862	15	2.13	15	88.2	0	50	6.2
	76981	77033	27	2.08	26	79.3	10.3	62	10.9
#+	77011	77057	24	1.96	24	78.3	0	59	10.6
+	77062	77132	30	2.48	29	74.4	4.7	70	13.7
#+	77497	77544	3	15.67	3	93.3	2.2	80	4.2
#+	77572	77601	12	2.5	12	94.4	0	53	3.3
	80780	80825	24	1.92	24	72.7	0	50	13
	87687	87726	20	1.95	21	80	5	52	9.8
	95349	95393	24	1.91	23	77.3	4.5	53	11.1
	97511	97560	25	2.04	25	76.9	3.8	58	11.8
#+	99940	99971	14	2.29	14	100	0	64	0
#+	100363	100399	18	2.06	18	89.5	0	60	5.4
#+	100449	100508	27	2.22	27	78.8	0	71	11.7
#+	101466	101506	15	2.73	15	88.5	0	54	9.8
	106182	106228	26	1.92	25	71.4	0	50	12.2
	107771	107817	23	2	23	75	4.2	50	12.8
	110647	110711	33	2.03	33	74.3	11.4	65	13.2
#+	113050	113096	3	15.67	3	88.6	0	73	6.4
#+	113285	113317	9	3.67	9	87.5	0	52	6.1
#+	116424	116467	21	2.1	21	78.3	0	53	11.4
#+	116489	116515	9	3	9	100	0	54	0
	118234	118280	24	1.96	24	75	8.3	50	12.5
#+	118468	118515	18	2.61	18	76.7	3.3	52	12.5
#+	120158	120184	5	5.4	5	100	0	54	0
#+	124924	124953	9	3.33	9	95.2	0	53	3.3
	129282	129322	19	2.16	19	83.3	16.7	50	9.3
	129692	129746	27	2.04	27	75	0	61	12.7
+	132935	132979	21	2.14	21	79.2	0	55	11.1
	133378	133418	20	2	20	81	4.8	52	9.8
	133426	133486	30	2.07	30	76.5	14.7	62	12.5
#+	139547	139605	24	2.46	24	82.9	0	76	10.2

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	141443	141489	24	2.04	23	76	8	50	12.5
	142557	142602	24	1.92	24	72.7	0	50	13
+	151474	151512	19	2.05	19	80	0	50	10.3
	151499	151534	20	1.9	20	83.3	11.1	51	7.9
#	151514	151566	25	2.2	25	80	6.7	64	10.9
	156062	156116	30	1.93	28	75.9	17.2	55	12.3
	162401	162463	32	1.94	33	74.2	3.2	70	12.5
	168115	168157	21	2.1	21	78.3	4.3	51	11.4
	175803	175854	26	2.08	25	75	7.1	53	13.2
	176473	176534	29	2.03	29	77.1	20	58	12.5
	176559	176602	24	2	23	79.2	16.7	51	10.6
	176958	177001	21	2.1	21	78.3	0	53	11.4
	184598	184656	27	2.04	28	75	12.5	56	13.3
#+	190123	190152	13	2.23	13	94.1	5.9	51	3.3
#+	192282	192321	3	13	3	89.5	5.3	57	7.5
#+	193957	194025	27	2.56	27	78.6	0	82	11.6
#+	196003	196039	18	2.11	18	85	5	53	7.9
#+	198830	198866	11	3.27	11	88.5	3.8	51	8.1
	200781	200841	32	1.94	31	75.9	10.3	64	13.1
	202638	202670	18	1.94	17	88.2	11.8	50	5.9
#+	204217	206360	135	15.88	135	91.1	0.3	2451	12.1
#+	206226	206632	135	3.01	135	85.7	0	590	7.9
#+	206737	206783	15	3.2	15	81.8	3	52	12.5
#+	206769	206830	15	4.07	15	83	2.1	71	11.1
	206990	207086	48	2.02	48	75.5	0	110	12.4
#+	207222	207286	21	3.1	21	81.8	0	88	9.2
#+	207609	207697	27	3.3	27	88.7	0	136	6.7
+	214006	214045	19	2.16	19	81.8	4.5	52	9.8
	216469	216523	27	2.11	27	73.3	6.7	54	14
	217349	217388	20	2	20	80	0	52	10
#+	219178	219214	15	2.47	15	86.4	0	60	5.4
#+	222276	222314	18	2.17	18	81	0	50	10.3
	222559	222776	108	2.02	108	82	1.8	294	9.1
#+	223114	223149	1	36	1	100	0	72	0
#+	229746	229806	15	4.07	15	80.4	0	73	11.5
#+	229945	229981	11	3.36	11	88.9	7.4	51	7.9
#+	230112	230201	6	15.5	6	87.8	11.1	136	6.4

Table A.2: Tandem repeats found by our algorithm in E. coli O157:H7.

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
#+	198	254	12	4.75	12	86.7	0	79	8.8
#+	208	255	6	8.17	6	81.4	2.3	54	12.2
	14452	14497	24	1.92	24	86.4	0	71	6.5
	18579	18632	27	2.04	27	79.3	10.3	64	10.7
	36404	36448	23	2.05	22	79.2	8.3	53	10.9
#	54992	55033	21	2	22	81.8	9.1	56	9.1
	58639	58690	28	1.93	27	73.1	7.7	53	13.2
	62879	62926	24	2	24	75	0	54	12.5
	63902	63958	30	1.9	30	74.1	0	65	12.3
#+	64015	64082	15	4.53	15	94.3	0	122	2.9
#+	71495	71675	85	2.13	85	92.7	0	313	3.9
	75274	75315	23	1.9	21	81	9.5	52	9.5
	82587	82639	24	2.25	24	76.7	3.3	57	13
+	88943	88979	17	2.12	17	85	5	51	8.1
	96804	96857	27	1.96	27	75	10.7	55	12.7
	107555	107602	23	2.04	23	76.9	11.5	50	12.2
	108558	108611	27	2.15	26	77.4	12.9	57	12.3
	110903	110953	24	2.04	25	77.8	7.4	58	11.5
#+	130499	131095	303	2.01	298	81.7	4.2	790	9.3
#+	133999	134273	96	2.89	96	81.8	1.1	368	9.4
#+	134666	134710	21	2.14	21	79.2	0	55	11.1
	181510	181555	24	1.92	24	72.7	0	50	13
	185304	185375	35	2.03	35	75.7	2.7	79	12.5
	199182	199228	23	2.09	23	76.9	11.5	50	12.2
	204093	204133	22	1.91	22	80	5	54	9.5
	211571	211609	21	1.9	21	78.9	5.3	50	10
	241744	241788	23	1.92	24	78.3	13	53	10.6
+	241885	241926	21	2	21	81	0	56	9.5
#+	248787	248856	27	2.5	26	84.4	11.1	65	12.7
	252237	252281	22	1.95	22	78.3	8.7	51	11.1
	266526	266574	25	2	25	76	4	56	12
	267754	267805	27	2.08	26	75.9	13.8	53	12.7
#+	271422	271477	6	9.33	6	100	0	112	0
	274673	274725	27	2.04	26	75	7.1	55	13
	288099	288145	24	1.96	25	75	8.3	52	12.2
	301283	301328	24	1.96	24	73.9	4.3	50	12.8
+	303746	303787	20	2.05	20	86.4	4.5	61	7.1
#+	303789	303857	23	3.04	23	85.4	6.2	87	9.9
#	314566	314625	30	1.91	32	76.7	10	69	11.3
	317569	317614	24	1.92	24	72.7	0	50	13
	339818	339869	24	2.04	25	75	10.7	51	13.2
	342165	342210	24	1.92	24	72.7	0	50	13
	350355	350407	28	1.93	28	73.1	3.8	57	13
	373311	373357	24	1.96	24	73.9	0	52	12.8
#+	382641	382696	18	3.11	18	81.6	0	63	12.5
#+	391253	391716	93	5.02	93	94.7	1.1	802	3.9
#+	411014	411759	94	8.02	95	92.3	1.7	1163	6.2
	417371	417416	24	1.92	24	72.7	0	50	13
	440781	440832	27	1.96	26	73.1	3.8	53	13.5

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
#	451613	451662	25	2.04	25	76.9	3.8	58	11.8
	468680	468727	26	1.96	24	76	12	50	12.2
#+	487334	487404	23	3.09	22	78	10	50	16.7
#	502429	502463	16	2.25	16	90	5	56	5.6
	522259	522304	22	2	22	80	16	51	10.6
	533600	533660	34	1.94	31	74.2	12.9	60	12.7
	547081	547127	21	2.23	22	74.1	7.4	52	12.2
	550962	551013	28	1.93	28	77.8	14.8	60	10.9
	551184	551229	21	2.05	22	80	12	53	10.6
	555796	555840	21	2.14	21	83.3	0	62	8.9
#	559475	559513	18	2.18	17	82.6	13	53	7.7
	565903	565964	29	2.07	29	72.7	6.1	57	14.5
	567815	567846	16	2.06	16	88.2	5.9	50	6.1
	583066	583112	25	1.92	24	73.9	4.3	50	12.8
	586772	586823	27	1.96	27	73.1	3.8	55	13.2
	587256	587302	24	1.96	25	75	8.3	52	12.2
	595451	595505	27	2.07	27	75.9	3.4	61	12.5
	595644	595693	25	1.92	26	76	8	56	11.8
#+	596193	596810	308	2	308	99.7	0.3	1227	0.2
	598401	598446	24	1.92	24	72.7	0	50	13
#+	607318	607364	25	2	24	76	12	50	12.2
	607399	607439	22	1.91	22	80	5	54	9.5
#+	608929	609008	33	2.41	32	75	4.2	77	13.8
	612720	612772	28	1.96	27	74.1	7.4	55	13
+	616217	616258	21	2	21	81	0	56	9.5
	619057	619104	26	1.92	25	76	16	50	12
	620396	620448	27	2.04	26	75	7.1	55	13
	630080	630127	25	1.92	25	73.9	0	54	12.5
	634515	634568	28	1.93	28	75.9	20.7	53	12.3
	643664	643712	25	1.92	25	76	12	52	12
	671298	671362	30	2.03	31	74.3	11.4	61	13.6
#	671873	671913	22	1.91	22	80	5	54	9.5
#+	671912	671955	17	2.47	17	81.5	3.7	56	9.1
	673282	673321	21	1.9	21	78.9	0	52	10
	699586	699639	28	1.93	28	75	14.3	55	12.5
+	702686	702728	20	2.14	21	79.2	8.3	51	11.1
	708812	708865	30	1.93	28	75.9	20.7	53	12.3
	714221	714264	23	1.96	24	78.3	13	53	10.6
	721205	721245	21	1.95	21	80	0	54	9.8
	730509	730559	27	1.92	26	76	4	58	11.8
	735796	735842	23	2	23	79.2	4.2	57	10.6
	744846	744894	24	1.92	25	76	12	52	12
	754854	754899	24	1.96	24	72.7	0	50	12.8
	756412	756447	20	1.9	20	83.3	11.1	51	7.9
+	771047	771084	20	1.9	20	83.3	0	55	7.9
#+	776475	776544	34	2.03	34	83.3	2.8	96	8.6
	776587	776808	113	2.03	110	81.9	6	291	9.3
	776916	777126	109	2.01	105	84.5	7.3	295	7.9
	799416	799482	32	2.12	33	74.4	17.9	60	13.9
	804774	804812	21	1.9	21	83.3	0	57	7.5
	811302	811354	28	1.93	27	74.1	11.1	53	13
	813815	813868	26	2	26	75	7.1	55	13

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	837632	837678	24	2.09	23	76.9	11.5	50	12.2
	852936	852979	24	1.91	23	77.3	9.1	51	11.1
	858107	858153	24	1.92	25	73.9	4.3	52	12.5
#	861246	861277	15	2.13	15	88.2	0	50	6.2
#+	861261	861311	24	2.12	24	77.8	0	60	11.8
#+	861331	861416	45	1.91	45	80.5	0	116	9.3
#+	861409	861487	36	2.11	37	79.5	11.4	96	9.9
#+	861470	861628	77	2.01	79	78	4.9	188	11.2
+	865062	865476	208	2.01	208	90.5	2.4	688	4.8
	867161	867206	24	1.92	24	72.7	0	50	13
	887009	887047	20	1.9	20	84.2	5.3	55	7.7
	887063	887139	35	2.05	37	73.8	11.9	71	13.9
	889264	889312	23	2.09	23	78.6	17.9	50	11.8
	892279	892325	25	1.96	24	75	8.3	50	12.5
	900281	900329	23	2.09	23	76.9	3.8	54	12.2
	912924	912956	17	1.94	17	87.5	0	52	6.1
	922439	922490	27	1.96	27	74.1	11.1	53	13
	944426	944465	19	2.1	20	81.8	9.1	52	9.5
#+	951746	951774	15	1.93	15	92.9	0	51	3.4
	954291	954330	21	1.9	21	80	10	50	9.8
	987156	987205	23	2.04	24	78.6	17.9	52	11.5
	991009	991047	20	2	20	80	5	50	10
	1009634	1009688	30	1.93	29	75.9	17.2	57	12.1
	1018182	1018228	24	1.96	24	75	8.3	50	12.5
	1023473	1023520	23	2.04	23	76.9	11.5	50	12.2
	1055675	1055714	21	1.9	21	78.9	0	52	10
+	1064943	1064977	18	1.94	18	88.2	0	56	5.7
+	1065508	1065587	39	2.05	39	82.9	0	111	8.8
#+	1066509	1066787	39	7.18	39	86.7	0.4	383	8.9
	1066598	1066644	24	1.96	24	73.9	0	52	12.8
	1084183	1084228	24	1.92	24	72.7	0	50	13
	1084222	1084271	27	1.92	26	75	4.2	56	11.8
	1085109	1085168	30	2	29	75.8	18.2	56	12.9
+	1101699	1101748	26	1.92	26	75	0	58	12
	1107627	1107665	21	1.9	20	84.2	5.3	55	7.7
#+	1116636	1116675	20	1.95	21	80	5	52	9.8
	1118692	1118735	22	1.95	22	77.3	4.5	51	11.4
	1130812	1130872	34	1.94	31	75	15.6	60	12.7
	1151406	1151446	20	1.95	21	81	9.5	52	9.5
	1152411	1152452	23	1.91	22	81	9.5	54	9.3
	1152817	1152858	20	2.05	20	82.6	13	52	9.3
	1162983	1163035	28	1.93	28	73.1	3.8	57	13
	1166611	1166660	24	1.92	26	73.1	11.5	54	11.5
	1168380	1168427	25	1.96	25	75	4.2	54	12.2
	1168417	1168465	26	1.92	26	75	4.2	56	12
#+	1170333	1170368	11	3.27	11	84	0	58	5.6
#	1170372	1170410	20	2.05	19	85.7	9.5	55	7.5
	1176068	1176111	23	1.92	24	77.3	9.1	53	10.9
#+	1183758	1183809	21	2.48	21	74.2	0	55	13.5
	1187019	1187078	33	1.9	31	80	10	74	9.8
#+	1208487	1208577	45	2.02	45	80.4	0	119	9.9
#+	1209270	1209883	141	4.35	141	98.5	0	1179	1.1

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	1218433	1218479	25	1.96	25	78.3	13	50	12
	1228652	1228690	18	2.17	18	81	0	50	10.3
	1228723	1228768	24	1.92	24	72.7	0	50	13
	1237461	1237501	20	2	20	81	4.8	52	9.8
	1237627	1237666	21	1.95	20	80	5	50	10
#+	1258968	1258999	11	3	11	86.4	4.5	50	6.1
#+	1263615	1263679	29	2.21	29	80.6	2.8	79	10.8
	1270020	1270069	27	1.92	26	76.9	15.4	54	11.5
	1274682	1274729	27	1.92	25	76	16	50	12
#+	1275728	1275780	12	4.42	12	82.9	0	78	7.5
+	1280030	1280069	21	1.9	20	85	10	55	7.5
	1280548	1280594	24	1.96	24	73.9	0	52	12.8
#+	1285194	1285241	23	2.18	22	77.8	7.4	52	12.2
	1285380	1285420	21	1.95	21	80	0	54	9.8
#+	1286351	1286441	45	2.02	45	87	0	140	6.6
#+	1287869	1287913	21	2.19	21	84	4	62	8.7
#+	1295551	1295589	15	2.6	15	83.3	0	57	7.7
	1308545	1308608	31	2.09	32	77.1	8.6	72	11.9
	1319289	1319338	24	2.08	25	77.8	7.4	58	11.5
	1319899	1319952	24	2.08	26	76.7	13.3	55	12.5
+	1321395	1321445	27	2	26	74.1	11.1	51	13.2
#+	1326597	1326638	20	2.05	20	81.8	4.5	54	9.5
+	1335618	1335661	23	2.05	22	79.2	12.5	51	10.9
	1340687	1340734	24	2.04	24	76	4	54	12.2
	1364014	1364063	25	1.93	27	76	8	58	11.5
	1366236	1366275	19	2.1	20	81.8	9.1	52	9.5
#+	1373654	1373689	18	2.06	18	89.5	5.3	58	5.4
	1377617	1377665	24	2.08	24	77.8	11.1	54	11.8
	1398133	1398181	25	1.96	25	80	8	61	10
#+	1399064	1399156	46	2.02	46	89.4	0	151	5.4
#+	1399871	1400170	149	2.01	149	86.1	0.7	451	7
#	1401221	1401253	17	2.06	16	88.9	11.1	50	5.9
	1401947	1401992	22	2	23	79.2	8.3	55	10.6
#+	1405286	1405385	47	2.2	46	80.4	5.4	121	10.8
#+	1406245	1406273	15	1.93	15	92.9	0	51	3.4
	1421972	1422017	23	2.13	23	76.9	11.5	50	12.2
#+	1425404	1425442	19	2.05	19	95	0	71	2.6
	1425494	1425543	26	2	25	76.9	7.7	56	11.8
#+	1443913	1443940	13	2.15	13	100	0	56	0
	1445301	1445340	21	1.95	21	81	14.3	50	9.5
	1448136	1448186	27	1.92	26	76.9	11.5	56	11.5
	1448493	1448539	23	2	23	75	4.2	50	12.8
	1455316	1455364	24	2.04	23	76.9	7.7	52	12.2
+	1463666	1463707	18	2.33	18	79.2	0	56	9.5
	1472053	1472102	27	1.96	26	76.9	11.5	56	11.5
	1488303	1488354	25	2.08	26	75	7.1	55	13
	1494411	1494456	23	2.04	24	76	12	50	12.2
	1498005	1498045	21	1.91	22	80	5	54	9.5
	1501225	1501278	24	2.08	25	76.7	13.3	53	12.7
	1510118	1510165	26	1.92	26	75	8.3	54	12
#+	1520733	1520758	6	4.33	6	100	0	52	0
	1521812	1521858	24	1.96	25	75	8.3	52	12.2

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	1525128	1525178	26	2.08	25	75	10.7	51	13.2
	1532488	1532532	24	1.92	24	77.3	4.5	55	10.9
	1536362	1536408	24	1.96	25	75	8.3	52	12.2
	1544528	1544574	25	1.92	25	73.9	4.3	52	12.5
	1545194	1545242	26	1.92	25	75	4.2	54	12.2
#+	1546194	1546219	10	2.6	10	100	0	52	0
	1550415	1550482	33	2.03	33	74.3	2.9	71	13.2
	1552551	1552594	23	1.92	24	77.3	9.1	53	10.9
+	1557627	1557673	23	2	23	75	4.2	50	12.8
	1560187	1560234	27	1.92	25	76	16	50	12
#+	1568084	1568729	162	4.01	162	92	1.4	959	7.2
	1574291	1574350	27	2.23	26	73.5	5.9	60	13.3
	1583210	1583248	21	1.95	21	80	10	50	9.8
	1588187	1588235	26	1.96	25	76	8	54	12
	1589314	1589353	21	1.91	22	78.9	5.3	52	9.5
	1594603	1594641	19	2.05	19	80	0	50	10.3
#+	1600691	1600741	21	2.43	21	86.7	0	74	7.8
	1600796	1600839	20	2.05	21	83.3	12.5	56	8.9
	1605065	1605108	21	2.05	21	78.3	4.3	51	11.4
	1605683	1605729	23	2.09	23	76	4	52	12.5
	1609519	1609609	48	1.92	48	74.4	0	105	12
#+	1618160	1618201	12	3.42	12	86.7	3.3	61	7.1
#+	1619929	1620019	47	2.02	45	77.6	12.2	99	11.7
	1628670	1628710	22	1.91	22	80	5	54	9.5
#+	1631353	1631394	21	2	21	81	0	56	9.5
	1641208	1641263	30	1.96	28	73.3	16.7	50	13.8
#+	1648311	1648352	19	2.2	20	83.3	8.3	56	9.1
+	1648331	1648383	26	2	26	75	10.7	53	13
	1650040	1650079	21	1.91	22	80	10	52	9.5
#+	1650890	1651223	132	2.53	132	78.9	2	377	12.2
	1652281	1652336	27	2.07	27	77.4	12.9	59	12.1
#+	1678731	1678808	33	2.36	33	97.8	0	149	1.3
	1679600	1679640	20	2	20	81	4.8	52	9.8
	1680700	1680747	22	2.09	23	76.9	7.7	52	12.2
+	1696443	1696489	23	2	23	79.2	4.2	57	10.6
	1702199	1702245	25	1.92	25	73.9	4.3	52	12.5
	1705823	1705874	27	1.96	27	75	17.9	51	12.7
#+	1711513	1711541	15	1.93	15	92.9	0	51	3.4
#+	1731891	1732307	178	2.34	176	89.2	1.7	570	8.6
+	1732109	1732677	297	1.92	298	81.4	4.3	766	9
#+	1754208	1754239	12	2.67	12	80	0	50	6.2
	1756792	1756835	23	1.92	24	77.3	9.1	53	10.9
#+	1757508	1757542	11	3	11	91.7	8.3	52	5.7
#	1759828	1759911	42	2.02	43	77.3	6.8	98	11.5
#+	1763778	1763827	22	2.19	21	80	16.7	57	10
	1765509	1765549	21	1.9	21	80	5	52	9.8
	1771070	1771113	23	1.92	24	77.3	9.1	53	10.9
	1779131	1779178	27	1.92	25	76	16	50	12
#+	1786557	1787203	162	4.01	160	87.6	2.6	831	9.6
	1792603	1792662	27	2.23	26	73.5	5.9	60	13.3
#+	1799875	1799965	45	2.02	45	80.4	0	119	9.9
#	1805029	1805090	30	2.07	30	78.1	0	75	11.3

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	1805603	1805655	28	1.93	28	74.1	11.1	55	12.7
	1811533	1811581	26	1.92	26	75	4.2	56	12
	1828895	1828938	22	2.05	22	79.2	12.5	51	10.9
#	1830277	1830307	16	2	16	93.8	6.2	55	3.1
#+	1833663	1833710	24	1.96	24	79.2	4.2	59	10.4
	1875046	1875098	28	1.93	28	76	8	55	12.7
	1878790	1878846	31	1.9	31	75	7.1	65	11.9
	1880803	1880851	24	1.92	26	76	12	54	11.8
	1884232	1884272	22	1.9	21	78.9	0	52	9.8
	1923653	1923695	22	2	22	77.3	4.5	51	11.4
	1929101	1929147	24	1.92	25	73.9	4.3	52	12.5
	1937514	1937557	23	1.91	23	76.2	0	53	11.4
	1940387	1940436	27	1.96	27	73.9	13	51	13.2
	1946429	1946476	27	1.92	25	76	16	50	12
#+	1953881	1954444	162	3.48	160	93.3	1	895	5.5
#+	1969375	1969465	45	2.02	45	84.8	0	133	7.7
#+	1984416	1984625	30	7.2	30	83.9	4.8	280	9.3
	1994356	1994411	27	2.07	27	77.4	12.9	59	12.1
	1995419	1995464	23	1.96	23	80	20	51	10.4
	2021124	2021169	26	1.92	24	79.2	16.7	53	10.4
	2030289	2030328	22	1.9	21	83.3	0	50	9.8
	2044562	2044608	25	1.96	24	75	8.3	50	12.5
	2045303	2045348	24	1.92	25	78.3	8.7	57	10.4
#+	2045334	2045386	24	2.21	24	89.7	0	85	5.7
	2047913	2047960	26	1.92	25	76	16	50	12
	2049252	2049304	27	2.04	26	75	7.1	55	13
#+	2050663	2050688	13	2	13	100	0	52	0
#+	2051838	2051870	8	4.12	8	100	0	66	0
	2064558	2064601	23	1.91	23	77.3	9.1	51	11.1
	2065760	2065814	28	1.93	28	75	10.7	57	12.5
#+	2065961	2065992	15	2.13	15	88.2	0	50	6.2
	2067866	2067904	21	1.9	20	84.2	5.3	55	7.7
	2073672	2073729	28	2.04	28	74.2	9.7	56	13.6
	2089851	2089896	24	1.92	24	78.3	8.7	55	10.6
	2095906	2095952	24	1.96	24	73.9	0	52	12.8
	2097778	2097829	26	2.04	26	75	10.7	53	13
	2114470	2114526	31	1.93	29	75.9	10.3	61	12.1
	2120463	2120517	30	2	28	73.3	16.7	50	13.8
	2128723	2128769	25	1.92	25	78.3	4.3	59	10.4
	2132306	2132345	21	1.9	21	78.9	0	52	10
	2138431	2138477	24	2	24	75	4.2	52	12.5
	2138862	2138907	24	1.91	23	78.3	8.7	53	10.9
#+	2161419	2161447	9	3.22	9	100	0	58	0
#	2163487	2163529	19	2.26	19	84	8	56	9.1
	2165278	2165320	22	1.95	22	76.2	0	51	11.6
#	2180760	2180801	21	2	21	81	0	56	9.5
#+	2183150	2183194	23	2.09	22	80	12	53	10.6
	2193461	2193514	30	1.93	29	75	14.3	57	12.3
	2194593	2194656	33	1.94	33	78.1	6.2	77	10.8
#+	2197861	2197895	6	5.83	6	86.2	0	56	5.7
	2197904	2197946	21	2.14	21	79.2	8.3	51	11.1
	2199098	2199142	24	1.91	23	77.3	4.5	53	11.1

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	2210450	2210502	28	1.93	28	73.1	3.8	57	13
	2210586	2210635	28	1.93	28	73.1	15.4	51	13
#+	2212142	2212170	9	3.22	9	100	0	58	0
	2215090	2215145	26	2	27	73.3	13.3	50	14
	2215999	2216041	22	1.95	22	76.2	0	51	11.6
#+	2226621	2227268	162	4.01	163	87.8	2.2	875	9
	2244345	2244412	34	2.03	34	74.3	2.9	73	13
	2245289	2245329	20	2	20	81	4.8	52	9.8
	2247180	2247221	19	2.1	20	82.6	8.7	54	9.3
#+	2247254	2247322	30	2.3	30	79.5	0	82	11.6
#+	2252889	2252935	21	2.19	21	81.5	11.1	55	10.4
	2252933	2252979	21	2.25	20	81.5	7.4	55	10.6
	2274169	2274209	22	1.91	22	81	14.3	52	9.3
+	2305798	2305835	19	2.11	18	85.7	9.5	53	7.7
#+	2308176	2308819	96	6.7	94	90.6	1.1	665	12.7
	2332652	2332705	28	1.93	27	75	14.3	53	12.7
	2335534	2335573	21	1.9	21	78.9	0	52	10
#+	2346145	2346307	81	2.01	81	89	0	263	5.5
	2346599	2346638	21	1.9	21	78.9	0	52	10
	2373365	2373417	28	1.93	27	74.1	11.1	53	13
	2387791	2387836	24	1.92	24	72.7	0	50	13
	2395443	2395478	18	2	18	83.3	0	51	8.3
+	2408437	2408478	20	2.15	20	82.6	4.3	56	9.3
#+	2430926	2430974	24	2.04	24	80	0	63	10.2
#+	2431827	2431872	23	2.04	23	83.3	4.2	64	8.5
	2465259	2465315	32	1.93	28	77.8	7.4	50	13.8
+	2484685	2484731	21	2.14	22	76.9	7.7	50	12.5
	2512910	2512956	25	1.96	24	75	8.3	50	12.5
	2517082	2517130	25	1.92	26	76	12	54	11.8
	2528345	2528413	27	2.46	28	77.3	15.9	62	13.9
+	2535112	2535160	23	2.17	23	77.8	3.7	56	12
#	2539827	2539866	15	2.73	15	76.9	3.8	52	9.8
	2540418	2540470	27	1.96	26	74.1	7.4	53	13.2
#+	2546049	2546281	97	2.38	98	91.2	1.5	387	4.7
	2549334	2549377	22	1.95	22	77.3	4.5	51	11.4
	2551369	2551436	36	2	35	75	11.1	71	12.7
	2557882	2557928	25	1.96	25	76	16	50	12
	2558680	2558726	24	1.96	24	75	8.3	50	12.5
	2573983	2574029	21	2.14	21	77.8	14.8	53	10.4
	2589175	2589237	31	2.06	31	75.8	3	70	12.5
	2595570	2595624	30	1.93	28	76	4	50	14.3
#	2608776	2608820	21	2.15	20	80	8	51	11.1
	2617728	2617778	26	2	26	73.1	3.8	53	13.5
	2625387	2625441	29	2.07	27	74.2	16.1	50	13.8
#+	2640987	2641124	62	2.26	62	97.4	2.6	262	1.4
	2651955	2652002	24	2	24	76	8	52	12.2
	2652189	2652253	33	2.03	33	74.3	11.4	65	13.2
	2658421	2658477	29	1.93	29	75	3.6	63	12.3
	2660084	2660123	17	2.29	17	75	8.3	50	10
	2668970	2669018	26	1.92	25	75	4.2	54	12.2
	2669012	2669064	29	1.93	28	74.1	11.1	55	12.7
	2669218	2669273	29	2.04	28	76.7	10	61	12.1

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
#+	2669778	2670543	141	5.45	140	96.2	0.5	1333	3.5
#+	2671239	2671329	45	2.02	45	80.4	0	119	9.9
#	2673437	2673479	19	2.26	19	84	8	56	9.1
	2675228	2675270	22	1.95	22	76.2	0	51	11.6
	2688244	2688287	24	1.91	23	76.2	4.8	51	11.1
	2689680	2689741	30	1.97	31	78.1	9.4	71	11.1
+	2690704	2690756	23	2.17	24	76.7	10	53	13
	2709757	2709807	21	2.48	21	71	3.2	53	13.5
	2709791	2709830	21	1.9	21	78.9	0	52	10
#+	2711234	2711276	20	2.1	20	87	4.3	63	7
#+	2711954	2711980	13	2.08	13	100	0	54	0
	2734311	2734350	20	1.95	21	80	5	52	9.8
	2739268	2739328	32	1.94	31	75	15.6	60	12.7
	2751677	2751725	24	2.04	25	76.9	7.7	56	11.8
	2754200	2754245	25	1.92	24	78.3	8.7	55	10.6
	2771475	2771523	27	1.92	26	76	12	54	11.8
	2781812	2781861	26	1.92	25	76	8	54	12
	2781910	2781960	28	1.96	27	74.1	14.8	51	13
	2784045	2784087	22	2	22	77.3	4.5	51	11.4
	2791528	2791581	24	2.24	25	75	12.5	57	12.3
	2798855	2798901	24	1.92	24	73.9	4.3	50	12.8
#	2809724	2809755	15	2.07	15	94.1	5.9	55	3.1
	2815598	2815642	24	1.91	23	77.3	4.5	53	11.1
	2824654	2824706	27	1.96	27	74.1	7.4	55	13
#+	2828402	2829311	328	2.75	328	90.2	1.2	1461	5.4
	2839060	2839122	33	1.91	35	75.8	18.2	68	11.8
	2861560	2861597	20	1.9	20	83.3	0	55	7.9
#+	2861998	2862044	21	2.09	22	80.8	11.5	55	10.4
	2862005	2862078	36	2.06	36	78.9	0	92	10.8
	2871300	2871361	30	2.03	29	74.3	20	51	14.1
	2875526	2875577	28	1.93	28	74.1	14.8	53	12.7
	2883640	2883702	30	2.07	30	73.5	8.8	59	14.1
#+	2897316	2897406	45	2.02	45	84.8	0	133	7.7
	2900390	2900445	26	2	27	73.3	13.3	50	14
	2901299	2901341	22	1.95	22	76.2	0	51	11.6
	2922593	2922642	25	1.92	26	76	12	54	11.5
	2931507	2931550	24	1.91	23	76.2	4.8	51	11.1
	2931539	2931608	23	3.36	22	78.8	9.6	52	15.8
#+	2931552	2931631	39	2.05	39	85.4	0	118	7.5
	2945099	2945152	29	1.93	27	74.1	7.4	55	13
	2968131	2968191	31	2.03	30	75.8	12.1	62	12.7
#+	2970885	2970940	26	2.07	27	80.6	12.9	66	10.3
	2971505	2971739	116	2.03	117	78.3	1.7	288	11
	2982735	2982783	25	1.92	26	76.9	19.2	52	11.5
	2990096	2990152	24	2.4	25	75	13.9	56	13.1
+	2990836	2990879	23	1.91	23	86.4	9.1	65	6.7
#+	2993248	2993289	18	2.16	19	83.3	12.5	52	9.3
#+	2994292	2994329	18	2.11	18	90	0	62	5.3
	2994717	2994774	30	1.97	29	73.3	10	56	13.6
	2995350	2995397	24	2	24	76	8	52	12.2
	2996839	2996888	28	1.92	26	76.9	15.4	54	11.5
	2999842	2999889	24	1.96	25	76	12	52	12

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	3025521	3025553	16	2	16	88.2	5.9	50	6.1
	3026580	3026626	24	1.92	25	73.9	4.3	52	12.5
	3027333	3027389	27	2.12	26	74.2	6.5	54	14
	3027777	3027825	26	1.96	25	76.9	15.4	52	11.8
	3033482	3033530	25	2.04	24	76.9	7.7	54	12
	3037425	3037463	20	1.95	20	78.9	0	50	10.3
	3038143	3038191	26	1.92	25	76	12	52	12
#+	3049821	3050045	114	2.01	112	81.7	3.5	299	9.3
	3062585	3062629	24	1.92	24	76.2	0	55	10.9
+	3063574	3063609	18	2	18	83.3	0	51	8.3
+	3067134	3067180	18	2.61	18	79.3	0	52	12.8
	3067423	3067474	26	2	25	74.1	7.4	51	13.5
	3069245	3069279	18	1.94	18	88.2	0	56	5.7
#+	3072944	3073134	81	2.36	81	92.8	1.8	338	3.1
	3075937	3075975	19	2.11	19	81	4.8	50	10
	3076019	3076063	24	1.92	24	77.3	4.5	55	10.9
	3076239	3076290	26	2.08	25	75	7.1	53	13.2
	3078362	3078411	25	1.96	25	76	8	54	11.8
+	3087027	3087084	24	2.43	23	80	5.7	63	12.1
	3092721	3092758	20	1.95	20	84.2	5.3	55	7.7
	3096307	3096355	26	1.92	25	75	4.2	54	12.2
	3101688	3101733	24	1.96	24	73.9	4.3	50	12.8
	3103196	3103247	29	1.93	27	76.9	15.4	51	13
	3114255	3114295	22	1.95	21	81	9.5	52	9.5
	3133240	3133278	20	1.95	20	78.9	0	50	10.3
#+	3157084	3157367	91	3.12	91	94.3	0	505	3.2
#	3157685	3157732	24	2	24	79.2	0	61	10.4
	3178680	3178726	25	1.92	25	73.9	4.3	52	12.5
	3179419	3179467	24	2.12	24	74.1	7.4	56	11.8
	3188845	3188896	28	1.93	27	76	8	53	13.2
	3196154	3196206	27	2.12	25	76.7	13.3	53	12.7
	3197371	3197430	29	2.03	29	75.8	15.2	58	12.9
+	3222738	3222780	21	2.05	21	77.3	0	51	11.6
	3228249	3228303	27	2.04	27	76.7	13.3	57	12.3
	3236646	3236694	26	1.92	25	78.3	8.7	52	12
	3244819	3245036	115	1.9	115	87.6	2.9	343	5.9
	3248337	3248386	26	1.93	27	76	8	58	11.5
#	3249429	3249780	119	2.92	120	81.4	4.6	416	11.3
	3255071	3255121	24	2.04	24	78.6	14.3	54	11.5
	3269103	3269156	23	2.25	24	74.2	6.5	57	12.7
	3290526	3290570	24	1.91	23	77.3	4.5	53	11.1
#+	3297762	3297793	14	2.29	14	88.9	0	50	6.2
#+	3297802	3297840	18	2.11	18	95.2	4.8	69	2.6
	3306412	3306455	24	1.96	23	78.3	13	51	10.9
	3317404	3317455	28	1.92	26	73.1	7.7	51	13.5
	3331761	3331815	26	2.12	25	73.3	6.7	50	14.5
	3335130	3335182	28	1.93	28	74.1	11.1	55	12.7
	3349002	3349048	25	1.92	24	73.9	4.3	50	12.8
	3349332	3349380	24	1.92	25	76	12	52	12
	3377943	3377994	28	1.96	28	75	17.9	53	12.5
	3390235	3390279	21	2	22	83.3	12.5	58	8.7
	3402689	3402736	24	1.91	23	80	16	53	10.4

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	3416361	3416414	28	1.96	28	75.9	17.2	55	12.3
	3420297	3420349	27	2	27	75	10.7	55	12.7
	3421853	3421907	29	2.04	27	76.7	13.3	57	12.3
	3424006	3424080	36	2.06	36	76.2	16.7	72	12.8
	3460242	3460291	24	1.96	25	73.1	11.5	52	11.5
	3471734	3471776	22	1.95	22	76.2	0	51	11.6
	3478317	3478366	22	2.04	24	78.6	17.9	52	11.5
+	3487380	3487425	23	2	23	79.2	8.3	55	10.6
#	3490138	3490178	21	2	21	81	4.8	54	9.5
#+	3490391	3490454	6	10.5	6	96.6	1.7	112	3.1
#+	3491791	3491949	6	26.5	6	98	0	297	1.9
	3492029	3492078	26	2	26	73.1	7.7	51	13.5
	3492430	3492468	21	1.9	21	78.9	5.3	50	10
	3493969	3494011	21	2.1	21	78.3	4.3	51	11.4
	3498420	3498467	25	2.04	24	76.9	11.5	52	12
#+	3501274	3501310	12	3.08	12	80	0	53	8.1
	3503709	3503750	21	1.95	21	81	4.8	54	9.5
#	3505850	3505883	15	2.27	15	89.5	0	54	5.9
+	3541923	3542285	182	2.01	181	88.2	3.8	566	6
#+	3542118	3542478	139	2.58	140	96.4	0.9	664	2.2
	3544527	3544576	26	2.04	26	74.1	11.1	51	13.2
	3552943	3552998	28	2.03	29	74.2	16.1	54	13.3
	3561882	3561927	24	2	24	79.2	8.3	57	10.4
	3563258	3563303	24	2	24	75	8.3	50	12.5
	3569736	3569779	23	1.91	23	77.3	9.1	51	11.1
	3570664	3570711	24	2.12	24	78.6	17.9	52	11.5
	3570756	3570794	21	1.9	21	78.9	5.3	50	10
	3570820	3570873	27	2	27	74.1	0	59	13
	3571498	3571534	20	1.9	20	83.3	5.6	53	7.9
	3577560	3577610	28	1.93	27	73.1	11.5	51	13.2
	3589038	3589086	24	2.04	24	80	0	63	10.2
	3598345	3598456	61	1.92	59	76.8	8.9	129	11.3
	3623891	3623946	30	1.96	28	79.3	10.3	66	10.5
	3633754	3633812	29	2.03	29	75	12.5	58	13.1
#+	3644203	3644234	15	2.13	15	88.2	0	50	6.2
	3644754	3644804	26	2.04	26	75	14.3	51	13
	3648967	3649014	22	2.04	23	76.9	11.5	50	12.2
	3652555	3652600	24	1.92	24	72.7	0	50	13
#+	3669784	3670108	110	3.02	111	91	4	510	6
	3685807	3685853	26	1.92	25	75	12.5	50	12.2
	3693381	3693426	21	2.19	21	76	0	50	13
#	3704787	3704834	21	2.18	22	81.5	7.4	59	10.2
+	3717890	3717942	25	2.12	26	76.7	13.3	62	10.7
+	3718570	3718618	24	2.04	24	80.8	7.7	61	10
+	3720043	3720085	21	2.05	21	77.3	0	51	11.6
	3722306	3722341	20	1.9	20	83.3	11.1	51	7.9
	3724210	3724252	22	1.95	22	76.2	0	51	11.6
	3729727	3729778	25	2.21	24	76.7	10	60	11.1
	3734870	3734916	23	2.09	23	76.9	11.5	50	12.2
	3741918	3741957	21	1.95	20	80	5	50	10
	3744230	3744287	27	2.04	28	75.8	21.2	52	13.1
	3746383	3746429	23	2.13	23	76.9	11.5	50	12

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	3758889	3758952	32	2.06	31	76.5	5.9	70	12.3
	3759750	3759799	25	1.92	26	80	8	63	9.8
	3782360	3782414	30	1.93	29	74.1	18.5	50	13.8
	3784326	3784373	26	1.92	25	75	8.3	52	12.2
	3793916	3794702	393	2.01	394	81.7	5.2	1045	9.1
	3803515	3803567	27	1.96	26	75	14.3	51	13
	3806772	3806821	24	2.08	25	77.8	7.4	58	11.5
	3823291	3823507	113	1.94	112	78.7	5.6	267	10.5
	3824068	3824105	21	1.9	20	82.4	0	53	7.7
	3826195	3826242	24	2.04	24	76.9	11.5	52	12
	3831675	3831728	29	1.93	27	75	14.3	53	12.7
#+	3839912	3840117	87	2.3	89	87.4	4.2	315	6.2
+	3849940	3849980	21	1.9	21	80	5	52	9.8
	3852114	3852152	20	1.95	20	78.9	0	50	10.3
+	3853442	3853494	27	1.96	26	74.1	7.4	53	13.2
	3860321	3860368	26	1.92	25	75	8.3	52	12.2
	3862531	3862574	23	1.95	22	77.3	4.5	51	11.4
	3876913	3876958	25	1.92	24	82.6	8.7	62	8.5
+	3890931	3891162	111	2.07	112	83.1	6.5	309	8.9
	3918385	3918450	34	2.06	33	72.2	11.1	60	14.5
	3926047	3926092	24	1.92	24	72.7	0	50	13
	3929245	3929287	21	2.05	21	77.3	0	51	11.6
#+	3934069	3934822	376	2	376	90.7	0.3	1261	4.6
	3935539	3935581	21	2.05	21	77.3	0	51	11.6
	3935764	3935813	25	1.92	26	76	8	56	11.8
+	3947413	3947452	21	1.9	21	78.9	0	52	10
	3952772	3952824	28	1.93	27	73.1	3.8	55	13.2
#+	3970781	3971183	100	4.02	100	91.7	0.3	622	6.5
	3972365	3972424	32	1.93	30	74.2	16.1	56	12.9
	3974095	3974153	29	2.03	30	80.6	6.5	76	9.8
	3975894	3975939	23	1.96	23	78.3	4.3	55	10.9
	3992359	3992406	24	2.04	24	77.8	18.5	50	11.8
#+	3995188	3995233	21	2.29	21	77.8	7.4	50	12.5
	4001346	4001404	30	2.03	30	75	12.5	60	12.9
	4009295	4009525	113	2.02	114	79.2	5.8	286	10.3
	4033166	4033219	26	2.12	26	77.4	16.1	55	12.3
	4036907	4036966	30	2	31	74.2	6.5	64	12.9
	4040276	4040332	28	2.07	28	75	15.6	54	13.3
	4040441	4040481	20	2	20	81.8	13.6	50	9.5
	4048526	4048617	46	2.02	46	76.6	2.1	107	11.8
#+	4048564	4048650	33	2.64	33	83.3	0	104	11.5
	4048952	4049001	24	2.08	24	84.6	0	72	8
#+	4057877	4057924	21	2.29	21	85.2	0	68	8.3
#+	4057911	4057960	24	2.08	24	88.5	0	79	6
#+	4058047	4058143	24	4.04	24	79.7	2.7	87	15.3
#	4058148	4058204	24	2.38	24	84.8	0	72	10.5
	4058877	4058929	27	2.08	26	75.9	10.3	55	12.7
	4063630	4063682	28	1.93	28	73.1	3.8	57	13
	4074350	4074390	20	2	20	81	4.8	52	9.8
	4104820	4104863	21	2.1	21	78.3	0	53	11.4
	4105564	4105625	33	1.94	33	74.2	6.5	68	12.5
	4117074	4117112	20	1.95	20	78.9	0	50	10.3

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	4138164	4138210	24	1.92	25	73.9	4.3	52	12.5
	4143134	4143172	20	1.9	20	84.2	5.3	55	7.7
	4153201	4153252	27	2	27	74.1	7.4	55	13
+	4153543	4153585	21	2.05	21	77.3	0	51	11.6
	4171970	4172009	21	1.9	21	78.9	0	52	10
	4192530	4192573	22	2.05	22	78.3	4.3	53	11.1
	4196361	4196404	22	2	23	78.3	8.7	53	10.9
#+	4196531	4196573	18	2.33	18	84	4	56	9.3
	4199504	4199566	33	2	32	78.8	9.1	75	10.8
	4212615	4212673	30	1.97	29	75.9	10.3	58	13.6
#+	4216866	4216895	12	2.42	12	94.4	5.6	51	3.3
	4233141	4233188	24	2.04	24	76.9	11.5	52	12
	4243065	4243124	31	1.97	30	73.3	3.3	62	13.3
	4253882	4253930	25	1.92	26	76	12	54	11.8
#+	4288073	4288105	14	2.29	14	89.5	5.3	50	6.1
#+	4292992	4293051	29	2.1	29	96.8	0	113	1.6
	4302303	4302347	22	1.96	23	78.3	8.7	53	10.9
#+	4323045	4323076	15	2.13	15	88.2	0	50	6.2
	4324527	4324562	18	2	18	83.3	0	51	8.3
	4333217	4333259	23	1.91	23	81	4.8	58	9.1
	4346067	4346111	23	1.91	23	77.3	4.5	53	11.1
	4359222	4359273	24	2.04	25	75	10.7	51	13.2
+	4365789	4365841	27	1.96	27	76.9	0	64	11.3
	4409843	4409893	28	1.93	27	73.1	11.5	51	13.2
#	4418060	4418102	22	2	22	81.8	4.5	58	9.1
#+	4432482	4432517	9	4	9	100	0	72	0
	4441608	4441646	21	1.9	21	83.3	0	57	7.5
	4446486	4446532	24	1.92	24	78.3	4.3	57	10.6
	4446772	4446811	21	1.91	22	80	10	52	9.5
	4449504	4449550	25	1.92	25	75	12.5	50	12.2
	4458109	4458158	27	1.92	26	78.3	4.3	54	11.5
	4458122	4458155	16	2.06	16	88.9	5.6	52	5.9
	4459659	4459699	21	2	21	85.7	4.8	61	7.1
	4464230	4464281	24	2.08	25	71.4	7.1	53	13.2
	4469857	4469924	33	2.06	33	72.2	5.6	64	14.5
	4471437	4471478	21	1.91	22	81	9.5	54	9.3
	4492673	4492719	23	2	23	75	4.2	50	12.8
	4493076	4493120	24	1.91	23	77.3	9.1	51	10.9
	4499601	4499646	23	1.92	24	78.3	8.7	55	10.6
	4504969	4505020	28	1.96	27	74.1	11.1	53	13
	4508177	4508229	28	1.93	27	74.1	11.1	53	13
	4518220	4518258	21	1.9	21	78.9	5.3	50	10
#+	4520570	4520633	21	3.05	21	75	4.5	56	15.4
+	4520601	4521111	255	2	256	80.2	0.8	665	9.9
#+	4520835	4520884	21	2.38	21	79.3	0	65	10
#	4520856	4521322	234	2	233	78	1.7	566	11.1
	4540529	4540576	25	1.96	25	76	12	52	12
	4550619	4550656	18	2.06	18	85	5	53	7.9
	4556940	4556996	32	1.93	29	74.1	7.4	52	13.6
	4584565	4584613	24	2.04	23	76.9	7.7	52	12.2
	4585875	4585921	22	2.13	23	76.9	7.7	52	12.2
#+	4589333	4589781	141	3.18	141	99	0	877	0.7

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	4592065	4592104	21	1.9	21	78.9	0	52	10
#+	4601000	4601057	18	3.22	18	82.5	0	67	12.1
	4601619	4601670	28	1.93	28	73.1	7.7	55	13
#+	4608698	4608734	18	2.06	18	89.5	0	60	5.4
	4609960	4609995	19	1.95	19	83.3	5.6	51	8.1
	4612931	4612978	20	2.4	20	75.9	10.3	52	12.2
	4614993	4615055	32	1.94	32	74.2	3.2	68	12.7
	4619446	4619507	30	2.07	30	75.8	6.1	66	12.7
	4620048	4620095	26	1.92	25	75	8.3	52	12.2
#+	4626907	4626935	15	1.93	15	92.9	0	51	3.4
	4654024	4654064	20	2.16	19	82.6	8.7	52	9.5
	4656712	4656768	28	2.07	28	74.2	9.7	56	13.6
#	4663160	4663191	15	2.13	15	82.4	0	50	6.2
	4674674	4674710	20	1.9	20	82.4	0	53	7.9
	4688184	4688233	27	1.92	25	76	8	54	12
	4690802	4690847	21	2.1	21	80	8	53	10.9
+	4690834	4690877	23	1.91	23	77.3	9.1	51	11.1
	4690863	4690906	22	2.05	22	78.3	4.3	53	11.1
	4696865	4696905	21	1.95	21	80	0	54	9.8
	4707851	4707894	24	1.91	23	77.3	9.1	51	11.1
	4714690	4714740	28	1.93	27	73.1	11.5	51	13.2
	4727255	4727290	18	2	18	83.3	0	51	8.3
#+	4743784	4743820	15	2.47	15	90.9	0	60	5.4
	4744010	4744060	28	1.93	28	73.1	11.5	53	13
+	4770604	4770650	25	1.92	25	73.9	4.3	52	12.5
	4774783	4774823	21	1.95	21	80	0	54	9.8
#+	4775955	4776428	225	2.11	226	98	0.8	913	1.1
#	4779965	4779994	16	1.94	16	93.3	6.7	53	3.2
#+	4781565	4781601	6	6.17	6	93.5	0	60	5.4
	4796631	4796673	21	2.05	21	77.3	0	51	11.6
#+	4799044	4799080	18	2.11	18	85	5	53	7.9
#+	4811539	4811577	19	2	19	85	5	55	7.7
#	4842737	4842776	19	2.05	19	81	4.8	50	10
	4847102	4847156	28	1.96	27	75	7.1	57	12.7
	4848177	4848229	27	2.04	26	75	7.1	55	13
	4853210	4853250	21	1.95	21	80	0	54	9.8
#	4858458	4858497	21	1.95	20	80	5	50	10
	4869224	4869269	25	1.92	25	73.9	8.7	50	12.5
	4886082	4886130	24	1.92	25	76	12	52	12
	4892002	4892044	20	2.1	20	78.3	4.3	56	9.3
	4896375	4896419	23	1.91	23	77.3	4.5	53	11.1
	4897956	4898000	21	2.1	21	79.2	4.2	53	11.1
	4898056	4898098	22	2	22	77.3	4.5	51	11.4
	4901879	4901930	26	2	26	74.1	7.4	53	13.2
	4917365	4917416	29	1.93	27	74.1	14.8	51	13
	4920611	4920661	25	1.93	27	76.9	11.5	58	11.3
#	4924240	4924281	20	2.1	20	82.6	8.7	54	9.3
	4924275	4924339	33	2.03	32	74.3	11.4	63	13.4
	4926163	4926211	24	1.96	25	76	8	54	12
	4926963	4927013	28	1.93	27	73.1	11.5	51	13.2
	4928962	4929013	28	1.96	27	74.1	11.1	53	13
	4932170	4932222	28	1.93	27	74.1	11.1	53	13

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
	4940776	4940819	23	1.96	23	77.3	4.5	53	11.1
#	4942184	4942233	23	2.08	24	81.5	7.4	63	9.8
	4949944	4949989	24	1.91	23	82.6	8.7	60	8.7
	4958216	4958251	17	2.06	17	89.5	5.3	56	5.6
	4959068	4959120	28	1.93	27	75	17.9	51	12.7
	4971525	4971564	21	1.9	21	78.9	0	52	10
	4989489	4989540	27	2.04	26	75	10.7	53	13
	4991664	4991722	30	2.03	29	74.2	6.5	60	13.3
+	4992407	4992442	18	2	18	83.3	0	51	8.3
	4994879	4994914	18	2	18	83.3	0	51	8.3
	4996179	4996225	24	1.96	24	75	8.3	50	12.5
	5010160	5010200	21	1.91	22	80	5	54	9.5
#+	5061675	5061729	6	9.17	6	98	0	103	1.8
	5067198	5067240	22	1.95	22	76.2	0	51	11.6
	5072803	5072849	25	2.04	23	80.8	15.4	55	10.2
	5076628	5076683	32	1.9	30	76.9	7.7	59	11.9
	5079547	5079588	22	1.91	23	81	9.5	56	9.1
	5080347	5080398	29	1.93	27	76	8	51	13
	5087098	5087137	21	1.95	20	78.9	0	50	10
	5097061	5097099	21	1.9	21	77.8	0	50	10
	5109003	5109049	22	2.09	22	76.9	11.5	55	10.4
	5110249	5110288	21	1.9	21	78.9	0	52	10
	5122056	5122095	19	2.16	19	77.3	4.5	52	9.8
+	5127652	5127695	21	2.05	21	78.3	4.3	51	11.4
+	5144492	5144533	21	1.95	20	81.8	13.6	50	9.5
	5147326	5147373	25	1.92	25	75	8.3	52	12.2
	5147420	5147480	32	1.94	32	73.3	3.3	66	12.9
	5154845	5154896	25	2.08	25	75	7.1	53	13.2
	5166852	5166899	25	1.92	24	75	8.3	50	12.5
#+	5167108	5167133	9	2.89	9	100	0	52	0
	5168935	5168988	30	1.93	28	75.9	20.7	53	12.3
	5171303	5171364	35	1.91	32	75	15.6	62	12.5
	5180652	5180846	94	2.02	95	83.5	8.7	266	8.1
	5184052	5184106	27	2.07	27	72.4	3.4	54	14.3
	5188286	5188325	20	1.95	20	80	5	50	10
	5190106	5190158	27	2.08	26	75.9	10.3	55	12.7
	5199352	5199407	28	1.93	29	75.9	13.8	59	12.1
	5202066	5202114	27	1.92	26	77.3	13.6	52	11.5
	5205521	5205563	21	2.05	22	78.3	8.7	51	11.1
	5226565	5226605	21	1.95	21	81	9.5	52	9.5
#+	5227081	5227116	9	4	9	92.6	0	58	5.6
	5229783	5229833	27	1.96	26	77.8	14.8	56	11.3
#+	5234811	5234848	18	2.17	18	85	0	55	7.7
	5235091	5235134	23	1.91	23	76.2	4.8	51	11.1
#+	5246049	5246385	112	3.02	110	91.7	3.1	508	6.5
	5248544	5248602	30	1.94	31	75.9	3.4	69	11.7
	5263282	5263336	28	1.96	27	75	7.1	57	12.7
	5268160	5268223	32	2.06	31	74.3	11.4	61	13.6
#+	5277382	5277422	15	2.73	15	80.8	0	54	9.8
	5281487	5281542	28	2	29	75.9	6.9	63	12.1
	5286256	5286294	19	2.11	19	76.2	4.8	50	10
	5287055	5287106	27	1.96	26	73.1	3.8	53	13.5

	Start	End	Period Size	Copy Number	Consensus Size	Percent Matches	Percent Indels	Score	Percent Consensus Errors
#+	5295259	5295313	17	3.24	17	97.4	0	103	1.8
	5318248	5318288	21	1.9	21	80	5	52	9.8
	5323695	5323740	24	1.92	24	72.7	0	50	13
	5325640	5325694	28	2	28	76.7	16.7	57	12.1
#+	5331530	5331726	18	11.06	18	92.8	1.1	168	16
	5332338	5332385	21	2.24	21	77.8	3.7	52	12.5
	5332383	5332422	16	2.62	16	88	4	52	9.5
#+	5339708	5340025	96	3.29	96	90.1	1.8	490	6.3
	5354955	5355011	30	1.97	29	73.3	13.3	54	13.6
#+	5355678	5355728	21	2.43	21	93.3	0	88	3.9
	5378478	5378534	31	1.9	30	75	7.1	63	12.1
	5385225	5385268	21	2.05	21	78.3	4.3	51	11.4
	5387552	5387594	23	1.91	23	76.2	4.8	51	11.4
#+	5424366	5424470	24	4.38	24	82.7	0	105	14.3
	5424450	5424583	24	5.58	24	79.1	0	149	12.7
#+	5424895	5424931	18	2.06	18	89.5	0	60	5.4
#+	5426061	5426110	6	8.33	6	100	0	100	0
	5431585	5431633	23	2.04	24	76.9	7.7	54	12
#+	5432831	5432887	25	2.28	25	100	0	114	0
+	5434746	5434791	23	1.96	23	78.3	4.3	55	10.9
	5444232	5444278	24	1.92	25	73.9	4.3	52	12.5
	5448603	5448648	24	1.96	24	73.9	4.3	50	12.8
#	5454852	5454892	19	2.05	20	81.8	9.1	52	9.5
#	5472324	5472519	101	1.94	101	88.4	0	315	5.6
	5472803	5472851	25	1.92	25	76.9	19.2	50	11.8
	5482094	5482147	29	1.93	28	75	14.3	55	12.5

Bibliography

[1] Altschul, S., Gish, W., Miller, W., Myers, E. and Lipman, D. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[2] Apostolico, A. Fast parallel detection of squares in strings. *Algorithmica*, 8:285–319, 1992.

[3] Apostolico, A. and Preparata, F.P. Optimal off-line detection of repetitions in a string. *Theoret. Comput. Sci.*, 22(3):297–315, 1983.

[4] Beckman, J., and Soller, M. Toward a unified approach to genetic mapping of eukaryotes based on sequence tagged microsatellite sites. *Biotechnology*, 8:930–932, 1990.

[5] Benson, G. A space efficient algorithm for finding the best nonoverlapping alignment score. *Theoret. Comput. Sci.*, 145:357–369, 1995.

[6] Benson, G. Tandem repeats finder: A program to analyze DNA sequences. *Nucl. Acids Res.*, 27:573–580, 1999.

[7] Benson, G. and Su, X. On the Distribution of K-tuple Matches for Sequence Homology: A Constant Time Exact Calculation of the Variance. *J. Comput. Biol.*, 5(1):87–100, 1998.

[8] Bowcock, A., Ruiz-Linares, A., Tomfohrde, J., Minch, E., Kidd, J., and Cavalli-Sforza, L. High resolution of human evolutionary trees with polymorphic microsatellites. *Nature*, 368:455–457, 1994.

[9] Butler, J.M. *Forensic DNA Typing: Biology and Technology Behind STR Markers.* Academic Press, London, UK., 2001.

[10] Campuzano, V., Montermini, L., Molt, M.D., Pianese, L., Cosse, M., Cavalcanti, F., Monros, E., Rodius, F., Duclos, F., Monticelli, A., Zara, F., Caizares, J., Koutnikova, H., Bidichandani, S., Gellera, C., Brice, A., Trouillas, P., De Michele, G., Filla, A., De Frutos, R., Palau, F., Patel, P.I., Di Donato, S., Mandel, J.-L., Cocozza, S., Koenig, M. and Pandolfo, M. Friedreich's Ataxia: Autosomal Recessive Disease Caused by an Intronic GAA Triplet Repeat Expansion. *Science*, 271(5254):1423–1427, 1996.

[11] Cohen, I., Liu, X., Schutz, C., White, B., Jenkins, E., Brown,W. and Holden, J. Association of autism severity with a monoamine oxidase A functional polymorphism. *Clin. Genet.*, 64:190–197, 2003.

[12] Crochemore, M. An optimal algorithm for computing the repetitions in a word. *Inform. Process. Lett.*, 12:244–250, 1981.

[13] Delgrange, O. and Rivals, E. STAR: An Algorithm to Search for Tandem Approximate Repeats. *Bioinformatics*, 20:2812–2820, 2004.

[14] Dokhylyan, N., Buldyrev, S., Halvin, S., and Stanley, H. Model of unequal chromosomal crossing over in DNA sequences. *Physica A*, 249:594–599, 1998.

[15] Feller, W. *An Introduction to Probability Theory and its Applications*, volume I. John Wiley & Sons, New York, NY., 3rd edition, 1968.

[16] Fischetti, V.A., Landau, G.M., Sellers, P.H. and Schmidt,J.P. Identifying periodic occurrences of a template with applications to protein structure. *Inf. Proc. Lett.*, 45:11–18, 1993.

[17] Fogu, G., Bertini, V., Dessole, S., Bandiera, P., Campus, P.M., Capobianco, G., Sanna, R., Soro, G. and Montella, A. Identification of a mutant allele of the androgen receptor gene in a family with androgen insensitivity syndrome: detection of carriers and prenatal diagnosis. *Arch. Gyn. Obstet.*, 269:25–29, 2003.

[18] Fu, Y-H, Pizzuti, A., Fenwick, R., King, J., Rajnarayan, S., Dunne, P.W., Dubel, J., Nasser, G.A., Ashizawa, T., de Jong, P., Wieringa, B., Korneluk, R., Perryman, M.B., Epstein, H.F., and Caskey, C.T. An unstable triplet repeat in a gene related to myotonic muscular dystrophy. *Science*, 255:1256–1258, 1992.

[19] Guerini, F.R., Ferrante, P., Losciale, L., Caputo, D., Lombardi, M.L., Pirozzi, G., Luongo, V., Sudomoina, M.A., Andreewski, T.V., Alekseenkov, A.D., Boiko, A.N., Gusev, E.I. and Favorova, O.O. Myelin basic protein gene is associated with MS in DR4- and DR5-positive Italians and Russians. *Neurology*, 61:520–526, 2003.

[20] Huntington's Disease Collaborative Research Group. A novel gene containing a trinucleotide repeat that is expanded and unstable on huntingtons disease chromosomes. *Cell*, 72:971–983, 1993.

[21] Inman, K., and Rudin, N. *An Introduction to Forensic DNA Analysis.* CRC press, Boca Raton, Florida., 1997.

[22] Jeffreys, A., Monckton, D., Tamaki, K., Neil, D., Armour, J., MacLeod, A., Collick, A., Allen, M., and Jobling, M. Minisatellite variant repeat mapping:

Application to DNA typing and mutation analysis. *In DNA Fingerprinting: State of the Science.*, pages 125–139, 1993.

[23] Jeffreys, A.J., Wilson, V. and Thein, S.L. Individual-specific fingerprints of human DNA. *Nature*, 316:76–79, 1985.

[24] Kannan, S.K. and Myers, E.W. An algorithm for locating nonoverlapping regions of maximum alignment score. *SIAM J. Comput.*, 25(3):648–662, 1996.

[25] Kolpakov, R. and Kucherov, G. Finding Approximate Repetitions under Hamming Distance. *Theoret. Comput. Sci.*, 303, 2003.

[26] La Spada, A., Wilson, E., Lubahn, D., Harding, A. and Fischbeck, K. Androgen receptor gene mutations in X-linked spinal and bulbar muscular atrophy. *Nature*, 352:77–79, 1991.

[27] Landau, G., Schmidt, J. and Sokol, D. An Algorithm for Approximate Tandem Repeats. *J. Comput. Biol.*, 8(1):1–18, 2001.

[28] Leibovitch, B.A., Lu, Q., Benjamin, L.R., Liu, Y., Gilmour, D.S. and Elgin, S.C. GAGA factor and the TFIID complex collaborate in generating an open chromatin structure at the Drosophila melanogaster hsp26 promoter. *Mol. Cell. Biol.*, 22:6148–6157, 2002.

[29] Levenshtein, V.I. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.*, 6:707–710, 1966.

[30] Lewin, B. *Genes VI.* Oxford University Press, New York, 1997.

[31] Licastro, F., Grimaldi, L.M., Bonafe, M., Martina, C., Olivieri, F., Cavallone, L., Giovanietti, S., Masliah, E. and Franceschi, C. Interleukin-6 gene alleles affect the risk of Alzheimers disease and levels of the cytokine in blood and brain. *Neurobiol. Aging*, 24:921–926, 2003.

[32] Main, M. and Lorentz, R. An O(n log n) algorithm for finding all repetitions in a string. *J. Algorithms*, 5:422–432, 1984.

[33] Majewski, J. and Ott, J. GT Repeats Are Associated with Recombination on Human Chromosome 22. *Genome Res.*, 10(8):1108–1114, 2000.

[34] Milosavljevic, A. and Jurka, J. Discovering simple DNA sequences by the algorithmic significance method. *Comput. Appl. Biosci.*, 9(4):407–411, 1993.

[35] Nakamura, Y., Leppert, M., OConnell, P., Wolff, R., Holm, T., Culver, M., Martin, C., Fujimoto, E., Hoff, M., Kumlin, E. and White, R. Variable number of tandem repeat (VNTR) markers for human gene mapping. *Science*, 235:1616–1622, 1987.

[36] Richards, R.I., Holman, K., Yu, S. and Sutherland, G.R. Fragile X syndrome unstable element, p(CCG)n, and other simple tandem repeat sequences are binding-sites for specific nuclear proteins. *Hum. Mol. Genet.*, 2:1429–1435, 1993.

[37] Rigoutsos, I. and Floratos, A. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics*, 14:55–67, 1998.

[38] Rigoutsos, I. and Floratos, A. Motif discovery without alignment or enumeration (extended abstract). In *RECOMB*, pages 221–227, 1998.

[39] Sagot, M.-F. and Myers, E.W. Identifying satellites and periodic repetitions in biological sequences. *J. Comput. Biol.*, 5(3):539–554, 1998.

[40] Schmidt, J.P. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM J. Comput.*, 27(4).

[41] Stewart, M., and Denell, R. The drosophila ribosomal protein s6 gene includes a 3' triplication that arose by unequal crossing-over. *Molecular Biology and Evolution*, 10(5):1041–1047, 1993.

[42] Stolovitzky, G., Gao, Y., Floratos, A. and Rigoutsos I. Tandem repeat detection using pattern discovery, with applications to the identification of yeast satellites. Technical report, IBM T. J. Watson Research Center, 1999.

[43] Stoye, J. and Gusfield, D. Simple and flexible detection of contiguous repeats using a suffix tree. *Theoret. Comput. Sci.*, 270:843–856, 2002.

[44] Verkerk, A., Pieretti, M., Sutcliffe, J., Fu, Y., Kuhl, D., Pizzuti, A., Reiner, O., Richards, S., Victoria, M., Zhang, F., Eussen, B., Vanommen, G., Blonden, L., Riggins, G., Kunst, C., Galjaard, H., Caskey, C., Nelson, D., Oostra, B. and Warren, S. . Identification of a gene (FMR-1) containing a CGG repeat coincident with a breakpoint cluster region exhibiting length variation in fragile X syndrome. *Cell*, 65:905–914, 1991.

[45] Wexler, Y., Yakhini, Z., Kashi, Y. and Geiger, D. Finding approximate tandem repeats in genomic sequences. *J. Comput. Biol.*, 12(7):928–942, 2005.

[46] Woo, K.J., Sang-Ho, K., and Jae-Kwan, C. Association of the dopamine transporter gene with Parkinsons disease in Korean patients. *Korean Medical Science*, 15(4), 2000.