To: Dr. Saundra F. DeLauder, Dean, School of Graduate Studies and Research

The members of the Committee approved the Thesis of Kenneth G. Shim

Candidate's Name

as presented on 11/12/15 .

Date

We recommend that it be accepted in partial fulfillment of the requirements for the degree

Master of Science in Computer Science

Degree Name Major/Program Name

Department CIS Date 11/12/2015

Advisor

Department CIS Date 11/12/2015

Member

Department CIS Date 11/12/2015

Member

Affiliation AGNR Date 11/12/2015

External Member

Approved

Department CIS Date 12/2/15

Department Chairperson or Designee

College CMNST Date 12/7/15

Academic Dean or Designee

Saundra F. DeLauder Date 12/4/15

Dean, School of Graduate Studies and Research

APPROACHING THE SEQUENCE ASSEMBLY PROBLEM WITH HYBRIDIZATION OF ANT COLONY OPTIMIZATION AND MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

By

KENNETH SHIM

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in the
Computer Science Graduate Program
of Delaware State University

DOVER, DELAWARE
December 2015

ACKNOWLEDGEMENTS

I would like to express my gratitude to all professors in the Department of Computer and Information Sciences at Delaware State University. It has been a great privilege to learn and grow under the supervision of highly respected computer scientists. In addition to all the knowledge and skills that I have accumulated, they have also provided an environment to help me become a better thinker to make better decisions in my life. I would especially like to thank my advisor, Dr. Tomasz Smolinski, for providing me the opportunities to work in the computational neuroscience and bioinformatics research field. Throughout my undergraduate and graduate studies, he has shown an exceptional willingness and patience to support my academic progress and also kindness to address my personal concerns.

ABSTRACT

Sequence assembly is an essential procedure taken in deciphering the underlying genetic information of a biological sample. A long strand of nucleotides embedded in a typical genetic sample necessitates random fragmentation of the original genetic information for sequencing. Hence, the challenge of finding the true genetic sequence from a set of randomly scattered sequence fragments still prevails despite the advancement of the modern high-throughput, paired-end Next Generation Sequencing (NGS) technologies. The De Bruijn Graph (DBG) sequence assembly has been the method of choice for assembly of NGS short sequences, mainly for its computational advantage over the Overlap Layout Consensus (OLC) sequence assembly. Unlike the DBG-based sequence assembly, the OLC sequence assembly method allows sequence assembly with flexible sequence alignments without involving any breaking of sequenced fragments of a genetic information further into smaller sequence fragments; thus, a higher genetic information integrity is retained during its sequence assembly. In addition, the OLC-based sequence assembly has demonstrated outperformance in assembly of long sequence fragments and has also shown higher tolerance to sequence error and low sequencing depth. Thus, OLC assembly is expected to be more practical with the advancement of sequencing technology.

Considering the benefits of OLC assembly, a new sequence assembly method was designed using a hybridization of Ant Colony Optimization (ACO) and Multi-Objective Evolutionary Algorithm (MOEA). The design of the proposed method mainly focused on carrying out its heuristic sequence assembly while ensuring high quality and accuracy of the contigs being assembled. The ACO component provides local improvement to a sequence assembly with construction of contigs based on quasi-optimal overlapping alignments between reads. Meanwhile, the MOEA component takes the role of further assembling the ACO-generated contigs based on multiobjective criteria to yield final quasi-optimal Pareto-front sets of assembled sequences; the incorporated multiobjective criteria are maximization of contig length and contig overlap, and minimization of contig gaps.

Based on various experiments, the most contributing and novel property of the proposed method was found to be having the advantage of producing diverse multiple sets of assembled sequences. Especially, a high leverage in the quality of the sequence assembly could be accomplished through the construction of an elite set of assembled sequence built by selection of high quality contiguous sequences found in each set of assembled sequences. The quality, accuracy, and usefulness of the elite set of assembled sequences were found to be competitive to the most widely used DBG-based assemblers, Velvet and Trinity; furthermore, a statistical analysis also suggests a significant improvement relative to a modern OLC-based assembler. A substantial enhancement in the current version of the proposed method could be attained through overcoming the

main processing bottleneck found in the construction of the adjacency matrix of maximal sequence-to-sequence alignments.

TABLE OF CONTENTS

LIST OF TABLES

Chapter 5.

LIST OF FIGURES

Chapter 1.

Chapter 2.

Chapter 3.

Chapter 4.

Chapter 5.

LIST OF PSEUDOCODES

Chapter 4.

CHAPTER 1
GENERAL INTRODUCTION

1.1 A Brief Overview of Bioinformatics

Bioinformatics is an interdisciplinary field that has emerged from the amalgamation of biology and computer science. A more informative description of bioinformatics can be given by looking at its main tasks [2]. First, providing publicly available web portals linked to databases of biological data that are organized according to their types and functions are at the core of bioinformatics. This allows researchers to conveniently access those databases to acquire the data relevant to their studies, but also to contribute their new findings. Especially, cross-database search system such as Entrez [3], InterPro [4], and UniProt [5], have been invaluable tools for cross-reference studies, providing their users with a pool of related data from a single user query input through bridging information existing across multiple databases. Second, bioinformatics is about developing and applying tools to analyze the collected biological data in order to address the scientific questions posed by the researchers. Such bioinformatics tools are developed by incorporating techniques from diverse computing disciplines such as data mining, database design, machine learning, software engineering, statistics, and so on, in order to aid in the exploration of a given biological dataset and acquiring of biological knowledge discovery. Some of the widely used bioinformatics tools include FastQC [6], BLAST [7], Velvet [8], Trinity [9], and BLAST2GO [10].

The surge of biological data is the main reason to why computer science has become the main engine in today's biological research studies. The rapid growth of sequencing technology has greatly reduced the time required for deciphering the strands of the nucleotide sequences of a biological sample. In turn, advancement of computing technology, data management, and algorithmic techniques has allowed for improved research workflow efficiency to handle the dramatic increase in biological data with the aid of efficient computing tools. However, the rate of sequencing throughput has been increasing at such a high pace that the expected boost of computer performance by the Moore's law could not catch up to the amount of biological data being produced for timely analysis. Hence, demand for application of parallel cloud computing has been rising in bioinformatics community [1].

1.2 The Central Dogma of Molecular Biology

Research studies in bioinformatics revolve around the central dogma of molecular biology, which describes the central "mechanics" that govern all cellular function. Consequently, it is the core knowledge that computer scientists need to be accustomed to in order to successfully cooperate with biologists and develop tools capable of performing biologically meaningful analysis. Since the discovery of DNA structure by James Watson and Francis Crick [11], studies of the central dogma have evolved with respect to specificity and generality with the aid of computing sciences. However, the simplified statement of '*DNA makes RNA makes protein*' can be used to summarize the foundation of the central dogma of molecular biology as shown in Figure 1-1.

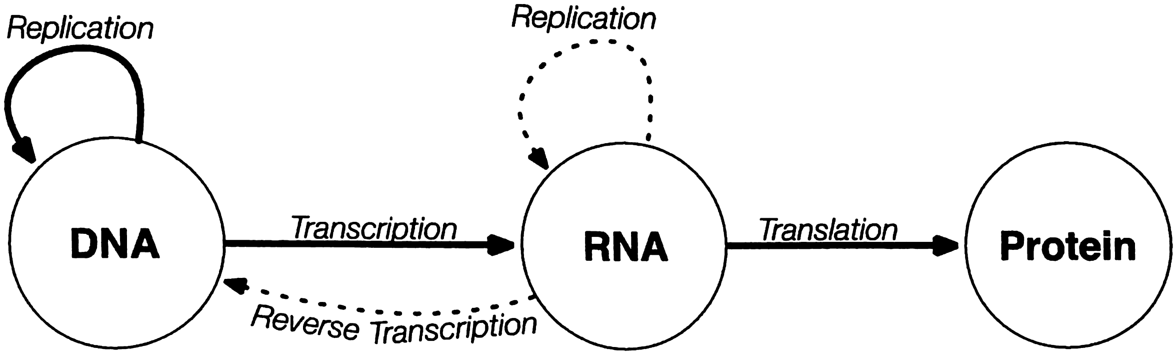


Figure 1-1: The Central Dogma of Molecular Biology

Deoxyribonucleic acid, or simply DNA, is composed of four distinct nucleobases: adenine (*A*), cytosine (*C*), guanine (*G*), and thymine (*T*). On the other hand, ribonucleic acid or RNA is composed of the same nucleobases as DNA, except it contains uracil (*U*) instead of thymine (*T*). Similarly to the binary digits of the machine language, subsequences of a DNA string, varying in length and nucleobase permutations, are used as the genetic instructions for making different types of proteins that account for all cellular functioning. The structure of the DNA is largely characterized by twisted chains of nucleotide molecules constituted by coupling of a nucleobase molecule, a phosphate molecule, and a sugar molecule. DNA keeps a copy of itself by having two strands of nucleotides interconnected by the hydrogen bonds between two complementary nucleotide bases. Adenine (*A*) always binds to thymine (*T*) and cytosine (*C*) always binds to guanine (*G*). The primary purpose of the double-stranded nature of DNA is to keep the integrity of the genetic information by being able to recognize the error introduced by a mutation with the proofreading of the other strand [7].

A nucleotide molecule has directionality based on the carbon positions of its sugar-ring molecule. As shown in Figure 1-2, the fifth carbon with a phosphate group is known as 5’ prime end or upstream while the third carbon with a hydroxyl group is known as 3’ prime end or downstream.

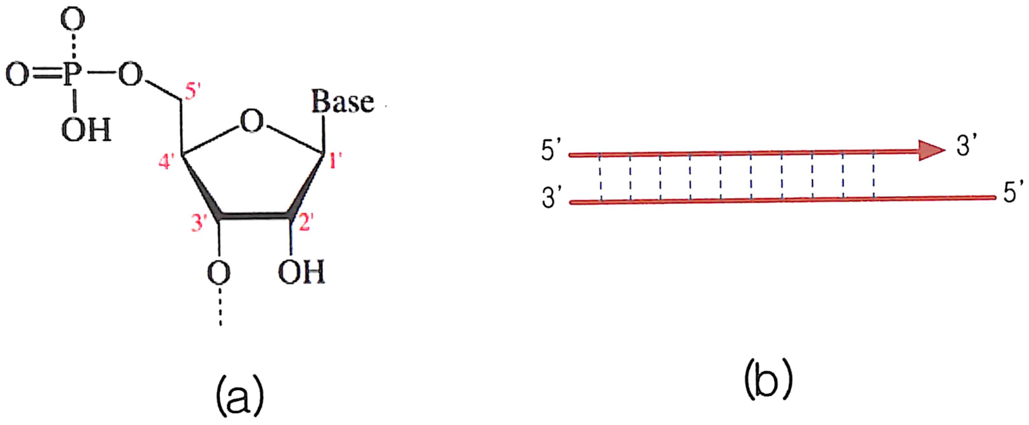


Figure 1-2: The directionality of a nucleic acid (a) 5’ and 3’ prime ends of a sugar-ring molecule (b) 5’->3’ replication

During DNA replication, a new chain of DNA string is always synthesized in the direction of 5’ to 3’, starting on the 3’ end of a template DNA copying towards the 5’ end. A chain of nucleotides extends as the 5’ end of a new nucleotide binds to its 3’ end. Thus, 5’ end of a strand complements the 3’ end of the other strand and vice versa.

A gene sequence undergoes three major molecular transformations before turning into a protein. First, gene expression regulating protein called transcription factor binds to a DNA promoting region followed by binding of RNA polymerase to initiate the transcription of a target gene. The RNA polymerase enzyme plays the role of unwinding the DNA and copying the strand of bases from 5' end towards 3' end. The synthesized string of transcribed gene sequence is known as pre-mRNA. Pre-mRNA is composed of non-coding regions called introns and the coding regions called exons. Thus, introns are removed and the exons are ligated together, in the process called splicing, before the forming a protein can take place. This task is accomplished by complex macromolecule machinery known as spliceosome. Here, a particular transcribed pre-mRNA sequence can undergo alternative splicing by choosing different sets of exons to produce different forms proteins isoforms. The splicing of pre-mRNA results in a string of exons known as messenger RNA or mRNA. Upon completion of splicing, the transcribed genetic information embedded in mRNA migrates from nucleus to the cytoplasm of a cell. A strand of mRNA is then translated into a protein with the aid of special protein molecules, ribosome and transfer RNA or tRNA. Each set of three nucleotide bases along the strand of mRNA forms a symbol of genetic code known as codon. During translation, the codons are sequentially translated into their corresponding elemental protein units known as amino acids. The codon chart in Figure 1-3 shows complete set of codons and their corresponding translated amino acids.

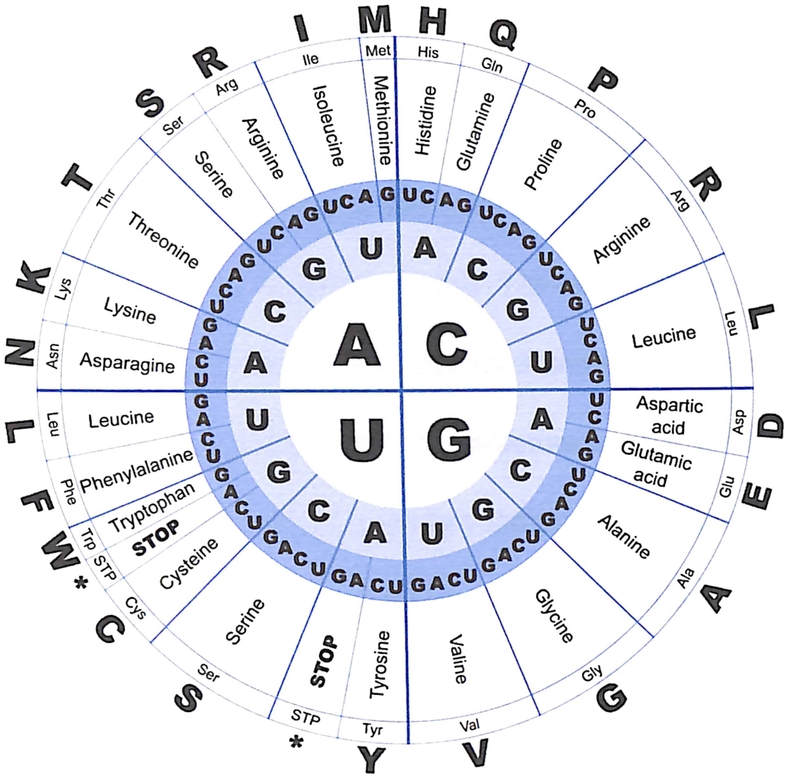


Figure 1-3: The codon chart [45]

Notice that the genetic code is redundant as shown in the codon chart. In other words, a single amino acid can be mapped to multiple codons. Consequently, although there are 64 possible codons, there exist 21 amino acids in total instead of 64 amino acids. The translation of mRNA begins with binding of ribosome (primarily made of ribosomal RNA or rRNA) at the site of start codon, 'methionine' (*AUG*), and terminates at a stop codon which can be, 'amber' (*UAG*), 'ochre' (*UAA*), or 'opal' (*UGA*). Transfer RNA or tRNA carries out the translating process of codons. tRNA molecule is made of anti-codon on one end and its corresponding amino acid on the other end like key-value pair. The anti-codon part of tRNA binds to its complement codon on mRNA resulting in an arrangement of amino acids. The strand of amino acids is glued together by peptide bond with the aid of ribosome to form a chain of amino acids known as polypeptides or simply protein.

Function of a protein is primarily determined by its structure [54]. A protein is shaped through a series of folding of polypeptides. The initial linear strand of amino acids known as primary structure transforms into another structure known as secondary structure as it folds onto itself to form a coil-like structure known as alpha-helix or a zig-zag structure known as beta-sheet. The secondary structure is further transformed into tertiary structure through a higher order of folding. Most proteins exist in a tertiary structure to serve their functional purpose; however, some proteins exist in a quaternary structure formed by joining of multiple polypeptides. Transformed proteins, in turn, are used to serve as cellular structural elements and provide cellular functions such as intercellular communication medium and catalyzing enzymes.

A complete set of all DNA sequences of an organism is known as genome. For example, a human genome consists of DNA sequences stored in 23 pairs of chromosomes with the base length of approximately 3 billion nucleotides. Interestingly, despite distinct distinguishable character traits, known as phenotypes, among humans such as disease vulnerability, response variation to drugs, and physical appearance, any two individuals share 99.9% of their genetic information. Variation among individuals originates from small genetic differences or alternative forms of a given gene known as alleles existing at a specific locus of a genome. An allele with a single nucleotide base difference is known as a single nucleotide polymorphism or simply SNP. Similar to the definition of genome, a complete set of expressed transcript sequences present in all mRNA molecules is known as transcriptome.

While genome content stays quite static, gene expression can dynamically vary depending on the environmental condition of a cell at a given time. Thus, growing collection of annotated genomes through high-throughput sequencing technologies and advanced bioinformatics tools can provide a deeper understanding of biological diversities among species through studies such as multiple genome alignment and phylogenetic comparative analysis. In addition, the ongoing international effort on predictive diagnostic and personalized medicine through genetic testing can allow understanding of a person's illness on genetic level instead of symptoms, and provide better preventive treatment. Overall, bioinformatics will play a critical role in the advancement of future medicine with the aid of computing science.

1.3 Sequencing Technology

The early revolutionary development of sequencing methods by Walter Gilbert and Frederick Sanger [55] opened the door to the possibility of studying and understanding the fundamental workings of biological systems from genetic level. Sequencing simply denotes the process of decoding the underlying nucleotide bases of a biological sample in the order of their arrangement. Hence, sequencing technology is an indispensable tool for generating the biological data needed for any genetic studies.

Polymerase chain reaction or simply known as PCR [56] plays an important role in sequencing technology. PCR is a widely used method to amplify a copy of a genetic sample by simulating the DNA replication process in an artificially designed environment. The basis of PCR involves repeated process of denaturing of double-stranded DNA samples and annealing of primers. In nature, DNA replication is carried out by an enzyme known as DNA polymerase that sequentially adds new nucleotide bases complementary to a sequence of bases present in the target DNA template. The replication process is initiated at the site of a short sequence known as primer sequence (~10 bp) on the target sequence being replicated. Starting at the site of the primer, the polymerase builds a new chain of nucleotide bases on the target DNA template by binding the 5' end of a new nucleotide to the 3' end of a sequence strand being extended. In PCR, the environment for DNA replication is prepared with mixture of a double-stranded DNA sample being copied, primers, raw nucleotides or dNTP, and DNA polymerases in a buffer solution. Then, DNA synthesis is carried out by systematically adjusting the temperature of the solution to denature the double-stranded sequences into a pair of single stranded DNA templates and to anneal the primer sequences onto the DNA templates for the polymerase to build a new chain of sequences. The process of denaturing and annealing is repeated to generate exponential number of copies of the original sample sequence in each cycle.

1.3.1 Sanger Sequencing

In Sanger sequencing [14], amplified DNA templates, DNA polymerases, primers, and raw dNTPs are also used as in PCR. However, unlike PCR, defected radiolabeled nucleotides, ddNTPs, are used to terminate the polymerase from extending the chain of nucleotides complement to the target template. The ddNTP lacks the hydroxyl group on its 3' end, thus preventing other nucleotides from binding to the chain. This is the reason why Sanger sequencing is also well known as chain-terminating sequencing method. By using the chain-terminating nucleotides, multiple copies of sequences with different length, called Sanger fragments, can be obtained. Then, all fragments are aligned based on their molecular size using capillary electrophoresis (CE) and x-ray image of those sequences with radiolabeled ends is obtained to decode the position of each nucleotide of the DNA sample. Four separate mixture reactions had to be used initially to carry out the process independently for each kind of radiolabeled ddNTPs in order to distinguish each type of the nucleotides on the x-ray image. Unfortunately, this pioneering technology was tediously time consuming, expensive, and unsuitable for sequencing long DNA fragments. Chain termination by ddNTPs typically occurs before reaching a long end of a target template. Thus, Sanger sequencing was only suitable for sequence length of less than 1000 nucleotides long.

1.3.2 Shotgun Sequencing

Frederick Sanger further developed a method known as Shotgun sequencing [16] to overcome the problem of sequencing a long DNA sequence. With Shotgun sequencing method, whole genome sequencing became possible for the first time. Early genome sequencing of many different species, including the approximately 3 billion nucleotides long human genome from the human genome project (HGP) [17], was performed with the Shotgun sequencing method. Shotgun sequencing is mainly characterized by shearing of a large DNA strand at random positions followed by sequencing the resulting DNA fragments using the Sanger's chain termination method. Instead of using the radiolabeled ddNTPs during the chain termination, however, use of fluorescently color dyed ddNTPs was largely incorporated during this time. This allowed identifying the sequence of nucleotides in a single capillary instead of four. Also, multiple lanes of capillaries, known as capillary array, were used to sequence the Sanger fragments simultaneously in parallel; in addition, automated shotgun sequencing systems became available with the technology advancement. However, in order to identify the sequence of nucleotides of each fragment using the chain terminating method, multiple copies of each fragment were required in an efficient manner compared to PCR. For this task, all fragments were digested into self-replicating plasmids known as vectors. Then, the vectors were placed into highly active bacteria such as E. coli for fertilization on a culture. The results were million copies of each fragment ready for the chain termination reactions. Despite the improvement from the Sanger sequencing, the Shotgun sequencing still lacked processing time efficiency and cost effectiveness. The main disadvantage of the Shotgun sequencing is the loss of genetic information originating from processes involved in the library preparation. Since only the sheared fragments with a proper base length can be used to bind to the plasmid, a large amount of loss of genetic information can result from filtering of fragments outside the acceptable base length. In addition, not all cloning vectors succeed in entering the bacteria, which accounts for another source of loss of the original genetic information. Thus, the resulting fragments from the amplification process do not cover the entire sequence of the original sample. The time consuming processes and the necessity for resequencing the sample to close in the gap of loss of coverage largely accounts for the considerable amount of time spent for decade-long human genome project (1990 – 2003).

1.3.3 Next Generation Sequencing

Modern sequencing technologies, collectively known as Next Generation Sequencing (NGS) [10], incorporate massively parallel techniques that allow high throughput sequencing at a high coverage level. A human genome can now be sequenced in a matter of a day with as little as 100 dollars. Compared to the duration of the human genome project that cost almost $3 billion dollars, it shows how sequencing technology has rapidly advanced. Illumina sequencing technology is one of the most widely used NGS technologies today [15]. NGS sequencing technologies are largely characterized by application of techniques known as bridge amplification and sequencing by synthesis.

As in Shotgun sequencing, fragments from random shearing of a DNA sample are first obtained. However, instead of cloning the fragments into vectors and bacteria, amplification of the fragments is accomplished through ligation of adapters on both ends of each fragments followed by cycles of replication of adapter-ligated fragments through bridge amplification. First, the adapter-ligated fragments are placed onto the lanes of glasses called flow cell. The flow cell is coated with a lawn of two types of primers that are complementary to the either side of the adapters on the fragments. Then, fragments bind to the primers complementary to one end of their adapters on the flow cell, followed by the other adapter ends hybridizing with the other primer type on the flow cell resulting in a bridge formation. Subsequently, similar to PCR, polymerase builds a chain of nucleotides complement to the bridged fragments. The two stranded fragments are then denatured into two single stranded fragments. A series cycle of bridging, generating new complementary fragments, and denaturing is repeated to generate amplified clusters of the original fragments.

The amplified fragments are then sequenced by a method known as sequencing by synthesis. Unlike the chain termination method, all raw nucleotides are fluorescently labeled and have the property to terminate the polymerase chain reaction to prevent other raw nucleotides from binding. However, this terminating property can be chemically cleaved off to transform them into normal extendable dNTPs. Using this property, all nucleotides of all fragments are identified one at a time simultaneously as they bind to the fragment templates and are being excited and emit light, which is then detected and recorded. In addition, the accuracy of each identified base is also measured and recorded based on the wavelength and the intensity of the light it emit.

Most modern sequencing adapts pair-end sequencing, which identifies bases on the 5' ends of both strands of each fragment instead of single ends. Sequenced fragments are generally called reads. Reads generated from the pair-end sequencing are called pair-end reads or forward/reverse reads as shown in *Figure 4*. By paring the both ends of fragments, it helps to recover the original orientation of sequenced reads during sequence assembly.

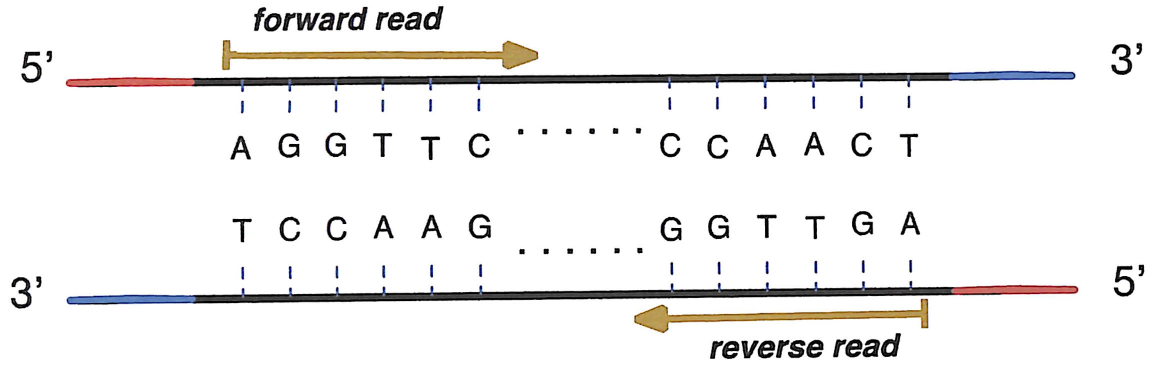

forward read
5'
3'
A G G T T C ...... C C A A C T
T C C A A G ...... G G T T G A
3'
5'
reverse read

Figure 1-4: The pair-end sequencing of Next Generation Sequencing

Despite the advantage of low cost, high-coverage, high-throughput sequencing technology, the reads generated by NGS are still considerably short. Typically, the length of these reads is less than 200 base pairs. Thus, there exists a great reliance on computing science to restore the fragmented reads back to their original long sequence during the sequence assembly.

CHAPTER 2

SEQUENCE ASSEMBLY PROBLEM

High throughput and massively parallel NGS technology allowed for a remarkable enhancement of sequencing rates while plunging the cost per sequenced reads. Yet, as a consequence of the random fragmentation of the genetic information taking place during NGS, the challenge of assembling the pieces of short sequence reads is still a necessary sequel procedure to finalize the process of decoding the genetic sequence of a biological sample. Fragmented short sequences are either assembled with or without the aid of a reference genome. In reference-based assembly, a known genome is used as a guide during assembly to assemble the generated reads by mapping them to their variant genome. Alternatively, in *de novo* assembly, reads are assembled from scratch in the absence of a reference genome. Due to the existing variability between genotypes of even the same species, referenced-based assembly, in a counterintuitive manner, does not guarantee a better assembly of reads. Instead, *de novo* assembly, despite its higher computational cost, has shown to perform better in the highly variant regions of a sample [21]. Thus, hybrid approaches that take the advantage of both assembly paradigms have been recently emerging [21].

2.1 Overlap Layout Consensus Assembly

Assembling sequenced reads is mainly facilitated with searching and matching the overlapping base patterns between fragmented strings of bases. The strength of overlap between two reads indicates how well the fragments fit together and can be computed by finding the maximum count of matching bases between the two aligned reads.

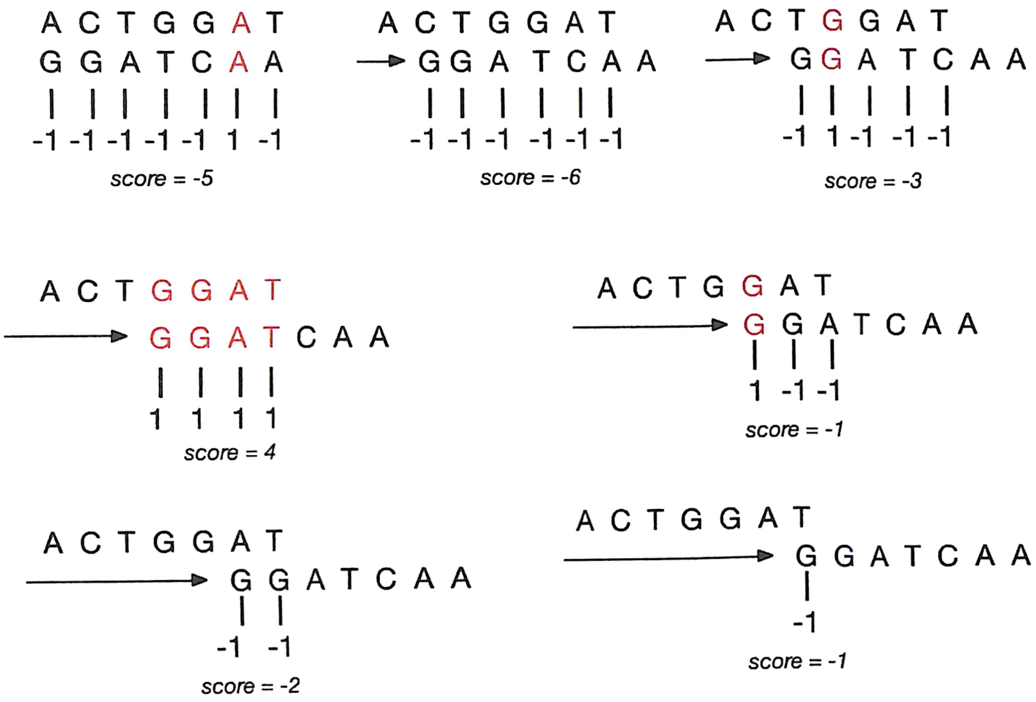


Figure 2-1: The computation of overlap score between two reads

As shown in Figure 2-1, two reads are aligned to each other starting from their first bases. Then, the overlapping scores are computed by summing the scores for matched and unmatched bases in the aligned bases. Out of the aligned bases, each set of matched bases contributes a score of one (+1) whereas the unmatched bases contribute score of negative one (-1). Here, the score can be adjusted to place different level of emphasis on each matching bases if needed. For instance, a greater negative value can be used for the unmatched bases to give more penalizing score for the unmatched bases. The maximum overlap score between two reads is found by iteratively computing each overlap scores by sliding one of the read against the other one base at a time. This way, an entire set of reads can be aligned against each other with the maximum overlapping score to output a contiguous read based on the most frequently occurring bases in each positions as shown in Figure 2-2. This method of assembling reads is known as the *overlap layout consensus* assembly.

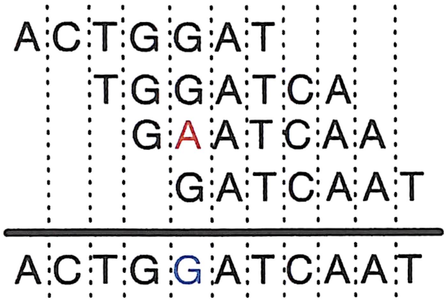


Figure 2-2: The formation of a consensus sequence (contig) from aligned reads with maximum overlap score

A contiguous read generated from assembling a set of reads is called a *contig*. The total overlap score in a contig indicates how likely the set of reads making up a contig originates from the same segment of the original sequence string.

The main challenge in the reconstruction of fragmented genetic information arises from exploring and finding the optimal arrangement and alignment of the reads that outputs the maximum overlap score. In addition, a small amount of sequencing error and loss of base coverage present in the generated reads can further complicate the challenge of the sequence assembly problem. The *coverage value* measures the number of reads covering each nucleotide bases in a genome on average. With a given number of generated reads (N), the average read length (L), and its genome length (G), the coverage value (C) is obtained with the following computation.

$$C = \frac{N \cdot L}{G}$$

Thus, a higher coverage value indicates increased likelihood of having overlapping regions between reads that span the whole genome. On the other hand, low coverage can lead to gaps between two assembled contigs with no overlapping bases. Particularly, assembling the improperly ordered reads is one of the main causes of the gaps between contigs. Importantly, a higher coverage also means that there is a higher chance of overcoming the erroneous bases present in a read with other properly sequenced reads in the same region during the overlap layout consensus assembly. However, higher coverage is directly related to higher processing time and cost. Therefore, optimal coverage amount relative to the genome length should be considered to avoid producing unnecessarily high redundant overlapping regions.

Even in the case of sequence assembly with ideally generated reads with no error and complete coverage, pervasively present repeating base-patterns in species can still aggravate the challenge of the sequence assembly problem. As a glaring example of this phenomenon, a recent study claims that over two third of human genome is believed to be repeats [19]. Sequence assembly can easily involve millions of reads; However, Figure 2-3, assuming an over-simplified case of sequence assembly with only seven reads with no error and no gaps in coverage, demonstrates how repeated reads can easily lead to the wrong ordering of reads and result in poorly overlapped consensus sequence.

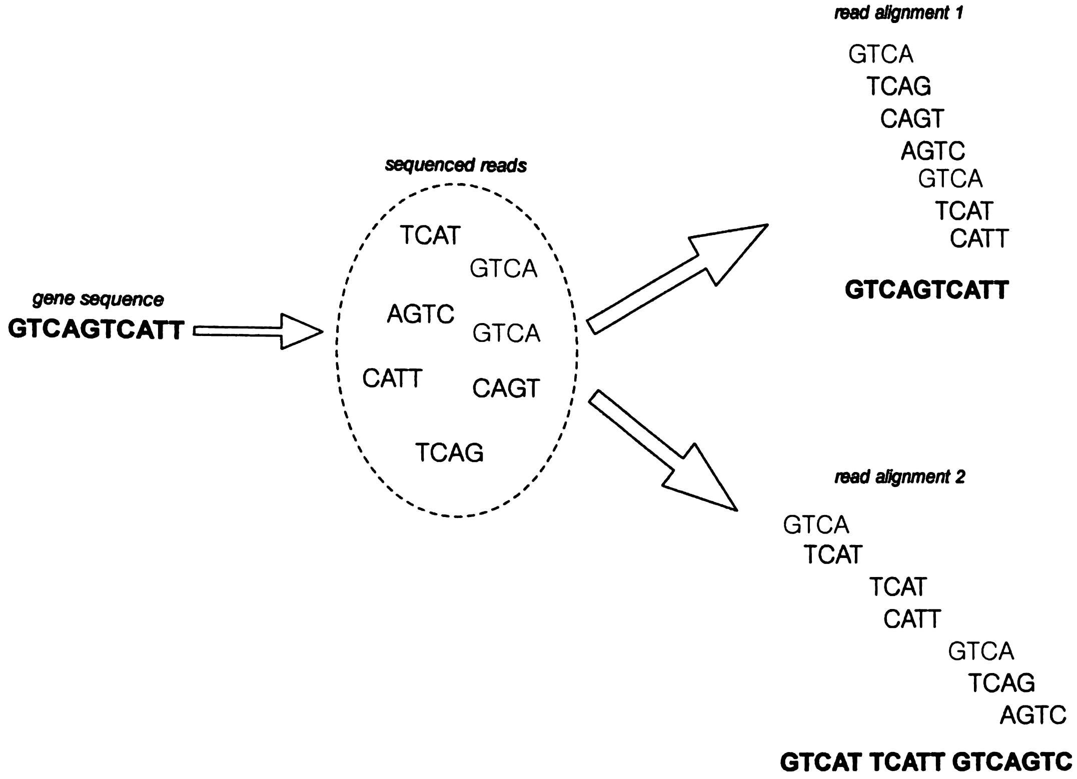


Figure 2-3: An example of alternative read alignments caused by a repeating base pattern

Here, the read '*GTCA*' can be matched with either '*TCAG*' or '*TCAT*' with an equal overlap score. However, the choice between the two alternative reads can result in dramatically different outcomes. As shown in the '*read alignment 2*' of Figure 2-3, improper alignment results in a poor average contig overlap score and average contig length. In addition, the number of resulting contigs and number of gaps between contigs has increased.

The overlap strength between reads can be expressed as a directed graph where its vertices are the reads and directed edges connect between overlapping reads with their respective overlap score. For example, *Figure 8* below shows an alternative representation of the read alignment from Figure 2-3 using a weighted directed graph.

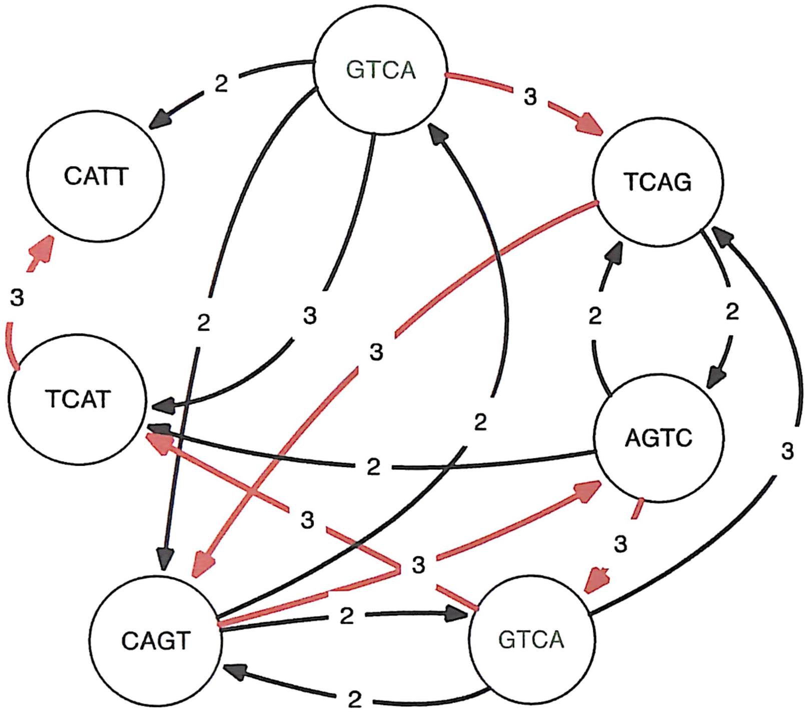


Figure 2-4: An example of representing read overlaps using a directed graph

The integers on the edges of the directed graph indicate the number of overlapping bases between reads. The directed edges colored in red show the path that leads to the optimum assembly of reads. The overlap strength between reads from Figure 2-4 can also be presented in adjacency matrix as shown Figure 2-5.

	GTCA$_1$	TCAG	AGTC	GTCA$_2$	CAGT	TCAT	CATT
GTCA$_1$	0	3	1	0	2	3	2
TCAG	1	0	2	1	3	0	0
AGTC	3	2	0	3	1	2	1
GTCA$_2$	0	3	1	0	2	3	2
CAGT	2	1	3	2	0	1	0
TCAT	0	1	0	0	0	0	3
CATT	0	3	1	0	2	3	2

Figure 2-5: An example of representing read overlaps using an adjacency matrix

Finding the optimum contig for a given set of reads can also be viewed as solving the shortest common superstring problem (SCSP) where aligned reads with maximum total overlap result in the shortest contig and the most reliable assembly. Also, if the overlap scores of the directed graph in Figure 2-4 were negated, then the problem of assembling the reads becomes strikingly similar to the traveling salesperson problem (TSP). Just as the shortest Hamiltonian cycle that takes a salesperson to each city exactly once and back to the starting city is sought in the TSP, the sequence assembly problem similarly explores the constructed directed graph to find the Hamiltonian path that visits each read exactly once with the minimum total overlap score. The complexity of the sequence assembly problem has been shown to be NP-hard [20]. Due to the similar computational complexity of the SCSP and the TSP, the optimization methods of computational intelligence have been widely used for approaching the sequence assembly problem. In general, the overlap layout consensus assembly can be described as finding the permutation of reads that maximizes the overall overlapping regions between the reads while minimizing the number of gaps between the contiguous regions to produce the contigs with optimum length that represents the original sequence string as much as possible.

The generated contigs can be further processed using the preserved information in the paired-end sequence data to arrange the contigs in the proper order, adjust their orientation respect to their original chromosomes, and to minimize the identified contig gaps. This procedure is known as *scaffolding* where a *scaffold* consists of linked contigs in their proper position and orientation as shown in Figure 2-6.

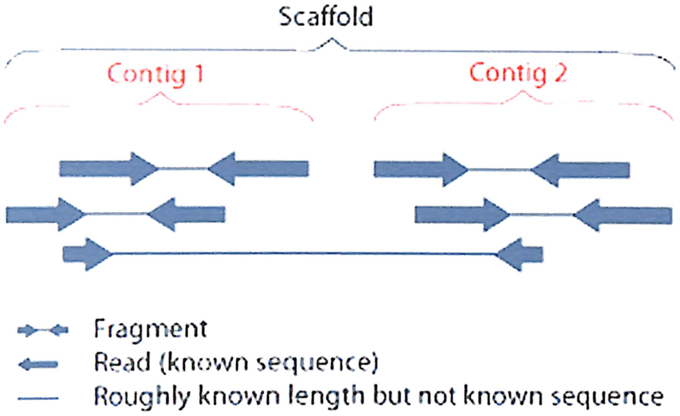


Figure 2-6: The composition of a scaffold [22]

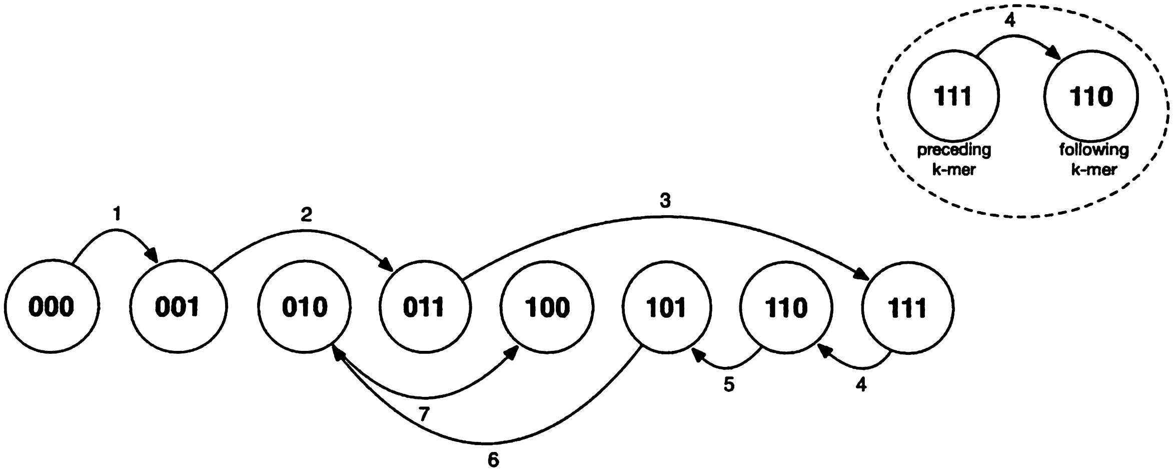
When only one end of the fragments is sequenced, the orientation of each of the sequenced reads is lost since they could be originating either from their forward strand or the reverse strand. However, this problem is alleviated by the pairing of both sequenced ends of forward and reverse strand for each fragment as described in the previous section.

In addition, the measured gap length between the reverse read and the forward read of a sequenced fragment provides a spatial relationship between contigs and assists in identifying specific regions of a chromosome that lacks coverage.

2.2 De Bruijn Graph Assembly

The most widely used modern sequence assembly method is *De Bruijn* graph assembly. *ABySS* [23], *Trinity* [24] and *Velvet* [25] are some of the most popular de novo assembly software tools based on the De Bruijn graph assembly method. The main accomplishment of De Bruijn graph assembly was overcoming the challenge related to the high computational cost in the overlap layout consensus assembly. The De Bruijn graph assembly is based on *Nicolaas De Bruijn*'s solution to the superstring problem that adapts the *Leonhard Euler*'s approach to solving the *Konisburg* Bridge problem [26]. Euler proved that a walk that visits all seven Konisburg bridges exactly once does not exist by pointing out the imbalanced number of bridges connecting to each city. In other words, Eulerian cycle that visits all edges exactly once exists if and only if the graph is strongly connected and balanced. A graph containing a Eulerian cycle is called *Eulerian graph*.

Consider a case of solving a superstring problem involving binary digits, 0s and 1s. For a set of all possible binary, so-called k-mers, the superstring problem finds the shortest binary string composed of exactly one of each binary k-mers. The term '*k-mer*' is a widely used term in bioinformatics to refer to a sequence string composed of k number of nucleotide bases. Thus, 'ACCT' is an example of a 4-mer and 'CGGGAT' is an example of a 6-mer. This way, a binary k-mer, '0110' is an example of a 4-mer. There are N^k possible k-mers where N denotes the number of possible elements for each positions of a k-mer. Using all unique binary 3-mers, a set of 2^3 binary motifs $(\mathbf{000, 001, 011, 111, 110, 101, 010, 100})$, a directed graph can be constructed as shown in Figure 2-7.



superstring for binary 3-mers: **0001110100**

Figure 2-7: A construction of a superstring for binary 3-mers

Here, nodes of the graph represent the binary k-mers whereas directed edges represent shared (k-1)-mers between two nodes. A directed edge is formed between two nodes if the suffix (k-1)-mer of preceding k-mer matches with the prefix (k-1)-mer of following k-mer. Then, the superstring is found by exploring a Hamiltonian path that visits each node exactly once. As discussed briefly in the previous section, the computational complexity of finding a Hamiltonian path is NP-complete. In other words, there are no known algorithms that can solve this problem in polynomial time.

Nicolaas De Bruijn reduced the computational complexity of solving the binary superstring problem to tractable polynomial time with an alternative method inspired by the Konigsberg problem. Instead of denoting each k-mer as a node of the directed graph, each of the k-mers is used to denote an edge of the directed graph. As shown in Figure 2-8, each k-mer is decomposed into two (k-1)-mers; preceding node takes the prefix (k-1)-mers and the following node takes the suffix (k-1)-mers.

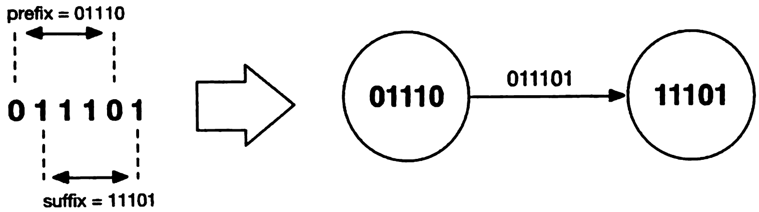


Figure 2-8: Decomposition of a k-mer into (k-1)-mers

In turn, the two decomposed (k-1)-mers are connected by their k-mer edge from its prefix node to its suffix node. Then, a directed graph is constructed by overlapping the matching nodes between (k-1)-mer suffix of a k-mer and a (k-1)-mer prefix of another k-mer as shown in Figure 2-9. A directed graph built in such a way is known as *De Bruijn graph.*

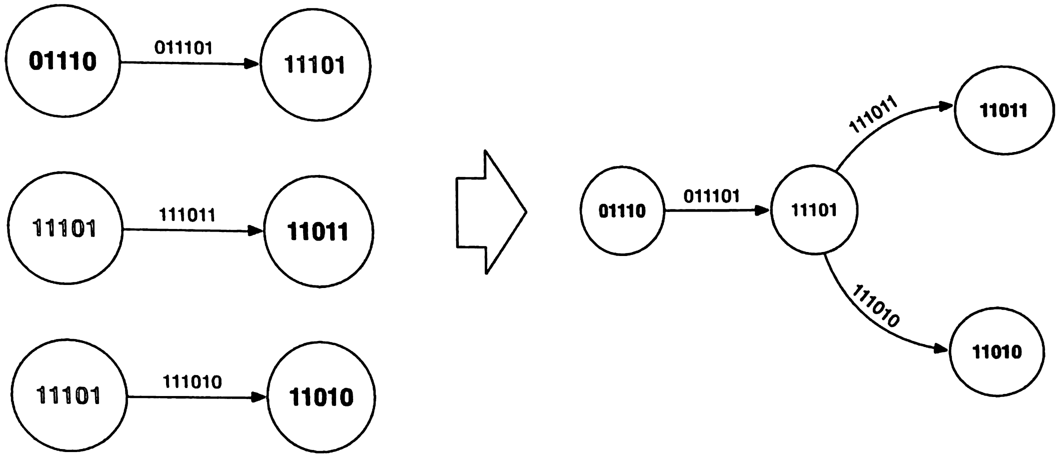


Figure 2-9: An example of constructing a De Bruijn graph with decomposed k-mers

This way, an alternative directed graph is constructed for the set of the binary 3-mers as following as shown in Figure 2-10. The depicted De Bruijn graph is strongly connected since any node can be reached from any node. Also, it is balanced since every node has an even number of incoming and outgoing edges. Therefore, the depicted De Bruijn graph is *Eulerian* and a superstring can be constructed by taking a *Eulerian walk* that visits all edges exactly once.

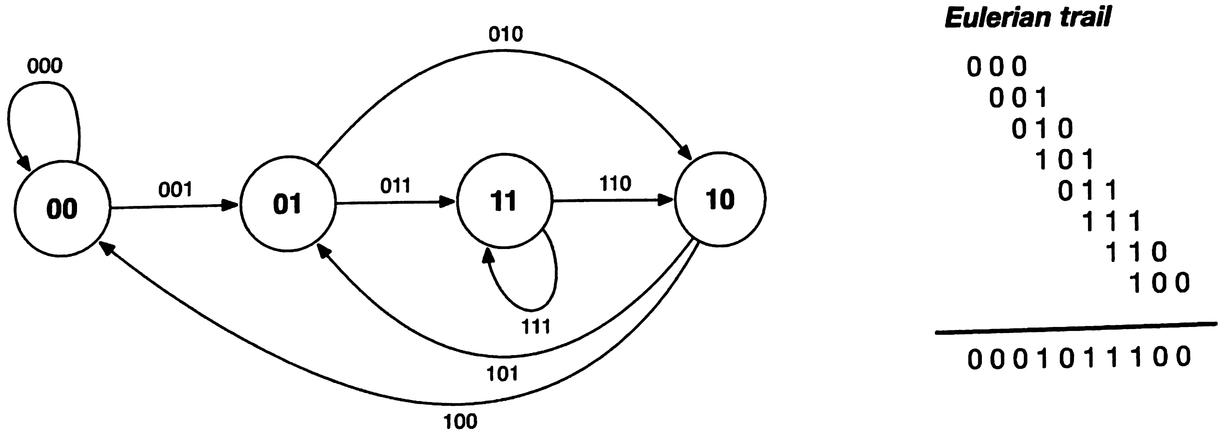


Figure 2-10: An example of constructed De Bruijn graph for binary 3-mers

In De Bruijn graph assembly, a De Bruijn graph is built with a given set of sequenced k-mer reads in the same manner. The Figure 2-11 shows an example of De Bruijn graph construction involving a set of 3-mer reads.

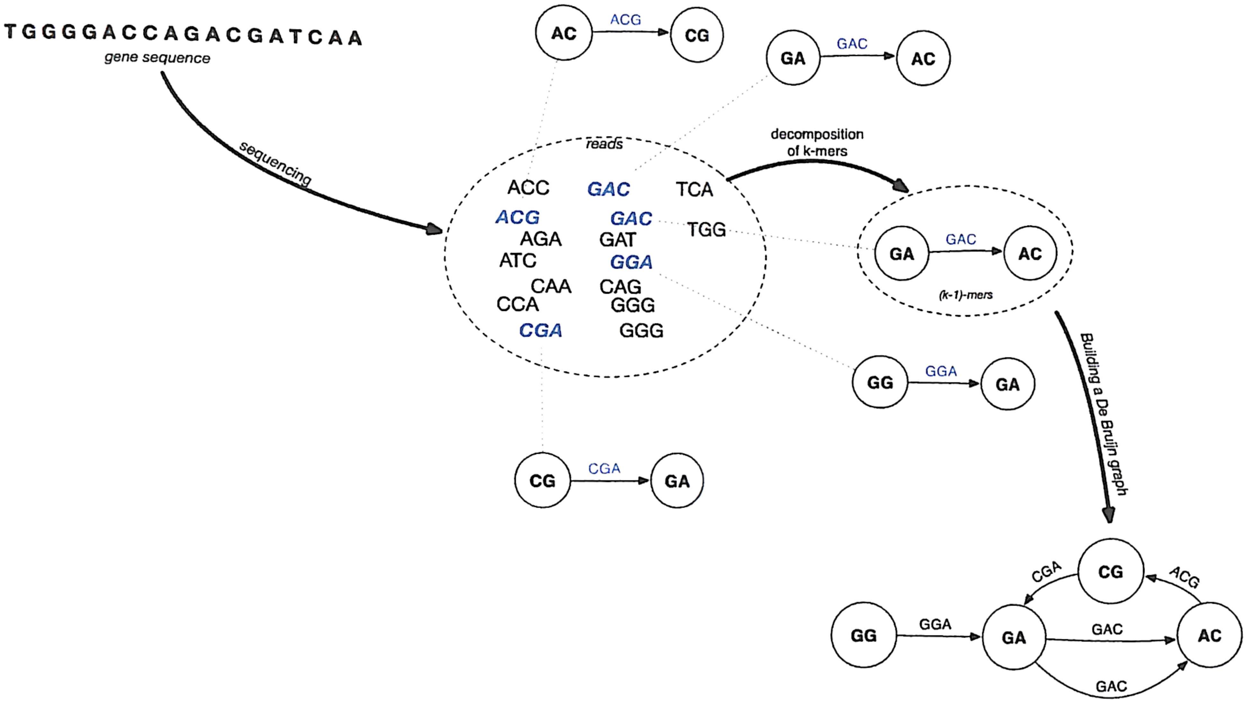


Figure 2-11: An example of constructing a De Bruijn graph with 3-mer sequence reads

As shown, each k-mer is decomposed into its (k-1)-mer prefix and its (k-1)-mer suffix to represent each node pairs connected by a directed edge denoted with their parent k-mer. Then, a De Bruijn graph is constructed by overlapping the matching nodes. Figure 2-22 shows the completed De Bruijn graph.

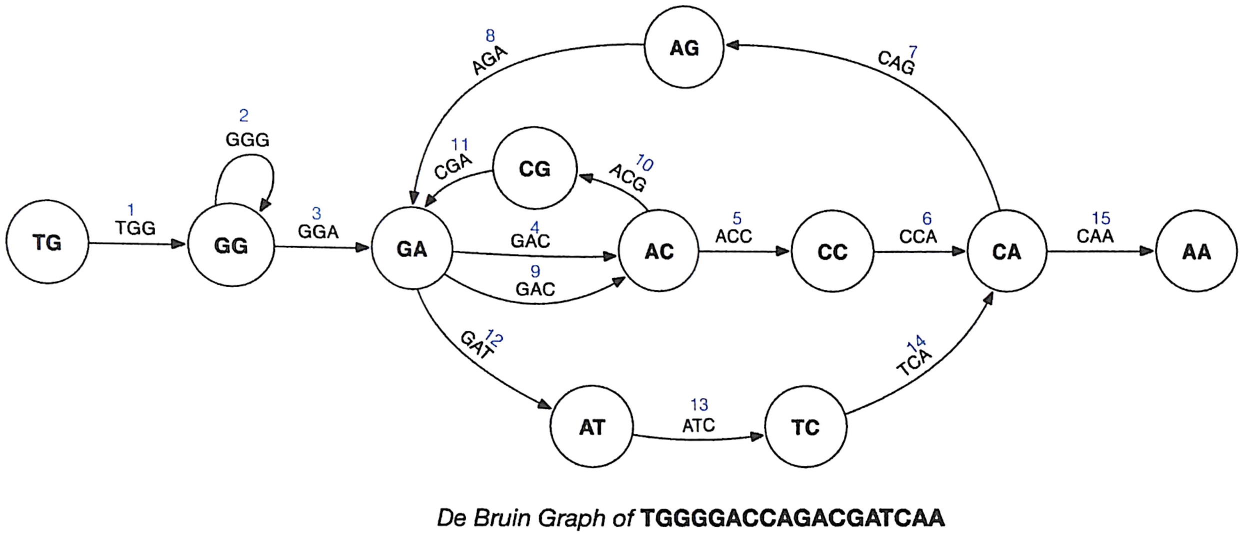


Figure 2-22: An example of constructed De Bruijn graph of 3-mer sequenced reads

Then, the original sequence is extracted by taking the marked Eulerian trail sequentially from the constructed De Bruijn graph.

Theoretically, by applying a small transformation, the complexity of the sequence assembly problem is reduced from finding a Hamiltonian path to finding an Eulerian path. However, the theoretical complexity advantage of De Bruijn graph assembly can lack in practicality in actual sequence assembly. As discussed, the sequenced reads contain errors, repeats, and incomplete coverage. Notice that the repeat read, 'GAC', introduces an alternative Eulerian path in the depicted De Bruijn graph in Figure 2-22, yielding an assembled sequence string of **TGGGG*ACGACCA*GATCAA**. Given an ideal sequence data such as the one used in the example, a multiple number of Eulerian paths often exist due to the repeats in the constructed De Bruijn graph. Unfortunately, there is no known method to identify the correct path to properly map the reads into their original sequence string prior to fragmentation. Thus, the constructed graph is often broken into its *maximal non-branching paths* to extract contigs that are present in any Eulerian path. Also, erroneous reads introduce erroneous nodes in the De Bruijn graph, which leads to formation of erroneous paths. In turn, correcting the erroneous graph can easily intensify the processing complexity and further raise the chance of producing erroneous contigs. Unlike overlap layout consensus assembly, a fixed k-mer length must overlap between reads in the De Bruijn graph assembly. This often necessitates De Bruijn graph assemblers to break the reads into much smaller sized k-mers in order to increase the coverage in De Bruijn graph as shown in Figure 2-23 [27].

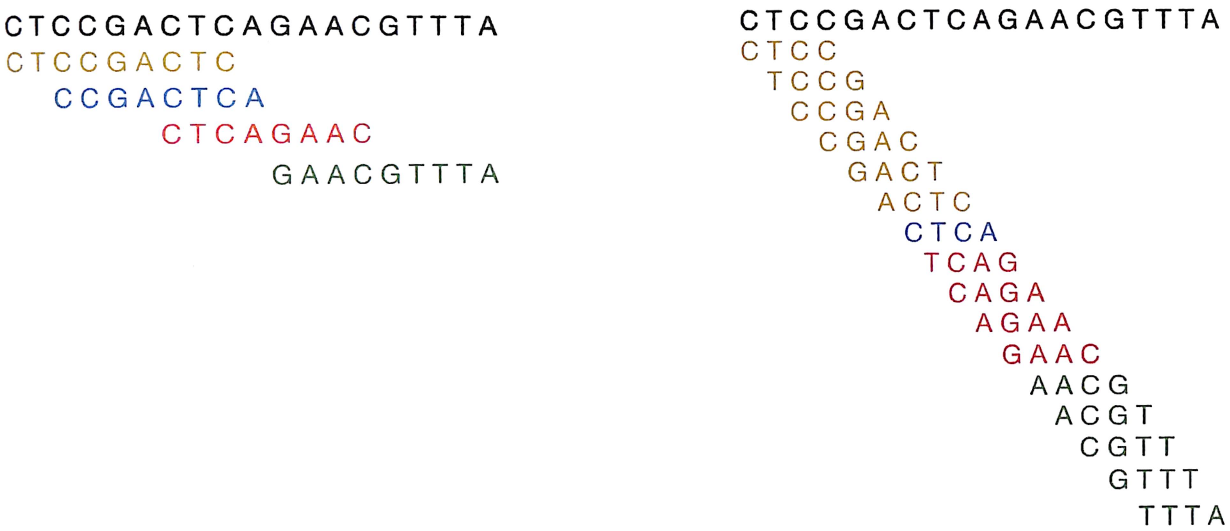


Figure 2-23: An example of read breaking

As a consequence, such read breaking not only jumbles the initial genetic information but also complicates the structure of the constructed De Bruijn graph. On the other hand, the overlap layout consensus assembly shows high tolerance to read errors and incomplete coverage. Through consensus choice of each base from the overlapping reads, correctly sequenced bases can overrule the erroneous bases. Unlike De Bruijn graph assembly, the length of overlapping regions between reads can vary which makes more sense intuitively since the original genetic information was fragmented at random locations. In turn, it allows preserving the sequenced genetic information as it is while maintaining their true coverage.

Studies have shown that overlap layout consensus assembly, in general, outperforms De Bruijn graph assembly for long sequence reads with low coverage. Meanwhile, De Bruijn graph assembly has demonstrated better performance in a short-read high-coverage setting. The length of sequenced reads is expected to grow in parallel with the advancement of the sequencing technology. Hence, the overlap consensus assembly is expected to be the leading assembly method in the future [28].

CHAPTER 3

AN OVERVIEW OF MULTI-OBJECTIVE OPTIMIZATION

The term, computational intelligence (CI), applies to a set of computing algorithms that tackle complex problems with nature inspired heuristic strategies. Common computational intelligence algorithms include artificial neural network (ANN) algorithms, which are based on the functioning of the nervous system, box counting algorithms based on fractal geometry of nature, negative selection algorithms based on the immune system, and so on. Particularly, the ant colony optimization algorithm (ACO) and evolutionary algorithms (EAs) have been widely utilized for solving combinatorial optimization problems such as the Knapsack Problem and the Traveling Salesperson Problem (TSP). Hence, these methods are also suitable for solving the Sequence Assembly Problem (SAP).

3.1 Ant Colony Optimization

The ant colony optimization algorithm [29] is founded on the exhibited collectively intelligent behaviors emerging from a colony of ants' decentralized simple behaviors. Particularly, the emergent intelligence observed in their foraging behavior has been well adapted for solving combinatorial optimization problems [30, 31, 32]. Ants exploit a discovered food source through a positive feedback mechanism known as *stigmergy*. Carrying a piece of food stimulates an ant to lay a chemical substance called *pheromone* on the path taken. Consequently, the pheromone attracts other ants to the path that leads to the food source. As a result, a pheromone trail leading to the food source is defined and strengthened relative to the amount of pheromone deposited by ants. Due to the evaporating property of pheromones, a longer path is likely to lose more pheromone strength in a given time compared to a shorter path. Thus, a shorter path to a food source from a nest is more likely to be exploited by more ants and maintain a strongly defined path. Therefore, when an obstacle is placed on a path being exploited, a shorter path around the obstacle becomes more defined over time as shown in Figure 3-1. The strength of the pheromone around the shorter side (*'C'*) of the obstacle becomes more defined over time than the longer side (*'H'*) of the obstacle due to the pheromone evaporation and difference in the distance traveled around the obstacle.

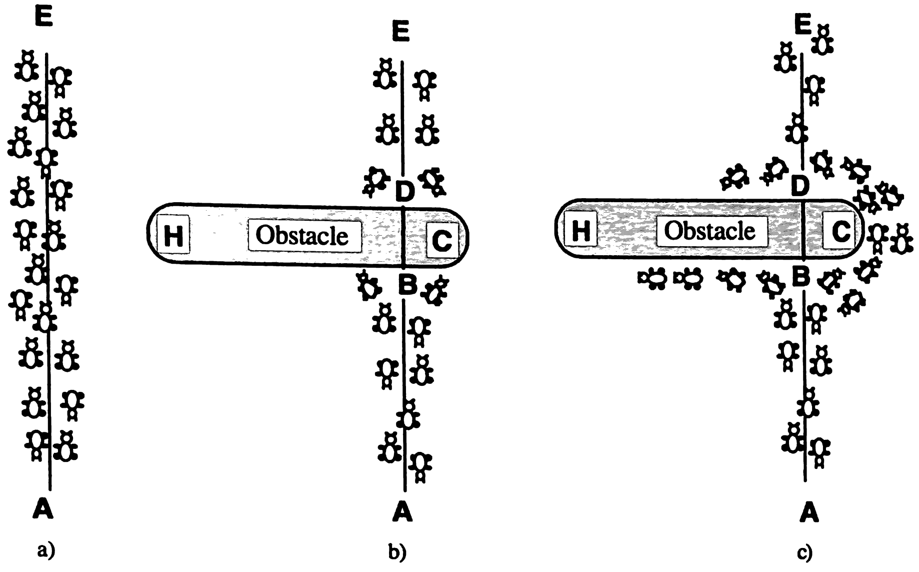


Figure 3-1: An illustration of stigmergy observed in ant colony foraging activity. [33]

Interestingly, not all ants are attracted to a particular path based on the pheromone strength. A few ants will continue to explore in random areas instead of exploiting the path taken by the majority of other ants. This behavior helps to explore for other food sources that could potentially be better.

The manifested collective intelligence of ants foraging behavior, largely characterized by the exploitation of the optimizing objectives and the exploration for a better solution, has made the ant colony optimization algorithm exceptionally suitable for solving of the Traveling Salesperson Problem. As discussed earlier, both the Traveling Salesperson Problem and the Overlap Consensus Sequence assembly problem search for an optimal Hamiltonian path for a given set of objectives. Hence, the basic concept of the ant colony optimization algorithm for the Traveling Salesperson Problem will be described as an introduction to the modified Ant Colony Optimization (ACO) algorithm used in the proposed system presented in a later section.

Given a connected graph, $G = (V_{cities}, E_{paths})$, a randomly distributed K ants build candidate solutions of unique Hamiltonian paths in parallel. The algorithm runs for t iterations and the best solution is found by iteratively updating the global optimal solution with the top solution returned by ants in each iteration stage. Each city-to-city transition taken by an ant can take place either to exploit or explore the solution search space. Exploration occurs with a very small probability to explore other random solutions not bound by a current environment setting such as a local optimal solution. During exploration, an ant k takes a transition from its current city i to a randomly chosen city r. When exploiting, a transition taken by an ant k from city i to a next city j is determined by the level of heuristic visibility (η_{ij}) and pheromone strength (τ_{ij}) present on the path between city i and city j. In TSP, the inverse distance between two cities can be used as the measured value of heuristic visibility so that a shorter path is preferred over a longer path. The pheromone strength on an edge is determined by the amount of pheromone deposited by the ants from previous iterations and the lost amount of pheromone due to evaporation. In addition, an ant can transition to a city only if the city was unvisited previously since the problem seeks a Hamiltonian path. To keep a track of unvisited cities, ants are given a simple memory location known as '*tabu list*'. In summary, the transition probability $p_{ij}^{k}(t)$ for an ant k at iteration t can be expressed as:

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l\in J_i^k}[\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} & ; if\ j \in J_i^k \\ 0\ ; otherwise \end{cases}$$

Here, the contribution level of the heuristic visibility and the pheromone amount in the transition probability can be controlled by the weighting exponent values of α and β. The normalizing denominator is computed by summing the product of heuristic visibility and pheromone level of each of the unvisited neighbor cities J_i^k.

Upon completion of a tour T in iteration t by an ant k, pheromone $\Delta\tau_{ij}^k(t)$ is deposited on each of the visited edges. Here, the amount of pheromone deposit is inversely related to the length of the tour $L^k(t)$ weighted by a user specified Q value.

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & ; if(i,j) \in T^k(t) \\ 0\ ; otherwise \end{cases}$$

This way, the total amount of pheromone $\Delta\tau_{ij}(t)$ deposited on an edge between cities i and j by all ants in iteration t is obtained by summing all their pheromone amounts laid on the edge with respect to their tour length.

$$\Delta\tau_{ij}(t) = \sum\nolimits_k \Delta\tau_{ij}^k(t)$$

Also, the current pheromone amount on an edge of city i to j is reduced by a pheromone evaporation rate ρ. Thus, the remaining pheromone amount after evaporation is considered for the pheromone update.

$$\tau_{ij}(t) - \rho \cdot \tau_{ij}(t) = (1-\rho) \cdot \tau_{ij}(t)$$

Therefore, the pheromone updates on the edges of the paths taken by ants are updated upon completion of their tour as follows:

$$\tau_{ij}(t) = (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t)\,, where\ \rho \in (0,1]$$

The flowchart shown in the Figure 3-2 outlines the sequence of the main operations of the ant colony optimization algorithm. The algorithm runs for a specified total iteration '*max_iter*' and the best solution is found by comparing the top solutions found in each iteration and updating the current best solution with a better solution.

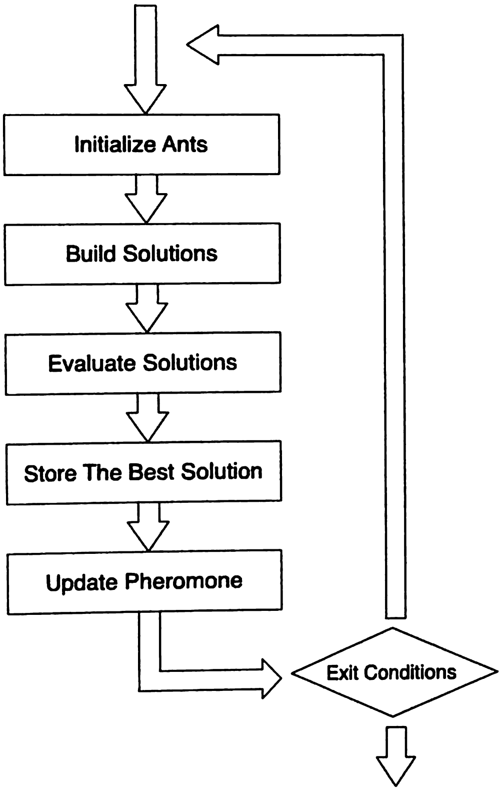


Figure 3-2: The flowchart of the Ant Colony Optimization (ACO)

The ant colony optimization algorithm for the traveling salesperson problem can be further improved by emphasizing the path taken by the best solution found so far up to the current iteration by an elitist ant during pheromone updates. With incorporation of such elitism, the pheromone update can be stated as follows.

$$\tau_{ij}(t) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}(t) + b.\Delta\tau_{ij}^{b}(t),$$

where,

$$\Delta\tau_{ij}^{b}(t) = \begin{cases} \frac{Q}{L^{best}} \; ; if(i,j) \in best \\ 0 \, ; otherwise. \end{cases}$$

3.2 Evolutionary Algorithm

Evolutionary algorithms, also referred to as genetic algorithms, are inspired by Darwin's discovery of evolving species on Earth where the fittest species in a given environment were observed to have a higher chance to pass their genetic material to the next generation. Through selective reproduction and random mutation, individuals are genetically evolved so that individuals equipped with genes with better adaptability and higher survivability are presented to their environment in each generation. Since the environment itself is also changing, its population's evolution is continuous and non-convergent. However, the evolutionary algorithm simulates the evolving process with a given set of candidate solutions for solving a problem with fixed objectives. Intermixed and randomly altered traits of individual candidate solutions are passed to the next generation to iteratively lead them toward a set of quasi-optimal, "near-convergent" solutions.

Figure 3-3 displays a schematic outline of the main procedures involved in an evolutionary algorithm in their sequential order.

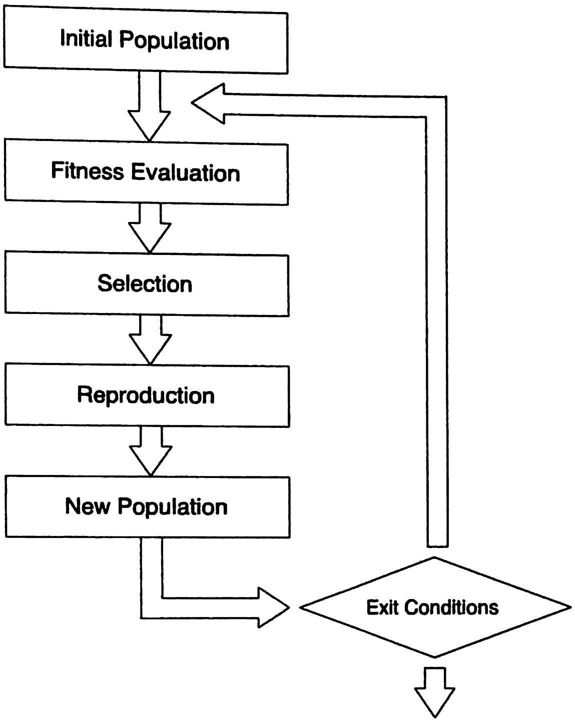


Figure 3-3: The flowchart of the Evolutionary Algorithm (EA)

Initial population is composed of randomly generated candidate solutions to a given problem. Each candidate solution is often referred to as chromosome, or simply, individual. Various types of chromosomes, or encoding of chromosomes, exist for different types of problems. Particularly, the permutation encoding is suitable for the combinatorial optimization problems such as the TSP. For instance, a chromosome for the TSP represents a unique sequence order of visited cities as shown in the Figure 3-4.

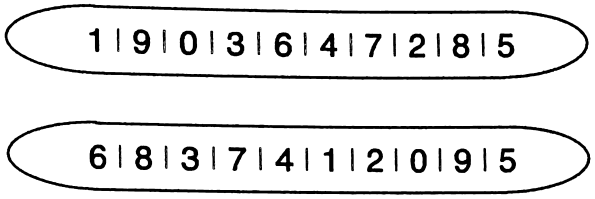


Figure 3-4: Two examples of permutation encoding for TSP involving ten cities.

Fitness of a chromosome measures the strength of the solution with respect to the objectives of a problem. In TSP, chromosome fitness can be measured by taking the inversed total distance of the route traverse according to its encoded sequential order of cities. Thus, the fittest chromosome in a population contains the shortest tour of the traversed cities.

Selection operation in genetic algorithm plays the role of passing the 'good' genetic material to the next generation by selecting chromosomes for reproduction based on their fitness. A chromosome's fitness is directly correlated with its chance of getting selected. Maintaining the diversity of chromosome prevents introducing biased and constricted solution search space into next generations. By assigning all chromosomes a probability of getting selected according to their fitness, premature convergence to a local optimum solution is prevented while keeping the solutions evolving toward the global optimum or quasi-optimum solution. Common selection methods include, but are not limited to, tournament selection, Roulette wheel selection, rank selection, and so on.

Reproduction in genetic algorithms is a process that simulates the exchange of genetic material between selected parent chromosomes using crossover, with the addition of some, usually of low-probability, mutation. This way, genetic recombination of chromosomes allows for exploration of different candidate solutions. In addition, highly customizable crossover and mutation operators allow flexible operation of genetic algorithm appropriate to the type of problem and the type of chromosome encoding. For example, Figure 3-5 shows a crossover operation called *order-1-crossover* for chromosomes encoded to represent the permutation of visited cities in the traveling salesperson problem.

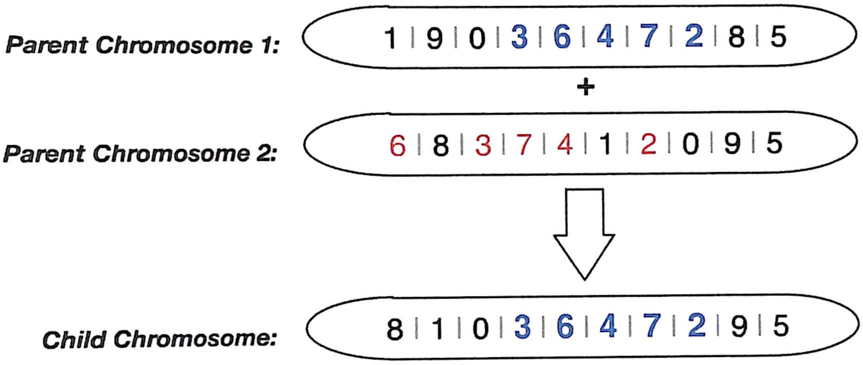
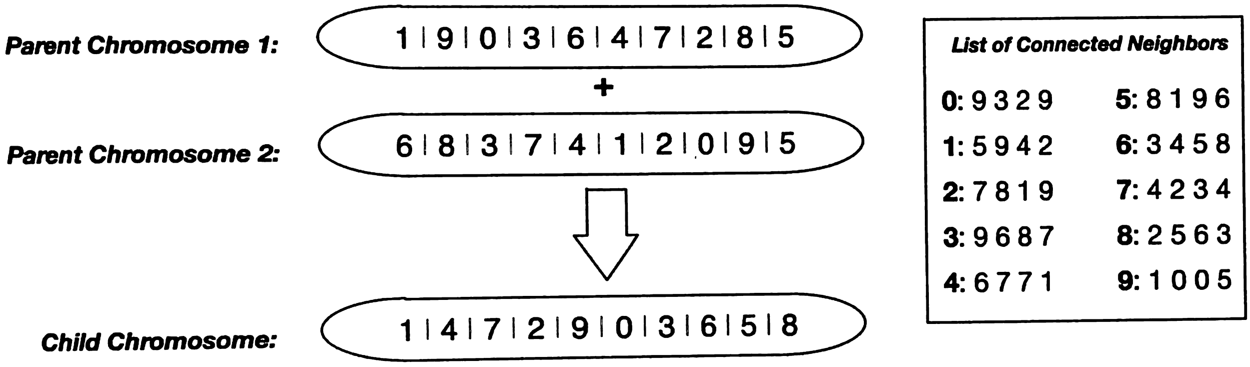


Figure 3-5: An example of order-1-crossover for chromosome with permutation encoding

Here, random consecutive bases are selected from the parent chromosome 1 as shown by the blue marks. Then, the child chromosome inherits the chosen consecutive bases in the same locus. Lastly, the rest of the bases are filled in by iteratively placing the bases from the parent chromosome 2 that are complementary to the selected consecutive bases from parent chromosome 1. In the same way, another child can be produced by selecting a string of random consecutive bases in the parent chromosome 2 instead of parent chromosome 1. Another common crossover operator for permutation-encoded chromosome is the *edge recombination crossover* operator [34]. This operator is especially useful to preserve the existing edges, or adjacent nodes, of the selected parent chromosomes during crossover. Such controlled genetic recombination technique can help prevent a potentially 'good' subset solution from getting easily lost from random shuffling techniques observed in other crossover operators. Figure 3-6 depicts an example of generating a new child chromosome from the selected parent chromosome using the edge recombination operator.



List of Connected Neighbors	
0: 9 3 2 9	5: 8 1 9 6
1: 5 9 4 2	6: 3 4 5 8
2: 7 8 1 9	7: 4 2 3 4
3: 9 6 8 7	8: 2 5 6 3
4: 6 7 7 1	9: 1 0 0 5

Figure 3-6: An example of edge recombination crossover for chromosome with permutation encoding

The child chromosome is built based on the neighbor relationships of each node of the selected parent chromosomes as shown in the adjacent list. Initially, a random parent node is selected as the first node of the child chromosome. Then, the rest of the nodes are filled in by iteratively taking the following two steps. First, the current selected parent node is removed from the adjacency list. Second, the neighbor of the current selected node with the minimum number of unselected neighbors is selected as the next node of the child chromosome. In the case of equal number of unvisited neighbors, a neighbor node having the highest number of redundant unvisited nodes takes the precedence.

Otherwise, a random neighbor of the selected node is chosen as the next node for the child chromosome. The edge recombination operator typically requires higher processing time than other crossover operators. However, it has shown to be advantageous over other crossover operators in combinatorial optimization problems such as the traveling salesperson problem and the sequence assembly problem. Mutation operator is applied to each child chromosome independently. Various mutation techniques exist for permutation chromosomes. However, they are mainly characterized by switching the position of randomly selected nodes as shown in Figure 3-7.

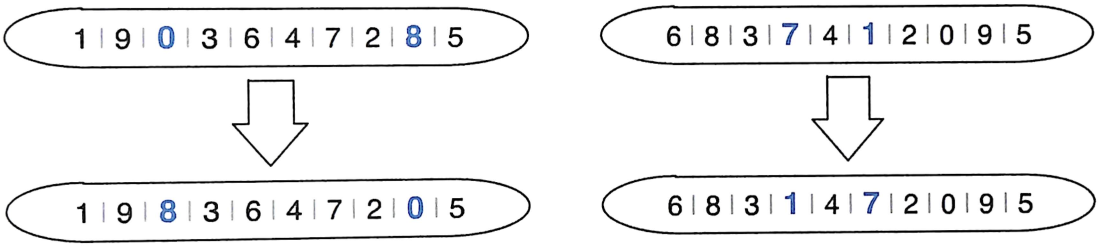


Figure 3-7: An illustration of swap mutation

Crossover and mutation operators dictate how the search space is explored in the evolutionary algorithms. Thus, the method of choice and their probability of occurrence can control the randomness in the solution exploration. In nature, next generations ultimately completely replace their previous generations. However, in evolutionary algorithms, individual chromosomes with good fitness can continue to contribute their genes to the following generations in order to help lead the next generation solutions to the optimal solution. In addition, it helps to preserve the best solutions found so far. This population selection method is known as *elitism*. Evolutionary algorithms continues to run the cycle of the main processes described above until one or more stop criteria are met. Generally, evolutionary algorithms stops its cycle when the specified number of evolution cycle is completed or no significant fitness improvement occurs in the population.

Multiobjective evolutionary algorithm (MOEA) considers multiple, potentially conflicting objectives as fitness evaluation criteria for evolving candidate solutions. For instance, in the traveling salesperson problem, a salesperson might have to consider minimizing the cost of traveling, but at the same time maximizing sales while also minimizing the total distance traveled. In the sequence assembly problem, the fitness of each candidate solution can be based on maximization of the overall overlap strength, minimization of the total number of contig gaps, and maximization of the average contigs length. With such fitness evaluation criteria, the multiobjective evolutionary algorithm evolves the candidate solutions toward Pareto optimal state that best satisfy the trade-off between the problem's objectives. Pareto optimal, also known as Pareto efficient, is a term used to denote a state of optimal allocation of values along problem's objectives where there exists no improvement of an objective without making another objective worse. Thus, Pareto improvement is only possible when there exists an improvement along a problem objective without worsening the conditions along the other objectives. A set of Pareto optimal solutions can also be stated as nondominated solutions given all other solutions with alternative allocations of values along the problem's objectives evaluate to be suboptimal.

3.2.1 NSGA-II

Some of the widely used multiobjective evolutionary algorithms are Nondominated Sorting Genetic Algorithm II (NSGA-II) [35], Strength Pareto Evolutionary Algorithm (SPEA) [36], and Pareto Archived Evolution Strategy (PAES) [37]. Particularly NSGA-II has shown to outperform other multiobjective evolutionary algorithms including SPEA and PAES in various test problems [35]. The strength of NSGA-II comes from its efficient sorting method that ranks each candidate solution based on their level of nondomination, ensuring the diversity of the solution using dynamically computed crowding distance measurement, and incorporating elitism in population selection.

Sorting by nondomination rank refers to assigning solutions according to their level of Pareto optimality. In NSGA-II, each level of nondominated solutions are iteratively found, leaving out the explored nondominated solutions from the solution set in each iteration until all solutions are sorted according to their nondomination rank. A crowding distance measures how crowded a solution is by its surround neighbor solutions and it is used to help maintain diverse solutions throughout generations. In NSGA-II, crowdedness of a solution is the averaged distance of two nearest adjacent solutions along each objective. A less crowded solution from a set of solutions belonging to same nondomination rank is the less common solution and therefore takes preference over more crowded solution since it contributes more to the overall diversity of the solution population.

In each generation, the parent solutions and their child solutions are combined to reinforce elitist selection strategy and sorted based on their nondomination rank. Subsequently, N numbers of next generation candidate solutions are selected based on their rank and their crowding distance. During population selection, the lower ranked solutions, *i.e.*, solutions that are less dominated by other solutions, take precedence over the higher ranked solutions, *i.e.*, solutions that are more dominated by other solutions. Meanwhile, solutions with higher crowding distance from less crowded solution set take precedence over the solutions with lower crowding distance. This way, the Pareto set iteratively evolves toward its true Pareto efficient solution set throughout generations.

CHAPTER 4

PROPOSED METHODOLOGY

Taking the advantages of the overlap layout consensus sequence assembly and the fast-growing sequencing technology into consideration, I propose a new method for solving the sequence assembly problem with hybridization of ant colony optimization and a multi-objective evolutionary algorithm. Figure 4-1s shows the overview of the proposed method.

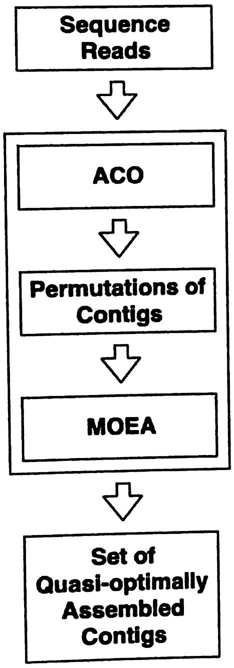
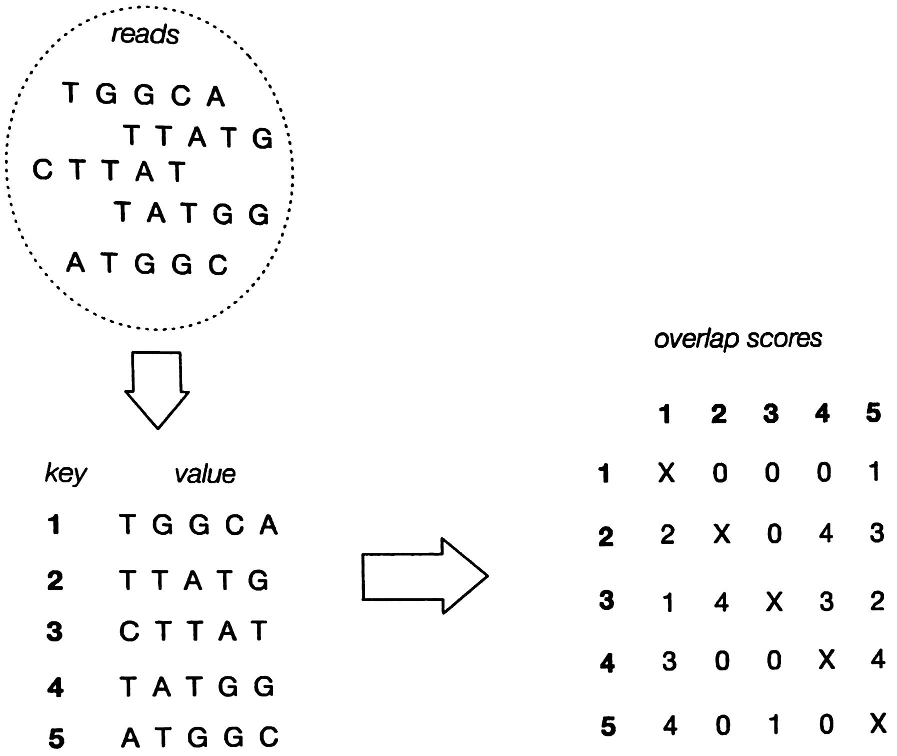


Figure 4-1: A schematic diagram of the hybridization of ACO and MOEA for solving the sequence assembly problem

The main purpose of the modified ant colony optimization algorithm is to assemble a set of short sequence reads and produce the quasi-optimal contigs that yield the highest average overlap score. The resulting contigs which span the whole input sequence reads are permuted to generate the individuals of the initial population for the further processing with a modified multiobjective evolutionary algorithm. A widely used multiobjective evolutionary algorithm, NSGA-II (Nondominated Sorting Genetic Algorithm II), is used to explore the optimal arrangement of the contigs through controlled shuffling with respect to the objectives of the sequence assembly problem. By applying the overlap layout consensus assembly method on the resulting arrangement of the contigs, the proposed system produces a set of quasi-optimally assembled contigs.

4.1 ACO - Contigs Generator

Initially, a set of artificial reads obtained from a sequencing simulator is recorded in an associative array where each read is uniquely identified by an integer value. Also, overlap strength between each of the reads is computed and stored in an adjacency matrix as shown in Figure 4-2.



key	value
1	T G G C A
2	T T A T G
3	C T T A T
4	T A T G G
5	A T G G C

overlap scores

	1	2	3	4	5
1	X	0	0	0	1
2	2	X	0	4	3
3	1	4	X	3	2
4	3	0	0	X	4
5	4	0	1	0	X

Figure 4-2: An example of constructing an adjacency matrix with read overlap scores

Overlap score between two reads is obtained using the method shown in Pseudocode 1 where the maximum overlap score between two reads is found by iteratively computing the overlap score as one read slides against the other read one base at a time.

```
FindMaxOverlapScore(read$_1$, read$_2$)
  Score$_{max}$ ← -1;
  Score$_{max\_index}$ ← -1;
  t ← read$_1$.length;
  FOR i=0 TO t
    Matched$_{bases}$ ← 0;
    Unmatched$_{bases}$ ← 0;
    Score$_{current}$ ← 0;
    Residual$_{bases}$ ← t - i;  //remaining num of bases
    IF Residual$_{bases}$ >= Score$_{max}$ DO
      FOR j=0 TO Residual$_{bases}$
        IF read$_1$[i+j] == read$_2$[j]
          Matched$_{bases}$ ← Matched$_{bases}$ + 1
        ELSE
          Unmatched$_{bases}$ ← Unmatched$_{bases}$ + 1
        END
      Score$_{current}$ ← Matched$_{bases}$ - Unmatched$_{bases}$
      IF Score$_{current}$ > Score$_{max}$ THEN
        Score$_{max}$ ← Score$_{current}$
        Score$_{max\_index}$ ← i
      END
    ELSE
      BREAK
    END
  END
  RETURN (Score$_{max}$, Score$_{max\_index}$)
```

Pseudocode 1: The algorithm used for finding the maximum overlap score of two sequence reads

Due to the overlap directionality between two reads, the overlap scores above and below the diagonal of the depicted adjacency matrix in Figure 4-2 are shown to be unequal.

The ant colony optimization component of the proposed method was implemented by modifying an open source code of ant colony optimization [40] written for solving the traveling salesperson problem using multithreaded programming in Python. The pseudocode for the modified ant colony optimization algorithm is shown in Pseudocode 2. Overall, the modified ant colony optimization generates a set of partially assembled diverse patterns of contigs as local improvement to solving the assembly problem.

```
$S_{best}$ ← RandomSolution(seqReadsInRandOrder);
$S_{best\_cost}$ ← AvgContigOverlap($S_{best}$);
Pheromone ← IntializePheromone(∅);
WHILE iteration < T DO
  ContigPermutations = []; //candidate solutions
  FOR i=1 TO K DO  //K ants
    $S_i$ ← ConstructContigs(OverlapAdjacencyMatrix, Pheromone, $\alpha$,
          $\beta$, minOverlap);
    $S_{i\_cost}$ ← AvgContigOverlap($S_i$);
    IF $S_{i\_cost} \leq S_{best\_cost}$ THEN
      $S_{best\_cost}$ ← $S_{i\_cost}$;
      $S_{best}$ ← $S_i$;
    END
    ContigPermutations ← $S_i$;
  END
  EvaporatePheromone(Pheromone, $\rho$);
  FOREACH path ∈ ContigPermutations DO
    $S_{elite}$ ← $S_{best}$;
    UpdatePheromone(Pheromone, path, $path_{cost}$ ,$S_{elite}$);
  END
END
RETURN $S_{best}$;
```

Pseudocode 2: The algorithm used for the main loop of the modified ACO

Based on the computed read overlap scores in an adjacency matrix, contigs are generated by each ant in each iteration using the algorithm shown in Pseudocode 3.

```
ConstructContigs(OverlapAdjacencyMatrix, Pheromone, $\alpha$, $\beta$,
        minOverlap)
  $R_{start}$ ← InitialStartNode();
  $Path_{explored}$ ← [$R_{start}$]; //path taken
  $Path_{unexpored}$ ← [ID_of_All_Reads - $R_{start}$];
  $R_{current}$ ← $R_{start}$;
  WHILE |$Path_{unexplored}$| > 0 DO
    $R_{next}$ ← ChooseNextNode($R_{current}$, OverlapAdjacencyMatrix,
            Pheromone, $\alpha$, $\beta$); //apply a transition rule
    IF OverlapScore($R_{current}$, $R_{next}$) < minOverlap DO
      $R_{next}$ ← PickRandomNode();
      $Path_{explored}$.append($R_{next}$);  //start of a new contig
    ELSE
      $Path_{explored}$[|$Path_{explored}$|-1].append($R_{next}$);
    END
    $Path_{unexplored}$.remove($R_{next}$);
    $R_{current}$ ← $R_{next}$;
  END
  RETURN $Path_{explored}$;
```

Pseudocode 3: The algorithm for constructing contigs per each complete tour around all sequence reads

The probabilistic transition, *ChooseNextNode(...)*, of an ant is made from a current sequence read to next sequence read, determined by the overlap strength and the amount of pheromone existing between the current sequence read and its neighbor sequence reads. Condition exists for extending a current contig where a specified minimum overlap score, *minOverlap*, must be satisfied between two reads. In the case of condition failure, a random neighbor sequence read is selected and becomes the beginning sequence bases of a new contig. Consequently, the transition condition helps to prevent the formation of contigs by reads with weak overlap strength. Pheromone deposit on the path traversed by an ant is directly correlated with the average overlap scores of the formed contigs. Formally, pheromone deposit on an edge between sequence read *i* and *j* is computed by taking the average of the overlap score C of n number of contigs formed by ant k at iteration t.

$$\Delta\tau_{ij}(t) = \begin{cases} Q \cdot \dfrac{\sum_{p=1}^{n} C_p^k(t)}{n} & ; if\ (i,j) \in T^k(t) \\ 0\ \ ; otherwise \end{cases}$$

Pheromone deposit amount is applied if and only if the edge is part of the traversed path T and is weighted by a specified value Q. Additionally, the best path found so far until current iteration t is further reinforced during pheromone updates through incorporation of elitism. The current pheromone level is also reduced by pheromone evaporation to prevent premature convergence to local optimal solution. Overall, pheromone update at iteration t is defined by reduction of current pheromone level by the pheromone evaporation rate ρ, pheromone deposit by all ants, and pheromone deposit by the elite ant with the best path so far.

$$\tau_{ij}(t) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) + b.\Delta\tau_{ij}^{b}(t)$$

At the completion of a specified total number of iterations, the optimal set of contigs with the maximum average overlap scores is obtained.

4.2 Permutations of Contigs

A new associative array and adjacency matrix are obtained based on the set of contigs output from the modified ant colony optimization. Consequently, by taking N random permutations out of the key values of the contigs from the associative array, initial population input to the subsequent multiobjective evolutionary algorithm is generated as shown in the Figure 4-3.

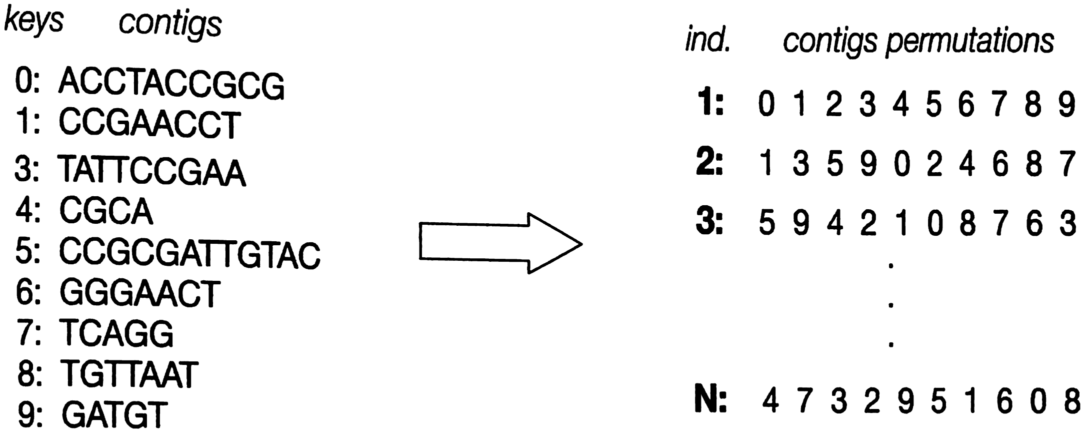


Figure 4-3: Reassigning IDs to new contigs and generating an initial population for MOEA

4.3 MOEA – Contigs Assembler

A nondomination-based multiobjective evolutionary algorithm, NSGA-II [35], was implemented using the framework provided by Python package DEAP (Distributed Evolutionary Algorithm in Python) [44] for processing of the individuals of contig permutations toward the final quasi-optimal solutions of the sequence assembly. The evolutionary algorithm component of the proposed system aims to heuristically explore the search space bounded by contig permutations to find a set of solutions that optimally dominates the objectives of the individual fitness criteria with the aid of custom genetic operators and elitist selection strategy incorporated into NSGA-II. The pseudocode shown in Pseudocode 4, Pseudocode 5, and Pseudocode 6, gives a high-level description of the main operations of NSGA-II.

```
initPop ← InitializePopulation(PopSize)
FastNonDominatedSort(initPop)
RankSolutions(initPop)  //based on nondomination level
$P_0$ ← Select($P_0$, N)
$Q_0$ ← MakeNewPop($P_0$)
WHILE $t \le T$ DO  //run for T generations
  $R_t$ ← $P_t \cup Q_t$  //combine parent and offspring
  F ← FastNonDominatedSort($R_t$)
  $P_{t+1} = \emptyset$  //next parents
  i = 1  //front(nondominated) level
  WHILE $|P_{t+1}|+|F_i| \le N$ DO  //N num of parents
    CrowdingDistanceAssignment($F_i$)
    $P_{t+1}$ ← $P_{t+1} \cup F_i$
    i = i + 1
  END
  Sort($F_i$, $\prec_n$)  //$\prec_n$ crowded-comparison operator
  $P_{t+1}$ ← $P_{t+1} \cup F_i[1:(N-|P_{t+1}|)]$  //fill the last by distance
  $Q_{t+1}$ ← MakeNewPop($P_{t+1}$)
  t ← t + 1
END
RETURN $Q_T$
```

Pseudocode 4: The algorithm used for the main loop of the NSGA-II MOEA [35].

```
FastNonDominatedSort(P)
  FOR EACH p ∈ P  //P all solutions
    $n_p$ = 0  //num of solution that dominates p
    $S_p$ = ∅  //solutions that p dominates
    FOR EACH q ∈ P
      IF p ≺ q THEN  //If p dominates q
        $S_p$ = $S_p$ ∪ {q}
      ELSE IF q ≺ p THEN
        $n_p$ = $n_p$ + 1
      END
    END
    IF $n_p$=0 THEN
      $p_{rank}$ = 1
      $F_1$ = $F_1$ ∪ {p}  //add p to front
    END
  END
  i=1  //front counter
  WHILE $F_i$ ≠ ∅  //initial $F_i$=$F_1$=Nondominated front
    Q = ∅  //members of the next front
    FOR EACH p ∈ $F_i$
      FOR EACH q ∈ $S_p$  //$S_p$ = P - $F_i$
        $n_q$ = $n_q$ - 1
        IF $n_q$ = 0 THEN
          $q_{rank}$ = i + 1
          Q = Q ∪ {q}
        END
      END
      i = i + 1
      $F_i$ = Q
    END
  END
```

Pseudocode 5: The algorithm for the fast nondominated sorting method [35]

```
CrowdingDistanceAssignment(I)
  L = |I| //number of solutions in I
  FOR EACH i
    set $I[i]_{distance}$=0  //initialization
  END
  FOR EACH objective m
    I = sort(I, m)  //sort for each objective
    $I[1]_{distance}$ = $I[l]_{distance}$ = ∞
    FOR i=2 TO (l-1)
      $I[i]_{distance}$ = $I[i]_{distance}$ + (I[i+1].m-I[i-1].m)/($f_m^{max} - f_m^{min}$)
    END
  END
```

Pseudocode 6: The algorithm for computing and assigning the crowding distance [35]

The objectives for the fitness evaluation of each candidate solutions are: 1) average contig overlap score, 2) the total number of contig gaps, and 3) average contig length. Assuming that C denotes the overlap score of a contig, and L denotes the length of a contig involving N total number of contigs, the fitness functions can be stated as follows.

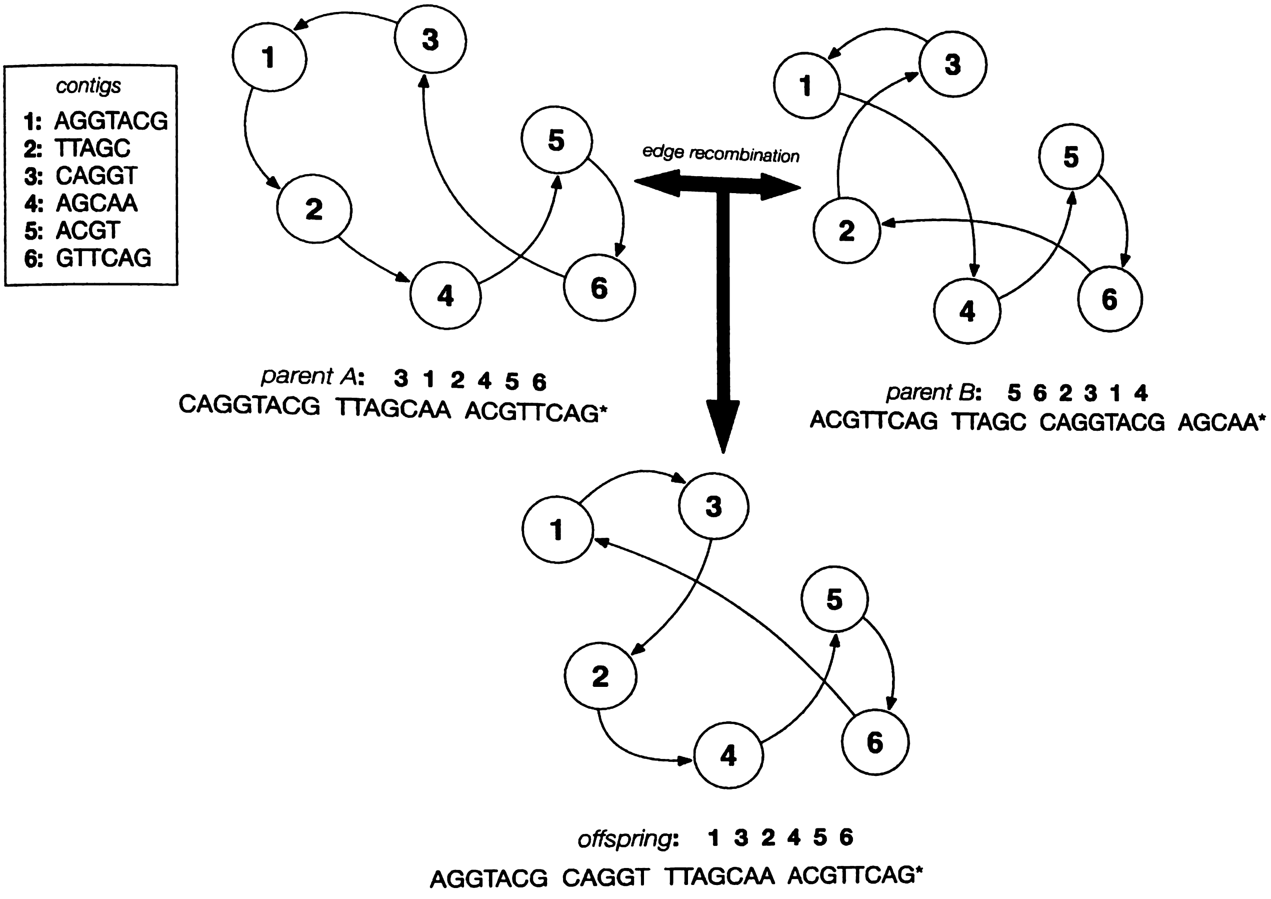
$$f_1 = \frac{\sum_i^N C_i}{N}$$

$$f_2 = N - 1$$

$$f_3 = \frac{\sum_i^N L_i}{N}$$

Then, the evolutionary algorithm evolves the candidate solutions towards the optimal tradeoffs between maximization of average contig overlap score, minimization of total number of contig gaps, and maximization of average contig length.

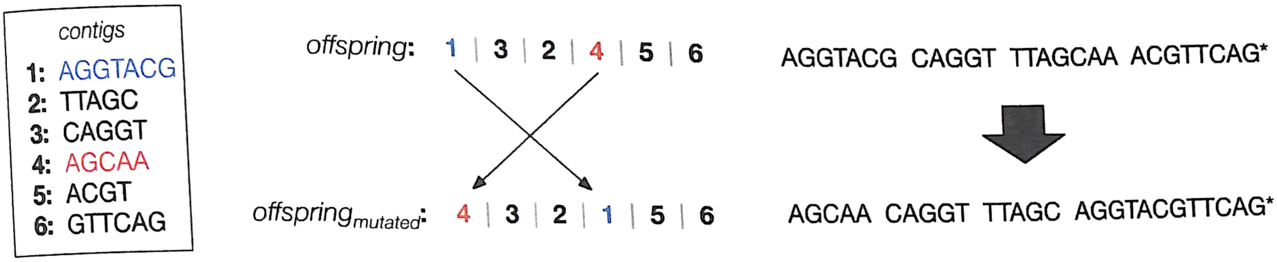
A set of candidate solutions for the next generation is produced by edge recombination crossover operator and swap mutation. As mentioned in the previous section, the edge recombination helps to retain the edge information of the parents during crossover operation. In turn, it prevents from random exploration in the search space and helps to pass the good partial solution content found in parents, assembled contigs with high overlap score, to their offspring in a controlled setting as shown in Figure 4-4.



* represents assembled contigs with minimum overlap score of 2

Figure 4-4: An illustration of edge recombination crossover operator used for the proposed assembly method

Swap mutation operation brings change in the contig permutation pattern of an offspring through random partial reformation of its genetic content. Mutation operators, in general, occur at a relatively low probability compared to crossover operators and they are primarily used to explore other random solution candidates in order to overcome converging into a locally optimal solution. Swap mutation operator in the proposed system is programmed to simply exchange the randomly chosen elements in two loci as shown in Figure 4-5.


contigs
1: AGGTACG
2: TTAGC
3: CAGGT
4: AGCAA
5: ACGT
6: GTTCAG
offspring: 1 | 3 | 2 | 4 | 5 | 6
AGGTACG CAGGT TTAGCAA ACGTTCAG*
offspringmutated: 4 | 3 | 2 | 1 | 5 | 6
AGCAA CAGGT TTAGC AGGTACGTTCAG*

* represents assembled contigs with minimum overlap score of 2

Figure 4-5: An illustration of swap mutation for the proposed assembly method

Taking the advantage of elitism, a specified N number of next generation candidate solutions are selected from a pool of both offspring and parents solutions based on their non-dominating ranks and their crowding distance. At the completion of t generations, a set of Pareto optimal solutions that represent the quasi-optimally fitting set of assembled contigs along the objectives are obtained.

CHAPTER 5
EXPERIMENTS

The main goal of conducted experiments was to study the advantages and disadvantages in various aspects of the proposed system, but also to estimate the degree of positive contribution made by the proposed system relative to a modern OLC-based sequence assembler. The areas of the studies included: 1) testing for proper functionality as designed, 2) studying the role of each component of the proposed system, 3) examining the diversity of Pareto-front solutions of assembled contigs, 4) analyzing the output variability under varying input sequence settings, and 5) assessing the quality, accuracy, and practicality of the assembled contigs in comparison with other sequence assembly methods.

5.1 Experiment Setups and Settings

Libraries of input sequence data were prepared for experiments using a sequencing simulation tool, ART [38]. The synthetic sequence data were sequenced in reference to randomly generated genomes and samples of known genomes. The simulated sequencing technology was Illumina sequencing technology, *HiSeq 2500* [53].

The default read length, read depth, and read error rate were set as shown in Table 1 for the sequencing simulation.

Read Length (nt)	Read Depth	Error Rate
50	10x	1.5%

Table 5-1: The default setting for single read sequencing simulation (HiSeq 2500)

The default settings for the control parameters of each component of the proposed system are displayed in Table 5-2 and Table 5-3.

Parameter	Parameter Value	Description
T	50	number of iterations
N	10% of number of reads	number of ants
O_{min}	$10\%\ of\ read\ length$	minimum read overlap score
ρ	0.35	pheromone evaporation rate
α	1.0	read overlap weight
β	1.0	pheromone strength weight
C_{ind}	0.25	wt. of avg. read overlap score in individual contig
C_{avg}	0.75	wt. of avg. read overlap score of all contigs

Table 5-1: The default control parameter setting for the ACO component of the proposed method

Parameter	Parameter Value	Description
T	200	number of generations
N	10% of number of reads	population size
O_{min}	$10\%\ of\ avg.\ read\ length$	minimum read overlap score
CX	1.0	crossover probability rate
MT	0.2	mutation probability rate

Table 5-2: The default control parameter setting for the MOEA component of the proposed method

The size of the ant colony in ACO and the population in MOEA were set to change dynamically relative to the size of the input sequence reads to provide sufficient amount of solution exploration. The minimum read overlap score required for contig formation was set to 10% of average read length to prevent formation of long contigs with poor read overlaps which in turn may lead to false qualification of each candidate solution.

5.2 Evaluation of System Reliability

Based on the default settings, the proposed system was evaluated for its performance reliability. The test was conducted using a set of synthetic input sequence reads sequenced based on a synthetic genome with length of 1000 nucleotides (nt). The reliability of the system was observed by inspecting the overall change in solution quality during the sequence assembly processing in each component of the proposed system. Specifically, the averaged contig overlap scores throughout iterations during the ACO processing and the individual fitness measurement along each objective throughout generations during the MOEA processing were plotted and observed. Figure 5-1 displays the best candidate solutions discovered throughout iterations during the ACO processing.

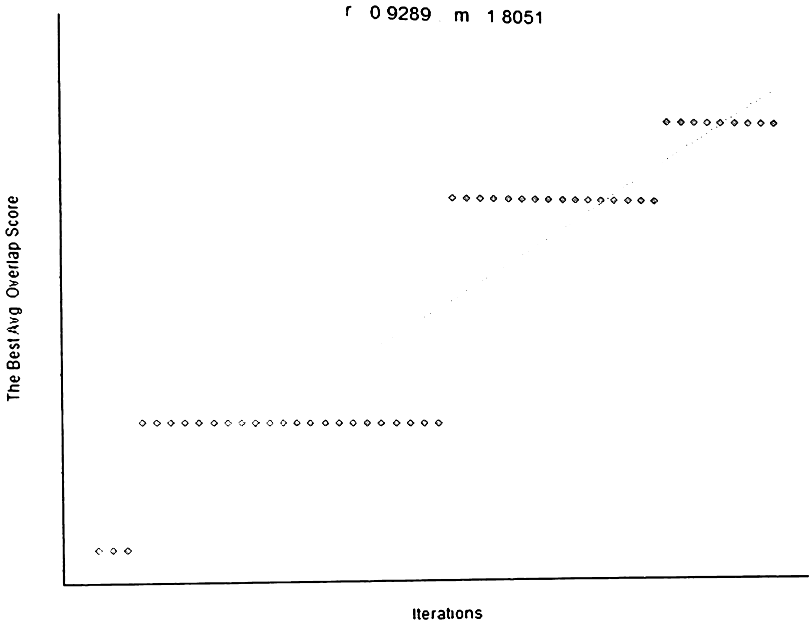


Figure 5-1: The scatter plot of the best avg. contig overlap scores throughout iterations during ACO processing

The observed correlation measurement of 0.9289 and the positive slope coefficient value of 1.8051 from a fitted simple linear regression model indicated an overall improvement of the averaged contig overlap scores throughout iterations. However, the appearance of staircase-like plot was called into question whether solutions were being discovered stochastically. In order to investigate the questioned behavior of the system, the top solution out of all solutions returned by all ants in each iteration was plotted as shown in Figure 5-2. For clarity, the best solution at an iteration refers to the best solution found out of all solutions from iterations preceding the iteration, meanwhile the top solution at an iteration refers to the top solution found out of all solutions only from the current iteration.

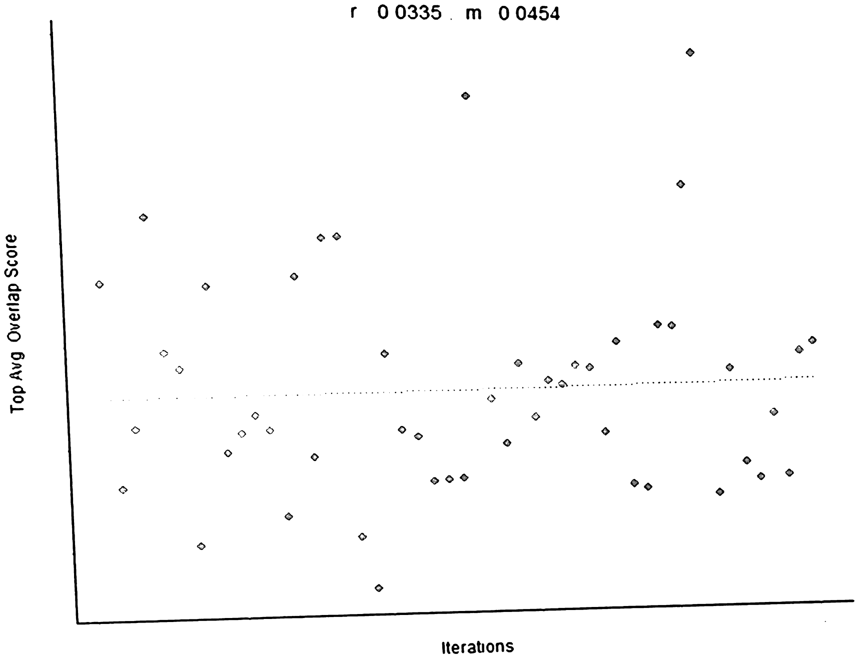


Figure 5-2: The scatter plot of the top avg. contig overlap scores throughout iterations

The plot shows that the top solutions found in each iteration does not suggest overall increment throughout iterations. To ensure that it is not caused by lack of solution exploration, the ACO component was tested with different ranges of ant colony sizes and iterations. However, similar behavior persisted throughout different parameter settings. The cause of the questioned behavior is highly suspected to be rooted in the random choice of initial start node/read and the multiple occurrence of random choice of node/read encountered whenever minimum overlap score is not satisfied. In other words, the minimum read overlap score constraint is suspected to be one of the main reason to observed difficulty in quasi-optimal solution convergence. Still, the overall best solution was used to produce the ACO-generated contigs that serves to form the initial population for the following MOEA processing. Similarly, to assess the reliability of the MOEA component of the proposed system, the fitness measurements along each objective throughout generations were plotted as displayed in Figure 5-3, Figure 5-4, and Figure 5-5.

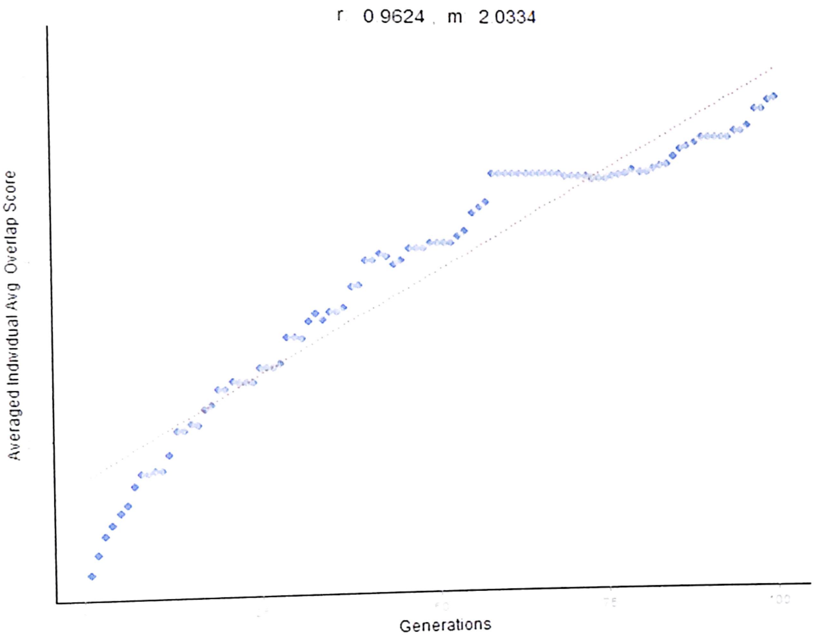


Figure 5-3: The scatter plot of averaged individual avg. contig overlap score fitness measurements throughout generations

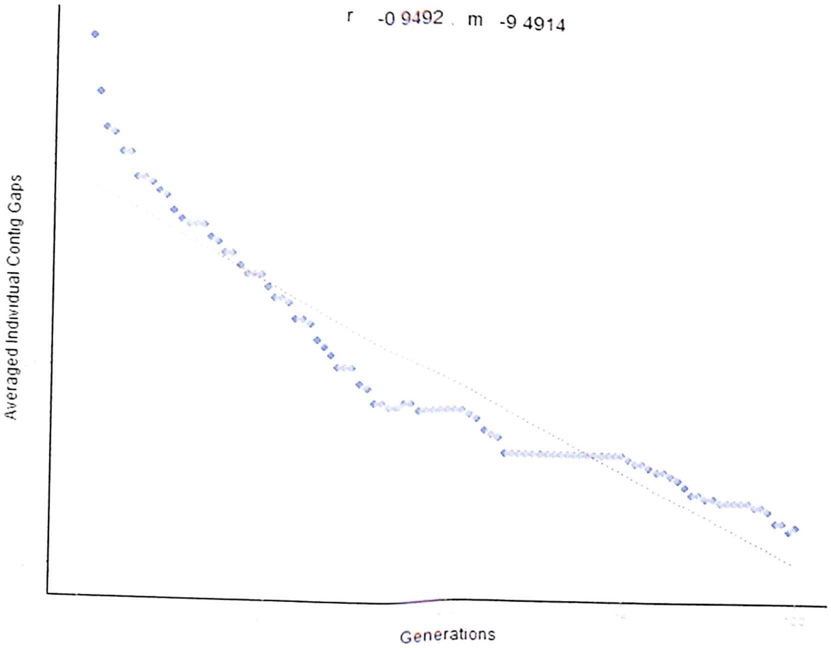


Figure 5-4: The scatter plot of averaged individual contig gap fitness measurements throughout generations

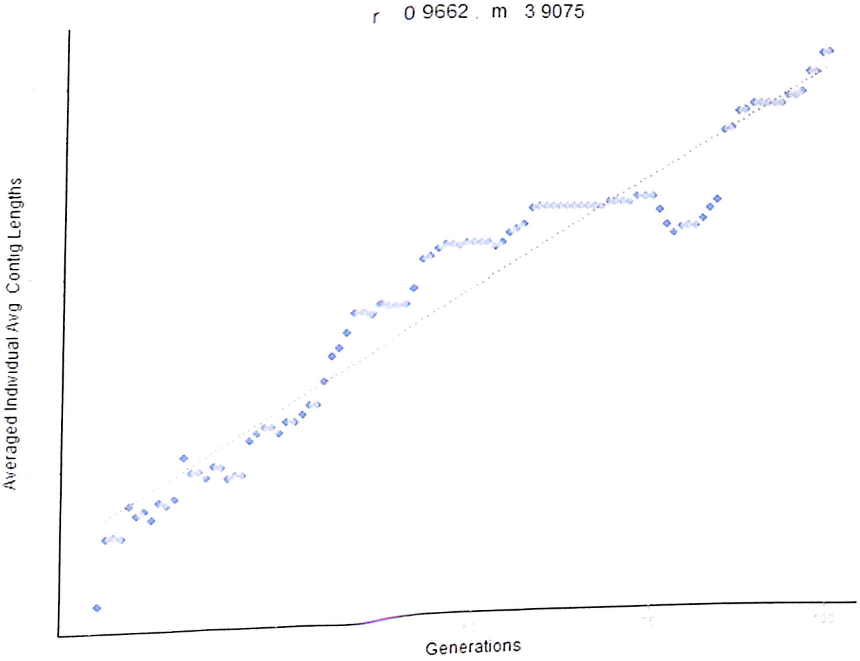


Figure 5-5: The scatter plot of averaged individual avg. contig length fitness measurements throughout generations

The plots display evolution of the initial population toward its quasi-optimal Pareto fitness with consistent maximization of individual average contig overlap scores, minimization of individual number of contig gaps, and maximization of individual average contig lengths throughout generations. As indicated by the slope coefficient of the fitted regression model, the MOEA component displayed lack of high leverage in fitness per generation. This observation suggested that incorporation of the edge recombination crossover operator might possibly be decelerating the rate of fitness improvement. In addition, incorporation of elite solutions from previous generations were considered to be the main contributing factor in the observed overall improvement of the population fitness throughout generations.

5.3 Evaluation of System Robustness

The robustness of the proposed system was evaluated to observe whether the variability in the output remains in an acceptable range when the system is executed multiple times with the same input settings. The behavior of the proposed system under varying input settings is also analyzed in the following sections. Using synthetic input sequence reads sequenced from a random artificial genome with 2000 nt in length, the proposed system was independently executed five times under the default parameter settings to observe the variability in the resulting fitness measurements. The distributions of each fitness objective measurement across all individuals are plotted for each trial and shown in Figure 5-6, Figure 5-7, and Figure 5-8.

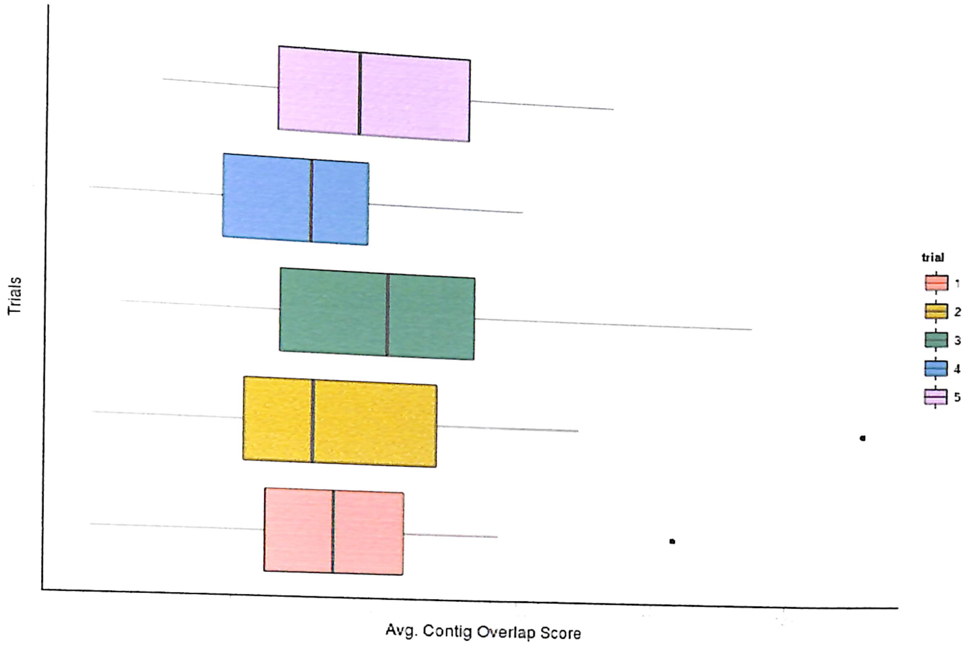


Figure 5-6: The boxplots of avg. contig overlap score fitness measurements of all individuals from five trials

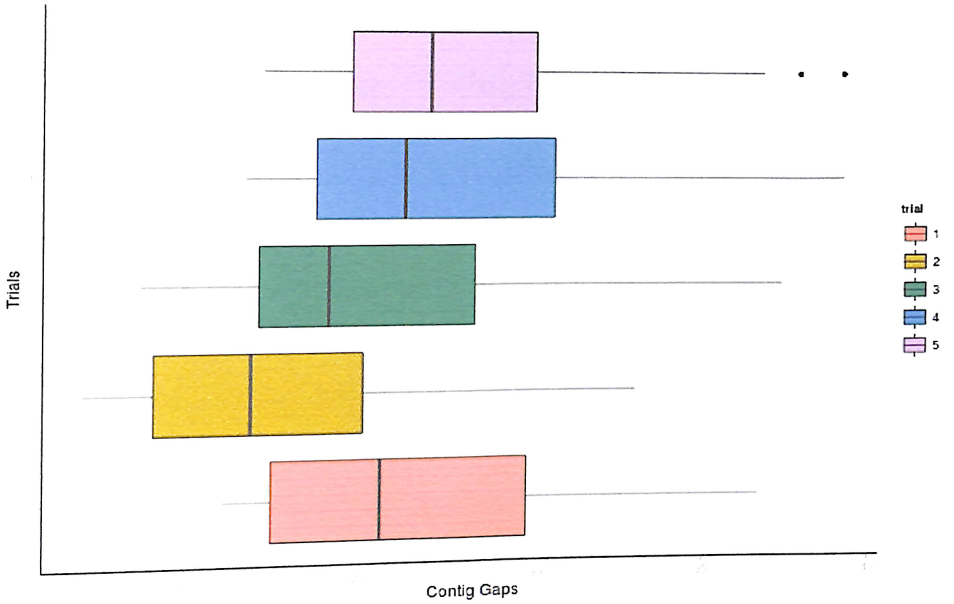


Figure 5-7: The boxplots of contig gap fitness measurements of all individuals from five trials

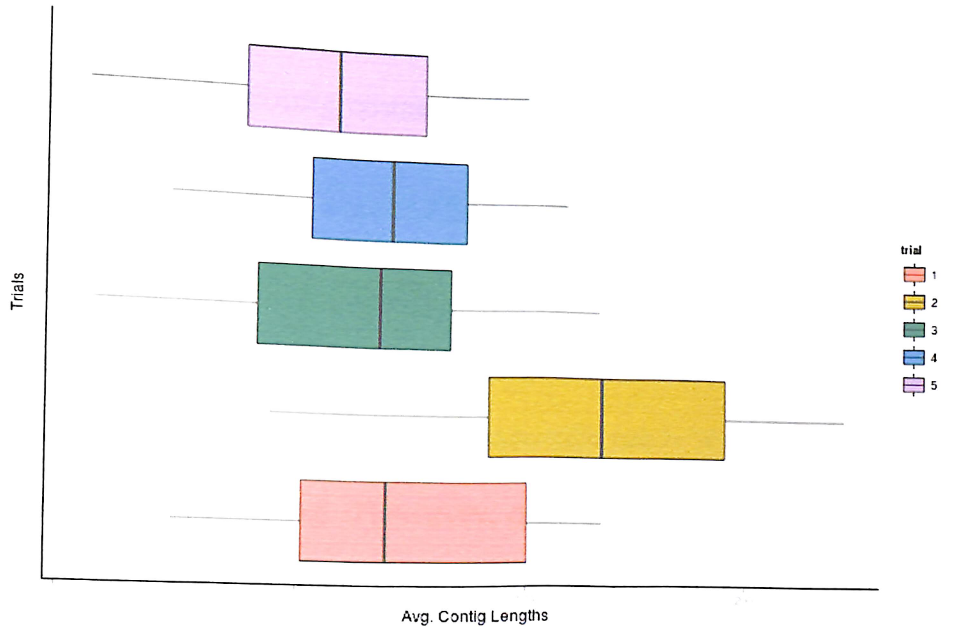


Figure 5-8: The boxplots of avg. contig length fitness measurements of all individuals from five trials

The interquartile ranges and the medians of each box plot show reasonably stable outcomes across all trials which further suggests that the outcome of the system does not occur by chance. Although the outcome from the second trial displayed population with overall superior fitness compared to other outcomes, no particular outcome appear to deviate in an abnormal amount to cast doubt on the stability of the proposed system.

.4 Assessing the performance of ACO and MOEA

The amount of sequence assembly performed by each component of the proposed ystem was quantified and compared to gain a better understanding of its role. The initial nput sequence reads for the study were prepared using a synthetic genome with 10,000 t in length. The count of the output contigs from the ACO and MOEA sequence assembly processing were observed as displayed in Table 5-4 in order to measure the percentage of reduction in the contig counts following each procedure.

Pre-ACO	Post-ACO	Post-MOEA
2000	753	$\bar{x}$: 706.51, σ: 3.38

Table 5-3: The count of initial input reads and contigs from the ACO and MOEA processing

A 62.35% reduction in the count of the initial sequence reads was observed following the ACO sequence assembly processing. However, only 6.17% reduction in the count of the ACO-generated contigs was observed following the MOEA sequence assembly processing. Further, the distributions of contig lengths resulting from the ACO and MOEA processing were observed as depicted in Figure 5-9 and Figure 5-10 and summarized in Table 5-5 and Table 5-6. The histogram in Figure 5-10 which depicts the MOEA-generated contig lengths represents the distribution of combined contig lengths of all individuals of the population.

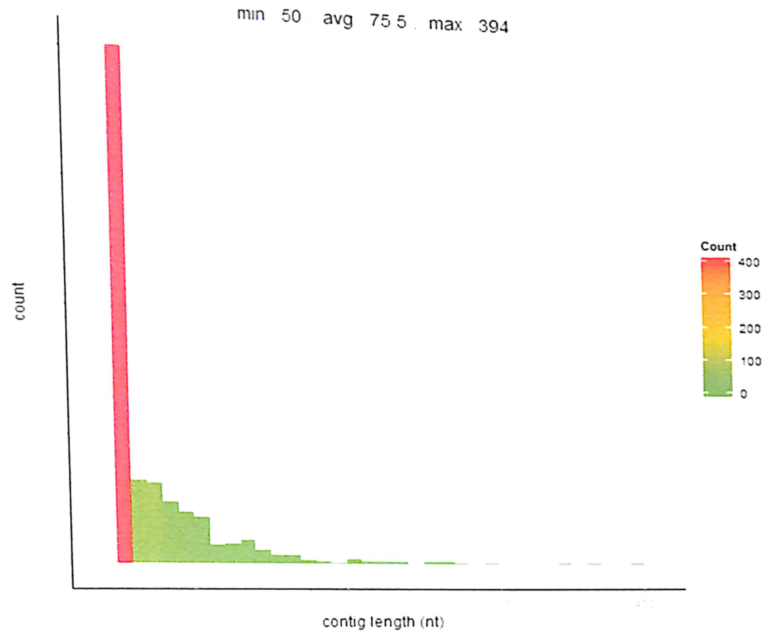


Figure 5-9: The histogram of ACO-generated contig lengths

Min	Q1	Median	Mean	Q3	Max
50.00	50.00	57.00	75.50	87.00	394.00

Table 5-4: The summary of the distribution of ACO-generated contig lengths

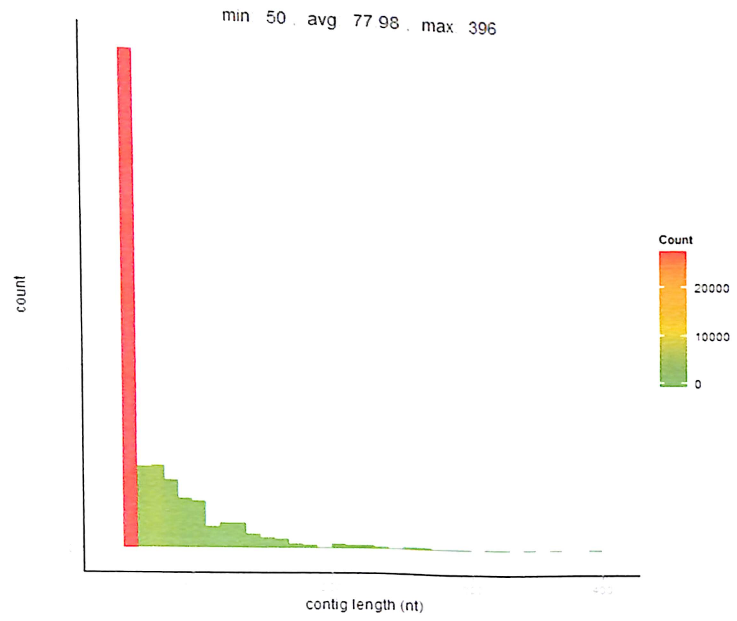


Figure 5-10: The histogram of MOEA-generated contig lengths

Min	Q1	Median	Mean	Q3	Max
50.00	50.00	58.00	77.98	90.00	396.00

Table 5-5: The summary of the distribution of MOEA-generated contig lengths

As demonstrated by the observed deficiency in the reduction of ACO-generated contig count following MOEA processing, no significant change was observed in the ACO-generated contig lengths post sequence assembly by the MOEA component. This observation was supported by the remaining high right-skewness and also the closeness in the values of the mean and median examined in both distributions. The percent change in the mean and median were 3.28% and 1.75% respectively. Importantly, the count of the initial sequence input reads remaining as unassembled throughout the assembly procedures were observed as shown in Table 5-7.

Pre-ACO	Post-ACO	Post-MOEA
2000	304	$\bar{X}$: 269.67

Table 5-6: The count of unassembled input sequence reads after ACO and MOEA processing

15.2% of the initial sequence reads remained unassembled post ACO processing while approximately 13.4% of the initial sequence reads remained unassembled across all individuals of the population from the MOEA processing.

The findings suggests that the majority of the sequence assembly was accomplished by the ACO component while the MOEA component took a role of further fine-tuning the contigs defined by the ACO assembly processing. If the MOEA component was instead used as the assembler of the initial sequence reads and the ACO component as the assembler of the MOEA-generated contigs, then it is expected that the MOEA component would handle the majority of the sequence assembly. However, reversing the role of each component would necessitate running the ACO processing on each individual solutions to preserve the benefit of having a set of multiple quasi-optimal solutions.

5.5 Evaluation of Pareto front individual solutions

The trade-off in the fitness measurements of the quasi-optimal Pareto-front solution set was observed to assess their overall diversity and degree of variability in each fitness objective. The size of the initial input sequence reads for the study was 2,000, sequenced based on a synthetic genome with length of 10,000 nt. Upon completion of the sequence assembly processing, 76 Pareto-front individuals containing unique set of contigs were obtained. Based on these individual solutions, the diversity and degree of variability in the individual fitness along each objective were analyzed through scatter plots and box plots respectively as shown in Figure 5-11, Figure 5-12, and Figure 5-13 and summarized in Table 5-8, Table 5-9, and Table 5-10.

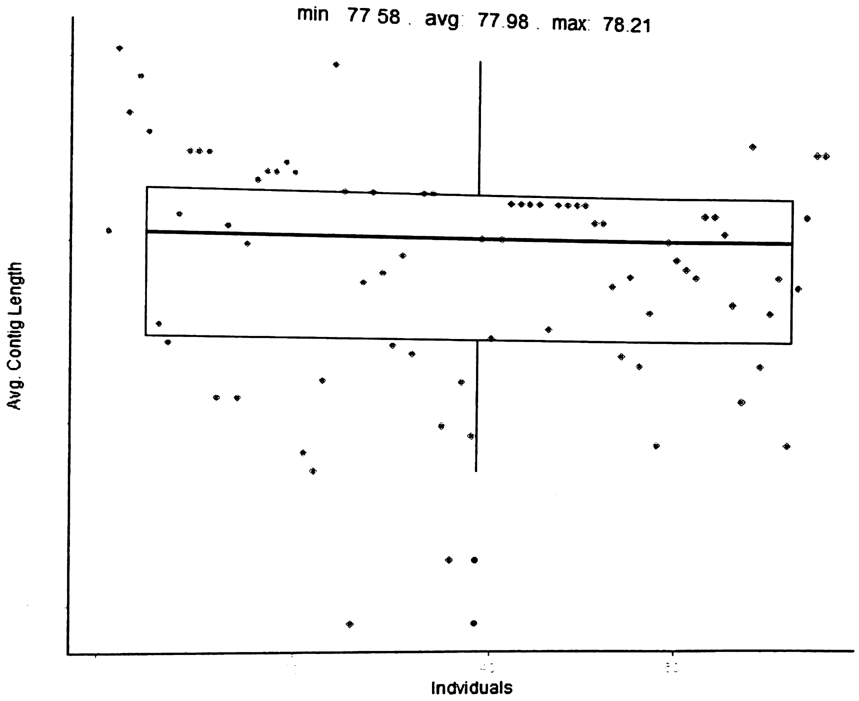


Figure 5-11: The scatter plot and box plot of avg. contig length fitness measurements of all Pareto-front individual solutions

Min	Q1	Median	Mean	Q3	Max
77.58	77.90	78.01	77.98	78.06	78.21

Table 5-7: The summary of the distribution of avg. contig length of each Pareto-front individual solution

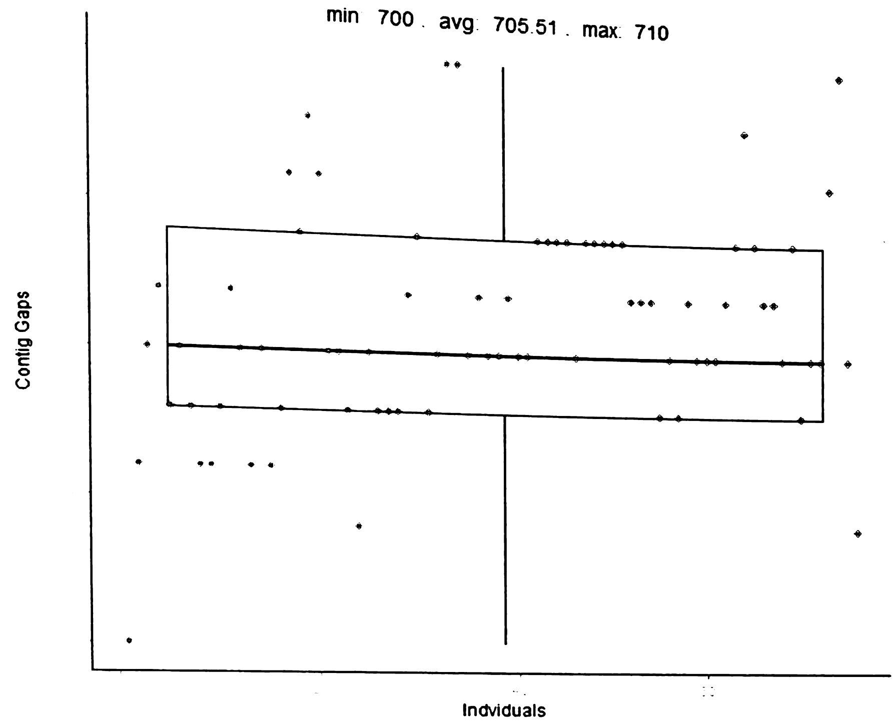


Figure 5-12: The scatter plot and box plot of contig gap fitness measurements of all Pareto-front individual solutions

Min	Q1	Median	Mean	Q3	Max
700.00	704.00	705.00	705.50	707.00	710.00

Table 5-8: The summary of the distribution of contig gaps of each Pareto-front individual solution

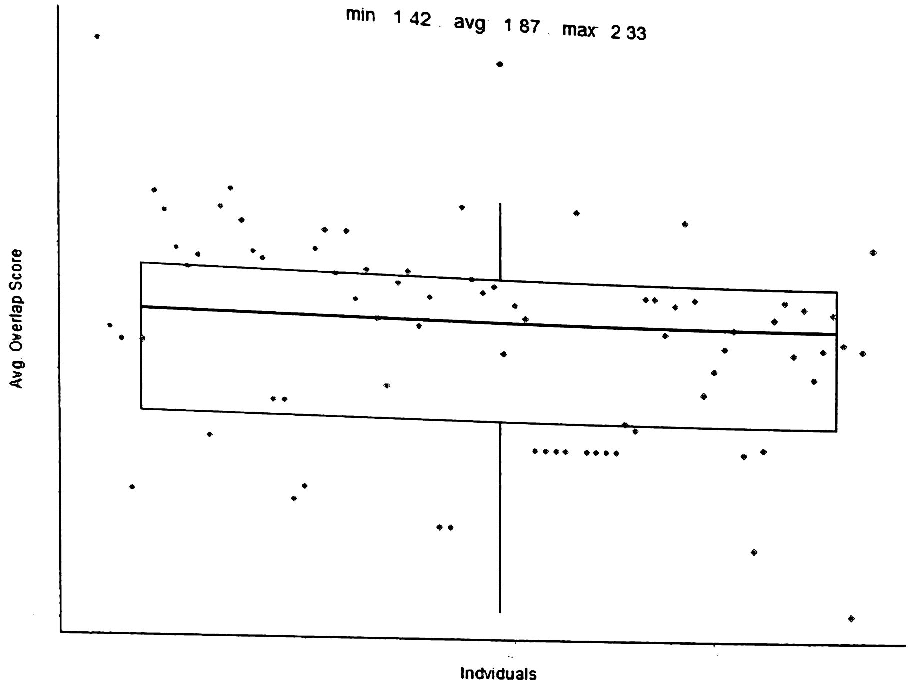


Figure 5-13: The scatter plot and box plot of avg. overlap score fitness measurements of all Pareto-front individual solutions

Min	Q1	Median	Mean	Q3	Max
1.420	1.737	1.900	1.870	1.973	2.330

Table 5-9: The summary of the distribution of avg. overlap score of each Pareto-front individual solution

Diverse individual fitness measurements were present within a small range in each fitness objective; there were no individual solution pairs with an identical fitness or contigs. The range measurements of 0.63, 10.00, and 0.91 were observed for the fitness objective measurements of average contig length, contig gap, and average contig overlap score, respectively. As observed in the section 5.4, the majority of the sequence assembly is handled by the ACO component, leaving a small range of improvement to take place during the execution of the MOEA component. The finding further suggested some potential modification in the default control parameter setting of MOEA component could widen the range of possible improvement along each fitness objective. Decreasing the minimum overlap constraint for assembling ACO-generated contigs in MOEA processing can increase the number of alternative formation of contigs which in turn will likely increase the overall fitness measurement along each fitness objective. In addition, increasing the mutation rate to increase the level of stochasticity in the solution exploration can leverage the rate of fitness improvement while keeping the benefit of the edge recombination genetic operator.

5.6 Evaluation of output behaviors under varying input settings

The system output behavior under different input sequence settings were examined to further characterize the strengths and weaknesses of the proposed method. The varied input settings for the study were: 1) genome size, 2) read length, 3) read noise level, and 4) sequencing coverage. The observed output behavior was analyzed by the fitness measurements along each fitness objective. The current version of the proposed system was unable to produce outputs under some input settings due to insufficient memory or its lack of efficient memory management; hence, output measurements under a diverse range of input settings were not possible for a better comprehensible characterization of the output behavior.

5.6.1 Impact of genome size variation

With the default sequencing control parameter setting unchanged, varying the size of the genome has a direct correlation with the number of sequenced input reads. Hence, the size of the genome also has a direct correlation with the size of the solution search space or computational complexity. Using genome sizes of 1 knt, 5 knt, 10 knt, and 15 knt, initial sequence reads with read counts of 200, 1000, 2000, and 3000 were generated. The observed influence of initial input reads with varying read counts on the fitness of output solutions is shown in Figure 5-14, Figure 5-15, and Figure 5-16.

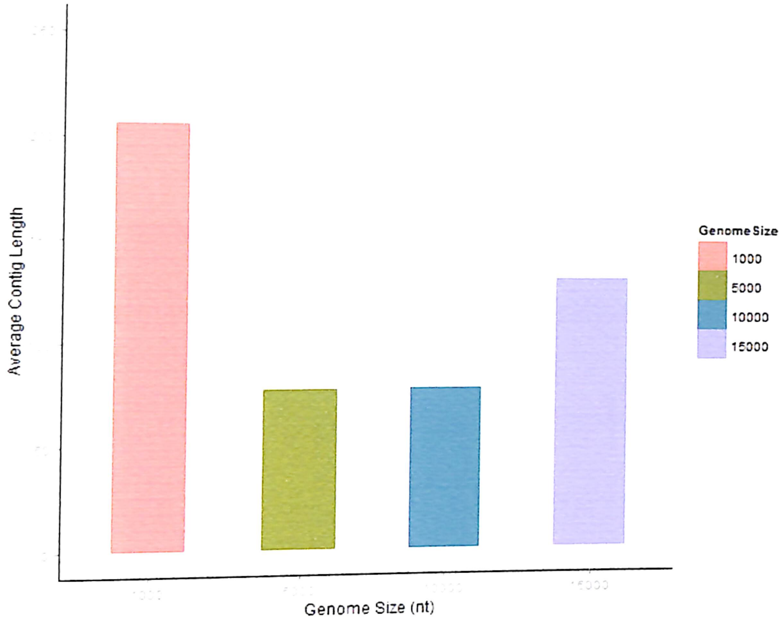


Figure 5-14: The bar chart illustrating the impact of genome size variation on the avg. contig length fitness measurement

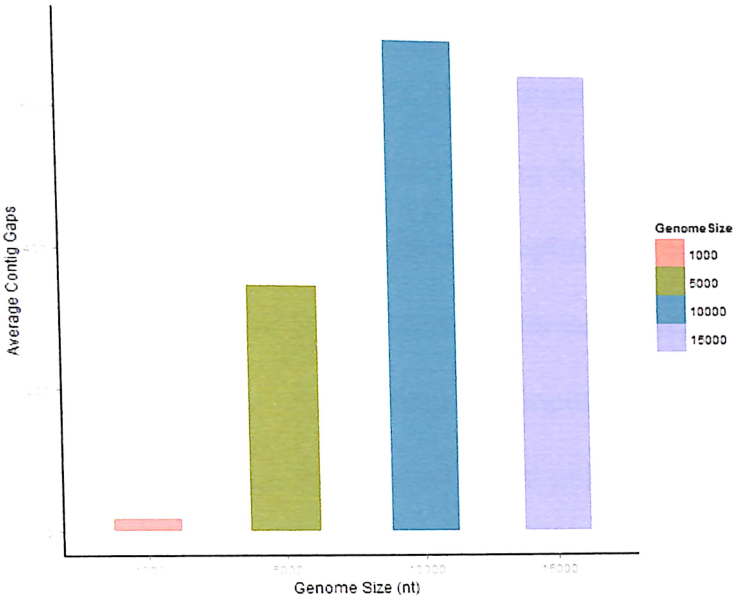


Figure 5-15: The bar chart illustrating the impact of genome size variation on the avg. contig gap fitness measurement

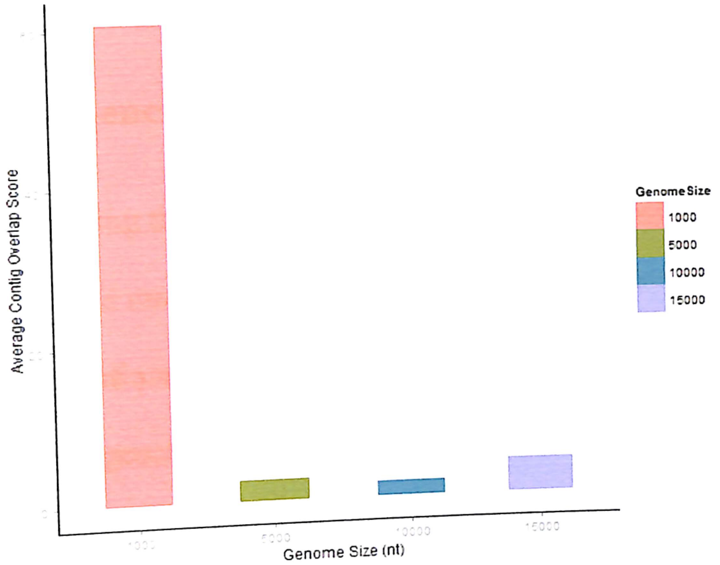


Figure 5-16: The bar chart illustrating the impact of genome size variation on the avg. contig overlap score fitness measurement

Distinctively superior solution fitness was observed in the output of 1 knt genome size while the other outputs presented similar fitness measurements. The finding suggests that either the default control parameter setting did not provide sufficient coverage for finding the Hamiltonian path with quasi-optimal fitness for genome size exceeding 1 knt or the proposed method lacks computational efficiency as shown by the sharp drop in solution fitness with a slight increase in input size. In addition, the proposed system was tested with genome size of 20 knt or input read counts of 4000. Upon several trials, the processing was constantly interrupted due to insufficient memory while constructing the adjacency matrix of the overlap scores based on the initial input reads. This suggests a relative computational inefficiency of the current version of the proposed system.

5.6.2 Impact of sequencing coverage level variation

Similarly to varying the genome size, varying the sequencing coverage level has a direct correlation with the number of sequenced input reads; therefore, it also has a direct correlation with computational complexity. However, varying the sequencing coverage level based on the same genome varies the number of sequenced reads that cover each base present in the genome. In other words, increasing the sequencing coverage level will increase the number of "hints" given to solve the sequence assembly problem and vice versa. Using an artificial genome with 10 kbp in size, a set of initial input reads were generated based on sequencing coverage level of 1X, 5X, 10X, and 15X. The read counts of the sequenced reads were 200, 1000, 2000, and 3000. As discussed in the previous section, the proposed system was not able to process through when the sequencing coverage level was 20X (4000 reads) due to insufficient memory. The fitness of output solutions from each sequencing coverage level is shown in Figure 5-17, Figure 5-18, and Figure 5-19.

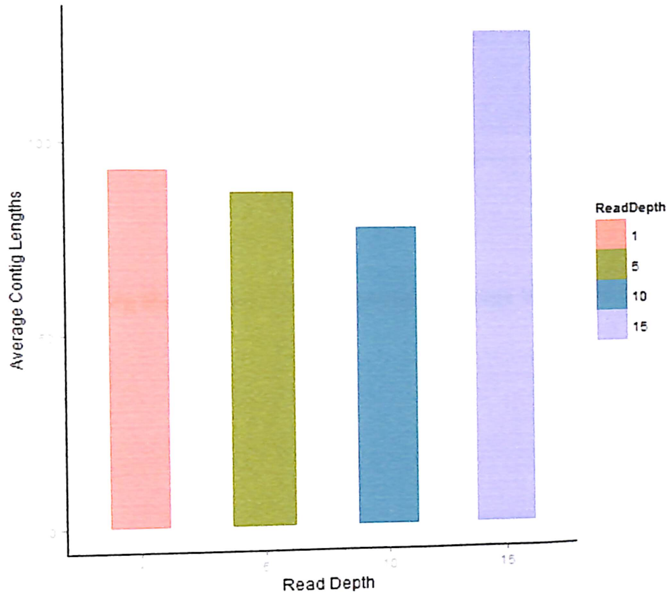


Figure 5-17: The bar chart illustrating the impact of sequencing coverage variation on the avg. contig length fitness measurement

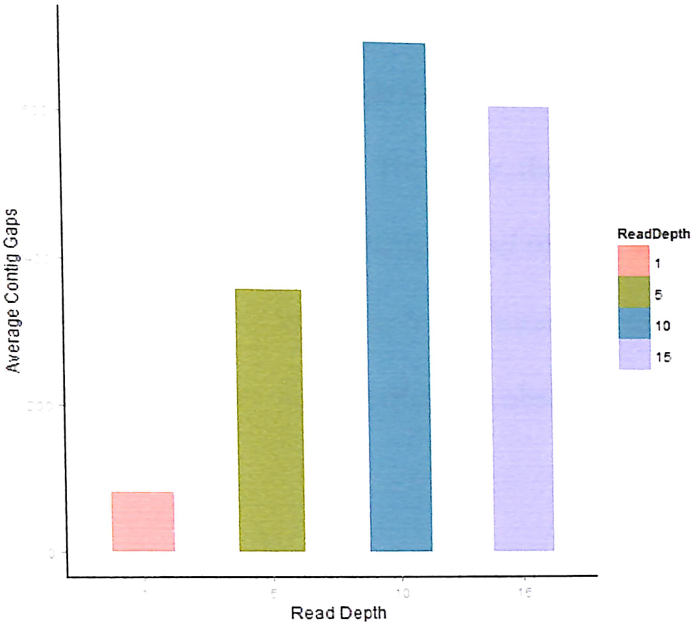


Figure 5-18: The bar chart illustrating the impact of sequencing coverage variation on the avg. contig gap fitness measurement

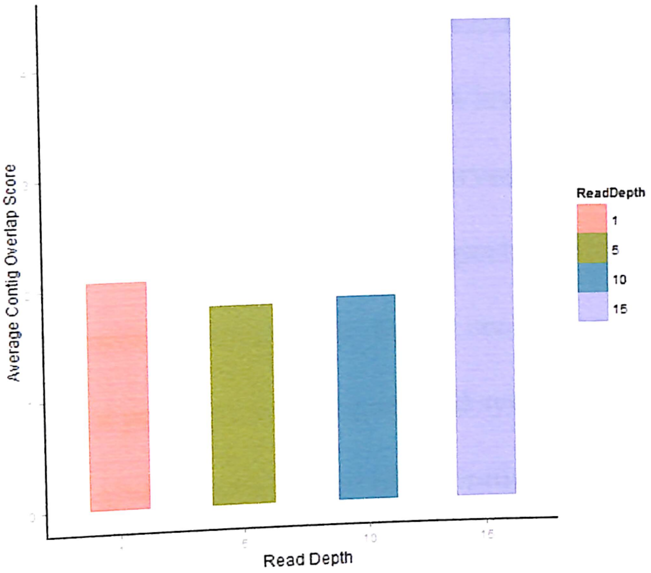


Figure 5-19: The bar chart illustrating the impact of sequence coverage variation on the avg. contig overlap score fitness measurement

The fitness measurements tend to become worse as the sequencing coverage level increases due to the increased number of reads that need to be assembled while the inefficiency to handle large dataset remains. However, the fitness measurements observed in the output of the sequencing coverage level of 15X were significantly superior than it was expected to be. The same input sequence reads sequenced with 15X coverage level was reassembled to assess whether the observed outcome could have been produced by chance.

Fitness Objective	H_0	Trial 1	Trial 2	Trial 3	$\bar{x}$	p-Value
Avg. Contig Length	128.93	128.45	129.35	128.70	128.83	0.753
Avg. Contig Gaps	620.63	619.04	621.42	622.40	620.95	0.777
Avg. Contig Overlap Score	4.40	4.29	4.35	4.49	4.37	0.732

Table 5-10: The reassessment of the output of the sequence assembly with read coverage level of 15X

With the questioned fitness measurements set as the values of the null hypotheses, two-tailed one sample t-tests were conducted based on 5% level of significance as displayed in Table 5-11. As indicated by the p-values, the observed t-statistic from each mean of sample fitness measurements failed to reject their respective null hypotheses. Thus, it is likely that the observed fitness measurements did not occur by a random chance. Possibly, there could exist a point where the proposed system can overcome the issue of computational complexity given a certain amount of initial sequence reads with a specific amount of computing resource. Generally, given a higher sequencing coverage level, an ideal sequence assembly system should produce less contigs with greater lengths that represent larger proportions of its original genome. In order to reach this level of processing efficiency, the findings suggest that the current version of the proposed method necessitate a better memory management and data structures.

5.6.3 Impact of sequence read length variation

With a genome size and a sequencing coverage level held constant, varying the read length has a direct correlation to the complexity of the sequence assembly problem. Decreasing the read length means more reads are required to satisfy a sequencing coverage level and vice versa. Since shorter read length generates more reads with shorter genuine genetic information, the complexity of the sequence assembly is expected to increase as the read length decreases. Using a genome size of 10 knt and read coverage level of 10X, sets of reads were obtained with varying read lengths of 50 nt, 75 nt, 100 nt, and 150 nt. Then, the outputs produced from each input sequence read set were observed as displayed in Figure 5-20, Figure 5-21, and Figure 5-22.

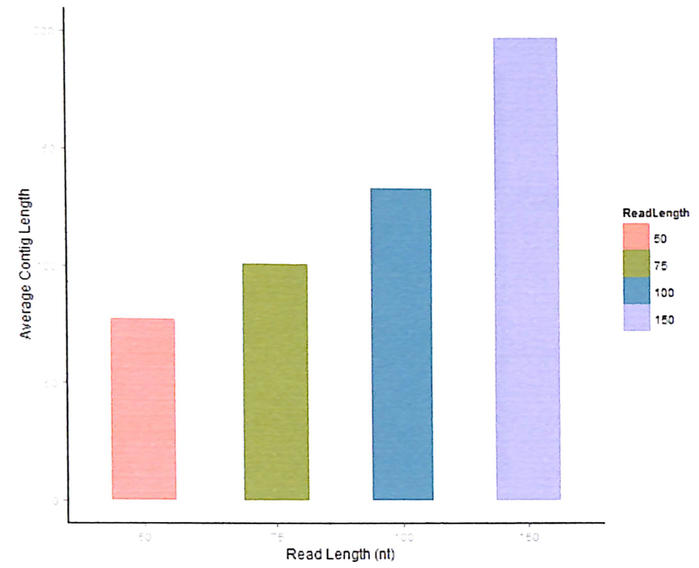


Figure 5-20: The bar chart illustrating the impact of sequence read length variation on the avg. contig length fitness measurement

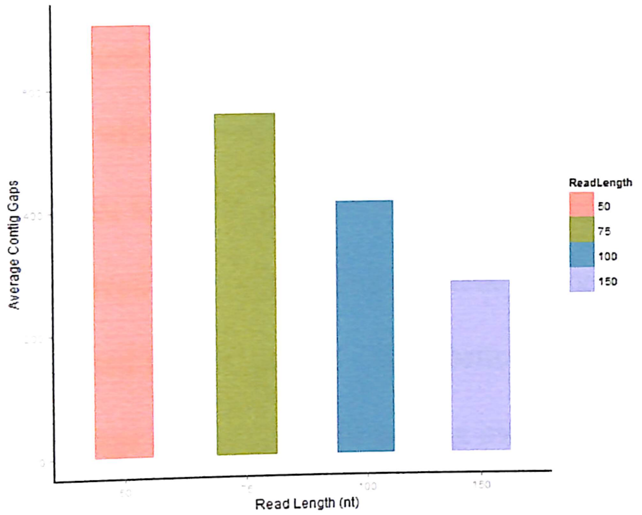


Figure 5-21: The bar chart illustrating the impact of sequence read length variation on the avg. contig gap fitness measurement

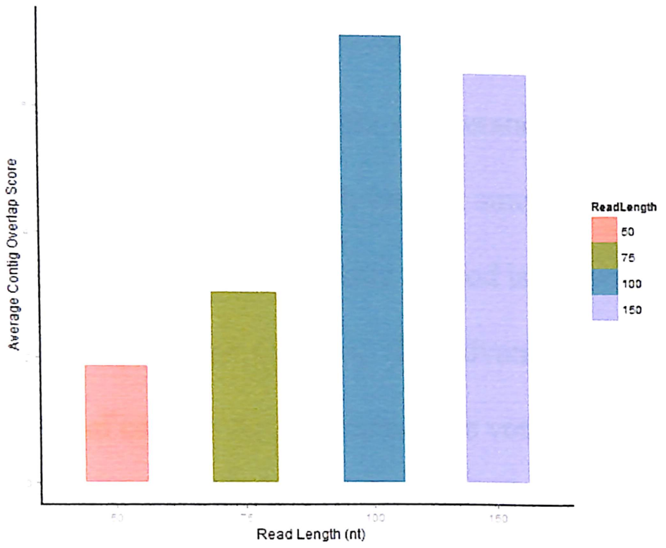


Figure 5-22: The bar chart illustrating the impact of sequence read length variation on the avg. contig overlap score fitness measurement

The read lengths exhibited expected effect on the outputs. With increase in read length, better solution fitness measurements were observed along each fitness objective. Starting out the sequence assembly with a set of longer input reads gives advantage of having a smaller solution search space to be exploited due to less number of reads; consequently, the proposed system was given more computing efficiency to process the initial sequence read which innately possessed higher quality. Thus, better fitness measurements of average contig length, average contig gap, and average overlap score were observed. In addition, a longer input read length results in a higher minimum overlap score requirement for contig construction but also a higher number of bases that can be overlapped, contributing to the observed effect on the average contig overlap score fitness.

5.6.4 Impact of read error variation

The proposed method was evaluated for its tolerance to erroneously sequenced reads using sets of initial sequence reads with varying amounts of read error. A sequence assembly based on the overlap layout consensus method is known to have more tolerance to read error than the de Brujin graph method. The advantage of read error tolerance comes from construction of contigs through consensus voting for each base of a contig from quasi-optimally aligned sequences. On the other hand, the construction of contigs in the de Brujin graph method is based on taking walks on a de Brujin graph built with a fixed kmer length; since erroneous reads can easily introduce erroneous paths in a de Bruijn graph, it can easily lead to a weak tolerance read error. Sets of input sequence reads with read error amounts of 1.5%, 5%, 10%, and 20% were obtained by randomly mutating a specified amount of bases in a set of reads obtained from sequencing a random genome with 10 knt in size. The percentage of random mutation refers to the percentage of the total sum of all read length to be mutated into a base other than their current bases. The observed outcomes from the varied read error amounts are displayed in Figure 5-23, Figure 5-24, and Figure 5-25.

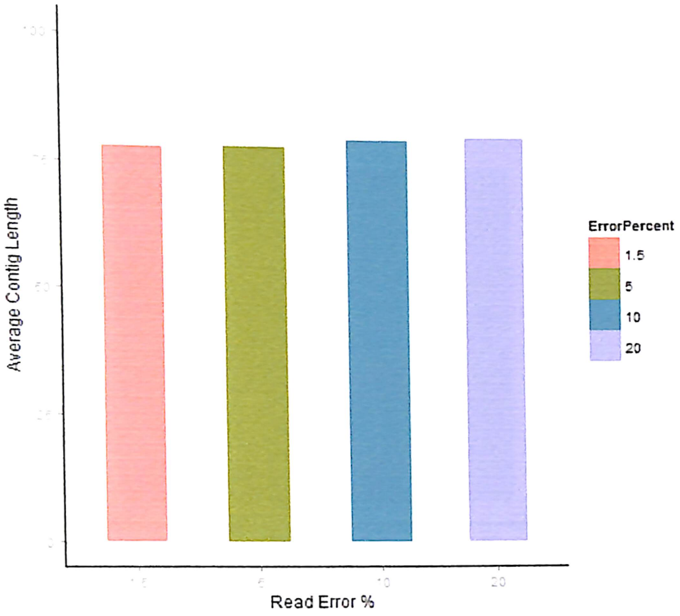


Figure 5-23: The bar chart illustrating the impact of read error amount variation on the average contig length fitness measurement

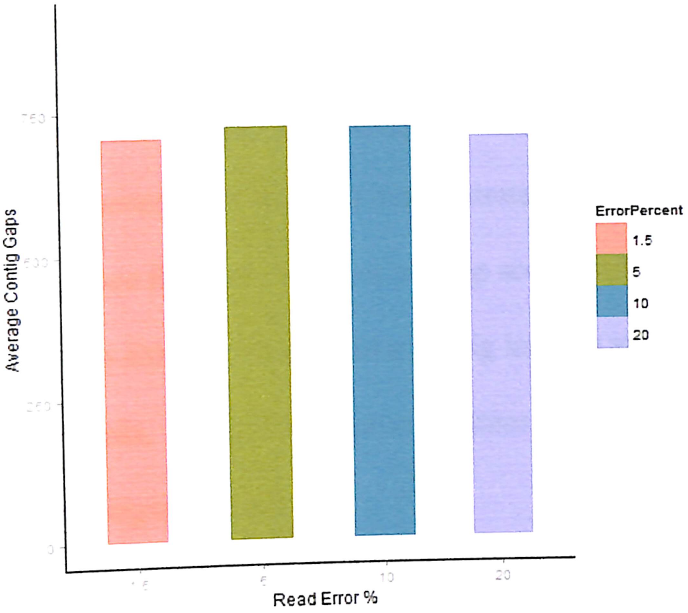


Figure 5-24: The bar chart illustrating the impact of read error amount variation on the average contig gap fitness measurement

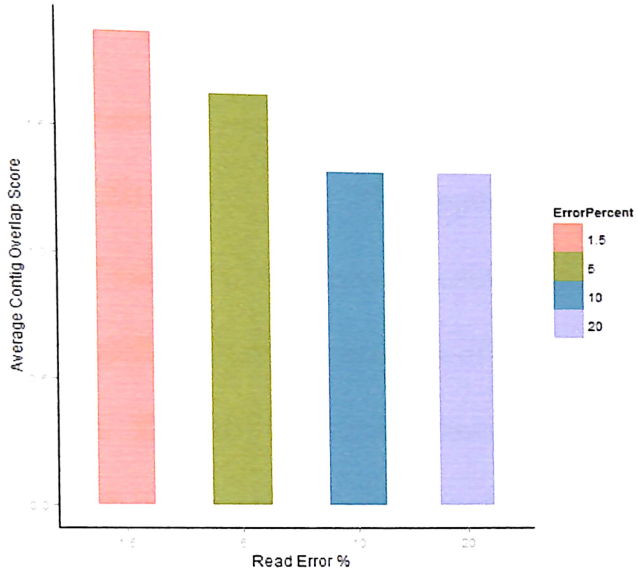


Figure 5-25: The bar chart illustrating the impact of read error amount variation on the average contig overlap score fitness measurement

Similar fitness measurements of average contig length and average contig gaps were observed while the fitness measurement of average contig overlap score decreased with increasing amount of sequence read error. It demonstrates that the proposed system was inevitably unable to produce the same level of overlap score as the amount of read error increased; however, it was able to retain similar contig lengths and contig gaps with all specified read error amounts. Thus, the proposed system can be considered to have a high tolerance to sequencing read error.

5.7 Evaluating the quality of the assembled contigs

The quality of the assembled contigs by the proposed sequence assembly method was measured using the NG50 statistic [41]. The lengths of assembled contigs can be used to assess the degree of success of sequence assembly based on the assumption that a longer assembled contig signifies a larger proportion of a genome that is recovered from a set of sequence fragments. The NG50 statistic is based on its predecessor N50 statistic that measures a contig length where sum of contig length equal to or greater than a particular contig length represents 50 percent of the total sum of contig lengths. The N50 statistic can be significantly misleading due to varying contig length distribution in each sequence assembly. Instead, the NG50 statistic measures the weighted median contig length based on a known genome length or estimated genome length for better representation of true quality of the assembled sequences. The averaged NG50 measurement of all individual outcomes from the proposed sequence assembly method is compared to the NG50 measurement of ACO-generated contigs as shown in Figure 5-26. The summarized NG50 distribution of all individual outcomes is shown in Table 5-12.

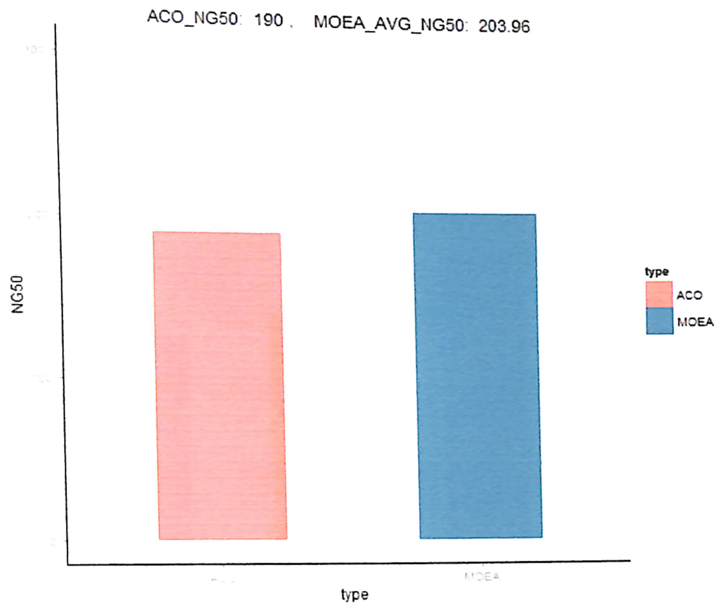


Figure 5-26: The NG50 measurements of ACO-generated contigs and MOEA-generated contigs

Min	Q1	Median	Mean	Q3	Max
190	204	204	203.96	206	211

Table 5-11: The summary of NG50 measurements from each Pareto-front individual solution

As expected, NG50s from both components were quite similar; the averaged NG50 of individual contig sets from the MOEA component were higher only by 7.34%.

The averaged NG50 statistic can be dramatically improved by taking advantage of the diverse individual solution set resulting from MOEA. An individual with superior NG50 statistic can be built by selecting only the high quality contigs from each individual while keeping the advantage of solution diversity. For instance, an individual solution with superior NG50 was built by selecting contigs with minimum length of 200 nt from each individual. The improved NG50 measurement of the individual is displayed in Figure 5-27.

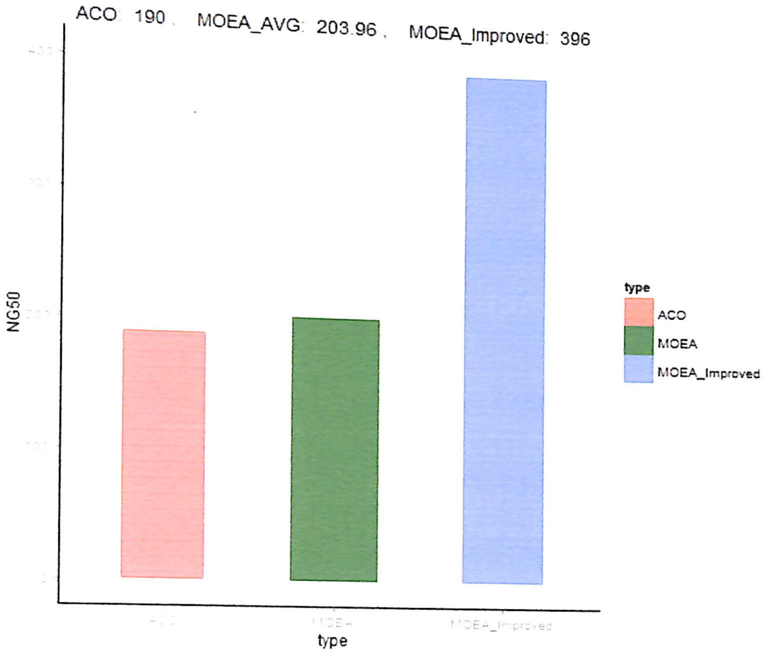


Figure 5: The NG50 measurement of a selectively created 'elite' individual solution; (min. contig length: 200 nt)

The observed NG50 of the selectively built individual solution were approximately two times greater than the NG50 of contigs from both ACO and MOEA. The measured improvements in NG50 were 108.42% and 94.16% for ACO and MOEA respectively.

5.8 Comparing the quality performance to DBG sequence assembly methods

A comparative study was conducted to determine whether the proposed method contributes any significant improvement to a current OLC method based on its quality performance comparison to modern DBG methods. In Assemblathon 2 [41], a recent

OLC-based assembler, SGA (String Graph Assembler) [49], was compared to various DBG-based sequence assemblers for quality performance comparison. The sets of input sequence reads involved in the study were prepared from sequencing full genomes of Bird (Melopsittacus Undulates), Fish (Maylandia Zebra), and Snake (Boa Constrictor); their estimated genome sizes are 1.2 Gbp, 1.0 Gbp, and 1.6 Gbp respectively. Due to the lack of processing inefficiency found in the current version of the proposed system, testing with the same sets of input sequence reads used in the study was infeasible. Instead, a similar comparative study was conducted using sets of input sequences prepared from sequencing sets of 10 knt sized sequences that were randomly sampled from common bean (Phaseolus Vulgaris) chromosomes [48]. Using the prepared sets of input sequences, the outcomes of sequence assemblies by the proposed method and two widely used sequence assembly software, Velvet [8] and Trinity [9], were analyzed as displayed in Table 5-13, Table 5-14, Table 5-15, and Table 5-16.

Quality Measure	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**
Avg. Contig Length	$\bar{x}$: 270.11, σ : 183.72	$\bar{x}$: 263.59, σ : 259.46	$\bar{x}$: 199.35, σ : 153.78	$\bar{x}$: 311.90, σ : 206.25	$\bar{x}$: 321.16, σ : 272.06
NG50	423	359	244	457	407
Contig Gaps	37	36	50	32	30

Table 5-12: The quality measurements for each set of Velvet-assembled contigs resulting from the assembly of the five sets of the common bean input sequence reads

Quality Measure	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Avg. Contig Length	$\bar{x}$: 1187.13, σ : 1021.20	$\bar{x}$: 978.33, σ : 633.60	$\bar{x}$: 784.91, σ : 533.59	$\bar{x}$: 874.40, σ : 498.18	$\bar{x}$: 1055.78, σ : 695.02
NG50	1905	1283	828	884	1504
Contig Gaps	7	8	10	9	8

Table 5-13: The quality measurements for each set of Trinity-assembled contigs resulting from the assembly of the five sets of the common bean input sequence reads

Quality Measure	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Avg. Contig Length	$\bar{x}$: 76.93, σ : 38.83	$\bar{x}$: 77.60, σ : 36.83	$\bar{x}$: 74.82, σ : 39.85	$\bar{x}$: 77.18, σ : 39.47	$\bar{x}$: 76.55, σ : 42.41
Avg. Ind. NG50	$\bar{x}$: 180.87, σ : 6.58	$\bar{x}$: 173.21, σ : 4.01	$\bar{x}$: 173.08, σ : 3.78	$\bar{x}$: 165.37, σ : 1.82	$\bar{x}$: 175.25, σ : 2.86
Avg. Contig Gaps	$\bar{x}$: 684.94, σ : 2.42	$\bar{x}$: 693.02, σ : 2.06	$\bar{x}$: 695.66, σ : 2.17	$\bar{x}$: 687.05, σ : 2.01	$\bar{x}$: 695.87, σ : 2.17

Table 5-14: The quality measurements for each set of contigs assembled by the proposed method using the five sets of the common bean input sequence reads

Quality Measure	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Avg. Contig Length	$\bar{x}$: 265.52, σ : 66.43	$\bar{x}$: 267.72, σ : 59.38	$\bar{x}$: 260.97, σ : 41.32	$\bar{x}$: 246.06, σ : 45.01	$\bar{x}$: 253.52, σ : 57.71
Improved NG50	302	271	259	230	285
Contig Gaps	52	35	38	31	63

Table 5-15: The quality measurements of a set of contigs selected from the set of Pareto-front contig set based on minimum contig length of 200 nt

The outcome of sequence assembly with Trinity was by far superior to both outcomes of sequence assembly with Velvet and the proposed method. Overall, the proposed method had the least favorable outcome. Interestingly, although the outcome of the proposed method was almost twofold worse than the outcome of the Velvet assembly in all measurements considered, the outcome based on combined high quality contigs from each individual solution was highly similar and competitive to the outcome of the Velvet assembly.

In order to determine whether the proposed method made any improvement as compared to a current OLC-based assembly method, a hypothesis test was conducted based on the observed performance comparison between OLC-based assembly and DBG-based assembly from the experiment above and the experiment from Assemblathon 2. Based on the result from Assemblathon 2, the ratio between NG50 of the best NG50 of a DBG-based assembly and NG50 of SGA assembly was taken used as the mean value assumed for the distribution assumed under the null hypothesis.

$$R_0 = \frac{NG50_{topDBG}}{NG50_{SGA}}$$

The measured ratio was approximately 6.5, indicating that the top NG50 from a DBG-based assembly was approximately 6.5 times higher than the NG50 from SGA assembly. Similarly, the following ratios were taken for observed outcomes of each sample.

$$R_1 = \frac{NG50_{Velvet}}{NG50_{proposed}}$$

$$R_2 = \frac{NG50_{Trinity}}{NG50_{proposed}}$$

Then, one-tailed t-tests with 5% level of significance were performed to assess whether the mean values of the ratio taken as R_1 and R_2 from each sample have a significant difference to the ratio value of R_0 with the following null and alternative hypotheses.

$$H_0\colon\ \mu = 6.5$$

$$H_1\colon\ \mu < 6.5$$

Note that the degree of decrement in R_1 and R_2 relative to R_0 is directly correlated to the degree of improvement in NG50 measurement of the proposed method relative to SGA. In other words, a smaller R_1 or R_2 ratio value relative to 6.5 indicates improvement in the NG50 measurement. The results of each t-test are display in Table 5-17, Table 5-18, Table 5-19, and Table 5-20.

$\bar{x}$	σ	se	**95% confidence interval**	t	*p-value*	**decision**
2.181	0.498	0.223	$1.563 < u < 2.799$	-19.40	$2.079e-5$	reject H_0

Table 5-16: The t-test for measuring the statistical significance of the ratio of NG50$_{\text{Velvet}}$/ NG50$_{\text{proposed}}$

$\bar{x}$	σ	se	**95% confidence interval**	t	*p-value*	**decision**
7.330	2.358	1.055	$4.402 < u < 10.258$	0.787	0.762	**fail to reject H_0**

Table 5-17: The t-test for measuring the statistical significance of the ratio of NG50$_{\text{Trinity}}$/ NG50$_{\text{proposed}}$

$\bar{x}$	σ	se	**95% confidence interval**	t	*p-value*	**decision**
1.416	0.374	0.167	$0.952 < u < 1.880$	-30.399	$3.488e-6$	reject H_0

Table 5-18: The t-test for measuring the statistical significance of the ratio of NG50$_{\text{Velvet}}$/ improved-NG50$_{\text{proposed}}$

$\bar{x}$	σ	se	**95% confidence interval**	t	*p-value*	**decision**
4.672	1.215	0.543	$3.163 < u < 6.181$	-3.363	0.014	**reject H_0**

Table 5-19: The t-test for measuring the statistical significance of the ratio of NG50$_{\text{Trinity}}$/ improved-NG50$_{\text{proposed}}$

A significant level of improvement was observed when the output of the proposed method was compared to the output of Velvet assembly. However, there was a strong evidence that no significant improvement was made when the outcome of the proposed method was compare to the outcome of Trinity assembly. Yet, when the output constructed from combining high quality contigs from each individual solution was compared to the output of Trinity assembly, there was a weak evidence suggesting that a significant improvement was made. In addition, a two-sample t-test with 5% level of significance was performed as shown in Table 5-21 to test a hypothesis claiming there exists a significant difference between the averaged NG50s of each individual solution and the NG50 of selectively constructed contig set. The null hypothesis and the alternative hypothesis are as follows.

$$H_0: \mu_d = 0$$

$$H_1: \mu_{pre-improvement} \neq \mu_{post-improvement}$$

$\overline{x}_{improved}$	$\overline{x}_{default}$	**95% confidence interval**	t	***p-value***	**decision**
269.400	173.556	$62.338 < u < 129.350$	7.707	0.001	Reject H_0

Table 5-20: The two-sample t-test for testing the significance of improved-NG50 relative to the averaged NG50

The t-statistic value of 7.707 indicates that the improved-NG50 deviates far from the mean of the distribution that assumes no difference between the two NG50s. Also, the 95% confidence interval suggests that the true difference between the two NG50s will lie between 62.338 and 129.350 for 95% of the times if comparisons are made with different samples. Thus, utilizing high quality contigs from each resulting Pareto-front solution can bring a significant improvement in overall solution fitness.

5.9 Assessing the accuracy of assembled contigs with NCBI-BLAST

Although a longer contig suggest that a bigger portion of the original genome was recovered through sequence assembly, it does not elucidate its relationship to the accuracy of its sequence read alignments. Thus, the NG50 statistic which is solely based on contig lengths does not provide any information on how accurately contigs were assembled. NCBI BLAST (Basic Local Alignment Search Tool) [50] is a widely used tool that searches for potential sequence identities exiting in an assembled contig by comparing portions of the contig sequence (query sequences) to databases of known sequences (subject sequences). Each identified query sequence comes with an 'identities' score that counts the number of identical base-to-base matches between the query sequence and a subject sequence. The percentage of identical match can be obtained by dividing an identities score with the length of the query sequence. In order to assess the accuracy of the contigs assembled by the proposed system, sets of reads sequenced based on a 10kbp sequence sampled from known genomes (Influenza Virus [47], Common Bean [48], E-coli [46]) were assembled by the proposed method, followed by sequence identification with NCBI-BLAST; then, the identities scores of correctly identified BLAST hits were examined. For comparison, the same procedure was followed with Velvet and Trinity assemblers. In addition, usefulness of the contigs from each assembly method was estimated with the count of correctly identified BLAST hits and the total count of BLAST hits. The observed outcomes are shown in Table 5-22, Table 5-23, and Table 5-24.

Sequence Assembler	Identities Score	Identical Match %	Correct Hit Counts	Total Hit Counts	Num of Contigs	Contig Length
Velvet	$\bar{x}$: 80.21, σ : 48.70	$\bar{x}$: 99.12%, σ : 1.34%	135	140	7	$\bar{x}$: 242.42, σ : 143.72
Trinity	$\bar{x}$: 283.55, σ : 35.55	$\bar{x}$: 99.55%, σ : 0.49%	40	40	2	$\bar{x}$: 857.00, σ : 111.00
Proposed	$\bar{x}$: 56.80, σ : 20.88	$\bar{x}$: 84.68%, σ : 17.40%	740	740	37	$\bar{x}$: 220.86, σ : 20.74

Table 5-21: The estimated measurements of the accuracy and usefulness of the contig sets assembled by Velvet, Trinity, and proposed method using sets of input sequence reads from sampled sequence of Influenza Virus Type A genome

Sequence Assembler	Identities Score	Identical Match %	Correct Hit Counts	Total Hit Counts	Num of Contigs	Contig Length
Velvet	$\bar{x}$: 76.88, σ : 48.87	$\bar{x}$: 97.60%, σ : 8.47%	555	737	37	$\bar{x}$: 273.92, σ : 148.08
Trinity	$\bar{x}$: 0, σ : 0	$\bar{x}$: 0%, σ : 0%	0	0	9	$\bar{x}$: 1127.22, σ : 921.68
Proposed	$\bar{x}$: 35.35, σ : 13.75	$\bar{x}$: 88.16%, σ : 11.21%	694	1253	96	$\bar{x}$: 216.75, σ : 69.28

Table 5-22: The estimated measurements of the accuracy and usefulness of the contig sets assembled by Velvet, Trinity, and proposed method using sets of input sequence reads from sampled sequence of Escherichia Coli genome

Sequence Assembler	Identities Score	Identical Match %	Correct Hit Counts	Total Hit Counts	Num of Contigs	Contig Length
Velvet	$\bar{x}$: 39.06, σ : 27.08	$\bar{x}$: 76.11%, σ : 13.53%	27	154	39	$\bar{x}$: 267.38, σ : 182.13
Trinity	$\bar{x}$: 45.14, σ : 63.25	$\bar{x}$: 72.07%, σ : 12.36%	25	63	9	$\bar{x}$: 1100.89, σ : 993.21
Proposed	$\bar{x}$: 35.78, σ : 15.78	$\bar{x}$: 79.50%, σ : 11.56%	25	151	54	$\bar{x}$: 265.40, σ : 65.82

Table 5-23: The estimated measurements of the accuracy and usefulness of the contig sets assembled by Velvet, Trinity, and proposed method using sets of input sequence reads from sampled sequence of Phaseolus Vulgaris genome

Most interestingly, although Trinity-assembled e-coli contigs displayed superior contig length (NG50) by far compared to e-coli contigs assembled by the proposed method and Velvet assembler, none of its contigs were identified by BLAST. Similar results were observed with no BLAST hits when e-coli sample sequence reads were reassembled multiple times with Trinity assembler. However, the Trinity-assembled Influenza-A contigs displayed the highest contig length with the highest identities score. Also, Trinity-assembled P.Vulgaris-contigs displayed a much higher contig length than the others; yet, its identities score was only slightly higher than the identities score of the other two methods, suggesting a long contigs does not necessarily result in higher identities score. Overall, performance of Trinity assembly fluctuated genome to genome compared to the other two methods. Meanwhile, the observed accuracy and usefulness of contigs assembled by the proposed method were highly similar to Velvet-assembled contigs based on the observed similar contig lengths and identities score.

The usefulness of the contigs assembled by the proposed method were also observed through the distribution of species identified from each set of contigs from the experiment above using BLAST2GO [10].

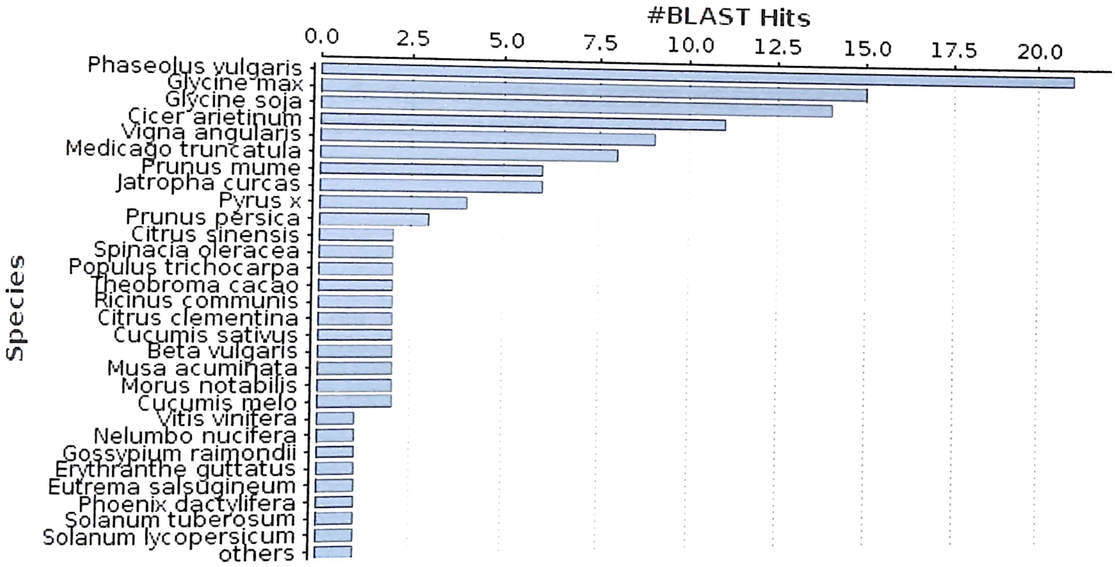


Figure 5-28: The species distribution chart generated by BLAST2GO using a set of contigs assembled by the proposed method based on P. Vulgaris input sequence reads

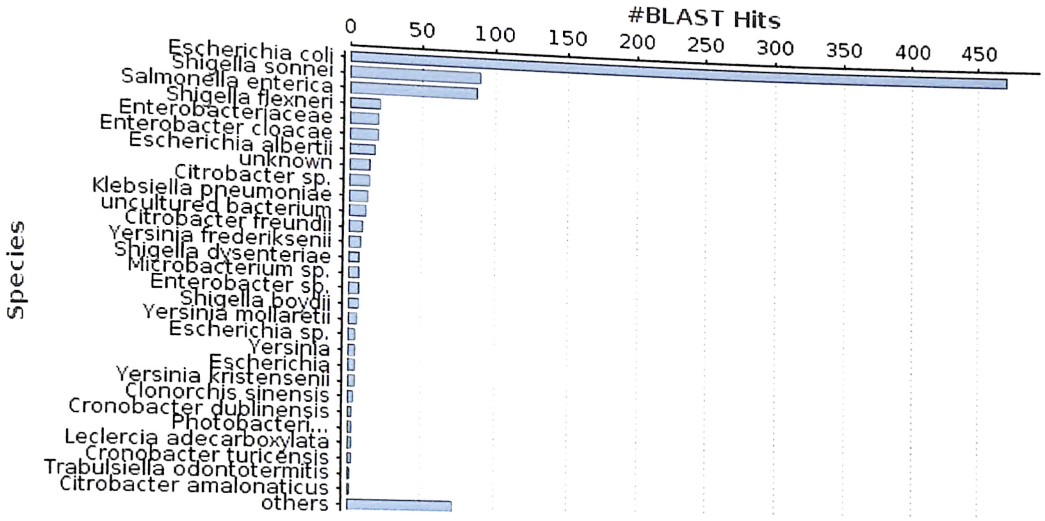


Figure 5-29: The species distribution chart generated by BLAST2GO using a set of contigs assembled by the proposed method based on E.coli input sequence reads

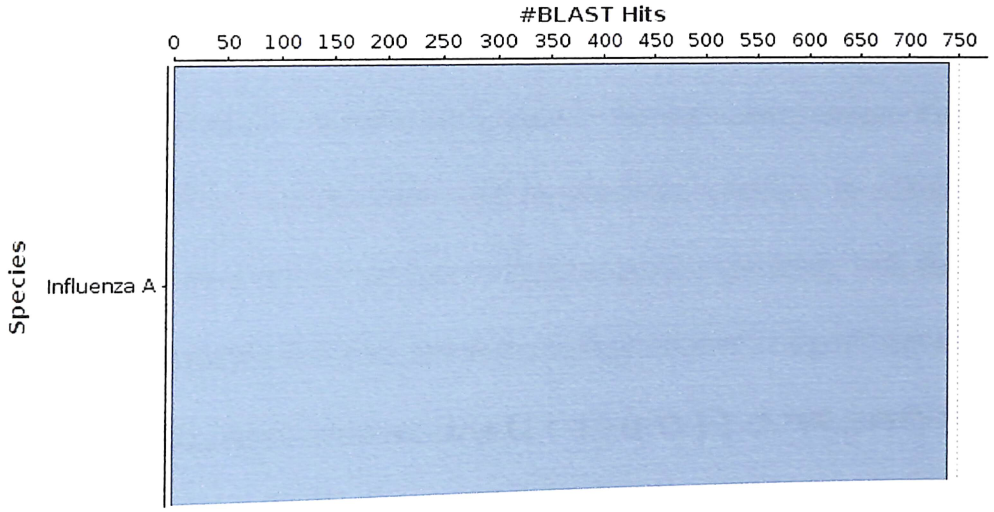


Figure 5-30: The species distribution chart generated by BLAST2GO using a set of contigs assembled by the proposed method based on Influenza virus type-A input sequence reads

As displayed in Figure 5-28, Figure 5-29, and Figure 5-30, correct species were displayed as the top results with the highest number of BLAST hits.

CHAPTER 6

DISCUSSION

The greatest contribution provided by the proposed method is the novel usage of multi-objective evolutionary algorithms to produce multiple diverse sets of assembled contigs. Since there exists variation in each set of contigs assembled by modern sequence assemblers, executing the proposed method once is equivalent to executing a modern sequence assembler multiple times. As demonstrated throughout the experiments, multiple sets of assembled contigs produced by the proposed method could also be utilized to elevate the overall contig quality through construction of an elite contig set via selection of quality contigs from each Pareto-front solution. In addition, the experiments demonstrated the advantage of local sequence assembly with the ACO. A set of ACO-contigs assembled based on read overlaps showed a significant reduction in the number of the initial input sequence reads. Consequently, it had greatly reduced the solution search space for the following sequence assembly processing with the MOEA that focuses on improving multiple quality aspects of the final contigs being assembled.

One of the drawbacks found in the proposed method was the relative lack of further sequence assembly performed by the MOEA component. The evident low solution fitness convergence per generation was observed throughout the experiments. As a result, the MOEA did not produce contigs with significantly improved quality relative to the ACO-contigs. Consequently, the MOEA-contigs had exceedingly low average contig overlap scores. As briefly discussed previously, the limitation could be alleviated by a heightened level of the mutation rate or by using other genetic operators that permit a higher level of stochastic solution exploration. Moreover, increasing the number of bases subject to each mutation could further amplify the stochasticity in the solution exploration. Because of the elitism selection strategy, increasing the stochasticity in the solution exploration, as mentioned earlier, cannot revert the solution fitness in each new generation. In other words, the solution fitness convergence toward its optimality will not be altered by the utilization of the suggested methods since a new set of N_{new} individual solutions are selected from the combined solution set of N_{parent} fittest candidate solutions from the previous generation, and N_{child} number of offspring candidate solutions from the current generation.

Another drawback found in the current version of the proposed method was its processing inefficiency during the construction of the adjacency matrix. Given a set of N number of input sequence reads with L read length, it takes approximately $\frac{1}{2} \cdot L^2$ comparisons to find the maximal overlap between two input read sequences and the construction of an adjacency matrix requires approximately N^2 read-to-read comparisons. Thus, the time complexity of constructing an adjacency matrix in the proposed method is $O(L^2N^2)$. Consequently, the observed failure to process through the input sequence reads that were prepared using a genome with length equal to or greater than 20 knt is likely caused by the brute-force comparisons employed in the current method. As another option, a heuristic method such as an evolutionary algorithm could be used for the construction of the adjacency matrix with the expected loss of precision in the maximal read overlaps, but improved efficiency. Alternatively, a distributed processing technique could be used with a programming framework such as Hadoop [57] to distribute the load of the initial input sequence reads to a cluster of commodity computers and thus reduce the number of sequence reads handled by each computing resource. However, both methods can greatly sacrifice the usability and practicality of a sequence assembler. Preferably, a sequence assembler should be able to perform on a typical modern computer and produce a reliable result.

A potential sequence assembly method that can avoid the processing issue found in the construction of the adjacency matrix while keeping the benefit of the OLC-based sequence assembly could be accomplished through hybridization of DBG-based sequence assembly and OLC-based sequence assembly. Taking the advantage of the time complexity of de Bruijn graph ($O(n)$ [58]) construction, the creation of the adjacency matrix and the local sequence assembly by the ACO could entirely be replaced with extraction of contigs based on maximal non-branching paths from a constructed de Bruijn graph. Consequently, the contigs generated from the constructed de Bruijn graph could further be processed by the MOEA to finalize the sequence assembly. Since the MOEA component employs an OLC-based sequence assembly method, a construction of an adjacency matrix is still required. However, the expected high reduction in the number of sequences following the contig extraction from a constructed de Bruijn graph could overcome the observed processing issue found in handling a large amount of the initial input sequence reads. Most importantly, the benefit of producing multiple diverse sets of assembled contigs is still retained.

CHAPTER 7

Conclusions and Future Work

Taking advantage of increasing read length along with the advancement in the sequencing technology, the design of the proposed sequence assembly method was based on the OLC sequence assembly known for its tolerance to sequence read error and its ability to assemble reads with flexible read overlaps without involving any read-breaking. The proposed method aimed at contributing improvement in the quality of the assembled contig with the aid of hybridization of ACO and MOEA in which their heuristic search for quasi-optimal sequence alignment focused on improving the quality aspect of the contigs being assembled.

Although each Pareto-front output from the proposed method did not exhibit high quality contig set on average, a high leverage in contig quality was observed in a solution set created through selecting quality contigs from each Pareto-front solution. Further, the selectively created solution set demonstrated competitiveness in quality, accuracy, and usefulness of the assembled contigs compared to the contig sets assembled by Trinity and Velvet assemblers; especially, significant improvement in contig quality was observed relative to a current OLC-based assembler. Based on the experiment outcomes, a preferred assembly is expected with the removal of minimum overlap constraint and usage of a higher mutation rate to decrease the stochasticity observed in ACO procedure and to leverage the rate of fitness improvement per generation in MOEA procedure.

Most importantly, resolving the processing inefficiency confronted during the construction of overlap score adjacency matrix is found to be essential to enhance the limited practicality of the current version of the proposed method. The processing bottleneck could potentially be overcome using a heuristic method such as single-objective evolutionary algorithm, distributed processing with HADOOP [51], or GPU parallel computing with CUDA [52]. Lastly, devising a technique to handle paired reads with the proposed method could further enhance the assembly outcome and its practicality.

REFERENCES

1. Schatz, Michael C., Ben Langmead, and Steven L. Salzberg. "Cloud Computing and the DNA Data Race." Nat Biotechnol Nature Biotechnology 28.7 (2010): 691-93.

2. Greenbaum, D., M. Gerstein, and N. M. Luscombe. "What Is Bioinformatics? An Introduction and Overview." Yearbook of Medical Informatics (2001): 83-99. Print.

3. Maglott, D., J. Ostell, K. D. Pruitt, and T. Tatusova. "Entrez Gene: Gene-centered Information at NCBI." *Nucleic Acids Research* 39.Database (2010): n. pag. Web.

4. Mulder, Nicola J., and Rolf Apweiler. "The InterPro Database and Tools for Protein Domain Analysis." *Current Protocols in Bioinformatics* (2002): n. pag. Web.

5. "Ongoing and Future Developments at the Universal Protein Resource." *Nucleic Acids Research* 39.Database (2010): n. pag. Web.

6. "Babraham Bioinformatics - FastQC A Quality Control Tool for High Throughput Sequence Data." Babraham Bioinformatics - FastQC A Quality Control Tool for High Throughput Sequence Data. N.p., n.d. Web. 21 July 2015.

7. Madden T. The BLAST Sequence Analysis Tool. 2002 Oct 9 [Updated 2003 Aug 13]. In: McEntyre J, Ostell J, editors. The NCBI Handbook [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2002-. Chapter 16.

8. Zerbino, D. R., and E. Birney. "Velvet: Algorithms for De Novo Short Read Assembly Using De Bruijn Graphs." Genome Research 18.5 (2008): 821-29.

9. Haas, Brian J., Alexie Papanicolaou, and Moran Yassour. "De Novo Transcript Sequence Reconstruction from RNA-seq Using the Trinity Platform for Reference Generation and Analysis." Nature Protocols Nat Protoc 8.8 (2013): 1494-512.

10. Conesa, A., S. Gotz, J. M. Garcia-Gomez, J. Terol, M. Talon, and M. Robles. "Blast2GO: A Universal Tool for Annotation, Visualization and Analysis in Functional Genomics Research." Bioinformatics 21.18 (2005): 3674-676. Web.

11. Watson, J. D., and F. H. C. Crick. "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid." *Nature* 171.4356 (1953): 737-38. Web.

12. Williams, George C. *The Pony Fish's Glow: And Other Clues to Plan and Purpose in Nature*. New York, NY: Basic, 1997. Print.

13. Bhardwaj, N. "Kernel-based Machine Learning Protocol for Predicting DNA-binding Proteins." *Nucleic Acids Research* 33.20 (2005): 6486-493. Web.

14. Sanger, F., S. Nicklen, and A. R. Coulson. "DNA Sequencing with Chain-terminating Inhibitors." *Proceedings of the National Academy of Sciences* 74.12 (1977): 5463-467. Web.

15. "Introduction to Next Generation Sequencing." (n.d.): n. pag. Web. <http://www.illumina.com/technology/next-generation-sequencing.html>.

16. Anderson, Stephen. "Shotgun DNA Sequencing Using Cloned DNase I-generated Fragments." *Nucl Acids Res Nucleic Acids Research* 9.13 (1981): 3015-027. Web.

17. "Human Genome Project." *Wikipedia*. Wikimedia Foundation, n.d. Web. 06 June 2015.

18. Compeau, Phillip, and Pavel Pevzner. Bioinformatics Algorithms: An Active Learning Approach. La Jolla, CA: Active Learning, 2014. Print.

19. Koning, A. P. Jason De, Wanjun Gu, Todd A. Castoe, Mark A. Batzer, and David D. Pollock. "Repetitive Elements May Comprise Over Two-Thirds of the Human Genome." PLoS Genetics PLoS Genet 7.12 (2011): n. pag. Web.

20. Nagarajan, Niranjan, and Mihai Pop. "Parametric Complexity of Sequence Assembly: Theory and Applications to Next Generation Sequencing." Journal of Computational Biology 16.7 (2009): 897-908. Web.

21. Lu, Bingxin, Zhenbing Zeng, and Tieliu Shi. "Comparative Study of De Novo Assembly and Genome-guided Assembly Strategies for Transcriptome Reconstruction Based on RNA-Seq." Sci. China Life Sci. Science China Life Sciences 56.2 (2013): 143-55. Web.

22. Wikipedia. Wikimedia Foundation, n.d. Web. 18 June 2015. <https://en.wikipedia.org/wiki/Contig>.

23. Simpson, J. T., K. Wong, S. D. Jackman, J. E. Schein, S. J.m. Jones, and I. Birol. "ABySS: A Parallel Assembler for Short Read Sequence Data." Genome Research 19.6 (2009): 1117-123. Web.

24. Haas, Brian J., Alexie Papanicolaou, and Moran Yassour. "De Novo Transcript Sequence Reconstruction from RNA-seq Using the Trinity Platform for Reference Generation and Analysis." Nature Protocols Nat Protoc 8.8 (2013): 1494-512. Web.

25. Zerbino, D. R., and E. Birney. "Velvet: Algorithms for De Novo Short Read Assembly Using De Bruijn Graphs." Genome Research 18.5 (2008): 821-29. Web.

26. Compeau, Phillip E C, Pavel A. Pevzner, and Glenn Tesler. "How to Apply De Bruijn Graphs to Genome Assembly." Nat Biotechnol Nature Biotechnology 29.11 (2011): 987-91. Web.

27. Compeau, Phillip, and Pavel Pevzner. Bioinformatics Algorithms: An Active Learning Approach. La Jolla, CA: Active Learning, 2014. Print.

28. Li, Z., Y. Chen, and D. Mu. "Comparison of the Two Major Classes of Assembly Algorithms: Overlap-layout-consensus and De-bruijn-graph." Briefings in Functional Genomics 11.1 (2011): 25-37. Web.

29. Dorigo, M., V. Maniezzo, and A. Colorni. "Ant System: Optimization by a Colony of Cooperating Agents." IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) IEEE Trans. Syst., Man, Cybern. B 26.1 (1996): 29-41. Web.

30. Bell, John E., and Patrick R. Mcmullen. "Ant Colony Optimization Techniques for the Vehicle Routing Problem." Advanced Engineering Informatics 18.1 (2004): 41-48. Web.

31. Levine, J., and F. Ducatelle. "Ant Colony Optimization and Local Search for Bin Packing and Cutting Stock Problems." J Oper Res Soc Journal of the Operational Research Society 55.7 (2004): 705-16. Web.

32. Liu, Anzuo, Guishi Deng, and Shimin Shan. "Mean-Contribution Ant System: An Improved Version of Ant Colony Optimization for Traveling Salesman Problem." Lecture Notes in Computer Science Simulated Evolution and Learning (2006): 489-96. Web.

33. "Engineering Projects." : ANT COLONY OPTIMIZATION. N.p., n.d. Web. 07 Aug. 2015. <http://ugpro143.blogspot.se/2009/11/ant-colony-optimization.html>.

34. D. Whitley, T. Starkweather and D. Fuquay, Scheduling problems and traveling salesmen: The genetic edge recombination operator, in:Proc. 3rd Int. Conf. on Genetic Algorithms (ICGA '89), George Mason University, Fairfax, VA (1989) pp. 133–140.

35. Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." IEEE Transactions on Evolutionary Computation IEEE Trans. Evol. Computat. 6.2 (2002): 182-97. Web.

36. Zitzler, Eckart, and Lothar Thiele. "Multiobjective Optimization Using Evolutionary Algorithms — A Comparative Case Study." Lecture Notes in Computer Science Parallel Problem Solving from Nature — PPSN V (1998): 292-301. Web.

37. Knowles, J., and D. Corne. "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation." Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406) (1999): n. pag. Web.

38. Weichun Huang, Leping Li, Jason R Myers, and Gabor T Marth. ART: a next-generation sequencing read simulator, Bioinformatics (2012) 28 (4): 593-594

39. Jia, Ben, Liming Xuan, Kaiye Cai, Zhiqiang Hu, Liangxiao Ma, and Chaochun Wei. "NeSSM: A Next-Generation Sequencing Simulator for Metagenomics." PLoS ONE 8.10 (2013): n. pag. Web.

40. "Trevlovett/Python-Ant-Colony-TSP-Solver." GitHub. N.p., n.d. Web. 12 July 2015. <https://github.com/trevlovett/Python-Ant-Colony-TSP-Solver>.

41. Bradnam, Keith R., Joseph N. Fass, and Anton Alexandrov. "Assemblathon 2: Evaluating De Novo Methods of Genome Assembly in Three Vertebrate Species." GigaScience Giga Sci 2.1 (2013): 10. Web.

42. Sadreyev, R. I., M. Tang, B.-H. Kim, and N. V. Grishin. "COMPASS Server for Homology Detection: Improved Statistical Accuracy, Speed and Functionality." Nucleic Acids Research 37.Web Server (2009): n. pag. Web.

43. Katoh, K., and D. M. Standley. "MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability." Molecular Biology and Evolution 30.4 (2013): 772-80. Web.

44. Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau and Christian Gagné, “DEAP: Evolutionary Algorithms Made Easy”, Journal of Machine Learning Research, vol. 13, pp. 2171-2175, jul 2012.

45. Genetic Code Clip Art." Genetic Code Clip Art at Clker.com. N.p., n.d. Web. 15 Aug. 2015. <http://www.clker.com/clipart-genetic-code.html>

46. Escherichia Coli, National Center for Biotechnology Information, n.d. Web. <ftp://ftp.ncbi.nih.gov/genomes/Bacteria/Escherichia_coli_Xuzhou21_uid163995 >.

47. Influenza Type A, National Center for Biotechnology Information, n.d. Web. <ftp://ftp.ncbi.nih.gov/genomes/Viruses/Influenza_A_virus__A_New_York_392_2004_H3N2__uid15622/>.

48. Phaseolus Vulgaris chromosomes, National Center for Biotechnology Information, n.d. Web. <ftp://ftp.ncbi.nih.gov/genomes/genbank/plant/Phaseolus_vulgaris/latest_assembly_versions/GCA_000499845.1_PhaVulg1_0/GCA_000499845.1_PhaVulg1_0_assembly_structure/Primary_Assembly/assembled_chromosomes/FASTA/>.

49. Simpson, J. T., and R. Durbin. "Efficient De Novo Assembly of Large Genomes Using Compressed Data Structures." Genome Research 22.3 (2011): 549-56. Web.

50. Wheeler D, Bhagwat M. BLAST QuickStart: Example-Driven Web-Based BLAST Tutorial. In: Bergman NH, editor. Comparative Genomics: Volumes 1 and 2. Totowa (NJ): Humana Press; 2007. Chapter 9.

51. Taylor, Ronald C. "An Overview of the Hadoop/MapReduce/HBase Framework and Its Current Applications in Bioinformatics." BMC Bioinformatics 11.Suppl 12 (2010): n. pag.

52. Nickolls, John, Ian Buck, Michael Garland, and Kevin Skadron. "Scalable Parallel Programming with CUDA." *Queue* 6.2 (2008): 40.

53. "Production Power." HiSeq 2500 Ultra-High-Throughput Sequencing System. N.p., n.d. Web. 06 Nov. 2015. <http://www.illumina.com/systems/hiseq_2500_1500.html>.

54. Wilson, John H., and Tim Hunt. Molecular Biology of the Cell, 4th Edition: A Problems Approach. New York: Garland Science, 2002. Print.

55. Lesk, Arthur M. Introduction to Genomics. Oxford: Oxford UP, 2007. Print.

56. Pevsner, Jonathan. Bioinformatics and Functional Genomics. Hoboken, NJ: Wiley-Liss, 2003. Print.

57. Taylor, Ronald C. "An Overview of the Hadoop/MapReduce/HBase Framework and Its Current Applications in Bioinformatics." BMC Bioinformatics 11.Suppl 12 (2010): n. pag. Web.

58. Shah, M.k., Hojoon Lee, S.a. Rogers, and J.w. Touchman. "Exhaustive Genome Assembly Algorithm Using K-mers to Indirectly Perform N-squared Comparisons in O(N)." Proceedings. 2004 IEEE Computational Systems Bioinformatics dConference, 2004. CSB 2004. (2004): n. pag. Web.

Kenneth Shim

Shim.Kenneth@yahoo.com
Cell: 302-784-5341
78 Avalon Ln.
Camden, DE 19934

Summary

Through the graduate program of computer science, I gained skills and experience in advanced computing theories and applications concentrating in the fields of computational intelligence, data mining, and machine learning. In addition, I attained practical data analytics skills through undergraduate and graduate research projects in bioinformatics and computational neuroscience.

Education

Delaware State University
2013 – Present

Master of Science in Computer Science (M.S.)
GPA: 3.81
Specialization:
Computational Intelligence and Bioinformatics
Thesis:
Approaching the Sequence Assembly Problem with Hybridization of Ant Colony Optimization and Multi-Objective Evolutionary Algorithms

Delaware State University
2010-2013

Bachelor of Science in Computer Science (B.S.)
GPA: 3.96
Capstone:
Autonomous Floorplan Generation Using Robots

Thomas Jefferson University
2004 - 2006

Bachelor of Science in Radiologic Science (B.S.)
GPA: 3.72
Specialization:
Nuclear Medicine

Awards

2013	Institutional Honorable Mention at Delaware State University
2012 - 2013	Computer and Information Science Department Scholar
2010 - 2013	President's List

Skills

Language	English: fluent (speaking, reading, writing) Korean: fluent (speaking, reading, writing)
Database	MySQL, MS-Access
Operating System	Linux, Mac OS X, Windows
Programming	Java, Python
Statistic	Matlab, R
Web Development	Ajax, CSS3, HTML, JavaScript, JQuery, PHP

Work History

2013 - Present	Graduate Research Assistant ● USDA Blueberry Transcriptome Analysis
2012 - 2013	Undergraduate Research Assistant ● Analysis of Conductance Correlation in STG Neurons
2011	Tutor, Undergraduate Computer Programming

Project and Research	Thesis 2014-2015	Approaching the Sequence Assembly Problem with Hybridization of Ant Colony Optimization and Multi-Objective Evolutionary Algorithm • Designed and implemented a novel heuristic method for DNA/RNA sequence assembly. The proposed sequence assembly method was evaluated for determining its advantages and disadvantages through various data analyses.
	USDA 2013-2015	Lab (CIBiL) Coordinator, Server Administrator, Assistant for undergraduate and graduate research projects
	Capstone 2012-2013	Applying Autonomous Robotic Exploration and Mapping for Generation of Floorplans
	Computational Neuroscience 2012-2013	Analyzing conductance correlations involved in recovery of bursting after neuromodulator deprivation in lobster stomatogastric neuron
	Combinatorics and Graph Theory 2011-2012	Exploring the Properties of Bit-String Arc Diagrams using Bijective Proofs, Recurrence Relations, and Advanced Counting Techniques

Volunteering and Other Activities	2010 - Present	Member, Computational Intelligence and Biological Informatics Lab (CIBiL)
	2013 - Present	Lab (CIBiL) Coordinator, Server Administrator, Assistant for undergraduate and graduate research projects
	2013 - Present	Member, Generic Model Organism Database (GMOD)
	2013	Attendee, GMOD Summer School
	2015	A referee in First Lego League at Delaware State University

Reference

Dr. Tomasz G. Smolinski

Associate Professor, Computer and Information Sciences Department
Science Center North, Room 344
1200 N. Dupont Highway
Delaware State University
Dover, Delaware 19901
Phone: (302) 857-7951
Email: tsmolinski@desu.edu

Dr. David Pokrajac

Professor, Computer and Information Sciences Department
Grossley Hall, Room 104
1200 N. Dupont Highway
Delaware State University
Dover, Delaware 19901
Phone: (302) 857-7412
Email: dpokrajac@desu.edu

Dr. Gary F. Holness

Director, Computer and Information Sciences Department
Science Center North, Room 342
1200 N. Dupont Highway
Delaware State University
Dover, Delaware 19901
Phone: (302) 857-7932
Email: gholness@desu.edu

Dr. Marwan F. Rasamny

Chairperson, Computer and Information Sciences Department
Science Center North, Room 330
1200 N. Dupont Highway
Delaware State University
Dover, Delaware 19901
Phone: (302) 857-7896
Email: mrasamny@desu.edu

Dr. Janko
Milutinovic

Associate Professor, Computer and
Information Sciences Department
Science Center North, Room 334C
1200 N. Dupont Highway
Delaware State University
Dover, Delaware 19901
Phone: (302) 857-6648
Email: jmilutinovic@desu.edu