Pipeline For the Quality Control of Sequencing

YUEQIN CHEN

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

MAY 2003

Your file Votre référence
ISBN: 0-612-83897-8
Our file Notre référence
ISBN: 0-612-83897-8

ABSTRACT

Pipeline For the Quality Control of Sequencing

Yueqin Chen

Sequencing technology has been developed for more than decades. In all methods applied, however, sequencing quality remains a problem. In most cases, raw DNA sequences obtained from automatic sequencing machines are not reliable, and the output sequence might contain errors, vector contamination might occur, Dye-terminator reaction might not occur, and segment migration might be abnormal in gel electrophoresis. In order to trim low quality regions and sequences, to remove contamination and to build reliable assembled contigs, we constructed the pipeline, which uses Phred, Lucy and Phrap tools together. Phred takes chromatogram data as input, then makes base calls and assigns quality values and finally generates sequence and quality files in FASTA format. In a subsequent step, Lucy trims vectors and low quality sequences, makes a clean range of each sequence. At the end Phrap will assemble these sequences using pairwise alignment information from the Smith-Waterman algorithm. The consensus sequences of the assemblies built by the pipeline are assumed to be reliable genes that can be used for gene annotation. Furthermore, we compared two pipelines, with and without the use of Lucy. The output clearly indicates that applying Lucy in the sequencing process generates more reliable consensus sequences.

Acknowledgements

I honestly thank my supervisor, Dr. Gregory Butler, for his encourage and invaluable guidance. His enrich knowledge in computer science and bioinformatics provided wide information and endless resource. Meanwhile, I sincerely appreciate the support and advices from Dr. Adrian Tsang and Dr. Clement Lam, and biological introduction and discussion from Dr. Jack Min, Yi Tao and Chellapa Gopalakrishnan, and technical support from Jian Sun, Sindhu K. Pillai. Also I would like to thank all fellow students in Dr. Butler's group for their helpful discussion and friendship. Finally, I dearly thank my parents and my wife for their understanding and supporting for my study.

Table of Contents

List of Figures

Chapter One

Introduction

1.1 Sequencing Background

The development of improved DNA (deoxyribonucleic acid) sequencing technologies in the 1980s and 1990s heralded the start of a new era in biology. Sequencing of DNA requires the following key steps: preparation and replication of short segments of DNA; creation of partial copies of the segments each, one base longer than the next; identification of the last base in each copy; and ordering of the bases [1]. The short segments of DNA are created by fracturing a source strand with sound or passing it through a nozzle under pressure. These short DNA segments are inserted in a plasmid vector, typically a bacterial virus (see Figure 1). The virus then infects a bacterium with the DNA segments, also known as an insert. These plasmid vectors are stored in cDNA library. Later on, in preparation for sequencing, the DNA insert must be extracted from the vector and then amplified. This is accomplished using a routine process known as polymerase chain reaction (PCR). For each sample to be sequenced, copies are made with each one varying in length by one base using restriction enzymes. The fragments are then labeled with one of four fluorescent dyes, corresponding to the last base. There are many sequencing methods to get this work done; each of these sequencing schemes consists of

enzymatic reactions, electrophoresis, and some detection technique [2]. The most popular method for doing this is called the dideoxy method. DNA is synthesized from four deoxynucleotid triphosphates. The top formula shows one of them: deoxythymidine triphosphate (dTTP). Each new nucleotide is added to the 3' OH group of the last nucleotide added (see Figure 2). The dideoxy method gets its name from the critical role played by synthetic nucleotides that lack the OH at the 3' carbon atom (red arrow). A dideoxynucleotide (dideoxythymidine triphosphate - ddTTP - is the one shown here) can be added to the growing DNA strand but when it is indeed added, the chain elongation stops because there is no 3' OH for the next nucleotide to be attached to. For this reason, the dideoxy method is also called the **chain termination method** (see Figure 3). First, four separate reactions are set up with a single-stranded DNA fragment annealed with an oligonucleotide primer. Each reaction contains the four normal precursors of DNA - that is, the deoxynucleotides dATP, dTTP, dCTP, and dGTP, together with the DNA polymeraseas being used in the natural DNA replication. In addition, appropriate amounts of four dideoxynucleotide terminators, ddATP, ddTTP, ddCTP, and ddGTP, are also present in the respective reactions. In the ddA reaction containing ddATP, polymers are extended from 5' to 3' end by polymerase according to the template DNA, and the elongation of new strands is stopped once a ddATP is incorporated. Because the incorporation of ddATP is random, the ddA reaction should produce many copies of each possible sub-fragment starting with the same primer and ending with ddATP. Similarly, the ddG, ddC, and ddT reactions will produce many copies of each possible sub-fragment ending with ddGTP, ddCTP, and ddTTP respectively [3, 4]. Next, electrophoresis is used to separate the DNA sub-fragments produced from the four reactions. DNA fragments are

negatively charged in solution. If we load the DNA fragments into a gel and apply an electric field, the fragments will move in the gel. The smaller the size of a fragment, the faster it runs through the electric field. Then electrophoresis uses a laser to activate the fluorescent dideoxynucleotides and a detector to distinguish the colors. At the end of the incubation period, the fragments are separated by length from longest to shortest. Each of the four dideoxynucleotides fluoresces at a different color when illuminated by a laser beam and automatic scanner provides a printout of the sequence.

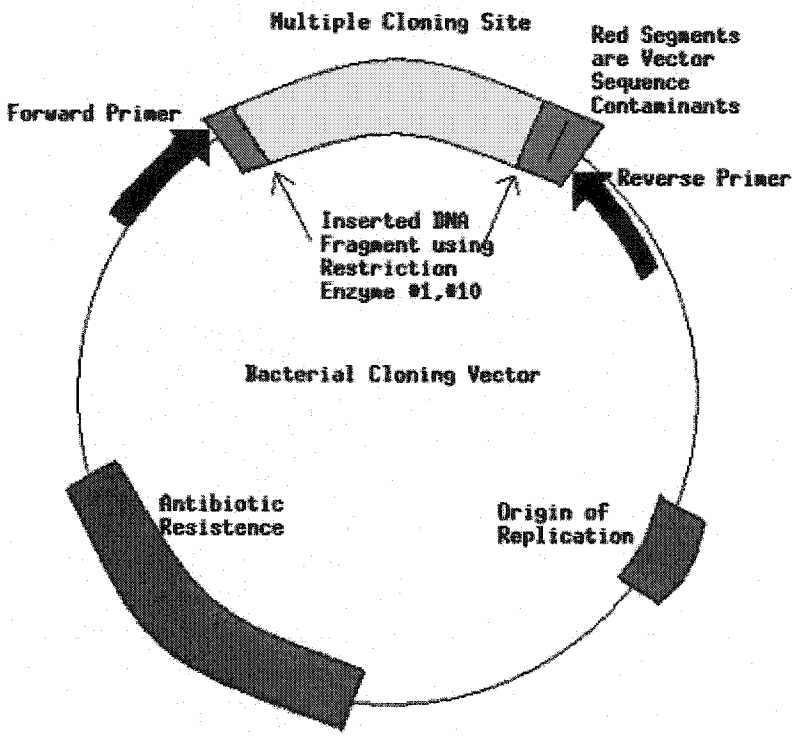


Figure 1: Vector diagram

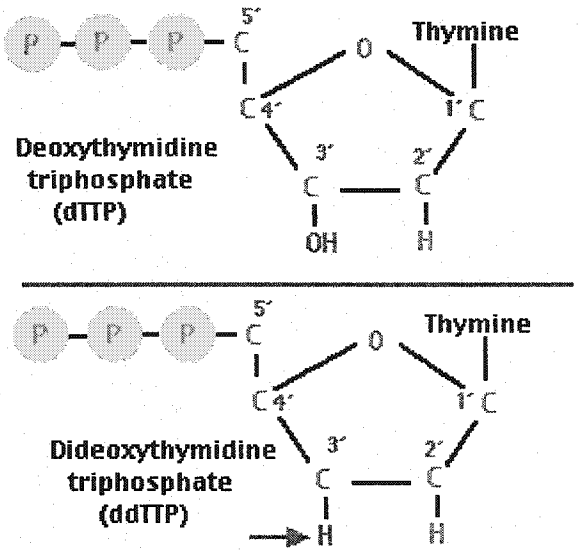


Figure 2: Deoxynuceotide and dideoxynucleotide triphosphate

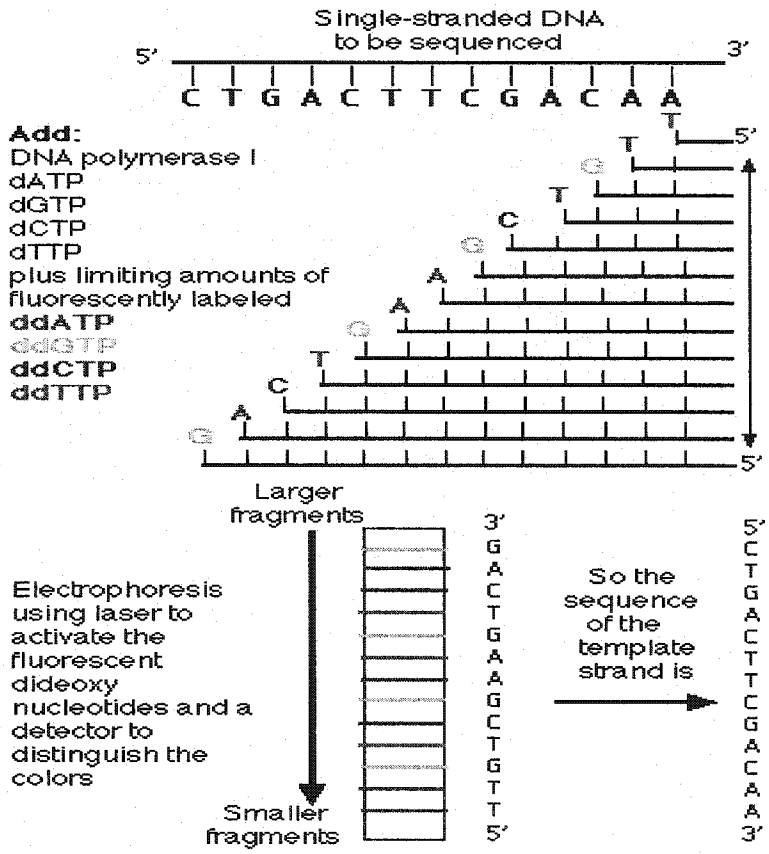


Figure 3: Sequencing diagram

1.2 Existing Quality Control Pipelines

The sequencing quality control pipeline consists of an ordered set of processing stages, each of which does some well-understood operation on its input data, and produces output data that may be used by some subsequent stage. Currently, there are several quality control pipelines designed by major sequencing centers.

The University of Washington designed "Hopper" pipeline (Figure 4). Following the analysis of the DNA sequencing gel files, data are transferred to a directory (hopper_dir) on a UNIX workstation. A central Perl program (Hopper) that sorts the data to project-specific directories is run nightly. Using chromatDump, it reads the chromatogram files in each data directory and creates the necessary directories and subdirectories according to the projects. In addition to data transfer, it runs a series of programs and analyzes data quality (read length estimate, fraction of indeterminate bases, and number of contaminating and repetitive sequences), assembles shotgun sequence data, and generates simple reports describing the results. It includes Phred, Cross-match and Phrap tools. It runs Phred to call bases and determine data quality, cross-match for quality screen and Phrap for sequence assembly work [6].

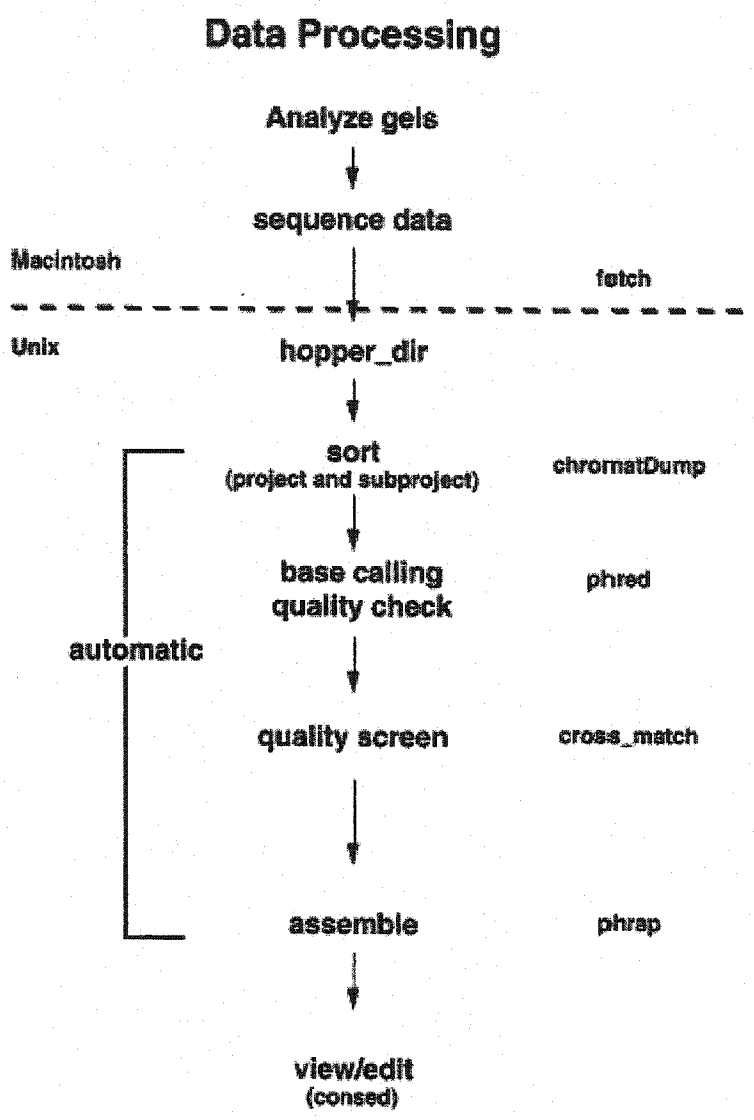


Figure 4: Hopper pipeline

The Institute for Genomic Research (Tigr) developed the Lucy program to do vector, splice site and poor quality trimming work [7] based on FASTA format sequence and quality files, and it applies Cap3 instead of Phrap to do assembly work based on their analysis results [8]. Cap3 has a capability to clip 5' and 3' low-quality regions of reads. It uses base quality values in computation of overlaps between reads, construction of multiple sequence alignments of reads, and generation of consensus sequences. The

program also uses forward-reverse constraints to correct assembly errors and link contigs. The performance of CAP3 was compared to that of CAP3 on four BAC data sets. Phrap often produces longer contigs than CAP3 whereas CAP3 often produces fewer errors in consensus sequences than Phrap. On low-pass data with forward-reverse constraints, it is easier to construct scaffolds with CAP3 than with Phrap. The major assembly steps are shown in the following diagram (Figure 5):

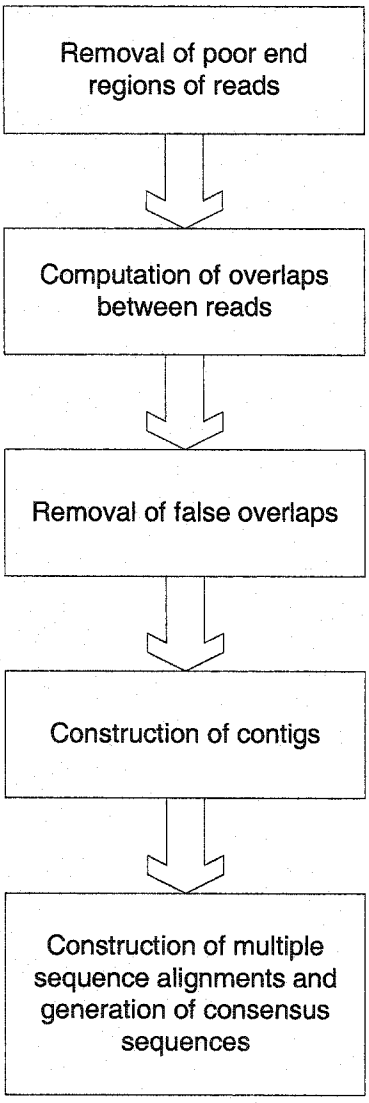


Figure 5: Flow chart of the assembly algorithm

Celera built the assembler consisting of a pipeline of several stages. The sequences flow from one stage to the next. Each stage performs work on its input stream, producing a stream of output messages reflecting its transformational function. In the first step, which is called "Screener", each fragment is checked for matches to known repetitive elements, either noting matched regions, a soft screen, or masking them from further consideration, a hard screen. In the second step, which is called "Overlapper", each fragment was compared with all fragments previously examined in search of overlaps with fewer than 6% differences and involving at least 40 bp of unmasked sequences. In the third step, which is called "Unitigger", collections of fragments whose arrangement is uncontested by overlaps from other fragments were assembled into unitigs. Each unitig was assessed as to whether it represented a unique or a repetitive sequence. Those unitigs which definitely represented unique DNAs were designated U-unitigs. Potential boundaries of repeat sequences were sought at the tips of the U-unitigs, and those found were used to extend U-unitig ends as far as possible into a repeat. In the fourth step, which is called "Scaffolder", all possible U-unitigs with mutually confirming pairs of mates were linked into scaffolds consisting of a set of ordered, oriented contigs for which the size of the intervening gaps is approximately known. When the left and right reads of a mate are in different unitigs, their distance relation orients the two unitigs and provides an estimate of the distance between them. In the final step, which is called "Consensus", reads were multiply aligned according to the consensus metric and consensus base calls were derived in the alignment columns. The quality of each consensus base was computed as the logarithmic-odds of correctness by using the quality values available for each read base [9].

Sanger built its sequencing quality control pipeline including the following major steps [10]:

Sequence pre-processing

Pre-assembly processing or sequence pre-processing is the preparation of the sequence prior to sequence assembly. This involves identification and removal of sequencing artifacts (such as vector sequence, and viral and host contaminants) and tagging of important sequence features. These form part of an important, automated quality control procedure, but also ensure efficient assembly later on. The package ASP encapsulates all these steps. It produces Staden format Experiment files as output. In particular, the following steps are performed on each sequence. 1) Traces are converted to SCF format and base-called using Phred, should they not be in this format already, 2) Quality clipping. The poor regions at the start and end of each sequence are identified, 3) Sequencing vector clipping. Sequence vector at the start and end of each sequence is identified using a number of methods of increasing sensitivity, 4) Cloning vector clipping. Cloning vector is identified using the Staden package vector_clip program. Reads found to be a complete vector are 'Failed'. Cloning vector removal is best performed after sequence assembly when the exact location of the start of the insert can be determined reliably, 5) Contaminant screening. E.coli, Yeast and Phage contaminants are identified by comparing the vector-clipped reads against whole genome databases using BLAST, 6) Feature marking. Features such as Repeat Family members and transposon sequences are identified (using cross_match) and tagged; 7) Traces, experiment files and Phred PHD files are moved to a 'Project' directory. Projects are

organized so that all files with related reads that are to be assembled together are found in the same directory.

DNA Fragment Assembly

We have developed a modular system for sequence assembly and post-assembly processing, based on a textual file format called Common Assembly Format (CAF.) CAF allows for a choice of sequence assembler, post-assembly modules and assembly editor. We routinely assemble using Phrap, and edit using GAP4 (Staden package). Modules vary according to sequencing project, but their function is to prepare the assembly for manual review by a trained finisher. In detail, the following steps are performed on each sequence;

1) Convert experiment files to CAF format;
2) Assemble using Phrap. The fasta and quality files required by Phrap are derived from the CAF file;
3) Adjust clipping for GAP4. Phrap (and Consed) uses quality values so suffers no penalty from using the full length of the reads. The clipping reported by Phrap reflects this. GAP4 requires more stringent clipping;
4) Resolve read discrepancies using auto-editor (Richard Mott.) The auto-editor addresses about 70% of read discrepancies, thus significantly reducing editing time using a '100% edit strategy' (where all reads must be edited to agree) in GAP4;
5) Automatically remove cloning vector. The vector is identified by comparing the assembly consensus against the known vector sequence using cross_match;

6) Sequence features (such as ALU repeats and transposes) are identified and tagged;
7) Reads are selected for re-sequencing to resolve low quality regions and close gaps using FINISH;
8) The consensus is compared with Sanger Institute databases containing finished and unfinished sequences using BLASTN.

National Center for Genome Resources (NCGR) has implemented a comprehensive integrated system for the storage, analysis and visualization of DNA sequences. It was called XGI (X Genome Initiative) [11], and was designed as a solution for high throughput expressed sequence tag (EST) and genomic sequencing operations. It is comprised of analysis pipeline, database, and user interface. The analysis pipeline performs a series of automated operations on sequences submitted to the database [12]. A typical series of operations is:

- Import
- Vector Screen & Quality Control
- Clustering
- Similarity search
- Motif search
- Annotation

The data that serve as input to the initial stage arrive from the sequencing group via FTP to a secure location at NCGR. Sequences are deposited in a popular format called FASTA. For each sequence file, a quality-score file is optionally also deposited, consisting of a space-delimited array of numerical scores. Each score indicates a confidence rating for the identity of the corresponding nucleotide, and is assigned automatically by Phred. It also includes a vector screening procedure, which responses to sequencing quality control. In the vector-screening stage, sequences are first scanned for the presence of vector contamination using BLASTN, which is a member of the BLAST family of algorithms. To screen for vector contamination, the sequences that represent the vector organism, and regions where the two match each other with high confidence are identified and removed. The sequence is then scanned from left to right for the presence of one of the low-quality markers. If the good part of the sequence happens to be 50 or less nucleotides long, this is considered too small to allow for an effective analysis; the processing of that particular sequence is terminated at this point, and a note to that effect can be seen via the Web interface. The vector screen stage also identifies (within the sequence of interest) other synthetic sequences left over from the cloning process, namely the restriction enzyme recognition sites and an adapter sequence that were used to insert the sequence into vector. After being screened, sequences that continue in the pipeline are assured of having an acceptable standard of quality and are free of contaminating vectors.

Operations are custom written programs or third party software (e.g. similarity or motif searches) that communicate with the pipeline in their own XML standard. This architecture allows for rapid addition of new pipeline stages, requiring the software

developer to understand only the XML definition and the operation that they are "wrapping". Meanwhile, the system supports distributed processing for computational-intensive tasks. It uses multi-processor computing to streamline the data processing functions across a collection of networked workstation or on a shared memory multi processor machine. Furthermore, the pipeline software and database schema are flexible and easily customized for more specific analysis needs, which may be requested by the scientific community.

The XGI data model was developed to be highly generic with respect to "operations" on sets of data such as sequences, features and annotations, thus making it simple to comprehend. Simultaneously, it is highly flexible towards the creation of custom pipeline operations as well as adding new analytical methods or operations to the system. The database is carefully organized into tables that make the administration and modification of key pipeline and database functionality easy for the system administrator.

The XGI Web Interface provides a powerful set of search tools to access data and easy to understand graphical displays that summarize analysis results. Results and annotations are also searchable enabling the end user to quickly identify sequences of interest. Primary users of XGI only need access to the Internet to make use of the XGI System. Sequence and analysis data are logically organized and searchable in a variety of ways. Its major futures include:

- Search across multiple species and libraries
- Clustering results are summarized as multiple sequence alignments

- The sequence display captures all information relevant at the individual sequence level, and other information can be accessed through this display
- Analysis results are displayed graphically and are linked directly to Gene Ontology annotations where appropriate
- User can create customized sequence subsets
- Sequences can be saved locally.

1.3 Motivation

DNA sequencing is a multiple process during which large numbers of small template clones are propagated, purified, sequenced and analyzed. In fact, this is a data transformation process, as Figure 6 shows, that aims to report on the quality of sequences produced in a sequencing project; assemble the high-quality fragments; and group contigs to determine a non-redundant set of normalized cDNA. In large genome sequencing centers, the raw DNA data comes directly from automatic sequencing machines and one cannot assume that the bases of sequences are correct. In fact, most raw DNA sequences have base-call errors. Consequently, the output sequence might contain errors, as vector contamination might occur, Dye-terminator reaction might not occur, segment migration might be abnormal in gel electrophoresis etc. Also, the different methods and processes represent different efficiencies and accuracy levels in sequence assembly. All these issues should be controlled before processing the output sequence data to later stages.

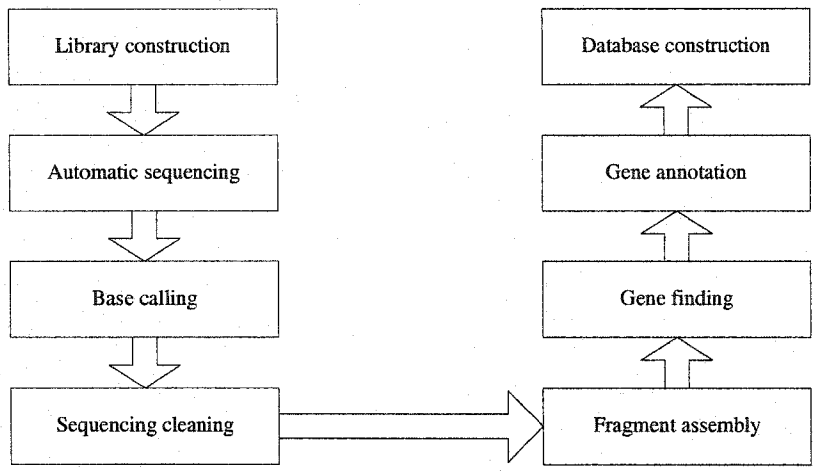


Figure 6: Genomic data processing pipeline

1.4 Method

In order to provide a better solution for the issues mentioned above, we designed a pipeline for quality control as shown in Figure 7. This pipeline includes a set of software applications Phred, Lucy and Phrap. The input of the pipeline is the chromatogram file from the sequencers. Phred is a base-calling program for DNA sequence traces, and it generates FASTA format sequence and quality files. It provides a better accuracy in base calling. Lucy was designed to take the base-call quality assessment of each base into consideration during the trimming process. It compares sequences against some other data, such as vector and contaminant sequences, trims low quality segments and vectors, generates clean range sequence and quality files in order to prepare them for later stages of genomic data processing pipeline [7]. Phrap is a leading program for DNA sequence assembly. It will read the generated clean sequences as input data to assembly contigs, to generate reliable consensus sequences, to estimate consensus quality and to handle the

repeats. The final output is a list of trimmed, full-length, normalized cDNA clones representing the "genes" of the organism. Through these steps, the raw data from sequencing machines can be enhanced to a more reliable level, and assembled more accurately.

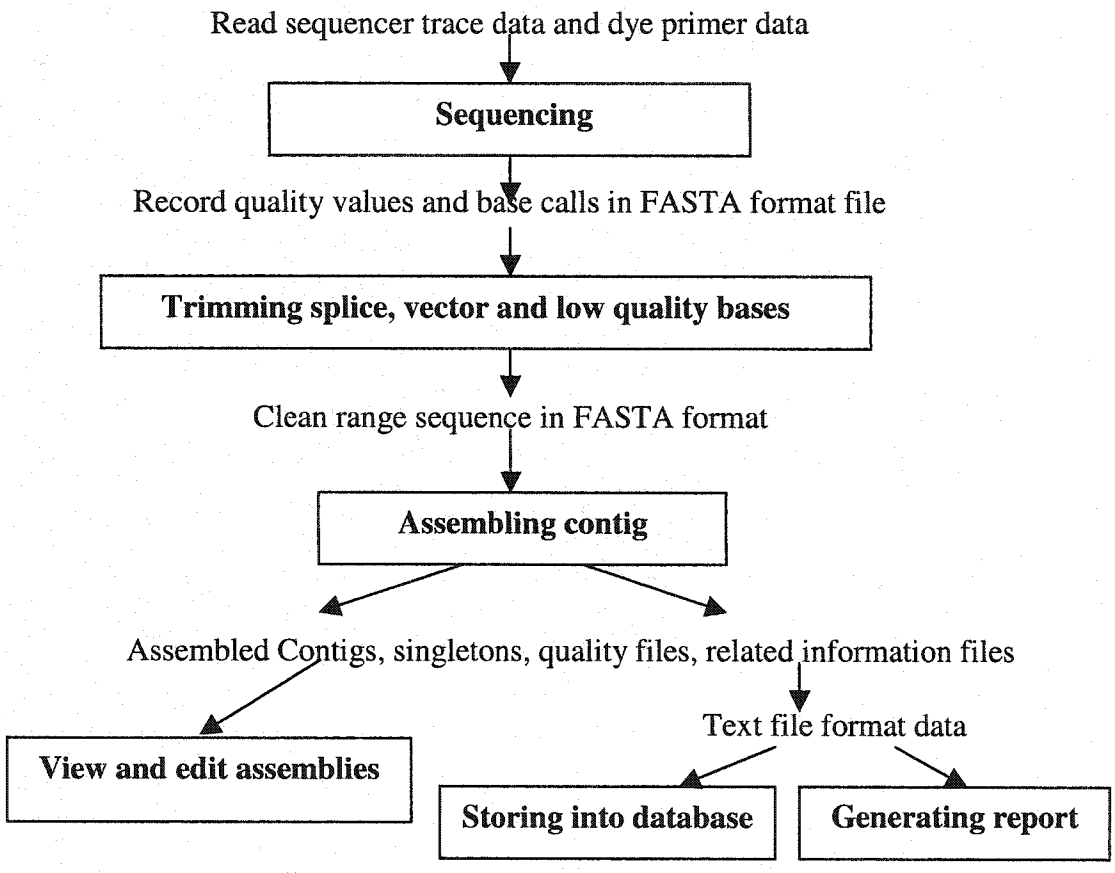


Figure 7: Workflow of pipeline

1.5 Application

This sequencing quality control pipeline is applied within the real fungal genomic project. The aim of the project is to identify novel enzymes in 14 species of fungi that have applications in industrial processes, particularly in the processing of wood fiber into paper. The general procedure of the project is to grow the fungi under a range of conditions of interest, and sample the mRNA present in the organisms under these conditions. From this, we expect to sample 18,000 cDNA clones per species. These clones, allowing for duplicates, should provide representatives of approximately 70,000 genes, of which at least 30% will be new genes. From these genes, by similarity and by gene expression studies, we expect to identify about 130 genes per species. These targets (80%) identified by similarity will be "easy" ones with known relatives within the enzyme families. They are easier to characterize, but have less novelty. Those (20%) identified from gene expression studies will be more difficult to characterize, however they are guaranteed to be novel in some sense. This is a typical genomics project, covering sequencing, sequence analysis and gene expression analysis with microarray. Especially, the sequencing and the assembly procedures will utilize this sequencing quality control pipeline to generate clean, no redundancy sequences and associated contigs [13].

1.6 System Requirements

There is a data server, web server and a computer server running for this project. These are identical Sunfire 280R servers, each dual 900MHz copper Ultrasparc III CPU, 4GB memory, dual 73 GB local disks, talking over gigabit Ethernet to a net work storage device. The network storage device has 1.3 TB and services several research groups and supports teaching: the project storage capacity is about 500GB. We use public domain software for database and web infrastructure (Apache, mySQL, PHP, Perl); applications in Perl and Java; the critical algorithm in C++; data communication in XML; and visualization in Java. We use Phred-Phrap-Consed Package and Lucy as major tools.

Chapter Two

Pipeline Architecture

2.1 Sequencing

At this stage, the pipeline will receive the chromatogram files from the automatic sequencing machine ABI-3700. After executing Phred, it will generate FASTA format sequence and quality file based on reliable base-call and quality value.

2.1.1 Phred Background

Phred reads DNA sequencer trace data, call bases, assigns quality values to the bases, and writes the base calls and quality values to output files. Phred can read trace data from SCF, ABI model 3700 and 377 DNA sequencer chromatograms files, and MegaBACE ESD chromatograms files. Then it makes base-call, and generates base error probabilities. After calling bases, Phred writes the sequences to files in either FASTA or the SCF formats. Quality values for the bases are written to FASTA format files, which can be used by Lucy contamination trimming program to generate the clean range of each sequence, and be used by the phrap sequence assembly program in order to increase the accuracy of the assembled consensus [14].

2.1.2 Motivation of Phred Application

The output of sequencing contains errors; because of anomalous migration of very short fragments and un-reacted dye-primer or dye-terminator molecules, usually the first 50 peaks of the trace are noisy and unevenly spaced; towards the end of the trace, the peaks become progressively less evenly spaced as a result of less accurate trace processing; in better resolved regions of the trace, the most commonly seen electrophoresis anomalies are compressions, these are visible as a peak shifted left of its expected position; weak or variable signal strength and noise peaks are not corresponding to a base. After executing Phred application, these noisy, inaccurate, shifted and weak bases will be assigned low quality values that will be recognized in the later stage. Lucy will trim these low quality sections and Phrap will not take these sections to do assembly.

2.1.3 Base Calling Algorithm

The Phred base-caller uses a four-phase procedure to reconstruct target DNA sequences from fluorescence intensities generated by sequencing machines [15].

In the first phase, idealized peak locations (predicted peaks) are determined; the idea is to use the fact that fragments are relatively evenly spaced to determine the correct number of bases and their idealized evenly spaced locations in regions where the peaks are not well resolved, noisy, or displaced. Peak prediction begins by examining the four trace arrays with Fourier method.

In the second phase, observed peaks are identified in the trace. Observed peaks are found by scanning the four trace arrays for regions that are concave. If this area exceeds 10% of the average area of the preceding 10 accepted observed peaks and 5% of the area of the immediately preceding peak, it is accepted as observed peak; otherwise it is ignored.

In the third phase, observed peaks are matched to the predicted peak location, omitting some peaks and splitting others; as each observed peak comes from a specific array and is thus associated with 1 of the 4 bases, the ordered list of matched observed peaks determines a base sequence for the trace.

In the final phase, the unmatched actual peaks that seem to represent a base but could not be assigned to a predicted peak in the third phase are checked with several conditions, if all conditions are met, an additional predicted peak is created and the observed peak is assigned to it.

2.1.3 Calculating Error Probability Algorithm

Data read from automated sequencers varies significantly in quality for a number of reasons. To use such data efficiently, quantitative knowledge is required. In Phred, the estimated error probability of the base-call is calculated using an empirically calibrated algorithm that considers the following four parameters:

a) Peak spacing (7-peak window), in a window of seven peaks on the current one, the ratio of the largest peak-to-peak spacing to the smallest peak-to-peak spacing, where the minimum possible value is one, corresponding to evenly spaced peaks

b) Uncalled/called ratio (7 peak window), in a window of seven peaks around the current one, the ratio of the amplitude of the largest uncalled peak to the smallest called peak. An uncalled peak is a peak in the signal that was not assigned to a predicted location. If there is no uncalled peak, the largest of the three uncalled trace array values at the location of the called peak is used instead

c) Uncalled/called ratio (3-peak window), same as b, but using a window of three peaks

d) Peak resolution, the number of bases between the current base and nearest unresolved base, times –1, the unresolved base means N, the minimum possible parameter value is half the number of bases in the trace, times –1, and the maximum value is 0.

It allows 50 different thresholds, then each 4-tuple of parameter thresholds (one for each parameter) is considered in turn, and the empirical error rate is computed for the set of the bases defined by those thresholds. The 4-tuple for which the empirical error rate is smallest is selected and this defines the first line of a lookup table. The set of bases defined by these thresholds is then eliminated, and the process is repeated using the remaining bases. Iteration of this procedure produces the desired lookup table.

For example, it is assumed that we have four parameters r, s, t, u and a 4-tuple (i, j, k, m), range between 1 and 50. For each cut (i, j, k m), we define $\mathbf{err}_{(i, j, k, m)}$ to be the total number of erroneous base calls (unmatched base calls and $r(b) << r_i$, $s(b) << s_j$, , $t(b) << t_k$, , $u(b) << u_m$) below the cut. Similarly, $\mathbf{corr}_{(i, j, k, m)}$ is the total number of correct base-calls (matched base calls) below the cut. The formula

$$e_{(i,j,k,m)} = \frac{\left(1.0 + err_{(i,j,k,m)}\right)}{\left(1.0 + corr_{(i,j,k,m)} + err_{(i,j,k,m)}\right)}$$

is used to calculate each base error probability, then the corresponding quality value is:

$$q_{(i,j,k,m)} = -10 \times \log_{10}\left(e_{(i,j,k,m)}\right)$$

The creation of the lookup table will stop if **err** and **corr** are 0 for all remaining cuts. Then it is used to assign quality values to the base-calls from a particular read. Phred computes the four-parameter values, and then searches the lookup table line by line, in order, until it finds a line in which each of the four-parameter values is at least as large as the corresponding parameter value for the base-calls. The quality value associated to that line is then assigned to the base [16].

2.1.4 Samples for Input and Output Data

The following diagrams are samples for input and output data. Figure 8 displays good quality, Figure 9 shows bad quality data.

Input Data: Chromatogram files

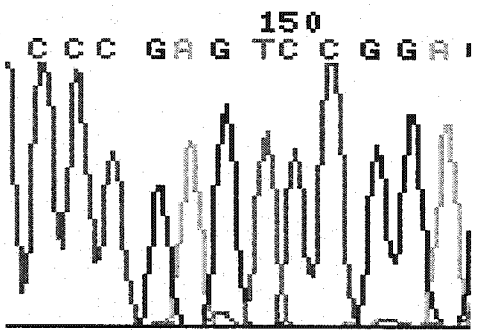


Figure 8: Good quality data

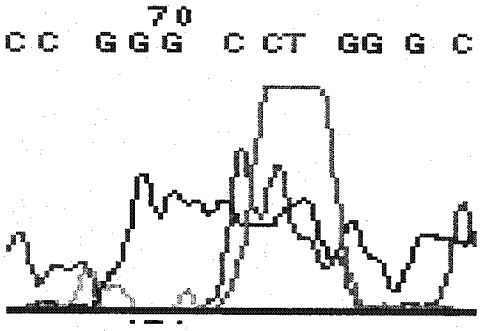


Figure 9: Bad quality data

Output Data: (FASTA format sequence and quality file)

>Asn_00001_MB97_001.ab1.ab1 CHROMAT_FILE: Asn_00001_MB97_001.ab1.ab1 PHD_FILE: Asn_00001_MB97_001.ab1.ab1.phd.1 CHEM: term DYE: big TIME: Fri Nov 15 15:08:17 2002

t a a a t t t g a g a t c c c a g

>Asn_00001_MB97_001.ab1.ab1 PHD_FILE: Asn_00001_MB97_001.ab1.ab1.phd.1

42 42 42 43 46 46 43 51 43 43 43 46 46 42 42 34 36

2.2 Trimming splice-site, vector and low quality sequence

After we received FASTA quality files and sequence files, we applied the Lucy application to trim the vector contamination, splice-site and low quality sequences.

2.2.1 Motivation of Lucy Application

After Phred program, the individual base quality assessment is generally reliable. However, as the output sequence from Phred might have vector contamination, we should assure that the processed sequences are reliable and have the best quality before we assemble these sequences. Although Phred will predict low quality segments, and can trim these segments after using an optional command on command line, it will not trim vector contamination, which can come from many sources, such as E.coli, so it is not the ideal selection for the trimming task. Therefore we find that Lucy can help us to identify the largest subsequence of sufficiently high quality and free of contamination.

2.2.2 Working Process of Lucy Application

Lucy runs through the following working phases as shown in Figure 10:

Phase 1: Counting the number of input sequences in the sequence file, and create internal data structures accordingly

Phase 2: Reading all sequence information of first sequence file, including name, length, and positions. To save memory, Lucy does not load all sequences data into the memory at once. Instead, it uses direct file addressing to access each sequence only when it is needed.

Phase 3: Reading the quality information for the sequence, and computing good quality regions, i.e., regions within sequences that have higher quality values and can be trusted

to be correct. Lucy determines a "clean" range, for which the average probability of error is no greater than the probability specified by the –error option.

Phase 4: Reading the second sequence file, and comparing its sequences to the first sequence file, and extending good quality regions if they both agree. If a second sequence file is not provided, this step is skipped.

Phase 5: Chopping off splice sites from the sequence. Lucy tries to compare all input sequences against splice site sequences in a splice site file which defines the vector sequences near the insertion point on the vector, if the splice site sequences are found on any input sequence, they will excluded from the good quality region so that the sequence assembly program will not mistakenly take them into account when trying to reassemble the sequences.

Phase 6: Removing vector insert sequences. All input sequences are checked against a full-length vector sequence in a vector, and sequences that are vector insert themselves will be detected and removed. Lucy uses a quick fragment match method to check for vector sequences. Both the target vector sequence and the input sequences are converted into fragments (range from 8 to 16 bases long, default is 10), and matching fragments are detected quickly. Vector sequences are detected when they contain more than normal matched vector fragments in their good quality region (already excluded of splice site sequences done previously). The default cutoff threshold is 20%. A sequence that contains over 20% match to a vector will be considered a vector insert and discarded.

Phase 7: Producing output sequences for fragment assembly. Lucy produces the cleaned sequence file with markers for good quality regions, and a companion quality file and a cleavage information file that contains information for the clean range.

In the Figure 10, "CLN" indicates the range after the removal of bad quality data; "CLZ" indicates the range after the removal of zgrasta (old alignment tool); "CLV" indicates the range after clearing of vector fragments; "CLR" indicates range after all low quality parts have been removed

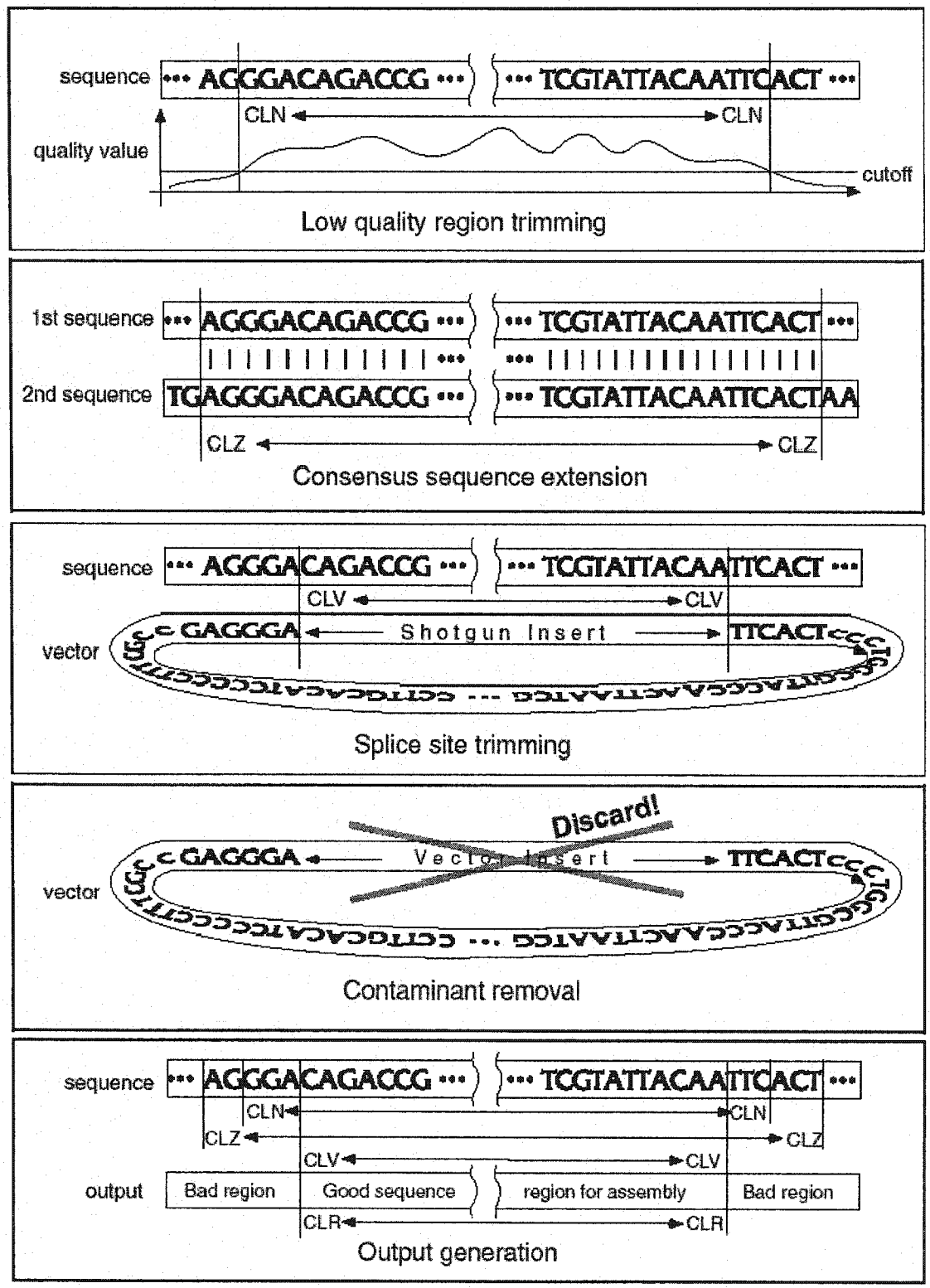


Figure 10: Workflow of Lucy application

2.2.3 Algorithm for generating clean range

Lucy first converts the data from qualities to error probabilities according to the formula:

$$q_{(i,j,k,m)} = -10 \times \log_{10}\left(e_{(i,j,k,m)}\right)$$

(e.g. q = 0 is converted to an error probability of 1.0). All of the major steps in the quality trimming process involve calculating average probabilities of errors within certain windows along the length of the sequence.

a) Trimming of low quality areas from each end

Since the beginning and end of each sequence are typically of low quality, the first step is to remove the lowest quality data from each end. This step is controlled by the *bracket* parameter, including window_size (default value is 10) and max_avg_error (default value is 0.2). At each end checks the window_size bases whether their average probability of error is less than max_avg_error. If not, these bases will remain in the sequences, otherwise, the bases will be excluded from further consideration.

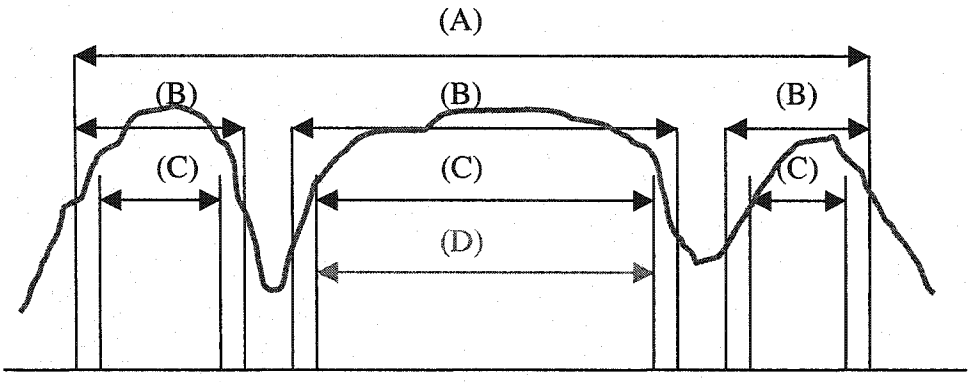
b) Removing poor quality regions

This step is to identify regions of the sequence that have unacceptably high error rates, and exclude those regions from the final clean range. It is controlled by the *window* parameter, which has window_size and max_avg_err parameter pairs. By default, Lucy uses two window sizes in this step: a 50-base window_size1 allowing a max_avg_err1 probability of 0.08, and a 10 base window_size2 allowing a max_avg_err2 probability of 0.3. The purpose of the large window_size1 is to exclude large regions of poor quality, but it may fail to exclude smaller regions of very low quality. The smaller window_size2 excludes those smaller regions. It calculates the average probability of error in each

window of size window_size1. Each of those windows that has an average error of max_avg_err1 or less is added to a new candidate clean range, which will keep growing until some window fails this criterion, and the current candidate clean range is terminated. Lucy continues with the next window, and begins a new candidate clean range with the first subsequent window that passes the same error criterion. Each of the candidate clean ranges produced by considering the largest window size is then subjected to the next window size (window_size2) in the same way, and so on. Each of these steps may produce further fragmentation of the candidate clean ranges. However, the candidate clean range that is smaller than the length specified by the overall *minimum* parameter (the default value is 100) is eliminated from further consideration.

c) *Trimming further to satisfy overall average probability of error criterion and the probability of error criterion at terminal bases*

The final quality-trimming step is controlled by the error option, with max_avg_error and max_error_at_ends parameters. The max_avg_error parameter specifies the maximum overall average probability of an error allowable in the final clean range, which has a default value of 0.025, while the max_error_at_ends parameter specifies the maximum probability of an error for each of the two bases at the ends of the clean range, which has a default value of 0.02. For each of the candidate clean ranges produced by the previous step, Lucy will find the largest subsequence that satisfies both of these criteria, and then it will become the final clean range (see Figure 11) [7].



(A). Low quality areas are trimmed from the each end

(B). Regions of poor quality are removed from the clean range

(C). Clean ranges are further trimmed in order to satisfy the overall average error probability of criterion

(D). The largest remaining candidate as the final clean range

Figure 11: Lucy trimming algorithm

2.2.4 Algorithm for trimming vector splice site

The quality of base calls is usually poor at the beginning of a sequence but gradually improves when moving into the sequence read. Lucy uses the two parameters of *range* and *alignment* to help recognizing vector splice sites. The range parameter consists of area1, area2, area3 with the associated alignment parameters of alignment1, alignment2, alignment3, which are the different alignment strengths for the three areas. An alignment within each area must be equal or longer than these values before it is considered a match

for the splice site. Default values are 8, 12, 16 for the three alignments of the first 40, 60, 100 bases in the range, respectively.

Lucy searches for splice sites in the first 200 DNA bases. If the splice site is not found in the first 200 bases, the next 100 (=area3) bases with a total length of 300 will be checked. Once a splice site is found, the rest of the sequence after the splice site is searched for the other end of the splice site, if any, in order to guard against inserts.

2.2.5 Modification of Lucy Application

1) Lucy comes from a different source as Phred/Phrap. Although it recognizes the Phred output, which is FASTA format sequence and quality file, we have to make some modification to the output of Lucy in order to execute later the Phrap application. The format of the head section of the Lucy output file has to conform to specific format required by Phrap. For example,

Original head of sequence file: ***>Asn_01441_MB121_001.abl***

Modified output will include chromatogram files, phd files and chemistry, Dye and running time information, which are all required by the Phrap application. For example,

>Asn_01441_MB121_001.ab1 CHROMAT_FILE: Asn_01441_MB121_001.ab1 PHD_FILE: Asn_01441_MB121_001.ab1.phd.1 CHEM: Unknown DYE: Unknown ET TIME: Thu Feb 6 16:23:56 2003

2). Although Lucy tracks the number of low quality, short and vector sequences, it does not provide further related information e.g. which sequences are made of those groups,

thus we modified source code to simultaneously also generate name lists for these poor quality sequences in order to store this information in a database.

3). The default Lucy quality file is named "lucy.qul", which is not recognized by Phrap, thus when Phrap takes the sequence file named as "lucy.seq", then it will only read the file named "lucy.seq.qual" file as its associated quality file. So we changed the name of the quality file to "lucy.seq.qual" associated with sequence file "lucy.seq", otherwise Phrap will not recognize it, and Phrap will use the default value 15 for each base.

4). Instead of a full length cleaned sequence – that is, the full sequence with low quality values in a non-clean region - we only take the clean region to do assembly, which is the better way to do the assembly more precisely.

2.2.6 Input/Output of Lucy Application

Lucy needs to read vector and splice site files, and takes a FASTA format sequence file and the associated quality files as input. The output files include a trimmed FASTA format sequence, quality files and a trimming information file, named as "lucy.info", which stores the clean range information for all sequence.

The contamination removal step does not produce trimming tags but sets the trash tag of contaminated sequences, which are represented by CLR left and right, being both 0. The modified Lucy application will generate extra low quality sequences, vector sequences, short sequences and a polyA/T sequences list, if existing, and summary information file which covers summarized data such as the number of total sequences, good sequence as

well as the number of each kind of bad quality sequences. These output data will be stored in the database as metadata.

2.3 Assembly Contigs

2.3.1 Motivation for using Phrap Application

After the previous two steps, we have clean, reliable sequences. However, current technology permits experimentalists to directly determine the sequence of a DNA strand of approximately 500 nucleotides in length. In order to make long sequences, a set of short sequences are taken and subsequently re-assembled. A major task at this stage is to precisely assemble contigs with using these reliable sequences. As a standard tool in DNA sequencing, Phrap is used to locate the overlapping regions within individual sequences and assemble them into longer contiguous sequences (contigs). Phrap is most commonly used for assembling data from shotgun sequencing but can also be used for EST clustering, genotyping, and identifying sequence polymorphism.

2.3.2 Features of Phrap

As a widely used assembly tool, Phrap's key features are:

(1) It allows using of an entire read, not just the highest quality part

(2) It uses a combination of user-supplied and internally computed data quality information to improve accuracy of assembling in the presence of repeats

(3) It constructs the contig sequence as a mosaic of the highest quality parts of the reads rather than a statistically computed "consensus".

(4) It uses quality scores to estimate whether discrepancies between two overlapping sequences are more likely to arise from random errors, or from different copies of a repeated sequence. For repeats with 95 to 98% identity and high quality sequence data, this typically yields correct assemblies.

2.3.3 Work flow of Phrap

Data processing by Phrap consists of the steps below:

1) Finding pairs of reads with matching words. Eliminating exact duplicate reads. Comparing the pairs of reads that have matching words using Smith-Waterman algorithm (swat), computing (complexity adjusted) swat scores. If the alignment score is less than Minscore, the sequence will not be considered matched.

2) Finding probable vector matches and marking them so they aren't used in the assembly process.

3) Finding near duplicate reads, and reads with self-matches.

4) Finding matching read pairs that are "node-rejected" i.e. do not have "solid" matching segments.

5) Using pair-wise matches to identify confirmed parts of reads; using these to compute revised quality values.

6) Computing LLR scores for each match (based on qualities of discrepant and matching bases) (Iterate above two steps).

7) Finding the best alignment for each matching pair of reads that have more than one significant alignment in a given region (highest LLR-scores among several overlapping ones).

8) Identifying probable chimeric and deletion reads (the latter are withheld from assembly).

9) Constructing contig layouts, using consistent pair-wise matches in decreasing score order (greedy algorithm). Layout consistency is checked at a pair-wise comparison level.

10) Constructing contig sequences as mosaics of the highest quality parts of the reads.

11) Aligning reads to contig, tabulating inconsistencies (read / contig discrepancies) and possible sites of misassembly. Adjusting LLR-scores of contig sequence.

LLR scores take into account the qualities of the base calls in the reads. This is a ratio for comparing the hypothesis that reads truly overlap to the hypothesis that they are from 95% similar repeats. Discrepancies between overlapping reads are due to base-calling errors, and will have a tendency to occur in low-quality bases, whereas reads from different repeats can have high-quality discrepancies that are due to sequence differences between the repeats. LLR is calculated by error probability, which generates from quality value of bases, such as:

Case 1 (match), assume the two bases are matched, two probability are calculated separately, the first one is the probability, which is that the pair of bases is really overlapping, expressed by

$$\Pr ob(agree \mid H1(overlap)) = (1-e)(1-f)$$

the second one is the probability, which is that the pair of bases is repeat, expressed

$$\Pr ob(agree \mid H1(repeat)) = (1-e)(1-f) \times 0.95$$

$$LLR = \log \frac{(1-e)(1-f)}{(1-e)(1-f) \times 0.95}$$

Case 2 (discrepancies), assume the bases are discrepancies, two probability are calculated separately, the first one is the probability, which is that the pair of bases is really overlapping, the difference comes from the error, expressed by

$$\Pr ob(agree \mid H1(overlap)) = e + f - ef$$

the second one is the probability, which is that the pair of bases is repeat, expressed by

$$\Pr ob(agree \mid H1(repeat)) = 0.05 + 0.95 \times (e + f - ef)$$

$$LLR = \log \frac{e + f - ef}{0.05 + 0.95 \times (e + f - ef)}$$

In the both cases, e and f represent the error probability of each base of the base pairs.

2.3.4 Smith-Waterman Algorithm

The Smith-Waterman algorithm is a database search algorithm. It implements a technique called dynamic programming, which considers alignments of any length, at any location,

in any sequence, and determines whether an optimal alignment can be found. Based on these calculations, scores or weights are assigned to each character-to-character comparison: positive for exact matches/substitutions, negative for insertions/deletions. In weight matrices, scores are added together and the highest scoring alignment is reported. To put it simply, dynamic programming finds solutions to smaller pieces of the problem and then puts them all together to form a complete and optimal final solution to the entire problem. Instead of looking at each sequence in its entirety this algorithm compares segments of all possible lengths (local alignments) and chooses the one that maximizes the scoring value. The scoring value is calculated by the equation below:

$$H_{ij} = \max\{H_{i-1,j-1} + s(a_i, b_j), \max\{H_{i-k,j} - W_k\}, \max\{H_{i,j-1} - W_l\}, 0\}$$

For every residue in the query sequence, there are four possible ways to form a score:

1) When aligning with the next residue of a db sequence, the score equals the previous score plus the similarity score for the two residues.
2) In case of a deletion (i.e. when matching the residue of a query with a gap), the score equals a previous similarity score minus the gap penalty, which depends on the size of gap.
3) In case of an insertion (i.e. matching a residue of a db sequence with a gap). The score equals the previous score minus the gap penalty that depends on the size of the gap.
4) Stopping when a score becomes zero. The score in each cell is the maximum possible score for an alignment of any length ending at those coordinates.

The pathway is traced back from the highest scoring cell in the matrix, and it can exist anywhere in the array, later aligning the highest scoring segment, then continues until the score falls to 0 (e.g. Figure 12) [17].

		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	5	0	0	0	0	0
W	0	0	0	0	2	0	20	12	4	0	0
H	0	10	2	0	0	0	12	18	22	14	6
E	0	2	16	8	0	0	4	10	18	28	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

Figure 12: Example of Smith-Waterman output

2.3.6 Input/Output of Phrap Application

Input files to Phrap are a FASTA format sequence file associated with a quality file. In addition to these files, Phrap needs to know three things for each read:

(i) The name of the subclone (or other template) from which the read is derived. This is used, for example, when checking for chimeric subclones (for which

purpose it is necessary to know when two reads are from the same subclone so that they are not regarded as independently confirming each other) and for certain other data anomalies.

(ii) The orientation of the read (forward or reverse) within the subclone. In cases where data are acquired from each end of a subclone insert (at present this information is used only for consistency checking following assembly, and not in the assembly itself, but this will change in future versions).

(iii) The chemistry used to generate the read (which influences Phrap's decisions regarding the treatment of discrepancies between potentially overlapping reads, and the adjustment of qualities for reads which confirm each other – confirmation of a read by another read with different chemistry counts with the same significance as confirmation by an opposite-strand read) [14].

Output files except stand output and stand error, also have the following components:

(i) Contig file. This is a FASTA file containing the contig sequences. These include singleton contigs consisting of single reads with a match to some other contig, but where consistent merging was not possible.

(ii) Contig quality file. This has the Phrap-generated quality files for the contig bases.

(iii) Singlets file. A FASTA file containing the singlet reads (i.e. the reads with no match to any other read).

(iv) Log file. This includes various diagnostic information, and a summary of all aspects of the assembly, for trouble shooting purposes.

(v) Ace file (produced when the option –new_ace or –old_ace is used). This file is required for viewing the assembly using Consed.

The standard output file will contain the following major information in the order below:

- Summary information about the run conditions (command line, Phrap version number, and parameter settings) and input data files (sequence, quality and read orientation, chemistry). It is worth reviewing this information for possible problems with the input data.
- Exact and near-exact duplicate reads. Exact duplicates, which probably represent duplicate entries of the same data, are excluded from the assembly. The near-exact duplicates are only listed if they are from different subclones.
- Summary of information about accepted pairwise alignments, including the number of confirmed reads (i.e. reads matching some other read), their average lengths (total, confirmed, "strongly confirmed" (i.e. matching a reverse sense read) and trimmed); a crude estimate of the clone size and depth of coverage; a histogram of depths (letting the depth of a read be the maximum depth that occurs in it).
- Isolated singlets. These are reads having no non-vector match (with scores that are at least min-score) to any other read. Shown for each read are its length, and the number of high quality non-X bases. For most reads, the latter number should be 0, indicating that the read is either entirely vector (its high-quality part has been completely X's out) or is of very low quality; reads for which this value is positive may be contaminant.

- Contigs. The following information appears for each contig: the number of reads; the total length of the contig sequence; the length of the trimmed sequence; and a list of the reads in the contig and information about their alignments. For each read the following information is given: a 'C' if the read is in reverse orientation; the starting and ending positions for the read with respect to the contig sequence; the read name; the score of the alignment of the read against the contig sequence; the mismatch, insertion, and deletion rates for the alignment of the read against the contig sequence; a table showing gaps on each strand, the closest read that could potentially be extended or edited to cover the gap, whether or not the inaccurate, unaligned part of that read already covers the gap, and the length of an accurate extended read that would be required to cover the gap.

2.4 Viewing and Editing Assembly

After assembling, our sequencing procedure arrives to the finishing steps. We need to view the assembly in order to decide whether editing is necessary. Subsequently, editing is required to correct errors in the assembly or in the consensus sequence. Although some of information is recorded in the ace file from the assembly process, they are all in text format, which is hard to read. At this stage we selected *Consed*, as viewing and editing tool. It helps the user to view the assembly output efficiently and easily. It also allows for flexible editing of the consensus.

2.4.1. Input files

Consed requires three main types of data input files:

- Chromatogram files, containing the fluorescence trace profiles
- Phd files, created initially by Phred or another base-calling program and containing the base calls, quality values, and trace peak positions for the read bases, as well as any tags attached to the read
- Ace file, created initially by Phrap or another assembly program and containing assembly information including the contig sequences and quality values, tags attached to the contig sequences, and real alignment information.

2.4.2. Major Functions

The major functions of Consed are described below:

1) Viewing the assemblies after the user has selected a specific contig and double clicked on it. The aligned reads window will display the consensus sequence and the individual read. Different colors are used to indicate base qualities, and discrepancies between contig and read bases.

2) Providing navigation window, which lists information such as low-quality regions in the contig sequences, defined as regions having a high expected number of errors; high-quality read bases that are aligned with the contig sequences but disagree with it and are not tagged as chimeric, vector, or contaminant etc.

3) Tracing individual read and contig sequences

4) Comparing contigs to investigate possible joins not made by the assembly program

5) Obtaining more data, such as chemistry (dye primer or dye terminator), template, and priming site in the template sequence etc.

6) Editing base-calling errors and assembly errors, and create or change tags. Assemly errors are corrected by marking reads in the region of the error with assembler directive tags. After reassembling, contig tags are transferred to the new assembly to generate a map relating the old contigs to the new contigs, and an image of each old tag is created within the appropriate new contigs [18].

Chapter Three

Pipeline Workflow

3.1 Loading Files

Before executing pipeline, the first task is to create directories: a plate directory and an associated subdirectory chromat_dir, phd_dir as well as an edit directory. Thereafter we will be able to load chromatograph files, the vector file and the splice file into the chromat_dir subdirectory. In order to execute Phrap properly, we have to modify default vector files (primerCloneScreen.seq; primerSubcloneScreen.seq; vector.seq) in directory /path/genome/lib/screenLibs.

3.2 Execution of pipeline

In order to run pipeline more flexibly and efficiently, we separate it into three different steps.

Step 1:

The first step will execute the cDNApipeline1_1.pl script file, using the cDNApipeline1_1.pl parameter1 [plateDir], and the algorithm as follows:

1. receive one parameter (plateDir---related plate directory)

2. change to phd_dir subdirectory of this specific directory, to remove existing phd files
3. change to cromat-dir subdirectory, to remove all files except vector and splice and chromatgraph files
4. execute phred and lucy

Step 2:

The only task for step2 is to append two different source data into one destination target, but we need to create a new directory – for combination output, it executes as cDNApipeline1_2.pl parameter1 [destinationDir] parameter2 [selectedPlate], the algorithm as follows:

1. change to workdir directory to open files (include fasta, quality, lucy.seq and lucy.info and lucy.qul etc)
2. change to selected plate to open read source
3. append read source information into associated files in workdir

Recursively execute this step, until all required plates are appended.

Step3:

As this destination directory might become the new source directory for later execution, we need to create a subdirectory chromat_dir so as to match other source directories which all have the subdirectory – chromat_dir. And all existing files should be copied into this subdirectory. Then we execute Phrap and perl scripts files to generate various reports in text file data format in order to load them into database. It executes as cDNApipeline1_3.pl parameter1 [destinationDir] parameter2 [newplateNumber]. The algorithm is as follows:

1. change to workdir directory
2. make subdirectory chromat_dir
3. copy all files into this subdirectory
4. execute scripts files to generate associated report files
5. backup lucy.seq, generate new lucy.seq based on lucy.info and lucy.seq
6. generate lucy.seq.qual (specific name as required) based on lucy.info and lucy.qul
7. generate text file of data from lucy output which will be stored into database
8. execute phrap application, generate phrap output files
9. open existing summary report file, append new information to that file
10. generate text file of data from phrap output which will be stored into database
11. execute script file to generate new sequence and new gene information
12. load data text files into database

3.3 Executed script files and the associated reports

First of all, as we discussed above, modified Lucy application will generate

i) lowquality.info ii) shortseq.info iii) vectorseq.info

Then at step3, we execute various script files in order to get critical information for our database version application, such as,

i) FastaQuality.pl ->Quality.report that lists each read, its length and associated high quality length

ii) Summarize.pl -> Summary.report which lists each read and its clean range and start position and end position, ->$directory_name which includes directory name, total

sequence number, good sequence number, low quality sequence number, vector sequence number, short sequence number and average of good quality bases number (total good quality bases/total sequence number)

iii) GetSeq.pl -> seqfile.info that stores each original sequence name and sequence

iv) Get Qul.pl ->qulfile.info that stores each original sequence name and associated quality values

v) GetPhrapOutPut.pl ->allplates.info, which includes all information in $directory_name report, also adds contig and singlet information

vi) FindClone.pl ->specialplate.info, which includes special plate number, $directory_ name, new clone number and new gene number

vii) GetPoorSeqID.pl ->poorseq.info, based on lowquality.info, shortseq.info and vectorseq.info report to generate summarized report, list sequence name and type of poor sequence.

viii) GetCloneArrayInfo.pl ->contig.info, includes contig number, $directory_name, contig name, contig sequence and contig quality; ->clonearray.info, which includes $directory_name, contig name and associated each clone

ix) GetSingletsInfo.pl -> singlet.info, includes singlet number (which is continuing number from contig number of "contig.info" report), $directory_name and singlet sequence.

3.4 Miscellaneous

1) Calculation of the new clone and new gene

The calculation of new clone and gene number from the new clone template gives us a useful prediction, whether sequencing process is working well. Assume we already have a set of output files for plate1-39 clones, now we will execute pipeline again for new clones from plate40-44. After executing pipeline we get the following information and put them into three data sets: i) array1---plate40-44 clones; ii) array2 --- plate1-44 singlets; iii) array3 --- plate1-44 contigs. Then, if the clones are in both array1 and array2, they are counted for both new gene and new clone, if the contigs of array3 are constituted by the clones only coming from array1, these contigs are counted for new gene and associated clones are counted for new clones.

2) Significant indicator of sequencing quality

Average good sequence length, which is calculated by total good sequence lengths divided by total sequence numbers, is used to predict general sequencing quality.

3) Pipeline major commands

In order to execute pipeline properly, we have to define the following parameter values and use these system commands:

```
$phredPhrap = "modifiedphredPhrapscriptfile";
$lucy = "pathtolucybinaryfile/lucy";
$phrap = "phrapscriptfile";
$lucyoption = " -v $vector $splice $seqfile $qulfile –debug lucy.info";
$phrapoption = " $lucysequencefile new_ace >$outputfile";
system ($phredPhrap);
system ($lucy $lucyoption);
system ($phrap $phrapoption);
```

4) Configuration of working directory and executable script files (Figure13)

The configuration of the working directory directly effects whether the pipeline will be executed successfully. We have three layers containing the home directory and subdirectories with the plate and work directory.

- the first one is the home directory that holds all executable script files.
- The second layer is the plate and work directory. The plate directory has three subdirectories, chromat, phd and edit, the work directory also has a chromat subdirectory and holds all generated FASTA format sequences and quality files, and other output files from Phred, Lucy and Phrap, and various reports generated after executing script files.
- The third layer is the chromat, phd and edit directory. The chromat directory of plate directory includes all chromatograph files, vector files, splice site files and all generated FASTA format sequences and quality files and Lucy output files and report files. The phd directory includes all phd files generated by Phred. The chromat directory of the work directory includes all generated FASTA format sequences and quality files and Lucy output files and report files.

5) Editing of Lucy's splice site file

If we assume all sequences from the input files are read in the same directory, then the splice file is made of two splice site sequences, such as

>5'-3'.forward.begin

gattaagttgggtaacgccagggttttcccagtcacgacgttgtaaacg

>5'-3'.forward.end

acggccagtgccaagcttgcatgcctgcaggtcgactctagagaggatcccc

However, if the input consists of the sequences from both forward and reverse reads of a clone, we should combine the splice site file with both the forward and reverse splice site sequence pairs. For example, the following reverse 3'-5' splice site sequence can be appended to the forward splice sequences above, and must be in order, otherwise trimming will not work properly:

>3'-5'.rev.begin ggggatcctctctagagtcgacctgcaggcatgcaagcttggcactggccgt

>3'-5'.rev.end cgtttacaacgtcgtgactgggaaaaccctggcgttacccaacttaatc

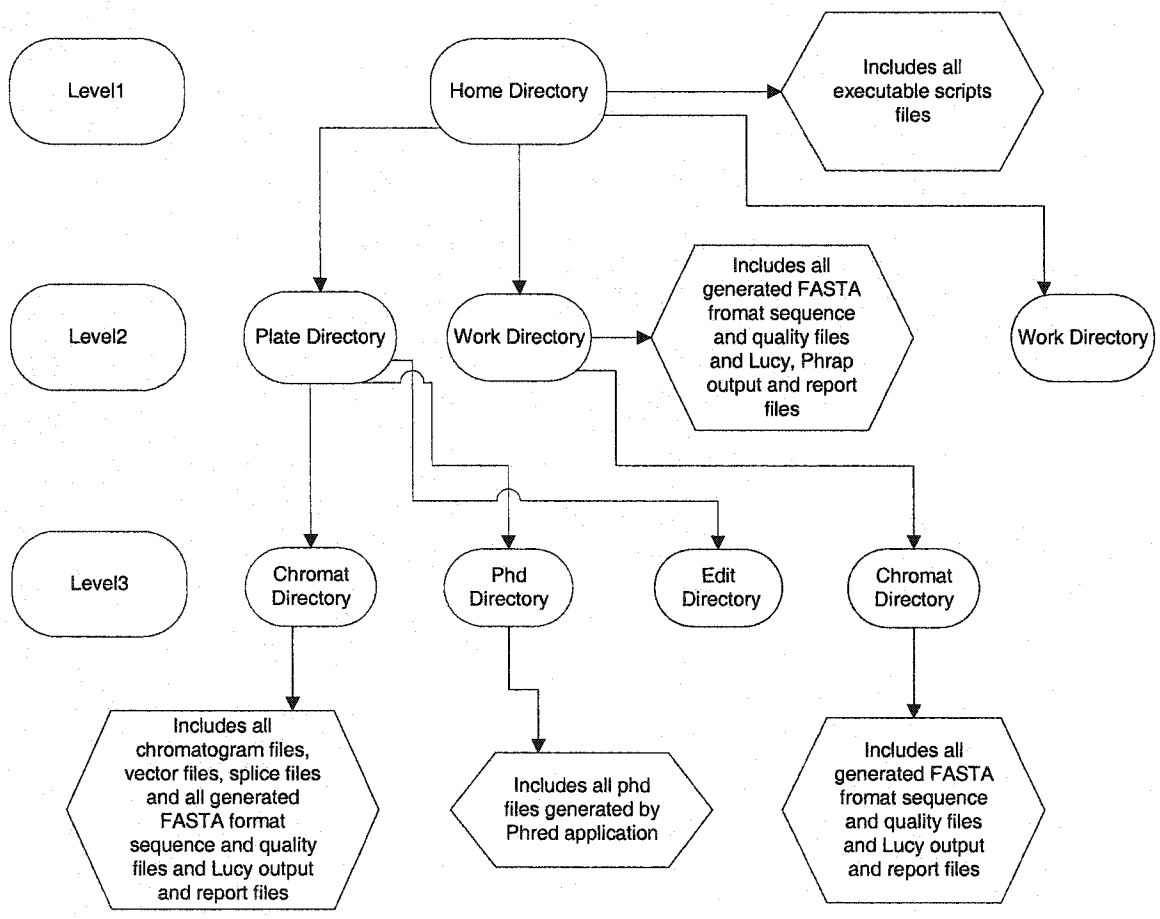


Figure 13: Diagram of directory configurations

Chapter Four

Database Schema of Pipeline

The pipeline collects information that generates during the pipeline process. The information includes input source data, intermediate data, and final outputs. In order to store this information, we build a relational DBMS with open source mySQL. The logical schema is shown in Figure 14.

i) Sequencing feature (sequence_info)

Information related to the sequence, such as creator of the sequence, the date, and the wet lab protocol for generating the raw data and pipeline protocol should be stored in order to evaluate the process later or re-generate the whole process.

Column	Type	Description
sequence_infoID	varchar (20) primary key	sequencing information identification
date	varchar (20)	sequencing date
creator	varchar (20)	person who executes sequencing procedure
protocal	text (2000)	experimental protocal for sequencing procedure

ii) System and Application information (system_application_info)

We should record general information of applied software, such as name, version and associated system information like operating system, database server, web server etc.

Column	Type	Description
system_application_infoID	varchar (20) primary key	system information identification
operating_system	varchar (20)	operating system introduction
web_server	varchar (20)	web server introduction
application1	varchar (20)	applied software introduction
application2	varchar (20)	applied software introduction
application3	varchar (20)	applied software introduction
application4	varchar (20)	applied software introduction

iii) Vector and splice file (vector, splice)

We have to record vector and splice files, as these files will be read by Lucy and Phrap procedure in order to do vector trimming and contamination screening

Column	Type	Description
totalplateNum	varchar (20) foreign key	specify which plate execute with vector
vector_content	text (1000)	vector sequence

Column	Type	Description
totalplateNum	varchar (20)	specify which plate execute with splice
forwardbegin	varchar (200)	5' to 3' begin site of splice
forwardend	varchar (200)	5' to 3' end site of splice
reversebegin	varchar (200)	3' to 5' begin site of splice
reversend	varchar (200)	3' to 5' end site of splice

iv) Raw sequence and quality information (raw_quality, raw_sequence)

Raw quality and sequence information that are generated via Phred application must be stored then further processed by Lucy application to do trimming, in order to get more reliable and accurate sequences, and also can be a reference for data tracing

Column	Type	Description
sequenceID	varchar (20) foreign key	sequence identification
sequence_content	text (2000)	sequence content

Column	Type	Description
sequenceID	varchar (20) foreign key	sequence identification
quality_content	text (4000)	quality content

v) Trimmed sequence information (high_quality_sequence)

After Lucy application, trimmed high quality sequences will be generated, and we have to store trimmed sequence information such as starting and ending position for a clean range, and sequence identification for tracing it back to the original one.

Column	Type	Description
sequenceID	varchar (20) foreign key	trimmed sequence identification
length	int (5)	after trimming, sequence length
CLRstartpos	int (5)	clean range start position
CLRendpos	int (5)	clean range end position
trimmed_sequence_content	text (2000)	sequence content
trimmed_quality_content	text (10000)	quality content

vi) Poor quality sequence information (poor_quality_sequence)

In order to trace the removed sequence later, we got to record these sequence information, such as identification and type (low quality, short sequence, vector)

Column	Type	Description
sequenceID	varchar (20) foreign key	record poor sequence identification
type	varchar (20)	poor sequence type

vii) Phrap parameter (phrapinfo)

Phrap's execution parameter data should be stored, such as min-match, min-score, bandwidth etc, as these are critical conditions, which directly effect the assembly results

Column	Type	Description
totalplateNum	varchar (20) foreign key	system information identification
maxmatch	varchar (20)	maximum match size
max_group_size	varchar (20)	maximum group size
minscore	varchar (20)	minimum score (threshold) to qualify
minmatch	varchar (20)	minimum match size (threshold) to qualify
bandwidth	varchar (20)	bandwidth size
maxgap	varchar (20)	acceptable maximum gap size

viii) Assembled gene information (gene)

Gene information, such as gene name, sequence and quality etc has to be stored, as they will be used in the gene analysis process,

Column	Type	Description
geneID	varchar (20) primary key	gene identification
totalplateNum	varchar (20) foreign key	operating system introduction
contigID	varchar (20)	contig which involved building this gene
singletID	varchar (20)	singlet which involved building this gene
gene_sequence_content	text (4000)	gene sequence content
gene_quality_content	text (10000)	gene quality content

ix) Reads of gene (contig_sequence)

These informations are required when the user analyzes the gene constitution, to observe more precisely the assembly of each read

Column	Type	Description
totalplateNum	varchar (20) foreign key	record plate number for these genes
contigID	varchar (20)	gene identification
sequenceID	varchar (20) foreign key	involved sequence for each gene

x) Summary of pipeline

Summary of pipeline execution must be stored for general viewing of pipeline's output. It covers total sequence number, good quality sequence number, each type of poor quality sequence number, average good sequence length and gene number etc. With the help of these informations, the user is able to have a general image regarding the quality of sequencing.

Column	Type	Description
totalplateNum	varchar (20) primary key	plate number identification
total_seq	int (6)	total sequence number
good_seq	int (5)	good sequence number
low_seq	int (5)	low quality sequence number
short_seq	int (5)	short sequence number
vec_seq	int (5)	vector sequence number
polyAT_seq	int (5)	polyAT sequence number
ave_good_seq_length	int (5)	average good sequence length
singlet	int (5)	singlet number
contig	int (5)	contig number
sequence_infoID	varchar (20) foreign key	sequence info identification
system_application_infoID	varchar (20) foreign key	system appliaction info identification

xi) Clone, template plate information (clone_plate, sequence_plate)

The information such as template number and well position will be used to track the source of the sequence, as the sequences are organized into template plates, which are 96-well micro-titer plates that are the input to the actual plates

Column	Type	Description
cloneID	varchar (20) primary key	source clone identification
TMPnumber	varchar (20)	template number
plate_row	varchar (5)	clone in template well row position
plate_column	varchar (5)	clone in template well column position

Column	Type	Description
sequenceID	varchar (20) primary key	source sequence identification
cloneID	varchar (20) foreign key	source clone identification
DNAplatenum	varchar (20)	DNA plate number
plate_row	varchar (5)	sequence in plate row position
plate_column	varchar 50)	sequence in plate column position

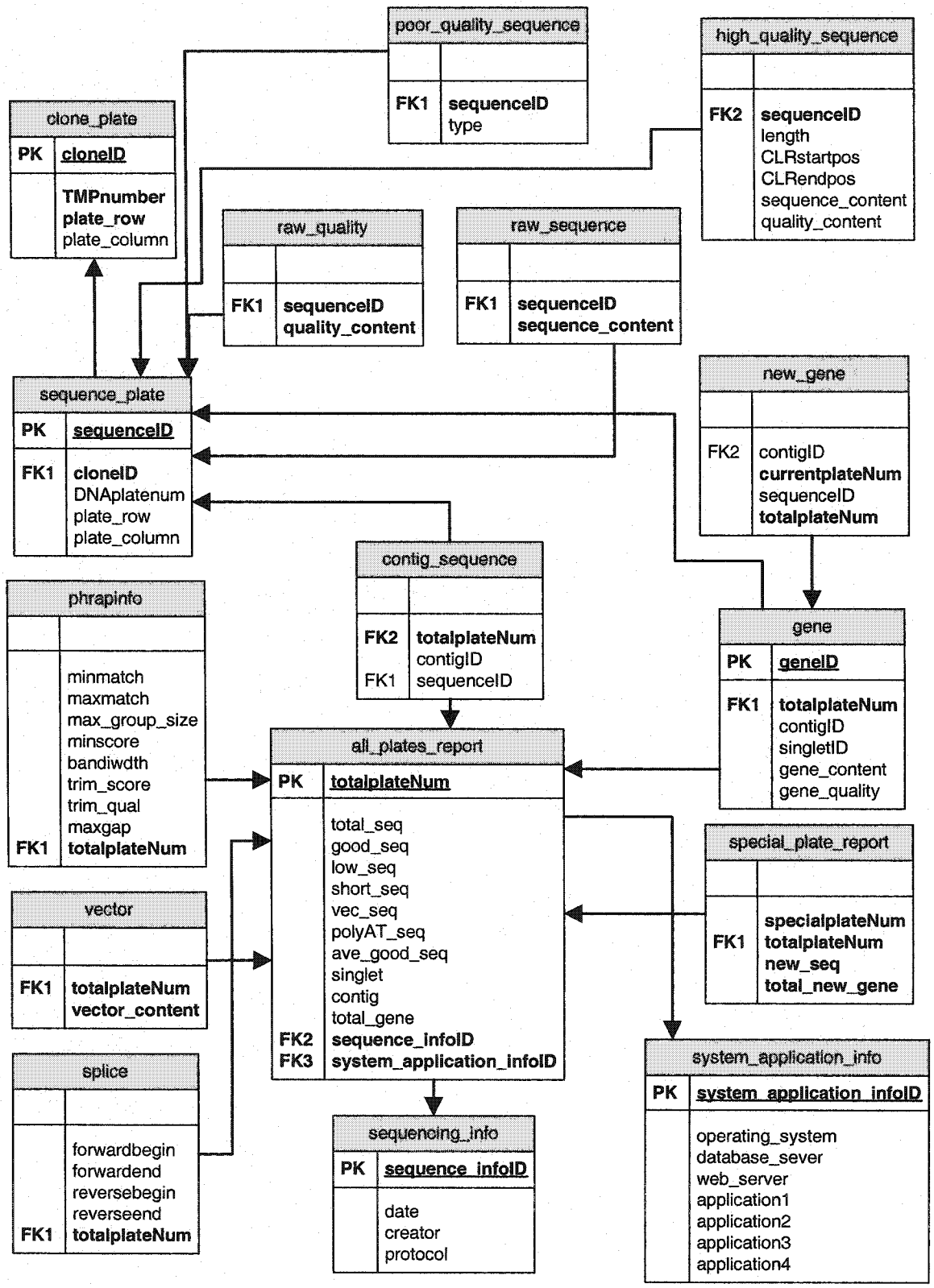


Figure 14: Relational database schema

Chapter Five

Advantages of Applying Lucy

As it is well known, Lucy not only trims low quality sequences, short sequences and vector sequences, it also trims poor quality regions. These trimmed sequences and regions will not be included in the contig assembly. However, how do we know that Lucy provides more accurate sequences and results in more precise assemblies? Should the trimmed sequences and regions also be used to assemble the contigs? In order to demonstrate the advantage of using Lucy before the assembly step, we did a comparison between two different processes, one uses the PhredPhrap Package without Lucy; the other uses Lucy and PhredPhrap package together. From the results, we were able to conclude that the process with Lucy provides more reliable contigs.

5.1 Process of comparison

We executed the two different processes on the same data set of chromatogram files, and obtained two different output data sets. Each output has its own contigs, singlets and ace file, separately. One output data set was selected to generate a database of sequences in FASTA format, and the other was used as a list of query sequences. We use the BLAST

program to compare the query sequences against the database to find the matching pairs of contigs from the two processes. Thereafter we used the BLAST program to compare these contigs against NCBI's nucleotide database. As a result, a list of pairs of contigs is generated. These pairs not only match each other, but also match the same Aspergillus niger gene. The workflow is displayed in Figure 15. We used "Consed"and "ClusterW" to view each of the two assemblies and then compared them manually.

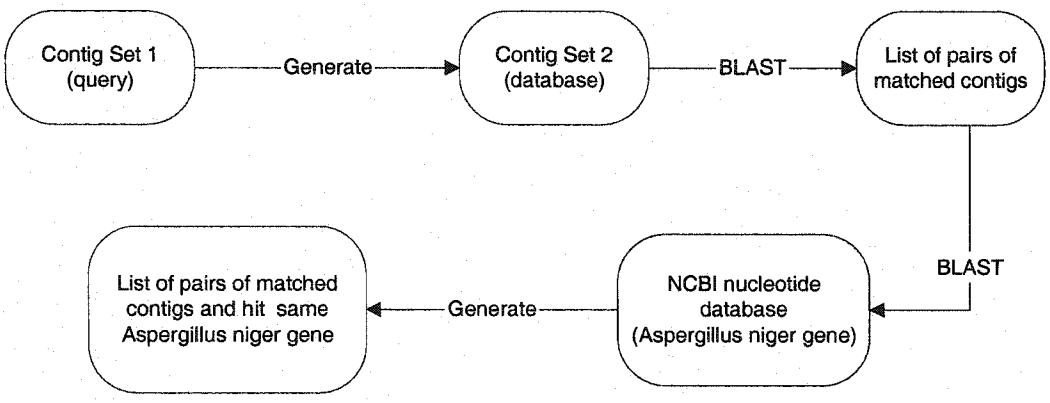


Figure 15: Workflow of comparison

5.2 Results and Analysis

Information on the data sets is shown in the following table. Meanwhile the following data was calculated, a) length of each contig in two processes; b) average of each contig's quality value in each procedure, in order to get a distribution diagram and to do t-test

	PhredPhrap	**PhredLucyPhrap**
Num of sequences in contigs	7338	6556
Trimed sequence number		2126
Contig Number	1668	1635
Singlets	3575	2221

The histogram for the length of individual contigs assembled by the PhredPhrap process is plotted in Figure 16 (A). Similarly, the histogram for the length of individual contigs

assembled by the PhredPhrap and Lucy process is plotted in Figure 16 (B). For convenience, we denote the set of the length of individual contigs assembled by the PhredPhrap process **A** and the set of the length of individual contigs assembled by the PhredPhrap and Lucy process **B**. Using standard two-side t-statistics technique, the absolute *t*-value between A and B is 17.33. This means that the difference between **A** and **B** is obvious, since $17.33 > t_{\frac{\alpha}{2}} = 1.96$ when $\alpha = 0.05$.

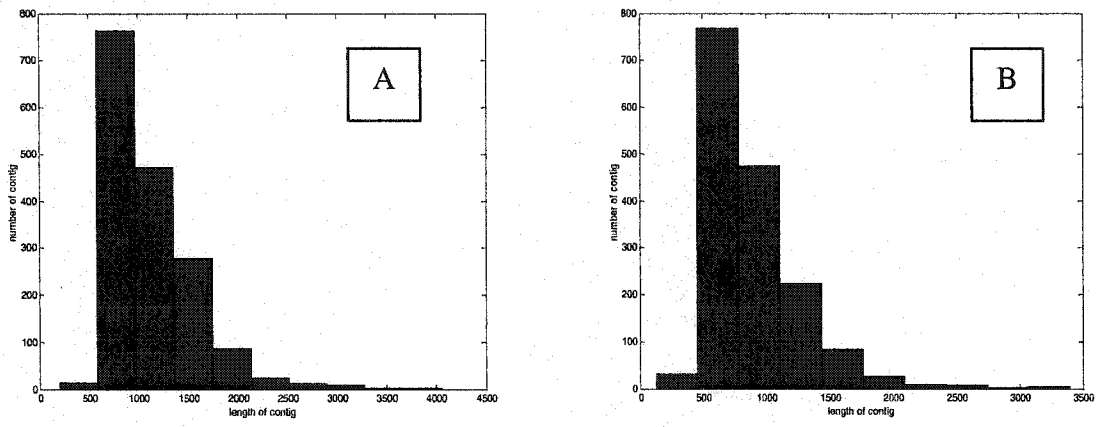


Figure 16: Histograms for the lengths of individual contigs

The histogram for the average quality of individual contigs assembled by the PhredPhrap process is plotted in Figure 17 (C). Similarly, the histogram for the average quality of individual contigs assembled by the PhredPhrap and Lucy process is plotted in Figure 17 (D). For convenience, we denote the set of the average quality of individual contigs assembled by the PhredPhrap process **C** and the set of the average quality of individual contigs assembled by the PhredPhrap and Lucy process **D**. Using standard two-side t-statistics technique, the absolute *t*-value between C and D is 10.87. This means that the difference between **C** and **D** is obvious since $10.87 > t_{\frac{\alpha}{2}} = 1.96$ when $\alpha = 0.05$.

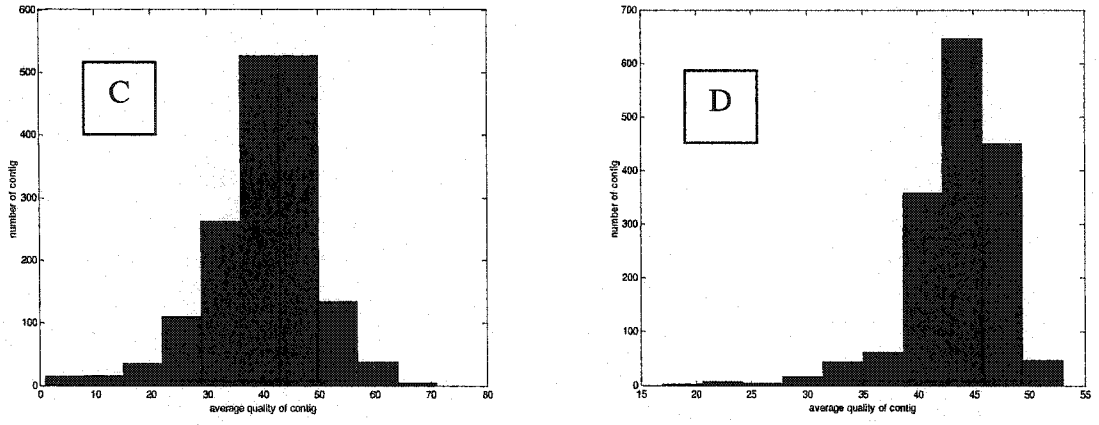


Figure 17: Histogram for the average quality of individual contig

In Figure 18, the red line stands for the distribution of the average contig quality generated after executing the PhredPhrap Process; the blue line stands for distribution of average contig quality generated after executing the PhredPhrap and the Lucy process. We can recognize from the diagram that most average quality values of contigs from PhredPhrap and Lucy process fell into the 60-70 region, however, the average quality values of contigs from Phredphrap process fell into the wider, 50-80 region. This indicates that the process with Lucy assembled more high quality contigs.

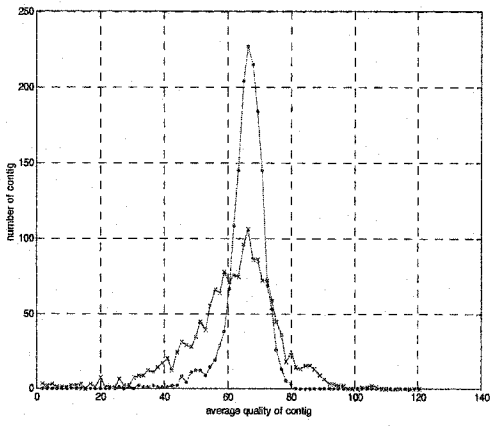


Figure 18: Distribution of average quality of contig

The raw data set contains 10913 sequences. Lucy trimmed 2126 poor quality sequences and the associated process assembled 1635 contigs that included 6556 sequences. However, the other process did not trim any sequences, and assembled 1668 contigs that included 7338 sequences. From the comparison process described above, we got 28 pairs of contigs, which matched each other and matched the same Aspergillus niger gene. Among the 28 pairs, 15 are assemblies of the same sequences and 13 are built from different sequences. In order to decide whether the trimmed sequences affected the contig assembly, we looked at three cases. For the first case, we chose one pair of contigs from the 13 pairs built from different sequences. This first case illustrates a (poor quality) sequence in the contig that is built by the PhredPhrap process (without Lucy) where the sequence does not contribute to the consensus sequence of the contig. For the second case, we used the pair of contigs to illustrate that the regions trimmed by Lucy do not affect the consensus sequence of the contig that is built by the PhredPhrap process (without Lucy). For the third case, we selected a pair of contigs to illustrate that the poor quality regions (trimmed by Lucy) used in the PhredPhrap process affect the consensus sequence of the contig, however, some of these bases within the poor quality region have been substituted by the overlapped high quality bases.

Case 1

In this example, Contig1103 from the PhredLucyPhrap process has 3 sequences involved in contig assembly, and Contig1239 from the PhredPhrap process has 4 sequences. The sequence Asn_10710 was trimmed and discarded in the PhredLucyPhrap process, but it is included in Contig1239. As the following graphics from Consed show, Asn_10710 did

not affect the consensus of the contig. In Figure 19, the top one is the output from PhredPhrap process and the second one is from PhredLucyPhrap process. In both diagrams, the first row is the consensus: the white part indicates that the quality is good; the gray parts are of medium quality; and the dark gray parts are of low quality. On the top one, the last row is the Asn_10710 sequence, where all the bases have a black background, which indicates very low quality. From position 94 to the end of the sequence, it overlaps the other three sequences, and as the other three sequences have higher quality bases (gray background), Phrap uses those bases to construct the consensus sequence.

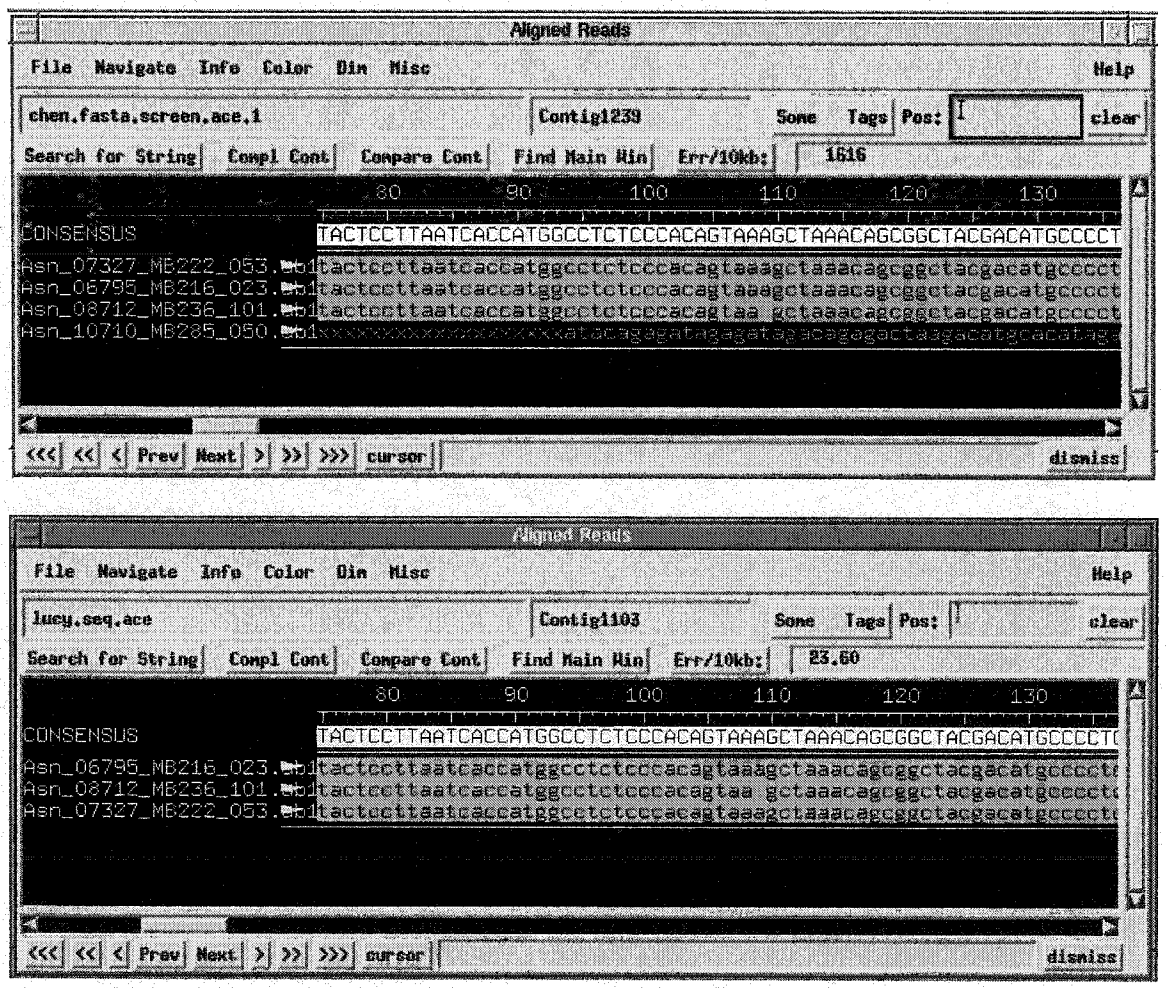


Figure 19: Contig1239 to Contig1103 comparison

Case 2

In this example, Contig927 from the PhredLucyPhrap process has 3 sequences involved in contig assembly, and Contig863 from the PhredPhrap process has 3 sequences. In Figure 20, the top one is the output from PhredPhrap process and the second one is from PhredLucyPhrap process. In the second one, the bases of sequence - Asn_01791 after position 592 have been trimmed, due to their low quality values, these bases are existing in the top one - PhredPhrap process, however, as their low quality values, they do not affect contig assembly.

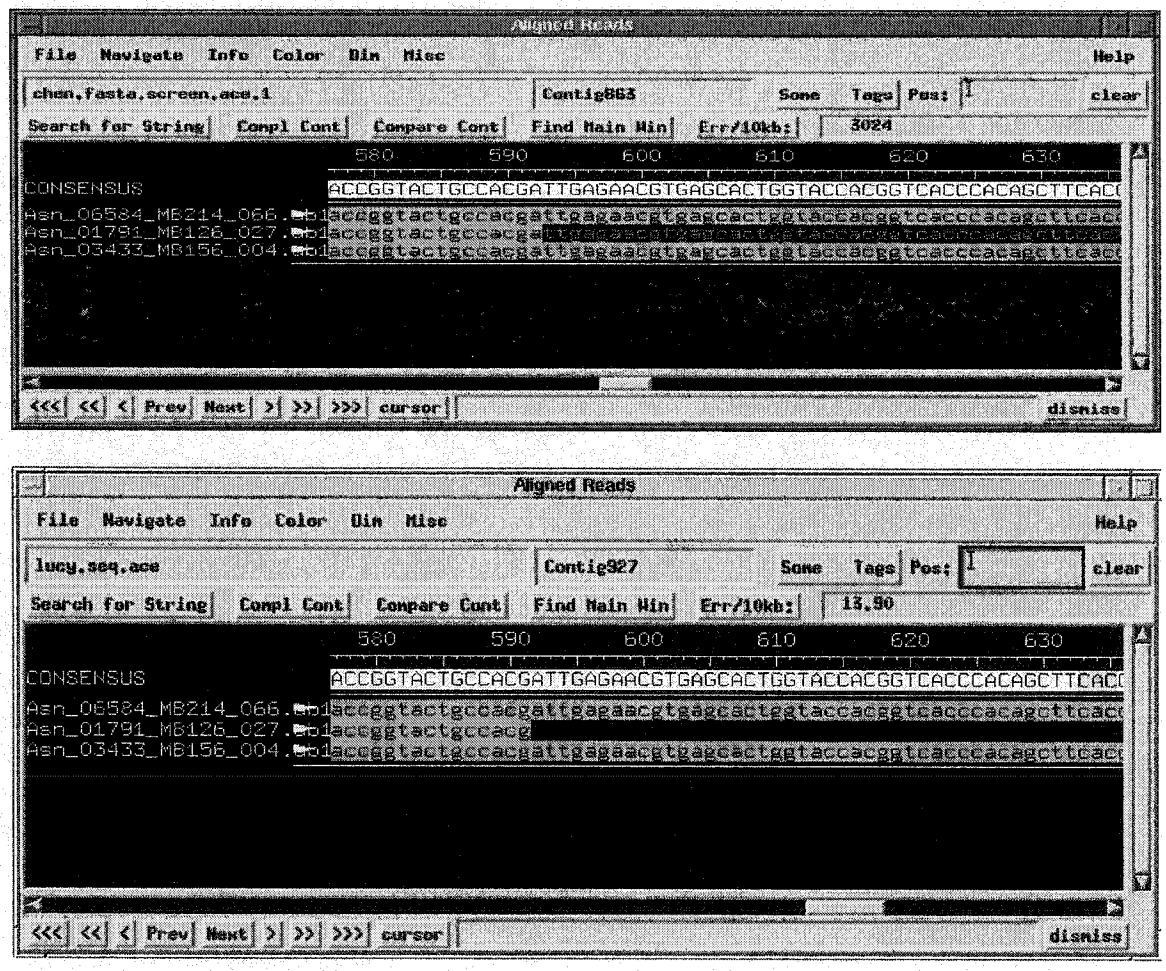


Figure 20: Contig863 to Contig927 comparison

Case 3

In this example, Contig248 from the PhredLucyPhrap process has 2 sequences involved in contig assembly, and Contig239 from the PhredPhrap process has 2 sequences. In Figure 21, the top one is the output from PhredPhrap process and the second one is from PhredLucyPhrap process. In the second one, the bases of sequence - Asn_01978 after position 260 have been trimmed, due to their low quality values, these bases are existing in the top one - PhredPhrap process and they involved in contig assembly. However, as their low quality values, some of them have been replaced by the other bases, which have high quality values.

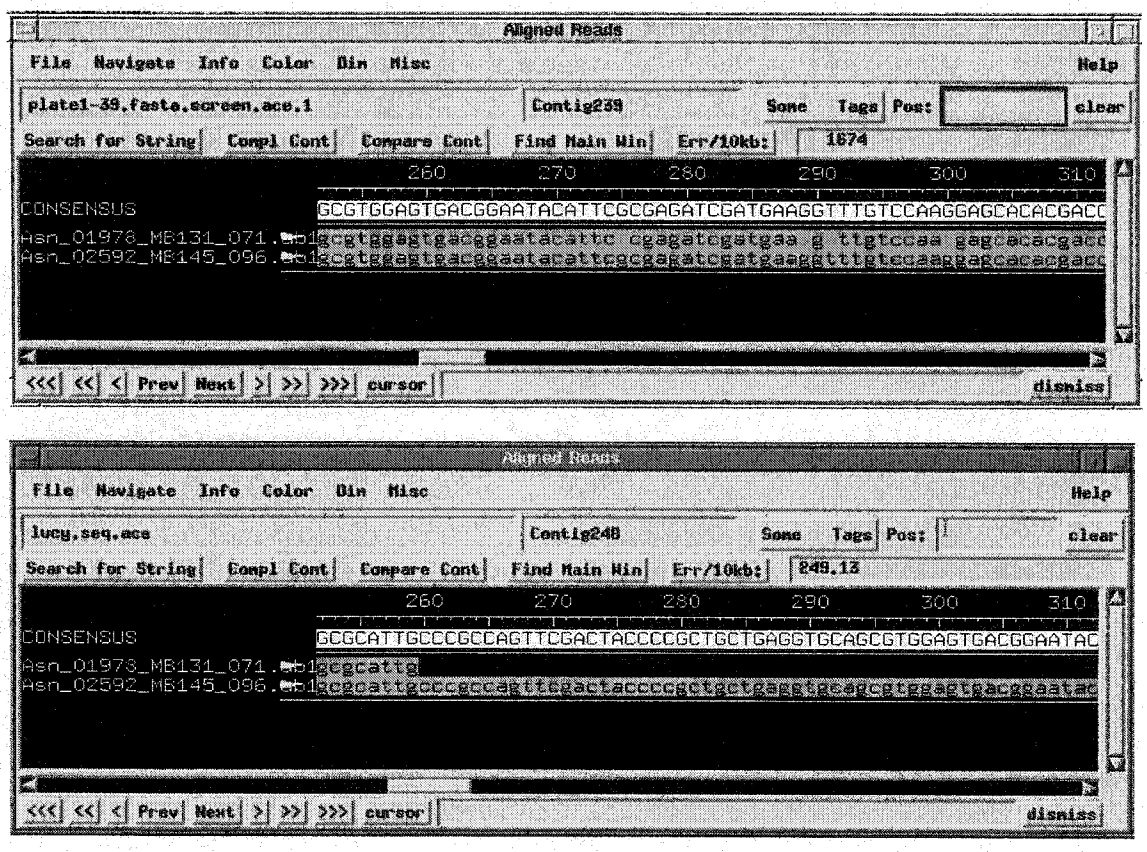


Figure 21: Contig239 to Contig248 comparison

From these examples, we see that the sequences and regions that are trimmed by Lucy are not significantly involved in the contig assembly. Furthermore, the use of Lucy might lead to more efficient assembly process.

On the other side, we considered each gene of Aspergillus niger from the NCBI's nucleotide database and the matching pair of contigs. We used ClustalW to cluster the gene and each pair of consensus sequences of the contigs. We found that the poor quality regions of each sequence result in unreliable consensus sequences of the contigs. The PhredPhrap process generates longer contigs, however the additional parts only have about 70% identity with the Aspergillus niger gene, so there are many substitutions, deletions, insertions and gaps in the alignment. We found that all these bases have low quality, and are regions that are trimmed by Lucy (see Figure 22, where triangle arrows point to unmatched bases, blue square arrows point to gaps). When these bases overlap with other high quality bases, they are ignored. If there is no overlap, then they build low quality regions of the consensus sequence of the contig.

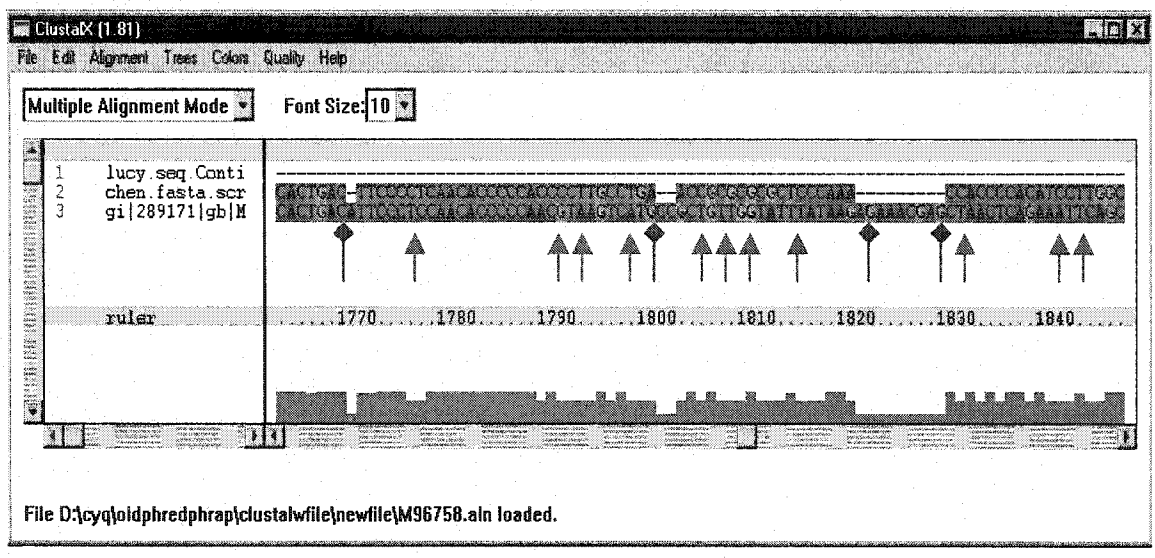


Figure 22: ClustalW output

Chapter Six

Conclusion and Future Work

The pipeline that is described in this thesis achieves the aim of providing quality control for the sequencing stage of cDNA library construction. Phred generates a more reliable base-call for each individual base, and produces quality information about each base-call. This quality value is the associated error probability. Lucy uses the individual base quality values to determine a quality value for regions. Lucy removes vector contamination, and trims low quality regions to provide the longest high quality region as the output sequence. Phrap assembles these high quality sequences into contigs.

We carried out two experiments to compare the effect of using Lucy in the pipeline. The results of the experiments demonstrate that the use of Lucy in the pipeline leads to more precise, more reliable contigs. Even though the PhredPhrap process (without Lucy) makes longer contigs, the regions that contribute the extra length have higher error probability and might be considered unreliable.

The current Phrap algorithm has its limitations. It works on reads (sequences), and their associated quality file. In the process it constructs contig layouts using consistent pair-wise matches in decreasing score order (greedy algorithm), and builds contig sequence as a mosaic of the highest quality parts of the reads. The algorithm assembles a set of reads

into contigs, and is not designed to be used incrementally. In our situation, we have a set of existing contigs, together with a set of new reads, and we wish to determine the combined set of new contigs.

In the future, we can modify the Phrap algorithm to support an incremental approach to contig assembly in order to make the overall process more efficient. Later on, we should consider creating a graphic user interface to let the user control the pipeline steps and to monitor and control the overall process. Adapting Consed to display not only the assembled sequences in a contig, but also related information, would help in determining the root causes of poor quality sequence data.

Chapter Seven

Glossary & Definition

Assembly: Putting sequenced fragments of DNA into their correct chromosomal positions

Base sequence: The order of nucleotide bases in a DNA molecule; determines structure of proteins encoded by that DNA.

Basecalling: Selecting the bases of the sequence from traces to make base sequence

BLAST: A computer program that identifies homologous (similar) genes in different organisms, such as human, fruit fly, or nematode.

cDNA library: A collection of DNA sequences that code for genes. The sequences are generated in the laboratory from mRNA sequences.

Chimera (pl. chimaera): An organism that contains cells or tissues with a different genotype. These can be mutated cells of the host organism or cells from a different organism or species.

Clone: An exact copy made of biological material such as a DNA segment (e.g., a gene or other region), a whole cell, or a complete organism.

Contig: A contiguous region of the genome that has been reconstructed by assembling smaller fragments, the result of joining an overlapping collection of sequences or clones

Cloning vector: DNA molecule originating from a virus, a plasmid, or the cell of a higher organism into which another DNA fragment of appropriate size can be integrated

without loss of the vector's capacity for self-replication; vectors introduce foreign DNA into host cells, where the DNA can be reproduced in large quantities. Examples are plasmids, cosmids, and yeast artificial chromosomes; vectors are often recombinant molecules containing DNA sequences from several sources.

Exon: The protein-coding DNA sequence of a gene.

Expressed sequence tag (EST)**:** A short strand of DNA that is a part of a cDNA molecule and can act as identifier of a gene. Used in locating and mapping genes.

FASTA: compare a protein sequence to another protein sequence or to a protein database, or a DNA sequence to another protein sequence or to a DNA library

Gene: The basic functional unit of heredity located on a chromosome. Genes are blueprints for proteins, which are central to all life-processes.

Homolog: the sequences that come from common ancestor and have similar sequence and structure.

Intron: DNA sequence that interrupts the protein-coding sequence of a gene; an intron is transcribed into RNA but is cut out of the message before it is translated into protein.

LLR score (log-likelihood ratios)**:** is a measure of overlap length and quality. High quality discrepancies that might indicate different copies of a repeat lead to low **LLR** scores

Mutation: An alteration in a gene that can be transmitted from one generation to the next. Although many mutations are associated with defects, some have no effect on health of an organism.

Non-coding region: The part of a gene that does not specify the structure of a protein. Non-coding regions of DNA often contain elements that regulate when a protein will be made, and how much of that protein will be produced.

Raw sequence: individual unassembled sequence reads, produced by sequencing of clones containing DNA inserts

Repeats: Sequences that occur frequently in the genome, usually with small variations.

Sequencing: The process of determining the specific order of nucleotides in a DNA molecule. Sequencing also refers to determining the order of amino acids in a protein.

Traces: The "raw" reads that come from a sequencing machine. It is a graph with four colored plots representing the probability of the four nucleotides at each location (T-Red, C-Blue, G-Black, A-Green).

Chapter Eight

References

[1] J. Hodgson, 2000. *Gene sequencing's Industrial Revolution.* IEEE Spectrum Nov: 36–42.

[2] M.D. Adams, C. Fields, J.C. Ventor, 1994. *Automated DNA sequencing and analysis.* Academic Press, London, San Diego.

[3] P.J. Russell, (1995). *Genetics*. Harpercollins College Publisher, New York.

[4] J.M. Prober, G.L. Trainor, R.J. Dam, F.W. Hobbs, C.W. Roberston, R.J. Zagursky, A.J. Cocuzza, M.A. Jense, and K. Baumeister, 1987. *A system for rapid DNA sequencing with fluorescent chain-terminating dideoxynucleotides*. Science 238: 336-341.

[5] F. Sanger, S. Nicklen, A.R. Coulson, 1977. *DNA sequencing with chain-terminating inhibitors*. Proc. Natl. Acad. Sci. 74: 5463-5467.

[6] T.M. Smith, C. Abajian, L. Hood, 1997. *Hooper: software for automating data tracking and flow in DNA sequencing.* CABIOS Vol. 13 No. 2: 175-182.

[7] H-H. Chou, M.H. Holmes, 2001. *DNA sequence quality trimming and vector removal.* Bioinformatics Vol 17 No 12: 1093-1104.

[8] F. Liang, I. Holt, G. Pertea, S. Karamycheva, J. Quackenbush, 2000. *An optimized protocol for analysis of EST sequences*. Nucleic Acids Research Vol 28, No 18: 3657-3665.

[9] E. Myers, G.G. Sutton, A.L. Delcher, I.M. Dew, D.P. Fasulo, 2000. *A Whole-Genome Assembly of Drosophila.* Science Vol 287: 2196-2204.

[10] The Sanger Institute, 1998. *Overview of Shotgun Sequencing Data Flow* http://www.sanger.ac.uk/Software/sequencing/process.html.

[11] National Center for Genome Resources. http://www.ncgr.org/programs/genomics.

[12] J.T. Inman, H.R. Flores, G.D. May, J.W. Weller, C. J. Bell, 2001. *A high-throughput distributed DNA sequence analysis and database system* IBM System Journal, Vol 40, No 2: 464-486.

[13] G. Butler, 2002. *Bioinformatics for a Fungal Genomics Project.* Concordia University Internal Paper.

[14] P. Green, B. Ewing, 2000. *PHRED Documentation.* **http://www.phrap.org/ phrapdocs/ phrap.html.**

[15] B. Ewing, L. Hillier, M. Wnedl, and P. Green, 1998. *Base-calling of automated sequencer traces using Phred. I. Accuracy assessment.* Genome Research. 8: 175-185.

[16] B. Ewing, P. Green, 1998. *Base-calling of automated sequencer traces using Phred. II. Error Probability.* Genome Research. 8: 186-194.

[17] T. Chen, S. Skiena, 2000. *A Case Study on Genome-Level Fragment Assembly.* Bioinformatics 16: 494-500.

[18] D. Gordon, C. Abajian, P. Green, 1998. *Consed: A Graphical Tool for Sequence Finishing.* Genome Research 8:195-202.