HÖGSKOLAN SKÖVDE

School of Humanities and Informatics
Final Year Project in Computer Science 30 ECTS
Advanced Level 2
Spring Term 2005

Detection and analysis of megasatellites in the human genome using *in silico* methods

Elís Ingi Benediktsson

HS-IKI-MD-05-204

Detection and analysis of megasatellites in the human genome using *in silico* methods

Submitted by Elís Ingi Benediktsson to the University of Skövde as a dissertation towards the degree of M.Sc. by examination and dissertation in the School of Humanities and Informatics.

2005-06-06

I hereby certify that all material in this dissertation, which is not my own work, has been identified and that no material is included for which a degree has previously been conferred upon me.

Signed: _____

Supervisor at the University of Skövde: Jane Synnergren

Supervisor at deCODE Genetics Inc.: Gísli Másson

To my beloved family

Detection and analysis of megasatellites in the human genome using *in silico* methods

Elís Ingi Benediktsson

Abstract

Megasatellites are polymorphic tandem repetitive sequences with repeat-units longer than or equal to 1000 base pairs. The novel algorithm Megasatfinder predicts megasatellites in the human genome. A structured method of analysing the algorithm is developed and conducted. The analysis method consists of six test scenarios. Scripts are created, which execute the algorithm using various parameter settings. Three nucleotide sequences are applied; a real sequence extracted from the human genome and two random sequences, generated using different base probabilities. Usability and accuracy are investigated, providing the user with confidence in the algorithm and its output. The results indicate that Megasatfinder is an excellent tool for the detection of megasatellites and that the generated results are highly reliable. The results of the complete analysis suggest alterations in the default parameter settings, presented as user guidelines, and state that artificially generated sequences are not applicable as models for real DNA in computational simulations.

Keywords: Genomic variation, repetitive sequences, tandem repeats, polymorphism, satellite DNA, megasatellites, Megasatfinder, *in silico* prediction, algorithm analysis method.

Table of contents

1 Introduction 1

2 Background 5

2.1 The genome and the central dogma of molecular biology 5

2.2 Tandem repeats 6

2.3 Satellite DNA 7

2.4 Megasatellites 8

2.4.1 The megasatellite RS447 8

2.4.2 The megasatellite D4Z4 9

2.5 The algorithm 10

2.5.1 The steps of the algorithm 10

2.5.2 The parameters in the algorithm 13

2.6 How are the presumptive megasatellites confirmed? 14

3 Problem statement 16

3.1 Problem description 16

3.2 Project aims and objectives 16

3.3 Project hypothesis 17

4 Related work 18

5 Experimental approach 21

5.1 The extraction and creation of the nucleotide sequences 21

5.1.1 The extraction of real nucleotide sequences from the human genome 21

5.1.2 The creation of random nucleotide sequences 21

5.2 The test scenarios constituting the analysis method 22

5.3 The scripts used for running the algorithm 23

5.3.1 megasat_creator_finder.py and megasat_creator_finder_mutation.py 23

5.3.2 megasat_finder_nomegs_random.py and megasat_finder_nomegs_DNA.py 24

5.3.3 megaresults_meanfinder.py and megaresults_mut_meanfinder.py 24

5.3.4 megaresults_clusterfinder.py 24

5.3.5 megasat_random_generator.py 25

5.3.6 wrapper.py 25

5.3.7 wrapper_6_mutations.py and wrapper_6_mutations_lvt.py 25

5.3.8 wrapper_nomegs_random.py and wrapper_nomegs_DNA.py 25

5.4 The various tools used during the analysis process 26

5.4.1 Gnuplot 26

5.4.2 Python 26

5.4.3 Msbar 26

6 Experimental results 27

6.1 False positive hit statistics for a random sequence 27

6.2 False positive hit statistics for a DNA sequence 28

6.3 Sensitivity analysis for repeat-unit sizes 29

6.4 Repeat-unit sensitivity analysis for mutated megasatellites 33

6.4.1 Mutating the artificial megasatellites 33

6.4.2 Altering the length variation tolerance (l.v.t.) 36

6.5 An example of a novel, unreported megasatellite, found by extensive genomic scans and confirmed by southern blotting 37

6.6 Complexity analysis 39

7 Analysis and discussion 42

7.1 General interpretation of the results 42

7.2 Generality of results 45

7.3 Algorithm improvements 46

7.4 Guidelines for algorithm usage 47

8 Conclusions 48

8.1 Conclusions 48

8.2 Future work 48

Acknowledgements 50

References 51

Lists of appendixes, figures, and tables

List of appendixes:

Appendix A A detailed description of the test scenarios .....
Appendix B The script megasat_creator_finder.py .....
Appendix C The script megasat_creator_finder_mutation.py.....
Appendix D The script megasat_finder_nomegs_random.py .....
Appendix E The script megasat_finder_nomegs_DNA.py.....
Appendix F The script megaresults_meanfinder.py .....
Appendix G The script megaresults_mut_meanfinder.py.....
Appendix H The script megaresults_clusterfinder.py.....
Appendix I The script megasat_random_generator.py .....
Appendix J The script wrapper.py.....
Appendix K The script wrapper_6_mutations.py.....
Appendix L The script wrapper_6_mutations_lvt.py .....
Appendix M The script wrapper_nomegs_random.py.....
Appendix N The script wrapper_nomegs_DNA.py.....
Appendix O Figures and tables from Chapter 6 .....

List of figures:

Figure 2.1 The central dogma of molecular biology.....6
Figure 2.2 A graphical illustration of each step in Megasatfinder.....12
Figure 2.3 Southern blotting confirms or rejects predicted megasatellites. 15
Figure 6.1 Mean sensitivity analysis for DNA sequences, showing the peak.....30
Figure 6.2 Mean sensitivity analysis for random sequences, showing the peak 31
Figure 6.3 Mean sensitivity analysis for various mutation levels, showing the peak.....35
Figure 6.4 A snapshot of the lstr104 megasatellite, the ZFP37 gene, and CpG-islands...38
Figure 6.5 A self-sequence comparison, conducted for the megasatellite lstr104 38
Figure 6.6 An alignment of sequences from the human genomic assembly (Build 34)...39
Figure O.1 Mean sensitivity analysis for DNA sequences, showing the fade-out .....
Figure O.2 Mean sensitivity analysis for random sequences, showing the fade-out.....
Figure O.3 Mean sensitivity analysis for various mutation levels, showing the baseline.....

List of tables:

Table 6.1 Number of clusters found, using various l.v.t. for a random sequence 27
Table 6.2 Number of clusters found, using various l.v.t. for a DNA sequence.....28
Table 6.3 Expressed maximum sensitivity and fade-out per probe size 32
Table 6.4 Mean success rates for Megasatfinder, using a DNA sequence.....32
Table 6.5 Mean success rates for Megasatfinder, using a random sequence 33
Table 6.6 Expressed maximum and baseline sensitivity for various mutation levels 35

Table 6.7 Mean mutational success rates for Megasatfinder, using a DNA sequence 36
Table 6.8 Mean mutational success rates for Megasatfinder, using a random sequence.. 36
Table 6.9 Cluster count for various l.v.t. and mutation levels, using a DNA sequence ... 36
Table 6.10 Cluster count for various l.v.t. and mutation levels, using a random seq. 37
Table 6.11 Time complexity for a complete genomic scan, using all probe sizes 40
Table 6.12 Time complexity, using a short DNA sequence and various parameters 40
Table 6.13 Time complexity, using a DNA sequence, applying various mutation levels 41
Table 7.1 The suggested set of default parameter settings for Megasatfinder …..……...47
Table O.1 Full version of Table 6.1 .....

1 Introduction

The human genome has intrigued scientists throughout the years. A formal definition of the term *genome* has been proposed by scholars of several disciplines from the time Mendel began his pioneer research in genetics. The most common definition is that a genome represents the total DNA (*D*eoxyribo*N*ucleic *A*cid) present in the nucleus of each cell of an organism. The organism in the context of this report is the human being (*Homo sapiens*). Parts of this everlasting interest among scientists are curiosity and strive to seek answers to the unknown, which has always been engraved in human nature. The human genome can be compared to a huge jigsaw puzzle, where some pieces are missing and some do not seem to fit at first sight. A paramount step towards increased knowledge was taken in 1990 when the Human Genome Project (HGP) formally began.

The HGP was coordinated by the U.S. Department of Energy and the National Institutes of Health. Originally, the project was estimated to last for 15 years, but due to brisk advances in technology it was completed in 2003. Among the goals put forward by the HGP was to identify the approximately 30,000 genes (exact number unknown) in the human DNA assembly and also to determine the sequence of the 3 billion base pairs (bp) which the human genome is comprised of (Sawicki et al., 1993). Although the HGP is formally finished, an enormous effort is required regarding the systematic analysis of the generated data. Thus, researchers will continue their work in positioning the pieces in the correct order to solve the puzzle of life.

During the decoding of the human genome an important discovery was made regarding the many forms of variation present in the genomic assembly. Among these genetic variations are single-nucleotide polymorphisms (substitutions), small insertion-deletion polymorphisms (often referred to as indels), variable numbers of repetitive sequences, and genomic structural alterations (Iafrate et al., 2004). These variations can have a substantial effect on the genomic content and thus on the individuals in question. The effects of genomic variation can be anomalies of several forms and degrees, e.g. several diseases. Among these variations, the repetitive sequences deserve specific interest, since as much as 50% of the eukaryotic genome, and thus the human genome, consists of various types of these sequences (Benson & Waterman, 1994; Lander et al., 2001; Näslund et al., 2005). Unfortunately, the exact function of repetitive DNA is not well understood, as reported by several papers concerning the subject area. According to Saitoh et al. (2000), it is lucid that the discovery of functions associated with repetitive DNA will aid researchers to increase understanding of diseases and their causality, associations between structure and function with regard to genomic architecture, genomic recombination, and possibly even the evolution of multicellular organisms (Nowak, 1994).

Various types of repetitive sequences have been reported in the literature and categorised according to several factors. Among these factors are repeat-unit size and copy number (Gondo et al., 1998; Okada et al., 2002). Tandem repeats (head-to-tail tandem reiterated DNA sequences) can be categorised as a subset of the repetitive sequences. Another subset contains the ordinary repeats, i.e. the ones which do not follow the tandem order. The formation of tandem repeats can be explained as tandem duplication, where a fragment of a DNA sequence is multiplied into two or more copies. Each copy follows

the foregoing one in a contiguous manner, thus the term *tandem* (Benson, 1999). These tandem repeats are present in the genomes of all living organisms (Näslund et al., 2005), but in this project the main focus is directed at the human genome. Although much is unknown regarding the exact function, behaviour, and role of tandem repeats, they have been connected to several important functionalities in humans and other eukaryotes (Benson, 1999). One of these roles is connected to gene regulation, where the repeats can have various effects (Benson, 1999; Hamada et al., 1984; Lu et al., 1993; Pardue et al., 1987; Richards et al., 1993; Yee et al., 1991). Furthermore, tandem repeats are widely applied in linkage analysis and DNA fingerprinting (e.g. used in medical forensics), due to the polymorphism of tandem repeat copy number in the population (Edwards et al., 1992; Weber & May, 1989).

Polymorphic tandem repetitive elements are referred to as *satellite DNA*. Megasatellites are the longest satellite DNA sequences, and according to their definition, which is arbitrary in the literature, have repeat-units that are longer than or equal to 1000 bp (Gondo et al., 1998; Saitoh et al., 2000). As indicated by the name of this project, the focus will be to study the detection and analysis of these sequences in the human genome. A low quantity of megasatellites, or only approximately five in total, has been reported in the literature. One part of the reason is the fact that tandem repeats, including megasatellites, have neither been systematically detected nor annotated in the various genome projects (Delgrange & Rivals, 2004). The other part is due to the degree of uncertainty in detection and verification processes, which can be rather high at times. Examples of the most acknowledged and studied of these are the megasatellites *RS447* (Gondo et al., 1998; Kogi et al., 1997; Okada et al., 2002; Saitoh et al., 2000) and *D4Z4* (Lemmers et al., 2004; Lyle et al., 1995; van Geel et al., 1999).

The aim of this project is to develop and conduct a structured method of analysing the megasatellite detection algorithm *Megasatfinder* in its present state. The algorithm will be tested with regard to various aspects. This involves analysing the general function of the algorithm, along with the effect that variations in parameter settings have on the resulting output. This work requires the systematic extraction and creation of both real and simulated nucleotide sequence data, structured means of setting up meaningful test scenarios, design of graphical visualisation and appropriate means of analysing the acquired results. Various runs, where specific optimising aspects are under investigation, are carried out for the purpose of gathering results data. The task underlying this work is to investigate the usability and accuracy of the algorithm, thus providing users with confidence and knowledge of algorithm applicability. The definition of the term *usability* in this context is directed towards the user. Questions like: “Which results can be expected when applying the algorithm on a specific sequence?”, “Is the algorithm applicable to all types of sequences?”, “Which parameter settings should be applied when searching for megasatellites?”, “What are the effects of increasing the variation between the repeat units in a megasatellite (i.e. the effects of mutations)?”, and “How long time does it take to run the algorithm on the whole genome?”; are questions which describe the meaning behind the term in this context. The definition of the term *accuracy* in this context is directed towards the generated output of the algorithm. Questions like: “To what extent can the results be trusted?”, “How many hits are false positive?”, “How accurate are the results?”; are questions which describe the meaning behind the term in this context. Together, these terms decide the applicability of the algorithm regarding the

detection of megasatellites in the human genome. Furthermore, the work will hopefully result in maximisation of the two concepts usability and accuracy from the user's point of view. This means that guidelines will be created for the use of the algorithm under specific circumstances concerning parameter settings, and might involve alterations in the implementation of the algorithm depending on the result of the complete analysis. These alterations could e.g. include various changes and/or extensions to the algorithm. An important part of the work is to increase the understanding of the output of the algorithm in general. This will hopefully lead to some ranking of the generated results.

The algorithm was written in the autumn of 2004 by Dr. Gísli Másson at the department of Data Analysis and Management (DAM) at deCODE Genetics Inc., Iceland. Due to the recency of the algorithm no results have been published yet. In addition, no evaluation of Megasatfinder has previously been performed, thereby making Megasatfinder even more interesting to work with. The task of the algorithm is to detect possible megasatellite sites in the human genome by applying *in silico* methods, without making any assumptions about the type or size of repeat-units. Although the algorithm was developed for the detection of megasatellites within the human genome, it was not tuned specifically for the human genome. The phrase *in silico* refers to the fact that the detection process is performed using computational simulations, as opposed to detection performed in the laboratory, which would be referred to as an *in vitro* detection process. The algorithm is comprised of seven steps which lead to the identification of possible megasatellites in genomic sequences. Megasatfinder contains a number of adjustable parameters, which affect the resulting output. The final step in the detection process is the verification of the resulting hits, i.e. to either confirm or reject the suggested sites. This verification is based on further studies performed in the laboratory.

In general, improved knowledge regarding the effects of DNA variation among individuals can lead to new evolutionary methods to diagnose, treat, and perhaps someday prevent the thousands of disorders and anomalies that affect humans. Megasatellite sites found in the human genome may very well be functional and therefore associated to disease, as is the case with the two megasatellites RS447 and D4Z4 mentioned earlier. Detecting and investigating the possible megasatellite sites in the human genome is therefore of high importance in order to be able to track possible genetic influences. Another problematic issue is that ordinary genotyping methods fail on megasatellite regions. This presents problems in the detection process and implies the need for other methods, applicable to this category of very long satellite DNA. Megasatellite repeats may be underrepresented in the genomic assembly due to the techniques used to sequence and assemble the genome. In particular, megasatellite repeats of highly homologous or identical elements will probably be represented as exactly two copies in the sequence. Further reasons why this work is important concerns the algorithm in general. As discussed above, the performed analysis will provide a window of knowledge regarding the performance and the limitations of the algorithm. This accumulated knowledge will assist in improving the functionality of the algorithm, thus making it a better tool in the search for possible megasatellites. Given these imperative identified and potential biological roles, increased knowledge and interest, along with a good megasatellite detection algorithm, might induce further study of megasatellites.

It is important to realise that applying an algorithm without the knowledge of its usability, applicability or accuracy implies a high degree of uncertainty in the generated results and their interpretation. Furthermore, it substantially increases the risk of faulty results or conclusions. Using these results in further work, e.g. in the laboratory, might involve superfluous work efforts, which in addition could return meaningless results. Thus, the use of an untested tool would undoubtedly bring about high risks and unnecessary expenses for the stakeholders and that highly motivates this work.

The report is structured as follows: The following chapter discusses the background in relation to topics such as tandem repeats and satellite DNA. Chapter 3 presents the project definition and hypothesis, along with the aim and objectives set up for the project. The subsequent chapter describes the work of others, related to the work in this project. Materials and methods can be found in Chapter 5 where various details of the experimental setup will be described. The sixth chapter presents the results obtained and the analysis and discussion can be found in Chapter 7, where a closer look will be taken on the meaning of the results. Chapter 8 finally concludes the work.

2 Background

2.1 The genome and the central dogma of molecular biology

The genome comprises the entire genetic material of an organism. It represents the total DNA present in the nucleus of each cell of an organism. DNA is structured as linearly linked *nucleotides*. This sequence of nucleotides forms the genetic information used as a blue print for the building of proteins. DNA is structured as a double helix, i.e. two interwoven strands of the nucleotides, along with one of the four bases adenine (A), thymine (T), guanine (G), and cytosine (C). The two strands are complementary and connected to each other by various bonds, where an A and T are complementary and G and C are complementary. The DNA material is organised into structures known as *chromosomes*. The complete set of chromosomes in the cells of an organism is referred to as its *karyotype*. The karyotype of humans (*Homo sapiens*) contains 23 pairs of homologous chromosomes. The breakdown in females is 22 pairs of autosomes and a single pair of X chromosomes. The breakdown for males consists of the same 22 pairs of autosomes, but one X and one Y chromosome, as opposed to the pair of X chromosomes in females. The X and Y chromosomes are referred to as the sex chromosomes. There are approximately 30,000 genes in the human genomic assembly and these are located heterogeneously on the chromosomes. Each gene codes for the creation of one or more proteins. Proteins are important for the normal functions of cells and serve as building blocks in the body (Campbell et al., 1999).

The central dogma of molecular biology describes the process leading from the DNA double helix to the formation of a functional protein (see Figure 2.1). Prior to cell division, each cell must contain all the genetic information and to ensure this the first step is to replicate the DNA. When signals arrive, indicating the shortage of certain proteins, the corresponding genes are transcribed into RNA (*R*ibo*N*ucleic *A*cid). The next step in the process is post-transcriptional modification, performed primarily by splicing, where the non-coding sequences (introns) are removed from the transcript, leaving the coding sequences (exons) behind. The remaining transcript then migrates out of the nucleus and into the cell's cytoplasm. When the transcript is out of the cell nucleus it is translated into a protein, based upon the amino acid code in the RNA (Campbell et al., 1999).

One should bear in mind that the above description is extremely simplified. A lot of cell organs, enzymes, and other factors are involved in this highly complex and constantly ongoing process within each cell in the body. As can be seen from the above discussion, tandem repetitive sequences can play a major role in the development of proteins. If the genomic sequences (DNA) are altered in any way, in this case by the decrease or increase of tandem repetitive sequences such as megasatellites, this will undoubtedly affect the transcription into RNA and ultimately the translation into proteins. This can in turn cause the various anomalies associated to these mutations in the genome.

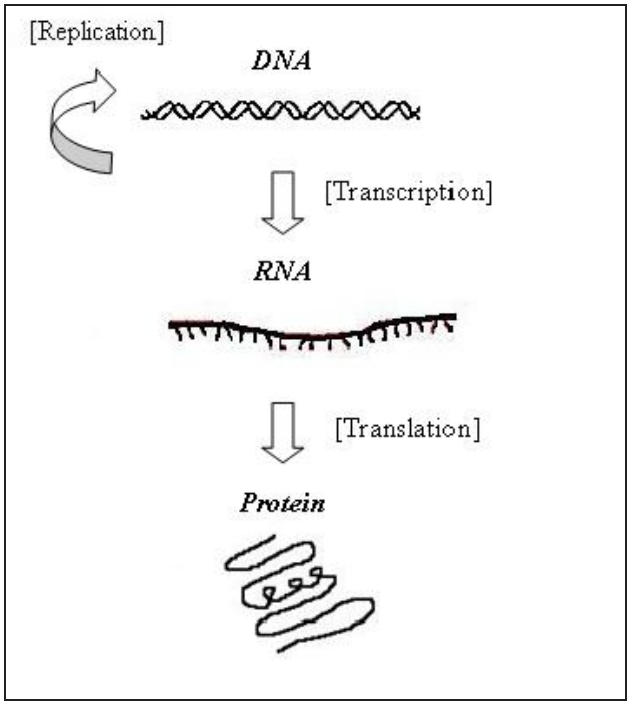


Figure 2.1 The central dogma of molecular biology.

2.2 Tandem repeats

Tandem repeats are defined as consecutive perfect or slightly imperfect copies of DNA motifs of variable lengths (Charlesworth et al., 1994). Benson (1999) describes tandem repeats in DNA as two or more contiguous and approximate copies of a pattern of nucleotides. The reason for the use of the word *approximate* in the definition by Benson (1995, 1999) is that the individual copies positioned within a particular tandem repeat can in time undertake some evolutionally uncoordinated mutations, which affect the DNA sequence. These mutations involve nucleotide substitutions, insertions and/or deletions.

Although much is unknown regarding the exact function, behaviour, and role of tandem repeats, they have been connected to several important functionalities in humans and other eukaryotes (Benson, 1999). One of these roles is connected to gene regulation, where the repeats can have various effects. The tandem repeats can e.g. interact with transcription factors (TFs), act as protein binding sites, or alter the structure of chromatin (a complex of DNA and histone proteins, the building material of chromosomes) (Benson, 1999; Hamada et al., 1984; Lu et al., 1993; Pardue et al., 1987; Richards et al., 1993; Yee et al., 1991). In addition, tandem repeats have according to Benson (1999) been shown to have an apparent function in the development of cells belonging to the immune system. Furthermore, tandem repeats are widely applied in linkage analysis and DNA fingerprinting (e.g. used in medical forensics), due to the polymorphism of the tandem repeat copy number in the population (Edwards et al., 1992; Weber & May, 1989).

Polymorphic tandem repeats are classified into some different categories. However, this classification is somewhat arbitrary in the literature. The term *polymorphic* means that different individuals constituting a population have a variant number of repeat-units at a specific site on a chromosome (Benson, 1995; Näslund et al., 2005). Tandem repeats can

also be *monomorphic*, meaning that all individuals in a population have the same number of repeats (Benson, 1995; Näslund et al., 2005).

2.3 Satellite DNA

Polymorphic tandem repetitive elements are referred to as *satellite DNA*, since after applying ultracentrifugation, they appeared as "satellite" bands in the centrifuge tubes, separated from the rest of the genomic DNA (Campbell et al., 1999; Charlesworth et al., 1994). The shortest tandem repeats containing 1-5 base pairs in each unit are named *microsatellites* (Tautz & Schlötterer, 1994); the next category is referred to as *minisatellites* (6-100 bp) (Bois et al., 1998; Tautz, 1993; Vergnaud & Denoeud, 2000); then there is one category labelled *macrosatellites* (101-999 bp) (Gondo et al., 1998); and finally the *megasatellites*, which according to Gondo et al. (1998) and Saitoh et al. (2000) have a repeat-unit larger than or equal to 1000 bp in length. According to Okada et al. (2002), it has been hypothesised that these satellite DNA sequences are considerably involved in genomic instability, due to their high mutation frequencies.

Among the categories described above, the micro- and minisatellites are the ones that have been studied the most, while the megasatellites belong to a newer area of interest among researchers. There are mainly two reasons for this unidirectional attention: The first is the incapability to easily detect megasatellites in genomic sequences due to their size, thus restraining the accurate characterisation of their properties; the second is that since the detection of the so called *trinucleotide repeat diseases* (TRDs), the interest has been diverted from the longer tandem repeats. When the copy number of tandem repeats of size 3 bp is altered, i.e. either decreased or increased, at specific chromosomal sites in the genome, the effect can be the onset of a TRD. An example of these is Huntington's disease (Benson, 1999; Kolpakov et al., 2003; Nag, 2003). It is important to bear in mind that the categorisation is arbitrary in the literature and thus slightly simplified in the above classification, but hopefully some consensus will be achieved in the near future.

According to Charlesworth et al. (1994), our knowledge of the forces that direct the evolution of satellite DNA is scarce. Their explanation for this limited knowledge is based on the large size of the clusters of satellite DNA, which prevent direct experimental analysis of these sequences. Furthermore, the large sizes hamper the collection of meaningful population data. Nevertheless, it has been elucidated that the advent of most of these tandem repeats is due to replication slippage or specific recombination events (Kolpakov et al., 2003). According to Kolpakov et al. (2003), these recombination events can involve unequal crossover and unequal sister chromatid exchange (see Axelrod et al., 1994). In addition, Charlesworth et al. (1994) explain that in several cases the repetitive sequences appear to be maintained exclusively by their ability to replicate themselves within the human genome, which occasionally presents significant fitness losses to the individual, such as the onset of a disease. This ability of self-replication within the genome is sometimes referred to as the "selfish DNA"-hypothesis.

2.4 Megasatellites

A megasatellite is a polymorphic tandem repetitive sequence with a repeat-unit longer than or equal to 1000 bp. Because of the arbitrariness in the literature, the term *megasatellite* is not recognised as general consensus, although it is widely used for this group of the longest satellite DNA. At deCODE Genetics Inc., these sequences have been referred to as *long-segment tandem repeats*, or LSTRs for short. There is however an interpretation difference between these two terms. An LSTR does not become a megasatellite until it is proven to be polymorphic by research studies performed in the laboratory. If the detected LSTRs are proven to be monomorphic, they are not referred to as megasatellites.

As for repetitive DNA in general, the exact biological function of megasatellites is not well understood. According to Saitoh et al. (2000), it is however recognised that some are accountable for the formation of the highly condensed heterochromatin. The heterochromatin part of the genome is characterised by relatively low gene density and chromosomes in this region are transcriptionally inactive.

In most natural disciplines the degree of uncertainty in detection and verification processes can be rather high. The degree of uncertainty for these processes with regard to possible megasatellites is no exception thereof. This is one of the reasons for the low quantity of known and verified megasatellites, as there are only approximately five recognised in the literature. A part of the reason is also the fact that tandem repeats, including megasatellites, have neither been systematically detected nor annotated in the various genome projects (Delgrange & Rivals, 2004). Examples of the most acknowledged and studied of these are the megasatellites *RS447* (Gondo et al., 1998; Kogi et al., 1997; Okada et al., 2002; Saitoh et al., 2000) and *D4Z4* (Lemmers et al., 2004; Lyle et al., 1995; van Geel et al., 1999). These two megasatellites and their influences in the genome will be described in the following subchapters (2.4.1-2.4.2).

2.4.1 The megasatellite RS447

The megasatellite RS447 was discovered by Kogi et al. (1997). The megasatellite has a repeat-unit size of 4746 bp and comprises a putative open reading frame (ORF) of 1590 bp. This sequence has an estimated copy number of 50-70 per haploid genome. According to Gondo et al. (1998) and Kogi et al. (1997), the megasatellite was found to reside on human chromosome 4p15 (chromosome 4, strand p, band 15). This was elucidated by applying Southern blotting (often called “zoo blot” hybridisation) and FISH (Fluorescence *in situ* hybridisation). Later investigation by Okada et al. (2002), using high-resolution FISH analysis, revealed that the RS447 locus was located on 4p16.1 instead of 4p15. Further research by Gondo et al. (1998) and Okada et al. (2002) resulted in the findings of another polymorphic tandem repeat consisting of several RS447 copies on the distal part of chromosome 8p (8p23). Okada et al. (2002) referred to the megasatellite on chromosome 4p16.1 as *major* RS447, and the megasatellite on chromosome 8p23 as *minor* RS447. Their research also confirmed that the RS447 tandem repeats were both hypervariable and polymorphic in the human population. Their conclusion is that RS447 does not appear to be either “selfish” or “junk” DNA, as is the case with many other repetitive sequences, due to the strong conservation and the putative large ORF of 1590 bp mentioned earlier. These results of Gondo et al. (1998) imply biological significance of the megasatellite. Megasatellites, such as RS447, are

known to form large domains at specific chromosomal sites in the genome (Giacalone et al., 1992; Müller et al., 1986; Saitoh et al., 2000; Saitoh et al., 1991; Tyler-Smith & Taylor, 1988).

Okada et al. (2002) postulate that the copy number and the status of methylation in megasatellite RS447 DNA may affect both the chromatin structure, as stated by Saitoh et al. (2000) above, and furthermore the expression of genes in the instant area. It has also been documented by Saitoh et al. (2000) that the RS447 tandem repetitive sequence encodes an intronless functional human gene and expresses a sense-transcript and an anti-sense-transcript of a de-ubiquitinating enzyme gene. This human de-ubiquitinating enzyme gene was labelled *USP17* (ubiquitin-specific protease 17). The function of USPs is e.g. to cleave ubiquitin from ubiquitin-protein conjugates or polyubiquitin. A connection has been established between a missense mutation in the ubiquitin carboxy-terminal hydrolase L1 (UCHL1) gene and the neurodegenerative Parkinson's disease in a German family (Leroy et al., 1998; McNaught et al., 2001). This mutation and the effect it has strongly indicate a connection between RS447 and the disease. It might be interesting to conduct further investigation on this subject in the future.

2.4.2 The megasatellite D4Z4

The megasatellite D4Z4 was described by Lyle et al. (1995) as a complex tandem repeat, with a repeat-unit size of 3.3 kb (kilobases), composed of several notorious sequence motifs. The sequence motifs include a double homeobox called *LSau* and *hhspm3* (Hewitt et al., 1994; Lyle et al., 1995; Winokur et al., 1994). A *homeobox* is a short nucleotide sequence, where the sequence is almost identical in the various genes that include it. According to Hewitt et al. (1994) and Lyle et al. (1995), the LSau homeobox contains an ORF when it is positioned within at least one D4Z4 repeat-unit copy, even though no ORF exists through the whole repeat-unit. Furthermore, LSau is associated with heterochromatic regions, i.e. the telomeres and centromeres, of the genome (Lyle et al., 1995; Meneveri et al., 1993), and hhspm3 is a low-copy GC-rich repeat (Lyle et al., 1995; Zhang et al., 1987). The megasatellite was assigned to human chromosome 4q35 by applying conventional linkage analysis (Lemmers et al., 2004; Lyle et al., 1995; van Geel et al., 1999). It has been established that this locus is involved in a disease referred to as *facioscapulohumeral muscular dystrophy*, or FSHD. As described by Lemmers et al. (2004), Lyle et al. (1995), and van Geel et al. (1999), FSHD is an autosomal dominant neuromuscular disorder. The characteristic effects for an individual suffering from FSHD are primarily progressive weakness and deterioration of the facial, shoulder girdle, and upper arm muscles; collectively referred to as muscular atrophy. The severity and age of onset of FSHD are variable, but nevertheless it has nearly complete penetrance (95%) by 20 years of age (Lunt & Harper, 1991; Lyle et al, 1995). According to van Geel et al. (1999), the frequency of FSHD is estimated to be 1 per 20,000 individuals.

It has been suggested by van Geel et al. (1999) that the D4Z4 repeat-units comprise a vital part of the structure of heterochromatin. As mentioned previously, this is also the case with RS447. Deletions of D4Z4 units infer alteration of chromatin structure and it is this reduction in unit number that affects the expression of nearby genes through a concept named *position effect variegation* (PEV) (Hewitt et al., 1994; van Geel et al., 1999; Winokur et al., 1994). The normal copy number of the D4Z4 repeat-unit on ordinary chromosomes varies between 11 and 100 (Lemmers et al., 2004; van Deutekom

et al., 1993). This number is significantly lower among affected individuals, or between 1 and 10 repeat-units (Lemmers et al., 2004; Wijmenga et al., 1992). In addition, an inverse correlation was observed between the number of repeat-units and the severity and age of onset of the disease (Lemmers et al., 2004; Lunt et al., 1995; Tawil et al., 1996). Despite this work, no specific causal genes have been discovered that can be associated with the FSHD phenotype (van Geel et al., 1999).

2.5 The algorithm

In order to fully comprehend the task, one must understand how this algorithm functions from input to output. The Megasatfinder algorithm was implemented in an object-oriented fashion using Python. Each step of the algorithm is explained in detail in the following subchapter (2.5.1). The subsequent subchapter (2.5.2) describes the parameters which affect the resulting output.

2.5.1 The steps of the algorithm

The algorithm will now be explained in a stepwise order. Readers are advised to take a look at Figure 2.2, where each step is illustrated graphically.

1. **Pick an arbitrary probe from the array of all probes of the same size.**
   This is the initial step of the algorithm. The selection of an arbitrary hexamer from the collection of all possible hexamers ($4^6 = 4096$ in total) could be used as an example. The size of the probe, i.e. the number of nucleotides it consists of, used in the search for possible megasatellites in Megasatfinder, is one of the input parameters to the algorithm. The selected size of the probe should be chosen carefully, since it determines which sizes of tandem repeats will be discovered, i.e. for which size of tandem repeats that the algorithm will perform optimally. One could e.g. also apply pentamers, heptamers, octamers, etc. By using other probe sizes, or even a mixture of sizes, the number of hits (hit count) in the sequence would maximise at other repeat-unit sizes. The algorithm generates all possible probes of the required size, and in this first step only a single one is used during the search.

2. **Locate all matches for the probe in the DNA sequence.**
   The second step in the algorithm is to perform a complete sequence scan in order to find all matches for the probe. This is achieved by picking up all sequence segments that match the probe.

3. **Calculate the sizes of segments between hits.**
   The third step in the algorithm is to calculate the sizes of all segments that lie in between the hits, which were located in step 2. This calculation is essential in the process of finding possible megasatellites, since the size of these segments in relation to each other indicate if there is a tandem repeat present or not.

4. **Mark site if a number of consecutive segments are of similar size.**
   This fourth step is visualised in part 4 of Figure 2.2. As illustrated in the figure, a number of hits have been made along the DNA sequence, using the hexamer probe AGCTGA (indicated by the vertical lines). The segments between the hits are of variable length. Three consecutive segments in the DNA sequence shown are of similar size (marked by three consecutive arrows below the sequence).

5. Record boundaries as a possible megasatellite.
The boundaries indicated by the position of the probe hits are now recorded as a possible megasatellite. In this step, one possible megasatellite has been located, but more hits using other probes are required to increase the likelihood that the discovered sequence segment could be a megasatellite, i.e. a hit using a single probe instance is not sufficient.

6. Repeat the process, using other probes among the ones generated in step 1.
In step 1, all possible probes of a specified size were generated. In this step, the process in steps 2-5 is repeated, using another probe among the ones generated in step 1. This generates more megasatellite predictions in the DNA sequence and hopefully a number of the probes will predict a megasatellite located in the same region of the sequence, thus increasing the quality of the prediction. In Figure 2.2, the hexamer probe AAGCTA has made a number of hits (indicated by the shorter vertical lines). The same region is indicated as a possible megasatellite using the latter probe, which overlaps with the previous megasatellite prediction.

7. Mark regions in the sequence where clusters are being formed.
The final step in the algorithm is to check whether many probes are detecting possible megasatellites in the same or similar positions in the DNA sequence. If a number of probes generate similar predictions, i.e. predictions which overlap and are of similar length, these regions in the sequence are marked. The required cluster size is set by the user as an input parameter. The more probes that are in agreement, the more likely it is that a true megasatellite has been found.

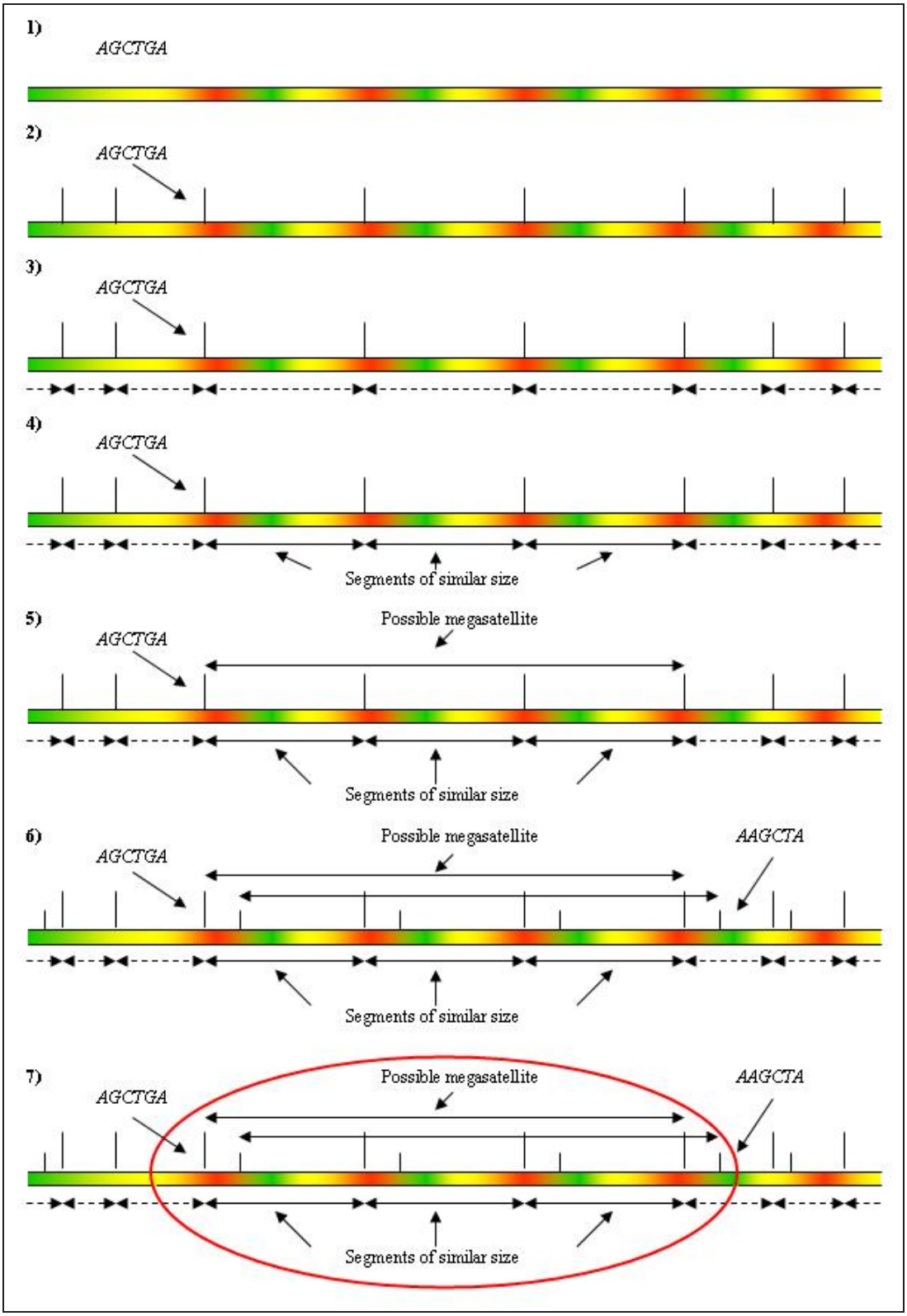


Figure 2.2 A graphical illustration of each step in Megasatfinder.

2.5.2 The parameters in the algorithm

Now that the steps of the algorithm have been discussed, it is appropriate to continue with a description of the parameters used to configure Megasatfinder. Following is a list of the parameters and their default settings, which can be adjusted in order to affect the sensitivity and specificity of the algorithm.

a) The size of the probe.
As previously discussed, the choice of probe size is central regarding the target size of the megasatellites being searched for. The default value was set to 6, i.e. a hexamer, but this value can be decided by the user. However, one must keep in mind that selecting a probe that is too short (or too long) does not return meaningful results. The algorithm is targeted at finding possible megasatellites (repeat-unit size larger than or equal to 1000 bp), and thus some clues regarding appropriate probe size can be deduced. There is no point in selecting probes smaller than pentamers, since pentamers should be appropriate for finding possible megasatellites with repeat-unit size of around 1000 bp, i.e. at the boundaries between macro- and megasatellites. The reason for this is that the distance between two consecutive hits in a totally random nucleotide sequence, using an arbitrary pentamer, should be $4^5 = 1024$ nucleotides. This is also the total number of possible pentamers, since an arbitrary pentamer should in theory occur once in a random sequence fragment of 1024 nucleotides.

This does however not imply a high probability of a by-chance megasatellite hit, i.e. a false positive megasatellite, since a single match in a nucleotide sequence is not sufficient to be considered a hit – a hit requires tandem repeats of the same repeat-unit. Therefore, a pentamer is assumed to be optimal for finding megasatellites with repeat-unit size of *around* 1000 bp. Following the same method of deduction, hexamers should be appropriate for repeat-unit size of around 4000 bp ($4^6 = 4096$), heptamers for repeat-unit size of around 16,000 bp ($4^7 = 16{,}384$), and finally octamers for repeat-unit size of around 65,000 bp ($4^8 = 65{,}536$). Going higher, e.g. nonamers or decamers, provides probes which locate satellites with enormous repeat-units that could be detected by alternative means, such as FISH, or by examination in a microscope. Tandem repeats of such dimensions would probably not be classified as megasatellites, but rather as chromosomal anomalies or genetic defects. Therefore, the probe size interval of 5-8 nucleotides is considered appropriate.

b) Number of consecutive segments of similar size.
One segment does not count as a possible megasatellite. Although the definition states that a megasatellite consists of two or more consecutive segments, the probability of a true positive megasatellite increases the more consecutive segments there are. Therefore the default value was set to 3, i.e. three or more consecutive segments are considered to be worth exploring further and any possible noise in the results is furthermore diminished by this strengthening of requirements.

c) Variation in the length of segments between hits.
Highly similar (or identical) length of the segments between the hits indicates that the tandem repetitive sequence is a possible megasatellite. The default allowed variation in the length of the segments between hits is 1%, since one cannot expect identical lengths. This default percentage was arbitrarily set by the author of the algorithm. Requiring identical length of the segments is too stringent due to the commonality of

indels in the genomic sequences. This is also in direct correlation with the definition of tandem repeats. Charlesworth et al. (1994) defined them as consecutive perfect or *slightly imperfect* copies of DNA motifs of variable lengths and Benson (1999) defined them as two or more contiguous and *approximate* copies of a pattern of nucleotides.

d) Minimum size of segments between hits.
In order to increase the likelihood of a megasatellite and remove the noise generated by short poly-fragment sequences, the default minimum size of segments between the hits is 200 base pairs. This minimum could as well have been set to 1000, since there is no interest in other satellites than megasatellites. However, the author of Megasatfinder decided to set the minimum at 200 base pairs, to be able to locate some satellites below the 1000 bp boundary and eliminate the really short satellites from appearing in the results.

e) Number of input probes.
The number of probes used during a run of the algorithm is important with regard to the number of probes that recognise the same possible megasatellite, i.e. cluster size. The default setting is 150, all randomly selected from the array of all possible probes of a certain size. However, all test scenarios use 300 patterns, since this increases the accuracy of the results (to be discussed in chapter 7). This provides an opportunity to calculate the ratio between the total number of probes used versus the number of probes that detected a possible megasatellite site in the DNA sequence. Using a single probe will result in several false positive and false negative matches. One would of course prefer to use a larger number of probes, but this increases the algorithm's run-time substantially.

f) Minimum number of probes in a cluster (cluster size).
The default minimum cluster size is 2. The same rule applies for cluster size as for many of the other parameters, i.e. that the more probes that pick up a signal at the same location in the DNA sequence, the more likely it is that the algorithm has detected a true positive megasatellite.

g) Genomic and input file scans.
The algorithm can conduct the detection process on different types and sizes of sequences. First of all, the user has the possibility to use a configuration file to control the run. When it comes to selecting a sequence, the user can choose between genomic scans or smaller scans using input sequence files. For the genomic scanning, the user can either scan the entire genomic build or a specified genomic range within the genome. Two types of input files are allowed, fasta files and sequence (binary) files. The main difference between genomic scans and scans performed on smaller input sequences is time complexity, since an algorithm run requires more run-time as the nucleotide sequence elongates.

2.6 How are the presumptive megasatellites confirmed?

As previously declared, the megasatellite detection algorithm reports *possible* megasatellites in a nucleotide sequence. The following step in the overall process is to either confirm or reject these predicted megasatellite sites. The method primarily used for this purpose at deCODE Genetics Inc. is *Southern blotting* (see Figure 2.3).

The process initiates with the digestion of human genomic DNA, using a restriction enzyme. The restriction enzyme cuts the DNA sequence into shorter fragments, but should not cut the suggested megasatellites into pieces. These fragments are then separated by applying gel electrophoresis, using an agarose gel. Due to the large number of various restriction fragments positioned on the agarose gel, it generally appears in the form of smear and not discrete bands. The next phase is the denaturation of the DNA, i.e. the double-stranded DNA is separated into single-stranded DNA by incubating it with sodium hydroxide (NaOH). The denaturised DNA is then transferred to a membrane (a piece of blotting paper). The DNA fragments, which were placed on the agarose gel, preserve the original pattern of separation through the transfer onto the membrane (Berg et al., 2002).

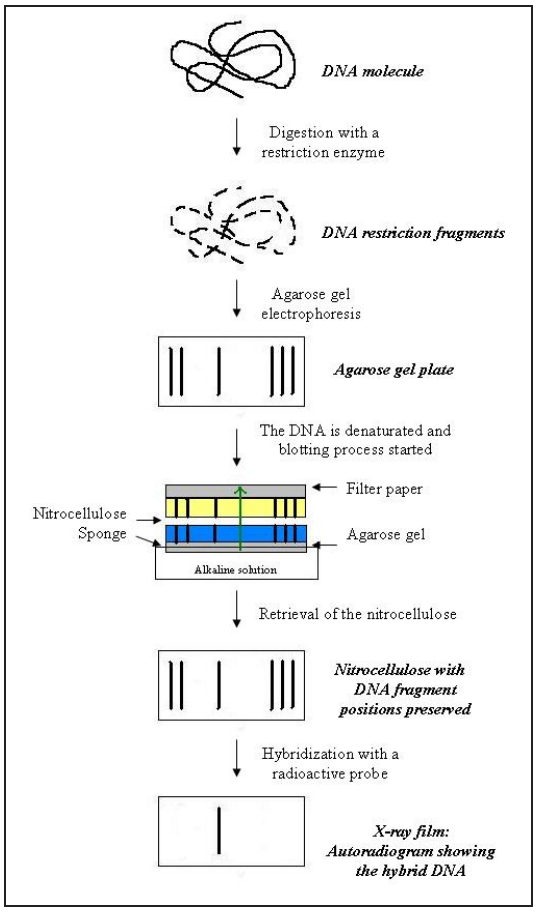


Figure 2.3 Southern blotting confirms or rejects predicted megasatellites.

The subsequent phase is to incubate the blot with numerous copies of a single radioactive probe (a short fragment of single-stranded DNA). The probe should match the fragment which contains the megasatellite. The probe binds with its complementary DNA sequence, forming a double-stranded DNA molecule. The position of the probe is finally revealed on an X-ray film, since the probe was radioactively labelled (Berg et al., 2002).

If the identified megasatellite is polymorphic, one expects to see two bands on the X-ray film, else only one. In order to prevent possible problems resulting in biased results, e.g. regarding homozygotes, the process is repeated for several individuals.

3 Problem statement

3.1 Problem description

In the biotech society there is a growing need for algorithms applicable to the detection of possible megasatellites in genomic sequences, since the importance of these sequences is becoming increasingly apparent to researchers. A suite of algorithms have already been developed for the detection of shorter satellite DNA sequences. According to the author's best knowledge, no algorithms solely intended for the detection of potential megasatellites on a genome-wide scale, have previously been developed. Additional reasons for the importance of this automated method of detection have already been stated in Chapter 1.

The problem is that the overall applicability of this novel algorithm is unknown, since no analysis has yet been conducted. It is therefore imperative to develop and perform a structured method for the analysis of Megasatfinder in its present state, hopefully leading to guidelines of usage that increase the reliability of its results. This work requires the systematic extraction and creation of both real and simulated nucleotide sequence data, structured means of setting up meaningful test scenarios, design of graphical visualisation and appropriate means of analysing the acquired results. The driving force behind this work is to investigate the usability and accuracy of the algorithm, thus providing users with the required knowledge of algorithm applicability. This implies the creation of guidelines for the use of the algorithm regarding appropriate parameter settings. The work might finally involve some improvements in the implementation of the algorithm, depending on the outcome of the analysis.

3.2 Project aims and objectives

The overall aim of the project is firstly to define a structured method of analysing the algorithm and secondly to improve the algorithm with regard to usability and accuracy. This aim is subdivided into two main aims. To achieve each aim, a number of objectives must be attained. These aims and objectives are listed below.

1. Define a structured method of analysing the megasatellite detection algorithm Megasatfinder.
    1.1. Inspect the methods for algorithm analysis developed by others and check whether any parts of these methods are applicable to the new method to be defined for Megasatfinder.
    1.2. Conduct a systematic extraction and creation of both real and artificial nucleotide sequences to be used in the analysis process.
    1.3. Define structured and meaningful test scenarios constituting the new analysis method.
    1.4. Create scripts that can run the algorithm using specific parameter settings. The scripts should be designed so that they can be easily altered for the various runs to be performed.

2. Improve the algorithm with regard to usability and accuracy, and evaluate the results of the suggested improvements.
   2.1. Apply the newly defined analysis method on the algorithm.
   2.2. Perform possible alterations in the implementation of the algorithm, depending on the result of the complete analysis. These alterations could e.g. include various changes and/or extensions to the algorithm or changes regarding the default parameter settings.
   2.3. Develop guidelines for the use of the algorithm with respect to parameter settings. These parameter settings should maximise the two concepts usability and accuracy on a case to case basis.
   2.4. An important part of the work is also to better understand the output of the algorithm in general.

3.3 Project hypothesis

As previously stated, the results of the complete analysis will have to lead the way regarding any improvements of the parameter settings or the implementation of the algorithm in general. The task is to develop and conduct an effective and organised method of analysis for the megasatellite detection algorithm, hopefully leading to its optimisation. Structuring the analysis process would imply reduced analysis time, generating a controlled collection of data for further evaluation. The hypothesis put forward is thus the following:

> *By creating a structured method of analysis and performing possible improvements following as a result, the megasatellite prediction algorithm will be enhanced regarding*
> *i) the user's understanding and*
> *ii) general performance, such as usability and accuracy,*
> *thus increasing the algorithm´s applicability in the search for possible megasatellites in the human genome.*

4 Related work

It seems like the general consensus regarding the detection of satellite DNA has been that scientists first come across abnormalities in a DNA sequence, i.e. some kind of repetitive nucleotide sequence, during laboratory studies, which are then followed by further research performed *in silico*. The idea behind Megasatfinder was however to go the other way around, i.e. to search through the whole human genome for possible megasatellites and then confirm these in the laboratory. Objective 1.1 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 1. The objective states the following:

> *Inspect the methods for algorithm analysis developed by others and check whether any parts of these methods are applicable to the new method to be defined for Megasatfinder.*

There are a number of algorithms that detect tandem repeats, either in a direct or indirect manner. As with other algorithms and computer programs in general, each is implemented for a specific purpose using specific methods, implying that each has some limitations. Benson (1999) divided some of these algorithms into three main categories. The first category is based on computing alignment matrices. Examples of these are the algorithms written by Benson (1995), Kannan and Myers (1996), and Schmidt (1998). The main drawback of these algorithms is their excessive run-time. According to Benson (1999), the best algorithm among the three is the one written by Schmidt (1998). However, this algorithm has time complexity $O[n^2 \text{ polylog}(n)]$ for sequence length n, which means that this algorithm is not usable for sequences that exceed several thousand bases (Benson, 1999; Schmidt, 1998). Taking a closer look at their methods of analysing the algorithms shows that Benson (1995) does neither perform any kind of analysis on the algorithm, nor does he compare it to other existing algorithms at that time. Kannan and Myers (1996) focused on complexity analysis, since their work improved the previously best-known bound for the worst-case complexity for the problem. Finally, Schmidt (1998) analyses the algorithm with regard to the use of weighted paths in directed grid graphs. None of the above, except for the general idea of including complexity tests in the analysis provided by Kannan and Myers (1996), is therefore applicable to the new analysis method to be developed for Megasatfinder.

The second category of algorithms detects tandem repeats by applying indirect methods derived from the area of data compression. Examples of these are the algorithms written by Milosavljevic and Jurka (1993), and Rivals et al. (1997). The former detects the so called “simple sequences”, meaning a combination of fragments that occur elsewhere according to Benson (1999). The existence of tandem repeats within these simple sequences is not guaranteed and the weakness of the algorithm is that it does not deduce a repeated pattern. Milosavljevic and Jurka (1993) do not perform an explicit algorithm analysis or any comparison to other similar algorithms. The latter one, by Rivals et al. (1997), founds the data compression on the occurrence of tiny patterns (1-3 bp in size), which are selected beforehand as input by the user. The algorithm is thus not readily generalised for longer patterns (such as megasatellites). Rivals et al. (1997) analysed the algorithm by testing it on four yeast chromosomes, thus providing the idea that testing Megasatfinder on real nucleotide sequences should be included in the new analysis method to be developed. A positive aspect of both algorithms is, according to Benson

(1999), that they provide a measure of statistical significance based on the amount of data compression.

The third and final category consists of algorithms that are more direct in their search for tandem repeats. Examples from this group are two exact algorithms written by Landau and Schmidt (1993), and Sagot and Myers (1998) and two heuristic algorithms written by Benson and Waterman (1994), and Karlin et al. (1988). The first one of these (Landau & Schmidt, 1993) is constrained by the definition of approximate patterns. By this definition, the algorithm requires that two pattern copies vary either by *k* or less substitutions, or by *k* or less substitutions and indels, thereby considering substitutions and indels equally (Benson, 1999). Landau and Schmidt (1993) focus on time complexity in their analysis of the algorithm, but also perform some experimental runs, providing indications of the performance and sensitivity of the algorithm. The second exact algorithm by Sagot and Myers (1998) has the limited pattern size of 40 bases and requires the specification of estimated pattern size and range of copy number beforehand (Benson, 1999). Sagot and Myers (1998) use experimental runs to derive the time complexity of the algorithm. The heuristic algorithm written by Benson and Waterman (1994) requires a predefined pattern size, provided as input on behalf of the user. The only analysis provided by Benson and Waterman (1994) consists of a few examples concerning run-time, supposed to derive some kind of empirical time complexity analysis. The last algorithm, written by Karlin et al. (1988), is mainly weakened by the use of corresponding blocks separated by error blocks of rigid size (Benson, 1999). Karlin et al. (1988) do not provide any kind of analysis or comparison to other similar algorithms. Summarising, it can be stated that the analysis of these four algorithms is mainly focused on complexity analysis and the conductance of test-runs, designed to provide indications of algorithm performance and sensitivity. These aspects will be included in the new analysis method to be developed for Megasatfinder.

The final example of other algorithms for the detection of tandem repeats is Tandem repeats finder (TRF) (Benson, 1998; Benson, 1999). According to TRF's author, the algorithm is comparatively automated, since it operates without needing any information regarding a specific pattern or pattern size. The tandem repeats are modelled using percent identity and the occurrence of indels between contiguous copies of the pattern (Benson, 1999). However, TRF reports only satellite sequences that have a repeat-unit smaller than 500 bp, which is by definition below the minimum limit for megasatellite sequences. Benson (1999) performs the analysis by showing results from test-runs using known genes and yeast chromosomes, and by providing examples of algorithm run-time. Again, the same types of analysis strategies are encountered, as for some of the other algorithms previously discussed.

There are of course similarities as well as differences between these algorithms and Megasatfinder, but it is, to the author's best knowledge, the first algorithm solely intended for the detection of potential megasatellites on a genome-wide scale. Additional features in relation to the three categories discussed earlier are that the algorithm accepts all sequences without any limits in length, and that it allows unlimited probe size. Multiple algorithms have on the other hand been created for the detection of the shorter satellite DNA sequences, which is most likely due to the fact that these sequences were discovered previous to the megasatellites.

There are many aspects to consider when performing analysis of an algorithm. The methods chosen depend mainly on two things; the algorithm and the analyser. The authors of most algorithms perform their own evaluation of the algorithm but this had, as previously mentioned, not been performed for Megasatfinder. Each author has often developed a personal way of performing the analysis based on the algorithm, but there are some measurements that are more popular than others, as proven by the choice of analysis strategies for the algorithms discussed in this chapter. Empirical complexity analysis is one example (this can also be performed theoretically); to run the algorithm using specific and often well-known sequences is another. Other parts of the analysis process which might be performed are algorithm specific, but since most of the analyses performed on the algorithms were of a relatively simple nature, any algorithm specific analysis strategies were not encountered. As can be concluded from Chapter 3, the idea is to develop a much more extensive analysis method for Megasatfinder, and therefore this new method will have to be developed from scratch, adding the ideas for complexity analysis and experimental test-runs.

5 Experimental approach

5.1 The extraction and creation of the nucleotide sequences

Objective 1.2 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 1. The objective states the following:

> *Conduct a systematic extraction and creation of both real and artificial nucleotide sequences to be used in the analysis process.*

The various test scenarios (see subchapter 5.2) use a set of three nucleotide sequence files. These contain a 10 Mb (megabase) nucleotide sequence extracted from the human genome, a 10 Mb nucleotide sequence created randomly by applying the 40/60 base rule, and finally a 1 Gb (gigabase) nucleotide sequence created randomly by applying an equal probability of 25% for each of the four bases (A, C, G, T). The *40/60 rule* is a term put forward by the author to describe the general relationship between the concentrations of the four bases in DNA. Cytosine (C) and guanine (G) make up approximately 40% of DNA (20% each), while adenine (A) and thymine (T) make up approximately 60% (30% each), hence the 40/60 rule. Originally, the idea was to find a tool which artificially generated "DNA sequences", but since the search failed it was decided to use this strategy instead. These files have no special start or end symbols in addition to the nucleotide sequence, such as for the fasta or pir formats. This characteristic makes these files very easy to create and handle. The extraction and creation of these sequences is described further in the following two subchapters (5.1.1-5.1.2).

5.1.1 The extraction of real nucleotide sequences from the human genome

The real nucleotide sequence was extracted from positions 150,000,000-160,000,000 on chromosome 1. The sequence data was taken from the NCBI Build 34 version of the genomic assembly, which can be considered as a consensus human genome. The choice of sequence from the genomic assembly was not random. The aim was to find a typical sequence without apparent unusual characteristics, which could affect the generality of the test results. Chromosome 1 was inspected using the UCSC genome browser (http://genome.ucsc.edu/; Kent et al., 2002) and this inspection resulted in the above positions. After inspection, the sequence was extracted from the UCSC genome browser database (Karolchik et al., 2003).

5.1.2 The creation of random nucleotide sequences

The random nucleotide sequences used in the test scenarios were created artificially by applying the Python script megasat_random_generator.py (see subchapter 5.3.5). By using the script, the user can either create a sequence applying the 40/60 rule or a sequence applying a probability of 25% per base. The purpose of performing the tests using real DNA sequence data, random sequence data applying the 40/60 rule, and random sequence data applying a 25% probability per base is comparative analysis. By comparing the results, one can not only deduce the random- or non-randomness of the human genome, but also detect whether it is sufficient to use a totally random or a 40/60 rule generated sequence as a model for real DNA in computational simulations.

5.2 The test scenarios constituting the analysis method

Objective 1.3 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 1. The objective states the following:

> *Define structured and meaningful test scenarios constituting the new analysis method.*

As a result of an inspection made on the algorithm, a set of six test scenarios were developed. These are supposed to reveal the overall behaviour of the algorithm and provide enough information to deduce indications of general performance. Following are the developed test scenarios. The parameters that are altered in each scenario are mentioned explicitly, the rest of the parameters for each scenario are kept at default settings. A detailed description of each scenario, along with parameter settings and general conductance, can be found in Appendix A.

1. Run the algorithm on a randomly generated sequence, applying a 25% probability per base, containing no LSTRs, i.e. no possible megasatellite sites. This test involves the use of fixed probes, successive alterations in the length variation tolerance (l.v.t.) of segments between hits, and various sizes of probe sets.
    - *Objective*: Find the "baseline", i.e. how frequently the algorithm indicates a signal in a sequence where no signal should occur. This provides hints towards the algorithm's accuracy.
    - *Motivation*: Knowing the result provides the user with a "baseline" cluster size, i.e. the maximum size of clusters that can be ignored. Example: Running with 300 hexamers over the entire human genome, do clusters of size 3 with repeat-unit size 5000 indicate a real signal?
2. Same as in test scenario 1, this time on a real DNA sequence extracted from the human genome, believed to contain no LSTRs. The retrieved results from test scenarios 1 and 2 are compared. This test involves the use of fixed probes and length variation tolerance (l.v.t.).
    - *Objective*: Investigate whether a randomly generated sequence is sufficient as a model for the simulation in test scenario 1, i.e. whether random sequences resemble DNA sequences enough to be used for *in silico* research purposes.
    - *Motivation*: Validates the use of an *in silico* generated sequence when doing this kind of analysis. A positive outcome would be preferable, since *in silico* generated sequences are easier to obtain and control.
3. Run the algorithm on randomly generated tandem repeats of fixed repeat-unit size and repeat count, using both a real and a random nucleotide sequence. This test involves the use of all four probe sizes (pentamers, hexamers, heptamers, and octamers) and varying repeat-unit sizes.
    - *Objective*: Find the "curve" for each probe size, i.e. find the repeat-unit size for each probe size at which the algorithm expresses maximum sensitivity. In addition, find when the sensitivity reaches a baseline, or alternatively when the signal fades out. This test scenario provides guidelines regarding the appropriate probe sizes to use when searching for megasatellites with a certain repeat-unit size.

- *Motivation*: Helps the user to interpret the signal: "Running with 300 hexamers, found a cluster of size 20 with repeat-unit size 15,000".

4. Same as in test scenario 3, after mutating the sequence. This test involves a variation in sequence similarity, fixed probes, and a range of repeat-unit size similarities (i.e. variation in length variation tolerance (l.v.t.)).
    - *Objective*: Reveal in what way the "curves" from previous simulations change with "evolution time", i.e. with the mutation level. In addition, using a fixed mutation level, find how the length variation tolerance (l.v.t.) affects the curves. This provides a measure of how sensitive the algorithm is with respect to changes in the sequences.
    - *Motivation*: Important to realise how base pair dissimilarity between repeat-units affects the performance of the algorithm.
5. Run the algorithm on the entire human genome, containing confirmed megasatellites. This test involves the use of all four probe sizes.
    - *Objective*: Explore in a real-life situation which parameter settings maximise the performance. Compare with previous simulations. This gives an example of usage on real data and which results are retrieved as an output from the algorithm. It is, in addition, also important to verify whether the algorithm predicts megasatellites which are already known and confirmed.
    - *Motivation*: Knowing how the algorithm works on a real DNA sequence containing megasatellites increases knowledge of its function and behaviour.
6. Run the algorithm on the 10 Mb real DNA sequence and the entire human genome, with the derivation of complexity analysis in mind. This test involves the use of all four probe sizes, a varying number of iterations for each repeat-unit size, and a varying size of the probe sets.
    - *Objective*: Derive time complexity, by measuring run-time. This knowledge is essential with regard to both usability and applicability.
    - *Motivation*: It is important to know the computational requirements, i.e. the amount of time it takes to perform a run.

5.3 The scripts used for running the algorithm

Objective 1.4 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 1. The objective states the following:

> *Create scripts that can run the algorithm using specific parameter settings. The scripts should be designed so that they can be easily altered for the various runs to be performed.*

A set of thirteen different Python scripts were used during the analysis process. The purpose of each of these scripts is described in the following subchapters (5.3.1-5.3.8).

5.3.1 megasat_creator_finder.py and megasat_creator_finder_mutation.py

The first script creates all possible probes (patterns) of a specified size. One of the script's functions takes care of performing all algorithm runs. For each run, a new artificial megasatellite is created. In addition, the predefined number of probes to use is

randomly selected from the set of all possible probes. Each probe is then placed as input into the algorithm, until all probes have been tried. The number of successful probes, i.e. which result in a hit in the sequence, for each repeat-unit size are then counted. The results are placed in the output file in two columns; the first one contains the repeat-unit size and the second one contains the number of successful probe hits. The runs are performed iteratively for each repeat-unit size, where the number of iterations is specified by the user. The number of iterations is very important, since a higher number of runs increases the accuracy of the generated results. All parameters used by the algorithm can be set by the user when calling the script. The script can be found in Appendix B.

The megasat_creator_finder_mutation.py script is the mutation variant of the previous script. In this script the artificial megasatellites created for each run are mutated, or evolved, by *X* percent, where the mutation percentage is selected by the user as an input parameter. The output for each repeat-unit size is the cluster sizes found by the algorithm along with its minimum and maximum segment size. The script can be found in Appendix C, where a 1% mutation is provided as an example.

5.3.2 megasat_finder_nomegs_random.py and megasat_finder_nomegs_DNA.py

The first script's purpose is to see if the algorithm reports possible megasatellites in a sequence where there should be no signals present. The 1 Gb random sequence is used by the script, which divides it into ten 100 Mb sequences for time and space complexity reasons. As with the two scripts discussed above (5.3.1), this script creates all possible probes of a given size and randomly selects a subset. In conclusion, it has a function which runs both the detection function of the algorithm and the clustering function. The clustering function groups the hits and provides information about clusters with a cluster size of two or more (default). The script can be found in Appendix D.

The megasat_finder_nomegs_DNA.py script functions in the same way as the previous one, except that it uses the 10 Mb real DNA sequence, which is thus not divided into any subsequences. The script can be found in Appendix E.

5.3.3 megaresults_meanfinder.py and megaresults_mut_meanfinder.py

The megaresults_meanfinder.py script calculates the mean and standard deviation of result values, i.e. the number of successful probes, for runs performed for clusters of iterations of a given number. The script can also be altered so that the median is calculated instead of the mean. The generated results, i.e. the mean/median and the standard deviation, are saved in two separate files, whose names are indicated by the user. The script can be found in Appendix F.

The second script functions in the exact same way as the previous one, except that it calculates the mean of cluster sizes for each repeat-unit size using mutated megasatellites, instead of the mean of successful probes. The script can be found in Appendix G.

5.3.4 megaresults_clusterfinder.py

The megaresults_clusterfinder.py script creates cluster size categories and calculates the total number of clusters belonging to each category. The generated results are saved in a file, which name is indicated by the user. The script can be found in Appendix H.

5.3.5 megasat_random_generator.py

This script was designed for the purpose of creating random nucleotide sequences for use in the test scenarios. The sequences can either be created by applying the 40/60 rule or in a completely random fashion, depending on the user's requirements. The size of the sequence can easily be set by the user and when the sequence is completed it is saved in a file, named by the user. The script can be found in Appendix I.

5.3.6 wrapper.py

The script wrapper.py is used to run megasat_creator_finder.py with various parameter settings in order to organise the runs and make them easier to perform. The algorithm runs must be performed on a specific cluster at deCODE Genetics Inc. (called lpbs), and therefore a security valve was integrated that checks whether the user is logged onto that server or not. Given that the user is logged onto the server, the set of repeat-unit sizes to be used is generated. The reason why this is done here is that the ranges of repeat-unit sizes differ based on the probe size. Each set begins with a repeat-unit size of 100 nucleotides, instead of the megasatellite minimum size of 1000 nucleotides, in order to retrieve a graph which provides a view of the whole curve. The subsequent repeat-unit sizes are then gradually increased by 5% to obtain a proportional augmentation, since denser steps are preferred in the beginning to retrieve a clearer curve in the graph and fewer steps in the end to reduce time complexity. A call is finally made on the megasat_creator_finder.py script (5.3.1), specifying each of the input parameters. This process is repeated for all four probe sizes, using both real nucleotide sequences and nucleotide sequences generated by applying the 40/60 rule. The script can be found in Appendix J.

5.3.7 wrapper_6_mutations.py and wrapper_6_mutations_lvt.py

The first wrapper is used in a similar way as wrapper.py in subchapter 5.3.6, except that it runs the megasat_creator_finder_mutation.py script (5.3.1). This process uses hexamer probes on both real nucleotide sequences and nucleotide sequences generated by applying the 40/60 rule, for each of the five mutation levels. The script can be found in Appendix K.

The second wrapper also runs the megasat_creator_finder_mutation.py script. The difference is that various length variation tolerance (l.v.t.) percentages are applied in addition to the mutational effects, using two out of the five mutation levels. The script can be found in Appendix L.

5.3.8 wrapper_nomegs_random.py and wrapper_nomegs_DNA.py

The first wrapper runs the script megasat_finder_nomegs_random.py (5.3.2). Runs are performed for various probe set sizes. For each of these sets, a run is performed using a set of different length variation tolerance percentages. The script can be found in Appendix M. The script uses the 1 Gb random sequence.

The second wrapper runs the script megasat_finder_nomegs_DNA.py (5.3.2). Runs are performed similarly to the runs in the previous wrapper. The script can be found in Appendix N. The script uses the 10 Mb DNA sequence, but can, as the previous wrapper, easily be changed to accept another sequence.

5.4 The various tools used during the analysis process

5.4.1 Gnuplot

The Windows version of the plotting tool Gnuplot (Version 4) was used to generate all graphs representing numerical results in the test scenarios. The software tool is described by its authors as a command-line driven interactive data and function plotting utility. The Gnuplot software is copyrighted, but freely distributed through its homepage (http://www.gnuplot.info).

5.4.2 Python

Python has for the past few years become an increasingly popular programming language within diverse natural science disciplines. Megasatfinder was programmed in Python (Version 2.3.3) and therefore it seemed appropriate to write all other scripts connected to the algorithm and its analysis in that language.

According to Python's homepage (http://www.python.org), Python is an interpreted, interactive, and object-oriented programming language. Python can in many aspects be compared to e.g. Perl and Java. It has classes, modules, dynamic data types, dynamic typing, and exceptions. Additionally, Python has interfaces to many system calls and various libraries. It is easy to write new built-in modules, using C or C++. The Python implementation is portable and runs on UNIX, Windows, Macintosh, and many other platforms. Python is copyrighted, but freely usable and distributable.

5.4.3 Msbar

The Msbar software tool is one among the many applications available in the EMBOSS suite (http://emboss.sourceforge.net; Rice et al., 2000). EMBOSS stands for the *E*uropean *M*olecular *B*iology *O*pen *S*oftware *S*uite and is a free open source software analysis package, specially developed for the needs of the molecular biology user community. The application Msbar was written in 1999 by Gary Williams at the MRC Rosalind Franklin Centre for Genomics Research in Cambridge, England.

First of all, Msbar is an acronym for "Mutate sequence beyond all recognition". This name does however not provide a comprehensive description of Msbar's function, since it can induce minor or major changes to a sequence. According to its author, the application attempts to emulate various forms of mutation, not selection. The number and types of mutations can be set by the user. The application offers three categories of mutations; point, codon, and block mutations. Seven mutation options are offered within each of these categories; none, insertions, deletions, substitutions, duplications, moves, and finally any of the above.

The user decides the number of times to perform the selected type of mutation and provides a file name, which the resulting mutated sequence will be placed in. In the analysis of this megasatellite algorithm it was decided to use any type belonging to the point mutation category. Firstly, the codon mutation category is not an option since the sequences consist of nucleotides, not amino acids. Secondly, the effects of block mutations are far too extensive for altering an artificially generated tandem repetitive sequence, since mutations of this scale could destroy existing tandem repeats completely and are therefore not suited for the test scenarios.

6 Experimental results

This chapter contains the results obtained by conducting each of the test scenarios in subchapter 5.2. Objective 2.1 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 2. The objective states the following:

Apply the newly defined analysis method on the algorithm.

The new analysis method was carried out and the results are divided into subchapters, based on the predefined test scenarios (see subchapter 5.2 and Appendix A).

6.1 False positive hit statistics for a random sequence

Algorithm runs were performed using the 1 Gb random sequence. The sequence was created with a 25% probability of each of the four bases to guarantee that the sequence would not contain any long-segment tandem repeats. The objective was to investigate how frequently the algorithm indicated a signal in a sequence, where no signal should be present. The results were then clustered and a list of the number of clusters belonging to a specific cluster size, using a certain probe set size and length variation tolerance percentage, is given in Table 6.1.

Table 6.1 The number of clusters belonging to a specific cluster size, found for various length variation tolerance thresholds (the five percentage columns), using a set of 100, 200, and 300 randomly chosen hexamer probes from the set of all possible hexamer probes (4^6 = 4096). The algorithm was run on the 1 Gb sequence, generated by applying a 25% probability of each base. Due to its size, a full version of the table, Table O.1, was placed in Appendix O.

Probe count	Cluster size	0%	0.5%	1%	2%	5%
100	2	-	101	500	3516	35805
100	3	-	-	3	124	6297
100	4	-	-	-	4	1471
100	5	-	-	-	-	431
100	6	-	-	-	1	160
100	7	-	-	-	-	64
200	2	-	366	2029	13330	44873
200	3	-	1	39	779	11908
200	4	-	-	-	83	4345
200	5	-	-	-	10	1994
200	6	-	-	-	1	1003
200	7	-	-	-	1	545
300	2	-	853	4535	28059	-
300	3	-	7	124	2610	-
300	4	-	-	4	393	-
300	5	-	-	-	63	-
300	6	-	-	1	7	-

Additional runs for each of the three probe sets were conducted using a length variation tolerance percentage of 10%. This resulted in abnormal clustering results. Most of the

predicted clusters were merged into a few massive clusters by the clustering function of the algorithm. The same applies to the runs conducted using a probe set of 300 probes, with a length variation tolerance percentage of 5%. The output does not provide meaningful results and is therefore not included in Table 6.1. Due to the high number of cluster sizes found for the runs using a probe set of 100 and 200 probes respectively, with a length variation tolerance percentage of 5%, the full version of Table 6.1, referred to as Table O.1, was placed in Appendix O. The results clearly indicate that both the number of cluster sizes and the number of clusters belonging to each cluster size increase substantially as the length variation tolerance percentage is increased, in addition to the effect of an increase in probe count.

6.2 False positive hit statistics for a DNA sequence

This test is equivalent to the one in subchapter 6.1, except that the algorithm was run on the 10 Mb real DNA sequence. The test was performed for two of the l.v.t. percentages, 1% and 2%, and proposed for comparison with the corresponding values when using a random sequence (see Table 6.1). The intention was to clarify whether a sequence, generated by applying the 40/60 rule, could be considered sufficient as a model for computer simulations, i.e. whether this random sequence resembles the DNA sequence enough to be used for analysing the algorithm (and possibly other similar algorithms), and also to elucidate the occurrence of random results, which is important regarding accuracy. The results are displayed in Table 6.2.

Table 6.2 The number of clusters belonging to a specific cluster size, found for various length variation tolerance thresholds, using a set of 100, 200, and 300 randomly chosen hexamer probes from the set of all possible hexamer probes ($4^6 = 4096$). The algorithm was run on a 10 Mb (megabase) real DNA sequence extracted from the human genome.

Probe count	Cluster size	1%	2%
100	2	3	26
100	3	1	2
100	29	-	1
100	30	1	-
200	2	23	67
200	3	-	6
200	4	-	1
200	5	-	6
200	49	-	1
200	51	1	-
300	2	32	156
300	3	2	13
300	4	2	2
300	81	-	1
300	86	1	-

The results strongly indicate that the number of clusters found, using a real DNA sequence, is significantly lower than using a random sequence with a 25% probability per base. The large cluster sizes for each probe count, containing a single cluster (see the last two rows for each probe count), are interesting. The ones in Table O.1 for the 5% mutation level (Appendix O) are however most likely generated by merging too many clusters in a single massive one and are not to be considered as reliable results.

6.3 Sensitivity analysis for repeat-unit sizes

Runs were performed applying fixed repeat-unit sizes and repeat counts, using artificially created megasatellites for both real DNA sequences and sequences generated by applying the 40/60 rule. The purpose of these runs was to retrieve basic statistics regarding the sensitivity of the algorithm for various repeat-unit sizes, when applying each of the four probe sizes (pentamers, hexamers, heptamers, and octamers). The main intention was to elucidate the maximum repeat-unit size sensitivity for each probe size as well as to find out at which repeat-unit sizes the signal faded out, i.e. reached zero.

Figure 6.1 visualises the mean sensitivity for each of the four probe sizes, when running the algorithm on the real DNA sequence. Parts a) to d) clearly show the peak of each curve along with the general behaviour regarding probe optimality for each of the four probe sizes. The last part of the curves, i.e. the fading out of the number of hits, is displayed in Figure O.1 in Appendix O. Notice how similar the shape of the curves is for all probe sizes. The results are summarised in Table 6.3.

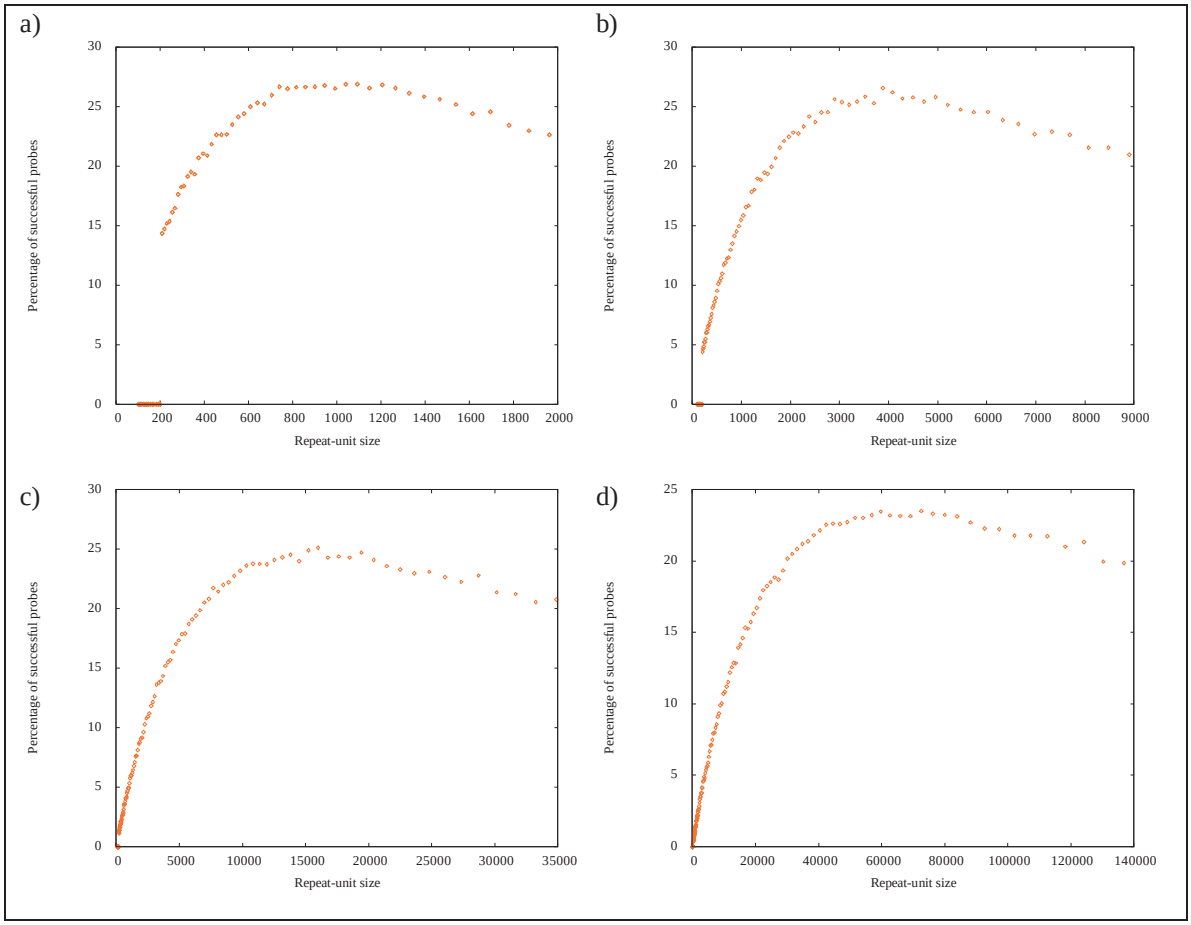


Figure 6.1 A graphical view of the results from running the algorithm with fixed repeat-unit sizes and repeat counts, using artificially created megasatellites. The figure shows the mean sensitivity, when running the algorithm on a real DNA sequence extracted from the human genome. Part a) using pentamers, b) using hexamers, c) using heptamers, and d) using octamers. The figure clearly shows the shape of the curves and their peaks. The repeat-unit size is indicated on the x-axis and the percentage of successful probes is indicated on the y-axis. A successful probe is defined as a probe which results in a hit in the sequence.

Figure 6.2 visualises the mean sensitivity for each of the four probe sizes, when running the algorithm on a sequence generated by applying the 40/60 rule. Parts a) to d) clearly show the peak of each curve along with the general behaviour regarding probe optimality for each of the four probe sizes. The last part of the curves, i.e. the fading out of the number of hits, is displayed in Figure O.2 in Appendix O. The results are summarised in Table 6.3. The last column of Table 6.3 states the approximate repeat-unit size at which the signal faded-out, i.e. when it reached zero. The overall behaviour of the last part of the curve can be visualised in Figure O.2 (Appendix O).

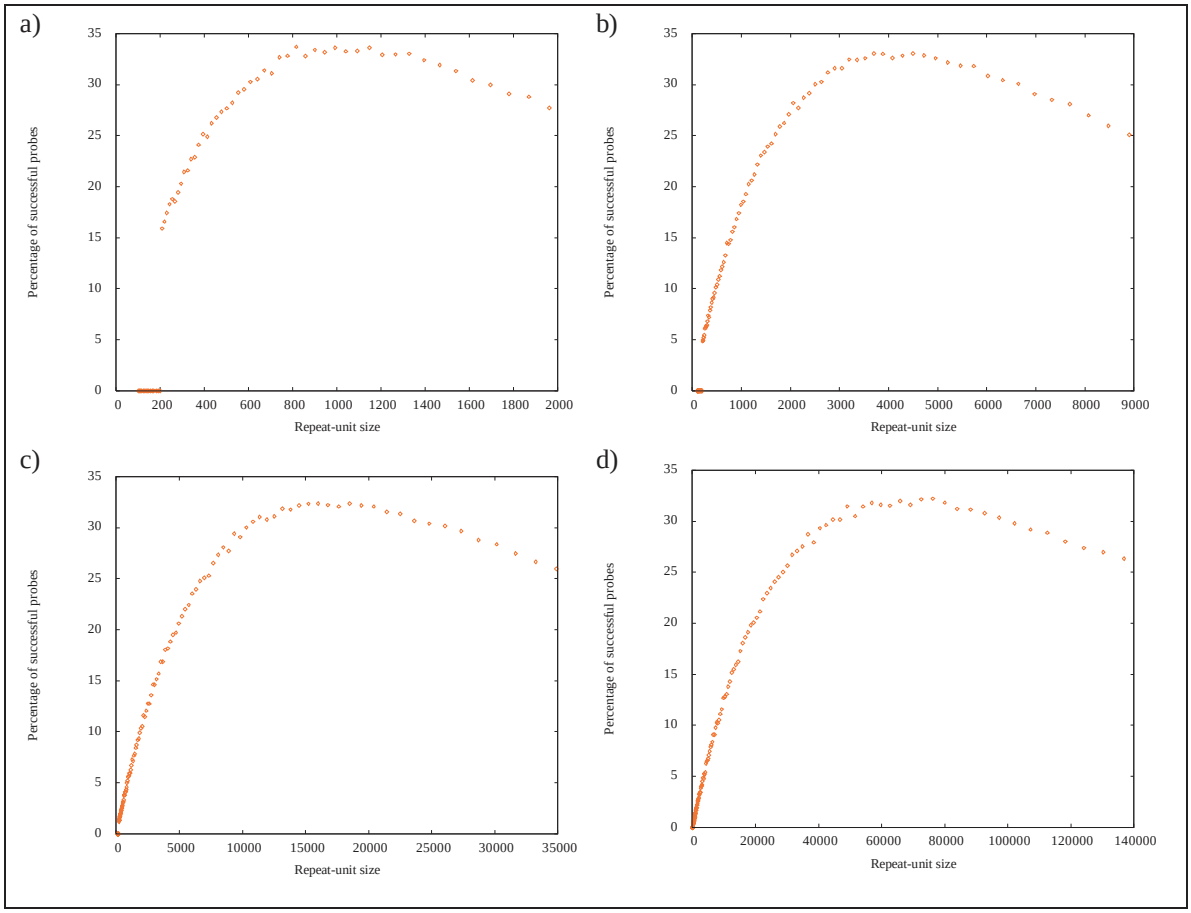


Figure 6.2 These figures show the results generated by performing the same type of tests as performed when generating the figures in Figure 6.2, except this time the algorithm is run on a sequence generated by applying the 40/60 rule. Part a) using pentamers, b) using hexamers, c) using heptamers, and d) using octamers.

Observe that the difference, regarding repeat-unit sizes, between the mean sensitivity peaks using real versus randomly generated nucleotide sequences is trivial (see Table 6.3). However, the slope of the curves for random sequences lowers significantly faster, meaning that the signal fades out at much shorter repeat-unit sizes than for real DNA sequences. This can be easily visualised by comparing Figures O.1 and O.2 in Appendix O.

Table 6.3 The expressed maximum sensitivity for each probe size and the signal fade-out sensitivity. There are two categories of sequences: The real DNA sequence extracted from the human genome and the random sequence generated by applying the 40/60 rule. Both sequences are of length 10 Mb. The sequences have not been exposed to mutations. The length variation tolerance of the segments between hits is kept at 1% (default setting).

Probe size	Sequence category (DNA / Random)	Maximum sensitivity (repeat-unit size)	Signal fades out (repeat-unit size)
Pentamer	DNA	~1000	130,265
Hexamer	DNA	~4000	1,011,063
Heptamer	DNA	~16,000	3,260,776
Octamer	DNA	~65,000	8,239,817
Pentamer	Random	~1000-1100	30,140
Hexamer	Random	~4000-4200	130,265
Heptamer	Random	~16,000	510,655
Octamer	Random	~65,000	1,568,488

Tables 6.4 and 6.5 show the percentage of successful probes for a number of distinct repeat-unit sizes. A successful probe is defined as a probe which results in a hit in the sequence. The first one (Table 6.4) shows the resulting success percentages using the real DNA sequence, while the second one (Table 6.5) shows the success percentages for the randomly generated sequence. The success percentages are calculated by dividing the number of successful probes with the total number of probes used in the test, thereby gaining a general ratio aspect instead of a fixed value, which depends on the number of probes used. The selection of repeat-units was chosen to provide a general view of the variations in hit count, based on repeat-unit size and probe size. Readers interested in comprehensive overviews are referred to Figures 6.1-6.2 and Figures O.1-O.2.

Table 6.4 The success percentages for Megasatfinder, for a selection of repeat-unit sizes, using a real DNA sequence extracted from the human genome. The table shows the success percentages for pentamers, hexamers, heptamers, and octamers using a randomly generated set of 300 probes. The percentage (successful probes / total number of probes) is indicated for a selection of repeat-unit sizes for each of the four probe sizes. The highest percentage for each probe size is indicated in bold.

Repeat-unit size	Pentamer	Hexamer	Heptamer	Octamer
500	22.66	9.53	2.69	0.78
1040	**26.88**	15.84	5.31	1.47
4077	13.53	**26.20**	15.47	5.16
15,984	5.48	13.88	**25.09**	14.59
65,793	0.60	6.23	14.19	**23.14**
102,066	0.20	4.17	11.26	21.78
510,655	-	-	3.83	11.76

Table 6.5 The success percentages for Megasatfinder, for a selection of repeat-unit sizes, using a sequence generated by applying the 40/60 rule. See Table 6.4 for further explanations.

Repeat-unit size	Pentamer	Hexamer	Heptamer	Octamer
500	27.68	10.41	2.93	0.69
1040	**33.27**	18.54	5.93	1.65
4077	12.20	**32.62**	18.16	5.40
15,984	0.16	12.89	**32.35**	18.06
65,793	-	0.25	13.40	**32.0**
102,066	-	0.02	6.03	29.78
510,655	-	-	0	3.98

All the results presented in this subchapter were generated by performing 100 iterations for each repeat-unit size and using a set of 300 probes. The probes in the set were randomly selected from all possible probes of each size (see Appendix A for further test scenario information).

During the first phases of the test scenario design and conductance, algorithm runs were performed using 150 probes and 10 versus 20 iterations respectively (results not shown). It was however decided to increase both the number of iterations and the number of probes in the set to decrease the standard deviation in the results. The parameter setting of 100 iterations and 300 probes was decided to be appropriate, based on several test runs, which were successively refined until the standard deviation reached acceptable values.

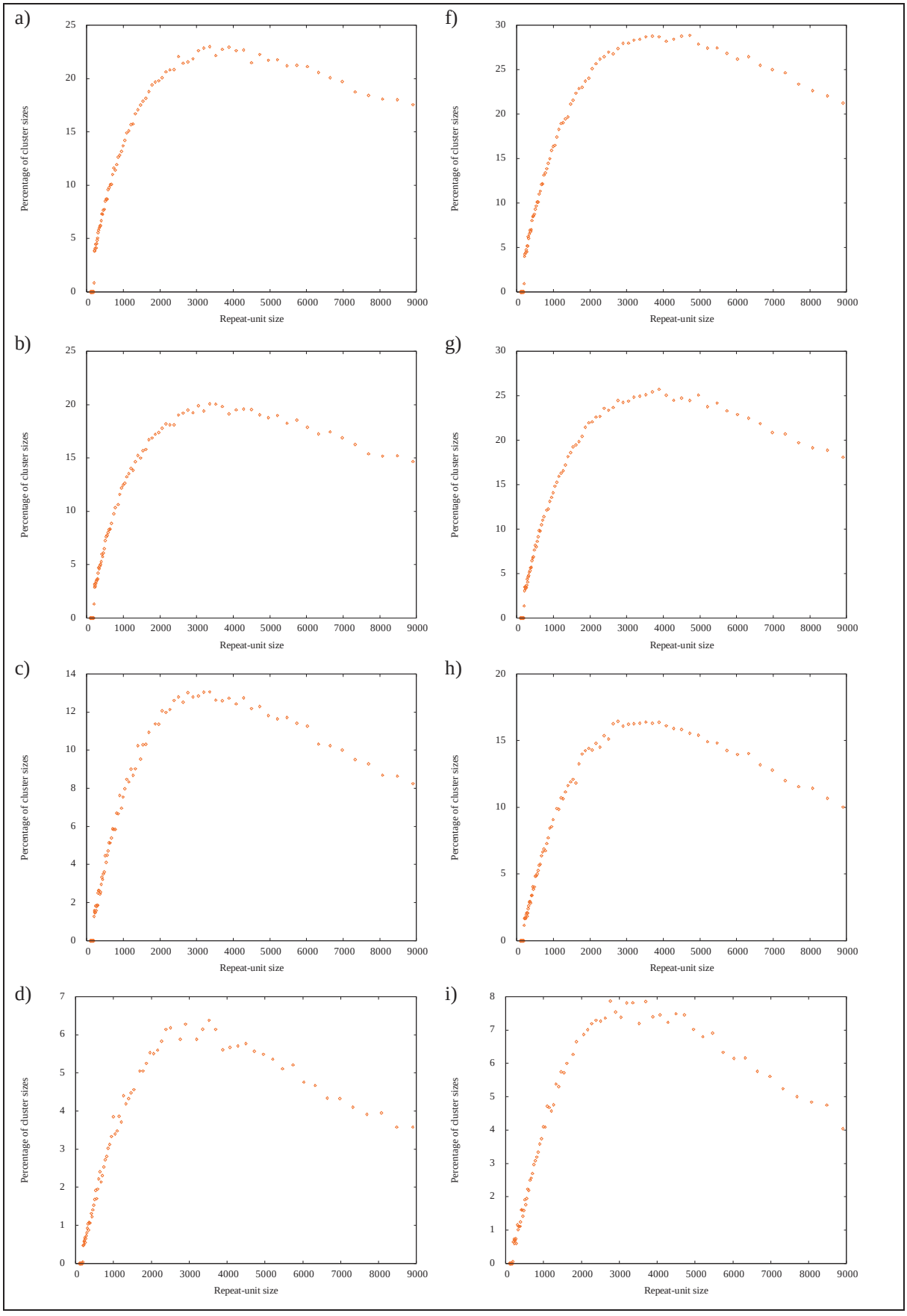
6.4 Repeat-unit sensitivity analysis for mutated megasatellites

The runs performed generating the results in this subchapter were similar to the runs performed in subchapter 6.3. The differentiating aspect was that the artificial megasatellites, created for each algorithm run, were mutated by a specific mutation percentage. The mutation levels were set to 1%, 2%, 5%, 10%, and 20%. The purpose of these runs was to retrieve basic statistics regarding the sensitivity of the algorithm for various repeat-unit sizes, using hexamer probes as representatives, adding the effect of base pair dissimilarity between repeat-units. The main intention was to elucidate the maximum repeat-unit size sensitivity for hexamer probes, as well as to find out at which repeat-unit sizes the signal faded out and to compare the output to the generated results in subchapter 6.3. Likely results for other probe sizes can be inferred, using the results based on hexamer probes.

Subchapter 6.4.1 provides the results of the detection process when the artificial megasatellites have been mutated before the sequence scan, and subchapter 6.4.2 provides the results when altering the length variation tolerance as well.

6.4.1 Mutating the artificial megasatellites

Figure 6.3 visualises the mean sensitivity for each of the mutation levels, when running the algorithm on the real DNA sequence (parts a) to e)), and on a random sequence generated by applying the 40/60 rule (parts f) to j)).


a)
Percentage of cluster sizes
Repeat-unit size
b)
Percentage of cluster sizes
Repeat-unit size
c)
Percentage of cluster sizes
Repeat-unit size
d)
Percentage of cluster sizes
Repeat-unit size
f)
Percentage of cluster sizes
Repeat-unit size
g)
Percentage of cluster sizes
Repeat-unit size
h)
Percentage of cluster sizes
Repeat-unit size
i)
Percentage of cluster sizes
Repeat-unit size

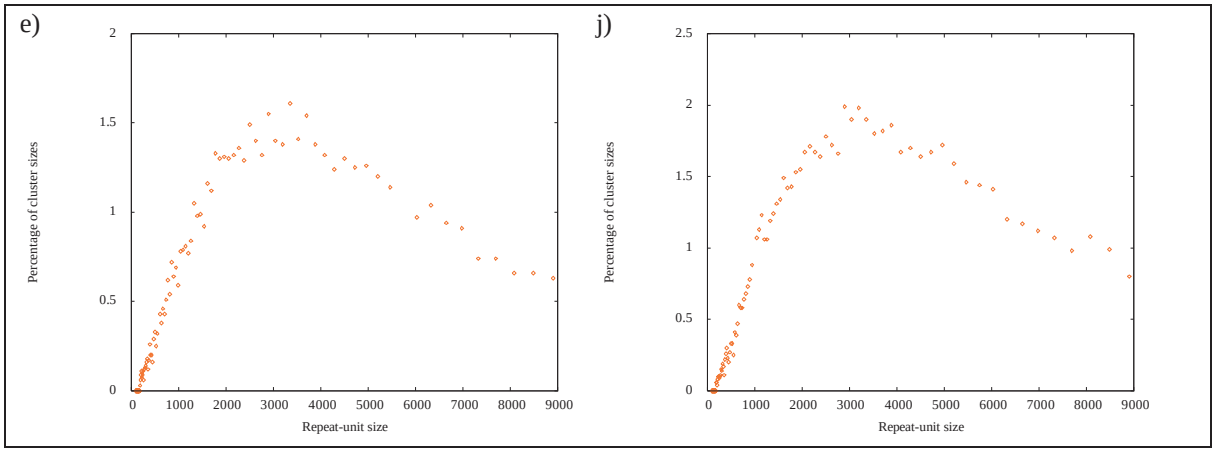


Figure 6.3 The mean sensitivity for each of the mutation levels using hexamers, when running the algorithm on a real DNA sequence extracted from the human genome (parts a) to e)), and on a random sequence generated by applying the 40/60 rule (parts f) to j)). Mutation levels: 1%, 2%, 5%, 10%, and 20% (top-down order). The repeat-unit size is indicated on the x-axis and the percentage of cluster sizes is indicated on the y-axis. A cluster size is defined as the size of the resulting cluster, i.e. the hit, and provides the same results as the successful probe percentages in subchapter 6.3.

The last part of the curves, i.e. the baseline of the number of hits, is displayed in Figure O.3 in Appendix O. Notice however that the curves in Figure O.3 do not reach zero, i.e. do not fade out. The results are summarised in Table 6.6. Tables 6.7 and 6.8 provide a comparison between success rates when mutating the artificial megasatellites.

As previously mentioned, the two sequences which are used in this test scenario are the real DNA sequence and the random sequence generated by applying the 40/60 rule. The random sequence, generated by applying an equal probability of 25% per base, is not used in this scenario since the intention was to compare the results when using a real sequence versus using a sequence that was created to simulate the nature of a real nucleotide sequence.

Table 6.6 The expressed maximum sensitivity for the hexamer probes and the baseline sensitivity. Two categories of sequences are used; the real DNA sequence extracted from the human genome and the random sequence generated by applying the 40/60 rule. Both sequences are of length 10 Mb. The sequences have been exposed to mutations of the following percentages: 1%, 2%, 5%, 10%, and 20%. The length variation tolerance of the segments between hits is kept at 1% (default setting).

Probe size	Sequence category (DNA / Random)	Mutation	Maximum sensitivity (repeat-unit size)	Signal baseline (repeat-unit size)
Hexamer	DNA	1%	~3500	~250,000
Hexamer	DNA	2%	~3500	~200,000
Hexamer	DNA	5%	~3500	~100,000
Hexamer	DNA	10%	~3500	~60,000
Hexamer	DNA	20%	~3500	~20,000
Hexamer	Random	1%	~4000	~60,000
Hexamer	Random	2%	~4000	~55,000
Hexamer	Random	5%	~3700	~45,000
Hexamer	Random	10%	~3700	~30,000
Hexamer	Random	20%	~3300	~20,000

Table 6.7 The mean cluster sizes for Megasatfinder, expressed in percent, for a repeat-unit size of 4077 bp, using a real DNA sequence extracted from the human genome.

Repeat-unit size (bp)	Probe count	Iteration count	Mutation	Mean cluster size
4077	300	100	1%	22.61
4077	300	100	2%	19.50
4077	300	100	5%	12.43
4077	300	100	10%	5.67
4077	300	100	20%	1.32

Table 6.8 The mean cluster sizes for Megasatfinder, expressed in percent, for a repeat-unit size of 4077 bp, using a random sequence generated by applying the 40/60 rule.

Repeat-unit size (bp)	Probe count	Iteration count	Mutation	Mean cluster size
4077	300	100	1%	28.19
4077	300	100	2%	25.05
4077	300	100	5%	16.11
4077	300	100	10%	7.45
4077	300	100	20%	1.67

6.4.2 Altering the length variation tolerance (l.v.t.)

Tables 6.9 and 6.10 express the mean cluster sizes for Megasatfinder, expressed in percent, for a real versus a random sequence, when varying the value of the length variation tolerance (l.v.t.). These tables can be compared directly to Tables 6.7 and 6.8. There are however some differences between the tables. The length variation tolerance percentage of 0% does not result in any output (see Table 6.1), the 1% level has already been tested (subchapter 6.4.1), and the 10% level generates erroneous results (subchapter 6.1). The l.v.t. percentages left are therefore 0.5%, 2%, and 5%. The mutation percentages of 2% and 5% were selected based on the intuition that these levels would clearly show variation in accordance to changes in the l.v.t. percentages.

Table 6.9 The mean cluster sizes for Megasatfinder, expressed in percent, for a repeat-unit size of 4077 bp, a set of 300 probes, and 100 iterations, using a real DNA sequence extracted from the human genome. The mutation levels of 2% and 5% were selected as representatives for comparison to the results in subchapter 6.4.1 (Table 6.7). The length variation tolerance percentages of 0.5%, 2%, and 5% were similarly selected as representatives for comparison purposes.

Repeat-unit size (bp)	Probe count	Iteration count	Mutation	L.V.T.	Mean cluster size
4077	300	100	2%	0.5%	18.77
4077	300	100	2%	2%	19.96
4077	300	100	2%	5%	19.90
4077	300	100	5%	0.5%	10.56
4077	300	100	5%	2%	13.12
4077	300	100	5%	5%	13.45

Table 6.10 These results are generated by performing the same tests as performed for generating the results in Table 6.9, except that the random sequence generated by applying the 40/60 rule is used instead. The mutation levels of 2% and 5% were selected as representatives for comparison to the results in subchapter 6.4.1 (Table 6.8).

Repeat-unit size (bp)	Probe count	Iteration count	Mutation	L.V.T.	Mean cluster size
4077	300	100	2%	0.5%	24.31
4077	300	100	2%	2%	25.87
4077	300	100	2%	5%	25.46
4077	300	100	5%	0.5%	13.39
4077	300	100	5%	2%	16.75
4077	300	100	5%	5%	17.61

6.5 An example of a novel, unreported megasatellite, found by extensive genomic scans and confirmed by southern blotting

Extensive scans were performed on the human genome (Build 34), which resulted in a total of 207 hits. The results were clustered and ranked based on the clustering output. A group of 25 top-score hits were selected, along with 22 other hits from the list, selected using other criteria by the laboratory. The potential megasatellites, summing up to a total of 47, were sent to the laboratory for further analysis and confirmation. The results of the southern blotting (see subchapter 2.6) confirmed that 23 of these were polymorphic and therefore established as real megasatellites. A smaller group of 14 megasatellites were monomorphic, and finally it was not possible to test a group of 9 megasatellites due to various problematic issues. Keep in mind that only a fraction of the list of possible megasatellites has yet been analysed in the laboratory, implying that there are several others present in the human genome. Among the ones remaining in the list are the known megasatellites RS447 and D4Z4 (see subchapters 2.4.1-2).

One of the top-score megasatellites is a novel megasatellite which has not previously been reported in the literature. It is referred to as *lstr104* in-house at deCODE Genetics Inc. The lstr104 is an excellent example of a megasatellite due to its pureness and the fact that it seems to contain CpG-islands which might control the expression of the proximate gene ZFP37, known to be involved in the development of the central nervous system (CNS). There is a TATA-box (approximately 200 bp) immediately upstream of the megasatellite start and the first exon of the ZFP37 gene starts approximately 2 kb upstream of the megasatellite on the negative strand. The megasatellite has a repeat-unit size of approximately 5.4 kb and is positioned on chromosome 9 (9q32). Figure 6.4 depicts a snapshot of the region, taken from the UCSC genome browser, clearly showing the relative positions of the ZFP37 gene, the CpG-islands, and the megasatellite.

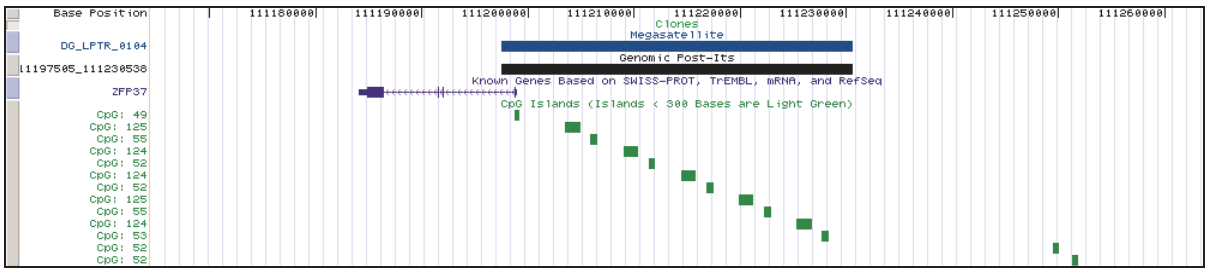


Figure 6.4 A snapshot of the megasatellite, referred to as lstr104 at deCODE Genetics Inc., and its closest surroundings, taken from the UCSC genome browser (http://genome.ucsc.edu). The figure shows the relative positions of the ZFP37 gene and the CpG-islands to the megasatellite.

A dot plot self-sequence comparison, conducted for the megasatellite sequence, further strengthens the inference that the lstr104 is a polymorphic tandem repetitive sequence, i.e. a megasatellite. This comparison is illustrated in Figure 6.5, where the megasatellite takes a rectangular form (middle of the figure). The figure also indicates that the sequence surrounding the megasatellite is free of local LCRs (low-copy repeats), except for a single inverted duplication located approximately 20 kb downstream of the megasatellite (the straight line above the megasatellite rectangle).

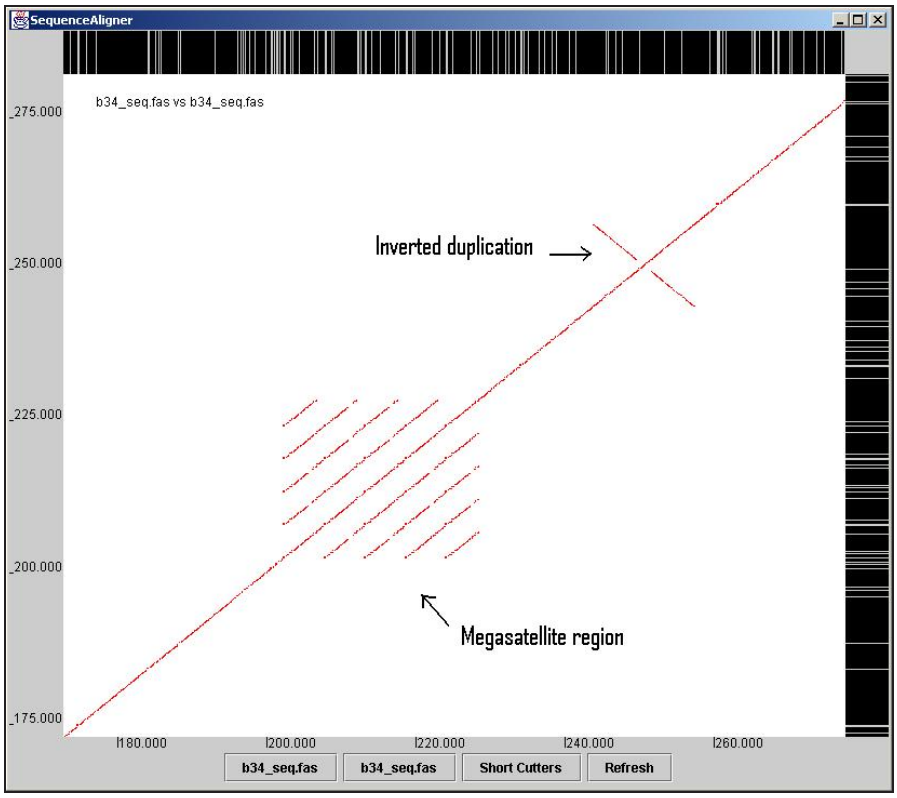


Figure 6.5 A dot plot self-sequence comparison, conducted for the megasatellite lstr104. The megasatellite sequence is compared to itself, base by base. Identical bases are marked with a dot on the diagonal. The sequence runs both along the x-axis and the y-axis.

A feature map of the human genome (Build 34) is shown in Figure 6.6, providing a positional correspondence between the various sequences discussed. The nature of this figure is similar to Figure 6.4, which was retrieved from the UCSC genome browser.

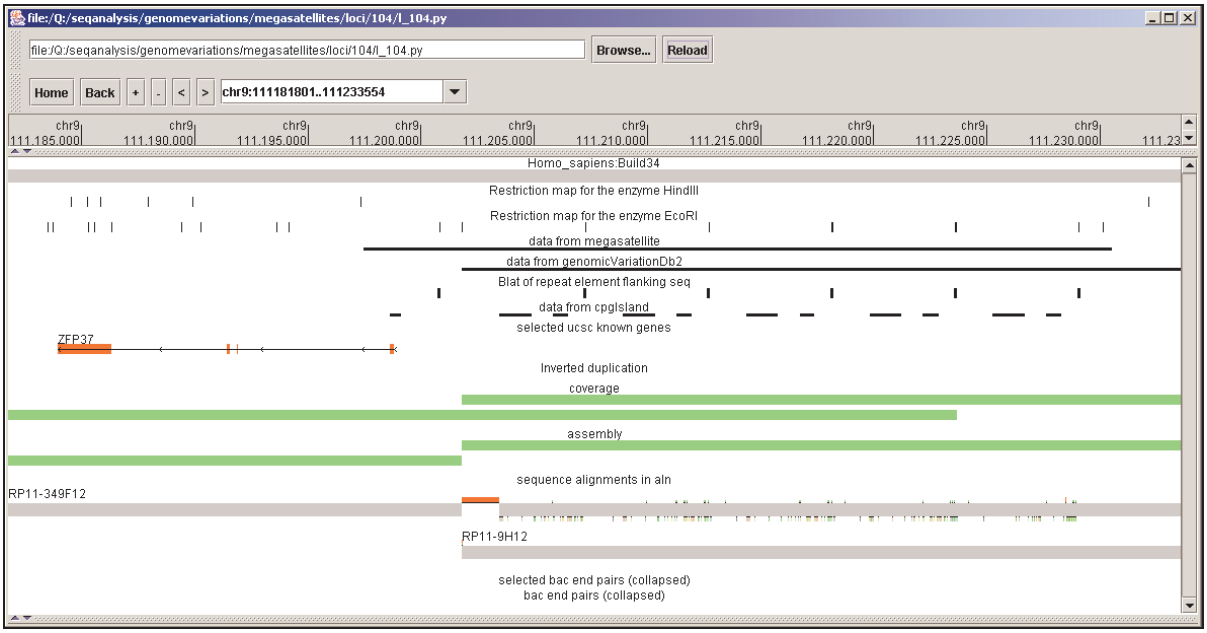


Figure 6.6 A feature map of the human genome, illustrating the positions of restriction sites for the restriction enzymes HindIII and EcoRI and the corresponding cut-sites in the megasatellite. The CpG-islands can be found around the centre of the figure, proving that the CpG-islands are located within the megasatellite sequence. Furthermore, the ZFP37 gene is depicted at the middle left of the figure. Notice how the end of the sequence overlaps with the first part of the megasatellite. The sequences marked RP11-349F12 and RP11-9H12 are two clone sequences, and according to Build 34 these clones overlap inside the megasatellite. The nature of this figure is similar to Figure 6.4, which was retrieved from the UCSC genome browser (http://genome.ucsc.edu).

6.6 Complexity analysis

The complexity analysis was conducted for three different scenarios. The first one shows the time required to perform complete genomic scans for each of the four probe sizes. The second scenario shows the amount of time it takes to run the algorithm on a 10 Mb DNA sequence, using various probe counts and number of iterations for each of the four probe sizes. These tests were performed for a single repeat-unit size, which expressed the maximum sensitivity for each of the probe sizes, according to the results in subchapter 6.3. The third and final scenario shows the time it takes to perform the detection process using hexamers on sequences which have been mutated variously. These measurements and the retrieved results provide good indications regarding the amount of time it takes to run the algorithm for various algorithm settings. The absolute time measurements are however dependant on the hardware used, and should therefore only be taken as general indications of time complexity. All run-time tests were performed on a Pentium IV 2.4 GHz computer with 2 Gb of RAM.

Table 6.11 shows the run-time for complete genomic runs, using 100 iterations and 300 patterns, for each of the four probe sizes. Table 6.12 shows the run-time when running the algorithm, using various probe counts and number of iterations, for each of the four maximum sensitivity repeat-unit sizes from Table 6.3. Finally, Table 6.13 shows the run-time when running the algorithm on mutated sequences for hexamer probes. By comparing Tables 6.12 and 6.13 one can clearly notice the increase in run-time when mutating the artificial megasatellites.

By applying these numerical results in an empirical analysis, it can be deduced that the time complexity of Megasatfinder is linear relative to sequence length, probe size, repeat-unit size, the number of probes used, and the number of iterations performed. Remember that the important concept is that the time complexity of the algorithm is *linear*, and not which absolute run-time values are retrieved, since the measurements are based on a specific hardware setup. The time complexity of Megasatfinder, based on the empirical analysis and expressed using big-O notation, is $O(n)$.

Table 6.11 The derived run-time for each of the four probe sizes. The runs were performed on the whole human genomic assembly (Build 34). The rest of the parameters were kept at their default values. The runs were performed on a Pentium IV 2.4 GHz computer with 2 Gb of RAM.

Probe count	Probe size	Genome, Build	Time (h:min:sec)
300	5	Homo sapiens, Build 34	11:28:06
300	6	Homo sapiens, Build 34	10:02:04
300	7	Homo sapiens, Build 34	09:45:48
300	8	Homo sapiens, Build 34	09:18:31

Table 6.12 The derived run-time for each of the four probe sizes, using a mixture of 150 versus 300 probes and 50 versus 100 iterations. The runs were performed for the repeat-unit size which resulted in the maximum sensitivity for each probe size. The rest of the parameters were kept at their default values. A 10 Mb real nucleotide sequence extracted from the human genome was used. The runs were performed on a Pentium IV 2.4 GHz computer with 2 Gb of RAM.

Probe count	Iteration count	Probe size	Repeat-unit size	Time (sec.)
150	50	5	1000	2
150	100	5	1000	5
300	50	5	1000	5
300	100	5	1000	11
150	50	6	4000	7
150	100	6	4000	13
300	50	6	4000	12
300	100	6	4000	24
150	50	7	16,000	20
150	100	7	16,000	39
300	50	7	16,000	40
300	100	7	16,000	79
150	50	8	65,000	76
150	100	8	65,000	161
300	50	8	65,000	153
300	100	8	65,000	306

Table 6.13 The derived run-time for hexamers. The runs were performed for the repeat-unit size which resulted in the maximum sensitivity for the probe size, i.e. 4000 bp (base pair). The rest of the parameters were kept at their default values. The algorithm was run on a 10 Mb (megabase) real DNA sequence, extracted from the human genome. The sequence was mutated, using five different mutation levels. The runs were performed on a Pentium IV 2.4 GHz computer with 2 Gb of RAM.

Probe count	Iteration count	Probe size	Repeat-unit size	Mutation	Time (sec.)
300	100	6	4000	1%	29
300	100	6	4000	2%	29
300	100	6	4000	5%	30
300	100	6	4000	10%	32
300	100	6	4000	20%	36

7 Analysis and discussion

7.1 General interpretation of the results

The false positive hit statistics for random sequences from subchapter 6.1 provide valuable indications regarding the general reliability of results. Firstly, the effect of increasing the number of probes in the probe set used means that a higher number of clusters are found for each cluster size, not that more cluster sizes are reported. These results support what could have been predicted, i.e. that using a larger probe set, thus containing a greater variety of base combinations, results in more hits. The results also indicate that the number of cluster sizes is primarily contingent upon the length variation tolerance (l.v.t.) percentage rather than the probe count. However, a combined effect is obtained when both the probe count and the l.v.t. percentage have reached the highest levels. An example of this is the test performed using a probe set of 300 probes with an l.v.t. of 5% (see Table 6.1). The effect of increasing the allowed l.v.t. is on the other hand obvious, no matter the probe count used. Secondly, considering the difference between the various l.v.t. percentages, a few things can be pin-pointed. Using an l.v.t. of 0% is simply stated unreasonable. This percentage allows no variation between the repeat-units whatsoever. In light of the genetic variation evident in the human genome, such a limitation would result in no, or at least a minimal number of hits, thus excluding true positive hits in a nucleotide sequence. The two levels 0.5% and 1% generate acceptable levels of by-chance results, meaning that generally clusters of sizes 2 and 3 would have to be considered as unreliable results, or at least treated with extra precaution. The subsequent level, 2%, is really a maximum variation percentage in this context. As can be seen in Table 6.1, the 5% level (and higher) results in far too much noise, rendering the results as quite unreliable. This behaviour is mainly due to the implementation of the algorithm. When the allowed variation is increased beyond reasonable limits the detected clusters, formerly positioned in close proximity, merge into larger clusters since these are currently overlapping because of the looser requirements. In general, Table 6.1 clearly shows that as the l.v.t. percentage is increased, the more cluster sizes can be ignored or at least treated as unreliable.

In relation to the false positive hit statistics for DNA sequences in subchapter 6.2, a hypothesis was put forward in the motivation section of test scenario 2 (subchapter 5.2). It focused on the question whether a sequence, generated by applying the 40/60 rule, could be considered sufficient as a model for computer simulations, i.e. whether this random sequence resembles the DNA sequence enough to be used for analysing the algorithm (and possibly other similar algorithms). Based on the results, displayed in Table 6.2, the answer is no. By comparing the output data, the results indicate that the number of clusters found using the real DNA sequence is considerably lower than for the random sequence. This result can perhaps be supported by the fact that the human genomic sequences are not created randomly by applying simple rules, and can therefore not be replaced by randomly generated data. In general, the results manifest that mimicking real DNA data is a difficult task, and that real nucleotide sequences should be used to the fullest extent for all research and analysis purposes. The rejection of the hypothesis is further supported by the results in subchapters 6.3 and 6.4.

Subchapter 6.3 provided an important overview of the sensitivity for each of the four probe sizes (pentamers, hexamers, heptamers, and octamers). The curves were created based on the numerical results, using a real DNA sequence, and illustrate that their shapes are extremely similar for each of the four probe sizes (see Figure 6.1). There are however two main differences; firstly how quickly the maximum sensitivity is reached, and secondly how steep the slope is after maximum sensitivity is reached, i.e. how quickly the signal fades out. The above also applies to the curves generated when random sequences are used instead of the real one (see Figure 6.2). An interesting aspect is however that although the maximum sensitivity repeat-unit sizes are similar for real and random sequences (see Table 6.3), the differences lie in the percentage of successful probes and the repeat-unit sizes for which the signals reach zero. Interestingly, the percentage of successful probes was approximately 25% when scanning the real DNA sequence (see Table 6.4), while the same value was significantly higher, approximately 32%, for the random sequences (see Table 6.5). The differences in successful probes between probe sizes were trivial. The fade-out repeat-unit sizes were however much larger for the DNA sequences than the random sequences. In addition, the size of the fade-out repeat-unit increased as probe size was increased.

Similar tests were performed in subchapter 6.4, with the additional effects of mutating the artificially generated megasatellites and varying the l.v.t. percentage. By inspecting Figure 6.3, four interesting observations can be made. Firstly, as the mutational force is increased, the percentage of cluster sizes decreases. This is in accordance to what could have been foreseen, since randomly changing the sequence of bases means that the selected probe will be less likely to detect the existing megasatellite. Secondly, the higher the mutational force is, the steeper the slope is after reaching the maximum sensitivity. This implies that the signal fades out at a faster pace as the mutational percentage is increased. The third observation is that higher percentages of cluster sizes are reached for all mutation levels using random sequences, as was the case for non-mutated megasatellites. The fourth and final aspect is the fact that the standard deviation is higher when the artificial megasatellites have been mutated, and the deviation increases as the mutation level is increased. This behaviour is especially evident around the peak of each curve, as well as around the baseline cluster sizes, making it increasingly difficult to determine maximum sensitivity and a baseline. In subchapter 6.4, the reader was encouraged to notice the baseline of the number of hits, displayed in Figure O.3, since the curves did not fade out (reach zero) at any point. This is due to the noise brought about by false positive clusters which pollute the results. In reality, the cluster size value of zero should be reached at a certain point, based on probe size and mutation level, in a similar way as for the non-mutated artificial megasatellites. The summarised results in Table 6.6 show that the baseline repeat-unit sizes, as for the fade-out sizes using non-mutated sequences, are smaller for random sequences than the real DNA sequences, thus supporting previous results. The surprising result is the difference in maximum repeat-unit sizes based on the mutation level, a difference which is also present between the real and random sequences. All repeat-unit sizes, no matter the mutation level, are approximately the same for real DNA sequences, or around 3500 bp, which is about 500 bp below the results for non-mutated megasatellites (Table 6.3). These repeat-unit sizes for random sequences are however not uniform. The sizes start at approximately 4000 bp for the 1% and 2% mutation levels, decreasing to around 3300 bp for the 20% mutation level. These tests were repeated to confirm the results, due to their unexpected nature.

Furthermore, these tests confirmed an additional hypothesis, namely that the mean cluster size would decrease as the mutation level was increased. The assumption is supported by the results in Table 6.7. As can be seen in Table 6.8, the same applies for random sequences, with the only difference that the mean cluster sizes are larger for each of the mutation levels in accordance to previous results. When taking a closer look at the two tables (6.7 and 6.8), it can easily be deduced that the 10% and 20% mutation levels are in fact far too high for the retrieval of valuable results. Furthermore, the author would not even consider the mutation level of 5% as an optional mutation percentage. This is due to the fact that although mutations occur within the human genome, these are not as extensive as these experimental percentages.

The second part of subchapter 6.4.2 included a variation in the l.v.t. parameter settings as previously mentioned. As can be seen in Table 6.9, the mean cluster size is practically unaffected by the variation of l.v.t., applied to both the 2% and the 5% mutation levels. The main jump is from the 0.5% l.v.t. to the 2% l.v.t. This also applies to the results when using random sequences (Table 6.10), were again the mean cluster sizes are larger than for the real DNA sequence.

The results from running the algorithm on the entire genome, displayed in subchapter 6.5, demonstrate that the algorithm detects a lot of possible megasatellites in the human genome. The exact number is of course based on the selected parameter settings, but this still indicates that there are a lot of interesting sites worth exploring further. In addition, the findings confirm that there are undetected megasatellites still waiting to be discovered and perhaps connected to anomalies in humans.

The final test scenario involved a complexity analysis. The author would again like to remind the reader that the numerical values are only indications of algorithm complexity and that these are dependent on the hardware used. The analysis method applied is empirical, i.e. based on experimental test-run result values. This analysis could also have been conducted theoretically, i.e. by inspecting the detailed implementation of the algorithm, e.g. with regard to various loops or other time consuming sections. There are two reasons why the empirical analysis method was selected instead of the theoretical one: Firstly, the information obtained from the empirical analysis was of much higher interest to the scientists at deCODE Genetics, and secondly it would have been very laborious to conduct a theoretical analysis on Megasatfinder. The analysis however showed that the algorithm is linear. It is interesting to inspect the time measurements for the genomic runs in Table 6.11. These mainly show two things; firstly that the run-time decreases as the probe size is increased, and secondly that a full genomic scan can be achieved in less than 12 hours, given that the processes for the four probe sizes are started at the same time on different machines comprised of the same hardware. The decreasing run-time for the increasing probe size is a quite natural consequence, since larger fragments require fewer comparisons to the sequence in question. Table 6.12 reveals a few noteworthy aspects of the algorithm function. Firstly, run-time increases as the repeat-unit size is increased. Remember that the run-time decreased as probe size was increased, so obviously there is not a correspondence between the two parameters. Secondly, the run-time increases as iteration count, probe count, or both are increased. Notice how similar effect these two parameters have. This can be seen by comparing rows 2 and 3 for each repeat-unit size in Table 6.12. Take a look at the results for pentamers for example: The parameter settings of 150 probes and 100 iterations result in

the same run-time as the settings of 300 probes and 50 iterations. This implies that the multiplicity effect of altering either value is the same for both parameters. Finally, Table 6.13 shows that the run-time is also due to the mutational force placed upon the artificial megasatellite, since the mutation tool Msbar requires some time to perform the various megasatellite sequence base changes. The time required increases as the mutation percentage is increased. The algorithm run-time might seem a bit long and it can certainly be optimised to perform faster. The algorithm is partially limited by the language in which it is written, i.e. Python, and partially by the way some things are handled with respect to implementation details. The algorithm would e.g. execute faster if written in C or C++, although the programming language is the factor which affects run-time the least when considering algorithm complexity. An example of suboptimal handling regarding implementation details is that all sequences are read into memory first, making the algorithm suboptimal with regards to run-time. One should for example read the sequence into a buffer as the search is in process. Further improvements regarding implementation can not be stated, since the analysis is performed empirically.

7.2 Generality of results

A set of three nucleotide sequences are used in the test scenarios. The real DNA sequence extracted from the human genome provides indications of usability and accuracy of the algorithm, when using an example sequence not generated by artificial means. When deciding which sequence size would be appropriate to use, two objectives guided the decision process: Is the sequence long enough to capture the general structure of the human genome (if one exists) and is the sequence short enough to enable acceptable algorithm run-time. After elaborate considerations, a sequence size of 10 Mb was decided as appropriate.

An important aspect regards the question whether the sequence is long enough to be able to represent the variety of the entire human genome. As discussed in the introduction chapter, the genome is comprised of approximately 3 billion base pairs. It was concluded that a 10 Mb sequence would provide enough data to express the general structure of the genomic sequences and at the same time be suitable regarding run-time. It is possible that increasing the length of the sequence would slightly improve the resulting values for each of the test scenarios, i.e. improve their accuracy, but at the same time believed that this would not affect the result of the complete analysis. Another aspect is the homogeneity of the human genome, raised by the fact that the example sequence was extracted from chromosome 1. It is possible, and in fact highly likely, that another sequence extracted either from another part of chromosome 1 or another chromosome altogether is different from the one used to some extent. However, the idea behind using a single real DNA sequence was only to provide general indications of performance, not the absolute values for each possible case. Requiring absolute values applicable to the whole genome implies that an extensive amount of sequences would have to be extracted from various parts of each chromosome, and using all of them during the tests would dramatically elongate the time-frame of the project.

The sequence generated applying the 40/60 rule and the sequence generated applying an equal probability of 25% per base are long enough for analysis purposes, since these are artificially generated by applying a known ratio between the bases.

7.3 Algorithm improvements

Objective 2.2 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 2. The objective states the following:

> *Perform possible alterations in the implementation of the algorithm, depending on the result of the complete analysis. These alterations could e.g. include various changes and/or extensions to the algorithm or changes regarding the default parameter settings.*

The objective above includes both program code alterations and the more lenient default parameter improvements. As a result of the complete set of tests conducted, which proves the functionality of Megasatfinder for the detection of megasatellites in the human genome, major algorithm alterations are considered unnecessary by the author. However, specific improvements or alterations regarding implementation details can not be elucidated, since the time complexity analysis was performed empirically. There is one aspect concerning time complexity which might slightly reduce run-time. The algorithm is written in Python which, as mentioned in subchapter 7.1, implies slower run-time than e.g. using C or C++. Therefore, rewriting Megasatfinder into either of the two languages would diminish run-time. This would however only be beneficial if the algorithm is to be used extensively and not occasionally as now.

The default parameter settings are on the other hand worth changing, based on the success of the conductance of the various tests. The probe size is a parameter constantly undergoing changes by the user, so the default setting of 6 (hexamer) is just a formality. An idea would be to somehow introduce the four probe sizes to the user, i.e. pentamers, hexamers, heptamers, and octamers, which are sensible to apply in the search for megasatellites. Limiting the algorithm to these four probe sizes is however unnecessary. The parameter indicating the minimum number of consecutive segments of similar size can be left untouched by the novice user. It is currently set to 3, which seems to be an appropriate setting since it detects most of the possible megasatellites without including the noise present for a parameter setting of 2. The allowed length variation tolerance (l.v.t.) is presently set to 1%. A closer look at the numerical results in subchapters 6.1 and 6.2 strongly indicates that this default value is appropriate. A lower value would probably be too stringent, i.e. exclude true positive hits, and a higher value would probably allow too much variation between the repeat-units, thereby including false positive hits. The l.v.t. parameter is therefore only to be altered under special circumstances by advanced users. Similarly, the minimum size of the segments between hits is a parameter not to be changed by the user. The current value of 200 was mainly set to minimise noise by the shorter sequences. Remember that according to the formal definition of a megasatellite, the minimum repeat-unit size is 1000 bp. It would therefore be advisable to increase this minimum value. An idea is to set the value at 900 bp, i.e. just below the defined minimum.

The current parameter setting for the number of input probes is 150. In accordance to the discussion in subchapter 6.3, it would be preferable to change this value to 300 in order to diminish the standard deviation in the results. This requires longer run-time, but the gain is higher due to the retrieval of more accurate results. It is however suggested that the user is able to modify this parameter. Cluster size is most certainly a parameter which could be left alone for most runs. A cluster size of 2 is default, which is of course a

minimum since it is difficult to define a single hit as a cluster. A more stringent demand, i.e. a higher value, might exclude true positive hits. Finally, as discussed in subchapter 6.3, performing iterative runs on a shorter sequence decreases random variation in the generated results. Initially, the test runs used an iteration count of 10. This resulted in extensive random variation. The next step was to try an iteration count of 20. This setting gave slightly better results. Finally it was decided to try runs using 100 iterations. This iteration count provided results with low standard deviation. Using a higher number of iterations slightly improves the results, with the addition of greatly increasing run-time.

7.4 Guidelines for algorithm usage

Objective 2.3 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 2. The objective states the following:

> *Develop guidelines for the use of the algorithm with respect to parameter settings. These parameter settings should maximise the two concepts usability and accuracy on a case to case basis.*

In conclusion to the previous discussion in subchapter 7.3, the user, or at least the novice or intermediate user should not have to change the default settings, i.e. including the changes suggested above, except for providing the algorithm with the appropriate probe size to be used. These results increase usability and provide the user with a sense of result accuracy. The suggested set of default parameter settings can be found in Table 7.1 below.

Table 7.1 The suggested set of default parameter settings for Megasatfinder. Parameters in bold should be controlled by users, while others can be left at their default values in most cases. Exceptions should only be performed by expert users. Parameters in italics have changed their values from the original settings.

Parameter	**Suggested default value**
Probe size	**5 / 6 / 7 / 8**
Number of consecutive segments of similar size	3
Variation in length of segments between hits (l.v.t.)	0.01
Minimum size of segments between hits	*900*
Number of input probes (probe count)	*300*
Cluster size	2
Number of iterations (iteration count)	*100*

8 Conclusions

8.1 Conclusions

Objective 2.4 (see subchapter 3.2) is one of the objectives that must be fulfilled in order to achieve aim 2. The objective states the following:

> *An important part of the work is also to better understand the output of the algorithm in general.*

In light of the experimental results in Chapter 6 and the analysis and discussion in Chapter 7 it can be stated that a comprehensive overview has been achieved regarding the overall algorithm function and the nature and meaning of the generated results. This should provide an insight which helps the user to understand the general meaning of the algorithm output.

The first aim, which was divided into a set of four objectives, has been achieved. The study of related work provided insight into other algorithms and how these were analysed. Although the study resulted in some ideas, the only idea used directly in the new analysis method for Megasatfinder was complexity analysis. This reveals the immense number of various methods, each dependant upon the nature of the algorithm. Furthermore, it shows the difficulty of matching an existing analysis method to a novel algorithm, based on a different approach. It would however be pleasant if others would find some of the test scenarios created for Megasatfinder inspiring for their own evaluation. The analysis method, consisting of a set of six test scenarios, has proven helpful in resolving the various question marks surrounding the algorithm and its general function. The extracted and created nucleotide sequences were necessary for the execution of these tests, controlled by the generated Python scripts.

The second aim, also divided into a set of four objectives, has also been achieved. The new analysis method was used on the algorithm, generating the results used in the analysis. The analysis indicated that any major algorithm alterations were not necessary, or at least not based on the current degree of usage. However, specific improvements or alterations regarding implementation details could not be elucidated, since the time complexity analysis was performed empirically. Some minor changes were however suggested to the default parameter settings. Based on these suggestions, a set of guidelines was created, which assists the user in the application of Megasatfinder. The combined results finally provide the user with confidence and general knowledge of usability and accuracy of results.

8.2 Future work

The work of most researchers has mainly been centered on micro- and minisatellites, mainly due to their visibility in the genomic sequences and the strong correlation to TRDs (Benson, 1999; Kolpakov et al., 2003). This does not imply that research involving megasatellites has been left completely untouched. However, for some reason, scientists have perhaps directed their focus more at analysing in detail the few megasatellites that have been confirmed, rather than actively seeking for new ones. As a result of these researches, megasatellites have been shown to be associated to the cause of human disease; to have regulatory, catalytic and evolutionary roles; and to serve as important

laboratory and analytic tools (Benson, 1994; Benson, 1999). Examples of these highly popular megasatellites among scientists are RS447 and D4Z4. According to the author of this project, more has to be done in the search for other megasatellites. Retrieving additional examples using the algorithm and performing extensive analyses on these would further broaden our perspectives regarding the function and evolutionary importance of these long satellite DNA sequences. This work has already been started at deCODE Genetics Inc., but the work is still in its infancy and much still waits regarding the confirmation of possible megasatellites and the thorough analysis of these true positive hits.

In many of the test scenarios, hexamers were used as an example probe size. This was decided for two reasons; to reduce the time it would require to repeat all the tests for each probe size and because of the limited space available to report the findings in the report. This is not to be considered as a negative aspect, since the behaviour of other probe sizes can be approximated or deduced from the hexamer results. However, it would be nice to conduct these tests for the rest of the probe sizes, i.e. pentamers, heptamers, and octamers, to retrieve the exact values for a precise comparison.

Improving and further evolving the usability of the algorithm is also a stimulating task. Firstly, given that the general usage of the algorithm increases, the run-time can be decreased by rewriting it in C or C++. Secondly, it would be beneficial if a graphical user interface (GUI) and/or a web user interface was created for the algorithm. Presently, the algorithm is executed from the command line. An interface, where the user could easily set the parameters, e.g. by using various types of boxes or drop-down lists, would increase usability. The user would of course have the possibility to select and browse for input data, and select from a list of various output formats. This would be an excellent addition, especially for the novice and intermediate users.

The novelty of this work concerning megasatellites is quite high. The main contribution involves both the development of the analysis method and the results generated when applying the method on the megasatellite detection algorithm Megasatfinder, leading to valuable and much needed information regarding its usability and accuracy. This newly acquired knowledge will hopefully guide the search for novel megasatellites in the future, which has proven to be both an exciting and an important task.

Acknowledgements

This project was conducted at deCODE Genetics Inc. in Reykjavík, Iceland, during the spring term of 2005 as a final year project at the University of Skövde, Sweden.

First of all, I would like to thank deCODE Genetics Inc. for providing the opportunity to conduct this final year project at their headquarters. It would not have been established without Kári Stefánsson (President and CEO), Hákon Guðbjartsson (VP Informatics), Sverrir Þorvaldsson (Director of Data Analysis & Management), and Gísli Másson (Research Scientist). I truly appreciate all the experience and knowledge I have obtained for the past few months.

I would like to express my sincere gratitude to my supervisor at deCODE, Gísli Másson, who offered the project idea, for all his support and encouragement. He provided an invaluable guidance during the birth of this thesis. He was always ready to answer a question or two, whether it was a programming or an algorithm issue.

I would evenly like to thank my supervisor at the University of Skövde, Jane Synnergren, for her observant eye, providing insightful comments and criticism concerning the thesis. She has simply stated showed that she is one of a kind, offering constructive second-opinions and pointers, which have undoubtedly increased the quality of my thesis.

Last, but not least, I would like to express my gratefulness to my family by dedicating this work to them for all their love, support, and encouragement - not only during the last few months, but through all my studies at the university. They have always stood by me.

References

Axelrod, D. E., Baggerly, K. A. & Kimmel, M. (1994) Gene amplification by unequal sister chromatid exchange: probabilistic modeling and analysis of drug resistance data. *Journal of Theoretical Biology*, 168, 151-159.

Benson, G. (1995) A space efficient algorithm for finding the best nonoverlapping alignment score. *Theoretical Computer Science*, 145, 357-369.

Benson, G. (1998) In: S. Istrail, P. Pevzner & M. Waterman (eds.), An algorithm for finding tandem repeats of unspecified pattern size (pp. 20-29). *Proceedings of the second annual international conference on computational molecular biology*, March 22-25, 1998, New York, United States.

Benson, G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, 27, 573-580.

Benson, G. & Waterman, M. (1994) A method for fast database search for all k-nucleotide repeats. *Nucleic Acids Research*, 22, 4828-4836.

Berg, J. M., Tymoczko, J. L. & Stryer, L. (2002) *Biochemistry* (5th Edition). New York: W. H. Freeman and Company.

Bois, P., Williamson, J., Brown, J., Dubrova, Y. E. & Jeffreys, A. J. (1998) A novel unbstable mouse VNTR family expanded from SINE B1 elements. *Genomics*, 49, 122-128.

Campbell, N. A., Reece, J. B. & Mitchell, L. G. (1999) *Biology* (5th Edition). California: Benjamin/Cummings.

Charlesworth, B., Sniegowski, P. & Stephan, W. (1994) The evolutionary dynamics of repetitive DNA in eukaryotes. *Nature*, 371, 215-220.

Delgrange, O. & Rivals, E. (2004) STAR: an algorithm to search for tandem approximate repeats. *Bioinformatics*, 20, 2812-2820.

Edwards, A., Hammond, H., Jin, L., Caskey, C. & Chakraborty, R. (1992) Genetic variation at five trimeric and tetrameric tandem repeat loci in four human population groups. *Genomics*, 12, 241-253.

Giacalone, J., Friedes, J. & Francke, U. (1992) A novel GC-rich human macrosatellite VNTR in Xq24 is differentially methylated on active and inactive X chromosomes. *Nature Genetics*, 1, 137-143.

Gondo, Y., Okada, T., Matsuyama, N., Saitoh, Y., Yanagisawa, Y. & Ikeda, J. (1998) Human megasatellite DNA RS447: Copy-number polymorphisms and interspecies conservation. *Genomics*, 54, 39-49.

Hamada, H., Seidman, M., Howard, B. & Gorman, C. (1984) Enhanced gene expression by the poly(dT-dG).poly(dC-dA) sequence. *Molecular and Cellular Biology*, 4, 2622-2630.

Hewitt, J. E., Lyle, R., Clark, L. N., Vallely, E. M., Wright, T. J., Wijmenga, C., van Deutekom, J. C. T., Francis, F., Sharpe, P. T., Hofker, M., Frants, R. R. & Williamson, R. (1994) Analysis of the tandem repeat locus D4Z4 associated with facioscapulohumeral muscular dystrophy. *Human Molecular Genetics*, 3, 1287-1295.

Iafrate, A. J., Feuk, L., Rivera, M. N., Listewnik, M. L., Donahoe, P. K., Qi, Y., Scherer, S. W. & Lee, C. (2004) Detection of large-scale variation in the human genome. *Nature Genetics*, 36, 931-932.

Kannan, S. K. & Myers, E. W. (1996) An algorithm for locating nonoverlapping regions of maximum alignment score. *Society for Industrial and Applied Mathematics Journal on Computing*, 25, 648-662.

Karlin, S., Morris, M., Ghandour, G. & Leung, M. Y. (1988) Efficient algorithms for molecular sequence analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 85, 841-845.

Karolchik, D., Baertsch, R., Diekhans, M., Furey, T. S., Hinrichs, A., Lu, Y. T., Roskin, K. M., Schwartz, M., Sugnet, C. W., Thomas, D. J., Weber, R. J., Haussler, D. & Kent, W. J. (2003) The UCSC genome browser database. *Nucleic Acids Research*, 31, 51-54.

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M. & Haussler, D. (2002) The human genome browser at UCSC. *Genome Research*, 12, 996-1006.

Kogi, M., Fukushige, S., Lefevre, C., Hadano, S. & Ikeda, J. (1997) A novel tandem repeat sequence located on human chromosome 4p: isolation and characterization. *Genomics*, 42, 278-283.

Kolpakov, R., Bana, G. & Kucherov, G. (2003) mreps: efficient and flexible detection of tandem repeats in DNA. *Nucleic Acids Research*, 31, 3672-3678.

Landau, G. M. & Schmidt, J. P. (1993) In: A. Apostolico, M. Crochemore, Z. Galil & U. Manber (eds.), An algorithm for approximate tandem repeats (pp. 120-133). *Proceedings of the 4th annual symposium on combinatorial pattern matching*, June 02-04, 1993, Berlin, Germany.

Lander, E. S. et al. (2001) Initial sequencing and analysis of the human genome. *Nature*, 409, 860-921.

Lemmers, R. J. L. F., van Overveld, P. G. M., Sandkuijl, L. A., Vrieling, H., Padberg, G. W., Frants, R. R. & van der Maarel, S. M. (2004) Mechanism and timing of mitotic rearrangements in the subtelomeric D4Z4 repeat involved in facioscapulohumeral muscular dystrophy. *American Journal of Human Genetics*, 75, 44-53.

Leroy, E., Boyer, R., Auburger, G., Leube, B., Ulm, G., Mezey, E., Harta, G., Brownstein, M. J., Jonnalagada, S., Chernova, T., Dehehia, A., Lavedan, C., Gasser, T., Steinbach, P. J., Wilkinson, K. D. & Polymeropoulos, M. H. (1998) The ubiquitin pathway in Parkinson's disease. *Nature*, 395, 451-452.

Lyle, R., Wright, T. J., Clark, L. N. & Hewitt, J. E. (1995) The FSHD-associated repeat, D4Z4, is a member of a dispersed family of homeobox-containing repeats, subsets of which are clustered on the short arms of the acrocentric chromosomes. *Genomics*, 28, 389-397.

Lu, Q., Wallrath, L., Granok, H. & Elgin, S. (1993) (CT)n (GA)n repeats and heat shock elements have distinct roles in chromatin structure and transcriptional activation of the Drosophila hsp26 gene. *Molecular and Cellular Biology*, 13, 2802-2814.

Lunt, P. W. & Harper, P. S. (1991) Genetic counselling in facioscapulohumeral muscular dystrophy. *Journal of Medical Genetics*, 28, 655-664.

Lunt, P. W., Jardine, P. E., Koch, M. C., Maynard, J., Osborn, M., Williams, M., Harper, P. S. & Upadhyaya, M. (1995) Correlation between fragment size at D4F104S1 and age at onset or at wheelchair use, with a possible generational effect, accounts for much phenotypic variation in 4q35-facioscapulohumeral muscular dystrophy (FSHD). *Human Molecular Genetics*, 4, 951-958.

McNaught, K. S., Olanow, C. W., Halliwell, B., Isacson, O. & Jenner, P. (2001) Failure of the ubiquitin-proteasome system in Parkinson's disease. *Nature Reviews Neuroscience*, 2, 589-594.

Meneveri, R., Agresti, A., Marozzi, A., Saccone, S., Rocchi, M., Archidiacono, N., Corneo, G., Valle, G. D. & Ginelli, E. (1993) Molecular organisation and chromosomal location of a human GC-rich heterochromatic blocks. *Gene*, 123, 227-234.

Milosavljevic, A. & Jurka, J. (1993) Discovering simple DNA sequences by the algorithmic significance method. *Computer Applications in Biosciences*, 9, 407-411.

Müller, U., Tantravashi, U., Monaco, A., Stroh, H., Kunkel, L. M. & Latt, S. A. (1986) Repeated DNA sequences in the distal long arm of the human X chromosome. *Human Genetics*, 74, 24-29.

Nag, D. K. (2003) Trinucleotide repeat expansions: timing is everything. *Trends in Molecular Medicine*, 9, 455-457.

Nowak, R. (1994) Mining treasures from 'junk DNA.' *Science*, 263, 608-610.

Näslund, K., Saetre, P., von Salomé, J., Bergström, T. F., Jareborg, N. & Jazin, E. (2005) Genome-wide prediction of human VNTRs. *Genomics*, 85, 24-35.

Okada, T., Gondo, Y., Goto, J., Kanazawa, I., Hadano, S. & Ikeda, J. (2002) Unstable transmission of the RS447 human megasatellite tandem repetitive sequence that contains the USP17 deubiquitinating enzyme gene. *Human Genetics*, 110, 302-313.

Pardue, M., Lowenhaupt, K., Rich, A. & Nordheim, A. (1987) (dC-dA)n.(dG-dT)n sequences have evolutionarily conserved chromosomal locations in Drosophila with implications for roles in chromosome structure and function. *The European Molecular Biology Organization Journal*, 6, 1781-1789.

Rice, P., Longden, I. & Bleasby, A. (2000) EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics*, 16, 276-277.

Richards, R. I., Holman, K., Yu, S. & Sutherland, G. R. (1993) Fragile X syndrome unstable element, p(CCG)n, and other simple tandem repeat sequences are binding sites for specific nuclear proteins. *Human Molecular Genetics*, 2, 1429-1435.

Rivals, E., Delgrange, O., Delahaye, J. P., Dauchet, M., Delorme, M. O., Hénaut, A. & Ollivier, E. (1997) Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences. *Computer Applications in Biosciences*, 13, 131-136.

Sagot, M. F. & Myers, E. W. (1998) In: S. Istrail, P. Pevzner & M. Waterman (eds.), Identifying satellites in nucleic acid sequences (pp. 234-242). *Proceedings of the second annual international conference on computational molecular biology*, March 22-25, 1998, New York, United States.

Saitoh, Y., Saitoh, H., Ohtomo, K. & Mizuno, S. (1991) Occupancy of the majority of DNA in the chicken W chromosome by bent-repetitive sequences. *Chromosoma*, 101, 32-40.

Saitoh, Y., Miyamoto, N., Okada, T., Gondo, Y., Showguchi-Miyata, J., Hadano, S. & Ikeda, J. (2000) The RS447 human megasatellite tandem repetitive sequence encodes a novel deubiquitinating enzyme with a functional promoter. *Genomics*, 67, 291-300.

Sawicki, M. P., Samara, G., Hurwitz, M. & Passaro, E. Jr. (1993) Human genome project. The *American Journal of Surgery*, 165, 258-264.

Schmidt, J. P. (1998) All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *Society for Industrial and Applied Mathematics Journal on Computing*, 27, 972-992.

Tautz, D. (1993) Notes on the definition and nomenclature of tandemly repetitive DNA sequences. In: S. D. J. Pena, R. Chakraborty, J. T. Epplen & A. J. Jeffreys (eds.), *DNA fingerprinting: State of science* (pp. 21-28). Basel: Birkhäuser Verlag.

Tautz, D. & Schlötterer, C. (1994) Simple sequences. *Current Opinion in Genetics & Development*, 4, 832-837.

Tawil, R., Forrester, J., Griggs, R. C., Mendell, J., Kissel, J., McDermott, M., King, W., Weiffenbach, B. & Figlewicz, D. (1996) Evidence for anticipation and association of deletion size with severity in facioscapulohumeral muscular dystrophy. *Annals of Neurology*, 39, 744-748.

Tyler-Smith, C. & Taylor, L. (1988) Structure of a hypervariable tandemly reapeated DNA sequence on the short arm of the human Y chromosome. *Journal of Molecular Biology*, 203, 837-848.

van Deutekom, J. C., Wijmenga, C., van Tienhoven, E. A., Gruter, A. M., Hewitt, J. E., Padberg, G. W., van Ommen, G. J., Hofker, M. H. & Frants, R. R. (1993) FSHD associated DNA rearrangements are due to deletions of integral copies of a 3.2 kb tandemly repeated unit. *Human Molecular Genetics*, 2, 2037-2042.

van Geel, M., Heather, L. J., Lyle, R., Hewitt, J. E., Frants, R. R. & de Jong, P. J. (1999) The FSHD region on human chromosome 4q35 contains potential coding regions among pseudogenes and a high density of repeat elements. *Genomics*, 61, 55-65.

Vergnaud, G. & Denoeud, F. (2000) Minisatellites: mutability and genome architecture. *Genome Research*, 10, 899-907.

Weber, J. L. & May, P. E. (1989) Abundant class of human DNA polymorphisms which can be typed using the polymerase chain reaction. *American Journal of Human Genetics*, 44, 388-396.

Wijmenga, C., Hewitt, J. E., Sandkuijl, L. A., Clark, L. N., Wright, T. J., Dauwerse, H. G., Gruter, A. M., Hofker, M. H., Moerer, P., Williamson, R., van Ommen, G. J., Padberg, G. W. & Frants, R. R. (1992) Chromosome 4q DNA rearrangements associated with facioscapulohumeral muscular dystrophy. *Nature Genetics*, 2, 26-30.

Winokur, S. T., Bengtsson, U., Fedderson, J., Mathews, K. D., Weiffenbach, B., Bailey, H., Markovich, R. P., Murray, J. C., Wasmuth, J. J., Altherr, M. R. & Schutte, B. C. (1994) The DNA rearrangement associated with facioscapulohumeral muscular dystrophy involves a heterochromatin-associated repetitive element: Implications for a role of chromatin structure in the pathogenesis of the disease. *Chromosome Research*, 2, 225-234.

Yee, H. A., Wong, A. K., van den Sande, J. H. & Rattner, J. B. (1991) Identification of novel single-stranded d(TC)n binding proteins in several mammalian species. *Nucleic Acids Research*, 19, 949-953.

Zhang, X. Y., Loflin, P. T., Gehrke, C. W., Andrews, P. A. & Ehrlich, M. (1987) Hypermethylation of human DNA sequences in embryonic carcinoma cells and somatic tissues but not in sperm. *Nucleic Acids Research*, 15, 9429-9449.

Appendix A

A detailed description of the test scenarios

Test scenarios

1. Run the algorithm on a randomly generated sequence, believed to contain no LSTRs.

- *Objective*: Find the "baseline", i.e. how frequently the algorithm indicates a signal in a sequence where no signal should occur. This provides hints towards the algorithm's accuracy.
- *Parameters*: Hexamer probes, length variation tolerance (l.v.t.) of segments between hits ranging from 0%-10%; the steps being 0%, 0.5%, 1%, 2%, 5%, and 10%, and finally probe sets of size 100, 200, and 300 respectively. All other parameters are kept at their default settings.
- *Motivation*: Knowing the result provides the user with a "baseline" cluster size, i.e. the maximum size of clusters that can be ignored. Example: Running with 300 hexamers over the entire human genome, do clusters of size 3 with repeat-unit size 5000 indicate a real signal?

Runs:

1. Generate the set of all possible hexamer probes. This procedure is included as one of the functions in the script megasat_finder_nomegs.py.
2. Generate a 1 Gb (gigabase) random sequence, created by applying a 25% probability of each of the four bases, using megasat_random_generator.py adjusted accordingly (1Gb_random.seq).
3. The script megasat_finder_nomegs.py is used for running the algorithm, and then clusters the generated results, providing easily interpreted results. The script calls are organised and controlled by the wrapper wrapper_nomegs.py, where the parameter settings can be adjusted.
4. Three sets, selected randomly from the set of all possible hexamers, are created containing 100, 200, and 300 probes respectively (the probe set size is set by the user).
5. For each of the three probe sets, run the algorithm using a length variation tolerance of 0%, 0.5%, 1% (default), 2%, 5%, and 10%, thereby creating a total of 18 different runs.

Analysis:

1. Join all the resulting files for each of the runs using megaresults_clusterfinder.py.
2. Create a table which displays the number of clusters found for each cluster size. The number is shown for each of the three probe sets, using the various length variation tolerance (l.v.t.) percentages.

2. Same as in test scenario 1, on a real DNA sequence extracted from the human genome, believed to contain no LSTRs. The retrieved results from test scenarios 1 and 2 are compared.

- *Objective*: Investigate whether a randomly generated sequence is sufficient as a model for the simulation in test scenario 1, i.e. whether random sequences resemble DNA sequences enough to be used for *in silico* research purposes.
- *Parameters*: Hexamer probes and length variation tolerance (l.v.t.) percentage of 1% and 2% respectively.
- *Motivation*: Validates the use of an *in silico* generated sequence when doing this kind of analysis. A positive outcome would be preferable, since *in silico* generated sequences are easier to obtain and control.

Runs:

1. Extract a 10 Mb nucleotide sequence from the human genome (Build 34), which is believed to contain no megasatellites (found by plotting the sequence against itself).
2. Generate the set of all possible hexamer probes. This procedure is included as one of the functions in the script megasat_finder_nomegs2.py.
3. The script megasat_finder_nomegs2.py is used for running the algorithm, and then clusters the generated results, providing easily interpreted results. The script calls are organised and controlled by the wrapper wrapper_nomegs2.py, where the parameter settings can be adjusted.
4. Three sets, selected randomly from the set of all possible hexamers, are created containing 100, 200, and 300 probes respectively (the probe set size is set by the user).
5. For each of the three probe sets, run the algorithm using a length variation tolerance of 1% (default) and 2%, thereby creating a total of 6 different runs.

Analysis:

1. Join all the resulting files for each of the runs using megaresults_clusterfinder.py.
2. Create a table which displays the number of clusters found for each cluster size. The number is shown for each of the three probe sets, using the two length variation tolerance (l.v.t.) percentages.
3. Compare the resulting tables from test scenario 1 and 2 with regard to l.v.t. 1% and 2%. Be aware that this test is performed for a 10 Mb sequence, while test scenario 1 is performed for a 1 Gb sequence.

3. Run the algorithm on randomly generated tandem repeats of fixed repeat-unit size and repeat count, using both a real DNA sequence extracted from the human genome and a random nucleotide sequence.

- *Objective*: Find the "curve" for each probe size, i.e. find the repeat-unit size for each probe size at which the algorithm expresses maximum sensitivity. In addition, find when the sensitivity reaches a baseline, or alternatively when the signal fades out. This test scenario provides guidelines regarding the appropriate probe sizes to use when searching for megasatellites with a certain repeat-unit size.
- *Parameters*: All four probe sizes (pentamers, hexamers, heptamers, and octamers), a set of 300 probes used for all runs, repeat-unit sizes varying from 100 and upwards, based on probe size and whether a real or a randomly generated nucleotide sequence is used.
- *Motivation*: Helps the user interpret the signal "Running with 300 hexamers, found a cluster of size 20 with repeat-unit size 15000".

Runs:

1. Run the algorithm for pentamers, using a real DNA sequence.
   Use the real DNA sequence extracted from positions 150,000,000-160,000,000 on chromosome 1 (Build 34). This sequence is 10 Mb in length.
   Parameter settings:
   Probe size (5), probe number (300), repeat-unit size (100-130,265) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-130,265: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit size:
   100, in order to get a good mean or median value such a high number of runs is to be preferred.

2. Run the algorithm for pentamers, using a random sequence.
   Use the generated random sequence, applying the 40/60 base rule. The sequence is 10 Mb in length and generated by using the script megasat_random_generator.py with the appropriate adjustments.
   Parameter settings:
   Probe size (5), probe number (300), repeat-unit size (100-30,140) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-30,000: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit length:
   100, in order to get a good mean or median value such a high number of runs is to be preferred.

3. Run the algorithm for hexamers, using a real DNA sequence.
   Use the real DNA sequence extracted from positions 150,000,000-160,000,000 on chromosome 1 (Build 34). The sequence is 10 Mb in length.
   Parameter settings:
   Probe size (6), probe number (300), repeat-unit size (100-1,011,061) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-1,011,061: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit length:
   100, in order to get a good mean or median value such a high number of runs is to be preferred.

4. Run the algorithm for hexamers, using a random sequence.
   Use the generated random sequence, applying the 40/60 base rule. The sequence is 10 Mb in length and generated by using the script megasat_random_generator.py with the appropriate adjustments.
   Parameter settings:
   Probe size (6), probe number (300), repeat-unit size (100-130,265) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-130,265: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit length:
   100, in order to get a good mean or median value such a high number of runs is to be preferred.

5. Run the algorithm for heptamers, using a real DNA sequence.
   Use the real DNA sequence extracted from positions 150,000,000-160,000,000 on chromosome 1 (Build 34). The sequence is 10 Mb in length.
   Parameter settings:
   Probe size (7), probe number (300), repeat-unit size (100-3,105,500) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-3,105,500: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit length:
   100, in order to get a good mean or median value such a high number of runs is to be preferred.

6. Run the algorithm for heptamers, using a random sequence.
   Use the generated random sequence, applying the 40/60 base rule. The sequence is 10 Mb in length and generated by using the script megasat_random_generator.py with the appropriate adjustments.
   Parameter settings:
   Probe size (7), probe number (300), repeat-unit size (100-510,655) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-510,655: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit length:

100, in order to get a good mean or median value such a high number of runs is to be preferred.

7. Run the algorithm for octamers, using a real DNA sequence.
   Use the real DNA sequence extracted from positions 150,000,000-160,000,000 on chromosome 1 (Build 34). The sequence is 10 Mb in length.
   Parameter settings:
   Probe size (8), probe number (300), repeat-unit size (100-8,239,814) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-8,239,814: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit length:
   100, in order to get a good mean or median value such a high number of runs is to be preferred.

8. Run the algorithm for octamers, using a random sequence.
   Use the generated random sequence, applying the 40/60 base rule. The sequence is 10 Mb in length and generated by using the script megasat_random_generator.py with the appropriate adjustments.
   Parameter settings:
   Probe size (8), probe number (300), repeat-unit size (100-1,568,488) and number of repeat-units in the megasatellite (4).
   Increase in repeat-unit length:
   100-1,568,488: increase by 5% between subsequent steps.
   Number of iterations for each repeat-unit length:
   100, in order to get a good mean or median value such a high number of runs is to be preferred.

Analysis:

1. Run the eight tasks above, using the script megasat_creator_finder.py.
2. The script calls are organised and controlled by wrapper.py, where the parameter settings can be adjusted.
3. Join the result files belonging to each of the eight tasks and use the script megaresults_meanfinder.py to generate the mean values and standard deviation. If required, the user can calculate the median instead of the mean values.
4. Generate graphs for each task, using the graphics visualisation tool wgnuplot, plotting each of the mean files. This shows the curve for each probe size, using a real DNA and a random sequence respectively. Two types of graphs should be generated; one showing the maximum sensitivity clearly and the other showing the whole graph until the signal reaches zero.
5. Create a table showing the expressed maximum sensitivity for each probe size, a table showing the mean success rates for a selection of repeat-unit sizes using real DNA sequences, and a table showing the mean success rates for a selection of repeat-unit sizes using random sequences.
6. Compare the results generated using real versus randomly generated sequences.

4. Same as in test scenario 3, after mutating sequence.

- *Objective*: Reveal in what way the "curves", from previous simulations in test scenario 3, change with "evolution time", i.e. with the mutation level. This refers to both the maximum sensitivity and the baseline sensitivity. In addition, using a fixed mutation level, find how the length variation tolerance affects the curves, testing it for a fixed repeat-unit size. This provides a measure of how sensitive the algorithm is with respect to changes in the sequences.
- *Parameters*: Sequence similarity ranging from 100%-80%; the steps being 99%, 98%, 95%, 90%, and 80%. Additionally, hexamer probes, probe count 300, and range of repeat-unit size similarities from 0%-10%; the steps being 0.5%, 2%, and 5% (i.e. the length variation tolerance, l.v.t.).
- *Motivation*: Important to realise how base pair dissimilarity between repeat-units affects the performance of the algorithm.

Runs:

1. Run the algorithm, using the script megasat_creator_finder_mutation.py, based on the mutation percentage required by the user. The script calls are organised and controlled by wrapper_6_mutations.py, where all parameter settings can be found and adjusted. The range of repeat-unit sizes is clearly stated in the script.
2. The difference between this test scenario and test scenario 3 is firstly that mutations are performed and secondly that hexamers probes are the only ones used for testing. The relationships for other probe sizes will be deducted from the results for hexamers. Both real DNA and random sequences will be applied.
3. The signal for the mutation runs should become very low or stop completely.
4. When a mutation level has been chosen, the length variation tolerance (l.v.t.) should be varied according to the above percentages on one or two mutation levels which are selected as appropriate. A mutation level(s) which clearly shows changes in success rates should be chosen as a representative. The script calls are organised and controlled by wrapper_6_mutations_lvt.py, where the parameter settings can be adjusted.

Analysis:

1. Join the result files belonging to each of the tasks and use the script megaresults_mut_meanfinder.py to generate the mean values and standard deviation. If required, the user can calculate the median instead of the mean values.
2. Generate graphs for each task, using the graphics visualisation tool wgnuplot, plotting each of the mean files. This shows the curve for each probe size, using a real DNA and a random sequence respectively. Two types of graphs should be generated; one showing the maximum sensitivity clearly and the other showing the whole graph until the signal reaches zero.
3. Create a table showing the expressed maximum sensitivity for the hexamers, a table showing the mean success rates for a selection of repeat-unit sizes using

real DNA sequences, and a table showing the mean success rates for a selection of repeat-unit sizes using random sequences.

4. Compare the results generated using real versus randomly generated sequences and deduct the behaviour regarding mutation using the other probe sizes in relation to using hexamer probes.
5. Generate tables showing the results when the l.v.t. percentages are altered.

5. Run the algorithm on the entire human genome, containing confirmed megasatellites.

- *Objective*: Explore in a real-life situation which parameter settings maximise the performance. Compare with previous simulations. This gives an example of usage on real data and which results are retrieved as an output from the algorithm. It is, in addition, also important to verify whether the algorithm predicts megasatellites which are already known and confirmed.
- *Parameters*: All four probe sizes, probe count 300. The rest is kept at default settings.
- *Motivation*: Knowing how the algorithm works on a real DNA sequence containing megasatellites increases knowledge of its function and behaviour.

Runs:

1. Perform a complete genomic scan of the human genome (Build 34).
2. Cluster the results, thereby combining the results for each of the four probe sizes.

Analysis:

1. Illustrate a piece of the results from the algorithm in a figure to explain how they are presented to the user.
2. Show sequence comparisons, megasatellite regions, and self-sequence comparisons of a detected megasatellite, selected as an example.
3. Discuss whether the algorithm located the already known megasatellites and how many were detected in total.

6. Complexity analysis.

- *Objective*: Derive time complexity, by measuring run-time. This knowledge is essential with regard to both usability and applicability.
- *Parameters*: All four probe sizes, different probe counts.
- *Motivation*: It is important to know the computational requirements, i.e. the amount of time it takes to perform a run.

Runs:

1. All runs are conducted from the command line.
2. Perform controlled runs on using the 10 Mb real nucleotide sequence extracted from the human genome.
    a. Use 150 and 300 probes respectively.
    b. Use 50 versus 100 iterations for each of the probe set sizes.
    c. Perform this for all four probe sizes, using a single repeat-unit size, which expresses the maximum sensitivity in test scenario 3.
    d. The rest of the parameters are kept at default settings.
3. Perform controlled runs on using the 10 Mb real nucleotide sequence extracted from the human genome. The artificial megasatellites have been mutated by 1%, 2%, 5%, 10%, and 20%.
    a. Use 300 probes.
    b. Use 100 iterations.
    c. Perform this for hexamers, using a single repeat-unit size, which expresses the maximum sensitivity in test scenario 3.
    d. The rest of the parameters are kept at default settings.
4. Perform runs on the whole genome (Build 34).
    a. Use all four probe sizes.
    b. Use 300 probes.
    c. The rest of the parameters are kept at default settings.
5. Specifically document the computer specifications for these runs, since they affect the time complexity measurements.

Analysis:

1. Make tables for the complexity results from runs number 2, number 3, and number 4 in the above list. Specifically indicate all parameter settings and the exact running time.

Appendixes B-N

The content of these appendixes is not included in this version of the thesis. All scripts are available upon request from the author.

Contact: a01elabe@student.his.se

Appendix O

Figures and tables from Chapter 6

Table O.1 The full version of Table 6.1. The number of clusters of a specific cluster size, found for various length variation tolerance thresholds, using a set of 100, 200, and 300 randomly chosen hexamer probes from the set of all possible hexamer probes (4^6 = 4096). The algorithm was run on a 1 Gb sequence, generated by applying a 25% probability per base.

Probe count	Cluster size	0%	0.5%	1%	2%	5%
100	2	-	101	500	3516	35,805
100	3	-	-	3	124	6297
100	4	-	-	-	4	1471
100	5	-	-	-	-	431
100	6	-	-	-	1	160
100	7	-	-	-	-	64
100	8	-	-	-	-	30
100	9	-	-	-	-	14
100	10	-	-	-	-	3
100	11	-	-	-	-	3
100	12	-	-	-	-	5
100	14	-	-	-	-	1
100	16	-	-	-	-	16
200	2	-	366	2029	13,330	44,873
200	3	-	1	39	779	11,908
200	4	-	-	-	83	4345
200	5	-	-	-	10	1994
200	6	-	-	-	1	1003
200	7	-	-	-	1	545
200	8	-	-	-	-	341
200	9	-	-	-	-	231
200	10	-	-	-	-	159
200	11	-	-	-	-	103
200	12	-	-	-	-	84
200	13	-	-	-	-	58
200	14	-	-	-	-	42
200	15	-	-	-	-	42
200	16	-	-	-	-	38
200	17	-	-	-	-	19
200	18	-	-	-	-	15
200	19	-	-	-	-	11
200	20	-	-	-	-	15
200	21	-	-	-	-	13
200	22	-	-	-	-	10
200	23	-	-	-	-	3
200	24	-	-	-	-	6
200	25	-	-	-	-	4
200	26	-	-	-	-	6
200	27	-	-	-	-	6

200	28	-	-	-	-	7
200	29	-	-	-	-	5
200	30	-	-	-	-	2
200	31	-	-	-	-	3
200	32	-	-	-	-	4
200	33	-	-	-	-	1
200	34	-	-	-	-	2
200	35	-	-	-	-	2
200	36	-	-	-	-	2
200	37	-	-	-	-	2
200	38	-	-	-	-	1
200	39	-	-	-	-	4
200	41	-	-	-	-	1
200	42	-	-	-	-	3
200	43	-	-	-	-	1
200	44	-	-	-	-	1
200	47	-	-	-	-	2
200	48	-	-	-	-	1
200	50	-	-	-	-	1
200	52	-	-	-	-	3
200	59	-	-	-	-	1
200	64	-	-	-	-	1
200	79	-	-	-	-	1
200	100	-	-	-	-	1
200	169	-	-	-	-	1
200	183	-	-	-	-	1
200	695	-	-	-	-	1
200	2100	-	-	-	-	1
200	2631	-	-	-	-	1
200	7638	-	-	-	-	1
200	8844	-	-	-	-	1
200	10,783	-	-	-	-	1
200	11,780	-	-	-	-	1
200	12,381	-	-	-	-	1
200	14,428	-	-	-	-	1
200	14,832	-	-	-	-	1
200	20,035	-	-	-	-	1
200	20,169	-	-	-	-	1
200	20,198	-	-	-	-	1
200	22,251	-	-	-	-	1
200	22,584	-	-	-	-	1
200	24,349	-	-	-	-	1
200	25,033	-	-	-	-	1
200	28,753	-	-	-	-	1
200	29,979	-	-	-	-	1
200	31,451	-	-	-	-	1
200	31,694	-	-	-	-	1

200	32,239	-	-	-	-	1
200	32,294	-	-	-	-	1
200	37,582	-	-	-	-	1
200	39,621	-	-	-	-	1
200	42,661	-	-	-	-	1
200	62,912	-	-	-	-	1
300	2	-	853	4535	28,059	-
300	3	-	7	124	2610	-
300	4	-	-	4	393	-
300	5	-	-	-	63	-
300	6	-	-	1	7	-

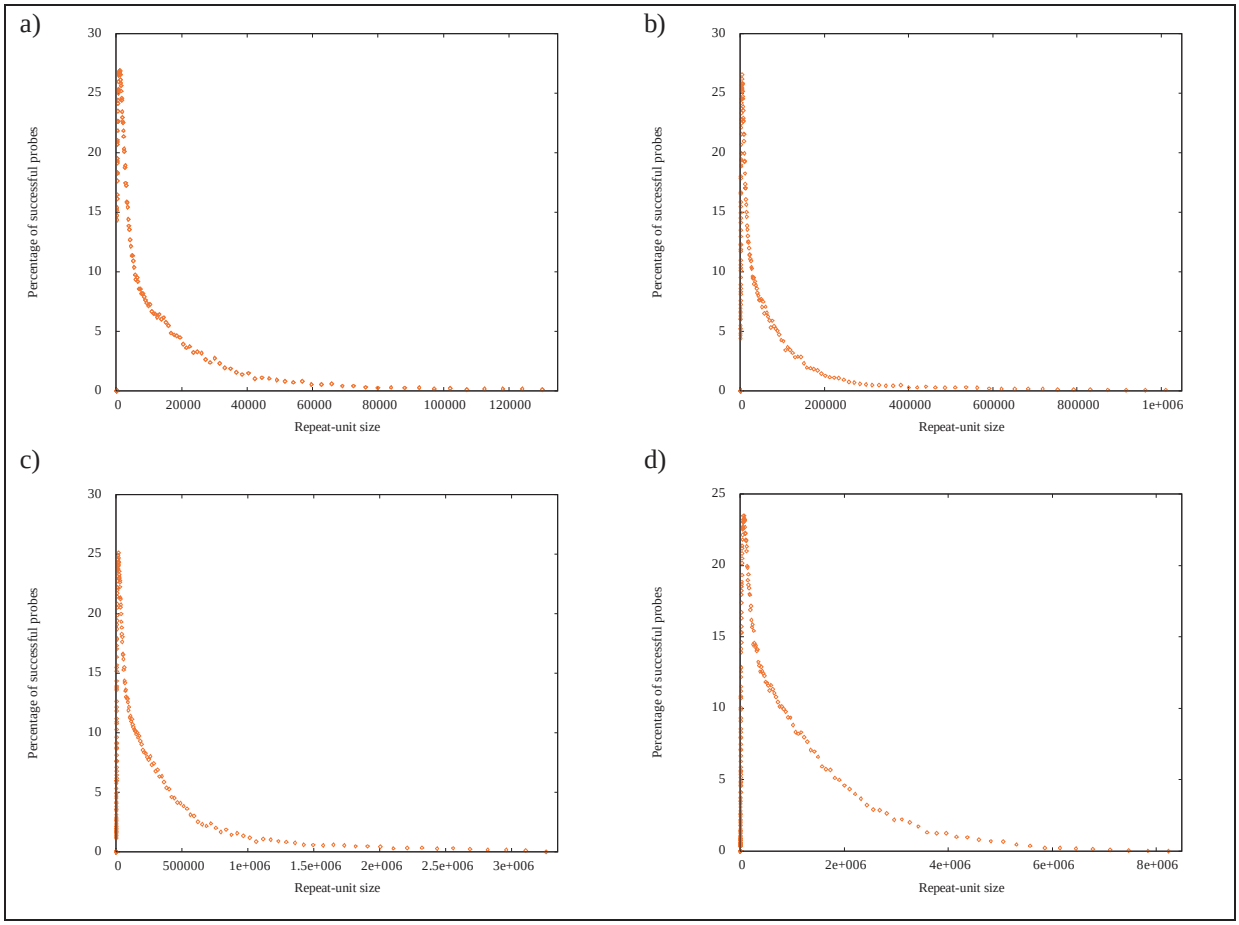


Figure O.1 These figures show the results generated by performing the same type of tests as performed when generating the figures in Figure 6.1, except this time the focus is placed on the end of the curve. Part a) using pentamers, b) using hexamers, c) using heptamers, and d) using octamers. The figure shows how the signal fades out until it reaches zero for extremely large repeat-unit sizes, i.e. results in no signal at all.

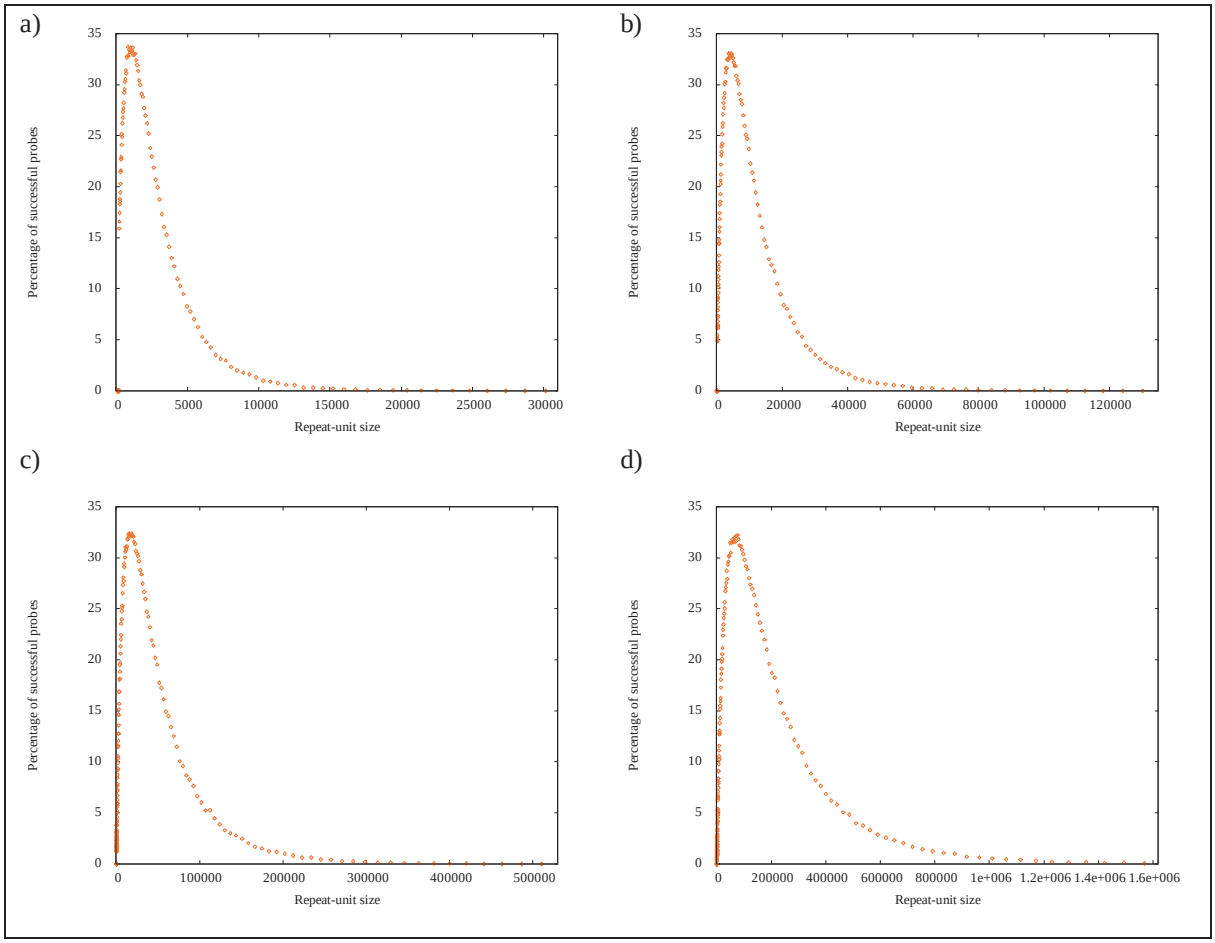
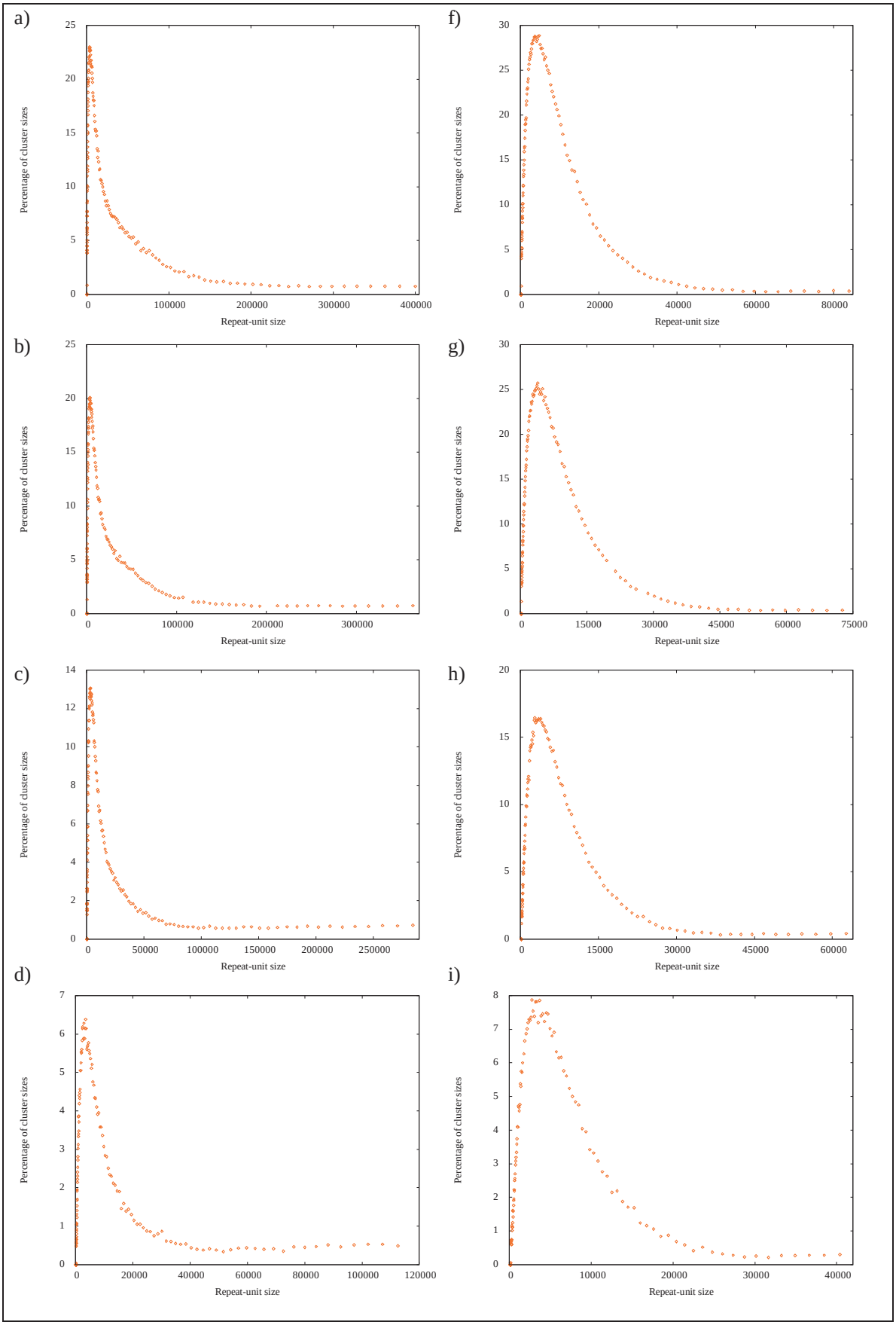


Figure O.2 These figures show the results generated by performing the same type of tests as performed when generating the figures in Figure 6.2, except this time the focus is placed on the end of the curve. Part a) using pentamers, b) using hexamers, c) using heptamers, and d) using octamers. The figure shows how the signal fades out until it reaches zero for extremely large repeat-unit sizes, i.e. results in no signal at all.


a)
Percentage of cluster sizes
Repeat-unit size
0
5
10
15
20
25
0
100000
200000
300000
400000
b)
Percentage of cluster sizes
Repeat-unit size
0
5
10
15
20
25
0
100000
200000
300000
c)
Percentage of cluster sizes
Repeat-unit size
0
2
4
6
8
10
12
14
0
50000
100000
150000
200000
250000
d)
Percentage of cluster sizes
Repeat-unit size
0
1
2
3
4
5
6
7
0
20000
40000
60000
80000
100000
120000
f)
Percentage of cluster sizes
Repeat-unit size
0
5
10
15
20
25
30
0
20000
40000
60000
80000
g)
Percentage of cluster sizes
Repeat-unit size
0
5
10
15
20
25
30
0
15000
30000
45000
60000
75000
h)
Percentage of cluster sizes
Repeat-unit size
0
5
10
15
20
0
15000
30000
45000
60000
i)
Percentage of cluster sizes
Repeat-unit size
0
1
2
3
4
5
6
7
8
0
10000
20000
30000
40000

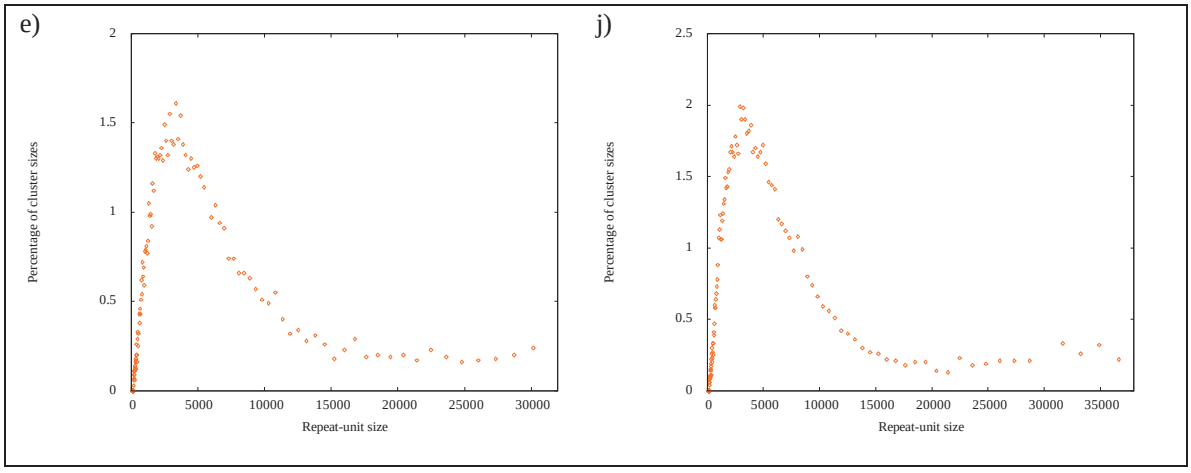


Figure O.3 The slope and baseline sensitivity for each of the mutation levels using hexamers, when running the algorithm on a real DNA sequence extracted from the human genome (parts a) to e)), and on a random sequence generated by applying the 40/60 rule (parts f) to j)). Mutation levels: 1%, 2%, 5%, 10%, and 20% (top-down order). The repeat-unit size is indicated on the x-axis and the percentage of cluster sizes is indicated on the y-axis. A cluster size is defined as the size of the resulting cluster, i.e. the hit, and provides the same results as the successful probe percentages in subchapter 6.3.