Fall 2015

Efficient Algorithms for Prokaryotic Whole Genome Assembly and Finishing

Abhishek Biswas
Old Dominion University, abisw001@odu.edu

EFFICIENT ALGORITHMS FOR PROKARYOTIC WHOLE GENOME

ASSEMBLY AND FINISHING

by

Abhishek Biswas
B. E. August 2007, Visvesvaraya Technological University, India

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2015

Approved by:

_____
Desh Ranjan (Director)

_____
David Gauthier (Member)

_____
Mohammad Zubair (Co-Director)

_____
Jing He (Member)

ABSTRACT

EFFICIENT ALGORITHMS FOR PROKARYOTIC WHOLE GENOME ASSEMBLY

AND FINISHING

Abhishek Biswas
Old Dominion University, 2015
Director: Dr. Desh Ranjan
Co-Director: Dr. Mohammad Zubair

De-novo genome assembly from DNA fragments is primarily based on sequence overlap information. In addition, mate-pair reads or paired-end reads provide linking information for joining gaps and bridging repeat regions. Genome assemblers in general assemble long contiguous sequences (contigs) using both overlapping reads and linked reads until the assembly runs into an ambiguous repeat region. These contigs are further bridged into scaffolds using linked read information. However, errors can be made in both phases of assembly due to high error threshold of overlap acceptance and linking based on too few mate reads. Identical as well as similar repeat regions can often cause errors in overlap and mate-pair evidence. In addition, the problem of setting the correct threshold to minimize errors and optimize assembly of reads is not trivial and often requires a time-consuming trial and error process to obtain optimal results. The typical trial-and-error with multiple assembler, which can be computationally intensive, and is very inefficient, especially when users must learn how to use a wide variety of assemblers, many of which may be serial requiring long execution time and will not return usable or accurate results. Further, we show that the comparison of assembly results may not provide the users with a clear winner under all circumstances. Therefore, we propose a novel scaffolding tool, Correlative Algorithm for Repeat Placement

(CARP), capable of joining short low error contigs using mate pair reads, computationally resolved repeat structures and synteny with one or more reference organisms. The CARP tool requires a set of repeat sequences such as insertion sequences (IS) that can be found computationally found without assembling the genome. Development of methods to identify such repeating regions directly from raw sequence reads or draft genomes led to the development of the ISQuest software package. ISQuest identifies bacterial ISs and their sequence elements—inverted and direct repeats—in raw read data or contigs using flexible search parameters. ISQuest is capable of finding ISs in hundreds of partially assembled genomes within hours; making it a valuable high-throughput tool for a global search of IS and repeat elements.

The CARP tool matches very low error contigs with strong overlap using the ambiguous partial repeat sequence at the ends of the contig annotated using the repeat sequences discovered using ISQuest. These matches are verified by synteny with genomes of one or more reference organisms. We show that the CARP tool can be used to verify low mate pair evidence regions, independently find new joins and significantly reduce the number of scaffolds. Finally, we are demonstrate a novel viewer that presents to the user the computationally derived joins along with the evidence used to make the joins. The viewer allows the user to independently assess their confidence in the joins made by the finishing tools and make an informed decision of whether to invest the resources necessary to confirm a particular portion of the assembly. Further, we allow users to manually record join evidence, re-order contigs, and track the assembly finishing process.

This thesis is dedicated to Old Dominion University, which has funded my education and it is where I have lived for 6 years and made lot of friends.

ACKNOWLEDGMENTS

The writing of this dissertation was a very challenging academic experience for me and I would have walked away if not for the support, patience, guidance and expectations of the following people.

- I would like to thank **Dr. Desh Ranjan** who decided to supervise my doctorate degree despite his many other academic and administrative commitments. His capability of proposing interesting solutions to problems and proving correctness of the proposed solution always inspired and motivated me. I would like to thank him for taking care of me financially and finding sources for funding my education.

- I would like to thank **Dr. Mohammad Zubair** who decided to co-supervise my doctorate degree and share his experience and knowledge of high performance computing. His ability to ask fundamental questions to understand the given problem has helped me to develop a rational thought process. In addition, I would also like to thank him for taking care of me financially and finding sources for funding my education.

- I would like to thank **Dr. David Gauthier** for being a constant source of real world biological problems and solutions that have given my dissertation its purpose with immense practical value. His patience and intuitively explanations of complex biological processes have sparked in me a newfound interest in life sciences. His work with students has inspired me to be a better teacher and better academic. His excellent writing skills have helped me to improve the

quality of my publications.

- I would like to thank **Dr. Jing He** for encouraging me to work in another domain of bioinformatics and for including me in her team. She has helped me to improve my publishing record with major journal publication that would not have been otherwise possible. Her excellent writing skills have made this document possible and inspired me during moments of writer's block.

- I would like to extend special thanks to **Dr. Ravi Mukkamala and Ms. Yasoda Mukkamala** for inviting me to their home every Thursday and being a constant source of love, support and valuable advice.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF GRAPHS

CHAPTER 1

INTRODUCTION

1. Overview

Genome sequencing is the method of breaking multiple copies of the genome of an organism into many small fragments (reads) whose sequence can then be determined using a genome sequencer machine. The problem of combining these reads to reconstruct the source genome is known as whole genome assembly. The human genome project completed in 2003, primarily used a technique called Sanger dideoxynucleotide termination sequencing to accomplish the goal of determining all ~3 billion DNA bases of the human genome. This technology used thousands of dedicated sequencing instruments running around the clock and serviced by full-time technical staff. In 2005, newer technology, so called "Next-Generation Sequencing" (NGS) was introduced, with the result that the sequencing capacity of an entire building of Sanger sequencers could be replaced with a single machine roughly the size of a large laser printer. NGS technology has since advanced to the point where gigabases (Gb) of data can be produced in a matter of hours, and generating sequence data for small genomes (such as bacteria) can be performed in hours for less than $1000.

Despite this massive advance in technology, sequencing still has the fundamental limitation that relatively short (<1000 bp) sequences are produced, and these sequences need to be put back together to recreate the genome of interest.
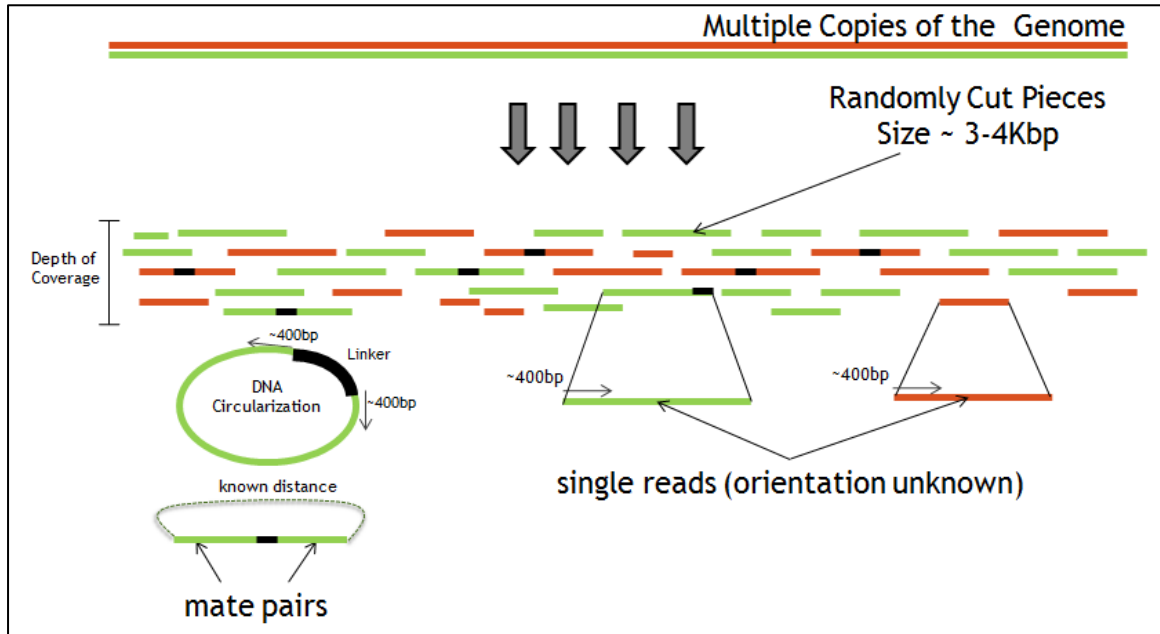
Fig. 1. Illustration of DNA sequencing

Next-generation sequencing technologies (e.g. Roche 454, Illumina®, Ion Torrent™, SOLiD™, etc.) provide unprecedented capacity for extremely high-throughput DNA sequencing relative to older Sanger-type methods. These methods are limited by size of individual reads (800bp, 454; 300bp, Illumina®; 400bp Ion Torrent™). However, these methods generate overlapping reads that cover the same portion of the genome many times over (see Figure 1). Therefore, *De novo* genome assembly from DNA reads is primarily based on overlapping sequence fragments (see Figure 2). The number of sequences covering a portion of the genome is called the coverage of the reads. In addition, mate-pair or paired-end reads can provide linking information for joining gaps and bridging problematic repetitive regions. This is done by generating sequence for two short reads that are a known distance apart in the genome.
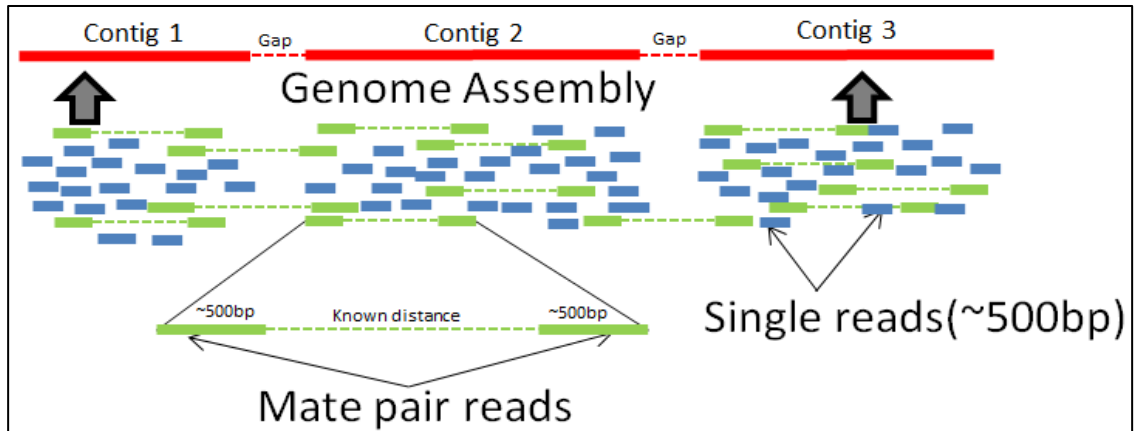
Fig. 2. Illustration of genome assembly process

A simplistic formulation of this problem, the Shortest Common Superstring (SCS), assumes that the original genome should be the shortest sequence that contains every fragment as a substring. Additional complexity arises when there are repeats i.e. there are multiple identical or nearly identical stretches of DNA in the original sequence and sequencing errors (see Figure 3). Generating a final genome entails correctly ordering the short sequence fragments and closure (joining) of all regions into a complete genome in presence of repeats and errors. Ambiguous and repeat elements are ubiquitous in all genomes, bacterial and eukaryotic, with the result that generating sequence data for a genome is quite simple, but reassembling the genome from these data can be quite challenging.

Several assemblers such as Celera WGS (Miller, *et al.*, 2008), MIRA (Chevreux, *et al.*, 1999), Newbler (Margulies, *et al.*, 2005) and ABySS (Simpson, *et al.*, 2009)have been developed to perform genome assembly from fragments; however, the effectiveness of these assemblers is impacted by the characteristics of the genome under assembly. For example, repetitive elements in genomes are well known to negatively affect assemblies.
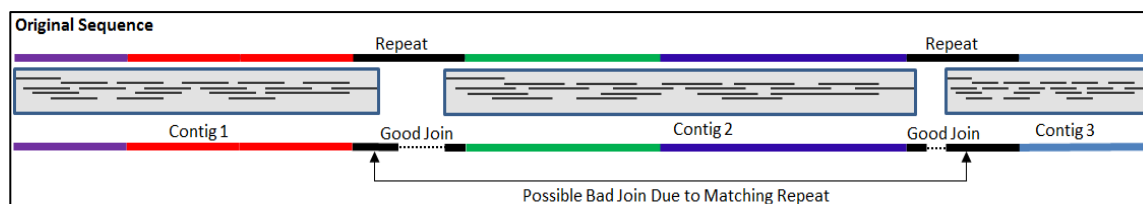
Fig. 3. The repeat problem and examples of "good" and "bad" joins

Moreover, assemblers may disagree on the assembly of a particular genome, even when working from the same fragment data, and certain assemblers have been shown to assemble some organisms better. Uncertainty in assembly accuracy is further complicated by lack of comprehensive measures for determining the quality of assembly. Even assembly of "simple" bacterial genomes, with very few repeat regions, usually results in multiple, unjoined large fragments that cannot be assembled automatically. These breaks in the assembly must be closed with relatively laborious PCR and Sanger sequencing methods, with the result that completing the last 5% of the final genome can often require significant time and expense.

When considering bacterial genomes published in public repositories such as GenBank, it is important to note that while a limited number are "final," and represented by one completed contiguous sequence (contig) of the bacterial chromosome, most are "draft" and composed of tens to thousands of unjoined contigs. Production of a final genome generally requires expensive PacBio® sequencing that generates long reads (up to 25,000bp). These long reads have high sequence error and cannot be used to directly assemble the genome accurately but are used to order the contigs assembled using Illimina reads that have high sequence fidelity. Further gap filling has to be done using older targeted PCR and Sanger sequencing techniques. Fragmented draft genomes are

still useful for many types of analyses, and can be used, for example, to generate genome-wide phylogenetic trees based on the presence of single nucleotide polymorphisms (SNPs) between strains. Many useful data are lost with this approach, however, including overall chromosomal arrangement and presence or absence of repetitive regions such as insertion sequences (IS) and phages (these are often excluded altogether from draft assemblies). Further, disruption of coding genes (such as via interruption by an IS) cannot be completely examined without a final genome, therefore relative analysis of bacterial metabolic capabilities is limited when using draft genomes.

We therefore developed an economical, user friendly, end-to-end computational pipeline for identifying insertion sequences and other repetitive elements, performing guided assembly of contigs around these elements, and producing more highly finished genomes from Illumina Paired-End data than have previously been possible. The goals of this approach are twofold: 1) to use computational methods to dramatically reduce the number of unresolved contigs resulting from standard sequence assembly, and 2) to provide a user-friendly framework for assessing the quality of a near-final genome and guiding gap-closure sequencing in the most efficient way possible. We propose a novel scaffolding tool, Correlative Algorithm for Repeat Placement (CARP), for computationally assembling and correctly placing repeat sequences in a genome from raw reads. Computational identification and assembly of the repeat elements is performed using a tool named ISQuest (Biswas, *et al.*, 2015) developed to provide CARP the required input data.

ISQuest uses BLAST search to identify reads belonging to known mobile elements. These reads are further assembled until unique sequence is encountered, and a

library of full and partial repeats is generated. We initially concentrate on finding insertion sequences and attempt to find all IS elements in a strain and map them based on a reference genome. The list of potentially interrupted genes is compiled from the above mapping to study large re-arrangements in the genome.

The scaffolding module using the assembled repeat regions is designed to join very low error contigs based on the assembled repeat elements placed correctly within the draft genome. The placement of the repeat elements is ensured using several lines of evidence such as: 1) presence of incomplete repeat element fragments on the ends of unjoined contigs, 2) mate-pair evidence, and 3) synteny (similarity in gene organization) with reference genomes. Importantly, any joins made by this method will be presented to the user along with the evidence used to make the joins. This will provide the end user with a much clearer picture of the likelihood of correctness of every join in a draft assembly, in order that the labor- and resource-intensive process of finishing via PCR amplification and Sanger sequencing can be made as efficient as possible by reducing attempts to join misassembled regions. Therefore, users can independently assess their confidence in the joins made by the tool.

The pipeline makes generation of near-final bacterial genomes accessible to smaller laboratories for which sequencing resources are more limited than major sequencing centers, and will thus make prokaryotic genomics accessible to a wider user base.

2. Thesis Statement

Our analysis of the assembly problem has revealed that different assemblers can generate different assemblies given the same data. These assemblers can make mistakes, which

can lead to very time-consuming and expensive trial-and-error when it comes to finalizing the genome as assemblers may take hours to complete an assembly. Further, there are few tools available that allow quick and intuitive comparison among assemblies, therefore one is often left to guess as to which assembly was "best," and more importantly, which joins in the assembly are "good", "bad" or "acceptable" for further analysis. Further, there are currently no adequate tools for intuitive and convenient visualization of draft genomes, which would assist users in the final assembly process and track joins that have to be manually verified before publication.

We therefore explored three major areas of research:

a. We explore a suite of quality measures for comparison of assemblies and assessment of accuracy and reliability of sequence assemblies.

b. We design and develop a parallel framework to for speeding up bacterial whole genome assembly and implementing it for a serial assembler so that at the quality of the assembler can be analyzed under various input parameters.

c. We develop a suit of intuitive tools for generation of draft genomes and guidance in joining of final sequences.

3. Thesis Organization

The thesis document is organized as follows:

a. Chapter 2 provides a detailed literature survey of the related works in the area of genome assembly and finishing. Relevant work on finding insertion sequences and other mobile genetic elements is also included.

b. Chapter 3 states the genome assembly problem in detail and provides a survey of assembly quality of popular assemblers using various assembly quality metrics.

The design and implementation details of an efficient parallel framework for assembly are provided along with results showing significant assembly speedup.

c. Chapter 4 describes the Correlative Algorithm for Repeat Placement (CARP) genome-finishing algorithm proposed in this thesis. The ISQuest tool designed to assemble the biologically significant genomic repeats from raw fragment sequence data is presented. The CARP algorithm steps are discussed in detail and results showing improved assemblies are presented.

d. Chapter 5 provides a concluding discussion on the utility and benefits of tools developed and presented in this dissertation.

CHAPTER 2

BACKGROUND AND RELATED WORK

1. Genome Assembly

The development of new genome assembly software is being driven by the emergence and evolution of sequencing technologies generating reads with significantly different lengths, overlap lengths and error characteristics. The first popular sequencing technology was based on the chain-terminating inhibitor method by Sanger *et al.* (Sanger, *et al.*, 1977). The technique was automated with a computer and fluorescence detection and generates low error reads over 1000bp in length (Smith, *et al.*, 1986). The assembly programs to assemble first generation sequences were based on greedy algorithms (Tarhio and Ukkonen, 1988) or the overlap-layout-consensus (OLC) graph model (Kececioglu and Myers, 1995). The prominent assemblers used to assemble drosophila and human genomes include Phrap (Green, 1996), Celera (Myers, *et al.*, 2000) and ARACHNE (Batzoglou, *et al.*, 2002).

The next generation sequencing technologies with massively-parallel flow-cell sequencing and sequencing-by-synthesis generate a large number of reads with shorter lengths and higher error than Sanger, but which are significantly more economical. Roche 454 (Margulies, *et al.*, 2005) can currently generate read lengths less than 800 bp, and Ion Torrent^{TM} (Rothberg, *et al.*, 2011) generates read lengths less than 400 bp, with longer reads projected in the future. Illumina (Quail, *et al.*, 2008) and ABI SOLiD (Pandey, *et al.*, 2008) are short read sequencers with typical read lengths less than 300bp. The second-generation sequencing technologies have also developed the capability to read from both ends of a fragment and produce reads with a pair at approximate distance. This

approach ranges from short-range (<1kb) paired ends (Illumina) to very long-range (>10 kb) mate-paired reads typically implemented in 454 sequencing. Paired reads have been shown to be sufficient for de novo assembly (Chaisson, *et al.*, 2009), although assembly problems persist when repeat elements are present. The read lengths of short-read sequencers are not expected to increase drastically and algorithms have been developed to handle large quantities of short sequence data. Additionally, error correction algorithms have been designed to improve assembly quality (Yang, *et al.*, 2012). Parallel implementations of various phases of the assembly algorithms have been developed to handle these large datasets efficiently. A popular model based on de-Bruijn graphs has been accepted by assembler developers for its ability to model repeat structure of genomes. The de-Bruijn graph model groups the reads into shorter stretches of length k (called k-mers) and representing each read as a path in the graph (Idury and Waterman, 1995). This model was improved by graph reduction to untangle the loops in the graph and model the graph traversal as an Eulerian walk (Pevzner, *et al.*, 2001). Major short read assemblers include Trinity (Grabherr, *et al.*, 2011), Velvet (Zerbino and Birney, 2008), ABySS (Simpson, *et al.*, 2009), ALLPATHS (Butler, *et al.*, 2008), SHORTY (Hossain, *et al.*, 2009) and Ray (Boisvert, *et al.*, 2010). ABySS and Ray are parallel implementations of this model.

Efficient implementations of the OLC graph model are also very popular for next generation genome assembly particularly to handle whole prokaryotic genomes. Major open source assemblers include Celera assembler (Pauchet, *et al.*, 2009), Arachne (Batzoglou, *et al.*, 2002) and MIRA (Chevreux, 2005). The OLC graph model was implemented for assembly of Roche/454 reads and the sequencer is distributed with

Newbler (Pauchet, *et al.*, 2009). A recently developed assembler based on this model is EDENA (Hernandez, *et al.*, 2008) and is capable of assembling short reads (35 bases). Parallel implementation of OLC model has been mostly limited to the overlap and layout phases of the process. However, a full parallel version of MIRA has been implemented (Biswas, *et al.*, 2013). A memory efficient representation of the OLC graph model uses string graphs (Myers, 2005). The String Graph Assembler(SGA) (Simpson and Durbin, 2012) implements distributed construction of FM-indices (Simpson and Durbin, 2010) used to represent the reads in the string graph and perform graph operations like overlap construction on the FM-index values instead of the reads, thus reducing memory footprint of the assembler. A parallel framework for string graph assembler has been proposed (Jackson, *et al.*, 2010).

"Third generation" sequencing machines capable of long- to very-long reads are in development but not yet commercially available, with the exception of the Pacific Biosciences. This instrument produces long sequences (e.g., median > 2kbp, maximum = 25kbp) and supports short turn-around time (Eid, *et al.*, 2009), however current data indicates this instrument suffers from low (81-83%) accuracy (Chin, *et al.*, 2011). The low accuracy of the data requires error correction before assembly and OLC model of assembly seems to be most appropriate (Koren, *et al.*, 2012). Assemblers supporting assembly of PacBio reads include Celera (Koren, *et al.*, 2012), ALLPATHS-LG (Gnerre, *et al.*, 2011) and MIRA (Chevreux, *et al.*, 1999). A detailed description of the assembly techniques and the history of their various implementations can be found in (Imelfort and Edwards, 2009; Miller, et al., 2010).

2. Genome Assembly Quality

The selection of the assembler is largely guided by the sequencing technology used to obtain the reads. However, considering the various assemblers available for each sequencing generation the selection process is not trivial and is generally based on guesswork and multiple assembly trials. The fundamental theoretical relationship between the input factors like read length, coverage, repeat lengths, mate distance etc. and the assembly problem has been developed (Nagarajan and Pop, 2009). Experimental results often show that certain assemblers perform better on some datasets and it is not easy to declare a clear winner (Lin, *et al.*, 2011; Narzisi and Mishra, 2011; Zhang, *et al.*, 2011). Certain inferences may be drawn from empirical data but the set of significant input parameters that determine the assembly quality generated by an assembler is not known. On the other hand metrics for assessing quality of an assembly and comparison of different assemblies have been extensively studied. The GAGE (Salzberg, *et al.*, 2011) assembler comparison attempts to provide some empirical assessment of assembly quality for some input datasets. The amosvalidate tool uses five basic characteristics to validate an assembly by measuring the goodness of fit of the input data and assembly output (Phillippy, *et al.*, 2008). The Assemblathon 1 (Earl, *et al.*, 2011) is a proposed annual assembly competition and lists an extensive list of assembly quality parameters for judging the best assembly. In presence of a reference genome or genome of a related organism a reference mapping can be performed using software like MUMMER 3 (Kurtz, *et al.*, 2004), progressiveMauve (Darling, *et al.*, 2010) and BLAST (Altschul, *et al.*, 1990). Comparing assembler quality requires studying the tradeoffs between various

quality measures and Feature-Response curves (FRC) have been proposed to account for such relationships (Narzisi and Mishra, 2011; Narzisi and Mishra, 2011). The impact of the various input parameters on the assembly quality metrics seems to be an open problem whose solution is vital in appropriate selection of the assembler for a project.

Various parameters of the given data can be used to compare assemblers. Read lengths have been shown to significantly affect the assembly quality. A study of the best possible assembly quality using short reads of size varying from 25bp to 1000bp is presented in (Kingsford, *et al.*, 2010). This work measured the complexity of the final assembly graph for 375 organisms and empirically derived an upper bound on the achievable assembly quality. The relationship between read lengths and the resolution of repeats and the expected number of gaps is explored in (Cahill, *et al.*, 2010). This work provided a measure of expected number of contigs, gaps and their sizes. The inherent repeat structure of a genome is an important input parameter as it is the property of the organism and not of the technology used to sequence the genome. Various techniques have been proposed to detect repeats and repeat families in complete and partial genomes. Though various models and parameters have been proposed to express the repeat structure of the genome, profiles have not been developed to classify the assemblers based on their capability to handle these repeat models. Two algorithms for derivation of repeat structure from a partially assembled genome are proposed in (Quitzau and Stoye, 2008). A repeat classification algorithm and a model for representing longer repeats as an overlay of sub-repeats is proposed in (Pevzner, *et al.*, 2004). The RepeatGluer algorithm identifies the repeats and generates their consensus sequence and copy number. A theoretical measure to estimate the repeat structure, DNA length, is

proposed by (Li and Waterman, 2003) using parameters derived from the input reads like coverage, nucleotide distribution and l-tuples. Finally, repeat sequence family detection in complete genomes (Bao and Eddy, 2002; Price, *et al.*, 2005) classify repeats based on length and frequency into various repeat elements.

3. Genome Assembly Finishing

Most assemblers generate a set of contiguous non-overlapping sequences covering some part of the genome. These contigs are ordered and oriented through the process of scaffolding to generate a gapped representation of the genome. Scaffolding algorithms can use mate pair information of the reads at the ends of a contig to join it to other contigs. Joining can also be done by mapping the contigs to a reference genome or by inspecting other assemblies and checking for possible joins missed by the assembler. Some of the assemblers like Celera WGS are capable of utilizing mate pair data for scaffolding. Other tools for scaffolding include Bambus (Pop, *et al.*, 2004), SUPERCONTIGS (Puiu, 2004) and Autofinish (Gordon, *et al.*, 2001).

4. Genomic Repeat Finding

High-throughput sequencing methods allow generation of large amounts of sequence data making the annotation process the bottleneck for genomic research. In addition to open reading frames (ORFs) and regulatory elements, correct annotation and regulatory elements, correct annotation of other features such as mobile genetic elements (MGEs) is also essential. These MGEs include bacteriophages, conjugative transposons, integrons, unit transposons, composite transposons and insertion sequences (ISs). Such transposable elements are defined as specific DNA segments that can repeatedly insert

into one or more sites in one or more genomes. ISs are transposable elements that are regarded as genomic parasites proliferating in their host and surviving only through horizontal gene transfer (Schaack, *et al.*, 2010). ISs play a major role in genome evolution and plasticity, mediating gene transfers and promoting genome duplication, deletion and rearrangement (Frost, *et al.*, 2005). Insertion sequences may be abundant in host genomes and are intimately involved in mediating horizontal gene transfer, generation of pseudogenes, genomic rearrangement and alteration of regulatory elements (Frost, *et al.*, 2005; Schaack, *et al.*, 2010).

The abundance and diversity of MGE elements in prokaryotic genomes poses significant challenges in automated identification and annotation using computational methods. The ISFinder database is currently the most comprehensive dedicated resource for high-quality, manually curated ISs annotations (ISFinder at https://www-is.biotoul.fr/). Therefore, we assume this database to be an accurate set of ISs, but incomplete because genomes are being sequenced faster than they are annotated to this extent. However, several studies have used the referenced sequences in the ISFinder database to mine various collections of genomic data using BLAST-based software (Cerveau, *et al.*, 2011; Filée, *et al.*, 2007; Leclercq and Cordaux, 2011; Mahillon and Chandler, 1998; Wagner, 2006).

The development of high-throughput sequencing techniques has led to the availability of thousands of sequenced genomes and metagenomes that require automated identification of ISs. Genome annotation pipelines such as Prokka (Seemann, 2014) and Manatee (Ablordey, *et al.*, 2005) stop at the point of labeling ORFs as 'transposase' or 'integrase' where sufficient homology is observed. Without classification of ISs into

families and enumeration within genomes, broad-scale comparison studies across closely related strains are not possible. The first automated approach to annotate ISs was used for an analysis of 19 cyanobacterial and 31 archaeal genomes, but this has yet to be made publicly available as an automated pipeline (Zhou, *et al.*, 2008). ISSaga is a web application pipeline that allows semi-automated IS annotation in complete genomes (Varani, *et al.*, 2011). ISSaga employs a library-based method using BLAST seeded with the ISFinder sequences to classify ORFs into IS families. Although ISSaga represents significant progress in automated IS annotation, the efficiency of this approach in identifying transposable elements is questionable due to its dependency on the ISFinder database; ISSaga cannot automatically identify novel ISs not already present in ISFinder. IScan is a publicly available application that makes use of BLAST with a single reference transposase sequence per IS family to scan whole genomes for ISs, and includes in its prediction pipeline searches for transposases and inverted and direct repeats (Wagner, *et al.*, 2007). IScan was used to investigate ISs in 438 prokaryotic genomes and found a limited number of ISs in most taxa (Wagner and de la Chaux, 2008). OASIS, or Optimized Annotation System for Insertion Sequences, is another publicly available computational tool for automated annotation of ISs (Robinson, *et al.*, 2012) in whole genomes. OASIS takes advantage of widely available transposase annotations to identify candidate ISs and then uses a computationally efficient maximum likelihood method of multiple sequence alignment to identify the edges of each element. Although OASIS is capable of predicting IS families, this functionality seems to be deprecated in the current version of the software. Through comparisons across 1319 genomes to a benchmark of ISFinder annotations, OASIS detected 37,427 ISs while IScan (Wagner, *et al.*, 2007)

detected only 2902 ISs.

Software tools have also been developed to predict IS sequences and families based on profile-sequence comparisons. These tools employ Hidden Markov Models (HMMs) based on transposases of characterized IS families. HMMs have been generated for transposases belonging to 19 characterized families of ISs in the PFAM database (Finn, *et al.*, 2014). The Superfamily database of structural and functional annotation of genomes currently hosts 6 HMM profiles from domains belonging to two prokaryotic families of transposases: mu bacteriophage transposase and IS200 (Gough and Chothia, 2002). The TnpPred web service provides profile HMMs for the remaining IS families and improves on the accuracy of the HMMs in the PFAM database (Riadi, *et al.*, 2012). Effective prediction of ISs and Miniature Inverted repeat Transposable elements (MITEs) using HMMs has been shown for 30 archaeal genomes (Kamoun, *et al.*, 2013), demonstrating that HMM-based predictions can augment BLAST-based sequence-sequence IS search methods to improve accuracy and find novel ISs.

The current software tools described above operate only on complete genomes with fully annotated ORFs. Complete genome assembly of a single strain of bacteria can be time-consuming and costly, and draft genomes or raw read sets are increasingly used for comparative genomics studies of prokaryotes. Here, we present the ISQuest tool for global investigation of ISs in unassembled or partially assembled prokaryote genomes.

The impact of the various input parameters on the assembly quality metrics seems to be an open problem whose solution is vital in appropriate selection of the assembler for a project. Comprehensive end-to-end genome assembly packages capable of assembling various sequencing reads are freely available for users to download and

install. Perhaps the most popular assembler is Celera WGS (Miller, *et al.*, 2010), which is capable of handling large number of reads from various sequencing machines. The Celera assembler in conjunction with the AMOS (Koren, *et al.*, 2012; Treangen, *et al.*, 2002) analysis package form a complete genome assembly package. A similar package designed specifically for prokaryotic genomes provides assembly capability with automated result analysis and gene annotation (Kislyuk, *et al.*, 2010). This package assembles the data using a small set of assemblers and selects the best assembly based on certain quality metrics. These assembly packages are, however, not capable of selecting an appropriate assembler based on the input characteristics of the dataset. In many cases, there is no clear winner in terms of standard assembly quality metrics. For example, an assembler may generate an assembly with very short contigs, which are all correct, but the assembly is too fragmented to be useful to the user while another assembler generated long useful contigs with some misassembles. The tool proposed here requires the user to assemble the read libraries using an assembler with strict thresholds to ensure no assembly errors. The proposed novel scaffolding tool, Correlative Algorithm for Repeat Placement (CARP) (Biswas, *et al.*, 2013), is capable of joining short low error contigs using mate pair reads, computationally resolved repeat structures and synteny with one or more reference organisms (Galardini, *et al.*, 2011). The CARP tool requires a set of repeat sequences such as insertion sequences (IS) that can be found computationally found without assembling the genome. Development of methods to identify such repeating regions directly from raw sequence reads or draft genomes led to the development of the ISQuest software package (Biswas, *et al.*, 2015). ISQuest identifies bacterial ISs and their sequence elements—inverted and direct repeats—in raw read data

or contigs using flexible search parameters. ISQuest is capable of finding ISs in hundreds of partially assembled genomes within hours, making it a valuable high-throughput tool for a global search of IS and repeat elements.

The CARP tool matches very low error contigs with strong overlap using the ambiguous partial repeat sequence at the ends of the contig annotated using the repeat sequences discovered using ISQuest. These matches are verified by synteny with genomes of one or more reference organisms. We show that the CARP tool can be used to verify low mate pair evidence regions, independently find new joins and significantly reduce the number of scaffolds. Finally, we are demonstrate, Unverified Join Viewer (UJV) (Biswas, *et al.*, 2015), a novel viewer that presents to the user the computationally derived joins along with the evidence used to make the joins. The viewer allows the user to independently assess their confidence in the joins made by the finishing tools and make an informed decision of whether to invest the resources necessary to confirm a particular portion of the assembly. Further, we allow users to manually record join evidence, re-order contigs, and track the assembly finishing process. The UJV finishing tool allows the user to track analyses PCR finishing (Kislyuk, *et al.*, 2010; Steve Rozen, 1998; Ye, *et al.*, 2012) of the current assembly. This tool is expected to reduce the time spent by biologists on end-to-end assembly, assembly analysis and computational finishing from months to a few days.

CHAPTER 3

GENOME ASSEMBLY AND ASSEMBLY QUALITY

1. Overview

The whole-genome assembly problem has been a center of significant research in the last 20 years. Assembly of a genome using the data available from genome sequencing processes is an NP-hard problem (shortest superstring problem (Kececioglu and Myers, 1995)) even in the absence of errors. Four major assembly modeling techniques have been proposed to solve the problem of combining short sequence reads to reconstruct the source DNA. Graph-based representation of the genome assembly problem has resulted in three models. The OLC model (Kececioglu and Myers, 1995) represents each read as a vertex in a graph connected by edges, weighted by their pairwise alignment scores. The assembly algorithm seeks to find a path in this graph such that all the nodes are included only once in the assembled sequence. A disadvantage of this method is that repeat sequences (identical or nearly identical stretches of DNA) can be collapsed and cause misassembled joins resulting in rearrangement of large genome fragments. The de-Bruijn graph model (Pevzner and Tang, 2001) groups the reads into shorter stretches of length k (called k-mers) and representing each read as a path in the graph. The assembly can then be represented as a superpath, a path that includes all of the input paths. Since an edge can be traversed multiple times, repeat sequences are not compressed during assembly. An alternative model for of sequence assembly uses string graphs (Myers, 2005). An overlap graph is built where nodes correspond to reads and edges correspond to overlaps. The shortest walk that includes all of the required edges represents the assembly. The assembly of very short read sequencers has been modeled as greedy algorithm using

index tables for faster assembly (Whiteford, *et al.*, 2005). Based on these techniques, over 30 assemblers have been developed. A major problem is that these assemblers do not agree on the assembly and certain assemblers have been shown to assemble some organisms better, but fail for others. Therefore, selection of an assembler for a particular project is an important task in itself. This task is non-trivial for a typical life science researcher who may not have a great deal of expertise in computing or access to resources or to determine in a reasonable time the accuracy of assembly produced by an assembler. Frequently, assemblers are customized to assemble reads generated from a certain sequencing technologies and the sequencing technology is the first parameter considered for assembler selection. Other parameters include coverage, uniformity of coverage, read lengths, GC-ratio, and repeat structure and frequency. These parameters of the input reads are properties of the sequencing technology or the original sequence and must be correlated to the assembly results of the assembler. Real-life genomes contain repeats of various lengths, making it unlikely that any assembler will reproduce the original complete genome. The heuristic algorithms for contig assembly (contiguous assembly of reads) are greedy by design as searching for the overall best read to assemble into a contig is computationally intractable even in absence of errors. Therefore, all the algorithms optimize a cost function such as overlap score to select the next read for assembly. For example, MIRA assembler builds a pairwise overlap graph with edge weights scoring the overlap. The pathfinder algorithm finds paths in this graph starting from high density low error start nodes and constructs the contigs. Celera assembler first eliminates reads that are substrings of other reads and then builds a best overlap graph. This graph is then traversed to find contigs and other reads aligned to the contig to get the

consensus. While both these assemblers are based on the OLC model various error thresholds and internal statistics calculation for error correction and consensus generation are different between assemblers and contribute to different assemblies.

2. Comparing Genome Assembly Quality

Next-generation sequencing technologies (e.g. 454, Illumina, Ion TorrentTM, SoLiD, etc.) provide unprecedented capacity for extremely high-throughput DNA sequencing relative to older Sanger-type methods. Like Sanger sequencing, however, these methods are limited by size of individual reads (800bp, 454; 300bp, Illumina; 400bp Ion TorrentTM), thus organismal genomes must be sequenced in fragments, rather than as a continuous molecule. The problem of combining sequence fragments to reconstruct the source genome is known as sequence (or genome) assembly. Several assemblers have been created to perform genome assembly from fragments; however, the effectiveness of these assemblers is impacted by the characteristics of the genome under assembly. Complete computational assembly of genomes is rare and assemblers generally generate a set of long contiguous sequences (contigs), which are disjoint portions of the genome, cannot be further joined. For example, repetitive elements in genomes are well-known to negatively impact assemblies as they represent ambiguous joins that are difficult to computationally join. Also, assemblers may disagree on the assembly of a particular genome, even when working from the same fragment data, and certain assemblers have been shown to assemble some organisms better than others. Uncertainty in assembly accuracy is further complicated by lack of comprehensive measures for determining the quality of assembly. Two commonly used assembly quality metrics are N50 score and CE statistic. N50 score is the length of the longest contig such that half of

the sequence fragments belong to longer contigs and CE statistic is the number of standard deviations the average local mate pair lengths differ from the global mean. Such quality characteristics like N50 score and CE statistic are not always conclusive in determining the best assembly. N50 scores, in particular, may be misleading, as they reflect only the length of assemblies, ignoring the fact that increased length may result from misassembly of fragments. CE statistics also may be satisfied by a poor quality assembly of short contiguous sequences that do not correctly assemble long repeat regions. Therefore, to study the correlation between input and output characteristics of assemblers we focus on output parameters derived from comparing the assembled contigs to the original sequences. In this study, we propose to answer the following questions. (a) What characteristics of a genome sequence and the sequenced read fragments make one assembler more suitable than others? (b) How do we know that a sequence assembler is generating a "good assembly" (i.e. faithful to the original sequence)? (c) Can we provide a simple yet effective model to estimate the expected error of an assembly for selection of the most appropriate assembler for a given genomic sequence?

Studying the assembly quality of genome assemblers to determine the correctness of assembly and achieve optimal assembly, reducing the need for expensive genome finishing, is of great interest to biologists. Broadly speaking, we focus on the following aspects, namely, (1) on investigating the assembly characteristics of an assembler as a whole or (2) on investigating the relationship between the input parameters and the assembly quality generated by the assembler. The first study is useful for comparison of assemblers and selection of the appropriate assembler. Likewise, the second study is useful for various purposes such as deciding on the sequencing

technology, determining the parameters such as read lengths, coverage and mate pair distances of the input fragments. The input parameters are classified into two categories - (1) Genome fragmentation parameters: read length, coverage and mate pair distances (2) Genome sequence parameters: repeat length, repeat frequency and insertion sequences. The assembly quality metrics used to assess correctness of assembly are also classified into two categories - (1) Metrics measured by direct comparison to the original sequence such as misassembled contigs and correctly assembled repeat areas (2) Metrics measured by testing the fit of input data to assembled contigs such as mate pair consistency and error rates of assembled reads.

The first big data challenge is the generation of the simulated read libraries with various input parameters varied to cover the spectrum of values obtained from major sequencers available to biologists today. To generate reads for experimentation we developed a simulator for generating read libraries. Earlier sequencing simulation techniques, such as Genfrag by (Engle and Burks, 1994) and CelSim (Myers, 1999) concentrated on shotgun data, and MetaSim (Richter, *et al.*, 2008) and Flowsim (Balzer, *et al.*, 2010) simulated data from 454 pyrosequencing process. Generating a simulator based on an empirical distribution is a better fit, we developed a fast simulator, that applies a parametric log normal distribution to simulate the shotgun process based on user specified read length and standard deviation. Quality values however are estimated from a position specific error function based on the read length and base type similar to (Balzer, *et al.*, 2010). The simulator allows us to quickly generate read libraries for assembly and allows us to vary certain basic fragmentation parameters such as read lengths, coverage and mate pair distances.

Table 1. MIRA assembly time breakup

Organism	Reads[a]	Total Time	Graph Const[b]	Path Finder[c]	Cons.[d]	Error Corr.[e]
M. Marinum	500,000	624	413	103	86	22
	1,000,000	1,327	867	261	146	53
E. Coli	500,000	589	342	132	79	36
	1,000,000	959	612	192	113	42
M. Tuberculosis	500,000	581	374	96	84	27
	1,000,000	1,123	712	219	130	62
Average %			63.51	19.05	12.65	4.68

[a]The number of simulated reads with mean length of 600bp and standard deviation of 100bp.
[b]The time (in minutes) to construct the assembly graph.
[c]The time (in minutes) to find all the paths in the graph and assemble the contigs.
[d]The time (in minutes) to construct the consensus sequence of the contigs.
[e]The time (in minutes) to error correct the contigs in the assembly.

The biggest computational challenge is the assembly of the simulated read libraries generated. Most assemblers take a long time to work with large number of sequences, for example, it takes around 18.3 hours to assemble a dataset with 1 million reads with MIRA (see Table 1). The comparative analysis of five assemblers show the time and memory requirement of some major assemblers on a 3GHz quad core machine (Kumar and Blaxter, 2010). This limits the number of genomes we can use to perform the study as we need to run the assembly process several times with different parameters. Currently there are over 2,773 strains of bacteria alone and creating simulated read libraries with various input parameters and assembling them is the major computational challenge. Additionally, assembling read libraries with multiple input parameters varying is too time consuming and the relationship among the input parameters becomes hard to explore. Therefore, in this study we vary the input parameters of the read libraries only along one dimension at a time.

Due to the above big data challenges we perform the study on a smaller scale by selecting a representative set of bacterial genomes. To perform this study we selected 20 sample prokaryotic genomes based on the genome structure. The first set of 10 sample

organisms was selected based on the number of repeat elements. The number of repeats sequences were counted in all known bacteria genomes from NCBI database using RepeatScout (Price, *et al.*, 2005). The top 10 genomes with the greatest number of repeat sequences were selected for the study. The second set of 10 sample organisms was selected based on the number of insertion sequences in the genome. Insertion sequences are mobile genetic sequences which copy themselves at different locations on the genome. The insertion sequences belonging to the same family are very close copies of each other and are often not correctly assembled by assemblers. Therefore we selected 10 genomes with large number of insertion sequences with largest insertion sequence copies from the ISFinder database (Kichenaradja, *et al.*, 2010). The sample genomes selected are highly repetitive real genomes and a simulator is used to generate fragment libraries with different read lengths, coverage and mate pair distances. The selection of only 20 prokaryotic genomes can be seen as a very small sample size but, the long execution time of most open source assemblers is the major limiting factor in the scale of this study. We selected assemblers with parallel implementations and covered a wide range of the input parameters to study the correlations between the input and output parameters in detail.

The fragment libraries are assembled using the four assemblers Celera WGS (Miller, *et al.*, 2008), ABySS (Simpson, *et al.*, 2009), Velvet (Zerbino and Birney, 2008) and parallel version of MIRA (Biswas, *et al.*, 2013). The assembly characteristics are correlated with the fragment library and genome structure parameters to derive a polynomial relationship that can be used to estimate the expected quality of assembly. The correctness of the polynomial regression is measured by 10 fold cross validation. The set of genomes is divided in to 10 subsamples out of which 9 subsamples are used for the

polynomial regression and the remaining subsample is used for calculating the mean square error (MSE). This process is repeated 10 times so that each subsample is used as test set in one of the iterations. The average MSE provides a measure of correctness of the model. The study in itself is interesting and useful for finding parameters that make significant differences to assembler output and must be considered during selection among assembler. For example high coverage seems to deteriorate assembler quality for Celera WGS and Velvet but, the does not make a significant difference to ABySS and MIRA assemblies.

In this section we present the results of the study correlating assembler output to the input parameters. The sample genomes selected are highly repetitive real genomes and a simulator is used to generate fragment libraries with different read lengths, coverage and mate pair distances. The fragment libraries are assembled using the four assemblers Celera WGS, ABySS, Velvet and MIRA. The assembly characteristics are correlated with the fragment library and genome structure parameters to derive a polynomial relationship. The degree of the polynomial used to approximate the correlation curve is progressively increased until no major improvement in the coefficient of determination (R2) is achieved. The range of values for R2 is between 1 and 0 where 1 indicates strong directly proportional relation and 0 indicated no correlation. Therefore, a value of R2 close to +1 indicates the strong relationship between the X and Y variables. The polyfit function from MATLAB was used to fit the data and obtain coefficients of the polynomial.

The correctness of the polynomial regression is measured by 10 fold cross validation. The set of genomes is divided in to 10 subsamples out of which 9 subsamples

are used for the polynomial regression and the remaining subsample is used for calculating the mean square error (MSE). This process is repeated 10 times so that each subsample is used as test set in one of the iterations. The average MSE from the 10 iterations provides the measure of correctness expected from the estimating polynomial.

2.1. Read Length Experiment

The read length experiment varies the average fragment length of the dataset and correlates it to the number of correctly assembled contigs. A contig is considered correctly assembled if there are no incorrect joins and the whole contig can be aligned to original genome sequence using MegaBLAST (Altschul, *et al.*, 1997) with standard parameters. We first generate 50 read length values in the range of 100bp to 500bp sampled uniformly at random. The simulator simulates the fragmentation process with each read length for each of the 20 genomes. This process generates 50 datasets with mean read length in the range of 100bp to 500bp for each of the 20 genomes. The mean coverage of the datasets is constant at 40 and the datasets contain no mate pair information. The read lengths in each dataset are normally distributed with a standard deviation of 50bp.

The fraction of correctly assembled contigs, i.e. number of correct contigs divided by the total number of contigs, is obtained from assembly of the 50 datasets for each of the 20 genomes. The fraction of correctly assembled contigs is averaged over the 20 genomes for each assembler to obtain the mean fraction of correctly assembled contigs at each of the 50 data points. This curve of 50 points represents the assembler misassembly characteristic over the range of read lengths from 100bp to 500bp.

The fraction of correctly assembled contigs for each assembler is averaged

within equal intervals of 25bp from 100bp to 500bp for easier visualization (see Graph 1.).

ABySS assembler performs best with only about 5% incorrect contigs and remains consistent for the whole range of read lengths. Celera WGS and Velvet have similar misassembly characteristics and MIRA performs worst with over 10% of contigs misassembled.
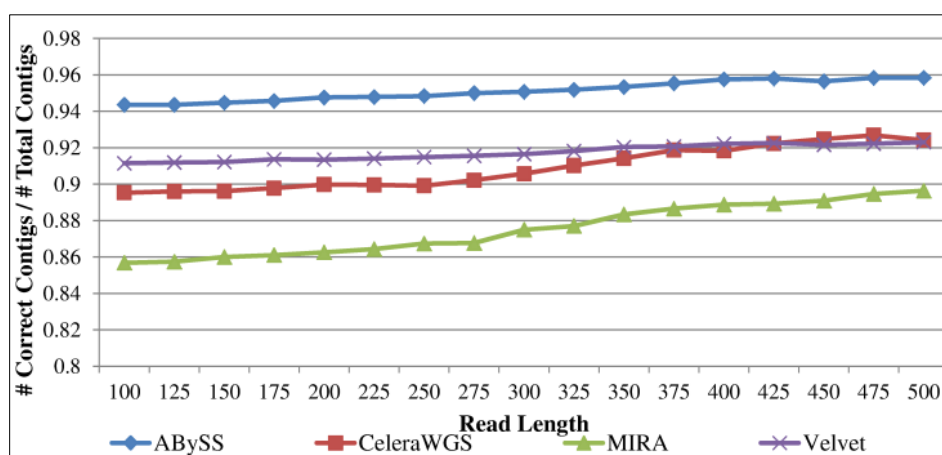
Graph 1. Mean of the fraction of correctly assembled contigs in intervals of 25 vs read length (bp)

The 50 data points generated for each assembler can be approximated by a polynomial and 10 fold cross validation is used to obtain the average error of the estimator polynomial (see Table 2). The curves can be approximated with low error using a quadratic or cubic polynomial with very strong coefficient of determination (Table 2 Column 3). This indicates that read length is a highly significant parameter for correct assembly for all the assemblers. The average mean square error is very low demonstrating that these polynomials are good predictors of misassembly (Table 2 Column 6).

Table 2. Correlating read length and correct contigs

Assembler	Degree[a]	R[2b]	D.F. [c]	P-value[d]	Avg. MSE[e]
ABySS	2	0.974	398	0.000	4.667e-4
Celera WGS	3	0.980	397	0.000	8.975e-4
MIRA	3	0.987	397	0.000	1.039e-2
Velvet	3	0.978	397	0.000	1.454e-3

[a]Degree of the polynomial fitting the assembler output characteristics.
[b]Coefficient of determination: Expresses the strength of the relationship between the X and Y variables.
[c]Degrees of freedom.
[d]Probability of getting an R^2 with a polynomial of this degree.
[e]Average of the mean square error generated by each iteration of 10-fold cross validation.

2.2. Coverage Experiment

The coverage experiment varies the coverage of the dataset keeping read length constant and correlates it to the number of correctly assembled contigs. We simulate fragmentation process for 20 different coverage values starting from 10 to 200 with equal gaps of 10 for each of the 20 genomes. The read length of the dataset is constant at 400bp and the datasets contain no mate pair information.
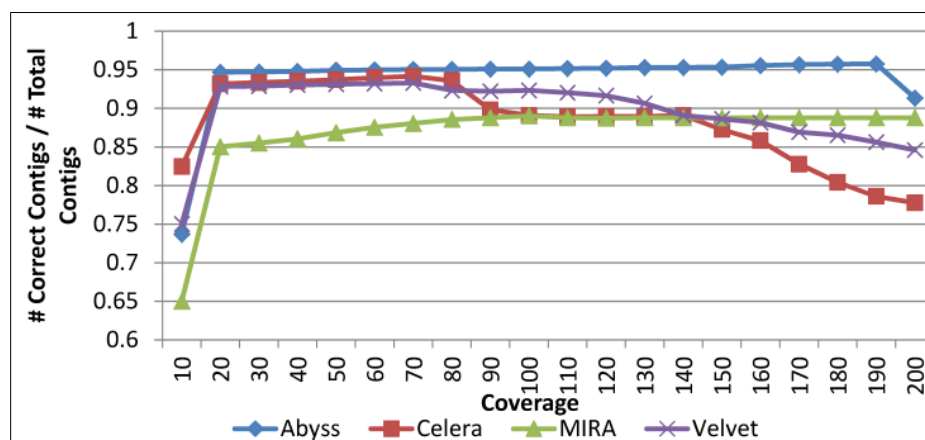
Graph 2. Correctly assembled contigs vs coverage

The fraction of correctly assembled contigs, i.e. number of correct contigs divided

by the total number of contigs, is obtained from assembly of the 20 datasets for each of the 20 genomes. The fraction of correctly assembled contigs is averaged for each assembler at each of the 20 coverage values to obtain the mean fraction of correctly assembled contigs at each of the 20 coverage points. This curve of 20 points represents the assembler misassembly characteristic over the coverage of 10 to 200 (see Graph 2.). The increase in coverage initially improves the percentage of correct contigs (see Graph 2.). However, at very large coverage the both Velvet and Celera WGS perform increasingly worse. MIRA and ABySS seem to perform consistently at higher coverage.

The 20 data points generated for each assembler can be approximated by a polynomial and 10 fold cross validation is used to obtain the average error of the estimator polynomial (see Table 3). The curves in this case are better approximated by a quartic polynomial with low coefficient of determination in case of ABySS and MIRA (Table 3 Column 3). This indicates that coverage plays a role in assembly generated by Celera and Velvet but, not quite as significant in the other two assemblers.

Table 3. Correlating coverage and correct contigs

Assembler	Degree[a]	R^{2}[b]	D.F.[c]	P-value[d]	Avg. MSE[e]
ABySS	4	0.753	15	1.83e-3	4.551e-3
Celera WGS	4	0.919	15	5.07e-8	2.750e-3
MIRA	4	0.814	15	2.3e-5	5.97e-3
Velvet	4	0.899	15	6.2e-8	1.250e-3

[a]Degree of the polynomial fitting the assembler output characteristics.
[b]Coefficient of determination: Expresses the strength of the relationship between the X and Y variables.
[c]Degrees of freedom.
[d]Probability of getting an R^2 with a polynomial of this degree.
[e]Average of the mean square error generated by each iteration of 10-fold cross validation.

Therefore, coverage seems to be a parameter that must be considered during assembly selection. The average mean square error is very low demonstrating that these

polynomials are good predictors of misassembly (Table 3 Column 6).

2.3. Mate Pair Experiment

The mate pair experiment varies the mate pair of the reads in the dataset keeping read length and coverage constant. The fragmentation simulation is done for each of the 20 genomes generating 5 datasets with mate pair distance from 2kbp to 6kbp with equal gaps of 1kbp. The mean coverage is constant at 40 and the mean read length is 400 bp. The percentage of mated reads in the dataset is also a constant at 70%.
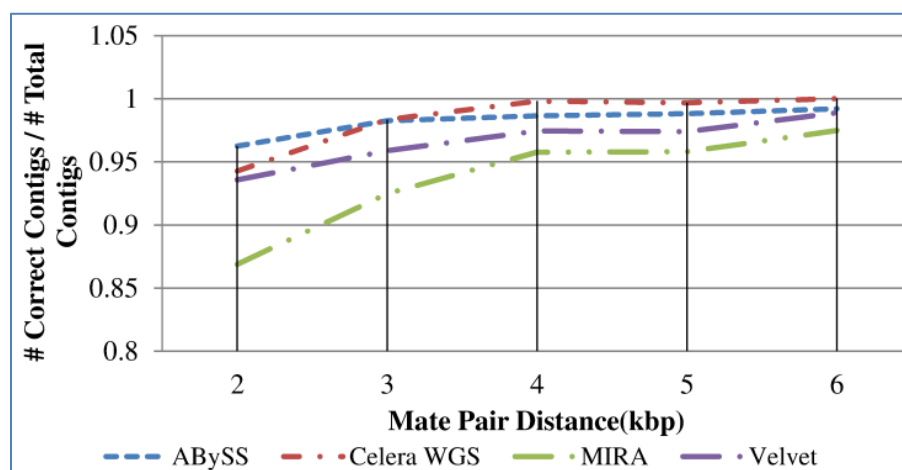
Graph 3. Correctly assembled contigs vs mate pair distance

The fraction of correctly assembled contigs, i.e. number of correct contigs divided by the total number of contigs, is obtained from assembly of the 5 datasets for each of the 20 genomes. The fraction of correctly assembled contigs is averaged for each assembler at each mate pair distance point to obtain the mean fraction of correctly assembled contigs at each of the 5 mate pair distances. This curve of 5 points represents the assembler misassembly characteristic over the mate pair distance of 2kbp to 6kbp (see Graph 3). In presence of mate-pair, data Celera assembler performs best and makes

almost no errors in presence of long distance mates. The performance of the other assemblers is comparably good with MIRA making the largest number of incorrect joins.

The 5 data points generated for each assembler can be approximated by a polynomial and 10 fold cross validation is used to obtain the average error of the estimator polynomial (see Table 4). Quadratic polynomials give good approximations of these curves showing high coefficient of determination (Table 4 Column 3). This indicates that mate pair distance is a highly significant parameter for correct assembly. The average mean square error is very low for all the polynomials (Table 4 Column 6).

Table 4. Correlating mate pair distance and correct contigs

Assembler	Degree[a]	R^{2}[b]	D.F. [c]	P-value[d]	Avg. MSE[e]
ABySS	2	0.974	2	0.127	2.47e-5
Celera WGS	2	0.982	2	0.087	5.31e-4
MIRA	3	0.993	1	0.106	1.83e-4
Velvet	2	0.927	2	0.288	2.76e-5

[a]Degree of the polynomial fitting the assembler output characteristics.
[b]Coefficient of determination: Expresses the strength of the relationship between the X and Y variables.
[c]Degrees of freedom.
[d]Probability of getting an R^2 with a polynomial of this degree.
[e]Average of the mean square error generated by each iteration of 10-fold cross validation.

2.4. Repeat Experiments

The repeat experiments study the assembly of repeat structure of the genomes. We perform four repeat experiments by measuring the fraction of repeats assembled and the longest repeat assembled by an assembler for given data of certain read length and coverage.
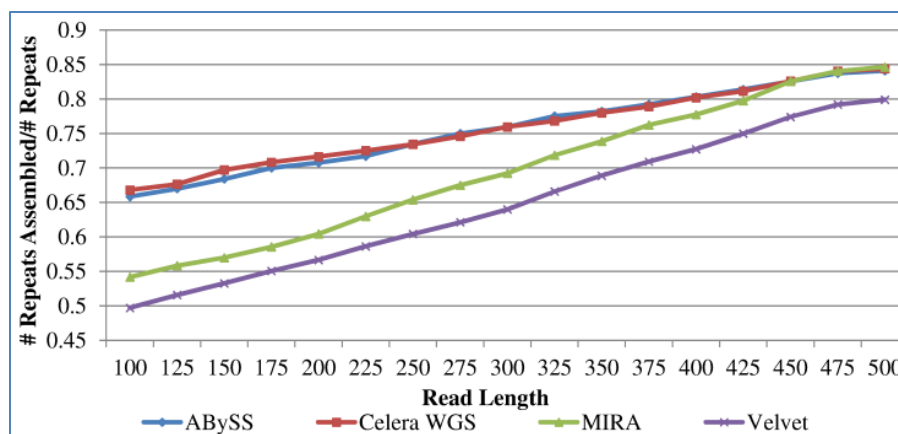
Graph 4. Mean of the fraction of repeats correctly assembled in intervals of 25 vs read length

2.4.1. Repeats Assembled for given Read Length

This experiment studies the correlation between assemblies of repeat structure over a certain read length for an assembler (see Graph 4). All of the assemblers are capable of assembling at least 50% of the repeat sequences. However, none of the assemblers can assembly more that 85% of the repeats and all the assemblers perform well for longer reads. However, ABySS and Celera WGS perform much better for shorter read lengths.

2.4.2. Repeats Assembled for given Coverage

This experiment studies the correlation between assemblies of repeat structure over a certain range of coverage (see Graph 5). All of the assemblers show moderate improvement in the number of repeat assembled as coverage increases. However, at very high coverage ABySS, Celera and Velvet show some deterioration in the percentage of repeats assembled.
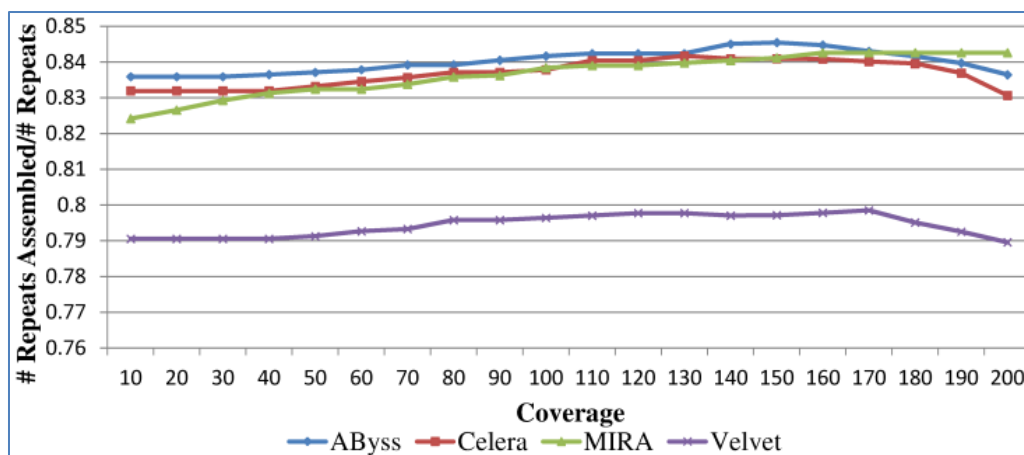
Graph 5. Fraction of repeats correctly assembled vs coverage

2.4.3. Longest Repeats Length Assembled for given Reads Length

This experiment studies the correlation between the longest repeat correctly assembled over a certain read length for an assembler (see Graph 6). This is interesting as we can estimate the longest repeat family that will be assembled by an assembler for input dataset.
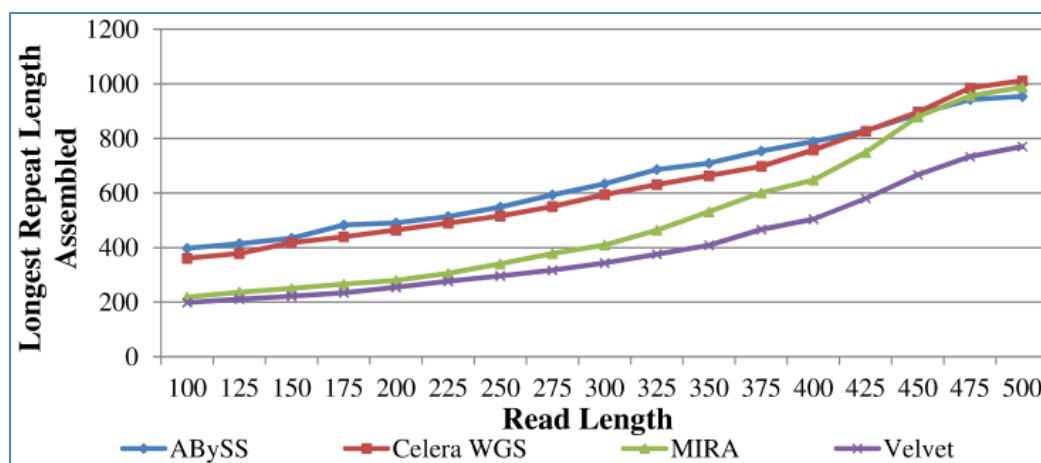
Graph 6. Mean of the length of the longest repeat sequence correctly assembled intervals of 25 vs read length

The longest repeat assembled seems to be close to twice to the read length and all the

assemblers seems to reach that limit for longer read length. However, at smaller read lengths ABySS and Celera WGS seem to perform best.

2.4.4. Longest Repeat Sequence Assembled given Coverage

This experiment studies the correlation between the longest repeat correctly assembled over a certain range of coverage for an assembler (see Graph 7). This is interesting as we can estimate the longest repeat family that will be assembled by an assembler for input dataset. The longest repeat assembled does not seem to be significantly correlated to the coverage. However, at higher coverage the some of the repeats are disassembled due to threshold miscalculations.
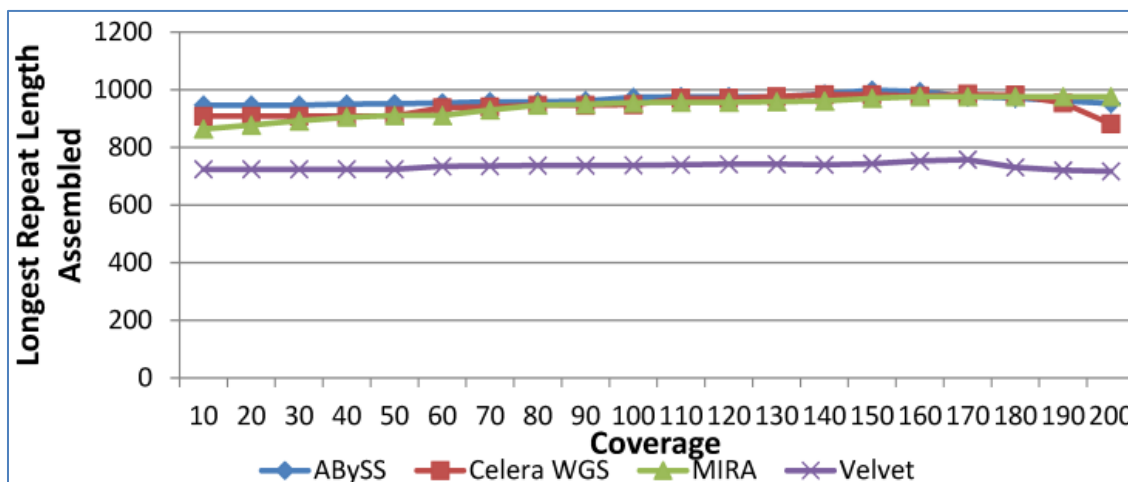
Graph 7. Length of the longest repeat sequence correctly assembled vs coverage

3. Parallel Genome Assembly

The strategy used for assembling a genome should be guided by a priori knowledge and the data available. As discussed in the earlier sections, nature of the genome, sequencing technology, read lengths, coverage etc. affect the choice of assembly technique. The choice of assemblers for a given set of input parameters is increasing and

requires intelligent selection. A detailed description of the assembly techniques and the history of their various implementations can be found in (Imelfort and Edwards, 2009; Miller, *et al.*, 2010).

The assemblers are the most computationally intensive processes and efficient execution of the assembly process is essential in scalability of this tool. The general process of genome assembly using graph algorithms is the most successful and has three basic stages. The first stage is the graph construction by overlap calculation with candidate selection or k-mer extension. The second phase is graph reduction for simplifying computation and error correction. Finally, the contig generation phase is implemented, where the graph is traversed to find long paths. Assemblers may take anywhere from several hours to few days to complete an assembly e.g. MIRA 3.2.0 (Chevreux, *et al.*, 1999) takes 18.3 hours to complete an assembly for 1 million 454 reads. A comparative study of assembly execution times and memory requirements in covered in (Kumar and Blaxter, 2010).

The first phase, i.e. overlap computation, is the most computationally expensive and memory- intensive phase and can account for 30-50% of the total assembly time. This phase can be easily parallelized to significantly reduce the assembly time (Miller, *et al.*, 2008). Our effort in parallel refactoring of this phase using OpenMP in MIRA 3.2.0 has significantly improved assembly time (Biswas, *et al.*, 2011). Specialized hardware (Sarje and Aluru, 2008) for this phase has also been proposed, however this is very expensive and not acceptable for small genomic labs. A distributed campus grid based approach to parallelize this phase has been proposed (Moretti, *et al.*, 2009), but this requires managing movement of sequence data across the network to

worker processes' local address space. Hadoop map-reduce algorithms for short read mapping (Schatz, 2009) have been proposed and we propose to extend this approach to improve the performance of this phase. A fast alignment toolbox will be developed leveraging Hadoop's map-reduce framework and will be used instead of the assemblers' native overlapper module.

The next two phases of assembly vary significantly from assembler to assembler. Each assembler implements different schemes for error correction and reductions for repeat handling. Efforts to parallelize these phases have not been very successful due the dependencies and inter-computation communication requirements. A Hadoop based assembler for de-novo assembly of genomes using de-Bruijn graph model is proposed in (Michael Schatz and 2010). Our implementation of a parallel framework for contig construction for OLC assemblers in MIRA 3.2.0 has improved performance without sacrificing assembly quality (Biswas, *et al.*, 2012). We here propose to develop a middleware for provisioning assemblers with required resources. The cloud application service will profile each assembler and provide required resources for execution. Scalability issues and implementation challenges must be overcome to deploy the tool as a cloud application service that will initiate multiple assemblers.

Most assemblers generate a set of contiguous non-overlapping sequences covering some part of the genome. These contigs are ordered and oriented through the process of scaffolding to generate a gapped representation of the genome. Scaffolding algorithms can use mate pair information of the reads at the ends of a contig to join it to other contigs. Joining can also be done by mapping the contigs to a reference genome or by inspecting other assemblies and checking for possible joins missed by the assembler.

Some of the assemblers like Celera WGS (Miller, *et al.*, 2008) are capable of utilizing mate pair data for scaffolding. Other tools for scaffolding include Bambus (Pop, *et al.*, 2004), SUPERCONTIGS (Puiu, 2004) and Autofinish (Gordon, *et al.*, 2001).

Current DNA sequencing methodologies (with the exception of emerging experimental technologies) cannot sequence DNA fragments of greater than ~1 kilobase (kB) in length. We rely on computational methods to assemble a complete DNA sequence from a large number of DNA fragments of smaller size. One popular and cost effective method of generating these short fragments of a genome is based on shotgun sequencing such as 454 pyrosequencing. Shotgun sequencing generates DNA fragments by breaking up multiple copies of the original sequence at random points. Next a software program is used to construct the original DNA from a large set of DNA fragments generated by shogun sequencing. The problem of combining DNA fragments (reads) to reconstruct the source DNA is known as sequence (or genome) assembly problem. The assembly problem is usually modeled as computing the shortest common superstring (SCS), which is a reasonable approximation of the original sequence. These assembled sequences are pieces of the original sequence and are called contagious sequences (contigs). The SCS problem can be modeled as a graph problem and is shown to be NP hard (Kececioglu and Myers, 1995; Wang and Jiang, 1994). Additional complexity arises when there are repeats in the original sequence. Repeats are multiple identical or nearly identical stretches of DNA which the SCS solution represents only once in the assembled genome. This problem is known as repeat collapse and can lead to serious assembly errors.

MIRA (Chevreux, *et al.*, 1999) is an open source assembler, which is widely used

by biologist and works effectively in presence of repeats. However, it is computation intensive, for example an assembly of one million reads requires about 18.3 hours. There is a need to parallelize the assembly process for speeding up this computation so as to take advantage of cheap parallel computing power available in multicore systems. This is a challenging task because (a) MIRA is complex software consisting of 90447 lines and uses a number of heuristics to generate a good quality assembly; and (b) the critical computation phase is inherently sequential. The MIRA assembler consists of four phases: (i) edge detection (ii) graph construction, (iii) contigs building, (iv) consensus computing and error correction. The contigs building phase of building non-overlapping paths in the underlying graph is inherently sequential. We propose a modification to this phase that enables building of non-overlapping paths concurrently while preserving the quality of assembly.

We implemented the modified MIRA assembler to speedup of contigs building phase. In addition we parallelize the other three phases which are straightforward. We implemented the modified MIRA assembler on a 64-core system with eight Intel(R) Xeon(R) X7560 processors. We were able to speedup the building contigs phase by a factor of 55 on the 64-core system. Additionally, we parallelized the other phases of the MIRA assembler. The speedup achieved for graph construction phase was 55.32 and the consensus computing with error correction was improved by a factor of 58.73. Finally, we were able to reduce the total sequential execution time of assembly from 18.3 hours to 3.4 hours (speedup of 5.57) without sacrificing assembly quality. It is worth noting that the overall speedup is limited by Amdahl's Law as parts of original MIRA assembler are inherently sequential. For example for one million reads the sequential portion of the

MIRA assembler takes about 2.78 hours doing I/O or other operations which limits the overall speedup to 6.58. Therefore, the overall speedup achieved was close to the limit with a parallel efficiency of 84.65%.

The sub-sections focus on the core assembly pipeline of MIRA assembler and describes the parallel algorithms for the assembly phases and parallel implementation details and the APIs used to implement the parallel algorithms. The experimentation results, experiment environment and resulting assembly quality are presented.

3.1. MIRA Assembler Overview

MIRA is an open source assembler based on the OLC graph model that addresses the assembly problem and is widely used by the life sciences community. MIRA is capable of handling next generation shotgun reads from 454, Ion Torrent, Solexa and PacBio machines along with Sanger sequences. MIRA has been used at IMB Jena Genome Sequencing Centre and has been shown to be capable of assembling cosmid sequences in Human genome (Chevreux, 2005). MIRA has also been used for de novo assembly of 454 pyrosequencing transcriptome projects (Barker, *et al.*, 2009; Papanicolaou, *et al.*, 2009; Pauchet, *et al.*, 2009; Pauchet, *et al.*, 2010; Roeding, *et al.*, 2009; Zagrobelny, *et al.*, 2009). The MIRA assembler is designed to work with a small memory footprint so that it can be executed on regular desktop computers and is generally used for small to medium scale assembly projects.

MIRA provides specific routines for handling various read types, for example, mate pair information can be leveraged to improve assembly. The assembly process is based on the Overlap Layout Consensus (OLC) graph model (Kececioglu and Myers, 1995) with critical code for handling repeats of various lengths. MIRA has four major

stages as shown by the assembly pipeline diagram (see Figure 4). The input reads are preprocessed based on quality values and ancillary data if provided and presented to the iterative portion of the assembly process.

Table 5. MIRA assembly time breakup (2)

Organism	Reads[a]	Total Time	Graph Const.[b]	Contigs Building[c]	Consensus[d] & Error Correction[e]
M. Marinum	500,000	624	413	103	108
	1,000,000	1,327	867	261	199
E. Coli	500,000	589	342	132	115
	1,000,000	959	612	192	155
M. Tuberculosis	500,000	581	374	96	111
	1,000,000	1,123	712	219	192
Average %			63.51	19.05	17.33

[a]The number of simulated reads with mean length of 600bp and standard deviation of 100bp.
[b]The time (in minutes) to detect potential edges and construct the assembly graph using smith-waterman overlap.
[c]The time (in minutes) to find all the paths in the graph and assemble the contigs.
[d]The time (in minutes) to construct the consensus sequence of the contigs.
[e]The time (in minutes) to error correct the contigs in the assembly.

3.1.1. Edge Detection Phase

The assembly process proceeds with each read as a vertex in a graph and the first phase determines the high confidence region (HCR) of each read and scans all the n2 edge possibilities using heuristic match algorithms (Grillo, *et al.*, 1996; Wu and Manber, 1992). The match determines if a sequence of length k is present in the matching read with at most l errors. For each sequence the complement is also matched to find all potential edges from it. This SKIM algorithm creates two potential edge files named, post-match files, for the forward and complement matches. Each record in these files corresponds to a potential edge, containing the identifiers of the two reads and some offset information for matching. This phase is implemented in parallel in the standard implementation of the software and uses the boost threading library.
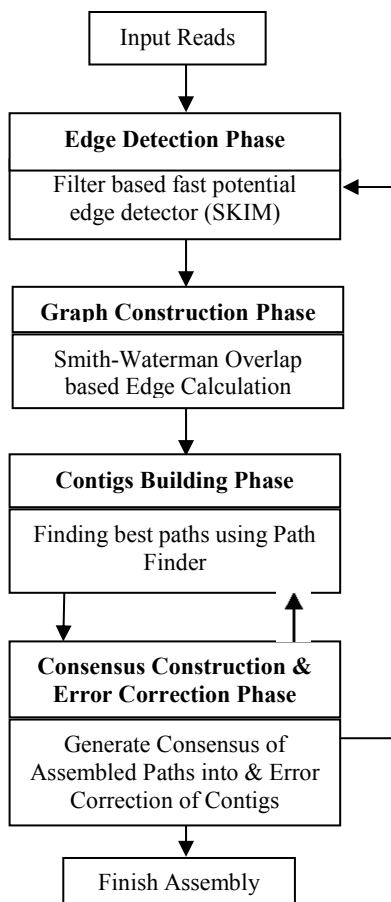
Fig. 4. MIRA assembly pipeline

3.1.2. Graph Construction Phase

The second stage of graph construction is the most time consuming phase of the assembler accounting for over 60% of the assembly time (Table 5, row 4, column 4). This phase processes the reads, finds the edges in the graph, and computes the edge weights of the graph. The edge weights are computed by banded Smith Waterman overlap calculation (Chevreux, 2005; Smith and Waterman, 1981) for each of the pair of reads generated by the SKIM algorithm. Some edges are rejected based on various conditions and overlap computation is avoided if the overlap length satisfies certain conditions.

The third phase of the assembly is the central path finding algorithm to determine the paths in the graph and consumed over 19% of the assembly time (Table 5, row 4, column 5). This phase is a greedy heuristic to find the partial paths in the graph and build the best contiguous sequences (contigs). In this phase the path finder algorithm identifies vertices with high degree and low error and begins the assembly process by adding neighboring reads to it and forming contigs. A contig can grow in length, depth or both when a read is added to it and every addition increases the expected error based on the edge weight.

The length of a contig refers to the number of base pairs the overlapping reads cover. The depth of a contig at each base pair position is the number of reads that overlap at that position. Each contig is a consensus sequence of all the overlapping reads that capture a certain region of the genome. Ideally, a contig should have a depth close to the coverage of the input data as each base in the contig should correspond to a base in reads stacked up correctly. Also, the length of the contig must be close to the length of the genome. However, due to presence of repeats and errors the contigs cannot be extended beyond a certain length as the total acceptable mismatch error crosses the allowed threshold. The backbone build strategy increases the length of the contig and the in-depth strategy adds reads to increase the coverage. Each has advantages and disadvantages, but, both must be used to successfully build non redundant and correct contigs (Chevreux, 2005).

A (n, m) look-ahead version of a simple greedy strategy is applied to select the most probable overlap candidate for a given contig. The algorithm extends n paths from

the last n vertices of a contig upto m levels and the vertex generating the best path upto m

levels is selected for assimilation into the contig. The new read selected is checked

against the existing contig consensus for errors and if the mismatches are within a certain

threshold, the read is accepted into the contig consensus. This read is then not used by the

other contigs in the same pass of the assembly. Therefore, a contig building iteration is

dependent on all the previous contigs making the process intuitively serial in nature.

3.1.3. Consensus Construction and Error Correction Phase

The next two steps are consensus construction and error correction. The error

correction routines apply thresholds based on sequencing technology and quality values

to detect misassemblies and chimeric reads. The final error correction phase detects and

corrects misassembles by computing the overall error of a read in a consensus and error

at a contig position. The error in assembly of a read is computed based on the difference

between the nucleotides in the read and the nucleotides in the contig consensus. The

number of differences between the read and the contig consensus sequence should not

exceed the expected overall sequencing error of the dataset. If the difference is beyond a

certain threshold the read must be removed from the contig assembly and marked as

misassembled. This phase is also responsible for generating the final sequence of each

contiguous path found in the graph. All the overlapping reads at a particular position in

the path contribute a single nucleotide weighted by the quality value if available. The

sequence is generated by taking consensus among the nucleotides. These last two steps

take up about 17% of the assembly time (Table 5, row 4, column 6, 7). MIRA routines

encode domain knowledge specific rules which are vital for correct assembly and must be

preserved in the parallel implementation.

3.1.4. Parallel MIRA Assembly Process

In this section we discuss the MIRA assembly pipeline and describe the parallelization strategy applied to different phases of the assembly process. The basic pipeline of the MIRA assembly process is shown in Figure 4. In this paper we propose parallel algorithms for graph construction, finding non-overlapping paths in the assembly graph, consensus construction and error correction.

3.1.4.1. Parallel Edge Detection Phase

This phase is implemented in parallel in the standard implementation of the software and uses the boost threading library.

3.1.4.2. Parallel Graph Construction Phase

This process can be implemented in parallel by matrix partitioning. However, most assembly graphs are sparse in nature as each vertex has a degree of 10-30. Therefore, the fast edge detection algorithms generates a list of potential overlaps reducing the number of edges requiring overlap computation which is $\theta(n^2)$. Therefore, we compute the edge weights of these potential overlaps iteratively in parallel.

The edges appear in random order in the potential edge files generated by the edge detection phase and weight calculation of one edge is not dependent on the others. We use OpenMP parallel pragmas and TBB containers to refactor this phase and execute it in parallel. We implement a single producer generating multiple tasks, each task computing a certain number of edges. This phase can account for over 30-50% in the serial pipeline and parallelization shows significant improvement in overall time. Complete details of the implementation and results showing linear speedup of the kernel can be found in our previous work (Biswas, *et al.*, 2011).

3.1.4.3.   Parallel Contigs Building Phase

The parallel contig building algorithm must independently construct contigs using a set of start vertices. The selected vertices must minimize sharing of vertices between two contigs so that the paths corresponding to any two contigs generated in parallel are non-overlapping.

The selection of the starting points of the parallel path construction threads has a significant effect on the contigs generated and the overlap among contigs. Contigs generated in serial execution of contig construction process are non-overlapping as vertices are removed from the graph after they have been included in the assembly of a contig. However, parallel threads are not constrained and are assembled independently. Therefore, selection of the start vertex is important to reduce the number of overlapping vertices among the parallel contigs. The following strategies have been explored for selection of the parallel start vertex:

- Random selection: The start vertices are selected at random by each parallel thread and contigs are built.

- Dense Vertices First: The start vertices are selected in order of their degree by each parallel thread and contigs are built.

- BLAST separated: The start vertices are selected in order of their degree and ensuring that the vertex sequences are divergent using BLAST search algorithm (Altschul, *et al.*, 1990). The selection process progresses in order of the degree of the vertices. The first thread is spawned with the highest degree vertex and the sequence of the start vertex is added to a BLAST database. The next highest degree vertex is selected and BLAST searched against the database. If a hit is returned, the vertex is skipped and

the next vertex is searched. Otherwise, a new thread is spawned with the start vertex. This process is repeated to start all parallel contig construction threads.

- N-Path separated: The start vertices are selected such that the selected vertices are N edges apart in the assembly graph. The selection process progresses in order of the degree of the vertices. The first thread is spawned with the highest degree vertex and the next highest degree vertex is selected and a BFS search checks if it is within N edges from the previous vertex. If it is found within the N edges, then it is skipped and the next vertex is checked. Otherwise, a new thread is spawned with the start vertex. This process is repeated to start all parallel contig construction threads.

- BLAST & N-Path separated: The start vertices are selected such that the both the BLAST search and N-Path restrictions are enforced.

Among the five options selecting the BLAST separated start vertices significantly reduces overlapping of contigs. The other option of selecting the n-path separated start vertex is time consuming due to need to perform n-level breath first search for each of the previously selected start vertices. However, in some cases it can be shown to generate contigs with fewer overlapping vertices.

The independent threads generate a contig with the best possible depth and length. The start vertex selection process reduces the probability of overlap among parallel contigs. However, none of the selection processes can ensure that all contigs are non-overlapping. So, the resulting contigs are analyzed to check for common vertices. In case, contigs contain common vertices, the longer contig is allowed to keep the vertex and the read is removed from the contig consensus of the other contigs. Therefore, a contig reduction phase is added to account for contigs using common vertices.

The process of spawning parallel threads for contig building is done in phases with k parallel contigs built in one phase and reduced. The vertices assembled in the first phase are removed from the graph and the parallel contig building phase is repeated until connected vertices in the graph are assembled or further assembly is not possible due to absence of acceptable edges.

3.1.4.4.   Parallel Consensus Construction and Error Correction Phase

The parallel algorithm for this phase divides the contig length into equal size partitions and each parallel thread performs the consensus computation for a given range of positions in the alignment. The consensus construction process calculates a probability value for each base at a given position. The base with the largest probability is taken as the consensus.

The error correction routing detects misassemblies and chimeric reads in parallel by dividing the reads aligned to a contig into groups and parallel threads are spawned for processing the reads in a group. The error in the assembly of a read is computed based on the difference between the nucleotides in the read and the nucleotides in the consensus. Reads with error beyond a certain threshold are removed from the consensus, as they have been misassembled. The error at each contig position is also checked and reads with strong variations at certain positions are misassembled due to similar surrounding sequence.

3.2. Implementation

The parallel implementation of MIRA is done through a refactoring process ensuring thread safety of existing routines. It is essential that in the parallel version of MIRA the basic assembly pipeline is not significantly changed and the assembly output is

similar to that of the sequential implementation. Therefore, we are interested in identifying parallelization opportunities in MIRA and evolve the sequential code to exploit parallelism.

We use a multicore environment for parallelization of MIRA to maintain the design philosophy of a low memory requirement desktop assembler. The parallel version of MIRA is capable of utilizing the increasing number of cores in modern processors found in most desktop and laptop computers. We found OpenMP (OpenMP, 2008) to be the best choice to refactor the MIRA C++ code as it provides a host of synchronization pragmas for parallel flow control. However, the extensive use of STL containers in the standard implementation causes performance bottlenecks in many cases. Therefore, we used concurrent collections provided by Intel's Thread Building Blocks (TBB) library (Blocks, 2011) interoperating with OpenMP to replace the STL containers as needed.

The parallel strategy for each phase of assembly was implemented by parallel refactoring of the MIRA 3.2.1 assembler. The three main challenges faced in refactoring the source code were the following. Firstly, MIRA is implemented using C++ and is optimized to reduce the memory utilization. So, many of the results at end of each stage are written onto the disk and a large number of disk writes are performed. This model would seriously impede parallelism as threads would compete for access to the disk. Secondly, MIRA uses Standard Template Library (STL) collections to implement data structures such as the adjacency list, repeat markers and the sequence read pool. Parallel updates on these data structures would have to be synchronized to maintain correctness. Synchronization of the threads by some locking mechanism will also affect parallelism. Finally, the source code also has lot of rule checking and conditional execution of error

flagging routines which are often sequential in nature and perform updates on global data structures or write to files on the disk. To refactor such a sequential code OpenMP was found to be the best choice as it provides a host of synchronization pragmas for parallel flow control. However, the use of STL objects would cause performance bottlenecks in many cases. Therefore, we used concurrent collections provided by Intel's Thread Building Blocks (TBB) library.

3.3. Experiment Results

In this section we describe the test data sets, discuss the assembly quality after parallel refactoring and present the results showing significant improvements in assembly time. Input data with required characteristics for experimentation is rarely available as the genome sequences are published in final form and the raw data underlying these genomes is not publicly released. The NCBI trace archive and CBCB published data are not sufficient for extensive systematic assembler testing. Therefore, for experimentation we developed a simulator for 454 pyrosequencing. Earlier sequencing simulation techniques, such as Genfrag by (Engle and Burks, 1994) and CelSim (Myers, 1999) concentrated on shotgun data, and only MetaSim (Richter, *et al.*, 2008) and Flowsim (Balzer, *et al.*, 2010) simulated data from 454 pyrosequencing process. Generating a simulator based on an empirical distribution is a better fit, we, for purpose of simplicity and lack of 454 pyrosequencing data sets, apply a parametric log normal distribution to simulate the shotgun process based on user specified read length and standard deviation. Quality values however are estimated from a position specific error function based on the read length and base type similar to (Balzer, *et al.*, 2010).

Table 6. Graph sizes

| Vertex Degree | Number of Vertices in the Graph | | | | | |
| | 100,000 | | 500,000 | | 1,000,000 | |
	Real	Simulated	Real	Simulated	Real	Simulated
0 – 10	80,977	94,201	310,372	137,317	296,096	62,215
11 – 20	5,640	4,338	153,604	313,465	402,450	632,455
21 – 30	779	602	4,636	16,417	158,698	224,215
31 – 40	287	293	2,033	1,312	67,367	12,366
41 – 50	145	194	1,399	964	37,012	4,909
51 – 60	63	167	1,110	1,055	21,326	3,481
61 – 70	32	85	904	1,206	9,505	2,829
71 – 80	33	51	630	1,141	4,887	2,156
81 – 90	26	29	433	974	3,003	1,958
91 – 100	12	12	280	641	2,066	1,507
100 –	31	6	547	1,358	35,093	3,852

An experiment to verify the similarity between the graphs generated by simulated and real reads was performed. Three data sets with 100K, 500K and 1 million reads were created from a large set of Roche 454 pyrosequencing real reads of *Mycobacterium pseudoshottsii*. The simulator was used to generate input read data sets with same number of reads and mean read length and standard deviation. The experiment was also conducted to explore the degree distribution of the vertices in the graph generated by real and simulated reads (Table 6). The vertex degree distribution of the graphs is similar e.g. for both the real and simulated graphs of 1 million reads most of the vertices have a degree between 0-20 and decline steadily thereafter (Table 6, column 6, 7). So, the assembly graph generated by the real sequencing machines and simulator are similar in terms of degree distribution and sparse in nature as $|E| = O(V)$.

The parallel framework proposed in the paper is expected to generate an assembly similar to the assembly generated by MIRA 3.2.1 (Chevreux, *et al.*, 1999). The experiment performed to compare the assembly of MIRA 3.2.1 with the assembly generated by the implementation of the parallel framework in MIRA 3.2.1 uses standard

assembly quality metrics such as N50 score, longest contig length, number of total contigs, coverage of the original sequence, base calling errors in the contigs and the number of reads assembled. The comparison was done for different input read numbers and parallel threads (Appendix A, Table 19). The experiment was performed using simulated reads of the genomes of *Escherichia coli HS, Mycobacterium vanbaalenii PYR-1* and *Mycobacterium marinum M* with mean length of 500bp and standard deviation of 100bp. The parallel assembly quality is comparable to MIRA assembly quality in most cases e.g. the assembly of *Escherichia coli HS* with 1 million reads generates the same number of large contigs (>100Kbp), same overall coverage of the genome, very close longest contig length and N50 scores (Appendix A, Table 19, major row 3). Also, the quality of assembly is not significantly affected by the number of threads used in the parallel process.

Table 7. Graph construction phase execution time (1st pass)

Smith Waterman Comp.	Execution Time on Threads (sec)						
	MIRA	2	4	8	16	32	64
100,000	8.48	4.65	2.34	1.53	0.96	0.96	0.96
500,000	38.84	20.92	10.42	5.59	3.36	1.74	1.01
1,000,000	80	43	21	10.6	5.24	3.42	1.69
5,000,000	470	236	120	62	30.86	15.58	8.46
10,000,000	956	468	249	132	64.49	32.87	17.28

The parallel implementation of MIRA targets the three major phases of graph construction, contig building and contig consensus construction. We present the improvement in execution time of the three parallel phases and study the effect on the overall assembly time. The read data sets for all the experiments are generated by the simulator with mean read length of 500bp and standard deviation of 100bp from the

original sequence of *Mycobacterium vanbaalenii*.

The first experiment shows the speedup of the graph construction algorithm on incrementally larger graphs. The time to calculate the Smith Waterman edge weights for the first pass is shown in Table 7. In this experiment the condition checking modules to bypass Smith Waterman overlap computation are disabled and the overlap is computed for all the edges in the graph. The primary producer thread spawns a task after reading 10,000 potential edge records from the post-match files. Therefore, the granularity of each task is 10,000 Smith Waterman calculations with average overlap length of 237bp. The experiments with various overlap lengths and granularity can be found in (Biswas, *et al.*, 2011). The computation of the graph construction shows close to linear speedup (Table 8, row 5). The average speedup achieved in the phase for various data sizes is 42.10 on 64 threads (Table 8, column 7). The average speedup is significant parameter to consider as subsequent iterations of the assembler often execute on a much smaller subset of the initial reads.

Table 8. Graph construction phase speedup (1st pass)

Smith Waterman Computations	Speedup on Threads (sec)					
	2	4	8	16	32	64
100,000	1.82	3.62	5.54	8.83	8.83	8.83
500,000	1.86	3.73	6.95	11.56	22.32	38.46
1,000,000	1.86	3.81	7.55	15.27	23.39	47.34
5,000,000	1.99	3.92	7.58	15.23	30.17	55.55
10,000,000	2.04	3.84	7.24	14.82	29.08	55.32

The contig building phase consumes significant amount of time and dominates the execution time after the linear speedup of the graph construction phase. The performance of the path finding module of MIRA 3.2.1 is compared with the parallel path finding

algorithm for various thread sizes (Table 9).

Table 9. Contig building phase execution time

Reads	Execution Time on Threads (minutes)				
	MIRA	8	16	32	64
100,000	38	6.56	3.02	1.58	1.06
500,000	110	18.30	10.01	4.16	2.34
750,000	151	22.65	12.21	5.28	2.82
1,000,000	263	41.86	21.63	9.38	4.78

The maximum speedup of 55.02 is achieved on 64 cores for a data set of 1 million reads (Table 10, row 4). The parallel module shows sub-linear speedup due to the reduction phase after the parallel contig construction. The serial reduction phase checks for paths with common vertices and reduces each contig to contain only unique reads. This phase also preforms repeat read tagging to find very high coverage regions that are most likely part of repeat sequences.

Table 10. Contig building phase speedup

Reads	Speedup on Threads (minutes)			
	8	16	32	64
100,000	5.79	12.58	24.05	35.85
500,000	6.01	10.99	26.44	47.01
750,000	6.67	12.37	28.60	53.55
1,000,000	6.28	12.16	28.04	55.02

The fourth phase parallel refactors the contig consensus construction and error correction phase of the assembler. In this experiment the consensus sequence and error rate of the assembled contigs is computed for various input reads sizes (Table 11).

Table 11. Consensus construction & error correction phase execution time

Reads	Execution Time on Threads (minutes)				
	MIRA	*8*	*16*	*32*	*64*
100,000	24.3	3.15	1.64	0.90	0.58
500,000	86.7	11.37	5.96	2.93	1.48
750,000	133.6	16.95	8.76	4.45	2.26
1,000,000	193.8	24.41	12.96	6.41	3.30

The speedup achieved in this phase is close to linear, achieving a maximum of 58.73 on 64 processors for one million reads (Table 12, row 4, column 5).

Table 12. Consensus construction & error correction speedup

Reads	Speedup on Threads (minutes)			
	8	*16*	*32*	*64*
100,000	7.71	14.82	27.00	41.90
500,000	7.63	14.55	29.59	58.58
750,000	7.88	15.25	30.02	59.12
1,000,000	7.94	15.02	30.23	58.73

The performance speedup of the parallel implementation is limited the serial components of the assembler (Table 13). The serial components include reading large input files (~1-2GB), writing output files in various formats, writing log files, tagging reads as repeats based on coverage, finding repeat regions, filter operations and reducing overlapping contigs. The serial components of the assembler account about 15% of the assembly time (Table 13, column 6). Therefore, the theoretical cap on the overall speedup achievable is approximately 6.67.

Table 13. Serial components table

Reads	MIRA Execution Time(min)	Serial Components Time (min)			Percentage
		I/O Processes	*Sorting & Filtering*	*Overlap Reduction*	
100,000	142	4.2	8.25	9.3	15.31
500,000	514	18.5	43.1	16.8	15.25
1,000,000	1,102	25.0	105.8	36.9	15.22

The overall performance speedup of the parallel implementation is compared to the total assembly time of MIRA 3.2.1 (Table 14). The average improvement in total execution time over various input sizes is about 5.57 times on 64 threads (Table 14, row 6 and column 8). This speedup is close to the theoretical limit as 15% of the assembler remains serial and the maximum possible speedup is 6.67.

Table 14. Overall speedup experiment table

Reads	Execution Time on Threads (minutes)						
	MIRA	*2*	*4*	*8*	*16*	*32*	*64*
50,000	71	34	24	17	14	12	11
100,000	142	69	42	32	25	23	21
500,000	514	334	211	154	133	113	106
750,000	748	492	321	234	190	171	158
1,000,000	1,102	657	429	316	253	213	208
Avg. Speedup	1	1.77	2.72	3.72	4.58	5.26	5.57

For completeness, we also compared our parallel implementation running on a single thread to the running time of MIRA 3.2.1 for 1 million reads (Table 14, row 6). The running time for our parallel implementation was 1,105 minutes which is slightly more than 1,102 minutes required by MIRA. The breakup of the total assembly time was as follows: 631 minutes for graph construction, 285 minutes for contig building, 189 minutes for consensus construction with error correction. Note that the total assembly time on 64 threads for 1 million reads is 208 minutes (Table 14, row 6 and column 8)

with serial a component of 167.7 minutes (Table 12, row 3). Consequently, we can now observe that with 64 threads, the time spent on graph construction, contig building and consensus construction is only 40.3 minutes (208-167.7). Hence, using 64 processors, we have been able to reduce the time required for these three parallel phases from 937.3 minutes (1105 – 167.7) to 40.3 minutes yielding a speed-up factor of 23.25.

Table 15. Real graph speedup experiment table

Reads	Execution Time on Threads (minutes)				
	MIRA	*8*	*16*	*32*	*64*
0.09×10^6	192	56.38	40.26	34.41	31.88
0.23×10^6	415	129.58	106.50	92.76	87.32
0.36×10^6	532	139.90	107.05	90.23	80.25
0.5×10^6	708	216.15	175.38	155.58	142.05
1.4×10^6	1,472	445.87	354.53	307.14	274.80
Average Speedup		3.39	4.36	5.06	5.55

The final performance experiment was performed on graphs generated by real sequencing data of three bacteria (Table 15). The rows in Table 7 correspond to graphs built for assembly of the following bacteria in top down order: *Mycobacterium vanbaalenii PYR-1, Mycobacterium marinum M, Mycobacterium shottsii* and two data sets of *Mycobacterium pseudoshottsii*. The total assembly time of the parallel implementation on different number of threads is compared to MIRA 3.2.1. An average speedup 5.55 was observed on 64 threads (Table 15, row 6 and column 6).

The experiments in this section are performed on Linux machine with 8 Intel(R) Xeon(R) X7560 octal core processors. The implementation of the parallel contig assembly framework is shown to be significantly beneficial for MIRA whole genome and EST assembler.

CHAPTER 4

CORRELATIVE ALGORITHM FOR REPEAT PLACEMENT (CARP)

1. Finding Repeating Sequences in Partially Assembled Genomes

The ever-increasing number of sequenced bacterial and archaeal genomes provides an opportunity to understand their architecture and evolution. However, as new high-throughput sequencing methods are developed, annotation quickly becomes the bottleneck for genomic research. In addition to open reading frames (ORFs) and regulatory elements, correct annotation of other features such as mobile genetic elements (MGEs) is also essential. These MGEs include bacteriophages, conjugative transposons, integrons, unit transposons, composite transposons and insertion sequences (ISs). Such transposable elements are defined as specific DNA segments that can repeatedly insert into one or more sites in one or more genomes. ISs are transposable elements that are regarded as genomic parasites proliferating in their host and surviving only through horizontal gene transfer (Schaack, *et al.*, 2010). ISs play a major role in genome evolution and plasticity, mediating gene transfers and promoting genome duplication, deletion and rearrangement (Frost, *et al.*, 2005). Insertion sequences may be abundant in host genomes and are intimately involved in mediating horizontal gene transfer, generation of pseudogenes, genomic rearrangement and alteration of regulatory elements (Frost, *et al.*, 2005; Schaack, *et al.*, 2010). Experimental evolution in the laboratory has demonstrated that both transpositions (Chou, *et al.*, 2009; Schneider, *et al.*, 2000) and rearrangements (Chou and Marx, 2012; Cooper, *et al.*, 2001; Dunham, *et al.*, 2002; Lee and Marx, 2012; Zhong, *et al.*, 2004) can generate beneficial mutations. Prokaryotic DDE

transposons (mainly ISs) can move in two different ways, depending on the donor site. Replicative transposons copy their DNA, leaving the parent site intact, while conservative transposons cut themselves out of the donor molecule in order to paste their DNA into the target.

Despite the development of various annotation programs for particular genomic features, some important features such as insertion sequences (ISs), the smallest and simplest autonomous mobile genetic elements, remain poorly annotated. In many cases, annotations of these elements include only ORFs and ignore terminal inverted repeats, which are an essential feature of their activity in mediating gene rearrangements. Moreover, partial ISs are rarely annotated, leading to the loss of potentially valuable evolutionary information. Another major limitation of current tools is the requirement of a complete annotated genome sequence for IS identification and analysis.

The majority of ISs are between 700-3000 bp and possess one or two open reading frames (ORFs) that encode transposases or helper proteins. For an IS element with more than one ORF, the first (upstream) ORF encodes a DNA recognition domain, while the second overlapping ORF encodes the catalytic domain. There are two types of IS: ISs carrying TIR (Terminal Inverted Repeats) elements; and ISs not carrying TIR elements. A TIR IS element carries a pair of partially conserved 7 to 20 bp inverted repeats at its terminus for cleavage and binding of the transposase. Upon insertion, ISs often generate short directed repeats from 2 to 14 bp immediately outside the IRs (Mahillon and Chandler, 1998). ISs of the non-TIR type do not have discernible conserved inverted repeats.

Metagenomic analysis has revealed that IS transposases are among the most

abundant and ubiquitous genes in nature (Aziz, *et al.*, 2010). Based on transposase sequence similarities, ISs have been classified in 25 different families that belong to three main classes of enzymes: DDE transposases; serine recombinases; and tyrosine recombinases (Mahillon and Chandler, 1998). Another recent classification of ISs categorizes them into 26 families based on transposase homology and overall organization, with some families divided further into groups (Zhou, *et al.*, 2008). An IS family can be defined as a collection of elements sharing conserved spacers between key residues, identical genetic organization, similar terminal sequence arrangements, and uniform target insertion behavior. However, not all families are so coherent. Consequently, some (e.g. families IS4 and IS5) are divided into subgroups composed of a core of closely related elements that can be linked to other members of the family by weaker but still significant similarities. The naming convention of transposable elements (insertion sequences, transposons, etc.) generally follows the recommendations of Campbell *et al.* (Chumley, *et al.*, 1979). However, in some cases a revised system of IS naming is used based on a registry where researchers can request for a new sequence number to define novel mobile elements (Roberts, *et al.*, 2008). IS and transposable element abundance in prokaryotes is highly variable (Touchon and Rocha, 2007) but they occupy a substantial fraction of some genomes. For example, 11% and 25% of the genome in *Clostridium difficile* and *Enterococcus faecalis* is composed of mobile elements (Paulsen, *et al.*, 2003; Sebaihia, *et al.*, 2006). Therefore, it is estimated that an average of up to 10% of bacterial (Mahillon and Chandler, 1998) and archaeal (Filée, *et al.*, 2007) genomes are comprised of MGEs.

Current IS-related software tools such as IScan and OASIS operate only on

complete genomes with fully annotated ORFs. Complete genome assembly of a single strain of bacteria can be time-consuming and costly, due in large part to ambiguities introduced by repetitive elements themselves. Consequently, most publicly available prokaryotic genomes are deposited as incomplete, contig- or scaffold-level assemblies, and IS and other repetitive elements may or may not be present in the deposited sequence. For example, Celera WGS (Myers, *et al.*, 2000), a widely used assembly software, commonly moves full or partial IS elements to a "degenerates" folder that is not frequently deposited as part of the draft genome. Therefore, to perform a global investigation of ISs in unassembled prokaryote genomes, we developed ISQuest (Biswas, *et al.*, 2015), or Insertion Sequence Quest, a computational tool for automated detection of ISs in unassembled or partially assembled genomes. ISQuest takes advantage of widely available transposase annotations to identify candidate IS seed regions and then uses a computationally efficient extension method based on BLAST (Altschul, *et al.*, 1990) to grow the seed regions and identify the edges of each IS element. ISQuest is capable of finding MGEs in hundreds of genomes within hours, making it a valuable high-throughput tool for a global search of IS elements. We applied ISQuest to 3810 sequenced bacterial genome and plasmid sequences. Compared to the benchmark of GenBank annotations, ISQuest identified 82% successfully with 80% sequence identity.

1.1. ISQuest Algorithm

ISQuest is a computationally efficient algorithm designed to find and annotate Insertion Sequences (IS) and transposases in fully assembled, partially assembled or unassembled genomes. The algorithm uses BLAST (Altschul, *et al.*, 1990) to determine potential IS locations by searching against an automatically curated database of IS and

transposase sequences derived from GenBank. The potential locations are further extended by Smith-Waterman alignment extension. The IS elements may occur once in a genome (single-copy) or may consist of a set of almost identical copies (multicopy). As there are distinct levels of information available in each of these cases, different algorithms perform better with each class. As such, we have designed ISQuest to find these two groups of ISs in two separate steps: first finding multicopy ISs and then single-copy ISs. The overall schematic pipeline is shown in Figure 5. The pipeline has been specially modeled to identify ISs but the algorithm is capable of detecting other mobile genetic elements (MGEs) and the generic steps are described below with IS elements as special cases.

1.1.1. Search Terms and TransposaseDB

ISQuest identifies single-copy and multicopy ISs and transposases in each genome by finding conserved regions of already-annotated transposase elements, which are identified by the word 'transposase', or 'insertion sequence' in the 'product' field of GenBank files. The search keywords may be extended by user-provided regular expressions since there is a significant amount of inconsistently annotated data in GenBank. For example, transposases are frequently misannotated as integrases. Generating the database of known MGEs is done once as a preprocessing step during the first run of ISQuest which generates a BLAST database called TransposaseDB. This database is stored for subsequent use by future executions. The user can force updates of the database when new versions of the GenBank files are available.
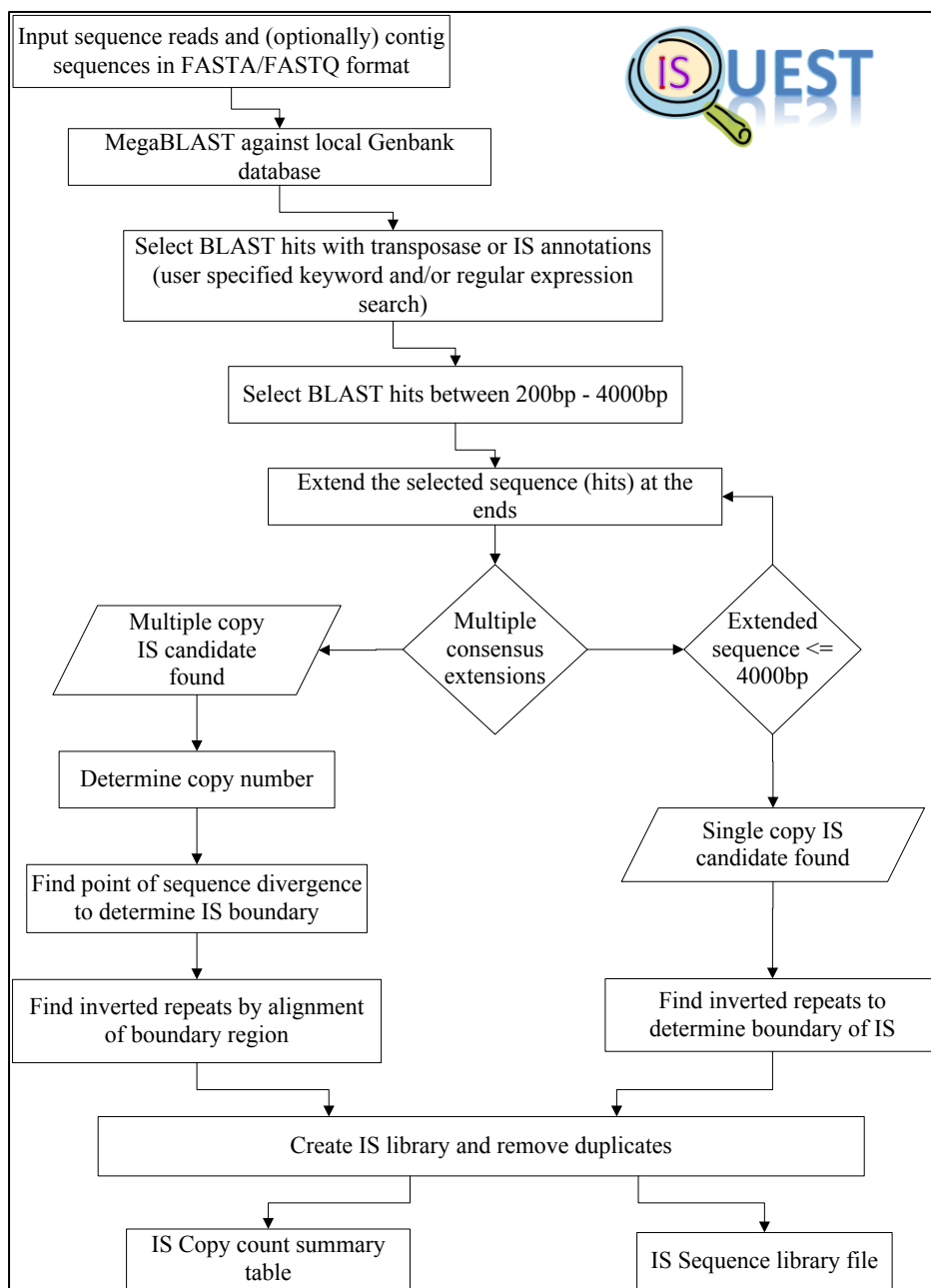
Fig. 5. Flowchart of the full workflow of ISQuest

1.1.2. BLAST Searching Parameters

A candidate sequence for extension is determined by a BLAST search against TransposaseDB. ISQuest can operate directly on raw reads provided in FASTA/FASTQ

format. Efficiency can be significantly improved by assembling the reads and providing a set of assembled contigs in FASTA format. This assembly can be performed using an appropriate assembler for the input reads. The assembled contigs are BLAST-searched against the TransposaseDB database to find potential seed locations for ISs and transposases. These seed locations represent all possible MGE locations that must be searched and analyzed. Therefore, we use MegaBLAST for finding matches with higher sequence similarity and better performance. Since we further extend these seed sequences to find the boundaries of the MGEs, we can tolerate partial or inexact matches.

1.1.3.   Extending Potential IS Matches

Once the possible MGE seed locations have been identified, raw reads are used to extend the seed sequences to determine boundaries. The extension is done by pairwise alignment of the raw reads to the ends of the seed sequence. This alignment algorithm is implemented using BLAST allowing 5 bit score errors. This parameter is configurable by the user depending on the sequencing technology used and the expected error profile of the reads. For Illumina reads we allowed a bit score error of 5, which corresponds to 98% sequence similarity using 250bp reads.

The extension step aligns all reads to the end of a seed sequence then executes the boundary detection step. The extension step does not align reads that do not have at least a partial overlap with the core seed sequence as we do not want to miss the boundary of the MGE by large extensions. Therefore, each extension step builds no more than twice the input read length. The seed sequence is expanded to include the aligned reads and the larger consensus sequence is used as the new seed. Therefore, the extension step is iteratively executed for the remaining sequences for which the boundary cannot be found

until the seed sequence becomes too long. The termination length of the seed sequence is user configurable and defaults to 4Kbp.

1.1.4. Determining IS Boundary

We apply different approaches to find the boundary of single- and multi-copy MGE elements. In the case of a single copy we can only find the boundary in cases where there are flanking inverted repeats. To define the edges of single-copy ISs, we use an approach first developed by IScan to find IRs around the transposases, which are present for the majority of ISs (Wagner, *et al.*, 2007). Briefly, a Smith–Waterman alignment, with a match score of 1, a mismatch penalty of -3 and a gap penalty of -4, is performed comparing the region upstream of the transposase (500 bp) with the reverse complement of the downstream region (500 bp) and the highest match with a score >10 is assumed to be the pair of terminal IRs.

Since the various copies of a multi-copy ISs are from different genomic loci, they have different unique sequence beyond the boundaries of the IS. Therefore, if the consensus of the aligned reads disagrees with the end of the seed sequence, this indicates that the boundaries of the IS have been reached. Based on the number of possible disagreements we calculate the number of possible sequence groups. If each group has coverage within a specified range we can be certain that we have reached the final boundary for all the sequence groups and have run into the flanking unique sequence. However, if a sequence group has coverage several times that of the expected coverage, we know that there exist longer MGEs the form of tandem repeats which will require further extension. These sequence groups are separated out for extension in the next iteration.

The sequence groups with appropriate coverage are processed to determine the IRs using a Smith–Waterman sequence alignment. The alignment parameters are the same as those described for the single copy IS case. In some cases the boundary defined by the IRs may disagree with the boundary defined by the synteny of the aligned reads due to nested repeats, flanking direct repeats at the ends, or inaccurate IR identification. ISQuest addresses this ambiguity by prioritizing the IR edges and changing the boundary to match the IRs. If IRs are found, a direct repeat finding subroutine attempts to align 10bp fragments on either side of the IRs to identify direct repeats. If no IRs are found, the edges of the MGE are solely determined by the alignment of the reads. This allows annotation of partial MGEs as many of these sequences do not have IRs. Thus, when present in multiple copies, ISquest finds partial ISs; it is not capable of finding these IS fragments when no intact copy with an annotated transposase is present in GenBank.

The same MGE element may result in one or more BLAST seeds and may cause redundant copies of the same IS to be generated. Therefore the redundant results within the final set are filtered out using a pairwise global alignment to identify groups of IS lengths, which are clustered together. The clustering algorithm groups sequences such that the mean lengths are within 100bp of each other. The cluster is then assumed to be the true copy size of the IS and any fragments that are shorter than that threshold are classified as partials.

1.1.5. Iterative Extension and Boundary Finding

Sequences with known boundaries are removed from the extension set and all remaining sequences are expanded based on the consensus of the reads aligned to the boundaries. Extension and boundary finding are performed iteratively until all seed

sequences have been processed. The end of each boundary finding step generates a new set of seed sequences. The new seed sequences are generated from the alignments that have no disagreement in the aligned reads, signifying that the boundary has not been reached. The consensus sequences generated from all these alignments is used as the fresh set of seeds in the extension step. Some new seed sequences may be derived from alignments with disagreements as well. In such cases, the alignment disagreements can be grouped such that some groups have a very large coverage. The consensus sequences generated from these large coverage groups are separated and treated as new seed sequences.

1.1.6. ISQuest Output

The output of the pipeline is a library of full and partial MGEs. IS elements in particular are composed of a transposase with one or more ORFs and appropriate upstream and downstream sequences. The extreme edges are annotated in GenBank format for IS elements and may include a partially conserved inverted repeat on each end ranging from 8 to 40 bp in length with direct repeats ranging from 4-8bp in length. Partial IS elements and other MGEs such as transposases do not have special annotations defining the boundary.

The final output of ISQuest includes two files for the given input of raw reads and contig(s): 1) a file in GenBank format listing each MGE and its characteristics, including the chromosome ID, start and end positions, direction, family and group, IRs (if found), DRs (if found) and whether the element is a partial element; and 2) a file containing the copy number of each identified IS in .csv format.

1.1.7. Using the ISQuest Tool

ISQuest is a free open source program implemented in C++. It is available at http://sourceforge.net/projects/isquest. ISQuest requires the read library of input reads in FASTA/FASTQ format and can be optionally provided with an assembly of the reads. The program accepts 4 command line parameters 1) the configuration file, 2) the raw reads, 3) the prefix of the output files and 4) the optional set of assembled contigs. The configuration file contains the required file paths to the local BLAST database and other configurable parameters such as the maximum number of iterations ISQuest performs, the maximum length of the MGEs to be built and the search terms for MGE's in GenBank. A complete wiki with required documentation is provided on the forge.

1.1.8. Preparation for ISQuest Tool Evaluation

To evaluate ISQuest we used 3810 microbial genomes and plasmid sequences > 100Kbp available in GenBank as of 15th October 2014. The ART tool was used to generate synthetic Illumina paired-end fragment libraries with read length of 250bp and 50× coverage. The read length of 250bp was used for experimentation because 250bp read lengths are typical for Illumina sequencing machines. ART simulates sequencing reads by mimicking real sequencing process with empirical error models or quality profiles summarized from large recalibrated sequencing data. ART can also simulate reads using a user specified error profile that requires the user to specify probability of sequencing errors at each base position of the read. ART was used as a primary tool for the simulation study of the 1000 Genomes Project (Huang, *et al.*, 2012). ISQuest performance was evaluated by first fragmenting each genome using the simulation process described above. We then used the Celera WGS assembler to assemble these simulated reads into contigs. The ISQuest algorithm was operated on these contig

sequences to generate a set of candidate MGEs. This run can be performed using the raw reads but will significantly slow down the execution. In addition, we ensure that the ISQuest testing algorithm does not include the genomes being processed in the search database to ensure that the test and training sets are disjoint.
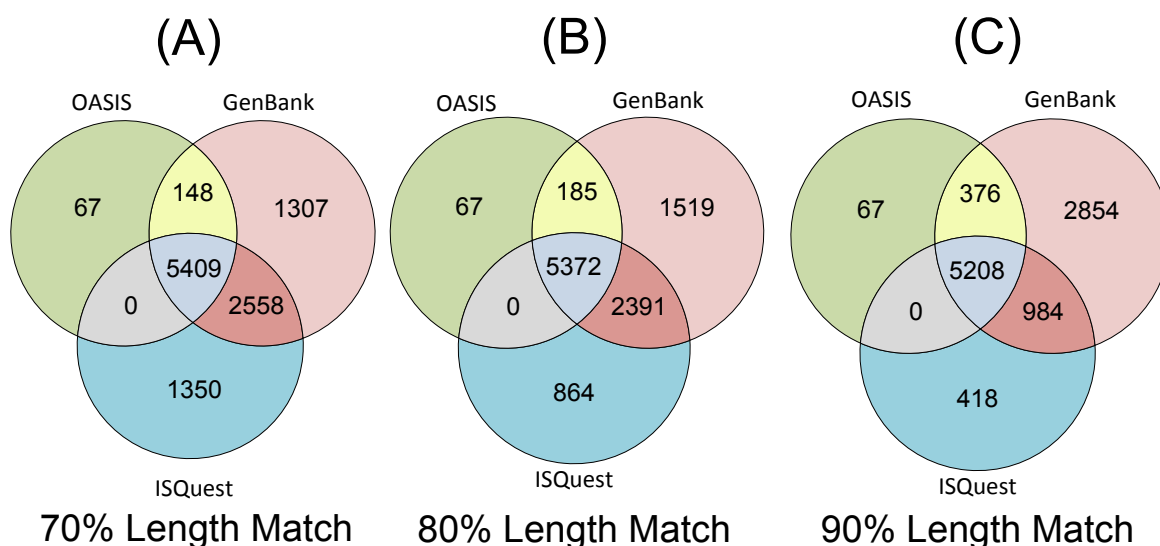
Fig. 6. Venn diagram illustrating the number of IS annotations identified by ISQuest and OASIS compared to GenBank at three length match thresholds. (A) ISQuest and OASIS both found 5409 ISs (in single copies) in the 3810 GenBank benchmarked genomes and plasmids. Additionally, ISQuest identified 2558 ISs that OASIS did not annotate and OASIS found 148 ISs that ISQuest failed to detect. OASIS found 67 insertion sequences that were not correctly annotated in GenBank as IS. ISQuest generated 1350 partial IS sequences that have not been annotated in GenBank. The intersection of ISQuest and OASIS is 0 as ISQuest cannot identify any sequence that has not been annotated in more than one GenBank submission using the keywords 'transposase', or 'insertion sequence' in the 'product' field. ISQuest does not take the annotated genome as input and therefore requires similar annotation to be present in other submissions. (B) same as (A) but only allowing 80% length matches as true positives. (C) same as (A) but only allowing 90% length matches as true positives.

1.2. ISQuest Test Results

We performed two experiments to show the MGE detection capability of ISQuest and present a summary of IS sequences found by ISQuest classified by IS family. The performance of the ISQuest tool was compared to that of OASIS using annotated transposases in GenBank as a benchmark. This first experiment compared the accuracy of ISQuest and OASIS by measuring the percentage of GenBank annotated ISs found by each tool. Unlike ISQuest, OASIS operates on completely assembled and annotated genomes and uses only the annotation information available in the genome. ISQuest operates on partially assembled contigs or directly on the raw reads and does not require annotation to identify the ORFs. This experiment shows the predictive capability of ISQuest to find ISs from a draft and un-annotated assembly and compares it to the predictive capability of OASIS using completely annotated sequences. The capability of ISQuest to find other repetitive elements (e.g. rRNA operons) is not measured in this experiment.

As ISQuest uses an un-annotated draft genome, ORFs are not clearly defined and finding the exact lengths of the MGEs is difficult using the seed extension algorithm. Therefore, due to these inaccuracies, the testing result in Figure 6(A) considers 70% sequence length match as a true positive; if ISQuest returns a sequence that matches a 70% of the length of an annotated sequence in GenBank with 95% sequence similarity we consider it a true positive. The count numbers in the figure represent IS counts in single copy; multiple copies of a particular IS are not included. Within the 3810 benchmarked genomes and plasmids, ISQuest found 84.5% of the 9422 unique GenBank annotations, whereas OASIS found 58.9%. The 5346 GenBank ISs found both by

ISQuest and OASIS represent insertion sequences with well-defined inverted repeats. The 2558 sequences found by ISQuest and also present in GenBank are full and partial transposase elements that do not contain completely defined inverted repeats and therefore cannot be identified by OASIS. The 1350 annotations found only by ISQuest include partially assembled insertion sequences and partial MGEs found by ISQuest that have not been annotated in deposited genomes. These sequences may also include potential sets of new insertion sequence and transposase elements identified by ISQuest based on sequence similarity to other ISs in GenBank. The intersection of ISQuest and OASIS is zero as ISQuest cannot identify any sequence that has not been annotated in more than one GenBank submission using the keywords 'transposase', or 'insertion sequence' in the 'product' field. ISQuest does not take the annotated genome as input and therefore requires similar annotation to be present in other submissions.

We further evaluated ISQuest under increasingly strict constraints by increasing the length match threshold which we accept as a true positive to 80% and 90% of the sequence length (see Figure 6). Figure 6(B) shows the results of considering only sequences with greater than or equal to 80% length matches with 95% sequence similarity with GenBank sequences as valid true positives of ISQuest. We notice a slight reduction in the number of insertion sequences detected by ISQuest to 82.2% of the 9422 unique GenBank annotations. Increasing the length match threshold to 90% (see Figure 6(C)) shows significant reduction in the number of insertion sequences detected by ISQuest to 65.7%. However, this shows that ISQuest is able to reproduce 90% of the actual IS sequence using the fast seed extension algorithm in the majority of cases.

Table 16. ISQuest annotations compared to GenBank annotations grouped by Phylum at 80% length match threshold

Phylum	Number of Genomes[a]	Number of GB IS[b]	Number of GB TP[c]	Number of ISQ IS[d]	ISQ TP[e]
Proteobacteria	1810	22375	31918	18412	14164
Firmicutes	794	7906	11029	6297	4962
Actinobacteria	520	4029	7970	3416	3513
Cyanobacteria	128	1590	3674	1267	1534
Bacteroidetes	92	1016	1342	858	582
Tenericutes	53	434	468	321	226
Spirochaetes	48	357	569	264	253
Deinococcus-Thermus	47	283	323	188	160
Others	318	3754	3097	2712	1373
Total	3810	41564	60309	33735	26767

[a]The number of genomes under each phylum.
[b]The number of IS annotations(multiple copies) in GenBank.
[c]The number of Transposase annotations in (multiple copies) GenBank.
[d]The number of IS detected (multiple copies) detected by ISQuest.
[e]The number of Transposase detected (multiple copies) detected by ISQuest.

1.2.1. MGE Detection using ISQuest

In order to study the overall sensitivity and specificity of ISQuest we directly compared its output to GenBank. Comparison to OASIS is problematic as OASIS only identifies insertion sequences with clearly defined inverted repeats. ISQuest can identify full ISs, partial ISs and other MGEs such as transposases. Table 16 shows the IS sequences found by ISQuest grouped by phylum. The numbers in the table represent ISs in multiple copies, i.e., the multiple copies of the IS are included (collapsed). Likely because of the number of sequenced genomes from Proteobacteria and Firmicutes, >50% of the ISs we found are from Proteobacteria and an additional 16% are from Firmicutes (Table 16, Column 3). ISQuest detected 82.2% of the Proteobacteria ISs and 81.1% on average from GenBank (Table 16, column 3, 5). The prediction capability of ISQuest is limited by the assumption that a similar annotation of the IS element is present in other genomes. So, in some cases we cannot identify certain ISs correctly due to sequence

divergence or absence of annotation. Also, the copy number computation based on the number of possible flanking unique sequence regions is conservative in estimating the number of copies and reduces the copy count to the least possible value.

ISQuest was also used to identify transposase elements and the sequences generated by ISQuest without clearly defined inverted repeats were compared to transposase annotations in GenBank. Similar to IS elements, Proteobacteria and Firmicutes account for majority of the transposase annotation in GenBank (52.3% and 18.3% respectively). ISQuest detected 57.7% of the Proteobacteria transposases and 44.4% of transposases from GenBank (Table 16, column 4, 6). The significantly lower detection accuracy relative to ISs is due to the presence of single copy transposases. These elements do not possess inverted repeats, and in single copy cases, do not possess multiple unique flanking sequences; therefore, their length cannot be estimated by ISQuest. Such single copy elements with no discernable end regions are extended to the default maximum length and often include unique sequence that does not match an existing transposase element from GenBank.

1.2.2. MGE Detection using ISQuest

It was also interesting to study the performance of ISQuest in terms of the IS families discovered. This provided insight into the annotations and predictive capability of ISQuest for mining ISs from families with high divergence. Table 17 shows the top 20 IS families detected, some of which are predicted better than others due to the inherent divergence in the IS families and inaccurate annotations from GenBank. IS4 family is the most annotated IS family in GenBank with a total of 5521 annotations. ISQuest identified the IS elements in IS4 family with ~ 60% accuracy which is significantly less that overall

accuracy of ISQuest. This is due to the high internal divergence of IS4 elements that makes classification and identification challenging.

Table 17. ISQuest annotations compared to GenBank annotations group by IS type

IS Family[a]	Number of GB[b]	Number of ISQ[c]	Percentage[d]	IS Family[e]	Number of GB[b]	Number of ISQ[c]	Percentage[d]
IS4	5521	3340	60.5	**IS110**	308	308	100
IS911	2496	1872	75	**ISL3**	308	298	96.8
IS902	1738	1603	92.2	**IS21**	233	232	99.6
IS3	1061	1060	99.9	**IS982**	229	171	74.7
IS5	772	679	88	**IS256**	223	222	99.6
IS66	568	426	75	**IS200**	190	190	100
IS1165	491	367	74.7	**IS1341**	146	146	100
IS605	377	376	99.7	**IS6**	98	98	100
IS30	362	361	99.7	**IS1182**	75	55	73.3
IS630	337	252	74.8	**IS1595**	55	54	98.2

[a]The top 10 IS families annotated in GenBank.
[b]The number of IS annotations (single copy) in GenBank.
[c]The number of IS detected (single copy) by ISQuest.
[d]The percentage IS detected (single copy) by ISQuest.
[e]The top 11-20 IS families annotated in GenBank.

Overall, 60,502 MGE elements representing 9317 unique IS sets and 26767 transposase annotations were identified by ISQuest in 3810 genomes and plasmids. ISQuest took a total of 23 h and 44 min to annotate all 3810 genomes on a 4x Intel Xenon X7550, 2.0-Ghz processor using partially assembled contigs. The maximum per-genome running time was 8 min.

2. Correlative Algorithm for Repeat Placement

The Correlative Algorithm for Repeat Placement (CARP) finishing tool we propose is based on the novel idea of assembling repeat elements separately from the rest of the genome, then placing these elements correctly within the draft genome using several lines of evidence to ensure that the correct placement is made. Evidence types include: 1) presence of incomplete repeat element fragments on the ends of unjoined

contigs, 2) mate-pair evidence, and 3) synteny (similarity in gene organization) with reference genomes. Importantly, any joins made by this method are presented to the user along with the evidence used to make the joins (see Figure 7). De-novo genome assembly from DNA fragments is primarily based on sequence overlap information.
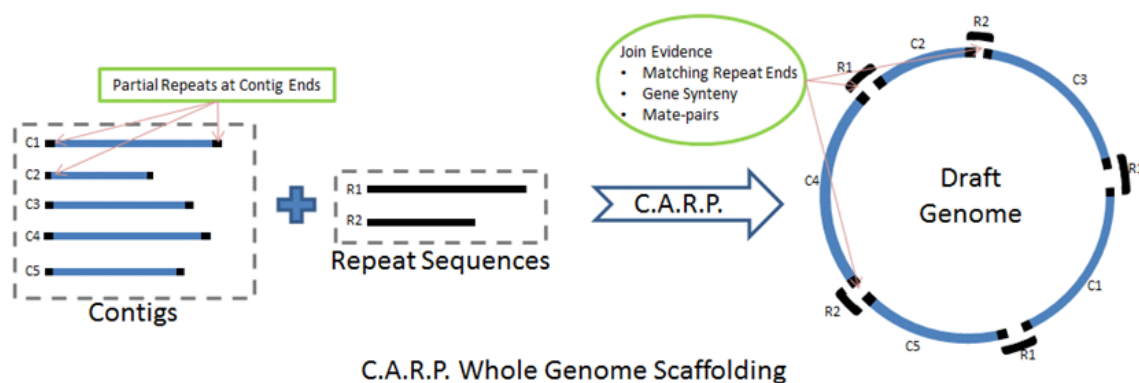

Fig. 7. Illustration of CARP scaffolding

In addition, mate-pair reads or paired-end reads provide linking information for joining gaps and bridging repeat regions. Genome assemblers in general assemble long contiguous sequences (contigs) using both overlapping reads and linked reads until the assembly runs into an ambiguous repeat region. These contigs are further bridged into scaffolds using linked read information. However, errors can be made in both phases of assembly due to high error threshold of overlap acceptance and linking based on too few mate reads. Identical as well as similar repeat regions can often cause errors in overlap and mate-pair evidence. In addition, the problem of setting the correct threshold to minimize errors and optimize assembly of reads is not trivial and often requires a time-consuming trial and error process. Therefore, we propose a novel scaffolding tool, Correlative Algorithm for Repeat Placement (CARP), capable of joining low error
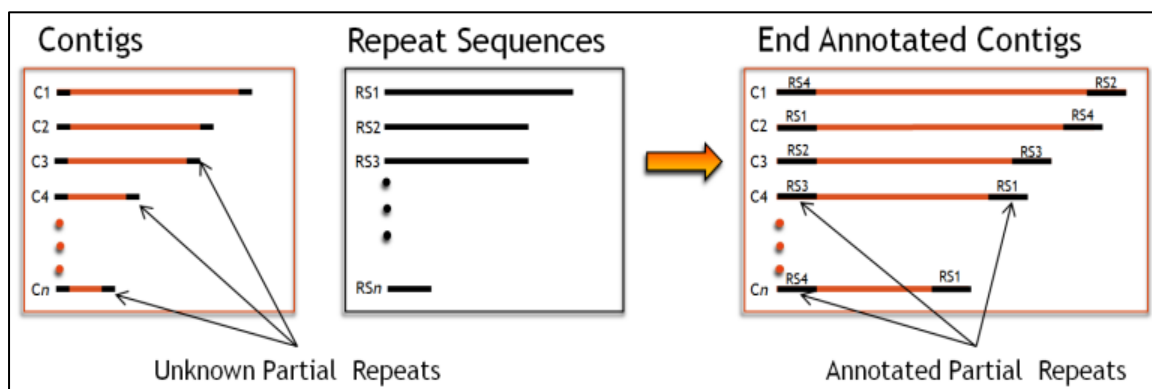
Fig. 8. Correlative Algorithm for Repeat Placement (CARP) step 1. The partial repeat elements flanking the contigs are identified.

contigs using mate pair reads, resolved repeat structures and verification of joins based on synteny with one or more reference organisms. The CARP tool requires a set of long repeat sequences such as insertion sequences that can be manually determined or found computationally. The tool is designed to match very low error contigs with strong overlap using the ambiguous partial repeat sequence at the ends of the contig. These matches are verified by synteny with reference to one or more related organisms. We show that the CARP tool can be used to verify low mate pair evidence regions, independently find new joins and significantly reduce the number of scaffolds.

2.1. Annotating the Partial Repeats at Each Contig Ends

The CARP tool requires as input a set of high quality contigs that are generally flanked by a partial repeat region that were not assembled by the assembler due to ambiguous choices. Figure 8 shows the annotation of partial repeats at the contig ends using the computationally determined repeats from ISQuest.
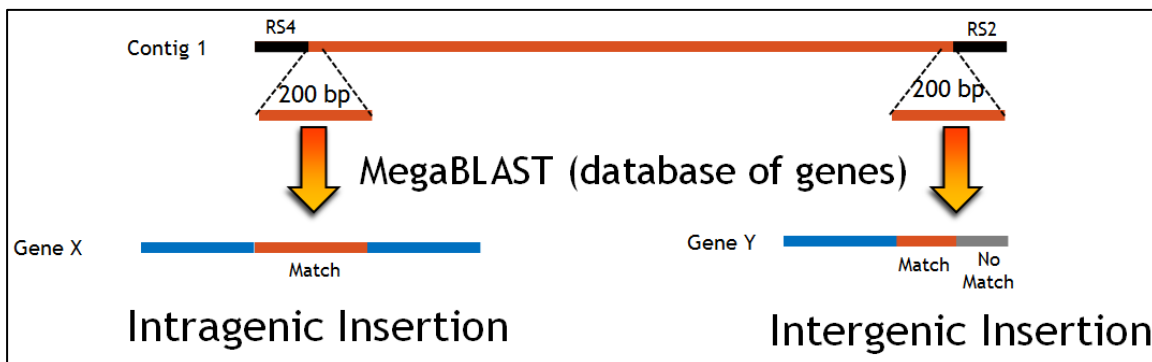
Fig. 9. Correlative Algorithm for Repeat Placement (CARP) step 2. The unique regions around the partial repeat elements are identified by BLAST to determine intergenic or interrupting insertions.

2.2. Identification of Intergenic or Interrupting Repeat Insertion

The annotation of flanking repeats reduced the match possibilities from $O(n^2)$. The possibilities can be further reduced based on the unique sequence flanking the partial repeat sequences. The flanking 200bp of the sequence are BLAST searched against a database of genes. The first case is intergenic insertion where the insertion sequence interrupts a gene. In that case, the BLAST search will return a match within a gene. However, if the sequence does not hit inside a sequence we have an intergenic insertion (see Figure 9). In case of intergenic insertion, we can match the contigs based on synteny with a reference genome.

2.3. Matching the Contigs Based on Lines of Evidence

Based on the first two steps we have two lines of evidence to make joins. We can first pair contigs based on matching complementary partial repeats at the contig ends. The number of possible pairs can be further reduced based on interrupted gene sequences and synteny with are reference (see Figure 10).
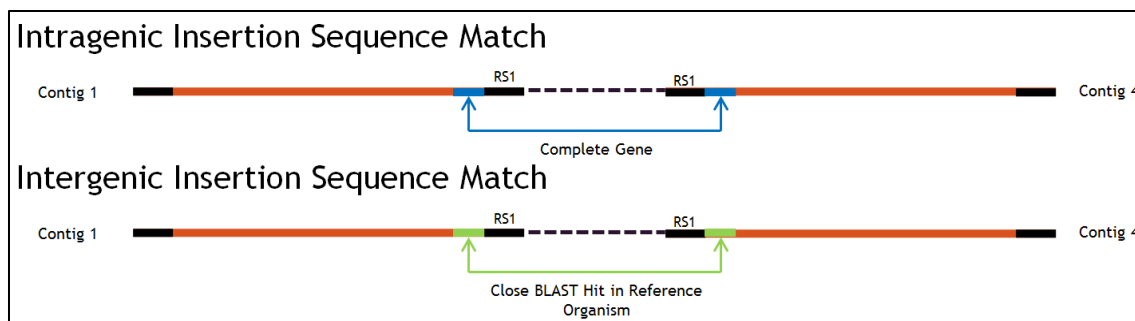
Fig. 10. Correlative Algorithm for Repeat Placement (CARP) step 3. The unique regions around the partial repeat elements are identified by BLAST to determine intergenic or interrupting insertions.

2.4. CARP Results

We experimented with CARP by selecting 12 genomes with high repeating regions (see Table 18). The ART tool was used to generate synthetic Illumina paired-end fragment libraries with read length of 250bp and 50× coverage. The read length of 250 bp is typical of Illumina sequencing machines and was selected for experimentation. ART simulates sequencing reads by mimicking real sequencing process with empirical error models or quality profiles summarized from large recalibrated sequencing data. ART can also simulate reads using a user specified error profile that requires the user to specify probability of sequencing errors at each base position of the read. ART was used as a primary tool for the simulation study of the 1000 Genomes Project (Huang, *et al.*, 2012). CARP performance was evaluated by first fragmenting each genome using the simulation process described above. We then used the Celera WGS assembler to assemble these simulated reads into low error overlap only contigs. The ISQuest algorithm was operated on these contig sequences to generate a set of candidate MGEs. CARP was used to call joins using the MGEs and the contigs. The Celera scaffolder was also used to call the joins and the results are compared (see Table 18). Both the scaffolders are checked for

incorrect joins by comparing the scaffolds to the original sequence of the genome. We can see that CARP consistently generates fewer scaffolds and fewer incorrect joins as compared to Celera WGS. For example, CARP was able to derive a single circular sequence genome with no errors for *M. marinum M*, where Celera WGS derived only 48 scaffolds.

Table 18. CARP finishing results

Organism[a]	Len[b]	Low Error Contigs[c]	Celera Scaffolds[d]	Incorrect Joins (Celera)[e]	CARP Scaffolds[f]	Incorrect Joins (CARP)[g]
M. nodulans	7.7	482	56	3	24	0
T. erythraeum	7.7	279	37	0	4	0
M. vanbaalenii	6.4	12	1	0	1	0
M. marinum M	6.3	773	48	5	1	0
M. acetivorans	5.7	28	20	0	6	0
B. halodurans	4.2	857	92	16	38	7
A. aurescens	4.5	683	97	7	21	2
M. silvestris	4.3	22	12	0	15	0
S. maltophilia	4.8	1289	266	31	49	1
R. rubrum	4.3	42	8	1	3	0
M. hungatei	3.5	654	107	14	19	4
H. marismortui	3.4	22	2	0	1	0

[a]The top 12 genomes with repetitive IS sequences.
[b]The length of the genome in million base pairs.
[c]The number of low error contigs assembled by Celera WGS assembler with only overlap information.
[d]The number of scaffolds generated by Celera using mate-pair reads.
[e]The number of scaffolds generated by Celera that do not match the original genome.
[e]The number of scaffolds generated by CARP using repeat placement and mate-pair reads.
[f]The number of scaffolds generated by CARP that do not match the original genome.

3. Unverified Join Viewer

The Unverified Join Viewer (UJV) was developed to help users track join information from CARP and update join data as per user requirement. This of this project was implemented by a team of undergraduate students as part of the CS410 & CS 411 Profession Workforce Development course requirements under the guidance of Abhishek Biswas (Biswas, et al., 2015).
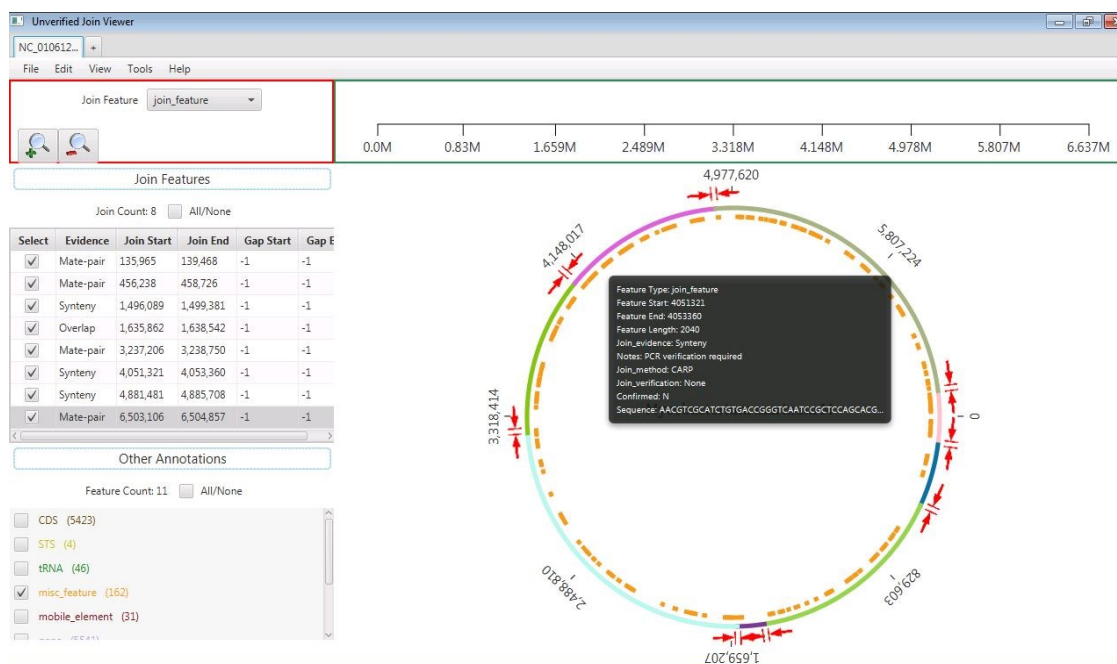
Fig. 11. Unverified Join Viewer: genome display and joins

The UJV viewer displays the bacterial genome in a circle after loading the GenBank file generated from CARP with join points annotated using the "join_feature". The joins annotated as "join_feature" in the GenBank file are shown on the outer periphery of the genome circle (see Figure 11). The joins shown in red have not been manually verified and confirmed and require further user review. The joins that have been have been manually verified and confirmed are displayed in blue. The user can use the mouse to linger over the join features to see the join information as a tooltip. The side panel can be used to view the join coordinates and control the central view. The other annotations of the genome are also shown in the inner periphery of the circle and are color coded. The lower side panel can be used to select the features to be displayed. The annotation features also have tooltips that the user may use to view the annotation details. Figure 11 shows a screenshot of the *M. Marinum M* genome artificially fragmented and

assembled back using CeleraWGS and CARP. The user is currently viewing particular

join information using tooltip and the "misc_feature" annotation is selected for view.
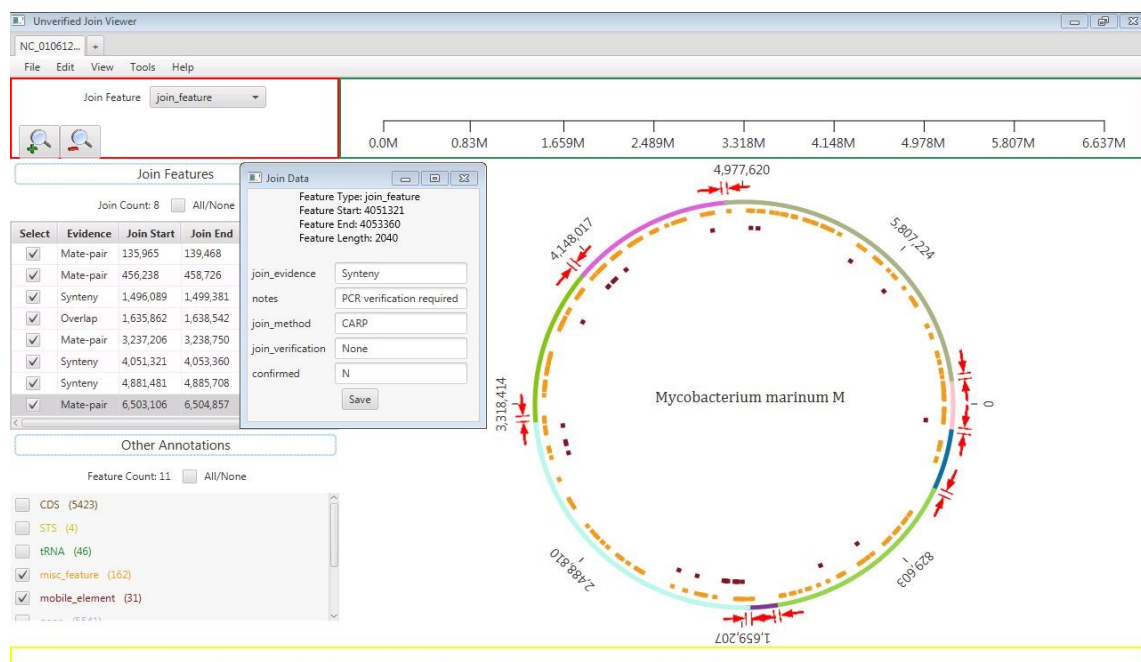
Fig. 12. Unverified Join Viewer: genome display and editable join information

The user can click on a join GUI element to edit the join information and track the

progress of the genome finishing process. The edited join annotations can be used to

track manual verification and validation process and rules can be set to confirm joins.

Clicking on a join feature opens an editable window where join related information could

be modified (see Figure 12). The application allows the user to re-order the contigs and

generate a NCBI compatible GenBank file of the bacterial genome for publication.

CHAPTER 5

CONCLUSION

This work focuses on providing a major sequence assembly resource to small- and mid-scale laboratories that may not have access to bioinformatics expertise and infrastructure available at larger institutions. Even small (e.g. prokaryote) genome projects can be challenging tasks for researchers without bioinformatics core facilities to call upon for expertise and advice. In our experience, choice of an appropriate assembler for prokaryotic genomes is often hampered by lack of information regarding how individual assemblers deal with various genomic structures, and researchers are often forced to "take a guess" about which assembler to use, or allow considerations of computational resources or user-friendliness to make their decision for them. When one considers the large amount of effort required to produce a finished sequence from a draft assembly, the inefficiency imposed by an inappropriate assembler creates clear problems. Further, the current lack of ability to bring together a comprehensive suite of assembly statistics creates a large potential for misassembled "final" sequences to make their way into public databases. Therefore, researchers without ready access to teams of trained bioinformaticists face a lack of centralized information and tools with which to generate sequence assemblies, and more importantly, to judge the quality of assemblies they generate. We present tools that develop this resource, and therefore to improve the accessibility of small-scale accurate genome assembly to a larger user base. This activity therefore has a wide variety of potential broader impacts. Generation of sequence resources, including finished genomes, is applicable to a wide variety of scientific

endeavor, including human and veterinary health (i.e. bacterial pathogens), environmental remediation (e.g. hydrocarbon-degrading organisms), and microbial ecology. While focus of large sequencing centers has shifted to resequencing large numbers of strains of already highly studied organisms (e.g. *Escherichia coli*), there is still considerable interest in the scientific community for development of genomic resources in less-well characterized prokaryotic taxa. Our goal is to facilitate this research by making accurate and efficient prokaryotic genome assembly more accessible to a wider range of laboratories.

REFERENCES

Ablordey, A*., et al.* (2005) Comparative nucleotide sequence analysis of polymorphic variable-number tandem-repeat loci in *Mycobacterium ulcerans*, *Journal of Clinical Microbiology*, **43**, 5281-5284.

Altschul, S*., et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Research*, **25**, 3389 - 3402.

Altschul, S.F*., et al.* (1990) Basic local alignment search tool, *Journal of Molecular Biology*, **215**, 403-410.

Aziz, R., Breitbart, M. and Edwards, R. (2010) Transposases are the most abundant, most ubiquitous genes in nature, *Nucleic Acids Research*, **38**, 4207-4217.

Balzer, S*., et al.* (2010) Characteristics of 454 pyrosequencing data—enabling realistic simulation with flowsim, *Bioinformatics*, **26**, i420-i425.

Bao, Z. and Eddy, S.R. (2002) Automated De Novo Identification of Repeat Sequence Families in Sequenced Genomes, *Genome Research*, **12**, 1269-1276.

Barker, M.S*., et al.* (2009) SCARF: maximizing next-generation EST assemblies for evolutionary and population genomic analyses, *Bioinformatics*, **25**, 535-536.

Batzoglou, S*., et al.* (2002) ARACHNE: A Whole-Genome Shotgun Assembler, *Genome Research*, **12**, 177-189.

Biswas, A*., et al.* (2015) Unverified Join Viewer. *Virginia Branch American Society for Microbiology Annual Meeting*. Virginia Branch American Society for Microbiology, Richmond, VA.

Biswas, A*., et al.* (2013) Correlative Algorithm for Repeat Placement. *Virginia Branch American Society for Microbiology Annual Meeting*. Virginia Branch American Society for Microbiology, Charlottesville, VA, pp. 13.

Biswas, A*., et al.* (2015) ISQuest: Finding Insertion Sequences in Prokaryotic Sequence Fragment Data, *Bioinformatics*.

Biswas, A., Ranjan, D. and Zubair, M. (2011) Parallelization of MIRA Whole Genome and EST Sequence Assembler, *Workshop on Parallel Algorithms and Software for Analysis of Massive Graphs (ParGraph)*.

Biswas, A., Ranjan, D. and Zubair, M. (2012) A Parallel Genome Assembler Based on an Overlap Layout Consensus Graph Model. Asia Pacific Bioinformatics Conference (Under Review).

Biswas, A., Ranjan, D. and Zubair, M. (2013) Genome Assembly on a Multicore System. *The 11th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA-13)*. Melbourne.

Blocks, I.T.B. (2011) Threading Building Blocks.

Boisvert, S., Laviolette, F. and Corbeil, J. (2010) Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies, *Journal of computational biology : a journal of computational molecular cell biology*, **17**, 1519-1533.

Butler, J*., et al.* (2008) ALLPATHS: De novo assembly of whole-genome shotgun microreads, *Genome Research*, **18**, 810-820.

Cahill, M.J*., et al.* (2010) Read Length and Repeat Resolution: Exploring Prokaryote Genomes Using Next-Generation Sequencing Technologies, *PLoS One*, **5**, e11518.

Cerveau, N*., et al.* (2011) Short- and Long-term Evolutionary Dynamics of Bacterial Insertion Sequences: Insights from Wolbachia Endosymbionts, *Genome Biology and Evolution*, **3**, 1175-1186.

Chaisson, M.J., Brinza, D. and Pevzner, P.A. (2009) De novo fragment assembly with short mate-paired reads: Does the read length matter?, *Genome Research*, **19**, 336-346.

Chevreux, B. (2005) MIRA: An Automated Genome and EST Assembler. In German, Department and Head (eds). German Cancer Research Center Heidelberg.

Chevreux, B., Wetter, T. and Suhai, S. (1999) Genome Sequence Assembly Using Trace Signals and Additional Sequence Information. *German Conference on Bioinformatics*. Hannover, Germany, pp. 45-56.

Chin, C.-S*., et al.* (2011) The Origin of the Haitian Cholera Outbreak Strain, *New England Journal of Medicine*, **364**, 33-42.

Chou, H.-H., Berthet, J. and Marx, C.J. (2009) Fast Growth Increases the Selective Advantage of a Mutation Arising Recurrently during Evolution under Metal Limitation, *PLoS Genet*, **5**, e1000652.

Chou, H.-H. and Marx, Christopher J. (2012) Optimization of Gene Expression through Divergent Mutational Paths, *Cell Reports*, **1**, 133-140.

Chumley, F.G., Menzel, R. and Roth, J.R. (1979) Hfr FORMATION DIRECTED BY Tn10, *Genetics*, **91**, 639-655.

Cooper, V.S.*, et al.* (2001) Mechanisms Causing Rapid and Parallel Losses of Ribose Catabolism in Evolving Populations of Escherichia coli B, *Journal of Bacteriology*, **183**, 2834-2841.

Darling, A.E., Mau, B. and Perna, N.T. (2010) progressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement, *PLoS One*, **5**, e11147.

Dunham, M.J.*, et al.* (2002) Characteristic genome rearrangements in experimental evolution of Saccharomyces cerevisiae, *Proceedings of the National Academy of Sciences*, **99**, 16144-16149.

Earl, D.*, et al.* (2011) Assemblathon 1: A competitive assessment of de novo short read assembly methods, *Genome Research*, **21**, 2224-2241.

Eid, J.*, et al.* (2009) Real-Time DNA Sequencing from Single Polymerase Molecules, *Science*, **323**, 133-138.

Engle, M.L. and Burks, C. (1994) GenFrag 2.1: new features for more robust fragment assembly benchmarks, *Computer applications in the biosciences : CABIOS*, **10**, 567-568.

Filée, J., Siguier, P. and Chandler, M. (2007) Insertion sequence diversity in archaea, *Microbiology and molecular biology reviews : MMBR*, **71**, 121-157.

Finn, R.D.*, et al.* (2014) Pfam: the protein families database, *Nucleic Acids Research*, **42**, D222-D230.

Frost, L.S.*, et al.* (2005) Mobile genetic elements: the agents of open source evolution, *Nat Rev Micro*, **3**, 722-732.

Galardini, M.*, et al.* (2011) CONTIGuator: a bacterial genomes finishing tool for structural insights on draft genomes, *Source code for biology and medicine*, **6**, 11.

Gnerre, S.*, et al.* (2011) High-quality draft assemblies of mammalian genomes from massively parallel sequence data, *Proceedings of the National Academy of Sciences*, **108**, 1513-1518.

Gordon, D., Desmarais, C. and Green, P. (2001) Automated Finishing with Autofinish, *Genome Research*, **11**, 614-625.

Gough, J. and Chothia, C. (2002) SUPERFAMILY: HMMs representing all proteins of known structure. SCOP sequence searches, alignments and genome assignments, *Nucleic Acids Research*, **30**, 268-272.

Grabherr, M.G.*, et al.* (2011) Full-length transcriptome assembly from RNA-Seq data without a reference genome, *Nat Biotech*, **29**, 644-652.

Green, P. (1996) PHRAP Documentation. University of Washington, Seattle, WA.

Grillo, G.*, et al.* (1996) CLEANUP: a fast computer program for removing redundancies from nucleotide sequence databases, *Computer applications in the biosciences : CABIOS*, **12**, 1-8.

Hernandez, D.*, et al.* (2008) De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer, *Genome Research*, **18**, 802-809.

Hossain, M., Azimi, N. and Skiena, S. (2009) Crystallizing short-read assemblies around seeds, *BMC bioinformatics*, **10**, S16.

Huang, W.*, et al.* (2012) ART: a next-generation sequencing read simulator, *Bioinformatics*, **28**, 593-594.

Idury, R.M. and Waterman, M.S. (1995) A new algorithm for DNA sequence assembly, *Journal of computational biology*, **2**, 291-306.

Imelfort, M. and Edwards, D. (2009) De novo sequencing of plant genomes using second-generation technologies, *Briefings in Bioinformatics*, **10**, 609-618.

Jackson, B.G.*, et al.* (2010) Parallel de novo assembly of large genomes from high-throughput short reads. *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. pp. 1-10.

Kamoun, C.*, et al.* (2013) Improving prokaryotic transposable elements identification using a combination of de novo and profile HMM methods, *BMC genomics*, **14**, 700.

Kececioglu, J. and Myers, E. (1995) Combinatorial algorithms for DNA sequence assembly, *Algorithmica*, **13**, 7-51.

Kichenaradja, P.*, et al.* (2010) ISbrowser: an extension of ISfinder for visualizing insertion sequences in prokaryotic genomes, *Nucleic Acids Research*, **38**, D62-D68.

Kingsford, C., Schatz, M. and Pop, M. (2010) Assembly complexity of prokaryotic genomes using short reads, *BMC bioinformatics*, **11**, 21.

Kislyuk, A.O.*, et al.* (2010) A computational genomics pipeline for prokaryotic sequencing projects, *Bioinformatics*, **26**, 1819-1826.

Koren, S.*, et al.* (2012) Hybrid error correction and de novo assembly of single-molecule sequencing reads, *Nat Biotech*, **30**, 693-700.

Kumar, S. and Blaxter, M. (2010) Comparing de novo assemblers for 454 transcriptome data, *BMC Genomics*, **11**, 571.

Kurtz, S.*, et al.* (2004) Versatile and open software for comparing large genomes, *Genome biology*, **5**, 1-9.

Leclercq, S. and Cordaux, R. (2011) DO PHAGES EFFICIENTLY SHUTTLE TRANSPOSABLE ELEMENTS AMONG PROKARYOTES?, *Evolution*, **65**, 3327-3331.

Lee, M.-C. and Marx, C.J. (2012) Repeated, Selection-Driven Genome Reduction of Accessory Genes in Experimental Populations, *PLoS Genet*, **8**, e1002651.

Li, X. and Waterman, M.S. (2003) Estimating the Repeat Structure and Length of DNA Sequences Using ℓ-Tuples, *Genome Research*, **13**, 1916-1922.

Lin, Y.*, et al.* (2011) Comparative studies of de novo assembly tools for next-generation sequencing technologies, *Bioinformatics*, **27**, 2031-2037.

Mahillon, J. and Chandler, M. (1998) Insertion sequences, *Microbiology and molecular biology reviews : MMBR*, **62**, 725-774.

Margulies, M.*, et al.* (2005) Genome sequencing in microfabricated high-density picolitre reactors, *Nature*, **437**, 376-380.

Michael Schatz, D.S., David Kelley and Mihai Pop and (2010) Contrail: Assembly of Large Genomes using Cloud Computing.

Miller, J.R.*, et al.* (2008) Aggressive assembly of pyrosequencing reads with mates, *Bioinformatics*, **24**, 2818-2824.

Miller, J.R., Koren, S. and Sutton, G. (2010) Assembly algorithms for next-generation sequencing data, *Genomics*, **95**, 315-327.

Moretti, C.*, et al.* (2009) Highly scalable genome assembly on campus grids. *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*. ACM, Portland, Oregon, pp. 1-10.

Myers, E.W. (2005) The fragment assembly string graph, *Bioinformatics*, **21**, ii79-ii85.

Myers, E.W.*, et al.* (2000) A Whole-Genome Assembly of Drosophila, *Science*, **287**, 2196-2204.

Myers, G. (1999) A Dataset Generator for Whole Genome Shotgun Sequencing. *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, pp. 202-210.

Nagarajan, N. and Pop, M. (2009) Parametric complexity of sequence assembly: theory and applications to next generation sequencing, *Journal of computational biology : a journal of computational molecular cell biology*, **16**, 897-908.

Narzisi, G. and Mishra, B. (2011) Comparing De Novo Genome Assembly: The Long and Short of It, *PLoS One*, **6**, e19175.

Narzisi, G. and Mishra, B. (2011) Scoring-and-unfolding trimmed tree assembler: concepts, constructs and comparisons, *Bioinformatics*, **27**, 153-160.

OpenMP (2008) *OpenMP Application Program Interface*.

Pandey, V., Nutter, R.C. and Prediger, E. (2008) Applied Biosystems SOLiD™ System: Ligation-Based Sequencing. In, *Next Generation Genome Sequencing*. Wiley-VCH Verlag GmbH & Co. KGaA, pp. 29-42.

Papanicolaou, A.*, et al.* (2009) Next generation transcriptomes for next generation genomes using est2assembly, *BMC bioinformatics*, **10**, 447.

Pauchet, Y.*, et al.* (2009) Pyrosequencing of the midgut transcriptome of the poplar leaf beetle Chrysomela tremulae reveals new gene families in Coleoptera, *Insect Biochemistry and Molecular Biology*, **39**, 403-413.

Pauchet, Y.*, et al.* (2010) Pyrosequencing the Manduca sexta larval midgut transcriptome: messages for digestion, detoxification and defence, *Insect Molecular Biology*, **19**, 61-75.

Paulsen, I.T.*, et al.* (2003) Role of Mobile DNA in the Evolution of Vancomycin-Resistant Enterococcus faecalis, *Science*, **299**, 2071-2074.

Pevzner, P.A. and Tang, H. (2001) Fragment assembly with double-barreled data, *Bioinformatics*, **17**, S225-S233.

Pevzner, P.A., Tang, H. and Tesler, G. (2004) De Novo Repeat Classification and Fragment Assembly, *Genome Research*, **14**, 1786-1796.

Pevzner, P.A., Tang, H. and Waterman, M.S. (2001) An Eulerian path approach to DNA fragment assembly, *Proceedings of the National Academy of Sciences*, **98**, 9748-9753.

Phillippy, A., Schatz, M. and Pop, M. (2008) Genome assembly forensics: finding the elusive mis-assembly, *Genome Biology*, **9**, 1-13.

Pop, M., Kosack, D.S. and Salzberg, S.L. (2004) Hierarchical Scaffolding With Bambus, *Genome Research*, **14**, 149-159.

Price, A., Jones, N. and Pevzner, P. (2005) De novo identification of repeat families in large genomes, *Bioinformatics (Oxford, England)*, **21 Suppl 1**, i351-i358.

Price, A.L., Jones, N.C. and Pevzner, P.A. (2005) De novo identification of repeat families in large genomes, *Bioinformatics*, **21**, i351-i358.

Puiu, D. (2004) SUPERCONTIGS: a contig scaffolding tool. *Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE*. pp. 736-737.

Quail, M.A.*, et al.* (2008) A large genome center's improvements to the Illumina sequencing system, *Nat Meth*, **5**, 1005-1010.

Quitzau, J.A.A. and Stoye, J. (2008) Detecting Repeat Families in Incompletely Sequenced Genomes. *Proceedings of the 8th international workshop on Algorithms in Bioinformatics*. Springer-Verlag, Karlsruhe, Germany, pp. 342-353.

Riadi, G., Medina-Moenne, C. and Holmes, D.S. (2012) TnpPred: A Web Service for the Robust Prediction of Prokaryotic Transposases, *Comparative and Functional Genomics*, **2012**, 5.

Richter, D.C.*, et al.* (2008) MetaSim—A Sequencing Simulator for Genomics and Metagenomics, *PLoS One*, **3**, e3373.

Roberts, A.*, et al.* (2008) Revised nomenclature for transposable genetic elements, *Plasmid*, **60**, 167-173.

Robinson, D.G., Lee, M.-C. and Marx, C.J. (2012) OASIS: an automated program for global investigation of bacterial and archaeal insertion sequences, *Nucleic Acids Research*.

Roeding, F.*, et al.* (2009) A 454 sequencing approach for large scale phylogenomic analysis of the common emperor scorpion (Pandinus imperator), *Molecular Phylogenetics and Evolution*, **53**, 826-834.

Rothberg, J.M.*, et al.* (2011) An integrated semiconductor device enabling non-optical genome sequencing, *Nature*, **475**, 348-352.

Salzberg, S.L.*, et al.* (2011) GAGE: A critical evaluation of genome assemblies and assembly algorithms, *Genome Research*.

Sanger, F., Nicklen, S. and Coulson, A.R. (1977) DNA sequencing with chain-terminating inhibitors, *Proceedings of the National Academy of Sciences*, **74**, 5463-5467.

Sarje, A. and Aluru, S. (2008) Parallel biological sequence alignments on the Cell Broadband Engine. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. pp. 1-11.

Schaack, S., Gilbert, C. and Feschotte, C. (2010) Promiscuous DNA: horizontal transfer of transposable elements and why it matters for eukaryotic evolution, *Trends in Ecology & Evolution*, **25**, 537-546.

Schatz, M.C. (2009) CloudBurst: highly sensitive read mapping with MapReduce, *Bioinformatics*, **25**, 1363-1369.

Schneider, D.*, et al.* (2000) Long-Term Experimental Evolution in Escherichia coli. IX. Characterization of Insertion Sequence-Mediated Mutations and Rearrangements, *Genetics*, **156**, 477-488.

Sebaihia, M.*, et al.* (2006) The multidrug-resistant human pathogen Clostridium difficile has a highly mobile, mosaic genome, *Nat Genet*, **38**, 779-786.

Seemann, T. (2014) Prokka: rapid prokaryotic genome annotation, *Bioinformatics*.

Simpson, J. and Durbin, R. (2012) Efficient de novo assembly of large genomes using compressed data structures, *Genome Research*, **22**, 549-556.

Simpson, J.T. and Durbin, R. (2010) Efficient construction of an assembly string graph using the FM-index, *Bioinformatics*, **26**, i367-i373.

Simpson, J.T.*, et al.* (2009) ABySS: A parallel assembler for short read sequence data, *Genome Research*, **19**, 1117-1123.

Smith, L.M.*, et al.* (1986) Fluorescence detection in automated DNA sequence analysis, *Nature*, **321**, 674-679.

Smith, T.F. and Waterman, M.S. (1981) Identification of common molecular subsequences, *Journal of molecular biology*, **147**, 195-197.

Steve Rozen, H.J.S. (1998) Primer3. Code available at http://www-genome.wi.mit.edu/genome_software/other/primer3.html. .

Tarhio, J. and Ukkonen, E. (1988) A greedy approximation algorithm for constructing shortest common superstrings, *Theor. Comput. Sci.*, **57**, 131-145.

Touchon, M. and Rocha, E.P.C. (2007) Causes of Insertion Sequences Abundance in Prokaryotic Genomes, *Molecular Biology and Evolution*, **24**, 969-981.

Treangen, T.J*., et al.* (2002) Next Generation Sequence Assembly with AMOS. In, *Current Protocols in Bioinformatics*. John Wiley & Sons, Inc.

Varani, A*., et al.* (2011) ISsaga is an ensemble of web-based methods for high throughput identification and semi-automatic annotation of insertion sequences in prokaryotic genomes, *Genome biology*, **12**, R30.

Wagner, A. (2006) Periodic extinctions of transposable elements in bacterial lineages: evidence from intragenomic variation in multiple genomes, *Molecular Biology and Evolution*, **23**, 723-733.

Wagner, A. and de la Chaux, N. (2008) Distant horizontal gene transfer is rare for multiple families of prokaryotic insertion sequences, *Mol Genet Genomics*, **280**, 397-408.

Wagner, A., Lewis, C. and Bichsel, M. (2007) A survey of bacterial insertion sequences using IScan, *Nucleic Acids Research*, **35**, 5284-5293.

Wang, L. and Jiang, T. (1994) On the complexity of multiple sequence alignment, *Journal of computational biology : a journal of computational molecular cell biology*, **1**, 337-348.

Whiteford, N*., et al.* (2005) An analysis of the feasibility of short read sequencing, *Nucleic Acids Research*, **33**, e171.

Wu, S. and Manber, U. (1992) Fast text searching: allowing errors, *Commun. ACM*, **35**, 83-91.

Yang, X., Chockalingam, S.P. and Aluru, S. (2012) A survey of error-correction methods for next-generation sequencing, *Briefings in Bioinformatics*.

Ye, J*., et al.* (2012) Primer-BLAST: A tool to design target-specific primers for polymerase chain reaction, *BMC bioinformatics*, **13**, 134.

Zagrobelny, M*., et al.* (2009) 454 pyrosequencing based transcriptome analysis of Zygaena filipendulae with focus on genes involved in biosynthesis of cyanogenic glucosides, *BMC Genomics*, **10**, 574.

Zerbino, D.R. and Birney, E. (2008) Velvet: Algorithms for de novo short read assembly using de Bruijn graphs, *Genome Research*, **18**, 821-829.

Zhang, W*., et al.* (2011) A Practical Comparison of <italic>De Novo</italic> Genome Assembly Software Tools for Next-Generation Sequencing Technologies, *PLoS One*, **6**, e17915.

Zhong, S*., et al.* (2004) Evolutionary genomics of ecological specialization, *Proceedings of the National Academy of Sciences of the United States of America*, **101**, 11719-11724.

Zhou, F., Olman, V. and Xu, Y. (2008) Insertion Sequences show diverse recent activities in Cyanobacteria and Archaea, *BMC genomics*, **9**.

APPENDIX

APPENDIX A

Table 19. Parallel assembly quality experiment

Reads[a]	Version[b]	N50 Score[c]	Longest Contig Length[d]	Number of Contigs[e] < 10Kbp	Number of Contigs[e] 10Kbp-100Kbp	Number of Contigs[e] >100Kbp	Coverage[f]	Base Calling Errors[g]	Reads Assembled[h]
Organism: E. coli HS (Length: 4643538)									
100,000	16 threads	9,771	32,894	1,229	35	0	99.7	36,258	96,754
	32 threads	9,812	32,894	1,435	37	0	99.7	36,258	96,823
	64 threads	9,812	32,894	1,331	37	0	99.7	34,875	97,618
	MIRA[i]	9,754	29,875	1,169	35	0	99.45	32,784	95,782
	16 threads	10,594	35,059	1,901	41	0	99.7	33,487	407,571
	32 threads	11,238	35,059	2,003	42	0	99.7	33,247	407,426
	64 threads	10,944	35,059	2,090	46	0	99.7	32,617	408,396
500,000	MIRA[i]	11,687	36,758	1,630	44	0	99.6	30,784	410,258
	16 threads	68,358	207,793	198	21	12	99.6	30,643	967,871
	32 threads	68,332	207,793	203	23	12	99.6	30,482	968,473
	64 threads	68,414	207,793	203	23	12	99.6	31,471	968,537
1,000,000	MIRA[i]	67,738	210,875	182	20	12	99.6	31,756	924,567
Organism:Mycobacterium vanbaalenii (Length: 6491865)									
100,000	16 threads	13,601	57,002	519	294	0	93.2	31,478	80,697
	32 threads	13,573	57,002	519	294	0	93.1	32,247	81,687
	64 threads	13,694	57,002	519	294	0	93.2	31,766	81,572
	MIRA[i]	13,892	57,470	541	192	0	92.7	28,745	83,687
	16 threads	13,694	103,880	1,234	68	1	95.1	29,683	438,745
	32 threads	13,614	103,880	1,231	71	1	95.1	29,676	441,359
	64 threads	13,632	103,880	1,228	71	1	95.15	29,875	442,978
500,000	MIRA[i]	17,486	125,784	1,120	26	2	96.34	30,875	468,751
	16 threads	26,176	178,654	4,708	72	7	95.84	34,894	109,0687
	32 threads	26,227	178,654	4,796	70	9	95.84	35,472	109,9367
	64 threads	26,229	178,654	4,777	75	9	95.84	35,217	109,8263
1,000,000	MIRA[i]	26,381	163,463	3,137	82	11	97.62	36,680	106,2354
Organism:Mycobacterium Marinum (Original Length: 6636827)									
100,000	16 threads	1,483	7,932	2,808	0	0	90.4	12,802	88,572
	32 threads	1,533	7,932	2,812	0	0	90.4	13,581	88,656
	64 threads	1,509	7,932	2,806	0	0	90.4	13,294	88,517
	MIRA[i]	1,478	7,874	3,204	0	0	90.6	12,879	91,478
	16 threads	14,642	45,350	690	35	0	89.65	1,102	464,924
	32 threads	14,755	45,350	698	38	0	89.4	1,567	451,483
	64 threads	14,153	45,350	716	38	0	87.6	2,638	426,874
500,000	MIRA[i]	13,478	47,896	600	42	0	89.7	7,845	447,851
	16 threads	22,587	87,255	568	238	0	92.81	12,301	923,248
	32 threads	22,607	87,255	565	232	0	92.8	11,854	923,248
	64 threads	22,623	87,255	569	237	0	92.8	11,933	923,248
1,000,000	MIRA[i]	18,996	62,028	695	221	0	92.6	13,748	872,568

[a]The number of simulated reads with mean length of 500bp and standard deviation of 100bp.
[b]The version of the assembler i.e. either a parallel implementation with 16/32/64 parallel threads for path finding or the serial MIRA version 3.2.1.
[c]The N50-Score of all the contigs.
[d]The length (in bp) of the longest contig.
[e]The number of contigs distributed in three intervals of (0,10,000], (10,000, 100,000], (100,000, ∞).
[f]The percentage of the original genome covered by the contigs.
[g]The number (in bp) the mismatches in the assembly.
[h]The number of reads in the assembly.
[i]The original version of MIRA 3.2.1 (Chevreux, et al., 1999).

VITA

ABHISHEK BISWAS

Department of Computer Science
Old Dominion University
Engineering & Computational Sciences Bldg,
4700 Elkhorn Ave, Suite 3300,
Norfolk, VA 23529-0162

Biographical Sketch

Abhishek Biswas received his B.E in Computer Science from Visvesvaraya Technological University, Belgaum, India in 2007. After graduation, he worked for Tata Consultancy Services Ltd. as a .NET and SQL Server developer. In January 2009 he joined as a research associate for a SAP Research Lab. funded project on Healthcare Data Mining at People's Education Society Institute of Technology (PESIT), Bangalore, India, where he worked for 6 months. In Fall 2009, Abhishek joined the Computer Science department at Old Dominion University as a MS student and later changed his level to pursue a Ph.D. degree in Fall 2010. He is currently working with Dr. Desh Ranjan and Dr. Mohammad Zubair as a research assistant, developing models and parallel solutions for genomic data analysis. He also works and regularly publishes with Dr. Jing He on modelling protein structures using medium resolution Cryo-EM images. He is currently a Ph.D. student at Old Dominion University and is expected to graduate on 12th December, 2015. His current GPA is 3.96.

Selected Publications

- A. Biswas, D. Gauthier, D. Ranjan, and M. Zubair, "ISQuest: Finding Insertion Sequences in Prokaryote Sequence Fragment Data", Bioinformatics, June 2015
- A. Biswas, D. Ranjan, M. Zubair and J. He," A Novel Computational Method for Deriving Protein Secondary Structure Topologies using Cryo-EM Density Maps and Multiple Secondary Structure Predictions", accepted at 11th International Symposium on Bioinformatics Research and Applications (ISBRA), Norfolk, Virginia, June 7-10, 2015
- A. Biswas, D. Ranjan, M. Zubair, J. He, "A dynamic programming algorithm for finding the optimal placement of a secondary structure topology in cryo-EM data", Journal of Computational Biology,, (9): 837-43, 2015.
- A. Biswas, D. Gauthier, D. Ranjan, and M. Zubair, "Big Data Challenges for Estimating Genome Assembler Quality", presented at the ACM International Workshop on Big Data in Life Sciences, Newport Beach, CA, USA, 2014.
- A. Biswas, J. Dailey, J. Ord, J. Berlin, J. Cooper, T. Holmes D. Gauthier, "Unverified Join Viewer", American Society for Microbiology, Virginia Branch Annual Meeting, Nov. 2015
- A. Biswas, D. Gauthier, D. Ranjan and M. Zubair, "Correlative Algorithm for Repeat Placement" American Society for Microbiology, Virginia Branch Annual Meeting, Nov. 2013
- A. Biswas, D. Ranjan and M. Zubair, "Genome Assembly on a Multicore System", 11th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2013: 1233-1240
- A. Biswas, D. Gauthier, D. Ranjan, and M. Zubair, "Shotgun Assembler Profiling and Benchmark" The World Congress on Engineering and Technology 2011, Beijing, China
- Biswas, A., Si, D., Al Nasr, K., Ranjan, D., Zubair, M., He, J. "Improved efficiency in cryo-EM secondary structure topology determination from inaccurate data" J. Bioinform Comput Biol, 10(3): 1242006, 2012.
- Biswas, A., Si, D., Al Nasr, K., Ranjan, D., Zubair, M., He, J. "A constraint dynamic graph approach to identify the secondary structure topology from cryoEM density data in presence of errors" The Proceeding of the IEEE International Conference of Bioinformatics and Biomedicine, p160-3, 2011, Nov 12-15, 2011.
- A. Biswas, B. V. Sagar, J. Srinivasan,"Managing and Correlating Historical Events Using an Event Timeline Datatype" DASFAA 2008: 604-612