INFORMATION THEORY, GRAPH THEORY AND BAYESIAN STATISTICS BASED IMPROVED AND ROBUST METHODS IN GENOME ASSEMBLY

A Dissertation

by

BILAL WAJID

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Erchin Serpedin
Co-Chair of Committee,	Hazem Nounou
Committee Members,	Aydin Karsilayan
	Byung Jun Yoon
	Mohamed Nounou
Head of Department,	Miroslav Begovic

August 2015

Major Subject: Electrical Engineering

ABSTRACT

Bioinformatics skills required for genome sequencing often represent a significant hurdle for many researchers working in computational biology. This dissertation highlights the significance of genome assembly as a research area, focuses on its need to remain accurate, provides details about the characteristics of the raw data, examines some key metrics, emphasizes some tools and outlines the whole pipeline for next-generation sequencing. Currently, a major effort is being put towards the assembly of the genomes of all living organisms. Given the importance of comparative genome assembly, herein dissertation, the principle of Minimum Description Length (MDL) and its two variants, the Two-Part MDL and Sophisticated MDL, are explored in identifying the optimal reference sequence for genome assembly. Thereafter, a Modular Approach to Reference Assisted Genome Assembly Pipeline, referred to as MARAGAP, is developed. MARAGAP uses the principle of Minimum Description Length (MDL) to determine the optimal reference sequence for the assembly. The optimal reference sequence is used as a template to infer inversions, insertions, deletions and Single Nucleotide Polymorphisms (SNPs) in the target genome. MARAGAP uses an algorithmic approach to detect and correct inversions and deletions, a De-Bruijn graph based approach to infer insertions, an affine-match affine-gap local alignment tool to estimate the locations of insertions and a Bayesian estimation framework for detecting SNPs (called BECA).

BECA effectively capitalizes on the 'alignment-layout-consensus' paradigm and Quality (Q-) values for detecting and correcting SNPs by evaluating a number of probabilistic measures. However, the entire process is conducted once. BECA's framework is further extended by using Gibbs Sampling for further iterations of

BECA. After each assembly the reference sequence is updated and the probabilistic score for each base call renewed. The revised reference sequence and probabilities are then further used to identify the alignments and consensus sequence, thereby, yielding an algorithm referred to as Gibbs-BECA. Gibbs-BECA further improves the performance both in terms of rectifying more SNPs and offering a robust performance even in the presence of a poor reference sequence.

Lastly, another major effort in this dissertation was the development of two cohesive software platforms that combine many different genome assembly pipelines in two distinct environments, referred to as Baari and Genobuntu, respectively. Baari and Genobuntu support pre-assembly tools, genome assemblers as well as post-assembly tools. Additionally, a library of tools developed by the authors for Next Generation Sequencing (NGS) data and commonly used biological software have also been provided in these software platforms. Baari and Genobuntu are free, easily distributable and facilitate building laboratories and software workstations both for personal use as well as for a college/university laboratory. Baari is a customized Ubuntu OS packed with the tools mentioned beforehand whereas Genobuntu is a software package containing the same tools for users who already have Ubuntu OS pre-installed on their systems.

DEDICATION

To my mother, Kaukab Wajid

father, Wajid Abbas Mian

and brothers, Danish and Imran Wajid

whom I love and admire $\cdots$

ACKNOWLEDGMENTS

I would like to thank my committee, TAMU staff, TAMU faculty and friends for making this possible. Special thanks to my family for their continued moral, financial and spiritual support.

NOMENCLATURE

GOLD	Genome OnLine Database
GPL	General Public License
LTS	Long Term Support
MARAGAP	Modular Approach to Reference Assisted Genome Assembly Pipeline
MDL	Minimum Description Length
NIH	National Institute of Health
NGS	Next Generation Sequencing
OS	Operating System
Opt.	Optimal
PCR	Polymerase Chain Reaction
Q-values	Quality values
SE	Sequencing by Expansion
Seq.	Sequence
SH	Sequencing by Hybridization
SL	Sequencing by Ligation
SMR	(Standard Method for choosing an optimal Reference
SMRT	Single Molecule Real Time
SS	Sequencing by Synthesis
SNPs	Single Nucleotide Polymorphisms
SRA	Sequencing Read Archive

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION *

Genome assembly involves taking millions, if not billions, of smaller fragments, called ‘reads’ and assembling them together to form a cohesive unit, called the ‘sequence’. However, simply assembling all the reads into one contiguous sequence, a ‘contig’, is not sufficient. One has to ensure that the assembled sequence does indeed resemble what is truly present in the cell. Some common hurdles are low coverage areas, false positive read-read alignments, false negative alignments, poor sequence quality, polymorphisms and repeated regions of the genome. An even more fundamental concern lies in the difficulty of determining which of the two strands was finally reported in the sequencing procedure. Moreover, as a number of research domains draw suitable conclusions from the sequence itself, a sequence that has not been reported accurately may potentially affect subsequent analyses [122].

Sanger’s deoxydinucleotide sequencing with large and accurate reads opened the door to whole genome sequencing, which deciphered the first human genome in 2001 [94, 114]. Sanger’s approach is still commercially available with improved capillary electrophoresis, enhanced speed and accuracy, and longer read lengths. The National Institute of Health’s (NIH) $1,000 genome project led researchers to develop efficient, economical and high-throughput sequencing platforms called next-generation-sequencing (NGS). For instance, Roche’s 454 GS, Illumina’s MiSeq and HiSeq, ABI’s SOLiD, and Life Technologies’s Ion Torrent and Proton Torrent plat-

*Reprinted with permission from “Do it yourself guide to Genome Assembly,” by Bilal Wajid and Erchin Serpedin, Briefings in Functional Genomics, Aug.’ 14.

Reprinted with permission from “Review of General Algorithmic Features for Genome Assemblers for Next Generation Sequencers,” by Bilal Wajid and Erchin Serpedin, Genomics, Proteomics & Bioinformatics, Elsevier, 2012.

forms all sequence the same genome at a fraction of the time and cost [125].

NGS platforms produce terabytes of data thereby challenging traditional software tools and hardware architectures which were not designed to process such large amounts of data. What is needed parallel to the development of NGS platforms is the development of algorithms and statistical tools with improved memory management and time complexity.

This chapter acts as an introductory note to scientists and researchers working in the area of genome assembly. Section 1.1 provides an overview of NGS platforms, Section 1.2 discusses raw data, including Sequencing Read Archive, Fasta and Fastq file formats. It provides particulars on filtering and correcting raw data in order to determine the 'right-set' of reads. Additionally, the second section enforces the need to report accurate results. Section 1.3 supplies necessary answers addressing the draft assembly process. Section 1.4 reviews common metrics employed to evaluate the assembly and Section 1.5 projects considerations on possible future research trends. Furthermore, to facilitate a better understanding of the research area, the Appendix A presents a 'Step-by-Step Guide to DNA assembly' which provides examples with real data to help reinforce the important concepts.

1.1 Overview of Next Generation Sequencing Platforms

Among NGS platforms, Roche's 454 sequencing is based on Nyren's pyrosequencing approach [3]. This method recognizes the individual bases of the DNA by flagging the bioluminescence produced by the bases as and when they attach themselves onto a primed DNA template. This scheme is referred to as 'sequencing by synthesis' (SS). SS takes one DNA strand as a template and then uses it to synthesize the sequence of its complementary strand. Roche's 454 SS uses four polymerase enzymes to extend several DNA strands in a parallel array. By correctly incorporating the required nucleotide, a pyrophosphate molecule is produced which emits light when triggered [2]. Some characteristics of Roche sequencing include its automated procedures and high speed, while some drawbacks are the lower read accuracy for homopolymer segments of identical bases and relatively high operating costs [74].

In varying degrees of comparison, Illumina differs from Sanger. Sanger's approach uses dideoxynucleotide for irreversible termination of primer extension, whereas Illumina employs reversible terminators for primer extension of the complementary strand. Illumina's 3-O-azidomethy reversible terminators are tagged with four different colored fluorophores to distinguish between the four nucleotides. Therefore, the use of these reversible terminators aids in observing the identity of the nucleotides as they attach onto the DNA fragment because the fluorophores are detected by highly sensitive CCD cameras [7]. Illumina's method significantly reduces the durations of sequencing and assumes a \$1000 price tag for $30\times$ human genome. It's the homopolymeric segments sequencing that show benefits over Roche's pyrosequencing; however, its characteristic short read lengths ($<$300bp) present challenges when resolving short sequence repeats.

In addition to Roche and Illumina, Applied Biosystems's SOLiD sequencer is another key player among genome sequencers. SOLiD uses the principle of 'sequencing-by-ligation' (SL). It differs from Illumina in its method for ligation of octamer oligonucleotides. It uses di-base fluorescent labeled octa-oligonucleotide adaptors which link the template DNA and are bound with 1-μm magnetic beads [102]. At each step, SOLiD's technique encrypts two bases simultaneously and every nucleotide is cross-examined twice: first as the right nucleotide of a pair, and then the left one. This approach reduces homopolymeric sequencing errors. However, similar to Illumina, SOLiD generates short read length data which present complications in sequence assembly.

Collectively, these high throughput sequencers have substantially reduced the cost ($\leq$\$0.1/Mb) and duration of genome sequencing. However, new innovators have appeared. The advent of non-optic, semiconductor-based genome sequencers have shown potential. Manufacturers like Life Technologies developed Ion Proton and Ion PGM, both of which use SS amplification and hydrogen ion sensing semiconductors [91]. The sequence is obtained by sensing hydrogen ions emitted when nucleotides incorporate themselves onto template DNA, a process catalysed by DNA polymerase. Massively parallel transistor-based integrated circuits with about two million wells allow simultaneous detection of multiple reactions. Signal processing tools translate voltage fluctuations into base calls for successive nucleotides.

Another technique which was recently promoted is the single-molecule-real-time (SMRT) sequencing, introduced by HeliScope [72]. SMRT sequencing scheme is free from library preparation or amplification errors. PacBio RS II (by HeliScope) utilizes SMRT sequencing and can produce about 50,000 reads ranging from 15,000 to 40,000 bases in length in just three hours. The extended read length facilitates sequence alignment and improves precision in drafting an assembly, simply because long repetitive DNA fragments can be easily spanned. Together with non-optic semiconductor nanopore technology, SMRT sequencers are referred to as 'third-generation-sequencers' [29]. Overall, the above mentioned high throughput sequencers have substantially reduced the duration and cost of sequencing (\$0.1/Mb).

Companies are investing significant resources to upgrade existing technologies and introduce newer machines. It is hoped that many third-generation-sequencers are expected to surface, coupling SMRT sequencing with principles of quantum physics, electro-thermodynamics and nanopore technology [77, 95, 113]. Existing platforms are currently designed to cater for de novo synthesis, whole genome/exome and transcriptome synthesis, targeted re-sequencing, RNA profiling, ChIP-Seq, mutation detection and metagenomics. Tables 1.1 and 1.2 presents some important details about current sequencers.

1.2 Preliminary Data Processing Steps

Software tools and applications enter the research process once the sequencers fulfill their role of generating reads. The aim of this and the next set of sections is to provide an outline of the individual steps involved in transforming raw data into the novel genome, as presented in Fig. 1.1. The set of interconnected methods are referred to as a 'pipeline'. For a more thorough study, readers are recommended to see the Appendix A, 'Step-by-Step Guide to DNA Sequence Assembly'.

The process starts by using the data generated by one's lab or by downloading the data from the Sequencing Read Archive (SRA) [58]. Data is described in the '.sra' format and must be converted into the .FASTQ file format by employing the SRA toolkit (`http://www.ncbi.nlm.nih.gov/Traces/sra/`). Once converted, the FASTQ format adopts a four line representation to display the sequence and its associated quality:

@ Sequence Identifier

sequence line(s)

+ Sequence Identifier

ASCII encoding of quality values

ASCII characters utilized in the last line of the above mentioned SRA format symbolize Quality values (Q-values). Q-values are log-probabilities illustrating the quality of each base call. For example, for Sanger the formula is:

$$Q_{PHRED} = -10 \times log_{10}(P_e)$$

Table 1.1: Comparison of current sequencing platforms, Key: PCR (polymerase chain reaction), SS (sequencing by synthesis), SL (sequencing by ligation), SH (sequencing-by-hybridization), SE (sequencing by expansion)

Platform	Biochemistry/ biotechnology	Amplification	Throughput	Reads per run	Read Length (bp)	Seq Run Time	Error rate (%)	Machine cost ×1000	Cost per run	Cost per unit data
Sanger (Applied Biosystems 3730xl)	Dideoxynucleotide termination of polymerase chain reaction	PCR	0.06Mb	9600	1000	2 Hrs	0.1	\$100	\$100	\$8,000-\$10,000
454 GS+	Bioluminescence on nucleotide incorporation	Emulsion PCR	∼70Mb	70k∼100k	∼700	18 hours	<1.0	\$125	\$1,000	\$28.50
454 GS FLX+	Bioluminescence on nucleotide incorporation	Emulsion PCR	700Mb	1M	∼1000	23 hours	<1.0	\$500	\$6,000	\$8.50
MiSeq	Cleavage of 3'-O-azidomethy reversible terminator and fluorescent tag on nucleotide incorporation	SS	15Gb	25 M	2W300	5 ∼ 55 hr	0.1	\$125	\$1.4k	\$93/Gb
HiSeq X Ten	Cleavage of 3'-O-azidomethy reversible terminator and fluorescent tag on nucleotide incorporation	SS	1000 Gb	4000M	2W125	7 hr ∼ 6 d	= 0.1	\$1,000	\$12k	\$7/Gb
NextSeq 500	Cleavage of 3'-O-azidomethy reversible terminator and fluorescent tag on nucleotide incorporation	SS	129 Gb	400M	2W150	26∼29 hr	= 0.1	\$250	\$4k	\$33/Gb
SOLiD 5500xl	Ligation of octamer oligonucleotide and cleavage of fluorescent tag	SL	180Gb	2.8B	2W60	150 hr	0.01	\$595	\$10K	\$9/Gb
Ion Proton I	Proton sensing by pH change	SS	10Gb	40∼80 M	200	2∼4 hrs	1.0	\$149	\$1K	100/Gb
Ion PGM 318	Proton sensing by pH change	SS	2Gb	5 M	400	7.3 hrs	1.0	\$52	\$750	350/Gb
Polonator G.007	Cleavage of 3'-ONH2 reversible terminator and fluorescent tag on nucleotide incorporation	SL	10Gb		26		N.A	N.A	N.A	N.A
Helicos HeliScope	Single-molecule real time sequencing	SS	35 Gb	20 M	35	8 Hrs	0.5	\$1,000	10K	\$330/Gb
PacBio RS II	Single-molecule real time sequencing	SS	1 Gb	50,000	15,000 bp	3 Hrs	15	\$700	\$400	∼\$1000/Gb

Table 1.2: Recent sequencing platforms: These platforms are relatively new and to date (Nov 15, 2014) there is not enough information to incorporate them into Table II. * Lasergne, Nabsys and Strato Genomics are working on newer platforms.

Platform	Company	Biotechnology	Resource
GENIUS	GenapSys	Proton sensing by pH and temperature change	`http://genapsys.com/`
NanoTag sequencer	Genia	Electric current change produced by nanotag released from incorporation of nucleotide	`http://geniachip.com/`
GnuBIO platform	GnuBIO system	Oligo hexamers hybridization in microfluidics	`http://gnubio.com/`
*	Lasergen	3'-OH unblocked reversible terminator	`http://lasergen.com/`
*	Nabsys	Hexamer oligonucleotides hybridization mapping through nanopore arrays	`http://nabsys.com/`
MinION and GridION	Oxford Nanopore technologies	Strand DNA or exonuclease cleaved nucleotides pass through nanopores change electric current flow rate	`https://nanoporetech.com/`
*	Strato Genomics Technology	Conversion of DNA into Xpandomer	`http://stratosgenomics.com/`

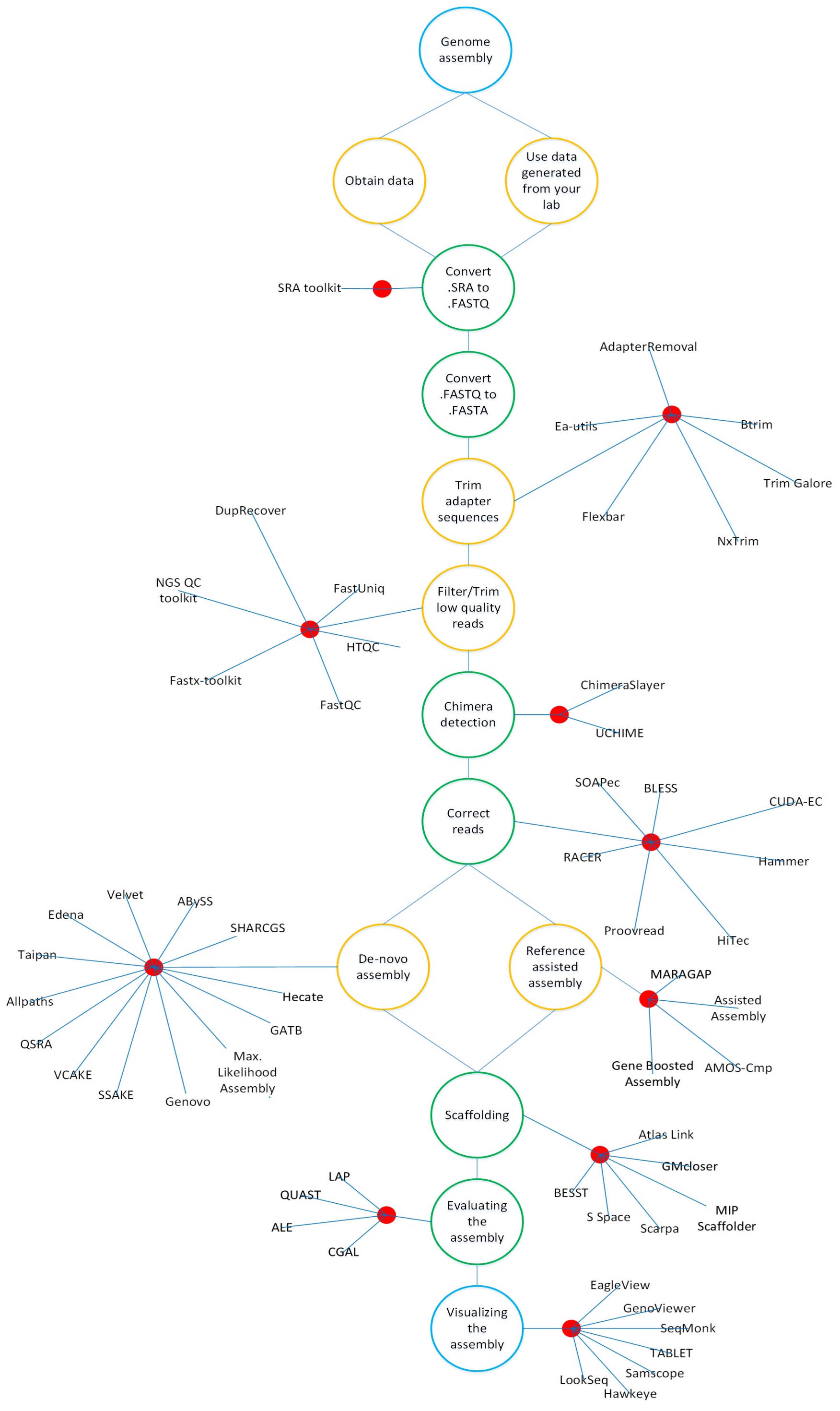


Figure 1.1: Flow chart for DNA assembly pipeline: Some commonly used tools are mentioned next to each step [39].

where P_e is the probability of determining a base incorrectly [118, 122]. For ASCII encoded quality values the following characters depict an increasing order of quality:

!"#$%&'()+,-./0123456789:;<=>?@ABCDEFGHIJKLM*

NOPQRSTUVWXYZ[\ ^ _ 'abcdefghijklmnopqrstuvwxyz{ }

Similar to FASTQ, FASTA format seems like an abridged version of FASTQ file format. It maintains a two-line arrangement to display the sequence and contains no mention of its quality:

@ Sequence Identifier

sequence line(s)

Once reads are received in their correct format, one must trim adapter sequences, filter or trim low quality ends and collapse identical reads. A naive approach is to remove all reads that contain the flag 'N'. An improved method retains all reads that have an overall quality $P_{qual} > q$, where q is a user defined parameter [38, 80]. A more enhanced approach consists in matching reads against known ribosomal and heterochromatin DNA and removing them should they match [75]. Nevertheless, since a significant portion of raw data contains errors one must correct them. Please refer to Appendix A 'Step-by-Step Guide to DNA Sequence Assembly' for a detailed study.

1.3 Assembly Process

The primary aim of the assembly process is to connect all reads together, one after another, to form a single contiguous sequence. Interestingly, due to the inherent nature of the problem, graph theory models very well such a process. In graphical models individual nodes symbolize reads whereas edges between the nodes emphasize ‘overlaps’ between reads. Once the overlap between all reads is established the task at hand is to generate a ‘layout’ by searching for a single path from beginning, i.e., the root of the graph structure to the end, the leaf of the graph structure, as illustrated in Fig. 1.2 and Fig. 1.3. As such, generating a layout is very challenging, because not one, but multiple disjointed graphs are constructed, each depicting a contig. In addition, each graph has many loops portraying repeat regions as well as multiple branches, both long and short. All these hazards need to be resolved. Branches that are small may be discarded, while longer branches compete with one another to serve as potential representatives for each contig. Loops portray repeat regions, so one must decide how many times the repeats should be placed within the final assembly. Nevertheless, assemblers do spend significant amounts of time resolving potential hazards, in multiple ways. Fig. 1.4 demonstrates these processes. The output is a collection of contigs that needs to be ordered, appended and elongated, a process called ‘scaffolding’ [121].

Interested readers are directed to Appendix A.

1.4 Evaluating the Quality of an Assembly

Evaluating the quality of an assembly requires analysing multiple metrics. These statistics measure an assembly from various standpoints. Table 1.3 illustrates some commonly used assembly metrics/statistics and their explanations. After evaluating the assembly one should visualize it as well. Appendix A provides practical examples for evaluating and visualizing an assembly.

1.5 Considerations and Concerns

Genome OnLine Database (GOLD) reports that as of May 2, 2015, 1,037 Archaeal, 44,576 Bacterial and 8,181 Eukaryotic genomes have been sequenced. There remains plenty of room for work. The \$1000 genome project has reduced the cost

Table 1.3: Some common assembly statistics. Here an ↑ indicates higher is better while a ↓ implies less is better.

↑/↓	Description
↑	N50: Quantifies the length of the scaffold at which 50% of the total assembled size of the sequence is covered. NG50: Similar to N50, NG50 quantifies the length of the scaffold at which 50% of the total length of the genome is covered. The length of the sequence is either known or predicted [9]. Continuity: Similar to N50 and NG50 there are other metrics like N75, NG75, N90 and NG90. Number of Genes: an assembly which exhibits more highly-conserved core Eukaryotic genes in the organism is considered better [9]. Accuracy: an assembly is considered accurate provided at least 90% of the bases reported have at least 5× coverage. Choppiness: The average contig length should be greater than a certain threshold. Otherwise, the assembly would be considered to have too many pieces and would need to be redrafted. Validity: the fraction of assembly that can be validated by a reference sequence [9]. Completeness: Should the scaffolds cover more than 90% of the actual genome then the draft assembly is considered complete. Length of the Longest scaffold: typically the greater the length of the largest contig, the better the assembly. Number of scaffolds $> X$, where X is a user-defined length. Similarly, %age of scaffolds $> X$. Total length of the scaffolds and total scaffold length as percentage of estimated genome size: the closer it is to 100%, the better. Percentage of contigs scaffolded: since not all contigs may be connected to form scaffolds [122].
↓	Number of gaps in the assembly: by aligning paired-read data onto scaffolds one may determine scaffolding errors [44]. Number of scaffolds: an assembly which has less number of scaffolds would be assumed better. For example, the optimum assembly would be one continuous sequence depicting the true sequence. LG50 scaffold count: number of scaffolds counted in reaching NG50 threshold. Similarly it would be the case of LG75 and LG90. Percentage of unscaffolded contigs: contigs may remain unscaffolded.

significantly, but if personalized medication is expected to be effective and available to everyone, the cost and time duration for sequencing need to be reduced further. Processing raw data needs to be done both cheaply and at ultra-fast rates. Spending about 50 hours of processing time on a system with 20 microprocessor cores and 20 GB RAM is not uncommon (as of 2014) [9]. Imagine trying to sequence the genomes of an entire country's population. Transferring all the raw-data via an Internet connection from one county to another is not feasible. Therefore, countries will have to provide for their own supercomputers and algorithms will need to be parallelized with careful attention to Hadoop and MapReduce architectures [5, 126, 132]. With so many obstacles ahead, genome assembly will remain challenging for many years to come.

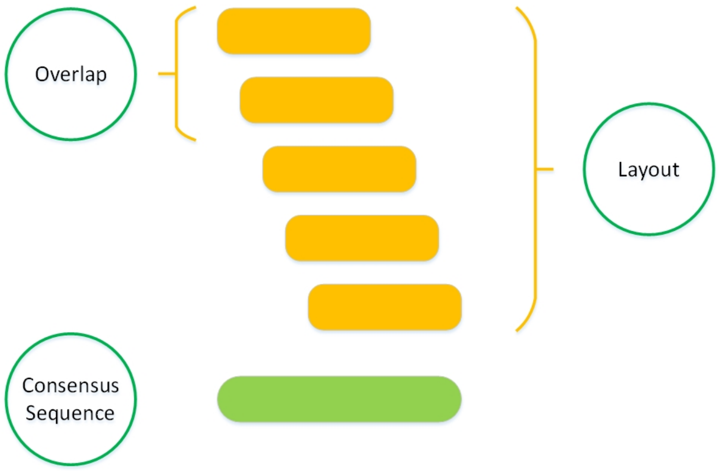


Figure 1.2: Do-novo assembly: Reads that overlap each other are shown to align at appropriate places with respect to one another thereby, generating the layout. The layout, in turn, constructs a consensus sequence, simply by basing itself on the majority base call. The above mentioned framework is called 'Overlap-Layout-Consensus' [121].

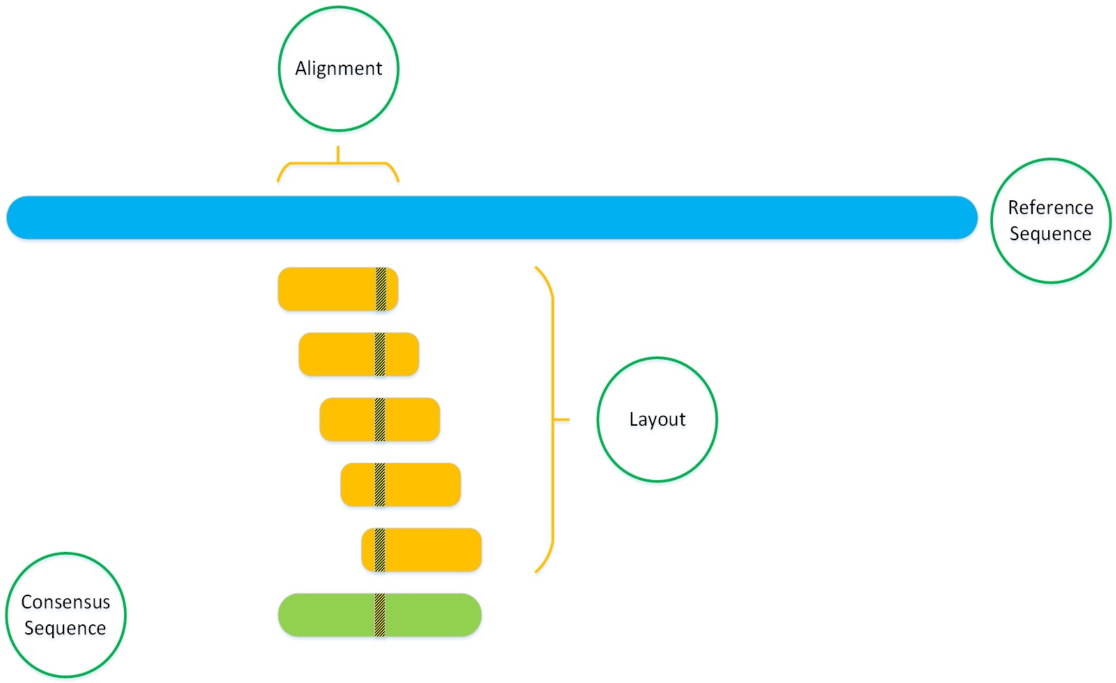


Figure 1.3: Reference assisted assembly: Reads align relative to a reference sequence setting up the layout. The layout, in turn, constructs a consensus sequence, simply by basing itself on the majority base call. Please note that the reads do not need to match perfectly with the reference. The example shows a shaded region where the consensus sequence differs from the reference. This working scheme is called 'Alignment-Layout-Consensus' [121].

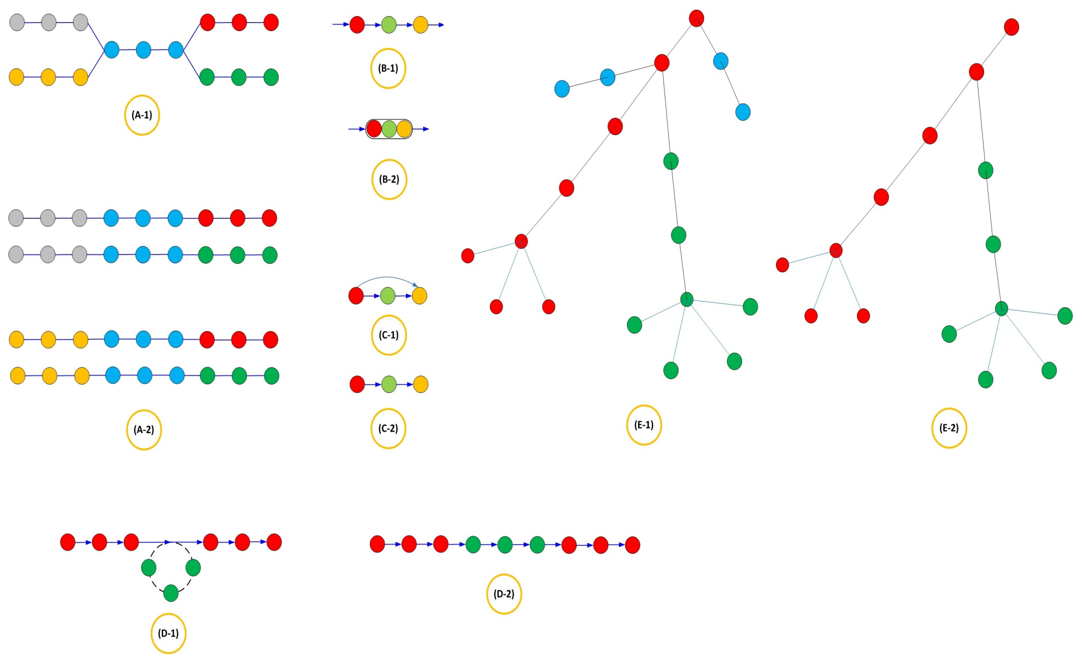


Figure 1.4: Graph simplification techniques: (A-1) Ambiguous paths; (A-2) Pulling apart operation: the resultant graph is divided into four possible paths. (B-1) Simplistic path; (B-2) Removing intermediate nodes: nodes that have an in-degree = out-degree = 1 are collapsed to form one giant node, also referred to as a 'unitig'. (C-1) Unnecessary edges; (C-2) Removing edges: an edge between two nodes is removed if there is an intermediate node between them which connects them simplistically. (D-1) Loop; (D-2) Disambiguation: the loop edge is unrolled and integrated in the continuous edge from left to right. (E-1) Shorter paths shown in blue; (E-2) Removing tips: a tip is defined as a chain of nodes that is disconnected at one end. Tips are removed if they are shorter than t, where t is a user-defined parameter. Furthermore, if there is a longer/common path, it will also trigger a tip's removal.

2. OPTIMAL REFERENCE SEQUENCE SELECTION FOR REFERENCE ASSISTED ASSEMBLY USING MINIMUM DESCRIPTION LENGTH PRINCIPLE [*]

Reference assisted assembly requires the use of a reference sequence, as a model, to assist in the assembly of a novel genome. The standard method for identifying the best reference sequence, henceforth referred to as SMR (Standard Method for choosing the optimal Reference), aims at counting the number of reads of the novel sequence that align to the reference sequence, and then choosing the reference sequence which has the highest number of reads aligning to it. This chapter explores the use of the Minimum Description Length (MDL) principle and its two variants, the Two-Part MDL and the Sophisticated MDL, in identifying the optimal reference sequence for genome assembly. A comparison of the proposed MDL scheme with SMR reveals that simply "counting the number of reads of the novel sequence found within the reference sequence" is not a sufficient criterion. Furthermore, the proposed MDL scheme provides improved results since it not only includes SMR within its framework, but also moves forward by taking the the model as well, i.e., the reference sequence into consideration while, at the same time, deciding the optimal (Opt.) reference sequence.

Rissanen's Minimum Description length (MDL) is an inference tool which op-

[*]Reprinted with permission from "Optimal reference sequence selection for genome assembly using minimum description length principle," by Bilal Wajid, Erchin Serpedin, Mohamed Nounou and Hazem Nounou, EURASIP Journal on Bioinformatics and Systems Biology, 2012(1), 18.

Reprinted with permission from "Minimum Description Length Based Selection of Reference Sequences for Comparative Assemblers," by Bilal Wajid and Erchin Serpedin, 2011 IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS11), Dec. 2011, San Antonio, TX. USA.

erates by learning regular features in the data through data compression. MDL uses “code-length” as a measure to identify the best model among a set of models. The model which compresses the data the most efficaciously and presents the smallest code-length is considered the best model. MDL stems from the principle of Occam’s razor which states that “entities should not be multiplied beyond necessity”, (`http://www.cs.helsinki.fi/group/cosco/Teaching/Information/2009/lectures/lecture5a.pdf`), which stated briefly means “the simplest explanation is the best one” [89, 112]. Therefore, MDL principle tries to find the simplest explanation (model) to match the phenomenon, which is the data.

The MDL principle has been used successfully in inferring the structure of gene regulatory networks [16, 17], compression of DNA sequences [52], gene clustering [47, 107], analysis of genes related to breast cancer [31] and transcription factor binding sites [100].

This chapter is organized as follows. Section 2.1 discusses briefly, the variants of MDL and their application to reference assisted assembly. Section 2.2 explains the algorithm used for the purpose. Section 2.3 elaborates on the simulations carried out to test the proposed scheme. Section 2.4 explains the results, and finally, Section 6.2 summarizes the main features of this chapter.

2.1 Methods

The relevance of MDL to genome assembly can be realized by understanding that genome assembly is an inference problem in which the task at hand is to infer the novel genome from read data obtained from sequencing. Comparative assembly, therefore, is an inference problem which requires identifying a model which best describes the data. It begins the process by identifying a model, the reference sequence most closely related to the set of reads. Comparative assembly then uses the set of reads to build on this model producing one which overfits the data, the novel sequence [120, 121].

MDL presents three variants Two-Part MDL, Sophisticated MDL and MiniMax Regret [89]. The application of these will now be briefly discussed.

2.1.1 Two-Part MDL

Also called old-style MDL, two-part MDL chooses the hypothesis which minimizes the sum of two components:

A) The code-length of the hypothesis.

B) Code-length of the data given the hypothesis.

The two-part MDL selects the hypothesis which minimizes the sum of (A) and (B) above [89].

2.1.2 Sophisticated MDL

The two components of the two-part MDL can be further divided into three components:

A) Encoding the model class: $l(M_i)$, where M_i belongs in model class, and $l(M_i)$ denotes the length of the model class in bits.

B) Encoding the parameters (θ) for any model $M_i: \; l_i(\theta)$.

C) Code-length of the data given the hypothesis is $log_2 \frac{1}{p_{\overline{\theta}}(\mathcal{X})}$.

where $p_{\overline{\theta}}(\mathcal{X})$ denotes the distribution of the Data $\mathcal{X}$ according to the model $\overline{\theta}$. The three part code-length assessment process again can be converted into a two-part code-length assessment by combining steps B and C into the single step B.

A) Encoding the model class: $l(M_i)$, where M_i belongs to any Model class.

B) Code-length of the Data given the hypothesis class $(M_i) = \ l_{(M_i(\mathcal{X}))}$, where $\mathcal{X}$ stands for any data set.

Item (B) above i.e., the 'length of the encoded data given the hypothesis' is also called the stochastic complexity of the model. Furthermore, if the data is fixed, or if item (B) is constant, then the job reduces to minimizing $l(M_i)$, otherwise, reducing part (A) [89, 119].

2.1.3 MiniMax Regret

MiniMax Regret relies on the minimization of the worst case regret [89]:

$$min_M max_{\mathcal{X}} \left[loss(M, \mathcal{X}) - min_{\widehat{M}} loss(\widehat{M}, \mathcal{X}) \right] \tag{2.1}$$

where M can be any model, $\widehat{M}$ represents the best model in the class of all models and $\mathcal{X}$ denotes the data. Regret, $R_{M_i,\mathcal{X}}$, is defined as

$$R_{M_i,\mathcal{X}} = \left[loss(M_i, \mathcal{X}) - min_{\widehat{M}} loss(\widehat{M}, \mathcal{X}) \right] \tag{2.2}$$

Here the loss function, $loss(M_i, \mathcal{X})$, may be defined as the code-length of the data, $\mathcal{X}$, given the model class M_i. The application of Sophisticated MDL in the framework of comparative assembly will be discussed in what follows.

2.1.4 Sophisticated MDL and Genome Assembly

In comparative assembly, a reference sequence is used to assemble a novel genome from a set of reads. Therefore, the best model is the reference sequence most closely related to the novel genome and the data at hand are the set of reads. It should be pointed out that the aim is not to find a general model, rather, the aim is to find a "model that best overfits the data" since there is just one or maybe two instances of the data, based on how many runs of the experiment took place. One "run" signifies that the genome was sequenced once and the data was obtained. The term "model that best over-fits the data" can be explained using the following example.

Assume one has three Reads {X, Y and Z} each having n number of bases. If reference sequences (L) and (M), where (L) = XXYYZZ and (M) = XYZ contain all three reads placed side by side. Since both models contain all the three reads, the stochastic complexity of both (L) and (M) is the same and both "over-fit" the data perfectly. However, since (M) is shorter than (L), (M) is the model of choice precisely because the model "best" over-fits the data.

To formalize the MDL process, the first step would be to identify the following considerations:

To formalize the MDL process, the first step would be to identify the following considerations:

A) Encoding the model class: $l(M_i)$, where M_i belongs to Model classes.

B) Encoding the parameters (θ) of the Model M_i : $l_i(\theta)$.

C) Code-length of the data given the hypothesis is $log_2 \frac{1}{p_{\overline{\theta}}(\mathcal{D})}$.

The model class in comparative assembly would be the reference (Ref.) sequence itself. The parameters of the model θ, are such that, $\theta \in$ {-1, 0, 1}. In the process of encoding, the model class regions of the genome that are covered by the reads of the unassembled genome are flagged with "1"(s). Areas of the Ref. genome not covered by the reads are flagged as "0"(s), whereas areas of the Ref. genome that are inverted in the novel genome are marked with "-1"(s). In the end, every base of the Ref. sequence is flagged with {-1, 0, 1}. Therefore, the code-length of the parameters of the model is proportional to the length of the sequence.

Table 2.1: Counting Number of Reads not enough. The table shows that SMR is not sufficient. Simply looking at "Data given the model" ≡ "Number of reads found" one ends up choosing Human Chromosome 21. However, as Chromosome 21 is about 9 times larger than S85 one concludes that S85 is the model of choice. Furthermore, S85 is a bacterial genome whereas Chromosome 21 comes from a eukaryote genome. PAb1 is also a bacteria, therefore, S85 is definitely the model of choice.

S. No.	Reference Sequence	Number of bases in genomes	Number of Reads found
1	Fibrobacter succinogenes subsp. succinogenes S85 (NC_013410.1)	3842635	157
2	Human Chromosome 21 (AC_000044.1)	32992206	158

In SMR data given the hypothesis is typically defined as "Number of reads that align to the Ref. sequence". In the case presented below "data given the hypothesis" is defined in an inverted fashion as the "Number of reads that do not align to the reference sequence". These two are interchangeable as the "Total number of reads" is the sum total of the "number of reads that aligned to the Ref." and the "number of reads that do not align to the Ref.".

Table 2.1 shows that choosing the reference sequence having the highest number of reads present is not a sufficient condition for selecting the optimal reference sequence. The simulation carried out compared two reference sequences Fibrobacter succinogenes S85 (NC_013410.1) [65], and Human Chromosome 21 (AC_000044.1) [46], with the reads of Pseudomonas aeruginosa PAb1 (SRX000424) [119]. It shows that in order to choose the optimal reference sequence one has to take into account both the "Code-length of the model" and "Number of reads found" to be the sufficient conditions for choosing the optimal reference sequence. This is a crucial insight.

Therefore, a simple, novel scheme is proposed for the solution to the problem,

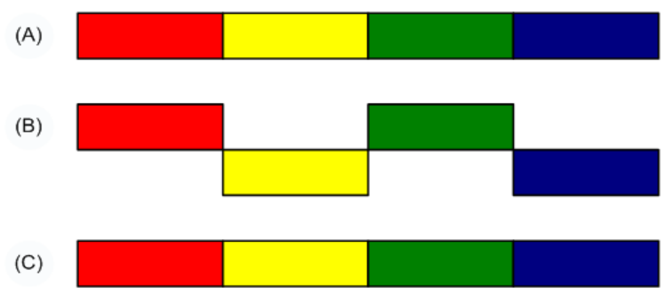


Figure 2.1: Correcting inversions in the Reference Sequence. (A) Reads are derived from the novel sequence. (B) The reference sequence, S_R, contains two inversions, shown as yellow and blue regions. (C) The sequence generated Θ has both yellow and blue regions rectified. Notice that using a simple ad-hoc scheme of counting the number of reads in the reference sequence one would have made use of (B) for assembly of novel genome. However, using MDL one can now use (C) for the assembly of the novel genome.

see Table 2.2. The proposed solution follows the "three assessment" process of Sophisticated MDL. The MDL-based idea stores the model class (Ref. sequence), the parameters of the model (where each base of the sequence is flagged with {-1, 0, 1}) and the data given the hypothesis (reads of the novel genome that do not align to the Ref. sequence) is one file. The file is then encoded using either Huffman Coding [43] or Shannon-Fano coding [32] to determine code-length. For simplistic, three-bits-per-character coding, the Code-length will equal ($Length_{\text{(Ref. Seq.)}} \times 3$) + ($Length_{\text{(Parameters of the Model)}} \times 3$) + ($Length_{\text{(Read)}} \times 3\times$ No. of Unique Reads which did not align). The proposed scheme not only allows the assembler to determine the best model, among a pool of models to choose from, but also improves the model making it better suited according to the novel genome to be assembled. This is done by identifying all deletions and inversions larger than one read length. It then rectifies those deletions and inversions to get a better model, better suited to assemble the novel genome compared to what it started from (see Figures 2.1 and 2.2).

Table 2.2: Summary of the experiment using three reads {ATAT, GGGG, CCAA} and three reference sequences {1, 2, 3}. Regret is defined as $R_{M_i,\mathcal{X}} = \Big[loss(M_i,\mathcal{X}) - min_{\widehat{M}} loss(\widehat{M},\mathcal{X})\Big]$. Here the loss function, $loss(M_i,\mathcal{X})$, happens to be code-length of the data $\mathcal{X}$, given the model class M_i. Whereas, "Data given the hypothesis", is the code-length of the "Reads that do not align to the reference sequence". The code-length in the last column is measured according to Eq. **??**. The experiment shows that given the MDL proposed scheme Ref. 1 is the optimal choice for a reference sequence.

S.No	Ref. Seq.	Model given by the Data	Reads that do not align to the Reference Sequence	Data given the hypothesis (Bits)	*Regret*	Proposed Scheme	Code-length (Bits)
1	ATATCGGGGCTATA	1111011110-1-1-1-1	CCAA	12	0	ATATCGGGGCATAT>1111 0 1111 0 -1-1-1-1>CCAA	102
2	ATGGGCCCTTATTGC	000000000000000	ATAT > GGGG > CCAA	42	30	ATGGGCCCTTATTGC> 000000000000000 >ATAT>GGGG >CCAA	138
3	GGGGCCCCGGGG	1111-1-1-1-11111	ATAT > CCAA	27	15	GGGGCCCCGGGG>1111-1-1-1-11111 > ATAT > CCAA	105

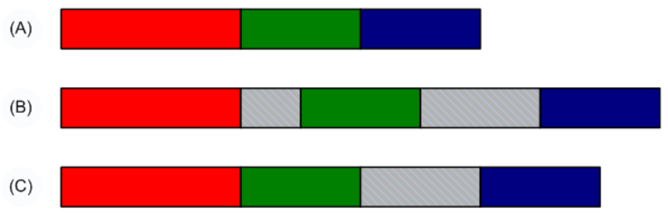


Figure 2.2: Removing deletions in the Reference Sequence. (A) Reads are derived from the novel sequence. (B) The reference sequence, S_R, contains two regions (shown as shaded gray boxes) that need to be deleted in order to make it more suitable to act as a reference for an assembly. (C) The proposed MDL process generates Θ. The process removes only those deletions which are larger than τ_1 but smaller than τ_2; where τ_1 and τ_2 are user-defined. To remove the other insertion the value of τ_2 could be increased.

2.2 MDL Algorithm

The pseudo code for Analysis using Sophisticated MDL and the scheme proposed in Section 2.1.4 is shown in Algorithm 1.

Given the reference sequence S_R and K set of reads, $\{r_1, r_2, \ldots, r_K\} \in R$, obtained from the FASTQ [19] file, the first step in the inference process is to filter all low quality reads. Lines 3-10 filters all reads that contain the base N in them and also the reads which are of low quality, leaving behind a set of O reads to be used for further analysis. This pre-processing step is common to all assemblers. Once all the low quality reads are filtered out, the remaining set of O reads are sorted and then collapsed so that only the unique reads remain.

Lines 12-26 describe the implementation of the proposed scheme as defined in Section 2.1.4. Assume that S_R is l bases long, and the length of each read is p. Therefore, ϕ_{S_R} picks up p bases at a time from S_R and checks whether or not ϕ_{S_R} is present in the set of collapsed reads R'. In the event that $\phi_{S_R} \in R'$, then the corresponding location on S_R i.e., $j \to j + p$ are flagged with "1(s)". If $\phi_{S_R} \notin R'$,

Algorithm 1 MDL Analysis of a Ref. Sequence given a set of Reads

```
1: Input reference sequence $S_R$;
2: Input read data set $\{r_1, r_2, \ldots, r_K\} \in R$;
3: for $i : 1 \rightarrow K$ do
4:     if $r_i$ contains base $N$ then
5:         remove $r_i$ from the set of reads;
6:     end if
7:     if $r_i$ has low quality bases then
8:         remove $r_i$ from the set of reads;
9:     end if
10: end for
11: Sort remaining set of reads $\{r_1, r_2, \ldots, r_O\} \in R'$, Collapse duplicated reads.
12: for $j : 1 \rightarrow l$ do
13:     read $\phi_{S_R} = \{S_R^j, S_R^{j+1}, \ldots, S_R^{j+p}\}$;
14:     if $\phi_{S_R} = r_k \in R'$ then
15:         flag 1(s) in locations $j \rightarrow j + p$
16:         flag read $r_k$ to be present.
17:     else
18:         invert read $\phi_{S_R} \rightarrow \psi_{S_R}$
19:         if $\psi_{S_R} = r_q \in R'$ then
20:             flag -1(s) in locations $j \rightarrow j + p$
21:             flag read $r_q$ to be present
22:         else
23:             flag 0(s) in locations $j \rightarrow j + p$
24:         end if
25:     end if
26: end for
27: Call out function $f(S_R, \tau_1, \tau_2)$
28: for $i : 1 \rightarrow O$ do
29:     if read $r_i$ is flagged, remove from $R$;
30: end for
31: $\zeta$ = Code-length of encoded modified sequence $\Theta$
32: $\gamma$ = Code-length of reads $R'$ not present in $S_R$
33: Total code-length $\xi = \zeta + \gamma$.
```

Algorithm 2 $f(S_R, \tau_1, \tau_2)$: Correcting all inversions and deletions

```
1: for $j : 1 \rightarrow l$ do
2:     modified sequence $\Theta \leftarrow S_R$
3:     identify all inversions by looking at -1 flags
4:     start $\equiv$ start of an inversion, end $\equiv$ end of an inversion
5:     invert genome $\Theta^{start} \rightarrow \Theta^{end}$
6: end for
7: for $j : 1 \rightarrow l$ do
8:     identify all deletion by looking at 0 flags
9:     start $\equiv$ start of a deletion, end $\equiv$ end of a deletion
10:    if $\tau_1 < end - start < \tau_2$ then
11:        remove segment of genome $\Theta^{start} \rightarrow \Theta^{end}$
12:    else
13:        segment of genome is either too large or too small.
14:    end if
15: end for
```

then invert $\phi_{S_R} \rightarrow \psi_{S_R}$ and check whether or not $\psi_{S_R} \in R'$. If yes, then one marks the corresponding location on S_R i.e., $j \rightarrow j + p$ with "-1(s)" and flag ϕ_{S_R} to be present in R'. Otherwise, the corresponding locations on S_R as "0(s)" are marked.

Lines 27-32 generates a modified sequence Θ which has all the inversions rectified in the original sequence S_R. Lines 33-41 identifies all deletion larger than τ_1 and smaller than τ_2 and removes them (see Fig. 2.2). Here τ_1 and τ_2 are user-defined. Lines 42-44 removes all the reads that are present in the original S_R and the modified sequence Θ identified by flags 1 and -1. In the end the code-lengths are identified by any popular encoding scheme like Huffman Coding [43] or Shannon-Fano coding [32] If ξ is the smallest code-length amongst all models then use Θ as a reference for the assembly of the unassembled genome rather than using S_R.

2.3 Results

Simulations were carried out on both synthetic data as well as real data. At first, the MDL process was analysed on synthetic data using four different sets of mutations by varying the number and length of {SNPs, inversions, deletions and insertions}. The experiments using synthetic data were carried out by generating a sequence S_N. The set of reads were derived from S_N and sorted using quick sort algorithm [41]. Each experiment modified S_N to produce two reference sequences S_{R1} and S_{R2} by randomly putting in the four sets of mutations. The choice of the best reference sequence was determined using the code-length generated by the MDL process. See Tables 2.3, 2.4, 2.5 and 2.6 for results.

Once robustness of the MDL scheme on each of the four types of mutations was confirmed, two-set of experiments were carried out on real data using Influenza viruses A, B and C which belong to the Orthomyxoviridae group. Influenza Virus A has five different strains i.e., {H1N1, H5N1, H2N2, H3N2, H9N2}, while Influenza Viruses B and C each have just one. The genomes of Influenza viruses are divided into a number of segments. Influenza virus A and B each have eight segments while virus C has seven segments [87]. Among the first segments of each of the viruses only one was randomly selected and then modified to be our novel genome, S_N. Reads were then derived from S_N and compared with all the seven reference sequences. Results are tabulated in Table 2.7.

Similarly, the second set of experiments analyzed the performance of the MDL proposed scheme on reference sequences of various lengths. The test was designed to determine whether the proposed scheme chooses a smaller reference sequence with more number of unaligned reads or does it choose the optimal reference sequence for assembly? The reads were derived from Influenza A virus (A Puerto Rico 834 (H1N1)) segment 1. All the reference sequences used in this test were also derived from the same H1N1 virus, however, with different lengths. Please see Tables 2.8 and 2.9 for details.

Table 2.3: Variable number of SNPs: The experiment shows the effect of increasing the number of SNPs. S_{R2} has a higher number of SNPs as opposed to S_{R1}. The code-length suggests that S_{R1} is the model of choice as it has a smaller code-length. The results indicate that the MDL based scheme works successfully on a variable number of SNPs by opting for the model with fewer numbers of SNPs.

Ref. Seq.	SNPs	No. of inversions	No. of insertions	No. of deletions	Code-length using Proposed Scheme (Kb)
S_{R1}	183	52 / 52	62 / 59	62	1815.14
S_{R2}	224	50 / 51	66 / 58	63	1843.35

Table 2.4: Variable number of deletions: The experiment shows the effect of a variable number of deletions where the locations and lengths of these deletions were random. $\frac{136}{196}$ shows that out of 196 deletions in S_{R1}, only 136 were removed. The remaining deletions were not recovered due to the choice of τ_1 and τ_2. S_{R2} has a higher number of deletions as opposed to S_{R1}. The code-length suggests that S_{R1} is the model of choice because it has a smaller code-length.

Ref. Seq.	SNPs	No. of inversions	No. of insertions	No. of deletions	Code-length using proposed Scheme (Kb)
S_{R1}	0	0	0	136 / 196	1200.3
S_{R2}	0	0	0	132 / 203	1228.25

Table 2.5: Variable number of insertions: The experiment shows the effect of increasing the number of insertions where the locations and lengths of deletions was random. S_{R2} has higher number of insertions. The code-length suggests that S_{R1} is the model of choice because it has a smaller code-length. The experiment shows that although no deletions were put in the actual sequence, yet still two and three deletions were found for S_{R1} and S_{R2} respectively. This may be due to a large section of reads which could not align to the reference sequence on the edges of these insertions.

Ref. Seq.	SNPs	No. of inversions	No. of insertions	No. of deletions	Code-length using Proposed Scheme (Kb)
S_{R1}	0	0	182	2 / 0	1997.28
S_{R2}	0	0	189	3 / 0	2015.35

2.4 Discussion

The MDL proposed scheme was tested using two-set of experiments. In the first set the robustness of the proposed scheme was tested using reference sequences, both real and simulated, having four types of mutations {Inversions, deletion, Deletions, SNPs} where the locations and the lengths of these mutations were chosen randomly. This was done with the help of a program called change_sequence. The program 'change_sequence' requires the user to input Υ_m, the probability of mutation, in addition to the original sequence from which the reference sequences are being derived. It start by traversing along the length of the genome, and each time it

Table 2.6: Variable number of inversions: The experiment shows the proposed scheme is robust to the number of inversions in the reference sequence. Both S_{R1} and S_{R2} have the same code-length. This is because the MDL scheme not only detected all the inversions for S_{R2} but also recovered all of them. So effectively $S_{R2} \equiv S_{R1}$ after utilizing the MDL process as explained in Fig. 2.1.

Ref. Seq.	SNPs	No. of inversions	No. of insertions	No. of deletions	Code-length using Proposed Scheme (Kb)
S_{R1}	0	0	0	0	586.04
S_{R2}	0	176 / 176	0	0	586.04

arrives at a new base, a uniformly distributed random generator generates a number between 0 and 100. If the number generated is less than or equal to Υ_m a mutation is introduced. Once the decision to introduce a mutation is made, the choice of which mutation still needs to be made. This is done by rolling a biased four sided dice. Where each face of the dice represents a particular mutation i.e., {inversion, deletion, insertion and SNPs}. The percentage bias for each face of the dice is provided by the user as four additional inputs, Υ_{inv}, for the percentage bias for inversions, Υ_{indel}, representing percentage bias for deletion and deletions and Υ_{SNP} for SNPs. If the dice chooses inversion, insertion or deletion as a possible mutation it still needs to choose the length of the mutation. This requires one last input from the user, Υ_{len}, identifying the upper threshold limit of the length of the mutation. A uniformly distributed random generator generates a number between 1 and Υ_{len}, and the number generated corresponds to the length of the mutation.

The proposed MDL scheme is shown to work successfully, as it chooses the optimal reference sequence to be the one which has smaller number of SNPs (Table 2.3), smaller number of deletions (Table 2.4) and smaller number of insertions (Table 2.5) when compared to the novel genome. The proposed MDL scheme is also seen to detect and rectify most, if not all, of the inversions present in the reference sequence (Table 2.6). Since the code-length of S_{R1} is the same as S_{R2}, and all the inversions of S_{R2} are rectified, the corrected S_{R2} sequence and S_{R1} sequence are equally good for reference-assisted assembly.

The experiment carried out using Influenza viruses is shown in Table 2.7. One sequence was randomly chosen amongst the seven sequences and modified at random locations, using the same 'change_sequence' program, to form the novel sequence S_N. The novel sequence contained {SNPs = 7, inversions = 4, deletions = 1, deletion = 3} as compared to the original sequence. The MDL process used the reads derived from S_N to compare seven sequences and determined Influenza Virus B to be optimal reference sequence as it had the smallest code-length. The MDL process rectified all inversions while only one insertion was found. This meant that the remaining two deletions were smaller than τ_1. The set of reads and Influenza virus B was then fed into MARAGAP (discussed in detail in chapter 3.8). The novel genome reconstructed by the MARAGAP pipeline was one contiguous sequence with a length of 2368 bases and a completeness of 96.62%.

The second-set of experiment tests the correctness of the MDL proposed scheme, by testing the MDL scheme on a single set of reads, but on a number of different reference sequences having a wide range of lengths. Results tabulated in Tables 2.8 and 2.9 prove that the MDL proposed scheme determines the optimal reference sequence even when all the contending reference sequences are closely related to one another in terms of their genome and length.

All simulations were carried out on Intel Core i5 CPU M430 @ 2.27 GHz, 4 GB

Table 2.7: Simulations with Influenza Virus A, B and C. One of the sequences from Influenza Virus {A, B, C} was randomly selected and modified to include {SNPs = 7, inversions = 4, deletions = 1, insertions = 3}. Out of the seven reference sequences Influenza Virus B (S_{R6}) was found to have the smallest code-length (68.62 Kb), and is therefore, the model of choice. The experiment also shows that given the optimal reference sequence, in this case S_{R6}, the MDL process rectifies all inversions (4/4). However, given non-optimal reference sequences, the proposed MDL process is not able to rectify the inversions (0/4). So the proposed algorithm chooses the optimal reference sequence, and given the optimal reference sequence it may correct most of the inversions.

S. No.	Ref. Seq. (Influenza Virus)	No. of Inversions	No. of Deletions	Code-length using Proposed Scheme (Kb)
S_{R1}	A, H1N1 (NC_002023.1)	0 / 4	1	254.109
S_{R2}	A, H5N1 (NC_007357.1)	0 / 4	1	254.109
S_{R3}	A, H2N2 (NC_007378.1)	0 / 4	1	254.109
S_{R4}	A, H3N2 (NC_007373.1)	0 / 4	1	254.109
S_{R5}	A, H9N2 (NC_004910.1)	0 / 4	1	254.109
S_{R6}	B (NC_002204.1)	4 / 4	1	68.62
S_{R7}	C (NC_006307.1)	0 / 4	1	254.027

RAM. Execution time of MDL proposed scheme have been provided in Table 2.8.

Table 2.8: The proposed MDL scheme is verified on a different set of reference sequences but the same set of reads R. R contained 3817 reads all of which were derived from 'Influenza A virus (A Puerto Rico 834 (H1N1)) segment 1, complete sequence'. Out of 3817 reads the method extracted 696 unique reads which were then used in the MDL proposed scheme. All the reference sequences were derived from the same Influenza A (H1N1) virus. Ref. Seq. 1% (S_{R1}) has a length which is only 1% of the actual genome. Similarly S_{R5} has a quarter of the length of the actual genome. All other genomes were derived in a similar way. For e.g., S_{R9} has two H1N1 viruses concatenated together making the length twice that of the original H1N1 sequence. The results show that the proposed MDL scheme chooses the best reference sequence, one which has the smallest code-length. The experiment also proves the correctness of the scheme as it determines S_{R7} as the optimal reference sequence. Note: It was S_{R7} from which all the reads were derived.

S.No.	Ref. Seq.	No. of Unaligned Reads	Code-length (KB)	Execution Time (s)	Length of New Seq
S_{R1}	1%	696	128.60	0.046	14
S_{R2}	2%	696	128.73	0.031	47
S_{R3}	5%	693	128.575	0.046	113
S_{R4}	10%	684	127.576	0.046	229
S_{R5}	25%	668	126.615	0.093	565
S_{R6}	50%	650	126.615	0.109	650
$\underline{S_{R7}}$	$\underline{100\%}$	$\underline{3}$	$\underline{14.276}$	$\underline{0.078}$	$\underline{2342}$
S_{R8}	150%	2	21.164	0.062	2341
S_{R9}	200%	2	27.808	0.124	2341
S_{R10}	300%	2	41.525	0.140	2341

2.5 Conclusions

The chapter explored the application of MDL based proposed scheme for selection of the optimal reference sequence for comparatively assembly. The proposed scheme

Table 2.9: The experiment tests the proposed MDL scheme on a single set of reads, and also on a number of reference sequences. The set of reads, 390 in total, were derived from 'Influenza A virus (A Puerto Rico 834 (H1N1)) segment 1, complete sequence' using the ART read simulator for NGS with read length 30, standard deviation 10, and mean fragment length of 100 [42]. Similarly the reference sequences were derived from the same H1N1 virus. S_{R1} signifies that it's length is 75% of the actual genome. Results show that the MDL proposed scheme chooses the correct reference sequence S_{R4} even when all the contending sequences are closely related to one another in terms of genome and length.

S.No.	Ref. Seq.	No. of Unaligned Reads	Code-length (KB)	Length of New Seq
S_{R1}	75%	172	25.91	1755
S_{R2}	85%	148	25.10	1989
S_{R3}	95%	123	24.20	2223
$\underline{S_{R4}}$	$\underline{100\%}$	$\underline{109}$	$\underline{23.62}$	$\underline{2341}$
S_{R5}	105%	108	24.22	2458
S_{R6}	115%	107	25.50	2692
S_{R7}	125%	106	26.78	2926

was compared SMR and found that the standard method is not sufficient for finding the optimal sequence. An alternative is needed. Therefore, the proposed MDL scheme encompassed within itself the standard method of 'counting the number of reads' by defining it in an inverted fashion as 'counting the number of reads that did not align to the reference sequence' and identified it as the 'data given the hypothesis'. Furthermore, the proposed scheme included the model, i.e., the reference sequence, and identified the parameters (θ_{M_i}) for the model (M_i) by flagging each base of the reference sequence with {-1, 0, 1}. The parameters of the model helped in identifying inversions and thereafter rectifying them. It also identified the locations of deletions. Deletions larger than a user defined threshold τ_1 and smaller than τ_2 were corrected. Therefore, the proposed MDL scheme not only chooses the optimal reference sequence but also fine-tunes the chosen sequence for a better assembly of the novel genome.

Furthermore, experiments conducted to test the robust and correctness of the MDL proposed scheme, both on real and simulated data, proved to be successful.

3. A MODULAR APPROACH TO REFERENCE ASSISTED GENOME ASSEMBLY PIPELINE - MARAGAP *

Information theory has had profound impacts on almost every brach of science, from analysing radio channels [1], to inferences in gene-regulatory networks [70, 130] and cancer informatics [81]. Rissanen's Minimum Description Length (MDL), an information-theoretic criterion, is an inference tool that learns regular features in data through data compression. MDL uses "code-length" as a metric to determine the optimum model [26, 89, 112]. Even within the area of computational biology and bioinformatics, the MDL principle has been used successfully in inferring the structure of gene regulatory networks [17,27], compression of DNA sequences [52,53], gene clustering [47,107], analysis of genes related to breast cancer [11,30,31], transcription factor binding sites [100] and to determine the optimal reference sequence selection for reference assisted genome assembly [116–119, 121, 123, 124].

This chapter presents MARAGAP, a modular approach to reference assisted genome assembly pipeline. MARAGAP employs concepts from MDL, graph theory and Bayesian statistics to present an algorithmic pipeline which is modular in nature, provides detailed information on the mutations of the novel genome when compared to the reference and assembles the novel sequence. The modular approach of the assembly pipeline allows each module to be updated or replaced without af-

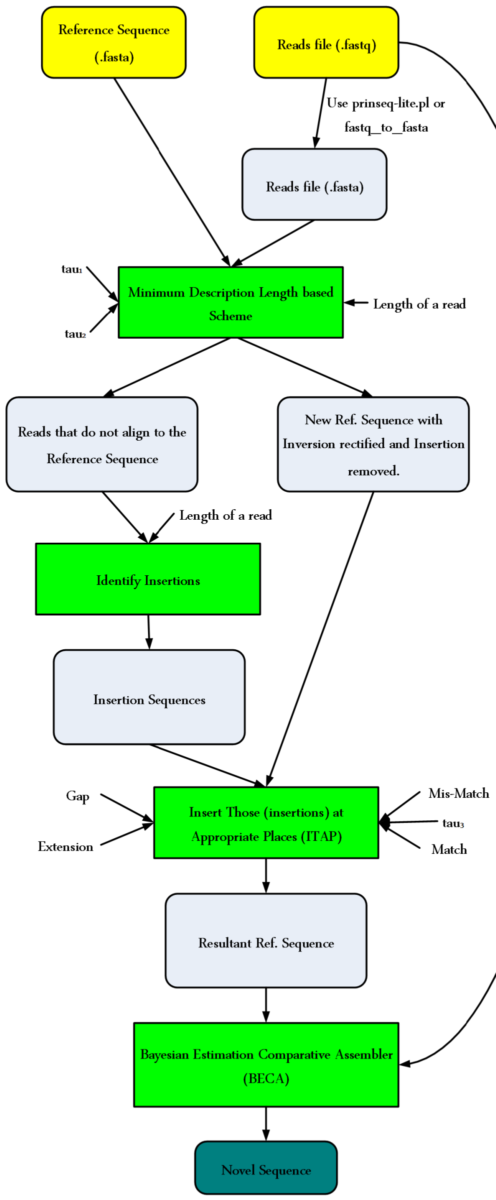
*Reprinted with permission from "A modular approach to reference assisted sequence assembly," by Bilal Wajid, Erchin Serpedin, Mohamed Nounou and Hazem Nounou, Int. J. Computational Biology and Drug Design, Apr. 14 (manuscript accepted).

Reprinted with permission from "MiB: A Comparative Assembly Processing Pipeline," by Bilal Wajid, Erchin Serpedin, Mohamed Nounou and Hazem Nounou, 2012 IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS12), Dec 2012, Washington DC., USA.

fecting the remainder of the pipeline. Each module itself uses the reference sequence and the set of reads to first determine the mutations in the novel sequence and then correct them.

Use of graph theory is somewhat inherent in the assembly pipeline as reads may be represented as nodes, and can overlap between the reads as edges. Once the graph is identified the assembly process may be defined as determining a Hamiltonian path or an Eulerian path from the root to the leaf. The nodes in the graph represent the intermediary reads while the path itself signifies the contiguous sequence.

MARAGAP initiates by deducing the optimal reference sequence using the MDL principle. Additionally, the proposed module infers the locations and lengths of inversions and deletions allowing the user to correct them [116, 117, 119, 123, 124]. Further on, a De-Bruijn graph-based technique infers the set of insertions while an affine-match, affine-gap local alignment search tool determines the locations for these insertions. The user is allowed to vary parameters to make the search space looser or tighter and then correct these mutations to obtain a better novel sequence. Lastly, the resultant sequence and the set of reads are passed onto a Bayesian detection/estimation tool which identifies SNPs and then estimates the most appropriate base for that position of the sequence. Together, via the inference of these mutations and subsequent corrections, the reference assisted genome assembly pipeline 'MARAGAP' provides a novel sequence. Figure 3.1 shows the proposed work flow diagram of the assembly pipeline. MARAGAP was compared against nine other assembly algorithms with sequences ranging from 0.15 million bases to 6.6 million bases in length. Larger sequences were not assembled owing to limitations in the available computing infrastructure.



Reference Sequence (.fasta)
Reads file (.fastq)
Use prinseq-lite.pl or fastq_to_fasta
Reads file (.fasta)
tau₁
tau₂
Minimum Description Length based Scheme
Length of a read
Reads that do not align to the Reference Sequence
New Ref. Sequence with Inversion rectified and Insertion removed.
Length of a read
Identify Insertions
Insertion Sequences
Gap
Extension
Insert Those (insertions) at Appropriate Places (ITAP)
Mis-Match
tau₃
Match
Resultant Ref. Sequence
Bayesian Estimation Comparative Assembler (BECA)
Novel Sequence

Figure 3.1: Work flow of MARAGAP: The MDL-based scheme requires τ_1 and τ_2, two user-defined parameters, where $\tau_1 > k$, where k is the length of a read, and τ_2 prevents the method from runaway. Similarly ITAP, requires four user-defined parameters: Match score, Mis-match score, gap penalty and extension. In addition, ITAP requires a strictness parameter, τ_s, which is increased for achieving more strictness and decreased for less strictness. Lastly, the Bayesian framework does not present any user defined parameters. MARAGAP's current design does not cater for either translocations or duplications.

3.1 Estimating the Optimal Reference Sequence using ‘Code-Length’

The first line of analysis for any reference assisted assembly tool deals with determining the optimal reference sequence for assembly. The optimal reference sequence is defined as the sequence which reports the lowest number of mutations when compared to the reads of the target sequence. There are a number of methods that determine the optimal reference sequence. The simplest approach is to count the number of reads that align to the reference sequence. The optimal reference being the one onto which most reads align to. A more sophisticated approach is to use the MDL framework which takes into consideration both the length of the reference and the number of reads that align to the reference in order to evaluate a ‘code-length’, the optimal reference being the one with the smallest code-length [116–119,121,123,124].

3.2 Detection and Correction of Inversions and Deletions

MARAGAP assumes that the reference sequence is a mutated version of the target sequence, and therefore, MARAGAP attempts to detect and then correct these mutations in order to bring the reference a step closer to the target sequence. Each module provides details about the length and locations of the mutations as a separate log file while the detection and correction of inversions and deletions are performed using MDL based method described earlier in chapter 2.

3.3 Detecting Insertions

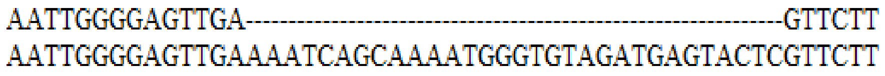
The previous module established reads that do/do not map to the reference sequence. If R is the entire set of reads from the novel genome, let R' identify the set of reads that did not align to the reference sequence. Reads not aligning to the reference sequence may be attributed to either SNPs or insertions. The current module uses a De-Bruijn graph structure to infer the set of insertions present in the novel sequence when compared to the reference sequence. The method works by looking at each read r'_i, within the set of reads R' as a node N_i of the graph. The parent Pa_i and child Ch_i of each read are sought for within the set of reads R'. Once the parents and the children of all the reads are identified such that a link from the parent to the child is established, $Pa_i \rightarrow N_i \rightarrow Ch_i$, determining the roots and leaves is relatively simple. Nodes of the graph structure that do not have any parents are the roots, whereas nodes that do not have any children are the leaves. In genome assembly, each graph has only one root, therefore, the total number of roots define the total number of graphs that need to be formed. Traversing from the root all the way to a leaf helps in finding one contig, (please refer to Fig. 3.2). The movement from the root to the leaf divides R' into smaller groups $\{R'_1, R'_2, \cdots, R'_p\}$, where R'_x represents the set of reads used in generating the x^{th} contig or the x^{th} insertion, and subscript p illustrates the total number of insertions. Furthermore, the number of reads that make up one insertion may not necesarily be equal to the number of reads that collectively make up another insertion. Therefore, $R' = [R'_1, R'_2, \cdots, R'_p]^T$, where each R'_x contains all the reads needed to define one insertion.

3.4 Estimating the Locations of Previously Identified Insertions

Once the insertion sequences have been inferred, the current module tries to estimate the location of these insertions in the novel sequence. This is accomplished via an affine-match, affine-gap local alignment search tool which prefers consecutive matches and gaps over sparsely distributed matches and gaps in the alignment with zero tolerance for mismatches. This finely-tuned local alignment tool ensures that the start and end sections of the insertions match perfectly with the reference sequence with continuous gaps in the middle. Should this happen, the area of the genome where such an alignment occurs will be considered as the exact location where the

```
ACTCTCCCGATACTTTGTAAGGCGAGATTCTTG   1
|CTCTCCCGATACTTTGTAAGGCGAGATTCTTGG   1
||TCTCCCGATACTTTGTAAGGCGAGATTCTTGGC   5
|||||||CGATACTTTGTAAGGCGAGATTCTTGGCACAAC   5
||||||||||||CTTTGTAAGGCGAGATTCTTGGCACAACCACGT   2
||||||||||||||TTGTAAGGCGAGATTCTTGGCACAACCACGTTA   3
|||||||||||||||||TAAGGCGAGATTCTTGGCACAACCACGTTAAGA   1
||||||||||||||||||AAGGCGAGATTCTTGGCACAACCACGTTAAGAT   1
|||||||||||||||||||AGGCGAGATTCTTGGCACAACCACGTTAAGATC   4
|||||||||||||||||||||||GAGATTCTTGGCACAACCACGTTAAGATCGTCT   4
|||||||||||||||||||||||||||TTCTTGGCACAACCACGTTAAGATCGTCTCTAC   5
||||||||||||||||||||||||||||||||GGCACAACCACGTTAAGATCGTCTCTACACGGC
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
ACTCTCCCGATACTTTGTAAGGCGAGATTCTTGGCACAACCACGTTAAGATCGTCTCTACACGGC
```

Figure 3.2: This figure shows how the insertion sequence is formed. The first read happens to be the root. The overlapping reads are shown aligned at the appropriate place with respect to the parent and the child of the read. The number shown after the read is an indication of the length of the read which did not overlap with its child, the less the better. For instance the number '1' means the node and its child shares an overlap of '$k-1$' set of bases. The last line shows the consensus sequence that was formed, i.e., the insertion sequence. Please note that the underlying principle is that the figure shows the formation of a graph structure with the parent nodes and child nodes determined by the degree of overlap between them. This figure may show similarity to a multiple sequence alignment, however, within the framework of genome assembly, it is strictly a graph construction.



```
AATTGGGGAGTTGA-------------------------------GTTCTT
AATTGGGGAGTTGAAAATCAGCAAAATGGGTGTAGATGAGTACTCGTTCTT
```

Figure 3.3: This figure shows how the locations of the insertions are determined. The sequence at the top is part of the reference sequence whereas the sequence at the bottom happens to be the insertion. Using a fine-tuned local alignment search tool, the alignment between the reference sequence and the insertion is shown. Identical start and end sections with continuous gaps in the middle illustrate the exact location of the insertion. Once the appropriate place is determined, the insertion is corrected.

insertion is to be corrected. Fig 3.3 represents such an alignment. Once all the insertions are corrected, the entire set of reads, R, of the novel genome and the resulting sequence are used as an input to the next module.

3.5 Bayesian Framework for Detection and Estimation of SNPs

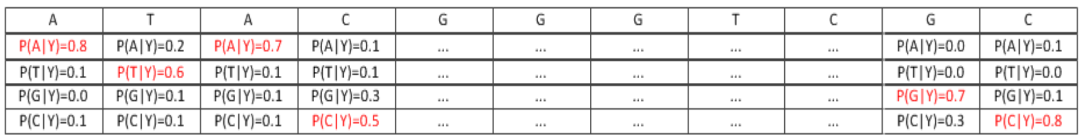
A Bayesian framework for the detection and estimation of SNPs is necessary in order to make effective use of the Q-values present in the .fastq file. These Q-values portray the probability of making an incorrect base-call. Let P_e denote the probability of an incorrect base-call, then e.g., for Solexa, the expression for P_e is given by $P_e = 1 - (1/(1 + 10^{-Q/10}))$. Assume a read r_i to be w bases long, and each position of the read r_i to be identified by a different base, $r_i^l = X$, where $X \in \{A, T, G, C\}$ and $l \in \{1, \ldots, w\}$. Therefore, assume $p(X) = 1 - P_e$ denote the probability that the base call X is correct.

As Q-values are derived by analyzing images taken of the cells on which sequencing is performed, let Y represent such an 'image data'. Therefore, if Y is used to derive $p(X)$, then $p(X)$ is in fact $p(X|Y)$. The Q-values stored in the .fatsq file do not provide the complete probability mass function and only capture one Q-value per base at any particular location of the read. Assume a uniform distribution for the remaining set of bases in the read such that of $\sum_{X=\{A,T,G,C\}} p(X|Y) = 1$. Let β_i^j

identify the probability that the i^{th} read, r_i, has a base X at location j, i.e., $r_i^j = X$. Therefore,

$$\begin{aligned} p(X|Y) &= p(r_i^j = X|Y) = \beta^j_{(i,X)} \\ \sum_{X=\{A,T,G,C\}} \beta^j_{(i,X)} &= 1. \end{aligned}$$

Using the above scheme one obtains a complete distribution β_i^j for each location j within the read r_i, and it does this for the entire set of reads, R, as illustrated in Fig. 3.4.



A	T	A	C	G	G	G	T	C	G	C
P(A|Y)=0.8	P(A|Y)=0.2	P(A|Y)=0.7	P(A|Y)=0.1	...	...	...	...	...	P(A|Y)=0.0	P(A|Y)=0.1
P(T|Y)=0.1	P(T|Y)=0.6	P(T|Y)=0.1	P(T|Y)=0.1	...	...	...	...	...	P(T|Y)=0.0	P(T|Y)=0.0
P(G|Y)=0.0	P(G|Y)=0.1	P(G|Y)=0.1	P(G|Y)=0.3	...	...	...	...	...	P(G|Y)=0.7	P(G|Y)=0.1
P(C|Y)=0.1	P(C|Y)=0.1	P(C|Y)=0.1	P(C|Y)=0.5	...	...	...	...	...	P(C|Y)=0.3	P(C|Y)=0.8

Figure 3.4: Each location of the read has its associated probability distribution as shown in the columns underneath it. The base-call for a particular location is the base which has the highest probability (shown in red).

As MARAGAP uses reference assisted assembly, let S_R denote the reference sequence and S_T be the target sequence that we wish to derive. Furthermore, assume S_R^k is the k-th base in the sequence S_R, where $k \in \{1, \ldots, n\}$ and n is the size of the reference genome. The Bayesian framework uses probability scores to evaluate the best alignment of a read with the reference sequence. The region along which a read r_i, w bases long, aligns to the reference sequence ranges from S_R^k to S_R^{k+w-1}, where k denotes the position where the alignment starts, and $k + w - 1$ represents the position where the alignment ends. The probability of the alignment 'a' of a

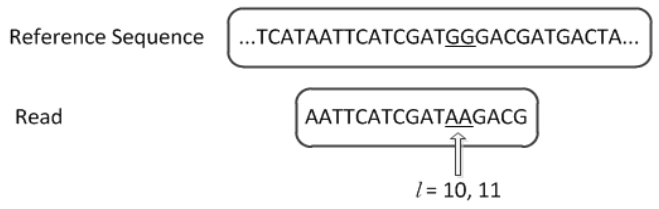


Figure 3.5: Alignment of a read with the reference sequence. The underlined bases differ in relation to each other.

particular read r_i onto a location k on the reference strain S_R is given by

$$p(a|r_i; S_R^k) = \prod_{l=1}^{w} \beta_i^l = \alpha_i^k. \tag{3.1}$$

The probability of alignment, α_i^k, measures how well any read aligns onto any location on the reference template, and, like all probabilities, follows the basic rule of summing up to one, i.e., $\sum_{k=1}^{n} \alpha_i^k = 1$, see Fig. 3.5.

Eq. (2.1) is used to calculate the alignment probability of all reads to all locations on the reference sequence. Let R represent all the reads and let φ denote all the alignment probabilities $\alpha(.)$, calculated above. Once the probabilities of all the alignments φ of all the reads R are calculated, the 'layout' is built. Only those alignments of the reads are taken into consideration greater than a user defined threshold 'τ', where $\tau \propto 1/n$, 'n' being the size of the reference sequence.

Establishing the layout allows the consensus sequence to be constructed via estimating 'site-specific-probabilities' of the novel sequence S_T. Let Φ_T^j be the site-specific-probabilities such that either of the bases $\{A, T, G, C\}$ may belong to the location j on S_T. Assuming m to be the total number of reads which align to a particular position j, then

$$\Phi_T^j = p(S_T^j|R, \varphi, Y, S_R) \equiv p(S_T^j|\vec{\mathcal{Z}}) \propto \prod_{k=1}^{m} \alpha_k^a \times \beta_k^{l=j-a+1}, \tag{3.2}$$

where a is the starting position of the alignment of the read r_k onto S_R, α_k^a is its probability score and β_k^l identifies the probability of the base call at the l^{th} location of the read r_k. In Eq. (2.2), the variable $\vec{\mathcal{Z}}$ sums all the variables, i.e., the combined effect of all the reads, R, all alignment probabilities $\varphi(.)$, the image data Y, and the reference sequence S_R. Thus, $\Phi_T^j = p(S_T^j|R, \varphi, Y, S_R) \equiv p(S_T^j|\vec{\mathcal{Z}})$. Therefore, by using Baye's rule twice, it follows that

$$\begin{aligned}
p(S_T^j|\vec{\mathcal{Z}}) &= \frac{p(\vec{\mathcal{Z}}|S_T^j)p(S_T^j)}{p(\vec{\mathcal{Z}})} = \frac{p(S_T^j)}{p(\vec{\mathcal{Z}})} \times \prod_{k=1}^{m} p(\vec{z_k}|S_T^j) \\
&= \frac{p(S_T^j)}{p(\vec{\mathcal{Z}})} \times \prod_{k=1}^{m} \frac{p(S_T^j|\vec{z_k})p(\vec{z_k})}{p(S_T^j)} \\
&= \frac{\prod_{k=1}^{m} p(\vec{z_k})}{p(\vec{\mathcal{Z}})} \times \frac{\prod_{k=1}^{m} p(S_T^j|\vec{z_k})}{[p(S_T^j)]^{m-1}} \\
&= \lambda_j \prod_{k=1}^{m} p(S_T^j|\vec{z_k}) && (3.3) \\
\lambda_j &= \frac{\prod_{k=1}^{m} p(\vec{z_k})}{p(\vec{\mathcal{Z}}) \times [(S_T^j)]^{m-1}}. && (3.4)
\end{aligned}$$

In Eq. (2.3), the variable $\vec{z}$ represents the effect of an individual read r with their individual alignments a, whereas the variable $\vec{Z}$ represents the combined effect of all the reads, R, and all alignment probabilities, $\varphi(.)$. Eq. (2.3) is used to calculate the site-specific-probabilities Φ_T^j and λ_j is a scaling factor such that $\sum_{j=\{A,T,G,C\}} \Phi_T^j = 1$. The base-call on each site of the novel genome S_T is given to the base which has the highest site-specific-probability, depicted in Fig 3.6.

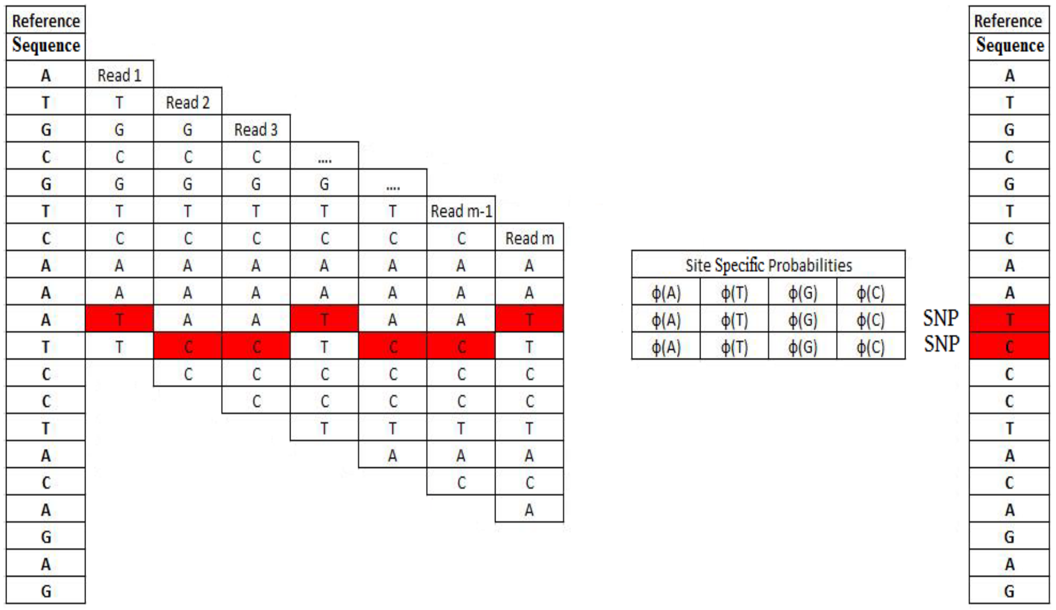


Figure 3.6: Alignment of the reads with the reference sequence and formulation of the site-specific-probabilities is based on $\alpha()$, probability of the alignments, and $\beta()$, probability mass function of the bases of the reads. The base-call of the new sequence depends on the highest site-specific-probability at that location. Therefore, for each location of the new sequence, one ends up with a complete distribution identified by $\Phi(.)$.

3.6 Results and Discussion

Four sets of experiments were conducted on genomes of varied lengths, as illustrated in Table 3.1. In the first three experiments, the reference sequence was obtained by modifying the original sequence at random locations with mutations of random lengths. For the assembly of Pseudomonas Aeruginosa PAb1, the reference sequences were real and belong to the same phylogeny as PAb1. They are Pseudomonas Aeruginosa PA14, PAO1 and PA7. Standard assembly metrics were used to compare the assembly, carried over by MARAGAP, with the assemblies produced by other algorithms such as Velvet [129], SSAKE [15], VCAKE [48], QSRA [10], SHARCGS [25], IDBA [?], MIRA [18] and Maq [http://maq.sourceforge.net/]. All these assemblers are part of the Baari and Genobuntu Package (chapters 4.5 and 5.2 respectively). Brief explanations of these metrics have been presented in Table 1.3.

The assembly of Mycobacterium NBB4 revealed that Mira's assembly was the best, followed closely by Maq. MARAGAP's assembly did not reveal assembly statistics as efficacious as either Mira or Maq because MARAGAP assembler could not rectify one insertion and one deletion. Comparing the assemblies of Nitrobacter Ha. X14 shows that MARAGAP provides a more improved assembly when compared to other assemblers. This improvement comes at the cost of a higher time complexity. Assembly of Vibrio Ch. O1 chromosome 2 using optimum reference shows that Mira and MARAGAP assemblies were similar, while using sub-optimum reference MARAGAP assembly was better than the rest. The assembly of Pseudomonas Aeruginosa PA7 chromosome using PA14 as an optimum reference illustrates that MARAGAP assembly performs the best. The assembly statistics of other assemblers were not presented in Table 3.7 since the other assemblers exhibited 'exceptions' during their run causing them to end prematurely. Overall, MARAGAP has shown performance exceeding most assemblers; exceptions include Mira whose performance closely matches MARAGAP's. In addition, MARAGAP provides details of all four mutations, i.e., insertions, deletions, inversions and SNPs as 'log' files. The location and size of these mutations relative to the reference sequence may be used in downstream analysis for disease association studies, understanding inter-individual variations and drug responses [101]. The larger time complexity of MARAGAP is primarily due to its inherent probabilistic framework and insertion location identification framework. As for medical and pharmaceutical reasons, one may largely favor accuracy of results over inherent time-complexity [69, 90].

In order to facilitate reproducibility of results the version number of these assemblers, along with the parameters used in the assemblies, have been provided. All experiments and all algorithms, except the Bayesian framework, were conducted on an Intel(R) Core(TM) i5 CPU M430 @ 2.27 GHz 2.26 GHz with 4 GB Ram. The Bayesian framework was executed on Intel Core2 Duo E8500, 3.16 Ghz, 4 GB Ram, CentOS 5.5 $\times$64 operating system. MARAGAP was not tested on larger species because of hardware limitations. In addition, estimated figures for time and memory have been avoided owing to the inherent random nature of the mutations. As MARAGAP is based on detecting and correcting mutations, the time it takes to assemble a genome would be based on the number of mutations present in the target genome in relation to the reference genome. However, it is not uncommon to assemble a bird, a fish or a snake using 50-100 cores, 512-1000 GB RAM with a reasonable execution time ranging from 24 hours to 1000 hours [9].

Source code of MARAGAP, MARAGAP's installation instructions and all necessary data can be downloaded from (`https://sourceforge.net/projects/refasp/`).

3.6.1 Experiment 1: Assembly of Mycobacterium Cholerae NBB4 Plasmid pMYCCH.02

The reference sequence was derived from Mycobacterium Cholerae NBB4 Plasmid pMYCCH.02, complete sequence (NC_018023.1), length 142623 base-pairs, using change_sequence.exe, with the following parameters, $\Upsilon_m = 2\%, \Upsilon_{SNP} = 60\%, \Upsilon_{inv} = 20\%, \Upsilon_{indel} = 10\%, \Upsilon_1 = 100, \Upsilon_2 = 1000$. This set-up introduced the following mutations in the reference sequence {SNPs: 451, inversions: 140, insertions: 85, deletions: 63}. Reads of length 30 were derived using the ART NGS read simulator using 10$\times$ fold coverage, standard deviation 30 and without paired-ends.

S. No	Sequence	Accession Number	Sequence Length (bp)
1	Mycobacterium Chubuense NBB4 plasmid pMYCCH.02	NC_018023.1	142623
2	Nitrobacter hamburgensis X14 plasmid 1	NC_007959.1	294829
3	Vibrio cholerae O1 str. 2010EL-1786 chromosome 2	NC_016446.1	1,046365
4	Vibrio cholerae IEC224 chromosome II	NC_016945.1	1,072136
5	Vibrio cholerae LMA3984-4 chromosome II	NC_017269.1	946986
6	Vibrio cholerae Ban5, ICEVchban5	GQ_463140	1,02122
7	Pseudomonas Aeruginosa PAO1 chromosome	NC_002516.2	6,264404
8	Pseudomonas Aeruginosa UCBPP-PA14 chromosome	NC_008463.1	6,537648
9	Pseudomonas Aeruginosa PA7 chromosome	NC_009656.1	6,588339

Table 3.1: This table shows the list of sequences used in the experiments.

The assembly was conducted using MARAGAP and compared with Velvet [129], SSAKE [15], VCAKE [48], QSRA [10], SHARCGS [25], IDBA [?], MIRA [18] and Maq (`http://maq.sourceforge.net/`). Results tabulated in Table 3.2 show that the reference assisted assemblies perform better than the denovo assemblies owing to their use of the reference sequence. However, it is interesting to note that the variation in execution times is significant with Velvet completing its assembly in 1 second and MARAGAP completing its assembly in 2,686 seconds. However, the results obtained show that the assembly completed by MIRA is the best, followed closely by MARAGAP. Maq lags behind by a decent margin. A more detailed analysis of the MARAGAP assembled genome conducted using the help of TABLET [71] is shown

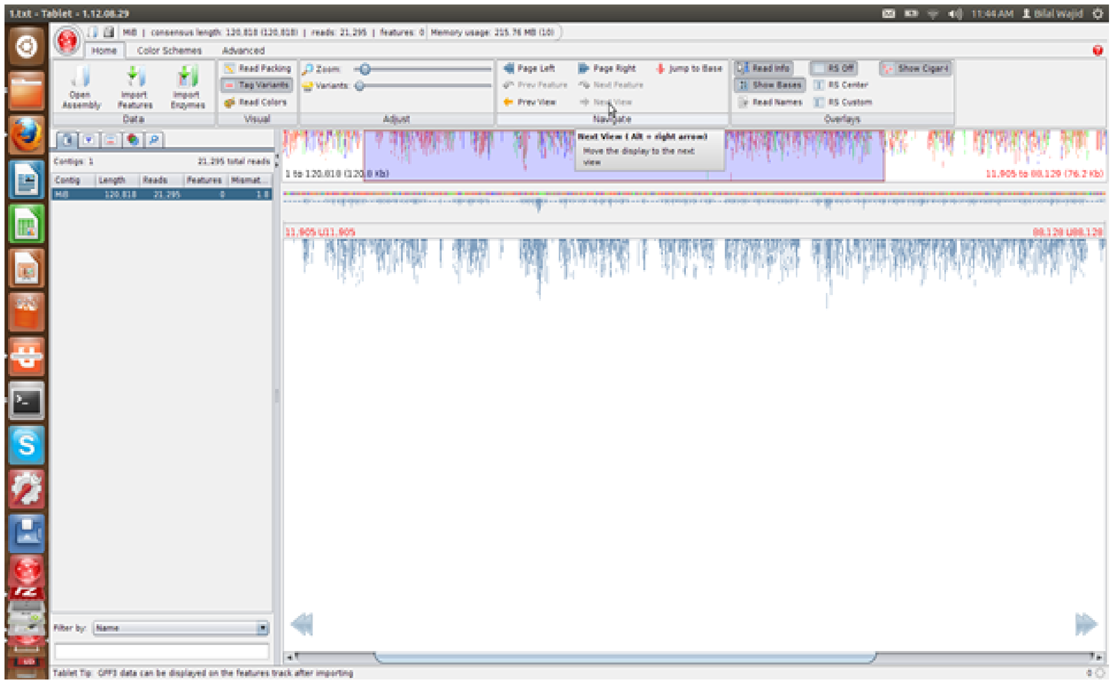
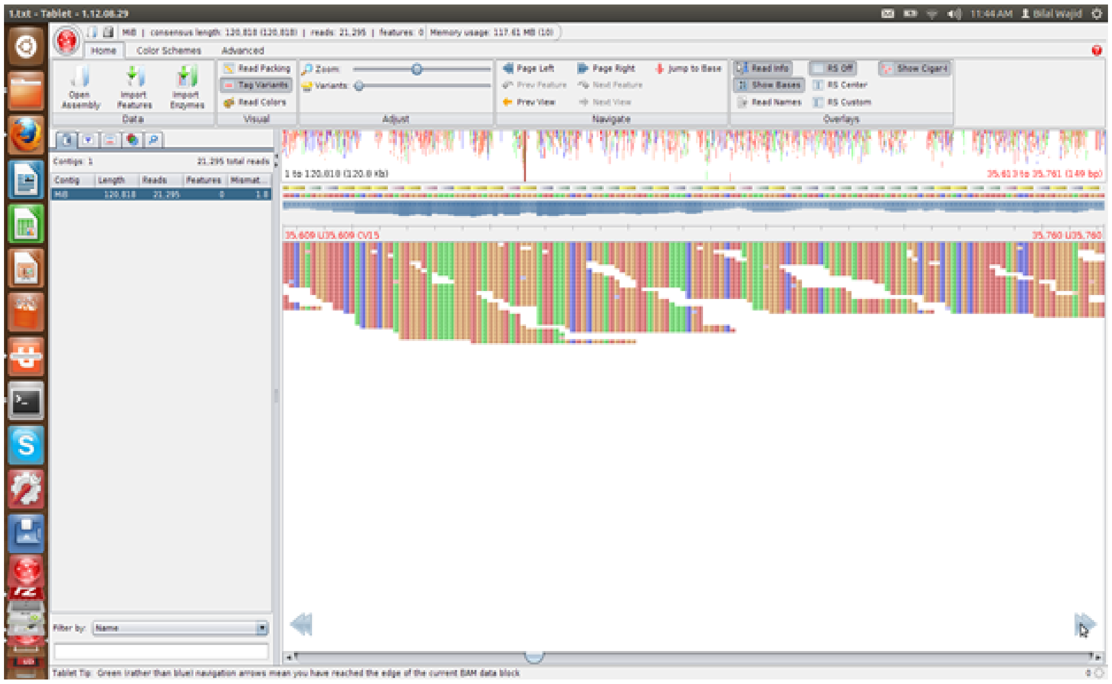


Figure 3.7: MARAGAP assembly of NBB4 Plasmid: This figure shows the output generated via TABLET for the MARAGAP assembly of Mycobacterium NBB4 Plasmid. Notice that there are areas of the genome where no read aligns possibly due to some insertions in the reference sequence which MARAGAP could not detect and remove. The figure also shows areas of the genome with large coverage.

in Figs. 3.7 and 3.8.

3.6.2 Experiment 2: Assembly of Nitrobacter Hamburgensis X14 Plasmid 1

The reference sequence was derived from Nitrobacter Hamburgensis X14 Plasmid 1, complete sequence (NC_007959.1), length 294829 base-pairs, by introducing 284 SNPs, 153 inversions, 26 insertions and 26 deletions. Paired-end reads of length 30 were derived using the ART NGS read simulator using 20× fold coverage, standard deviation 30 and mean fragment length of 1000. Employing the fastx toolkit and the fastq_quality_filter, with a quality cut-off of 18 and a minimum percentage 90, discarded about 6400 low-quality forward-biased reads and about 1900 low quality



Tablet Tip: Green (rather than blue) navigation arrows mean you have reached the edge of the current BAM data block

Figure 3.8: High coverage regions of NBB4 plasmid assembly using MARAGAP: This figure shows one such high coverage region of the assembly of Mycobacterium Cholerae NBB4 Plasmid using MARAGAP.

S. No	Assembly Metrics	Velvet	VCAKE	SSAKE	QSRA	SHARCGS	IDBA	Mira	Mira	Maq	MARAGAP
Ver		1.2.07	1	3.8	1	1	1	Denovo	Comparative	1	1
1	No. of Contigs	448	2110	2034	1646	1286	402	1081	1	125	1
2	Length of Largest Contig	1994	1556	611	1490	1047	2267	179	156868	107611	120818
3	N50	471	240	67	115	115	512	56	156868	107611	120818
4	N75	297	45	49	70	79	282	50	156868	107611	120818
5	N90	155	36	39	45	61	162	46	156868	107611	120818
6	NG50	471	372	63	121	111	506	-	156868	107611	120818
7	NG75	297	225	44	79	75	279	-	156868	107611	120818
8	Contigs $\geq$ N50	94	227	648	412	391	82	-	1	1	1
9	Contigs $\geq$ 200 bp	250	274	12	108	85	236	0	1	1	1
10	Mean	321.9	98.6	63.9	92.6	105.9		-	156868	928.328	120818
11	Median	232	40	55	72	88	241	-	-	-	120818
12	Sum of the Contig Lengths	14425	208050	130048	152395	136188	141984	-	156868	116041	120818
13	Coverage	9	6	11	9	10	10	-	9	12	11
14	Run time (s)	1	66	11	6	910	8	15	10	3	2686
15	Parameters Used	K = 15 exp_cov =auto cov_cutoff =auto min_contig_lgth =31	-e 20 -k 30 -n 16 -t 5 -m 15 -o 31 -v 3	-w 3 -m 16 -r 0.6 -t 0 -z 31 -c 0.6	-k 30 -o 31 -u 16 -l 15 -t 3	-d 0 -l 31 -t 0	–mink 15 –maxk 30 –step 1 –min_count 2 –min_contig 31	denovo genome accurate solexa	mapping genome accurate solexa bbq=30 -AS:nop=1		$\tau_1 = 100$ $\tau_2 = 1000$ $\tau_3 = 7$ Match 4 Mis-Mat -5 Gap 0 Exten. 1

Table 3.2: The assembly of Mycobacterium Cholerae NBB4 Plasmid pMYCCH.02 utilized 10 state-of-the-art assemblers and compared using standard assembly metrics.

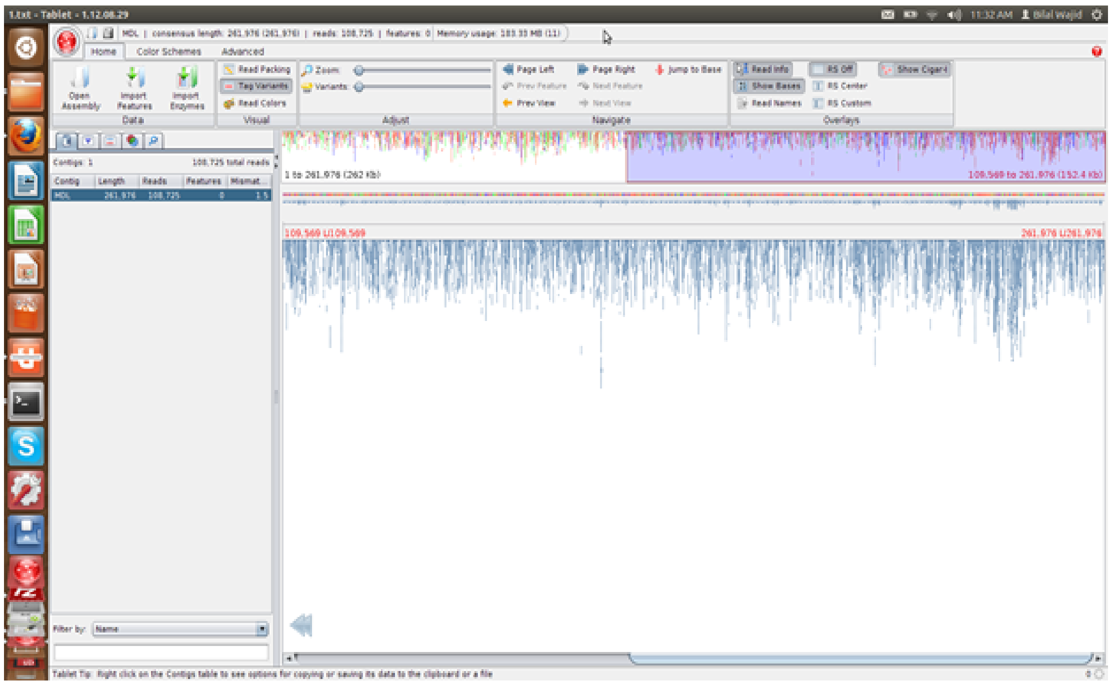
reverse-biased reads. Read error correction was completed using SHREC [99] with 5 iterations and 4 parallel threads. Out of 71,000 forward-biased reads, SHREC corrected about 16,000 reads, whereas out of 77,000 reverse-biased reads SHREC corrected about 3,500 reads. The resulting filtered and corrected reads were used in the assembly process utilizing different assemblers and then compared with standard assembly metrics using the program assembly_statistics.exe. Results, tabulated in Table 3.3, show that the assembly done by MARAGAP outperforms all other assemblies with a higher cost in terms of its time complexity. This is due to the use of the Bayesian framework which aligns all the reads to all possible locations on the reference sequence to locate the best probabilistic alignments. The results of an analysis conducted with TABLET [71] are depicted in Figs. 3.9, 3.10 and 3.11.

3.6.3 Experiment 3: Assembly of Vibrio Cholerae O1 strain 2010EL-1786 chromosome 2

The choice of the reference sequences for the assembly of Vibrio Ch. 01 strain (str.) 2010EL-1786 chromosome (chr.) 2 was determined via the MDL-based proposed scheme. Results tabulated in Table 3.4 reveal that Vibrio Cholerae IEC224 is the optimal (Opt.) reference sequence with 'Ref 0: Mutated reference' coming last. Vibrio Cholerae Ban5, with 0 reads found, does not qualify as a possible option. Therefore, the assembly process proceeded with Vibrio Cholerae IEC224 chr. II as the reference. Paired-end reads of length 30 were derived using the ART NGS read simulator using 20× fold coverage, standard deviation 20 and mean fragment length of 10000. Employing the fastx toolkit and the fastq_quality_filter with a quality cut-off of 18 and a minimum percentage 90 helped in discarding about 23,000 low-quality, forward-biased reads. The quality filter was not applied on reverse-biased reads because the quality box plot and the nucleotide distribution graphs of the re-

S. No	Assembly Metrics	Velvet	VCAKE	SSAKE	QSRA	SHARCGS	IDBA	Mira	Mira	Maq	MARAGAP
Ver		1.2.07	1	3.8	1	1	1	Denovo	Comparative	1	1
1	No. of Contigs	486	2991	2782	3948	3184	975	4795	1	129	1
2	Length of Largest Contig	5437	1937	1606	1707	481	1544	864	297417	259323	261976
3	N50	1188	187	61	78	96	388	65	297417	259323	261976
4	N75	678	88	47	51	61	207	52	297417	259323	261976
5	N90	277	41	38	39	42	126	46	297417	259323	261976
6	NG50	1059	213	40	75	88	335	67	297417	259323	261976
7	NG75	478	122	0	47	48	151	54	259323	259323	120818
8	Contigs > N50	71	514	886	1069	864	208	1649	0	0	0
9	Contigs > 200 bp	276	458	29	81	115	449	33	1	1	1
10	Mean	533	112.6	61	71	81.56	267	66	297417	-	261976
11	Median	270	60	51	55	65	185	58	297417	0	261976
12	Sum of the Contig Lengths	268979	337003	169568	279928	259694	260617	315930	297417	265460	261976
13	Coverage	8	6	13	8	9	9	7	7	8	8
14	Run time (s)	4	112	16	10	843	12	88	10	8	8544
15	Parameters Used	K = 15 exp_cov =auto cov_cutoff =auto min_contig_lgth =31	-e 20 -k 30 -n 16 -t 5 -m 15 -o 31 -v 3	-w 3 -m 16 -r 0.6 -t 0 -z 31 -c 0.6	-k 30 -o 31 -u 16 -l 15 -t 3	-d 0 -l 31 -t 0	–mink 15 –maxk 30 –step 1 –min_count 2 –min_contig 31	denovo genome accurate solexa	mapping genome accurate solexa bbq=30 -AS:nop=1		$\tau_1 = 100$ $\tau_2 = 3000$ $\tau_3 = 7$ Match 4 Mis-Mat -5 Gap 0 Exten. 1

Table 3.3: Assembly of Nitrobacter Hamburgensis X14 Plasmid 1



MDL | consensus length: 261,976 (261,976) | reads: 108,725 | features: 0 | Memory usage: 183.33 MB (11)

Contigs: 1 108,725 total reads

Contig	Length	Reads	Features	Mismat...
MDL	261,976	108,725	0	1.5

1 to 261,976 (262 Kb)

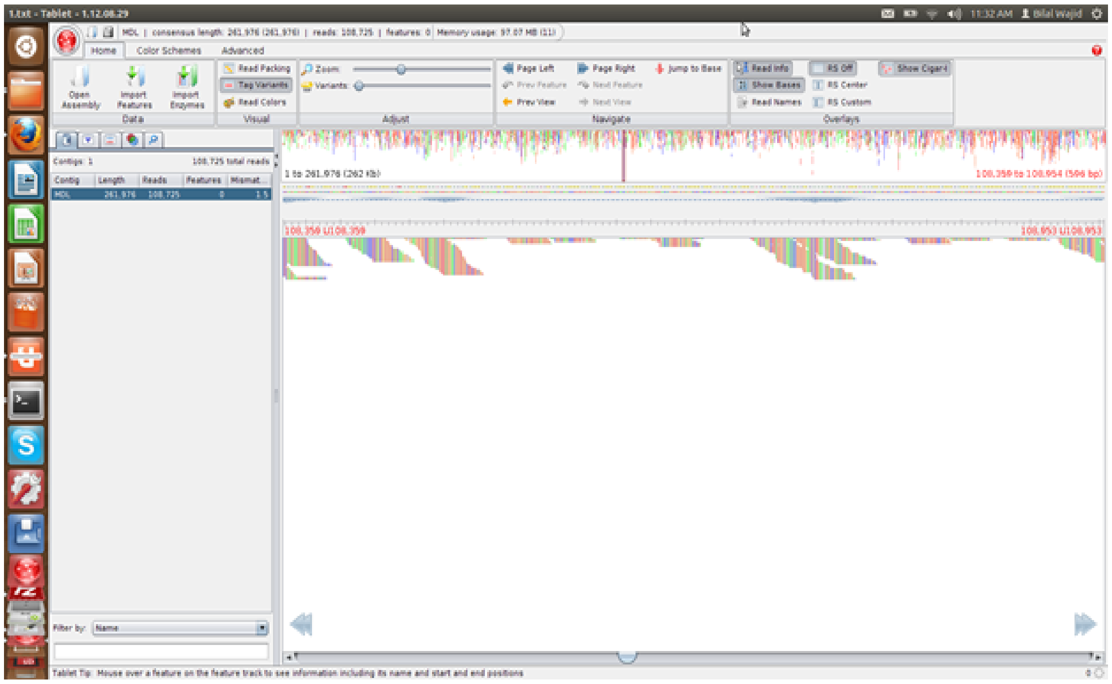
109,569 to 261,976 (152.4 Kb)

109,569 U109,569

261,976 U261,976

Tablet Tip: Right click on the Contigs table to see options for copying or saving its data to the clipboard or a file

Figure 3.9: MARAGAP assembly of X14 Plasmid: Compared to Fig. 3.7, MARAGAP was able to achieve better assembly for X14 than for the NBB4 Plasmid in Experiment 1. The MARAGAP assembly shows large coverage regions spanning the entire sequence, illustrating an overall good assembly.



MDL | consensus length: 261,976 (261,976) | reads: 108,725 | features: 0 | Memory usage: 97.07 MB (11)

Contigs: 1 108,725 total reads

Contig	Length	Reads	Features	Mismat...
MDL	261,976	108,725	0	1.5

1 to 261,976 (262 Kb)

108,359 to 108,954 (596 bp)

108,359 U108,359

108,953 U108,953

Tablet Tip: Mouse over a feature on the feature track to see information including its name and start and end positions

Figure 3.10: Low coverage regions of X14 plasmid assembly by MARAGAP: The figure shows a particular low coverage region of the assembly. A closer inspection reveals that many of these regions are smaller than one read length, and therefore, difficult to detect and correct.

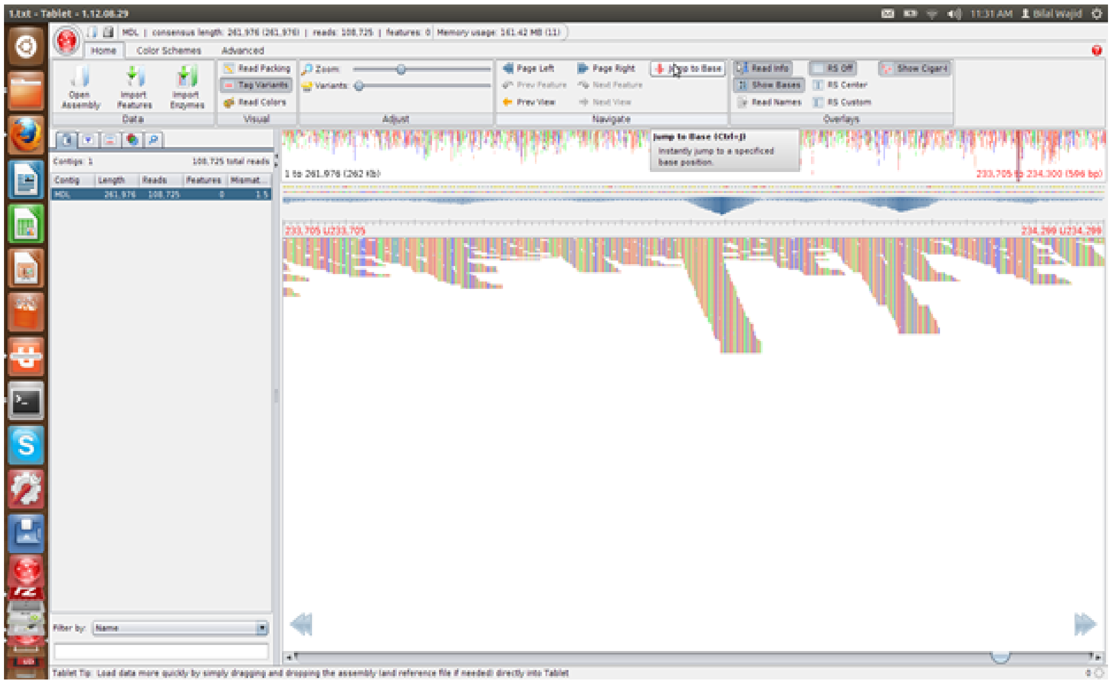


Figure 3.11: High coverage regions of X14 assembly: The figure shows a particular high coverage region of the assembly conducted by MARAGAP of Nitrobacter Hamburgensis X14 Plasmid 1.

verse biased reads revealed the reads were of good quality. Read error correction was employed using SHREC [99] with 5 iterations and 4 parallel threads. Out of 326,000 forward-biased reads SHREC corrected about 9,150 reads, whereas out of 309,000 reverse-biased reads SHREC corrected about 8,600 reads. The resultant filtered and corrected reads were used in the assembly process.

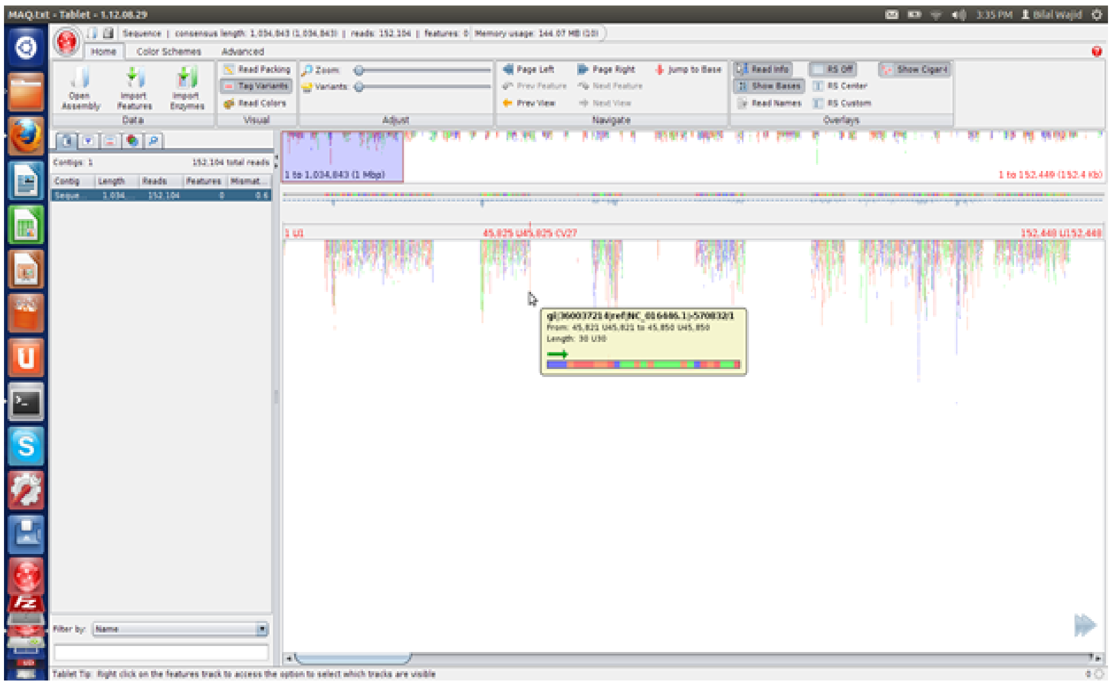
The assembly statistics reveal that given IEC224 as the reference sequence both MIRA [18] and MARAGAP produce similar assemblies, as confirmed in Table 3.5. However, given a suboptimum reference like reference 0 Mutated Reference, MARAGAP assembly was better than the remaining assemblies. MARAGAP was found to be slower than MIRA but faster than SHARCGS [25]. For an analysis using TABLET [71], please refer to Figs. 3.12, 3.13 and 3.14.

3.6.4 Experiment 4: Assembly of Pseudomonas Aeruginosa PAb1 Chromosome

The choice of the reference sequences for the assembly of Pseudomonas Aeruginosa PAb1 was determined via the proposed MDL scheme. Results tabulated in Table 3.6 reveal that PA14 is the Opt. reference sequence. The assembly was carried out with reads from the run SRR001657 drawn from SRA (`http://www.ncbi.nlm.nih.gov/sra`). Reads were filtered using the fastx toolkit. The assembly of PAb1 was carried out using VCAKE, QSRA, IDBA and MARAGAP. The results are depicted in Table 3.7. The assembly statistics reveal that MARAGAP assembly was better with fewer contigs. MARAGAP also presented the largest single contig with a decent coverage. Although MARAGAP took a longer time to complete the assembly process, the improvement in the assembly justifies the time taken. The results from other assemblers were not presented in Table 3.7 because the other assemblers ended their runs prematurely.

S. No	Reference Sequence (Accession No.)	Code-length (MB)	No. of Reads found	No. of Reads not found
1	Vibrio Cholerae IEC224 chr. II (NC_016945.1)	11.34	225629	55582
2	Vibrio Cholerae LMA3984-4 chr. II (NC_017269.1)	18.644	137023	144188
3	Ref 0: Mutated Reference (Self Generated)	21.67	109180	172031
4	Vibrio Cholerae Ban5 (GQ_463140)	26.138	0	281211

Table 3.4: This table shows the list of sequences used in Experiment 3. The list has been arranged in increasing order of the code-lengths with reference Seq. 1 having the smallest code-length, with reference Seq. 4 having the largest. In accordance with the MDL principle, the reference Seq. with the smallest code-length is the Opt. reference sequence. Therefore, we use IEC224 as our Opt. reference sequence in Experiment 3. However, in order to test the robustness of assemblers Maq, Mira and MARAGAP we repeat the assembly process, this time using reference 3 to evaluate the strength of each assembler when the assembly process uses a sub-optimum reference sequence.



Sequence | consensus length: 1,034,843 (1,034,843) | reads: 152,104 | features: 0 | Memory usage: 144.07 MB (10)

Contigs: 1 152,104 total reads

Contig	Length	Reads	Features	Mismat...
Seque...	1,034...	152,104	0	0.6

1 to 1,034,843 (1 Mbp)

1 to 152,449 (152.4 Kb)

1 U1 45,825 U45,825 CV27 152,448 U152,448

gi|360037214|ref|NC_016446.1|-570832/1
From: 45,821 U45,821 to 45,850 U45,850
Length: 30 U30

Tablet Tip: Right click on the features track to access the option to select which tracks are visible

Figure 3.12: MAQ Assembly of Vibrio Cholerae O1 str. 2010EL-1786 chr. 2 Vibrio Cholerae IEC224 chr. 2 as a reference: Note that there are many areas of the genome which have little or no coverage. This is because MAQ places a series of N(s) in areas of the genome it could not confirm.

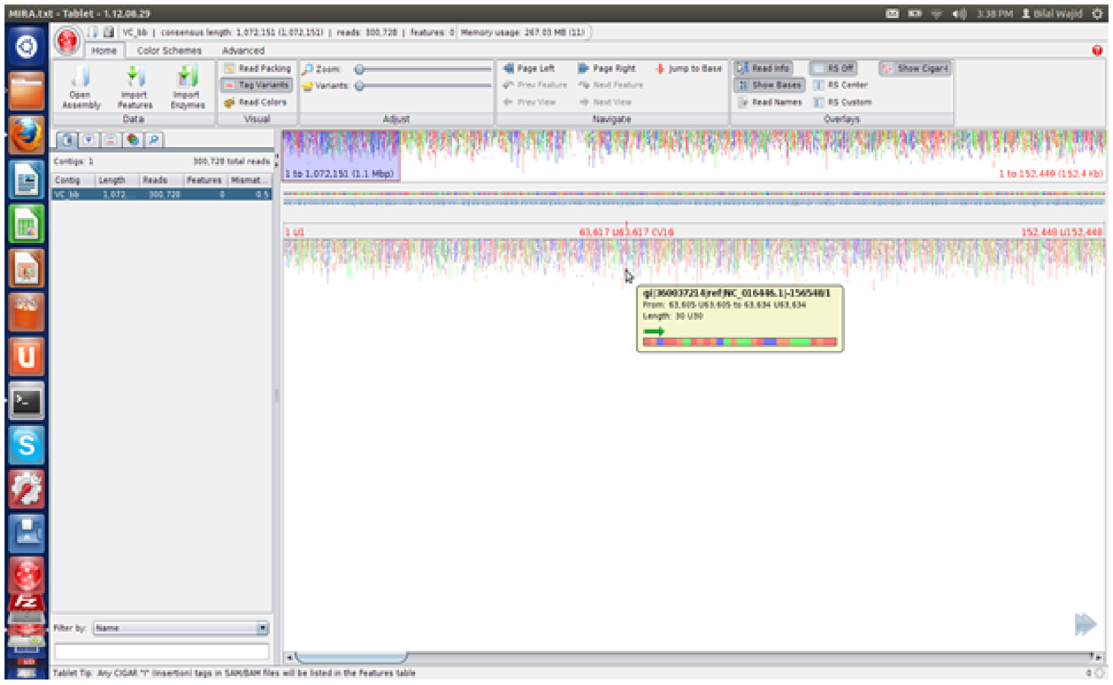


Figure 3.13: MIRA Comparative assembly of Vibrio Cholerae O1 str. 2010EL-1786 chr. 2 using Vibrio Cholerae IEC224 chr. 2 as a reference: The target sequence assembled by MIRA shows overall good coverage. Comparison with Figure 3.12 shows that the assembly conducted by Mira is better than that the one performed by MAQ.

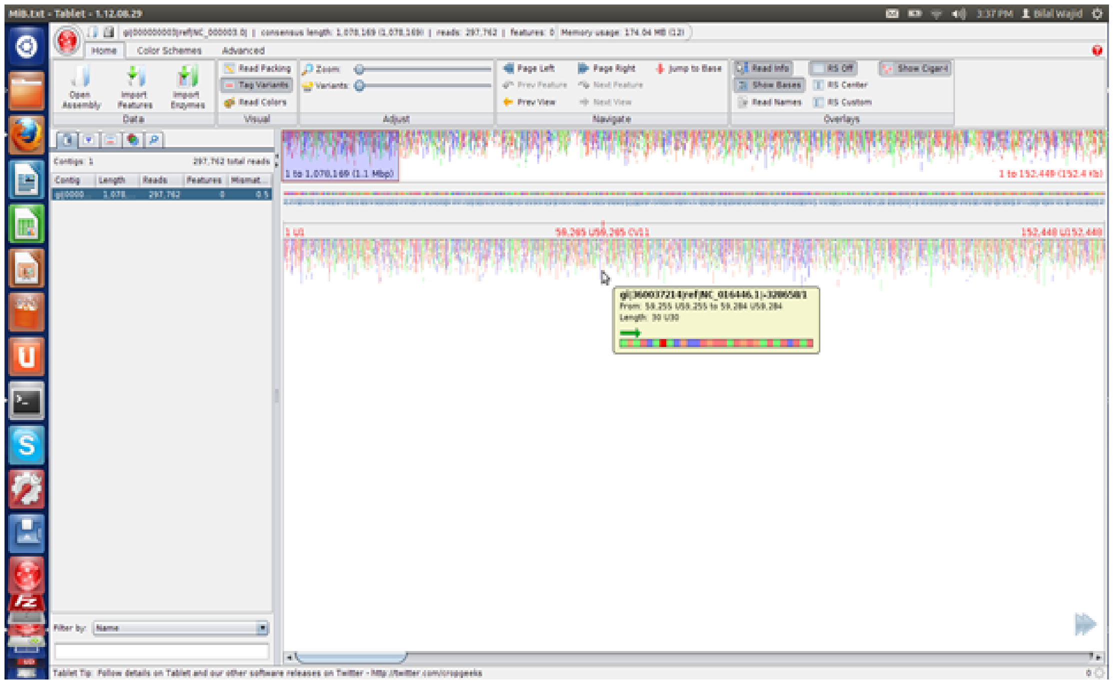


Figure 3.14: MARAGAP assembly of Vibrio Cholerae O1 str. 2010EL-1786 chr. 2 using Vibrio Cholerae IEC224 chr. 2 as a reference sequence: The assembly by MARAGAP shows overall good coverage.

S. No	Assembly Metrics	Velvet	VCAKE	SSAKE	QSRA	SHARCGS	IDBA	Mira	Mira	Maq	MARAGAP
Ver		1.2.07	1	3.8	1	1	1	Denovo	Comparative	1	1
1	No. of Contigs	989	13630	9631	15058	1220	4741	18510	1	1	1
2	Length of Largest Contig	10971	1275	1285	1265	65	1281	819	1,072127	513584	1,078169
3	N50	2265	167	62	80	34	260	72	1,072,127	513584	1,078169
4	N75	1250	68	47	52	32	157	55	1,072127	513584	1,078169
5	N90	618	39	39	39	31	101	46	1,072127	513584	1,078169
6	NG50	5551	614	115	227	0	669	163	1,072127	513584	1,078169
7	NG75	4963	542	102	201	0	595	150	1,072127	513584	1,078169
8	Contigs $\geq$ N50	139	2318	3191	4250	508	1185	6396	0	0	0
9	Contigs $\geq$ 200 bp	656	1760	27	307	0	1761	72	1	1	1
10	Mean	1020	99.4	60	71.5	35	203	71	1,072127	513584	1,078169
11	Median	531	48	53	57	34	158	62	1,072127	513584	1,078169
12	Sum of the Contig Lengths	1008821	1354636	573815	1075846	42959	961797	1306583	1,072127	513584	1,078169
13	Coverage $\times$	8	6	15	8	210	9	6	8	17	8
14	Run time (s)	18	457	51	50	57447	39	575	146	20	130,214
15	Parameters Used	K = 15	-e 20	-w 3	-k 30	-d 0	–mink 15	denovo	mapping		$\tau_1 = 150$
		exp_cov =auto cov_cutoff =auto min_contig_lgth =31	-k 30 -n 16 -t 5 -m 15 -o 31 -v 3	-m 16 -r 0.6 -t 0 -z 31 -c 0.6	-o 31 -u 16 -l 15 -t 3	-l 31 -t 0	–maxk 30 –step 1 –min_count 2 –min_contig 31	genome accurate solexa	genome accurate solexa bbq=30 -AS:nop=1		$\tau_2 = 10,000$ $\tau_3 = 7$ Match 4 Mis-Mat -5 Gap 0 Exten. 1

Table 3.5: Assembly of Vibrio Cholerae O1 strain 2010EL-1786 chr. 2

S. No	Reference Sequence (Accession No.)	Code-length (MB)	No. of Reads found	No. of Reads not found
1	Pseudomonas Aeruginosa UCBPP-PA14 chr. (NC_008463.1)	141.04	1,226133	1,031404
2	Pseudomonas Aeruginosa PAO1 chr. (NC_002516.2)	152.50	1,095037	1,162500
3	Pseudomonas Aeruginosa PA7 chr. (NC_009656.1)	223.713	399181	1,858356

Table 3.6: This table shows the list of sequences used in the Experiment 4. The list has been arranged in increasing order of code-lengths with PA14 has the smallest code-length while PA7 chr. having the largest code-length. In accordance with the MDL principle, the reference Seq. with the smallest code-length is the Opt. reference sequence. Therefore, PA14 acts as the Opt. reference sequence in Experiment 4.

3.7 Experimental Methods

3.7.1 Simulating Reference Sequences

In all experiments, the original sequence was modified at random locations by selecting one of four mutations, i.e., inversion, deletions, insertions and SNPs with random lengths. The method requires the user to input five parameters, i.e., Υ_m: probability of mutation, Υ_{inv}: percentage bias for inversions, Υ_{indel}: bias for insertions and deletions, Υ_{SNP}: denoting the corresponding percentage for SNPs and Υ_{len}: largest possible length of the mutation. The method was described previously in Section 2.4.

S. No Ver	Assembly Metrics	VCAKE 1	QSRA 1	IDBA 1	MARAGAP 1
1	No. of Contigs	156834	76899	34775	1
2	Length of Largest Contig	4195	5832	5285	6261358
3	N50	92	132	176	6261358
4	N75	297	66	112	6261358
5	N90	155	44	83	6261358
6	NG50	227	159	148	6261358
7	NG75	140	89	84	6261358
8	Contigs $\geq$ N50	26985	14907	8860	1
9	Contigs $\geq$ 200 bp	10428	7550	6904	1
10	Mean	76.312	98.71	157.810	6261358
11	Median	45	61	118	6261358
12	Sum of the Contig Lengths	11968268	7590560	5487859	6261358
13	Coverage	6	10	14	13
14	Run time (hours)	2	0.25	0.1	31
15	Memory used (GB)	1.3	1.6	0.5	4
16	Parameters Used	-e 20 -k 33 -o 34 -n 17 -t 5 -m 16 -v 3	-u 17 -k 33 -o 34 -l 16 -t 3 -c 0.6	–mink 17 –maxk 33 –step 1 –min_count 2 –min_contig 34	$\tau_1 = 1000$ $\tau_2 = 100000$ $\tau_3 = 10$ Match 4 Mis-Mat -5 Gap 0 Exten. 1

Table 3.7: Reads were derived from the run 'SRR001657' from the Sequencing Read Archive. The assembly was conducted using four assemblers and compared using standard metrics utilizing the program 'assembly_statistics'. Pseudomonas Aeruginosa UCBPP-PA14 was employed as a reference Seq. by MARAGAP for the assembly of PAb1.

3.7.2 Generating, Filtering and Correcting Reads

In simulations, reads were derived from the original sequence using the ART NGS read simulator with read length 30, utilizing either 10× or 20× fold coverage and a standard deviation of 20 or 30 [42]. Reads of length 30 were chosen to ensure consistency in comparison and to check for the robustness of different assemblers. First, it was found that most of the assemblers functioned smoothly at read length 30, while some assemblers exhibited 'unstable' behavior at different lengths causing them to terminate pre-maturely during the assembly process. Second, real data for Experiment 4 also presented read length 30, therefore, the simulated data also used read length 30. Thirdly, assemblers provide better assembly at larger read lengths, therefore, in order to test the robustness of any assembler shorter read length are preferred over larger read lengths as the differences in the assemblies become magnified.

As for paired reads, Experiment 1 was carried out without paired-end reads while Experiment 2 and Experiment 3 were completed with mate-pair information. Experiment 1 was performed without filtering low quality reads or undertaking error correction, while Experiments 2 and 3 were conducted after filtering low quality reads, using the fastx toolkit (`http://hannonlab.cshl.edu/fastx_toolkit/`), and error correction using SHREC [99].

For Experiment 4, real data was used and reads were obtained from the Sequencing Read Archive (SRA) for the Illumina sequencing of Pseudomonas aeruginosa PAb1 (`http://www.ncbi.nlm.nih.gov/sra`). SRA contains two runs of PAb1 sequencing, i.e., 'SRR001656' and 'SRR001657'. Quality of these data sets were tested using the fastx toolkit which showed that run SRR001657 was better than SRR001656. For SRR001656 the quality of the reads reduced after the seventh base.

Therefore, the experiment was continued with only SRR001657 which was subsequently filtered to remove low quality reads.

3.8 Addendum

Availability and requirements:

- Project name: MARAGAP
- Project home page: `https://sourceforge.net/projects/refasp/`.
- Operating system (OS): Linux, Windows and MAC OS.
- Programming language: C.
- License: GPL 3.0
- Any restrictions to use by non-academics: None, as long as the users cite the paper.
- Contact: Please email: bilalwajidabbas@hotmail.com for assistance.

4. GIBBS-BECA: EMPLOYING GIBBS SAMPLING AND BAYESIAN ESTIMATION FOR SNPS DETECTION AND CORRECTION *

In Chapter 3.8 we discussed BECA, a framework that uses Quality (Q) values for identifying probabilities of the base calls for every read and subsequently employing them to find probabilistically the best alignments and the consensus sequence. BECA used the 'alignment-layout-consensus' paradigm where each step was executed after evaluating probabilistic measures, thereby treating every quantity as a random variable. This chapter extends BECA by using Gibbs sampling for further iterations of BECA. After each assembly the reference sequence is updated and the probabilistic score for each base call is then used further to identify the alignments and consensus sequence, yielding Gibbs-BECA. Although BECA rectifies most of the SNPs, Gibbs-BECA further improves the performance both in terms of rectifying more SNPs and offering a robust performance even in the presence of poor reference sequences.

The remainder of the chapter is divided as follows. Section 4.1 provides a quick recap of BECA, Section 4.2 discusses Gibbs sampling and Section 4.3 discusses the application of Gibbs sampling on detecting and correcting SNPs. Finally, Section 4.4 describes the computer simulations to assess the performance of the proposed

*Reprinted with permission from "Gibbs Sampling and Bayesian Estimation for Comparative Assembly," by Bilal Wajid, Erchin Serpedin, Mohamed Nounou and Hazem Nounou, 3rd International Conference on Biomedical Engineering, Electronics and Nanotechnology (MIC-BEN'13), June 2013, Valencia, Spain.

framework.

4.1 BECA

Bayesian Estimation for Comparative Assembly or BECA is used in the MARA-GAP pipeline to rectify SNPs in the sequence. This is accomplished by making effective use of a reference sequence 'S_R', the set of reads 'R' $= \{r_1, r_2, \cdots, r_n\}$ and their associated Q-values to evaluate a set of probabilistic measures. Firstly, the alignments of the reads along the reference sequence is measured probabilistically using α_i^k:

$$\alpha_i^k = p(\Omega | r_i; S_R^k) = \prod_{l=1}^{w} \beta_i^l \tag{4.1}$$

Here 'Ω' denotes the alignment of a particular read r_i (w bases long) onto a location k on the reference sequence S_R. Additionally, $\beta_i^l = p(r_i^l = X)$ represents the probability that base 'X' $= \{A, C, G, T\}$ is present at location 'l' on the i^{th} read.

Once the probability of all the alignments (symbolized by 'φ') of all the reads R is calculated, the layout is made. The layout establishes the consensus sequence by evaluating the 'site-specific-probabilities' of the target sequence S_T. Let Φ_T^j denote the site-specific-probabilities, that either of the bases $\{A, T, G, C\}$ may belong to the location j on S_T.

$$\Phi_T^j = p(S_T^j | R, \varphi, Y, S_R) \equiv p(S_T^j | \vec{\mathcal{Z}}) \propto \prod_{k=1}^{m} \alpha_k^a \times \beta_k^{l=j-a+1}, \tag{4.2}$$

where a is the starting position of the alignment of the read r_k onto S_R. Here $\vec{\mathcal{Z}}$ presents the combined effect of all the reads R, all alignment probabilities $\varphi(.)$, image data Y, and reference sequence S_R. However, rather than using $\vec{Z}$, one may also use $\vec{z}$ to represent the effect of individual reads r with their individual alignments Ω.

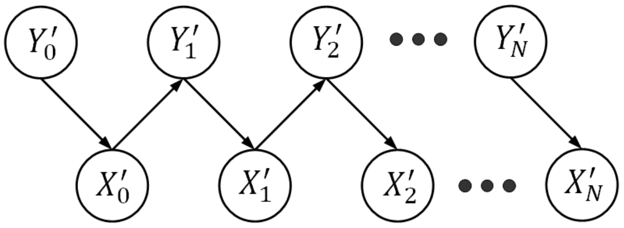


Figure 4.1: Gibbs Sampling: In the two variables case the initial value $Y_0^{'}$ is specified and $X_0^{'}$ is obtained using the conditional probability density $f(X|Y = Y_0^{'})$. Thereafter, $Y_1^{'}$ is obtained using $f(Y|X = X_0^{'})$. The process iterates, each time generating a set of random numbers, $X_k^{'}$ and $Y_k^{'}$, using the conditional densities until the distribution of X converge to $f(X)$ and the distribution of Y converges to $f(Y)$.

Therefore:

$$p(S_T^j|\vec{\mathcal{Z}}) = \lambda_j \prod_{k=1}^{m} p(S_T^j|\vec{z_k}) \tag{4.3}$$

$$\lambda_j = \frac{\prod_{k=1}^{m} p(\vec{z_k})}{p(\vec{\mathcal{Z}}) \times [(S_T^j)]^{m-1}}. \tag{4.4}$$

Eq. (4.3) is used to calculate the site-specific-probabilities Φ_T^j and λ_j is a scaling factor such that $\sum_{j=\{A,T,G,C\}} \Phi_T^j = 1$. The base-call on each site of the novel genome S_T is given to the base that has the highest site-specific-probability. In BECA's framework all the above mentioned calculations are executed once in order to determine the target sequence. In Gibbs-BECA, the target sequence is obtained after a number of iterations. After each iteration the target sequence of the previous iteration becomes the reference sequence of the next iteration. This process continues until no new SNPs are corrected. The next sections discuss how Gibbs sampling is employed to extend BECA's framework.

4.2 Gibbs Sampling

Gibbs sampling offers a framework to generate random variables from a distribution indirectly rather than having to calculate the densities. In the two variable case (X, Y), Gibbs sampler generates a random variable from $f(x)$ successively by generating a Gibbs sequence of random variables by sampling from $f(x|y)$ and $f(y|x)$. The initial value $Y_0^{'} = y_0^{'}$ is specified and the remainder of variables are obtained iteratively by alternately generating values from

$$X_j^{'} \sim f(x|Y_j^{'} = y_j^{'}); \quad Y_{j+1}^{'} \sim f(y|X_j^{'} = x_j^{'})$$

The distribution of $X_K^{'}$ converges to $f(x)$ as $K \to \infty$, see Fig. 4.1, thereby obtaining

$$Y_0^{'}, X_0^{'}, Y_1^{'}, X_1^{'}, Y_2^{'}, X_2^{'}, \cdots, Y_K^{'}, X_K^{'}.$$

The three variants of Gibbs sampling: standard, grouped and collapsed are extensions of one another. In its standard form, Gibbs sampler updates the coordinates one by one whereas in its grouped form the Gibbs sampler updates the two coordinates together. Finally, in its collapsed form the Gibbs sampler updates many coordinates at a time [64], see Fig. 4.2.

4.3 Gibbs Sampling and Genome Sequencing

Looking at the two-component collapsed form of Gibbs sampling, the aim is to identify the two components X and Y, and the two conditional density functions $f(X|Y)$ and $f(Y|X)$. In the method described above, using a reference sequence S_R, a set of reads R, probability mass function $\beta(.)$ and Eq. (4.1), one is able to identify the alignment probabilities. Similarly using Eq. (4.3), one is able to identify the site-specific-probabilities for each base at every location of the final sequence.

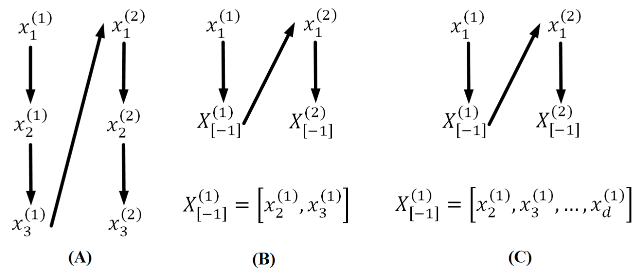


Figure 4.2: Types of Gibbs Sampling: Graphical representation of systematic forms of Gibbs sampling. (A) Standard form, (B) Grouped form, (C) Collapsed form of Gibbs sampling.

For each read r_i, define the best p alignment probabilities as $\zeta_i \in \{\alpha_{[i,1]}, \cdots, \alpha_{[i,p]}\}$. Therefore, for the entire set of m reads, R, the set of best p alignments for each read can be defined as $\varphi \in \{\zeta_1, \zeta_2, \cdots, \zeta_m\}$. Let $S_R \equiv S_{T0}$. Eq. (4.1) can now be modified to account for the best p alignments for the entire set of reads. Therefore, for one read r_i and one alignment location k, the alignment probability is

$$p(a|r_i; S_{T0}^k) = \prod_{l=1}^{w} \beta_i^l = \alpha_i^k$$

For the set of best p alignments of all the reads R, the equation above can be modified as $p(\varphi|R; S_{T0})$. Also, since

$$p(S_T^j|R, \varphi, S_R) \equiv p(S_T^j|\vec{Z}) = \lambda_j \prod_{k=1}^{m} p(S_T^j|\vec{z}),$$

the two predictive distributions are:

$$p(\varphi_{(k+1)}|R, S_{T(k)}) \text{ and } p(S_{T(k+1)}^j|R, \varphi_{(k)}, S_{T(k)}). \tag{4.5}$$

In comparative genome assembly, the starting point is naturally the reference sequence $S_{T(0)}$. Since every iteration generates the site-specific-probabilities Φ^j_T, and the probabilities change at every subsequent iteration, it is appropriate to mark the site-specific-probabilities as $\Phi^j_{T(q)}$, where q marks the q^{th} iteration. Similarly, $S^j_{T(q)}$ denotes simply the base call for the highest $\Phi^j_{T(k)}$. It was mentioned earlier that:

$$p(a|r_i; S^k_R) = \prod_{l=1}^{w} \beta^l_i = \alpha^k_i.$$

Incorporating the effects of Φ^j_T the above equation takes the form:

$$p(a_{(k+1)}|r_i; \Phi_{T(k)}) = \prod_{l=1}^{w} \beta^l_i \Phi^{a+l-1}_{T(k)} = \alpha^{a(k+1)}_i.$$

Notice the $\Phi_{T(k)}$ has replaced $S^j_{T(k)}$. This is because $S^j_{T(k)}$ is simply the base call for the highest $\Phi^j_{T(k)}$. Similarly for all the reads R and all the alignments $\varphi(.)$, the first predictive distribution can be modified to incorporate the effects of $\Phi^j_{T(k)}$ yielding $p(\varphi_{(k+1)}|R, \Phi_{T(k)})$. Similarly the effects of $\Phi^j_{T(k)}$ are also reflected in the second predictive formula, changing it from $p(S^j_{T(k+1)}|R, \varphi_{(k)}, S_{T(k)})$ to $p(S^j_{T(k+1)}|R, \varphi_{(k)}, \Phi_{T(k)})$.

Therefore, reflecting $\Phi^j_{F(k)}$ onto probability densities (4.5) updates them to get the following two predictive conditional densities:

$$p(\varphi_{(k+1)}|R, \Phi_{F(k)}) \text{ and } p(S^j_{T(k+1)}|R, \varphi_{(k)}, \Phi_{F(k)}).$$

Therefore, the Gibbs sampler generates the following sequences:

$$S_{T(0)}, \varphi_{(0)}, S_{T(1)}, \varphi_{(1)}, S_{T(2)}, \varphi_{(2)}, \cdots, S_{T(k)}, \varphi_{(k)}$$

$$as \quad k \to \infty, \; S_{T(k)} \to S_T.$$

4.4 Results and Discussion

Experiments were conducted on both real and synthetic data. For synthetic data, the simulations were carried out by generating a reference sequence S_R of length L. The novel sequence S_T was generated by making random substitutions in the reference genome. The percentage of SNPs in the reference sequence was increased in steps. This was done to evaluate the robustness of the algorithm by observing the predictive quality of inferring $\widehat{S_T}$ given varying degrees of diverging S_R, see Table 4.1. Thereafter, S_T was split into reads starting at random locations on the novel genome ensuring that no two reads start from the same location. All base locations of the reads were assigned a probability distribution $\beta(.)$ such that the overall quality of the reads was good with $P(X|Y) > 0.7$. The set of reads R, their probability distributions $\beta(.)$, and the reference sequence S_R were then used to infer the novel genome $\widehat{S_T}$. As far as the alignment phase is concerned, the alignments of all the reads $\varphi(.)$ were evaluated and all alignments having a $p(a|r_i, S_R^k) < 0.25$ were discarded. In all the experiments, the layout and consensus sequence was established resulting in one contiguous sequence, $\widehat{S_T}$, thereby, resulting in clean assemblies. In order to evaluate the performance, the assembly of $\widehat{S_T}$ was compared to S_T and the number of false-positives were recorded. The results are tabulated in Table 4.1.

Table 4.2 shows the result of the assembly when conducted with real data using genomes of varying sizes. Comparing the performance of BECA with Gibbs-BECA, it follows that although the Gibbs-BECA assembly takes more than one iteration yet it rectifies most of the SNPs introduced. Therefore, Gibbs-BECA provides a solution $\widehat{S_T}$, which is very close to the actual genome S_T.

The execution time of the first iteration of Gibbs-BECA is exactly the same as BECA, as the first iteration is common to both. However, as Gibbs-BECA uses

Table 4.1: Experiment conducted on a reference genome, S_R, having 5000 bases. In every subsequent test, the number of SNPs introduced in the novel genome S_T gets higher and higher. The results show that when S_R diverges further from the actual sequence, it takes more iterations for Gibbs-BECA to converge to the actual sequence. The results show that even if S_R diverges as much as $\frac{2252}{5000} \times 100 = 45\%$, most of the SNPs are recovered with only 26 remaining at the end of the 6^{th} iteration by the time the process converges and stops. This proves that Gibbs-BECA is very robust.

S. No.	No. of SNPs	No. of Iteration required to Converge	No. of SNPs between $\widehat{S_T}$ and S_T
1	915	2	8
2	1355	2	18
3	1801	2	34
4	2252	6	26

multiple cycles to converge, the time taken by Gibbs-BECA is equal to the number of cycles multiplied by the time it takes for one execution of BECA. Please refer to Table 4.2 for details.

4.5 Conclusions

This chapter demonstrates the use of probabilistic measures and Gibbs sampling in every step of the alignment-layout-consensus paradigm. The proposed Gibbs-BECA method, an extension of our earlier work BECA, shows that through the use of Gibbs sampling, the proposed scheme was able to rectify more SNPs compared to its predecessor and also was robust even when the choice of the reference sequence is poor. Future work could involve improving the run time of the algorithm by parallelizing the alignment and layout phase using MPI and OpenMP.

Table 4.2: Experiment conducted with real data shows the assembly of three genomes and compares the performance of BECA with Gibbs-BECA. The original sequence was modified by inserting a variable number of SNPs. ‘SNPs remaining’ identify SNPs that were not identified or corrected by the assembler. BECA requires only one iteration, while Gibbs-BECA illustrates that iterating the process improves performance as more and more SNPs are rectified with the assembly.

S. No.	Genome	Genome Length	Number of SNPs	BECA: SNPs remaining	Gibbs-BECA: SNPs remaining	Gibbs-BECA: No. of iterations required to converge
1	Hepatitis C virus genotype 1	9646	1769	170	88	4
2	Pneumocystis carinii mitochondrion	22893	4246	553	106	7
3	Acidiphilium multivorum AIU301 plasmid pACMV4	40588	7460	863	345	11

5. BAARI: A CUSTOMIZED OPERATING SYSTEM DESIGNED FOR GENOME ASSEMBLY [*]

While previous chapters discuss different software pipelines, Chapters 4.5 and 5.2 discuss cohesive software platforms that combine many different pipelines in one unit. Efforts like these are especially important in Life Sciences. This is because ever since the publication of the human genome [68], biological data throughput has exploded with proportions that has surpassed Moore's law [33]. Since then many software packages have been proposed that help translate data into meaningful knowledge. These software not only helped speed up biological research, but also compelled biologists to spend increasing amount of time and resources in installing, configuring and maintaining software.

To address these problems in the field of biology, not one, but many solutions have been proposed, all of which are built on Linux based operating systems. Examples like BioLinux [34], Scibuntu [110] (`http://scibuntu.sourceforge.net/`), Open Discovery [115], BioSLAX (`http://www.bioslax.com`) [86], LXtoo [128] and Scientific Linux (`https://www.scientificlinux.org/`) address a broad set of users, however, as there is no 'one size fits all' solution, scientists have come with other solutions each tailored for a specific problem. For instance, BioBrew provided an 'over-the-counter' cluster functionality [86, 131]. DNALinux [6] provided a pre-configured virtual machine that runs on top of the free VMWare Player on Windows XP and Vista, meaning that one could use Windows in parallel with running one's bioinfor-

[*]Reprinted with permission from "Life Sciences Driven Customized Linux Distributions," by Bilal Wajid and Erchin Serpedin, OA Bioinformatics, 2014, Jan 18; 2(1):1.

Table 5.1: Brief comparison of Linux distributions used in Life Sciences. $\star$ Gentoo is available in multiple environments like Gnome, KDE, Xfce, LXDE, i3, etc. The number of packages pre-compiled in Gentoo is not known to the authors.

Operating system	Last Updated	Free	GUI	Approx. number of packages pre-compiled	Size of Media	RAM
SLAX	2013	✓	KDE Plasma workplace	2050	200 MB	256
Knoppix	2013	✓	LXDE	3600	4.2 GB	256
Puppy Linux	2014	✓	JWM	128	161 MB	256
Gentoo Linux	2012	✓	$\star$	$\star$	3.9 GB	512
Xubunutu	2014	✓	Xfce	52,541	700 MB	512
Ubunutu	2015	✓	Unity	52,541	965 MB	512
Enterprise Linux	2014	×	GNOME	12,516	255 MB	512

matics application in DNALinux [6]. BioPuppy, (`http://www.biopuppy.org/`) built on Puppy Linux, is a very compact distribution addressed to a class room setting where the instructor aims to teach the students use of specific tools. Other Ubuntu based distributions like BioconductorBuntu [36] and PhyLIS [109] are tailored for a specific type of scientific analysis. For example, BioconductorBuntu provides a microarray processing platform, whereas, PhyLIS tries to fulfil the needs of Phylogenetics and Phyloinformatics. Table 5.1 provides details on Linux-based operating systems which are used to derive Life-Linux distributions (highlighted in Table 4.2).

5.1 Approach

Baari is a customized distribution of Ubuntu 13.10 operating system containing more than 70+ software and packages oriented towards Next Generation Sequencing. It uses Ubuntu's user friendly Unity Desktop environment to serve as its GUI. Installation can be done by booting it from DVD image or even a USB. Its current license allows free distribution with no restriction to use by non-academics as long as the use of Baari is acknowledged. As Baari is built on top of Ubuntu 13.04, its valid-

Operating system	Free	Reliable	Base OS	Software	Open source	LTS	GUI	Security	Threat detection	×86/×64	Cloud	Script Files
Baari	✓	✓	Ubuntu 13.10	70+ Genome Assembly tools	✓	✓	Unity	✓	✓	×64	×	✓
Bio Puppy	✓	✓	Puppy Linux	Bioinformatics applications for a class room setting	✓	×	X.org, Xvesa	×	×	×86	×	×
BioSLAX	✓	✓	SLAX	modualize almost and bioinformatics application	✓	✓	KDE X Window	✓	✓	×86	×	×
Lxtoo	✓	✓	Gentoo Linux 11	Sequence Analysis, Protien-Protien interactions	✓	✓	X11 Desktop	✓	✓	x86/×64	×	×
Open Discovery 3	×	✓	Fedora Sulphur 9	molecular dynamics, docking, sequence analysis	✓	✓	GNOME 2.22	✓	✓	×86/×64	✓	×
BioBrew	✓	✓	Red Hat 7.3	Cluster Software	✓	×	KDE, GNOME	✓	✓	×86	×	×
PhyLIS	✓	✓	Ubuntu 8	Phylogenetics	✓	×	Unity	✓	✓	×86/×64	×	×
DNALinux	✓	✓	Xubuntu	DNA and protein analysis. Also contains Virtual Desktop	✓	×	XFCE 4.2.2	✓	✓	×86	✓	×
Scientific Linux	✓	✓	Linux 6.4	For general audience, not specifically Biologists	✓	✓	GNOME	✓	✓	×86/×64	×	×
Bioconductor Buntu	✓	✓	Ubuntu 12.04	Bioconductor Buntu 2.11	✓	✓	Unity	✓	✓	×86/×64	×	×
Scubuntu	✓	✓	Ubuntu 9.1	For general audience not specifically Biologists	✓	✓	Unity	✓	✓	×86/×64	×	×
BioLinux 7	✓	✓	Ubuntu 12.04	500+ Bioinformatics application with 7 Assembly tools	✓	✓	Unity	✓	✓	×64	✓	×

Table 5.2: Comparison of Different Linux Distributions

ity, robustness and long-term-support (LTS) are already guaranteed by the Ubuntu community. Nonetheless, the stability of Baari was tested on a number of systems and Baari can run safely on hardware that supports an $\times 64$ OS.

In its current version, Baari supports pre-assembly tools, genome assemblers as well as post-assembly tools. Furthermore, a library of tools developed by the authors for NGS data and commonly used biological software have also been provided. Baari represents the first piece of work to produce an environment that can easily install a wide range of tools and software packages oriented towards genome assembly. It is free, easily distributable and facilitates building laboratories and software workstations both for personal use and for a college/university setting. Baari is under active development and will undergo periodic updates to incorporate the latest version of software. Users are requested to contact the author for additional software that could enhance the usage of Baari. The 'Installation Manual', 'Supplementary Section' and the distribution (Baari) itself are available at (`http://people.tamu.edu/~bilalwajidabbas/Baari.html`). For a list of pre-installed software see Tables 5.3 and 5.4.

In addition to various software packages, Baari also includes example script files for various assembly pipelines. Example shell script files for Velvet [129], SSAKE [15], VCAKE [48], SHARCGS [25], QSRA [10], IDBA [?], Maq (`http://maq.sourceforge.net/`), MARAGAP [123] and ABySS [103] have also been provided. These shell scripts may help in both teaching and research, and can be updated for more extensive assembly pipelines.

5.2 Addendum

Availability and requirements:

- Project name: Baari,

Table 5.3: Genome Assembly tools present in Baari v. 1.0

	Pre-assembly Tools		
1	Reads present in the Ref. Seq. [119]	4	HiTEC [45]
2	Minimum Description Length [116, 119, 124]	5	SOAPaligner [62]
3	SRA Toolkit [58]	6	FASTQC [98]
	Assemblers		
1	Velvet [129]	7	MARAGAP [123]
2	SSAKE [15]	8	Celera [75]
3	VCAKE [48]	9	ABySS [103]
4	IDBA [82]	10	TAIPAN [96]
5	Edena [40]	11	SOAPdenovo [63]
6	Maq (`http://maq.sourceforge.net/`)	12	IDBA-UD [83]
	Post-assembly Tools		
1	TABLET [71]	3	ReAligner [4]
2	JEMBOSS [13]	4	SOPRA [20]

Table 5.4: Common Life Sciences tools present in Baari v. 1.0

	Generic Biology Software Tools		
1	AmapAlign [108]	22	MAFFT [50]
2	BioPerl [106]	23	MassXpert [93]
3	BioRuby [37]	24	MCL [111]
4	Blast2 [49]	25	MUMmer [22]
5	Bowtie [54]	26	MUSCLE [28]
6	Bwa [59]	27	NCBI Libraries (`http://www.ncbi.nlm.nih.gov/IEB/ToolBox/`)
7	ClonalFrame [23]	28	NCBI-BLAST+ (`http://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP\_DOC/`)
8	Clustal W [61]	29	NCBI-ePCR (`http://www.ncbi.nlm.nih.gov/sutils/e-pcr/`)
9	Clustal X [55]	30	PerlPrimer [67]
10	DIALIGN2 [73]	31	PLink [85]
11	EMBOSS [88]	32	Primer3 [92]
12	Exonerate [104]	33	ProbCons [24]
13	FASTX-Toolkit	34	Samtools [60]
14	GenomeTools [57]	35	Sim4 [84]
15	GMAP [127]	36	ART NGS Reads Simulator [42]
16	HMMER [35]	37	Staden library [105]
17	Infernal [76]	38	T-Coffee [78]
18	JELLYFISH [66]	39	TIGR-Glimmer [21]
19	Jemboss [13]	40	TREE-PUZZLE [97]
20	Kalign [56]	41	UGENE [79]
21	LAST-align [51]	42	WISE2 (`http://www.ebi.ac.uk/~birney/wise2/`)

- Project home page: `http://people.tamu.edu/~bilalwajidabbas/Baari.html`,
- Operating systems: Ubuntu 13.10,
- License: GPL 3.0,
- Any restriction to use by non-academics: no, as long as Baari is acknowledged in their work,
- Contact: `bilalwajidabbas@hotmail.com`

6. GENOBUNTU: GENOME ASSEMBLY UBUNTU PACKAGE

In Chapter 4.5 we introduced Baari, a customized Ubuntu OS comprised of genome assembly tools. Baari presents an environment that can install a wide range of tools and software packages oriented towards genome assembly. It is free, easily distributable and facilitates building laboratories and software workstations. However, since Baari presents a customized Ubuntu OS, researchers who are already using Ubuntu OS or for that matter BioLinux do exhibit some hindrance in installing another OS on their machines. Therefore, we present Genobuntu, a software package containing more than 70 software and packages oriented towards genome assembly. Genobuntu can be installed on any Ubuntu $\times 32$ or $\times 64$ OS and is available at (`http://people.tamu.edu/~bilalwajidabbas/Genobuntu.html`).

6.1 Discussion

Genobuntu contains additional software and packages on top of the ones present in Baari (see Tables 5.3 and 5.4). Genobuntu includes pre-assembly tools, assemblers, post-assembly software and script files mentioned in Section 5.1. The additional software provided in Genobuntu (and not in Baari) are Allpaths [12], Celera [75], Euler [14,15] and SSPACE [8]. In its current version, Genobuntu has been tested on both Ubuntu 12.04 LTS and Ubuntu 13.04 OS. Although, Genobuntu can be installed on both $\times 32$ and $\times 64$ Ubuntu OS, a $\times 64$ architecture is preferred as a number of tools work better on a $\times 64$ system. Genobuntu was tested on 2.27 GHz Intel Core i5 CPU, 4 GB RAM with 100 GB of hard-drive space. It is recommended that Genobuntu be installed on a system with minimum 4 GB RAM and 100 GB hard-drive space. Some software may require additional memory and hard-disk space, and therefore, the author is not responsible for any software that misbehaves due to hardware limitations on part of the user's system. Genobuntu is freely available at (`http://people.tamu.edu/~bilalwajidabbas/Genobuntu.html`). Installation requires an active internet connection and instructions for installation are provided in the Supplementary Section present on the same website (`http://people.tamu.edu/~bilalwajidabbas/Genobuntu.html`).

6.2 Addendum

Availability and requirements:

- Project name: Genobuntu,
- Project home page: `http://people.tamu.edu/~bilalwajidabbas/Genobuntu.html`,
- Operating systems: any Ubuntu ×32 or ×64 system,
- License: GPL 3.0,
- Any restriction to use by non-academics: no, as long as Genobuntu is acknowledged in their work,
- Contact: `bilalwajidabbas@hotmail.com`

7. CONCLUSION

This dissertation highlighted the significance of genome assembly as a research area, examined key metrics, emphasized tools and outlined the whole pipeline for next-generation sequencing. It discussed principles of MDL in identifying the optimal reference sequence for genome assembly and MARAGAP for assembly pipeline. MARAGAP uses an algorithmic approach to detect and correct inversions and deletions, a De-Bruijn graph based approach to infer insertions, an affine-match affine-gap local alignment tool to estimate the locations of insertions and a Bayesian estimation framework for detecting SNPs (called BECA). BECA's framework is further extended by using Gibbs Sampling for further iterations of BECA. After each assembly the reference sequence is updated and the probabilistic score for each base call renewed. The revised reference sequence and probabilities are then further used to identify the alignments and consensus sequence, thereby, yielding an algorithm referred to as Gibbs-BECA. Gibbs-BECA further improves the performance both in terms of rectifying more SNPs and offering a robust performance even in the presence of a poor reference sequence.

Lastly, two software platforms Baari and Genobuntu are introduced. Baari and Genobuntu support pre-assembly tools, genome assemblers as well as post-assembly tools. Additionally, a library of tools developed by the authors for Next Generation Sequencing (NGS) data and commonly used biological software have also been provided in these software platforms. Baari and Genobuntu are free, easily distributable and facilitate building laboratories and software workstations both for personal use as well as for a college/university laboratory. Baari is a customized Ubuntu OS packed with the tools mentioned beforehand whereas Genobuntu is a software package containing the same tools for users who already have Ubuntu OS pre-installed on their systems.

REFERENCES

[1] Qammer Abbasi, Mohammad Monirrujjman Khan, Akram Alomainy, and Yang Hao. Radio channel characterisation and ofdm-based ultra wideband system modelling for body-centric wireless networks. In *Body Sensor Networks (BSN), 2011 International Conference on*, pages 89–94. IEEE, 2011.

[2] Afshin Ahmadian, Maria Ehn, and Sophia Hober. Pyrosequencing: history, biochemistry and future. *Clinica Chimica Acta*, 363(1):83–94, 2006.

[3] Afshin Ahmadian, Baback Gharizadeh, Anna C Gustafsson, Fredrik Sterky, Pål Nyrén, Mathias Uhlén, and Joakim Lundeberg. Single-nucleotide polymorphism analysis by pyrosequencing. *Analytical Biochemistry*, 280(1):103–110, 2000.

[4] Eric Anson and Eugene Myers. Realigner: a program for refining dna sequence multi-alignments. *Journal of Computational Biology*, 4(3):369–383, 1997.

[5] Jeff Augen. *Bioinformatics in the Post-Genomic era: Genome, Transcriptome, Proteome, and Information-based Medicine.* Addison-Wesley Professional, 2004.

[6] Sebastian Bassi and Virginia VC Gonzalez. Dnalinux virtual desktop edition. 2007.

[7] David Bentley, Shankar Balasubramanian, Harold Swerdlow, Geoffrey Smith, John Milton, Clive Brown, Kevin Hall, Dirk Evers, Colin Barnes, Helen Bignell, et al. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456(7218):53–59, 2008.

[8] Marten Boetzer, Christiaan Henkel, Hans Jansen, Derek Butler, and Walter Pirovano. Scaffolding pre-assembled contigs using sspace. *Bioinformatics*, 27(4):578–579, 2011.

[9] Keith Bradnam, Joseph Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, İnanç Birol, Sébastien Boisvert10, Jarrod Chapman, Guillaume Chapuis, Rayan Chikhi, et al. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *(Giga)^n science*, 2.

[10] Douglas Bryant, Weng-Keen Wong, and Todd Mockler. QSRA–a quality-value guided de novo short read assembler. *BMC Bioinformatics*, 10(1):69, 2009.

[11] Alexander Bulyshev, Serguei Semenov, Alexandre Souvorov, Robert Svenson, Alexei Nazarov, Yuri Sizov, and George Tatsis. Computational modeling of three-dimensional microwave tomography of breast cancer. *Biomedical Engineering, IEEE Transactions on*, 48(9):1053–1056, 2001.

[12] Jonathan Butler, Iain MacCallum, Michael Kleber, Ilya Shlyakhter, Matthew Belmonte, Eric Lander, Chad Nusbaum, and David Jaffe. Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome Research*, 18(5):810–820, 2008.

[13] Tim Carver and Alan Bleasby. The design of jemboss: a graphical user interface to emboss. *Bioinformatics*, 19(14):1837–1843, 2003.

[14] Mark Chaisson, Dumitru Brinza, and Pavel Pevzner. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Research*, 19(2):336–346, 2009.

[15] Mark Chaisson and Pavel Pevzner. Short read fragment assembly of bacterial genomes. *Genome Research*, 18(2):324–330, 2008.

[16] V. Chaitankar, P. Ghosh, E. Perkins, P. Gong, Y. Deng, and C. Zhang. A novel gene network inference algorithm using predictive minimum description length approach. *BMC Systems Biology*, 4(Suppl 1):S7, 2010.

[17] Vijender Chaitankar, Chaoyang Zhang, Preetam Ghosh, Edward Perkins, Ping Gong, and Youping Deng. Gene regulatory network inference using predictive minimum description length principle and conditional mutual information. In *Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS'09. International Joint Conference on*, pages 487–490. IEEE, 2009.

[18] Bastien Chevreux, Thomas Pfisterer, Bernd Drescher, Albert Driesel, Werner Müller, Thomas Wetter, and Sándor Suhai. Using the miraest assembler for reliable and automated mrna transcript assembly and snp detection in sequenced ests. *Genome Research*, 14(6):1147–1159, 2004.

[19] Peter Cock, Christopher Fields, Naohisa Goto, Michael Heuer, and Peter Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic Acids Research*, 38(6):1767–1771, 2010.

[20] Adel Dayarian, Todd Michael, and Anirvan Sengupta. Sopra: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11(1):345, 2010.

[21] Arthur Delcher, Kirsten Bratke, Edwin Powers, and Steven Salzberg. Identifying bacterial genes and endosymbiont dna with glimmer. *Bioinformatics*, 23(6):673–679, 2007.

[22] Arthur Delcher, Steven Salzberg, and Adam Phillippy. Using mummer to identify similar regions in large sequence sets. *Current Protocols in Bioinformatics*, pages 10–3, 2003.

[23] Xavier Didelot and Daniel Falush. Inference of bacterial microevolution using multilocus sequence data. *Genetics*, 175(3):1251–1266, 2007.

[24] Chuong Do, Mahathi Mahabhashyam, Michael Brudno, and Serafim Batzoglou. Probcons: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340, 2005.

[25] Juliane Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. Sharcgs, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research*, 17(11):1697–1706, 2007.

[26] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[27] John Dougherty, Ioan Tabus, and Jaakko Astola. Inference of gene regulatory networks based on a universal minimum description length. *EURASIP Journal on Bioinformatics and Systems Biology*, 2008:5, 2008.

[28] Robert Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[29] John Eid, Adrian Fehr, Jeremy Gray, Khai Luong, John Lyle, Geoff Otto, Paul Peluso, David Rank, Primo Baybayan, Brad Bettman, et al. Real-time dna sequencing from single polymerase molecules. *Science*, 323(5910):133–138, 2009.

[30] Emad El-Sebakhy, Kanaan Faisal, Tarek Helmy, Farag Azzedin, et al. Evaluation of breast cancer tumor classification with unconstrained functional networks classifier. In *AICCSA*, pages 281–287, 2006.

[31] S.C. Evans, A. Kourtidis, T.S. Markham, J. Miller, D.S. Conklin, and A.S. Torres. MicroRNA target detection and analysis for genes related to breast cancer using MDLcompress. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007:1–16, 2007.

[32] Robert Fano and David Hawkins. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794, 1961.

[33] Dawn Field, Linda Amaral-Zettler, Guy Cochrane, James Cole, Peter Dawyndt, George Garrity, Jack Gilbert, Frank Oliver Glöckner, Lynette Hirschman, Ilene Karsch-Mizrachi, et al. The genomic standards consortium. *PLoS Biology*, 9(6):e1001088, 2011.

[34] Dawn Field, Bela Tiwari, Tim Booth, Stewart Houten, Dan Swan, Nicolas Bertrand, Milo Thurston, et al. Open software for biologists: from famine to feast. *Nature Biotechnology*, 24(7):801–804, 2006.

[35] Robert Finn, Jody Clements, and Sean Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic Acids Research*, 39(suppl 2):W29–W37, 2011.

[36] Paul Geeleher, Dermot Morris, John P Hinde, and Aaron Golden. Bioconductorbuntu: a linux distribution that implements a web-based dna microarray analysis server. *Bioinformatics*, 25(11):1438–1439, 2009.

[37] Naohisa Goto, Pjotr Prins, Mitsuteru Nakao, Raoul Bonnal, Jan Aerts, and Toshiaki Katayama. Bioruby: bioinformatics software for the ruby programming language. *Bioinformatics*, 26(20):2617–2619, 2010.

[38] GJ Hannon. Fastx-toolkit, 2010.

[39] Vincent Henry, Anita Bandrowski, Anne-Sophie Pepin, Bruno Gonzalez, and Arnaud Desfeux. Omictools: an informative directory for multi-omic data analysis. *Database*, 2014:bau069, 2014.

[40] D. Hernandez, P. François, L. Farinelli, M. Østerås, and J. Schrenzel. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Research*, 18(5):802, 2008.

[41] Charles Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

[42] Weichun Huang, Leping Li, Jason Myers, and Gabor Marth. Art: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2012.

[43] David Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[44] Martin Hunt, Taisei Kikuchi, Mandy Sanders, Chris Newbold, Matthew Berriman, and Thomas Otto. Reapr: a universal tool for genome assembly evaluation. *Genome Biology*, 14(5):R47, 2013.

[45] Lucian Ilie, Farideh Fazayeli, and Silvana Ilie. Hitec: accurate error correction in high-throughput sequencing data. *Bioinformatics*, 27(3):295–302, 2011.

[46] Sorin Istrail, Granger Sutton, Liliana Florea, Aaron Halpern, Clark Mobarry, Ross Lippert, Brian Walenz, Hagit Shatkay, Ian Dew, Jason Miller, et al.

Whole-genome shotgun assembly and comparison of human genome assemblies. *Proceedings of the National Academy of Sciences of the United States of America*, 101(7):1916–1921, 2004.

[47] Anil Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[48] William Jeck, Josephine Reinhardt, David Baltrus, Matthew Hickenbotham, Vincent Magrini, Elaine Mardis, Jeffery L Dangl, and Corbin D Jones. Extending assembly of short dna sequences to handle error. *Bioinformatics*, 23(21):2942–2944, 2007.

[49] Mark Johnson, Irena Zaretskaya, Yan Raytselis, Yuri Merezhuk, Scott McGinnis, and Thomas L Madden. Ncbi blast: a better web interface. *Nucleic Acids Research*, 36(suppl 2):W5–W9, 2008.

[50] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma, and Takashi Miyata. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 2002.

[51] Szymon Kiełbasa, Raymond Wan, Kengo Sato, Paul Horton, and Martin Frith. Adaptive seeds tame genomic sequence comparison. *Genome Research*, 21(3):487–493, 2011.

[52] Gergely Korodi and Ioan Tabus. An efficient normalized maximum likelihood algorithm for DNA sequence compression. *ACM Transactions on Information Systems (TOIS)*, 23(1):3–34, 2005.

[53] Gergely Korodi, Ioan Tabus, Jorma Rissanen, and Jaakko Astola. DNA sequence compression-Based on the normalized maximum likelihood model. *Signal Processing Magazine, IEEE*, 24(1):47–53, 2006.

[54] Ben Langmead, Cole Trapnell, Mihai Pop, Steven L Salzberg, et al. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology*, 10(3):R25, 2009.

[55] Mark Larkin, Gordon Blackshields, Paul McGettigan, Hamish McWilliam, Franck Valentin, Iain Wallace, Andreas Wilm, Rodrigo Lopez, et al. Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[56] Timo Lassmann and Erik Sonnhammer. Kalign–an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics*, 6(1):298, 2005.

[57] William Lee and Swaine Chen. Research report genome-tools: A flexible package for genome sequence analysis. *BioTechniques*, 33(6):1334–1341, 2002.

[58] R. Leinonen, H. Sugawara, and M. Shumway. The sequence read archive. *Nucleic Acids Research*, 39(suppl 1):D19–D21, 2011.

[59] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[60] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[61] Kuo-Bin Li. Clustalw-mpi: Clustalw analysis using distributed and parallel computing. *Bioinformatics*, 19(12):1585–1586, 2003.

[62] Ruiqiang Li, Chang Yu, Yingrui Li, Tak-Wah Lam, Siu-Ming Yiu, Karsten Kristiansen, and Jun Wang. Soap2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15):1966–1967, 2009.

[63] Ruiqiang Li, Hongmei Zhu, Jue Ruan, Wubin Qian, Xiaodong Fang, Zhongbin Shi, Yingrui Li, Shengting Li, Gao Shan, Karsten Kristiansen, et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, 2010.

[64] Jun Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2008.

[65] Chengwei Luo, Despina Tsementzi, Nikos Kyrpides, Timothy Read, and Konstantinos Konstantinidis. Direct comparisons of illumina vs. roche 454 sequencing technologies on the same microbial community dna sample. *Plos One*, 7(2):e30087, 2012.

[66] Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 2011.

[67] Owen Marshall. Perlprimer: cross-platform, graphical primer design for standard, bisulphite and real-time pcr. *Bioinformatics*, 20(15):2471–2472, 2004.

[68] John McPherson, Marco Marra, LaDeana Hillier, Robert Waterston, Asif Chinwalla, John Wallis, Mandeep Sekhon, Kristine Wylie, Elaine Mardis, Richard K Wilson, et al. A physical map of the human genome. *Nature*, 409(6822):934–941, 2001.

[69] Joshua Mendell and Harry Dietz. When the message goes awry: disease-producing mutations that influence mrna content and performance. *Cell*, 107(4):411–414, 2001.

[70] Patrick Meyer, Kevin Kontos, Frederic Lafitte, and Gianluca Bontempi. Information-theoretic inference of large transcriptional regulatory networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007, 2007.

[71] Iain Milne, Micha Bayer, Linda Cardle, Paul Shaw, Gordon Stephen, Frank Wright, and David Marshall. Tabletnext generation sequence assembly visualization. *Bioinformatics*, 26(3):401–402, 2010.

[72] Patrice Milos. Helicos biosciences. 2008.

[73] Burkhard Morgenstern. Dialign 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3):211–218, 1999.

[74] Olena Morozova and Marco Marra. Applications of next-generation sequencing technologies in functional genomics. *Genomics*, 92(5):255–264, 2008.

[75] Eugene Myers, Granger Sutton, Art Delcher, Ian Dew, Dan Fasulo, Michael Flanigan, Saul Kravitz, Clark Mobarry, Knut Reinert, Karin Remington, et al. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204, 2000.

[76] Eric Nawrocki, Diana Kolbe, and Sean Eddy. Infernal 1.0: inference of rna alignments. *Bioinformatics*, 25(10):1335–1337, 2009.

[77] Thomas Niedringhaus, Denitsa Milanova, Matthew Kerby, Michael Snyder, and Annelise Barron. Landscape of next-generation sequencing technologies. *Analytical Chemistry*, 83(12):4327–4341, 2011.

[78] Cédric Notredame, Desmond Higgins, and Jaap Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217, 2000.

[79] Konstantin Okonechnikov, Olga Golosova, Mikhail Fursov, et al. Unipro ugene: a unified bioinformatics toolkit. *Bioinformatics*, 28(8):1166–1167, 2012.

[80] Ravi Patel and Mukesh Jain. Ngs qc toolkit: a toolkit for quality control of next generation sequencing data. *Plos One*, 7(2):e30619, 2012.

[81] Yanxiong Peng, Wenyuan Li, and Ying Liu. A hybrid approach for biomarker discovery from microarray gene expression data for cancer classification. *Cancer Informatics*, 2:301, 2006.

[82] Yu Peng, Henry Leung, Siu-Ming Yiu, and Francis Chin. Idba–a practical iterative de bruijn graph de novo assembler. In *Research in Computational Molecular Biology*, pages 426–440. Springer, 2010.

[83] Yu Peng, Henry Leung, Siu-Ming Yiu, and Francis Chin. Idba-ud: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, 28(11):1420–1428, 2012.

[84] Alison Pidoux, William Richardson, and Robin Allshire. Sim4 a novel fission yeast kinetochore protein required for centromeric silencing and chromosome segregation. *The Journal of Cell Biology*, 161(2):295–307, 2003.

[85] Shaun Purcell, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, Manuel Ferreira, David Bender, Julian Maller, Pamela Sklar, Paul De Bakker, Mark Daly, et al. Plink: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, 81(3):559–575, 2007.

[86] Antonia Rana and Fabrizio Foscarini. Linux distributions for bioinformatics: an update. *EMBnet. news*, 15(3):pp–35, 2009.

[87] Kathryn Renegar. Influenza virus infections and immunity: a review of human and animal models. *Laboratory Animal Science*, 42(3):222, 1992.

[88] Peter Rice, Ian Longden, and Alan Bleasby. Emboss: the european molecular biology open software suite. *Trends in Genetics*, 16(6):276–277, 2000.

[89] Teemu Roos. *Statistical and Information-Theoretic Methods for Data Analysis*. Citeseer, 2007.

[90] Allen Roses. Pharmacogenetics and drug development: the path to safer and more effective drugs. *Nature Reviews Genetics*, 5(9):645–656, 2004.

[91] Jonathan Rothberg, Wolfgang Hinz, Todd Rearick, Jonathan Schultz, William Mileski, Mel Davey, John Leamon, Kim Johnson, Mark Milgrew, Matthew Edwards, et al. An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, 475(7356):348–352, 2011.

[92] Steve Rozen and Helen Skaletsky. Primer3 on the www for general users and for biologist programmers. In *Bioinformatics methods and protocols*, pages 365–386. Springer, 1999.

[93] Filippo Rusconi. massxpert 2: a cross-platform software environment for polymer chemistry modelling and simulation/analysis of mass spectrometric data. *Bioinformatics*, 25(20):2741–2742, 2009.

[94] Frederick Sanger, Steven Nicklen, and Alan Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[95] Eric Schadt, Steve Turner, and Andrew Kasarskis. A window into third-generation sequencing. *Human Molecular Genetics*, 19(R2):R227–R240, 2010.

[96] B. Schmidt, R. Sinha, B. Beresford-Smith, and S.J. Puglisi. A fast hybrid short read fragment assembly algorithm. *Bioinformatics*, 25(17):2279, 2009.

[97] Heiko Schmidt, Korbinian Strimmer, Martin Vingron, and Arndt von Haeseler. Tree-puzzle: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18(3):502–504, 2002.

[98] Robert Schmieder and Robert Edwards. Quality control and preprocessing of metagenomic datasets. *Bioinformatics*, 27(6):863–864, 2011.

[99] Jan Schröder, Heiko Schröder, Simon Puglisi, Ranjan Sinha, and Bertil Schmidt. Shrec: a short-read error correction method. *Bioinformatics*, 25(17):2157–2163, 2009.

[100] Jonathan Schug and Christian Overton. Modeling transcription factor binding sites with gibbs sampling and minimum description length encoding. In *Ismb*, volume 5, pages 268–271, 1997.

[101] Barkur Shastry. Snps in disease gene mapping, medicinal drug development and evolution. *Journal of Human Genetics*, 52(11):871–880, 2007.

[102] Jay Shendure, Gregory Porreca, Nikos Reppas, Xiaoxia Lin, John McCutcheon, Abraham Rosenbaum, Michael Wang, Kun Zhang, Robi Mitra, and George Church. Accurate multiplex polony sequencing of an evolved bacterial genome. *Science*, 309(5741):1728–1732, 2005.

[103] Jared Simpson, Kim Wong, Shaun Jackman, Jacqueline Schein, Steven Jones, and Inanç Birol. Abyss: a parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009.

[104] Guy Slater and Ewan Birney. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics*, 6(1):31, 2005.

[105] Rodger Staden. The staden sequence analysis package. *Molecular Biotechnology*, 5(3):233–241, 1996.

[106] Jason E Stajich, David Block, Kris Boulez, Steven E Brenner, Stephen A Chervitz, Chris Dagdigian, Georg Fuellen, James GR Gilbert, Ian Korf, Hilmar Lapp, et al. The bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12(10):1611–1618, 2002.

[107] Ioun Tabus and Jaakko Astola. Clustering the non-uniformly sampled time series of gene expression data. In *Signal Processing and Its Applications, 2003. Proceedings. Seventh International Symposium on*, volume 2, pages 61–64. IEEE, 2003.

[108] Devarajan Thangadurai and Jeyabalan Sangeetha. Bioinformatics tools for the multilocus phylogenetic analysis of fungi. In *Laboratory Protocols in Fungal Biology*, pages 579–592. Springer, 2013.

[109] Robert Thomson. Phylis: a simple gnu/linux distribution for phylogenetics and phyloinformatics. *Evolutionary Bioinformatics Online*, 5:91, 2009.

[110] Zyl Van. Empowering african scientists-an investigation into a cd-based installer for scubuntu. 2008.

[111] Stijn Van Dongen. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141, 2008.

[112] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag New York Inc, 2000.

[113] Bala Murali Venkatesan and Rashid Bashir. Nanopore sensors for nucleic acid analysis. *Nature Nanotechnology*, 6(10):615–624, 2011.

[114] Craig Venter, Mark Adams, Eugene Myers, Peter Li, Richard Mural, Granger Sutton, Hamilton Smith, Mark Yandell, Cheryl Evans, Robert Holt, et al. The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001.

[115] Umashankar Vetrivel and Kalabharath Pilla. Open discovery: An integrated live linux platform of bioinformatics tools. *Bioinformation*, 3(4):144, 2008.

[116] Bilal Wajid, Roberto Aramayo, and Erchin Serpedin. Exploring Minimum Description Length and Probabilistic Distributions of the Reference Sequences for Comparative Assembly of Genomes. *Proceedings of International Conference on Genomic Signal Processing, Bucharest, Romania*, 2011.

[117] Bilal Wajid, Ali Riza Ekti, Amina Noor, Erchin Serpedin, Muhammad Naeem Ayyaz, Hazem Nounou, and Mohamed Nounou. Supersonic mib. In *Genomic Signal Processing and Statistics (GENSIPS), 2013 IEEE International Workshop on*, pages 86–87. IEEE, 2013.

[118] Bilal Wajid, Mohamed Nounou, Hazem Nounou, Muhammad Naeem Ayyaz, and Erchin Serpedin. Gibbs-beca: Gibbs sampling and bayesian estimation for comparative assembly. In *3rd International Conference on Biomedical Engineering, Electronics and Nanotechnology (MIC-BEN 2013)*, volume 3, pages 1–0. Mosharaka for Researches and Studies, 2013.

[119] Bilal Wajid and Erchin Serpedin. Minimum description length based selection of reference sequences for comparative assemblers. In *Genomic Signal Processing and Statistics (GENSIPS), 2011 IEEE International Workshop on*, pages 230–233. IEEE, 2011.

[120] Bilal Wajid and Erchin Serpedin. Supplementary information section: Review of general algorithmic features for genome assemblers for next generation sequencers. 2011.

[121] Bilal Wajid and Erchin Serpedin. Review of General Algorithmic Features for Genome Assemblers for Next Generation Sequencers. *Genomics, Proteomics & Bioinformatics*, 10(2):58–73, 2012.

[122] Bilal Wajid and Erchin Serpedin. Do it yourself guide to genome assembly. *Briefings in Functional Genomics*, page elu042, 2014.

[123] Bilal Wajid, Erchin Serpedin, Hazem Nounou, and Mohamed Nounou. Mib: A comparative assembly processing pipeline. In *Genomic Signal Processing and Statistics (GENSIPS), 2012 IEEE International Workshop on*. IEEE, 2012.

[124] Bilal Wajid, Erchin Serpedin, Mohamed Nounou, and Hazem Nounou. Optimal reference sequence selection for genome assembly using minimum description length principle. *EURASIP Journal on Bioinformatics and Systems Biology*, 2012(1):1–11, 2012.

[125] David A Wheeler, Maithreyan Srinivasan, Michael Egholm, Yufeng Shen, Lei Chen, Amy McGuire, Wen He, Yi-Ju Chen, Vinod Makhijani, G Thomas Roth, et al. The complete genome of an individual by massively parallel dna sequencing. *Nature*, 452(7189):872–876, 2008.

[126] Tom White. *Hadoop: the definitive guide*. O'Reilly, 2012.

[127] Thomas Wu and Colin Watanabe. Gmap: a genomic mapping and alignment program for mrna and est sequences. *Bioinformatics*, 21(9):1859–1875, 2005.

[128] Guangchuang Yu, Li-Gen Wang, Xiao-Hua Meng, and Qing-Yu He. Lxtoo: an integrated live linux distribution for the bioinformatics community. *BMC Research Notes*, 5(1):360, 2012.

[129] Daniel Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18(5):821–829, 2008.

[130] Xiujun Zhang, Xing-Ming Zhao, Kun He, Le Lu, Yongwei Cao, Jingdong Liu, Jin-Kao Hao, Zhi-Ping Liu, and Luonan Chen. Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information. *Bioinformatics*, 28(1):98–104, 2012.

[131] Tao Zhu, Jianfeng Zhou, Yuming An, Jinhua Zhou, Hongyu Li, Gang Xu, and Ding Ma. Construction and characterization of a rock-cluster-based est analysis pipeline. *Computational Biology and Chemistry*, 30(1):81–86, 2006.

[132] Albert Zomaya. *Parallel Computing for Bioinformatics and Computational Biology: Models, Enabling Technologies, and Case Studies*, volume 55. John Wiley & Sons, 2006.