MICROSATELLITE DETECTION AND CONSENSUS SEQUENCE VERIFICATION

BY VIRTUAL PCR AND MACHINE LEARNING

by

DMITRI KOLYCHEV

(Under the Direction of KHALED RASHEED)

ABSTRACT

Microsatellites, or simple sequence repeats, are genetic loci where several nucleotide bases are repeated in tandem. Since they can be easily found by Polymerase Chain Reaction (PCR) using unique flanking primers, they are considered excellent genetic markers in making genetic linkage maps among other things. In this thesis we present Microsatellite Polymorphism Finder (MSPF), a program that detects and then verifies microsatellites by modeling PCR digitally from an Expressed Sequence Tag (EST) or a shotgun-sequencing database without the overhead required to perform PCR chemically with human intervention. Moreover, a machine learning enhanced version of MSPF improves the accuracy of microsatellite verification.

INDEX WORDS: microsatellite detection, neural networks, polymerase chain reaction, consensus sequence verification

MICROSATELLITE DETECTION AND CONSENSUS SEQUENCE VERIFICATION BY VIRTUAL PCR AND MACHINE LEARNING

by

DMITRI KOLYCHEV

A Thesis Submitted to the Graduate Faculty of the University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2003

MICROSATELLITE DETECTION AND CONSENSUS SEQUENCE VERIFICATION BY VIRTUAL PCR AND MACHINE LEARNING

by

DMITRI KOLYCHEV

Major Professor:	Khaled Rasheed
Committee:	M.M. Cordonnier-Pratt Walter D. Potter

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2003

DEDICATION

To all I ever cared about…

ACKNOWLEDGEMENTS

Thanks to UGA faculty for providing me with valuable experience during the last four years: Lee Pratt, Marie-Michele Cordonnier-Pratt, and Khaled Rasheed for giving me insight into the biological domain, providing me with access to data and leading me in this research area, Walter D. Potter for support and suggestions. Special thanks to Manish Shah, Chun Liang, and Robert Sullivan for implementation suggestions.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

This chapter seeks to introduce biological concepts behind microsatellite detection and verification in DNA sequences, algorithms currently used by major sequencing laboratories, and the machine learning method that has been used to enhance a new approach of microsatellite detection/verification in Microsatellite Polymorphism Finder (MSPF). Currently, these important genetic markers are detected in the consensus sequences produced by flawed fragment assembly methods and verified chemically through human intervention. The approach described in this thesis detects microsatellites by using redundant individual fragment information and verifies them by modeling the chemical process of the Polymerase Chain Reaction (PCR) in-silico, producing fast and accurate results. Section 1.1 provides a glossary of terms. Section 1.2 explains the biological background and significance of microsatellites. Section 1.3 explains common computational methods used in DNA sequencing. Section 1.4 describes algorithms used for generic microsatellite detection. Finally, section 1.5 describes specific machine learning method used.

1.1 GLOSSARY

-Chromatogram: a plot of a detector's response versus time; in genome sequencing, the relative emission peaks of the four fluorescently labeled nucleotides (ATGC) as a function of time (e.g., as sequencing proceeds).

-Contig: consensus sequence or a continuous region of genomic DNA that has been cloned as a series of identifiable overlapping DNA clones.

-Microsatellites: simple sequence repeats (SSRs) or genetic loci where several nucleotide bases or motifs are repeated in tandem (i.e., agagagagagagag).

-Oligonucleotide: short segment of DNA.

-Polyacrylamide gel electrophoresis: technique used to separate macromolecules carried out in a polyacrylamide gel; movement of molecules is slowed according to the dimensions relative to the size of pores in the matrix and the size of the macromolecules.

-Polymerase Chain Reaction: *in vitro* method of exponential DNA amplification, involving the use of specific oligonucleotides to prime DNA synthesis from a specified target DNA sequence and nucleotides; amplifies specific lengths of DNA located between two annealed primers by repeated 'thermal cycling' reactions using a thermostable polymerase enzyme. This mixture is thermally cycled through three steps; denaturation, obtained by melting of the target DNA double strand, annealing of the primers to the target sequence, and extension of the primers by the DNA polymerase using the nucleotides to form a second strand.

-Polymorphic: differing between distinct alleles (e.g., a microsatellite that has different lengths in different alleles; for example, *agagag* in one allele and *agag* in another).

-Shotgun sequencing: sequencing method that involves assembling randomly sequenced cloned pieces of the genome into a consensus sequence, with no foreknowledge of the location from which the piece originated.

-Slipped-strand mispairing: process in which the single strands of a double helix pair out of register, often at short tandem repeated sequences (Figure 1). The resulting

daughter DNA strands will have corresponding deletions or insertions.

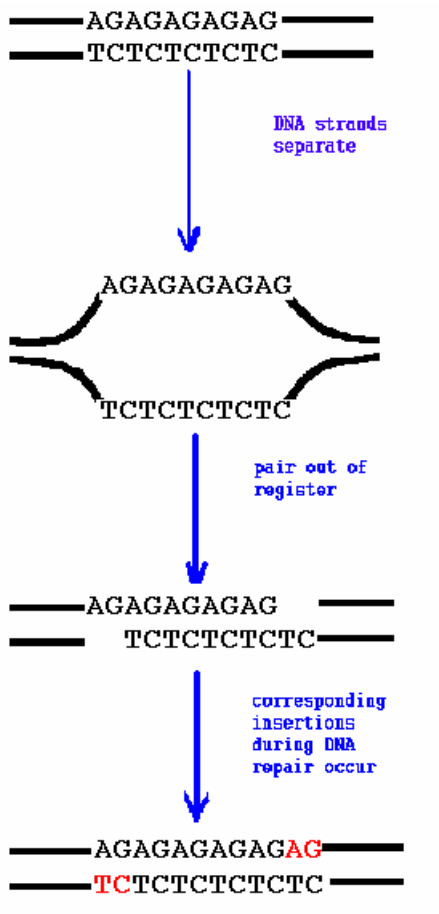


Figure 1: Slipped-strand mispairing

-Thermostable DNA polymerase: enzyme that allows the synthesis at high temperature of double-stranded DNA by extension of a primer annealed to single-stranded DNA.

1.2 BIOLOGICAL BACKGROUND

A key practice in current genomic research is large scale genotyping. The availability of dense, highly polymorphic, and uniformly distributed genetic markers is crucial for mapping and subsequent genotyping, which are used to determine which

genes are responsible for specific inherited traits. Microsatellites or simple sequence repeats are genetic loci where a few nucleotide bases are repeated in tandem. They can be found anywhere in the genome, both in protein-coding and noncoding regions. The occurrence of polymorphic microsatellites is mostly due to slipped-strand mispairing during DNA replication, repair, or recombination, which makes them exhibit relatively frequent length polymorphism (Figure 1). Point mutations may also accumulate in the microsatellite regions, producing corresponding substitutions, insertions, or deletions, and the mutation rates usually increase with an increase in the length of the microsatellite (Katti *et al.* 2001).

The discovery of diseases associated with microsatellites (e.g., Fragile-X retardation, Friedreich ataxia, myotonic dystrophy, Huntington's disease) and the potential role of microsatellites in gene regulation sparked an interest in tandem repeats among the scientific community. In addition, they have proven useful in the analysis of paternity and kinship (Queller *et al.* 1993) and in sample identitification at both the individual (Edwards et al. 1992) and population levels (Paetkau et al. 1995). Furthermore, microsatellites are more desirable than larger tandem repeat loci that have a motif length of six or longer because they can be analyzed more easily via the Polymerase Chain Reaction and the alleles can be sized more unambiguously on polyacrylamide gels.

Currently, PCR uses unique flanking primers or oligonucleotides designed from the flanking regions around the microsatellite in the consensus sequence. This is followed by electrophoresis for fragment-length sizing which forms the basis for microsatellite genotyping. Since PCR involves the exponential amplification of DNA from a template

using thermostable DNA polymerase, specific oligonucleotides are designed to direct amplification between two specific sites on the template around the microsatellite where each of those oligonucleotides bind. Since DNA has negatively charged phosphate groups that make up the DNA phosphate backbone, gel electrophoresis can separate DNA by fragment size: larger DNA pieces will progress more slowly through the gel matrix toward the positive cathode and vice versa. Thus, verification and microsatellite polymorphism detection is performed by looking at the size difference of the amplified fragments. This size difference is viewed as a migration of respective bands formed after gel electrophoresis. If the average amplified fragment size is different for some genotypes, that fragment is considered to be polymorphic. Edwards et al. (1991) examined the frequency of five microsatellite loci (tri and tetra repeats) on the X chromosome, finding that for either the tri or tetra microsatellite loci, any given repeat was found every 300 to 500 Kb. From this, they estimated that for all the 44 possible unique trimeric and tetrameric repeats there are 400,000 loci or about 1 every 10 to 20 Kb (Edwards et al. 1991). This frequency did not change after complete sequencing of other genomes occurred. Of the class of loci examined by Edwards et al. (1991), 50% were polymorphic.

In this thesis, we use a large number of Expressed Sequence Tags (ESTs) from sorghum (see www.fungen.org) to verify our approach. An EST database is created from single-pass sequences of complementary DNA (cDNAs), which in turn are derived from messenger RNA sequences (mRNAs). Each mRNA segment represents the expression of a gene, that is composed of several joined exons, and codes for a specific protein produced in the tissue sample. Hence ESTs are tags for those expressed genes. Every

unique mRNA segment is transformed to a cDNA sequence and inserted into a bacterium's plasmid with the help of a phage. Then bacterium is grown overnight to produce a colony of bacteria containing the same cDNA insert in each member. This cDNA insert has an average size of 1,500 bases which may or may not cover the complete cDNA length. Both ends of the insert are sequenced to produce two EST reads. Due to sequencing limitations (see next section) only about 600 bases can be sequenced during each dye terminator cycle sequencing reaction. If the ends of two sequencing reads overlap, a complete sequence of the cDNA is found, but sometimes, the reaction still runs short and leaves the space between the ends un-sequenced. Even though we used an EST database, our approach could be used in any shotgun sequencing (see next section) project which includes information from regions of DNA that do not code for proteins since the methods of sequencing remain similar.

1.3 COMPUTATIONAL OVERVIEW OF DNA SEQUENCING/ASSEMBLY

A very popular DNA sequencing strategy known as "shotgun sequencing," which is similar to EST sequencing, takes maximum advantage of the speed and low cost of automated sequencing. This strategy relies totally on software to assemble a jumble of essentially random sequence reads (which can contain errors) of length 100 to 1,000 nucleotide bases into coherent and accurate consensus sequences (contigs). In this type of sequencing, one or more identical double stranded DNA sequences are put through the automated dye terminator cycle sequencing reaction, in which the new DNA strand is built along the original template. The elongation stops with the incorporation of the fluorescently tagged terminating dideoxyribonucleotide (ddNTP) analogue inhibitor that

identifies the last base in the new fragment because each of the four different ddNTPs emits a unique wavelength profile following irradiation.

Using gel electrophoresis to separate each DNA fragment that differs by a single nucleotide will band each ddNTP analogue inhibitor and produce a sequencing read describing the intensity of each nucleotide base in a certain position in the read (Figure 2).

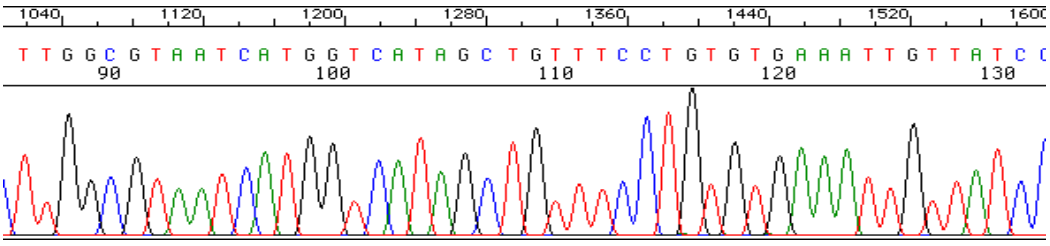


Figure 2: Sequencing read trace

These trace diagrams are analyzed by base calling programs. One of the most widely used programs is Phred, which calls bases along with their quality values using Fourrier transforms to match predicted to observed base intensity peak locations (Edwing et al. 1998). Phred's quality value for each called base corresponds to the probability of an error at that position. Phred's factors (distance between the peaks, peak heights, and areas under the peaks) have been empirically determined to predict the error probability fairly well, however, exact error probability is unknown.

In theory, optimal alignment of multiple sequences is possible by extension of pairwise algorithms, but since the problem can be represented as a Hamiltonian path, the

number of calculations needed equals the sequence length raised to the power of the number of sequences. Thus, fragment assembly programs are forced to use heuristic/approximation algorithms to construct a consensus sequence out of error-containing reads, which inevitably creates errors in contigs. In most fragment assembly programs the consensus sequence is constructed through the greedy traversal of the overlap graph (proven to form a sequence at most 2.75 times longer than the optimal sequence). A weighted, directed graph is formed where each directed edge (*u,v*) is weighted with the length of the maximal overlap between a suffix of fragment *u* and a prefix of fragment *v*. Representation of maximum overlap differs between the algorithms; some use distinct parameters found empirically to improve the accuracy of the assembly. The highest-weight path that touches every node of the graph represents the consensus sequence (path *abcd* represents the sequence of CTGCCATATA in Figure 3).

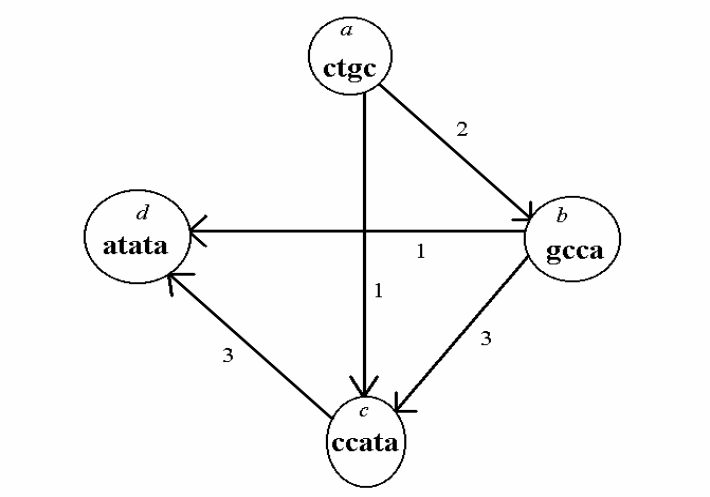


Figure 3: Fragment assembly as a Hamiltonian path problem

Moreover, these programs often misassemble contigs by either joining different repeat copies or by including many fragments from different repeat copies (Pevzner et al. 2001).

Phrap, the most widely used fragment assembly program, calculates overlap scores using pairwise sequence alignments and error probabilities. Individual LLR scores are calculated as log likelihood ratios (LLRs) for each base-pair in the overlap. Furthermore, base-pair score calculation if the bases are the same will be different from the score calculation if the bases do not match (Figure 4).

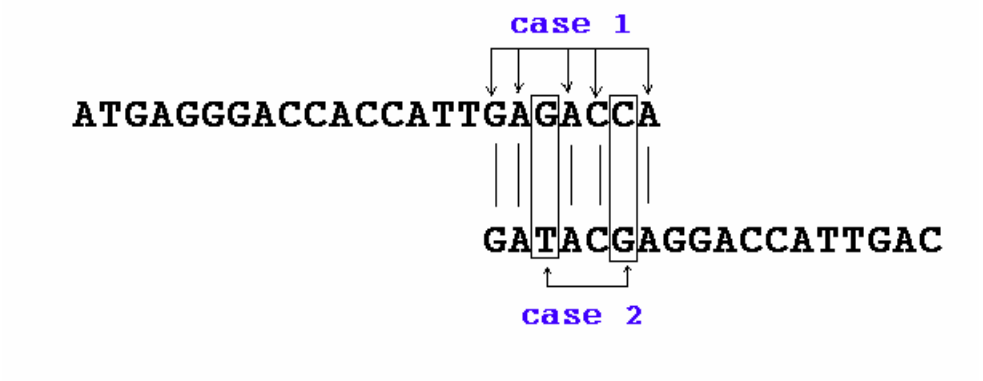


Figure 4: LLR calculation

If the bases in the same base pair are found to be the same then the base-pair LLR is equal to log(1/0.95). If the bases are different then the base-pair LLR is calculated as log((e+f-ef)/ (0.05+0.95(e+f-ef)) where e and f are base error probabilities. The LLRs of each base pair are summed up to form the overlap LLR score. Moreover, this LLR calculation makes the assumptions that alignment positions are independent of each

other, base calls in the two reads disagree at the error position, and errors in reads are independent of each other. The validity of these assumptions is supported by the fact that each read was produced by a separate sequencing reaction.

The greedy algorithm that constructs the layout processes read pair alignment in decreasing LLR overlap score order and merges the read with the contig if there are no negative LLR scoring pairs and the positive LLR scoring pairs cover the region of the overlap. This greedy algorithm usually gets the layout correct due to the ability of LLR scores to distinguish most repeats. Finally, the mosaic consensus sequence is constructed by applying Tarjan's maximum-weight tiling path algorithm to a weighted directed graph that represents the layout (hypothetical path shown in Figure 5). The nodes or ends of heavy-colored arrows in this graph are the read positions in the layout. The edges are either bidirectional between aligned positions in two overlapping reads (vertical arrows) with a weight of zero or unidirectional between positions in the same read in the layout direction (horizontal arrows) with a weight equaling the sum of Phred's base quality values.

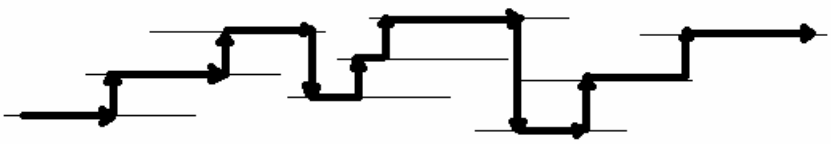


Figure 5: Mosaic consensus: a maximum weight tiling path

Phrap produces a text file with all the contigs and the sequences belonging to them. Our laboratory parses this text file into the database for easier storage and analysis.

1.4 MICROSATELLITE DETECTION ALGORITHMS

Several algorithms are used to pick the primers used during PCR microsatellite polymorphism detection by viewing the flanking regions around possible microsatellites in consensus sequences. However, there is no guarantee of a contig being assembled correctly in this region due to the presence of a microsatellite or a repeat. Thus, MSPF attempts to find and verify microsatellites from sequence overlaps without actual reliance on the consensus sequence information apart from loose clustering of sequences formed by fragment assembly algorithms during the construction of the contig.

While searching for a microsatellite, definition of the minimum number of repeats and mismatch considerations are important. Most of the previous studies have considered only perfect repeats without any mismatches. Many long repeats, however, contain mutations (Katti *et al.* 2001). Such microsatellites are likely to be counted as several separate repeats of shorter length. Moreover, interruptions in a microsatellite could be only a transitional state and could be removed by DNA replication slippage or reverse mutations (Katti *et al*. 2001). Rather than searching for perfect repeats, the detection algorithm must use uniform fuzzy measures to select a probable microsatellite region in the individual fragment that can contain mutations or read errors.

Many algorithms for tandem repeat detection exist. One group is based on computation of alignment matrices with the best running time complexity of O[N^2 polylog(n)] for a sequence of length n and is therefore not useful for long sequences.

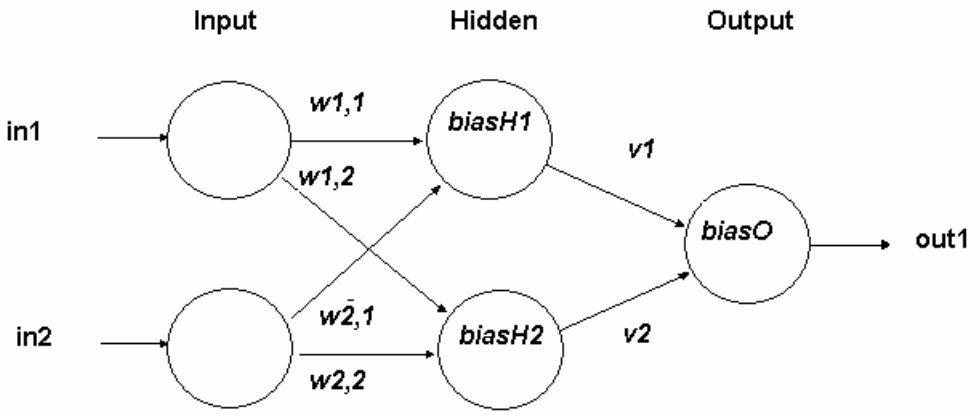
Another group of algorithms finds tandem repeats indirectly using data compression methods (i.e. by detecting 'simple sequence' mixtures of fragments that occur elsewhere), but does not guarantee to give true microsatellites (i.e. CABIOS) (Benson 1999). A third group of algorithms aims more directly at tandem repeat detection but is either too biased toward one type of microsatellite or requires user input concerning microsatellite features such as motif or length. Hence, Tandem Repeats Finder (TRF) was used to detect all microsatellites found in individual sequences. This unique algorithm was chosen according to many factors, including its method of k-tuple matching in avoidance of full scale alignment matrix computations, the use of percentage differences between adjacent copies and separate treatment of substitutions and insertions/deletions, and determination of different repeat units in the same region. TRF also does not require a-priori knowledge about microsatellite's motif, its size, or number of motif copies present (Benson 1999). TRF detects microsatellite regions by scanning the sequence with a small window, determining the distance between exact matches, and testing the statistical criteria calculated dynamically for each microsatellite. These statistical criteria are based on four distributions that depend on microsatellite length, matching probability, insertion/deletion probability, and motif size. Therefore, this algorithm is able to select the best microsatellites without bias toward any distinct types. Since the individual fragments can have sequencing errors, we can measure the quality of each microsatellite using Phred's error probabilities (see implementation).

1.5 MACHINE LEARNING: NEURAL NETWORKS

A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E* (Mitchell 1997). These machine learning methods play essential roles in data-mining and difficult-to-program applications (Mitchell 1997). Since Phred's error probabilities are not exact and an algorithm for accurate determination of these error probabilities does not exist, a machine learning method described below has been used to more accurately determine the quality of each base in the fragment's microsatellite region. Thus, in the supervised learning model used for this project, the machine learning method was trained on sample patterns (*E*) and attempted to minimize (*T*) the sum of differences (*P*) between error predictions with values between 0 and 1 and apparent error occurrences (binary values) for all samples in *E*. A validation set of examples was created to make sure the method was not approximating the training patterns too closely (i.e. overfitting) and was able to handle examples outside the training set.

The Neuroshell2 neural network was used in this research. Artificial neural networks (ANNs) are mathematical models of biological neural systems. An ANN is formed by interconnected processing elements inspired by biological neurons. The interconnections that are similar to synapses within the feed-forward network are such that every neuron in each layer is connected to every neuron in the adjacent layers. Each interconnection has a scalar weight that is adjusted during the training phase. The basic feed-forward network performs a non-linear transformation of input data to approximate the output data by adding up all the incoming signals multiplied by the connection

weights to the neuron's bias weight in each neuron, putting the total through the transfer function, which is usually sigmoidal, and signaling the next layer through all of the outgoing connections with the transfer function output (Figure 6).



out1 = f(biasO+v1*f(biasH1+w1,1*in1+w1,2*in2)+v2*f(biasH2+w2,1*in1+w2,2*in2))
transfer function: $f(x)=1/(1+(1/\varepsilon^{x}))$

Figure 6: Feed-forward neural network topology

A number of papers have shown that a three-layered feed-forward network has the ability to approximate any non-linear continuous function to an arbitrary degree of exactness, given that the hidden layer contains a sufficient number of nodes (Rumelhart 1994). The problem of determining the network weights is essentially a non-linear optimization task. The gradient descent back-propagation method is the most popular training algorithm for the determination of these weights. Each pattern is presented to the network, propagated forward to the output layer, and gradient descent (see below) is used to minimize the total error on the patterns in the training set by changing the weights in proportion to the negative of an error derivative with respect to each weight (Rumelhart 1994).

For example, the change in the output connection weight is calculated using the chain rule as follows. If error $E=1/2(out1-target)^2$, $y= biasO + v1*f(biasH1+w1,1*in1+w1,2*in2) + v2*f(biasH2+w2,1*in1+w2,2*in2)$, and $z_k= biasH_k+w_{1,k}*in1+w_{2,k}*in2$, then $\Delta E/\Delta v_k=\Delta E/\Delta out1 * \Delta out1/\Delta y * \Delta y/\Delta v_k = (out1-target)*out1*(1-out1) * \Delta y/\Delta v_k = p * \Delta y/\Delta v_k$ where $p=(out1-target)*out1*(1-out1)$. Therefore, $\Delta E/\Delta v=p*f(z_k)$ and change in v is equal to $-1* (p*f(z_k))$. Calculation of the change in $w_{k,j}$ is similar to the calculation of the change in $v_{k,j}$ where $E/\Delta w_{k,j} =\Delta E/\Delta f(z_k) * \Delta f(z_k)/\Delta z_k * \Delta z_k/\Delta w_{k,j}$. Moreover, the derivative of the network's error with respect to a hidden node's output is the sum of that hidden node's contributions to the errors of all the output nodes (Smith 1993). Since there is only one output node in our example, $\Delta E/\Delta f(z_k)=p * \Delta y/\Delta f(z_k)=p*v_k$, making the change in $w_{k,j} = -1(E/\Delta w_{k,j}) = -1(p*v_k*f(z_k)(1-f(z_k))*(in_k))$ or $-1(q*in_k)$. For $biasO$ the change is equal to $-p$, and for $biasH_k$ the change is equal to $-q$.

After a certain number of patterns have been processed, a neural network evaluates the validation set and saves the weights that gave the best results for the validation set.

CHAPTER 2

MICROSATELLITE DETECTION

This chapter describes how MSPF can be used in in-silico detection and verification of microsatellites as well as in the overall fragment assembly verification. Section 2.1 explains the application of MSPF in microsatellite detection and verification. Section 2.2 describes the machine learning method used in enhancing the microsatellite detection process.

2.1 MICROSATELLITE POLYMORPHISM FINDER (MSPF) APPLICATION

Due to aforementioned contig errors, in-silico verification is rarely possible. To lower the time required to search through all contig-level microsatellites using chemical methods, redundancy of low-level genotype-specific data in an existing database can be exploited in order to more accurately identify high quality, polymorphic microsatellite markers. MSPF attempts to reliably detect microsatellites, measure the size of genotype-level microsatellites, and to verify the assembly of the consensus sequence through the implementation of virtual PCR. By taking all sequences present in the contig and using sequence-level microsatellite information produced by TRF and machine learning results from Phred/Phrap data, MSPF verifies genotype-level microsatellite length by modeling PCR. Virtual PCR was made possible by calculating the average difference between the positions of best alignments of oligonucleotide primers designed from the highest quality micorosatellite borders to other sequences. For example, Figure 7 shows single “ag”

repeat genotype difference, or the average genotype difference of –2 bases for genotype A003 and the average difference of 0 for genotype A002.

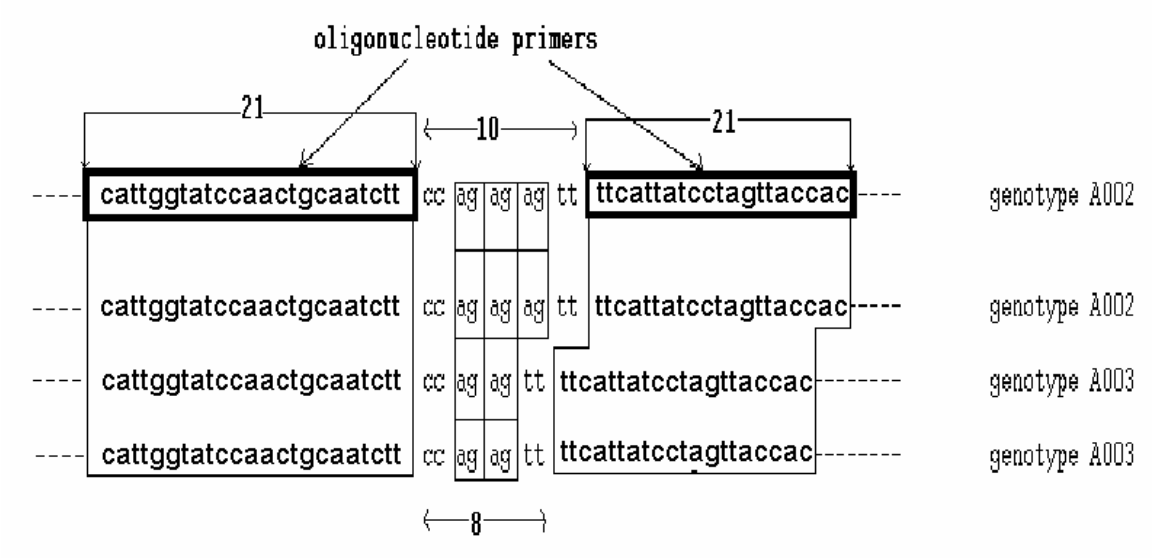


Figure 7: Virtual PCR

That is, by detecting probable microsatellites in sequences, organizing them into groups based on their positions in the consensus sequences, and analyzing the groups to detect the inter-genotypic length differences, it was possible to find a subset of microsatellite polymorphisms as a byproduct of initial sequencing without extra work for microsatellite genotyping done separately. A polymorphic microsatellite identified by MSPF is shown in the viewing interface in Figure 8, where the stars in reads represent the insertion/deletion symbols used by Phrap for fragment alignment. This interface accesses the database (see section 3.4) to display the microsatellite regions found in individual sequences on the right side. A single sequence with a shorter microsatellite

length has a genotype code of A011 which is included at the end of the sequence name on the left side.

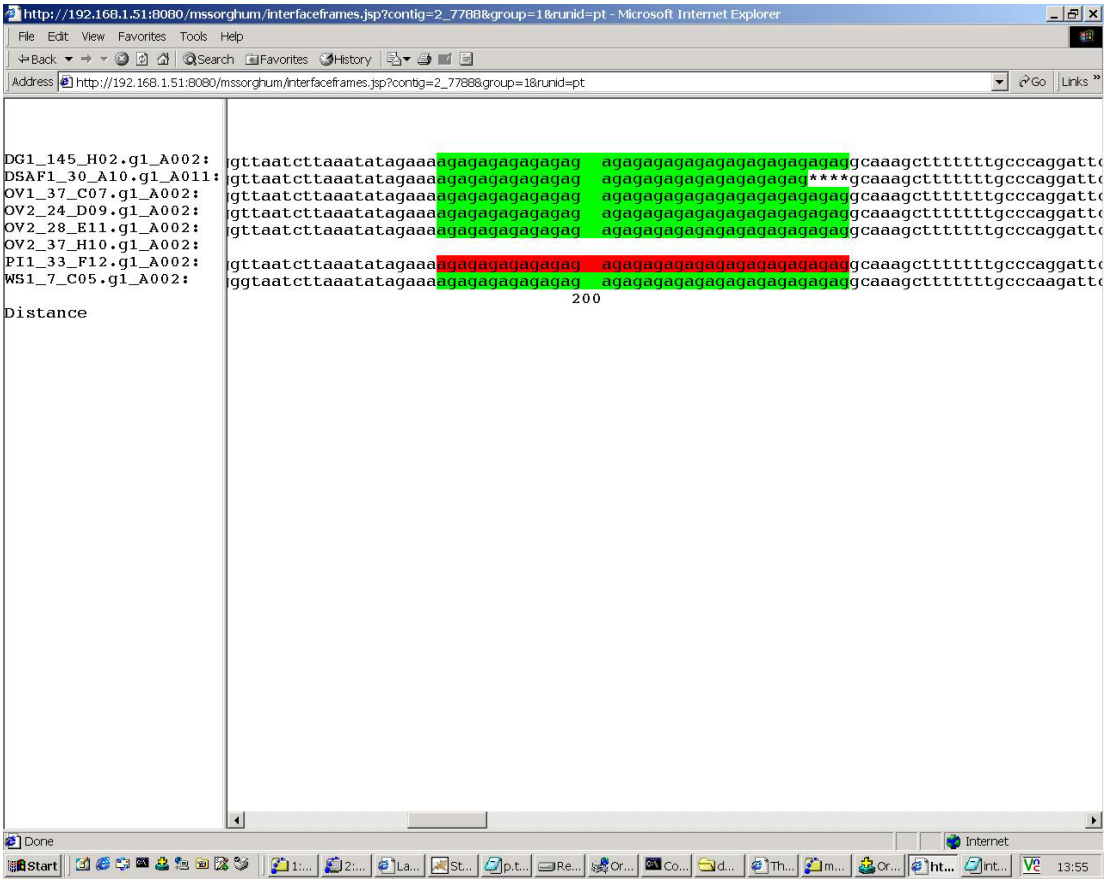


Figure 8: Polymorphic microsatellite

Moreover, misassembled contigs were identified by detecting contigs that either had a large variance in microsatellite lengths found in individual sequences of the same genotype or had sequences missing a specific microsatellite that occurred in other sequences.

2.2 MACHINE LEARNING METHODS IN MICROSATELLITE DETECTION

Phred error probabilities have been confirmed experimentally. The accuracy of each base call, however, remains questionable. In an effort to improve the accuracy of base quality values and, subsequently, the accuracy of microsatellite quality values, machine learning has been used to modify base quality values in sequences where a microsatellite has been detected. By compiling Phred errors, a neural network was used to predict the probability of error around a given base based on Phred statistics as well as a short stretch of chromatogram data around the base (see section 3.2). Since Phred uses the function (quality value)= -10*log(error probability), Phred error probability and the machine learning method probability were weight-averaged in an effort to more precisely identify the accuracy of microsatellite region base-calls and produce a better overall quality value for improved virtual PCR verification. If a machine learning error probability was strong (within 0.2 of 1 or 0), Phred's error probability for the same base was extracted from the quality value, pulled one third of the way toward neural network error probability ($1/3^{rd}$ gave the best results), and a new quality value was calculated from the resulting error probability. Once all base quality values were dynamically generated for each detected microsatellite in the fragment, the overall microsatellite quality value was recalculated.

2.3 CONSENSUS SEQUENCE VERIFICATION

Since sequence assembly programs often generate misassembled contigs, some misassembled contigs can be identified by finding sequences that overlap the contig-level microsatellite but do not contain a copy of that microsatellite. This usually occurs when

a different sequence overlaps with the contig outside a short microsatellite region and causes the greedy algorithm to form a wrong layout. Since the consensus sequence ends up being misassembled, it is unlikely that it represents a valid coding region for a protein sequence.

CHAPTER 3

IMPLEMENTATION

This chapter describes the implementation of MSPF. MSPF has been modeled and implemented through an Oracle relational database management system (RDBMS), but it could have been written to use Phrap text files just as easily. Techniques used to ensure the reliability of the results included the use of Phred error probabilities (averaged with machine learning error probabilities) for the bases in the sequences' microsatellite regions and Phrap consensus sequence alignment information. Section 3.1 deals with sequence-level microsatellite detection and explains contig-level microsatellite clustering. Section 3.2 explains machine learning methods used in microsatellite detection. Section 3.3 explains inter-genotypic polymorphism detection. Finally, section 3.4 describes the design of the MSPF schema and flow of data.

3.1 SEQUENCE-LEVEL MICROSATELLITE DETECTION AND CONTIG-LEVEL MICROSATELLITE CLUSTERING

Contig-level microsatellites were detected from sequence and contig data produced by Phred/Phrap. A graphical user interface for MSPF (Figure 9) was written for passing parameters and the structure of the database to the program in order to analyze data internally, and insert it back into the database in the new schema.

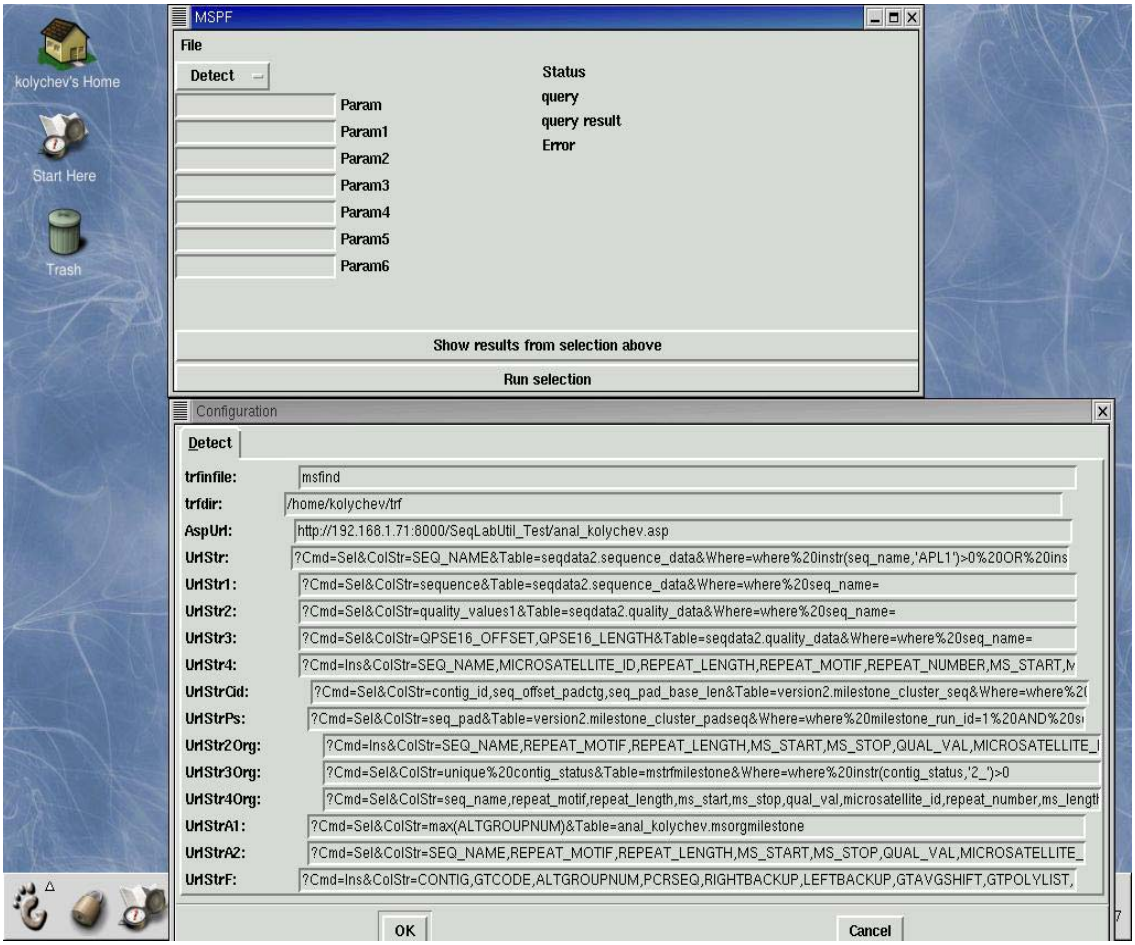


Figure 9: MSPF interface

The first step involved looking for the microsatellites, both high and low quality ones, and computing their attributes (length, quality, etc.). TRF was used to locate the microsatellites in Phred-produced sequences. Since sequencing errors and mutations are usually found in the individual fragments, TRF identifies the parts of the sequence that are most likely to be microsatellites. Moreover, because TRF identifies variable number tandem repeats (VNTRs) with long motifs, only probable microsatellites with motif length from two to six were used.

An average quality value of each microsatellite was computed by reading TRF's alignment of the found microsatellite to the perfect copy of that microsatellite. Reading the alignment sequentially, if the actual base was the same as the base in the perfect theoretical microsatellite, the quality value of the base in the actual copy was multiplied by two. If an error occurred, the quality value (or previous quality value in the case of deletion) was multiplied by negative seven. The parameters of two and seven are the strictest among the most common ones used for local Smith-Waterman alignment algorithm and have been widely used in sequence comparison search algorithms with good results. To calculate the average quality value, the sum of all quality values was divided by the number of bases in the perfect microsatellite (Figure 10).

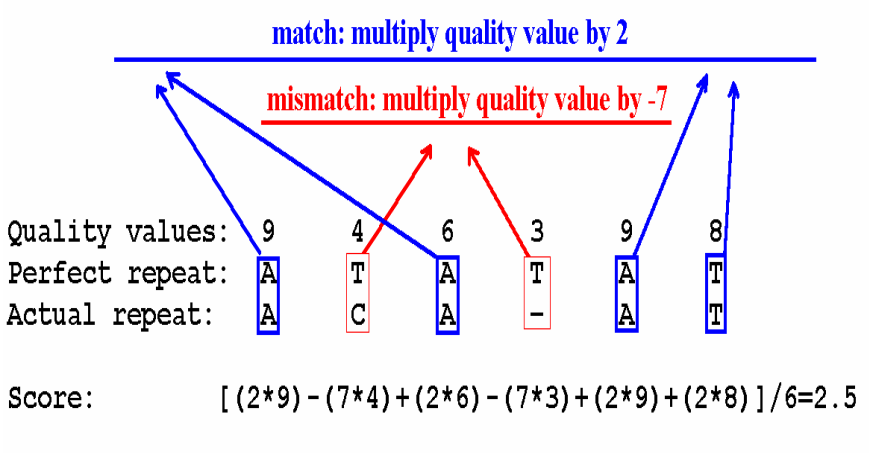


Figure 10: Microsatellite quality calculation

Then, the microsatellites were analyzed and sorted based on their positions in the contigs. Since consensus sequence microsatellite detection is error-prone, only low-granularity alignment information was used. That was accomplished by creating

groups/clusters of same-motif microsatellites for each contig where each microsatellite in a group either started or ended within a user-provided threshold (a parameter of 7 was found to give good groups), or was completely "inside" a bigger microsatellite (Figure 11).

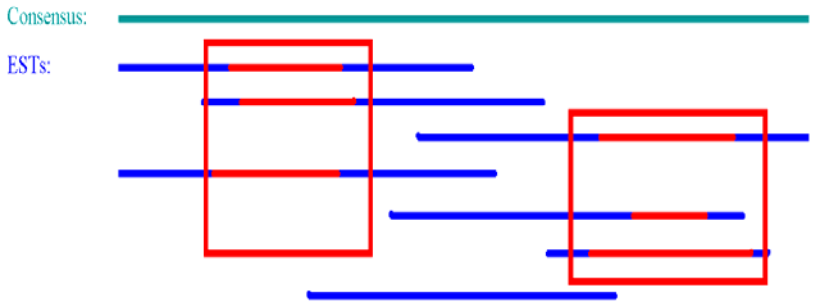


Figure 11: Contig-level grouping

All other properties were determined by correlating results of TRF to the database information. This sequence-level microsatellite information was inserted into the schema in the ms_in_seq table (Figure 12, see section 3.4).

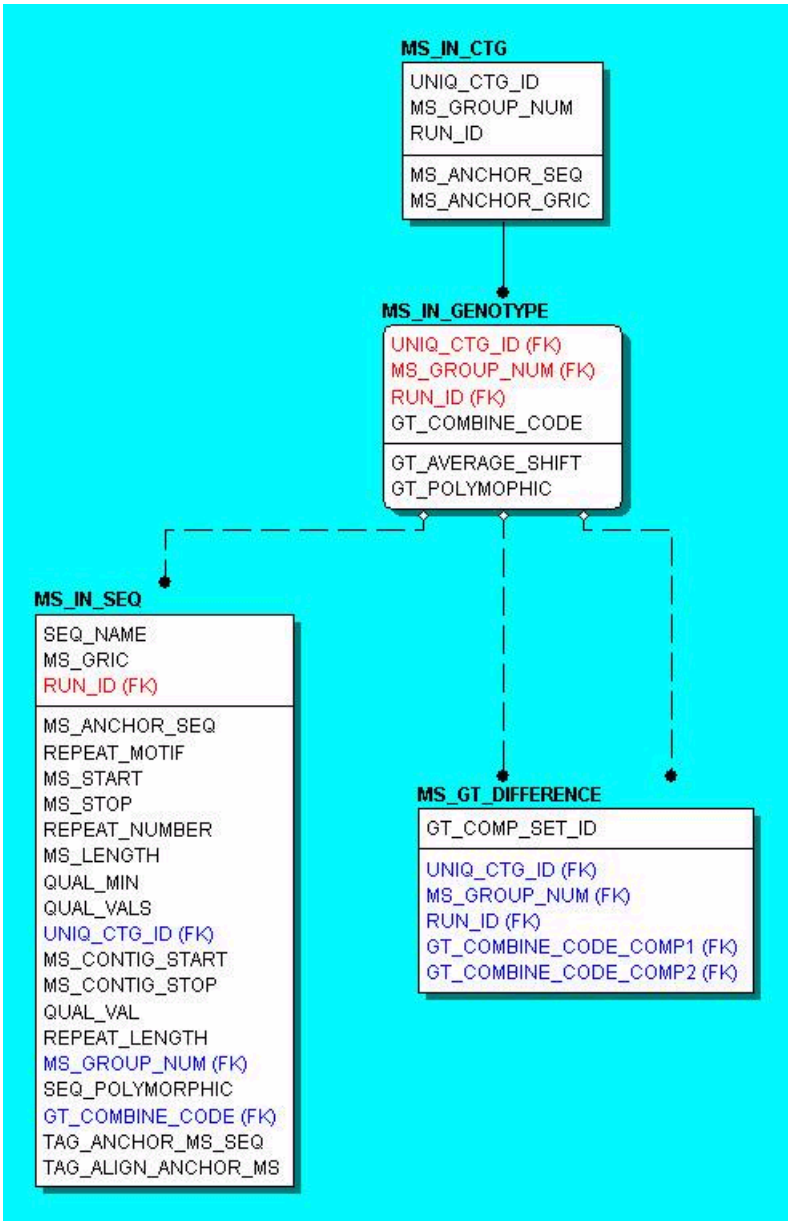


Figure 12: MSPF schema

3.2 NEURAL NETWORK SETUP

To provide the problem/solution training set for the neural network, apparent Phred sequence errors were extracted from the database. Information included from Phred results contained a stretch of five bases around the error, their quality values, and their distance from the apparent error based on the chromatogram. Information from the sequence chromatogram (Phred source code was modified to output the trace values) included the trace position of the error and four stretches of fifty trace values around the error, creating a total of 216 inputs. The inputs were chosen to correspond with the information Phred uses to calculate the error probability. To extract examples where Phred made an error, well-aligned contigs and sequences belonging to them were scanned. Locations in sequence overlaps that had different bases, including the padding symbol used to align the sequences, from the majority of corresponding bases in different sequences and the contig were tagged. Tagging occurred after verifying the sequence in which an error was found had no more than two insertions/deletions in the window of length ten that was centered on the error location and had overlaps formed with at least three sequences of the same genotype and the contig with more than 95% of bases matching. The parameter of 95% was used since Phrap uses that value during LLR calculation. The window length of ten was chosen because it was long enough to produce a high Phred error probability, yet short enough to match locations close to the ends of the sequences which were given preference during the insertion into the training set since traces are more prone to distortion near the termini of a polymer-filled capillary. Then, the error position in the sequence was aligned to the trace location and the trace values around the error location were extracted. Around eight hundred positive (different

base) and three thousand negative (error-free, random location) samples were located. Approximately, 500 positive and 2000 negative samples were extracted into the training set with the rest of the samples placed in the validation set.

On a sidenote, an attempt was made to use the XCS classifier system instead of a neural network. XCS maintains a population of classifiers where each has a fitness based on a measure of the prediction accuracy (Wilson 1995). XCS executes a genetic algorithm (where each classifier represents a single rule a.k.a. Michigan approach) to find the optimal classifiers in different accuracy niches defined by training samples (Kovaks 1996). In the result evaluation stage, each classifier is matched to the validation set. If the classifier matched a sample, the prediction with a value 0 or 1 was multiplied by classifier's accuracy/fitness to produce the error probability estimate and compared to the correct sample output. XCS classification resulted in eleven classifiers after five runs with a population size of five hundred and two million fitness determinations. Unfortunately, XCS was unable to generalize well, correctly matching only one resulting classifier with the validation set sample in the fourth run and one in the fifth run and incorrectly matching one classifier in the fifth run. Furthermore, it required several days for each run to produce the classifiers. Better results were not found after changing the scaling criteria and using a smaller number of fitness evaluations.

Thus, the NeuroShell2 feed-forward, three-layer, gradient descent back-propagation network with 990 hidden nodes, which achieved seventy five percent accuracy on the validation set after about 150 epochs or twenty-four hours, was used. The number of approximately 700 hidden nodes was suggested by Neuroshell2 after

defining the inputs and selecting the "complex" granularity function estimate. Samples were selected randomly from the training set. The number of training samples processed between each validation set evaluation was set to 7600, or about three epochs. Since the training took approximately 24 hours, the small number of epochs between validation set evaluations was used to visually track the improvements during the neural network training process. Overfitting was prevented by saving the configuration that gave the best results for the validation set. No scaling was done, and the comparison of the runs with varying number of hidden nodes showed that the most accurate validation set classification was provided by the network with 990 hidden nodes.

3.3 MICROSATELLITE VERIFICATION

Polymorphisms were identified based on the analysis of the microsatellites in the unique contig/group combination to see if some of them had different numbers of repeats that could be correlated to the genotype from which they originated. While a straightforward approach of seeing whether different genotypes have a different number of repeats in the same place is fairly sensitive, it also produces a large number of false-positives because of errors in the original sequences or misassembled contigs. To get around the problem of having to review a large number of possible polymorphisms manually, the average genotype fragment length between two primers produced by virtual PCR was used instead of the numbers of repeats. Since TRF often does not identify the exact starting and ending position of the microsatellite in the contig, the primers were recorded from the best sequence's flanking regions outside the longest, highest-quality microsatellite in a group ([(quality value*length)+(Σ(border quality

values))] parameter was maximized). Thus, the highest-quality microsatellite was picked based on quality values produced by Phred error probabilities or, in the neural network version, on a weighted average of Phred and machine learning error probabilities. A certain number of bases (parameter set to two) were skipped before the start and after the end of the microsatellite to make sure the recorded adjacent borders of length twenty-one (oligonucleotides primers) were outside the microsatellite region. These oligonucleotide lengths are commonly used during chemical PCR since the probability that they will match in more than one place in the genetic sequence is fairly low. Then, every other microsatellite in the same group was analyzed to see whether the length between the recorded parts (borders) was different for some genotypes on average. This was done by sequentially comparing the recorded border with a same-length part (even a low quality part not used in the contig) of the sequence in which the other microsatellite was found and recording the location of the best match. Comparison was done by calculating the Levenshtein's edit distance or the number of insertions/deletions between two strings. Clearly, the accuracy of this approach will increase with the number of sequences in the contig. While this approach is unable to find microsatellites that are located close to the ends of the sequence, it proved to be very accurate. Results were parsed into ms_in_ctg, ms_in_genotype, and ms_gt_difference tables.

Fail-safe conditions: To make sure that results were accurate, a certain number of parameters that were briefly mentioned above were included, and most of the values were borrowed from specific parameters used for sequence assembly and similarity search programs and adjusted slightly to maximize the quality of results. An option was set to pass the parameters from user input, or the defaults (see below) were used if the

parameters remained undefined. If a microsatellite had an average quality below a user-supplied threshold, started extremely close to the beginning of the sequence, or ended close to the end of the sequence, it was not used. The default value of 32 for the threshold was chosen since it would correspond to a perfect microsatelite with all Phred quality values of 16, which is a threshold quality value used in the lab to truncate low quality sequences. To make sure that the borders had no chance to be identified in other parts of the sequence, sequence-level microsatellites were not considered if the border difference on each side of the microsatellite was higher than a certain threshold (20) of the calculated average or if the border match was lower than 0.81. Polymorphisms were identified if the average border-gap difference was higher than two, or the least value of difference for the polymorphic microsatellite with the motif length of two.

3.4 MSPF SCHEMA DESIGN AND FLOW OF DATA

Microsatellites were grouped on three levels shown in separate tables in Figure 8. Sequence-level information was recorded in ms_in_seq table, contig-level information in ms_in_ctg table, and genotype-level information in ms_in_genotype table. Moreover, the genotype difference or polymorphic genotype-level microsatellite information was recorded in the ms_gt_difference table. Primary/foreign keys were used to keep the relationships valid. Every table contained a RUN_ID attribute to separate different executions of MSPF.

Table ms_in_seq had a primary key with three fields: SEQ_NAME, a sequence in which a microsatellite has been found, MS_GRIC, microsatellite's general record insert count which is unique in the same SEQ_NAME, and RUN_ID. REPEAT_MOTIF

attribute contained the microsatellite motif that repeats throughout the microsatellite region REPEAT_LENGTH attribute described the number of bases in the motif. MS_START contained position where the microsatellite starts in the fragment, MS_STOP enclosed the position where the microsatellite ends in the fragment, and MS_LENGTH contained the difference of the two previous values. MS_CONTIG_START recorded the position where the microsatellite starts in the contig and MS_CONTIG_STOP enclosed the position where the microsatellite ends in the contig. REPEAT_NUMBER contained TRF's best guess at the number of repeats present in the fragment's microsatellite. After extracting Phred base values for each base in the fragment's microsatellite region, QUAL_VALS contained the extracted sequence, QUAL_MIN enclosed the lowest quality value in the extracted sequence, and QUAL_VAL described the overall value of the microsatellite computed with the procedure described in section 3.1.

Since microsatellites were clustered into groups based on fragments' alignments in a contig table ms_in_ctg had a primary keys with four attributes: UNIQ_CTG_ID which contained the unique identification string for a contig, MS_GROUP_NUM that enclosed unique group id of all microsatellites that overlapped within a specific RUN_ID, UNIQ_CTG_ID combination, and RUN_ID. MS_ANCHOR_SEQ and MS_ANCHOR_ID specified the best-quality microsatellite that provided the oligonucleotide borders used for virtual PCR verification. TAG_ANCHOR_MS_SEQ was used if the microsatellite was found to be the best one in the contig-level group, and TAG_ALIGN_ANCHOR_MS was used if it was verified by virtual PCR.

Moreover, these contig groups were broken up into genotype-level groups in the ms_in_genotype table. It had a primary key of length four where the first three fields were also used as a foreign key from the ms_in_ctg table: UNIQ_CTG_ID, MS_GROUP_NUM, RUN_ID, and the GT_COMBINE_CODE which described a specific genotype. Also, GT_POLYMORPHIC tag was used if a genotype-level microsatellite was different from another one and GT_AVERAGE_SHIFT was updated with the average length difference of a genotype-level microsatellite from the best sequence-level microsatellite in a unique RUN_ID, UNIQ_CTG_ID, MS_GROUP_NUM combination .

If two distinct genotypes from the same primary key in ms_in_ctg table had different GT_AVERAGE_SHIFT values in the ms_in_genotype table, the difference was recorded in the ms_gt_difference table. It had a GT_COMP_SET_ID primary key, which provided a unique identification number of each difference in genotype-level microsatellites. Foreign key combinations of UNIQ_CTG_ID/MS_GROUP_NUM/RUN_ID/ GT_COMBINE_CODE_COMP1 and UNIQ_CTG_ID/MS_GROUP_NUM/RUN_ID/ GT_COMBINE_CODE_COMP2 referenced specific primary keys in the ms_in_genotype table.

Because the ms_in_seq and ms_gt_difference tables contained the primary keys from ms_in_genotype table which, in turn, referenced ms_in_ctg primary keys, those primary keys were inserted first, in the ms_in_ctg table, and then in ms_in_genotype tables as the contig-level groups were found. Relevant ms_in_seq attributes were inserted after the contig-level clustering . After virtual PCR verified each genotype-level microsatellite in the contig group using the primers from the best-microsatellite in the

group, non-primary attributes in ms_in_ctg, ms_in_genotype and ms_genotype_difference tables were updated. The MS_ANCHOR_SEQ, MS_ANCHOR_ID, TAG_ANCHOR_MS_SEQ, and TAG_ALIGN_ANCHOR_MS attributes were also updated in the ms_in_seq table at that time.

CHAPTER 4

ANALYSIS

This chapter provides the details of the execution of MSPF on highly inbred sorghum sequences (low amount of genotype variability). Moreover, since the total number of genotypes was fairly low, not many polymorphisms have been detected. Section 4.1 describes the input. Section 4.2 describes the output from MSPF.

4.1 INPUT

TRF was executed on a total of 158,901 sorghum sequences. Genotypic counts were:

Genotype code	Number of sequences
A001	1,210
A002	115,914
A003	28,318
A010	4,608
A011	8,448

Approximately 58,000 of these sequences were assembled by Phrap into a total of about 9000 contigs with average length of 651 nucleotides, each consisting of two or more sequences. Approximately half of the contigs contained sequences of different genotypes.

4.2 RESULTS

A total of 719 microsatellite regions (with motif lengths from two to six) were found with an average length of 26. Approximately two days were needed to produce all sequence- level microsatellite information using TRF and Phred results on a 2Ghz Intel P4 machine with 512 MB of RAM. PCR verification took less than an hour on a 4,400 member subset of all sequence-level microsatellites which were organized into contig-level groups. Since, neural network evaluations were preprocessed, enhanced PCR verification did not need more processing power. A large majority of microsatellite regions found contained a motif length as a multiple of three, the size of a codon (Figure 13). The most common motifs and their frequencies are illustrated in Figure 14.

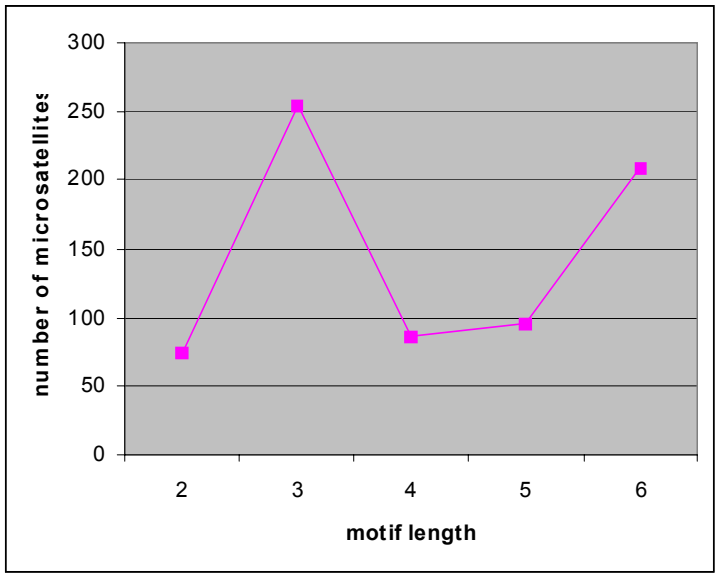


Figure 13: Microsatellites of specific motif lengths

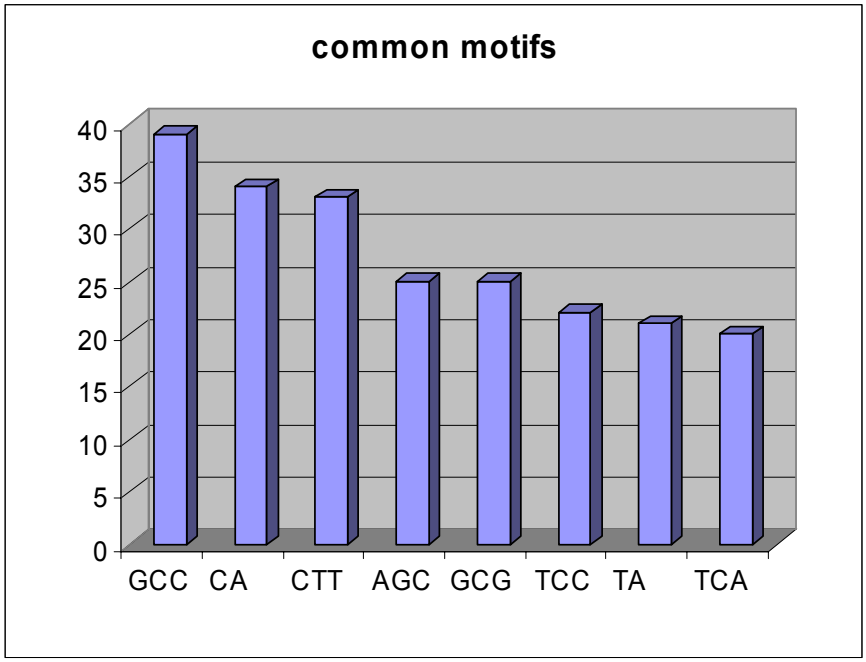


Figure 14: Common motifs

Among all microsatellites, slippage-mediated expansions/deletions of only repeats with a motif length as a multiple of three can be tolerated in coding regions because they do not disturb the reading frame. If an insertion/deletion of length other than a multiple of three occurs, the resulting protein sequence will be different in all amino acids inserted after the mutation point since mRNA is translated in sets of three bases. This change is likely to destroy any chance that the protein will fold into a functional enzyme and make the organism unfit for future selection.

On average, one microsatellite was found for less than every 8.5 kb of contigs surveyed, performing better than other estimates that were done conventionally (Edwards et al. 1991). MSPF, because of its use of virtual PCR, determines SSR length accurately

irrespective of accumulated mutations. For example, a sequencing error at the end of a microsatellite (Figure 15) did not result in erroneous detection of a polymorphism (Figure 16).

Figure 15: Single nucleotide polymorphism

Figure 16: Microsatellite example

The neural network enhanced version was able to detect several contig-level microsatellite regions more accurately by picking better sequence-level microsatellites (best microsatellite changed in forty cases or in 5.5% of the dataset) for the basis of digital PCR, thus producing lower average genotype shifts (average difference of 0.005 for the entire dataset) and, in several cases, including more sequences in the contig-level microsatellite. Since the locations close to the ends of the reads were given preference for extraction into the training set, the neural network improved results in the lower quality regions close to the end of the sequences, the regions that are usually not completely included in the contig but still used in microsatellite verification.

Presumably due to the fact that 73% of the ESTs came from a single, highly inbred sorghum genotype, polymorphic SSRs were identified in only 1.5% of all contigs, for a total of 10 (Table 1).

Table 1: Microsatellite polymorphisms

Contig number	Genotype code	Number of sequences	SSR motif	SSR length
2_3653	A010	1	CTACA	22
	A002	2	CTACA	27
2_5460	A010	1	GA	48
	A011	1	GA	56
2_7830	A002	3	GGCGCT	28
	A010	1	GGCGCT	22
2_840	A003	1	CACTG	25
	A001	1	CACTG	20
2_8380	A002	5	ACCCA	30
	A010	1	ACCCA	20
2_3241	A010	1	AC	53
	A002	1	AC	63
2_8299	A003	1	CTG	36
	A002	4	CTG	51
2_7121	A011	1	TAA	24

	A003	1	TAA	30
	A002	2	TAA	24
2_7788	A002	5	AG	38
	A011	1	AG	34
2_6757	A002	3	AG	25
	A003	1	AG	23

All of the polymorphic microsatellites differed by a whole number of repeat units since they were produced by slipped-strand mispairing.

Twenty two contigs out of 719 that contained verified microsatellites have also been found to be misassembled by Phrap where a sequence missing a specific microsatellite was included in the consensus sequence (Figure 17).

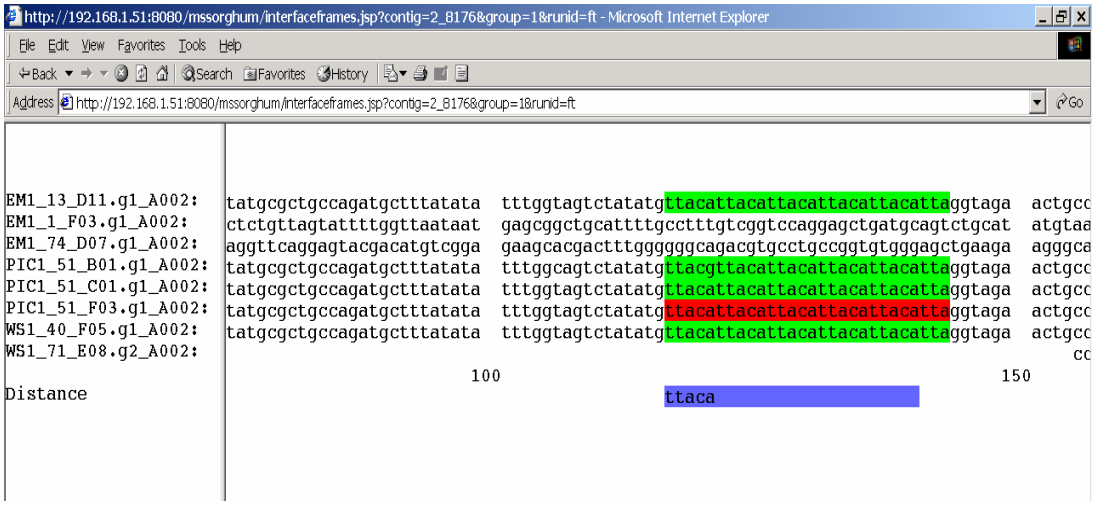


Figure 17: Misassembled contig

CHAPTER 5

CONCLUSIONS

Since MSPF finds reliable microsatellites and microsatellite polymorphisms using redundant low-level data with minimal reliance on contig assembly information, it can be used as a reference for faster, cheaper PCR verification. Moreover, this is the first in-silico approach that accurately detects microsatellite polymorphisms. It was able to correctly identify 1.5% of all microsatellites as being polymorphic even with the low coverage provided by the dataset. On average, it detected a microsatellite for approximately every 7 kb of the coding regions surveyed. As the sizes of shotgun-sequencing/EST databases continue to increase exponentially worldwide it can provide an automated, low-cost method of genotyping that is run completely in-silico and without the requirement for expensive materials or human intervention used in chemical verification. Furthermore, we have shown that sequencing accuracy can be improved using machine learning methods. All of the reviewed methods classified training samples found in old data and bootstrapped the existing algorithm to a higher degree of accuracy. An artificial feed-forward, three-layer, gradient-descent back-propagation neural network has been found to be more useful than the XCS classifier system in this domain. The neural network was able to classify more training samples into general groups in less than 24 hours as opposed to several days needed to evolve the XCS classifiers. It achieved 75% accuracy on the validation set, and produced accurate predictions that were weight-averaged with the existing values. These new solutions improved the quality of virtual

PCR verification by more accurately predicting base error probabilities in individual sequencing reads. Genotype shift attributes produced by virtual PCR improved by approximately 0.4% on average after using forty new solutions produced by the neural network. More experimentation with machine learning methods could be performed to achieve higher quality solutions not only for virtual PCR microsatellite verification but for fragment assembly problems as well by using higher-accuracy base quality values.

REFERENCES

Benson, G., "Tandem repeats finder: a program to analyze DNA sequences", *Nucleic Acids Research*, Vol. 27, No. 2, pp. 573-580, 1999.

Chan, S. C., Wong, A. K. C., and Chiu, D. K. Y., "A survey of multiple sequence comparison methods", *Bull. Math. Biol*. 54:563-598, 1992.

Edwards A,. Civitello A, Hammod H and Caskey C, "DNA typing and genetic mapping with trimeric and tetrameric repeats", *Arn. J. Hurn. Genet*. 49: 746-756, 1991.

Ewing, B., Hillier, L., Wendl, M., Green, P., "Base-Calling of Automated Sequencer Traces Using Phred. I. Accuracy Assessment", *Genome Res*. 8: 175-185, 1998.

Ewing, B., Green, P., "Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities", *Genome Res*. 8: 186-194, 1998.

Goldberg, D., **"**Genetic Algorithms in Search, Optimization & Machine Learning"**,** Addison-Wesley, 1989.

Katti, M., Ranjekar, P., Gupta, V., "Differential Distribution of Simple Sequence Repeats in Eukaryotic Genome Sequences", *Mol. Biol. Evol. 18(7)*: 1161-1167, 2001.

Kovacs, T., "Evolving Optimal Populations with XCS Classifier Systems", MSc. Dissertation, Univ. of Birmingham, UK, 1996.

Mitchell, T., "Machine Learning", McGraw Hill, 1997.

Paetkau D., Cavert W, Sterling I., Strobeck C., "Microsatellite analysis of population structure in Canadian polar bears", *Mol. Ecol. 4*: 347-354, 1995.

Parsons, R., Forrest, S., Burks, C., "Genetic Algorithms for DNA Sequence Assembly", ISMB 1993: 310-318

Pevzner P., Tang H., Waterman M., "An Eulerian path approach to DNA fragment assembly", Proc. *Natl. Acad. Sci*., 2001 Aug 14; 98(17):9748-53.

Pevzner P., Tang H., Waterman M., "A New Approach to Fragment Assembly in DNA Sequencing", Proceedings of *The 5th Annual International Conference on Computational Molecular Biology*, pp.256-267, Canada. ACM Press, 2001.

Pevzner P., Tang H., "Fragment assembly with double-barreled data", *Bioinformatics*, 2001 Jun;17 Suppl. 1:S225-33, (Special ISMB 2001 issue).

Queller, D., Straussman, Joan, E., Hughes, C., "Microsatellites and kinship", *Trends in Ecology and Evolution 8*: 285-288, 1993.

Rumelhart, D., Widrow, B., Lehr, M., "The basic ideas in neural networks", *Communications of the ACM*, v.37, n.3, pp.87-92, March 1994.

Sun Kim, Liao, L., Perry, M., Zhang, S., Tomb, J., "A Computational Approach to Sequence Assembly Validation", Poster in *The 8th International Conference on Intelligent System for Molecular Biology*, San Diego, California, August, 2000.

Smith, M., "Neural Networks for Statistical Modeling", Van Nostrand Reinhold, 1993.

Sun Kim, Liao, L., Tomb, J., "A probabilistic approach to sequence assembly validation", *1st Workshop on Data Mining in Bioinformatics*, San Francisco, CA, pp. 38 – 43, August 26, 2001

J. Weissenbach et al, "A second generation linkage map of the human genome", *Nature* 359: 794-801, 1992.

Wilson, S.W., Goldberg, D.E., "A Critical Review of Classifier Systems", in

Proceedings of the Third International Conference on Genetic Algorithms, pp. 244-255, Los Altos, California, 1989.

Wilson, S.W., "Classifier Fitness Based on Accuracy", *Evolutionary Computation*, 3 (2), MIT Press, 1995.