1-1-2014

De Novo Co-Assembly Of Bacterial Genomes From Multiple Single Cells

Narjes Sadat Movahedi Tabrizi
Wayne State University,

DE NOVO CO-ASSEMBLY OF BACTERIAL GENOMES FROM MULTIPLE SINGLE CELLS

by

Narjes Sadat Movahedi Tabrizi

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2014

MAJOR: COMPUTER SCIENCE

Approved By:

Advisor Date

DEDICATION

To my family.

ACKNOWLEDGEMENTS

I would like to sincerely thank my adviser, Professor Hamidreza Chitsaz, for his support, novel ideas and the opportunity he provided for me to conduct my research. His patience and encouragements inspired me during the course of my graduate studies. I would also like to thank my dissertation committee members, Professor Zhu, and Professor Kotov for accepting to be on my dissertation committee. I like to acknowledge chitsaz lab members especially Dr. Zeinab Taghavi for being such great colleagues. I have been blessed with unconditional and endless love my family. Finally I would like to express my sincere gratitude to my husband, my love for his great companionship and unconditional support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

Today, a variety of studies in medical and environmental sciences benefit from genomic analysis of bacteria. A staggeringly large portion of the genomic content in environmental samples is comprised of sequences that have never been observed before. These uncharacterized sequences in diverse places ranging from oceans to human mucus are guessed to be predominantly of bacterial or viral origin. This demonstrates the extent of the complexity and ubiquity of bacterial genomes in our environment. Several projects including Human Microbiome Project (HMP) and Earth Microbiome Project (EMP) [4] aim at cataloguing microbial communities in various locations.

One significant requirement for bacterial genome analysis is the genome sequence, which is aimed to be obtained via the genome assembly process. Genome assembly is the procedure of extracting genome sequence using short subsequences generated from fragments of DNA. The first step of genome assembly is extracting the DNA and then breaking it into several small fragments. Next, the fragments are sequenced in a specific size from both sides; these sequences are called *reads*. Unfortunately, the genome assembly using the reads generated from one single cell is not possible due to the lack of enough DNA material to generate sufficient sequencing reads. The inadequate material for sequencing of one species at a time is often the main limiting factor in those microbiome studies. Therefore, to sequence the genome of a new bacterial species, we require a sample containing numerous identical cells to extract enough DNA material, which is in the order of micrograms.

The conventional way consists in extracting a few bacterial cells and culturing them separately. If a cell cultures successfully, then the resulting colony provides the required amount of good quality DNA material. Unfortunately, the proportion of microbes that can successfully be cultured in the lab is less than 1% [16].

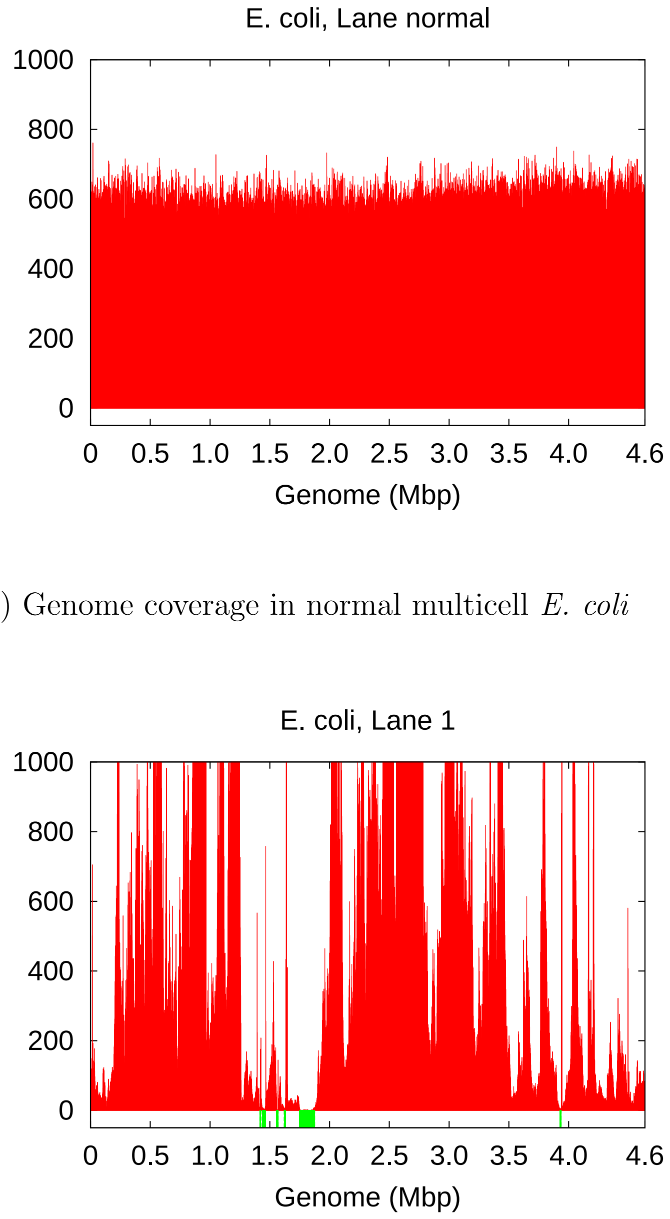
Many bacterial cells cannot be cultured in the lab since they often require complex symbiotic environments to grow. In the case of such uncultivable species, the only existing method to proceed involves whole genome amplification a billion fold from femtograms to micrograms. The technology of assembling one cell is called *Single Cell Sequencing*, which is selected by Nature in 2013 as its "`Method of the Year`" [21].

Different amplification methods have been proposed that provide enough DNA material from very few cells or even a single cell: (1) PCR such as primer extension pre-amplification (PEP) and degenerate oligonucleotide primed PCR (DOP) whose products are usually short DNA fragments (less than 1 kbp), and (2) multiple displacement amplification (MDA) which generates long DNA products (up to 100 kbp with an average length of 12 kbp) [8, 9, 17]. MDA works using random primers and a special DNA polymerase called $\Phi29$ with interesting characteristics. The enzyme $\Phi29$ is able to open up double stranded DNA and continue its way without external thermal help if it encounters a double stranded region while it is synthesizing its complementary strand [16, 17]. This unique property of $\Phi29$ makes MDA an isothermal reaction that does not suffer from the side effects of thermocycling such as GC-bias as opposed to PCR. MDA provides a better coverage of the genome in comparison to other methods and it generates a data with more uniform coverage, less amplification bias by 3-4 orders of magnitude than previous PCR-based methods [15, 34], however, still some parts of the genome are lost or poorly covered while some others are orders of magnitude more abundant in the final product . In contrast, the datasets of cultured cells have uniform coverage distributions (see Figure 1.1).

Recently, a new amplification method has been proposed for human genome, which is called multiple annealing and looping-based amplification cycles (MALBAC)[34]. More detailed study is required to address the amplification bias reduction by this method [21]. Although MDA is not a perfect amplification method, it is currently the method of choice because of its efficacy and lower coverage bias in comparison to other methods [16]. Fortunately, a computational solution to overcome the MDA bias was proposed by Chitsaz *et al.* [7] in 2011, which handles the un-even depth of coverage challenge. The method is applied by all single cell sequencing tools such as SPAdes [1] and IDBA-UD [25].

Studies show that the larger amounts of DNA as the initial template can decrease this type of amplification bias. This is due to the fact that amplification bias in multiple MDA reactions of the same genome happens randomly (at least to a good extent) in different regions [29]. This suggests that lost or severely under represented genomic regions in one MDA reaction may be present in another MDA reaction of the same genome. To benefit from this fact, the product of different MDA processes on several cells with the same genome can be combined and used together. In this way, some MDA products represent those sequences that are lost in the other processes. As a related but different strategy, single cells can be pooled prior to amplification, and MDA reaction can be done on the pool which again helps to generate more uniform coverage of the genome. In fact, it has been shown that starting with 5 to 10 initial cells as opposed to a single cell significantly improves the result.

These pooling strategies are applicable only if we are able to isolate a few single cells with identical genomes. However, this turns into a chicken-and-egg problem: how can we know that a few cells are identical before sequencing their genomes while we require to know that they are identical for a complete sequencing of their genomes? If the cells are guessed to be identical but are not actually identical, then mixing their DNA before or after amplification results in a chimeric assembly which has to be strongly avoided.



(a) Genome coverage in normal multicell *E. coli*

(b) Genome coverage in single-cell *E. coli* lane 1

Figure 1.1: Genome coverage for both (a) single-cell *E. coli* lane 1, and (b) normal multicell *E. coli*. Both have an average coverage of $\sim 600\times$. The coverage for all bases in the multicell case is uniform. On the other hand, the MDA bias in the single cell case causes some bases to have high coverage and some bases to be covered poorly or not covered at all, which are called blackout bases (represented in green regions). The data is reported in [7].

1.1 Our Contribution

We propose an elegant solution to this problem that assembles a number of guessed to be identical single cells together. Meanwhile, it can detect the non-identical cells and eliminate the false sequences which are originated from the outlier datasets. Our algorithm is based on colored de Bruijn graph [12], which was proposed in a different context (for structural variation detection and genotyping using multicell datasets). We use the colored de Bruijn graph for a different application and for more general type of datasets (including single cell datasets). Our co-assembly tool HyDA [19], which can assemble a number of datasets simultaneously, is based on a simple single cell assembly tool, Velvet-SC [7]. The main contribution that has been done in this work is the introduction of our algorithm for assembling a genome using a number of single cell datasets, which possibly are not of the same genome. The quality of the generated result is similar to the genome assembly of normal multicell datasets (which have the uniform coverage distribution) when we have sufficient number of single cells. In the proposed algorithm, the result of co-assembly process (which has been done using colored de Bruijn graph) is analyzed to indicate the identical cells and then the result is trimmed by removing the false contigs, originated from the co-assembled cells that are of another genome.

1.2 Related Works

To date, colored assembly graphs and co-assembly is employed in calling genomic structural variations. Cortex [12] is the first assembly tool that uses the co-assembly approach in the context of structural variation detection and genotyping. Since Cortex is designed to process multicell datasets and use the information of known coverage distribution, the challenges related to uneven depth of coverage and blackout regions, which are introduced by single cell sequencing technology, are not considered.

Although, Cortex is the only assembly tool that does co-assembly process using colored de Bruijn graph, there is another tool, Magnolya [23], that applied co-assembly approach in the purpose of detecting Copy Number Variation. The colored graph employed by Magnolya works based on the Overlap/Layout/Consensus method. Both Cortex and Magnolia use algorithms that require known coverage distribution and cannot work with single cell datasets (whose average coverage is random).

On the other hand, some other tools have been developed to assemble single cell datasets and handle their uneven depth of coverage characteristic. The first work, Velvet-SC and E+V-SC, was proposed by Chitsaz *et al.* [7] , which has been followed by other single cell assembly tools (all employ de Bruijn graph) such as, SPAdes [1] and IDBA-UD [25].

The assembler we employ in this work is HyDA [19], which is a redistilled/reimplemented version of Velvet-SC, the first tool for single cell sequencing. Other single cell assembly tools follow Velvet-SC and apply some further strategies to improve their results. Each new method employed by those state of the art assemblers is a module that can be implemented in other assembly tools (including HyDA). Using read error correction, iterative assembly with variable k-mer size, aligning reads to the graph, and considering the information of paired end reads are examples of further steps implemented in some state of art single cell sequencing tools. Obviously, applying such algorithmic modules improve simple tools such as Velvet-SC and HyDA. On the other hand, we expect the incorporation of our colored assembly method into these state of the art assembly tools to improve their assembly result. Also, implementation of the methods applied by them is listed as future work to improve our tool, HyDA.

1.3 Organization

The rest of the document is organized as follows. In Chapter 2, we describe the basics of genome assembly using de Bruijn graph algorithm and the strategies for single cell sequencing. In Chapter 3, we describe the colored de Bruijn graph as the main structure of co-assembly process. In Chapter 4, we introduce our software, HyDA, with some programming details. In Chapter 5, we present the experiments that have been done and evaluate the results. Finally in Chapter 6, we summarize the results and present possible directions for future research.

Chapter 2: Background

The first bacterial genome was sequenced using conventional Sanger sequencing data in 1995 [11]. A decade later, in 2005, a revolution happened in the field of genetics, by the emergence of the Next Generation Sequencing (NGS) technology, which is also called High Throughput Sequencing (HTS) technology. This technology has reduced the delivery time of bacterial genome sequences from months (or years) to hours (or days) and for a much lower cost [18]. The outputs of Next Generation Sequencing platforms, which are datasets of short sequencing reads with high depth of coverage, are used by computational assembly tools as the input of the genome assembly process.

2.1 *De novo* Assembly

DNA is a long chain of 4 types of nucleic acids: adenine, thymine, cytosine, guanine, which are represented by A, T, C, and G respectively. In other words, a genomic sequence is shown by a string in the alphabet $\Sigma = \{A, T, C, G\}$. A fundamental problem, called *shortest common superstring*, is the problem of interest in genome assembly. The problem is defined as: given a set of substrings $S = \{s_1, s_2, \cdots, s_n\}$ in alphabet $\Sigma = \{a_1, a_2, \cdots, a_m\}$, the goal is determination of the shortest string that contains all substrings $s_i \in S$. It has been shown that the problem is NP-complete even when we have only two alphabet characters, such as $\{0, 1\}$ [**?**].

In the field of genomics:

- The alphabet is the nucleic acids set $\{A, T, C, G\}$.
- Substrings are the sequencing read dataset.
- The shortest common sequence is the genome.
- The problem of finding shortest common superstring is the assembly.

There are four main strategies that are used to find shortest common superstring as the assembly result:

1. Greedy algorithm
2. Overlap/Layout/Consensus (OLC)
3. String graph
4. De Bruijn graph

2.1.1 Greedy Algorithm

A simple solution for shortest common superstring problem is a naïve greedy algorithm which is generally used for Sanger data and is not suitable for short reads generated by NGS platforms. In each iteration of the greedy solution, two substrings of the set are taken and the shortest string that contains both substring is calculated and replaced with its parent substrings (the substrings are merged to generate a superstring with the maximum overlap score). The final result is one long superstring left in the set [28].

2.1.2 Overlap/Layout/Consensus

Overlap-layout-consensus algorithm is another solution for shortest superstring problem, which is used by most established assemblers developed for Sanger technology. The algorithm has three steps. In the first step, which is called overlap, every two reads are compared to one another and the overlapping score is calculated. In the second step, an overlap graph G is constructed in which the reads are represented by nodes and significant overlaps between two reads are represented by an edge. Finally, the computed layout in the second step is used to generate an assembled sequence in the consensus stage, whose goal is finding a Hamiltonian path (a path that traverses all the nodes just once) in the graph. A modified version can be used for paired end reads [2].

2.1.3 String Graph

String graph, another algorithm for genomic assembly was introduced by Myers [20] as an alternative form of de Bruijn graph. The method is similar to Overlap/Layout/Consensus strategy in which each read is represented by a node in the graph and an overlap between them is demonstrated by an edge. The strategy relies on edge reduction: when node x and y are connected by an edge; and also y and z are connected, if there is an edge between x and z, it will be removed as a redundant transitive edge. Based on the statistical factors, the edges in the graph are classified as `1-exact,` `2-required,` and `3-optional`. To find the shortest path (shortest common superstring), the graph is traversed while each `exact` edge should be traversed exactly once; however, crossing `required` edges is allowed zero or one time and `optional` edges can be crossed any number of times.

2.1.4 De Bruijn Graph

The most applied method for *de novo* assembly using short reads generated by High Throughput Sequencing platforms is done by constructing a de Bruijn graph [27, 26]. Each vertex of the graph represents a length-k subsequence, called k-mer. A set of k-mers is generated for each sequencing read. In fact, a length-m read contains $m - k + 1$ k-mers. Figure 2.1 illustrates a read and its corresponding k-mers.

Two consecutive k-mers in a read are represented by two connected vertices in the graph. That is, if $k - 1$ last bases of vertex u is the same as $k - 1$ first bases of vertex v in some read, u and v are connected by an edge from u to v. For example, in Figure 2.1, GCT and CTT are two consecutive 3-mers (last 2 bases of GCT is the same as first 2 bases of CTT) which are connected by a directed edge from GCT to

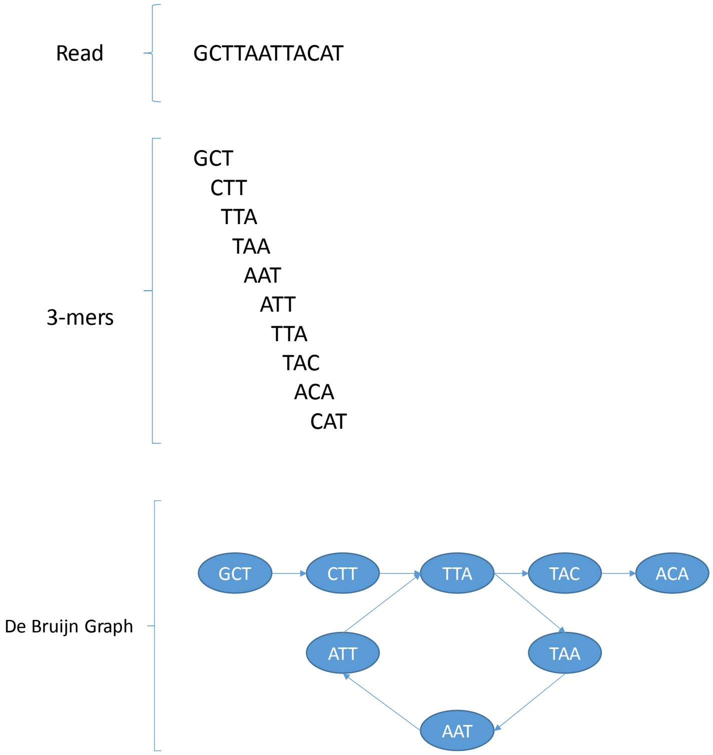
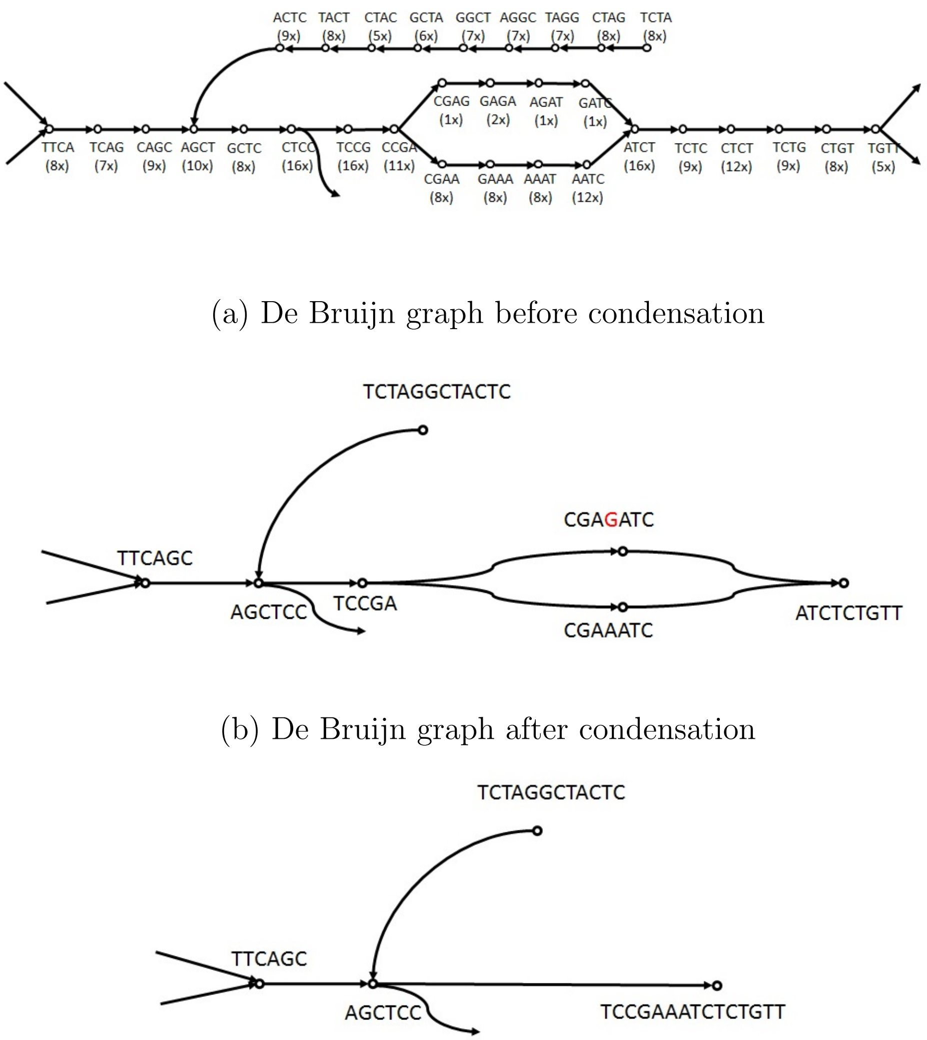


Figure 2.1: The construction of de Bruijn graph for a given read with $k = 3$ is illustrated. Ten 3-mers are generated for the 12 character long sequencing read. Each 3-mer is represented by a node in the graph. Two consecutive 3-mers are connected by a directed edge. A node can represent more than one loci if the sequence of all representative 3-mers are equal (node TTA represents two loci).



(a) De Bruijn graph before condensation

(b) De Bruijn graph after condensation

(c) De Bruijn graph after error removal and recondensation

Figure 2.2: De Bruijn graph condensation with or without error removal is displayed. To condense the graph, all one-in one-out chains of nodes in (a) are merged and replaced with the constructed supernode. That is a simple path without any branching is represented by one supernode. De Bruijn graph (a) is condensed and modified to graph (b). The coverage of each supernode is the average coverage of the nodes in the condensed path. If the average coverage is less than the cut off, the supernode is removed from the graph and the graph is recondensed. Let $cutoff = 2$. Then, the only supernode in graph (b) that has the average coverage less than the cut off is CGAGATC (the average coverage is 1.25), which is removed and the graph (c) is constructed after recondensing.

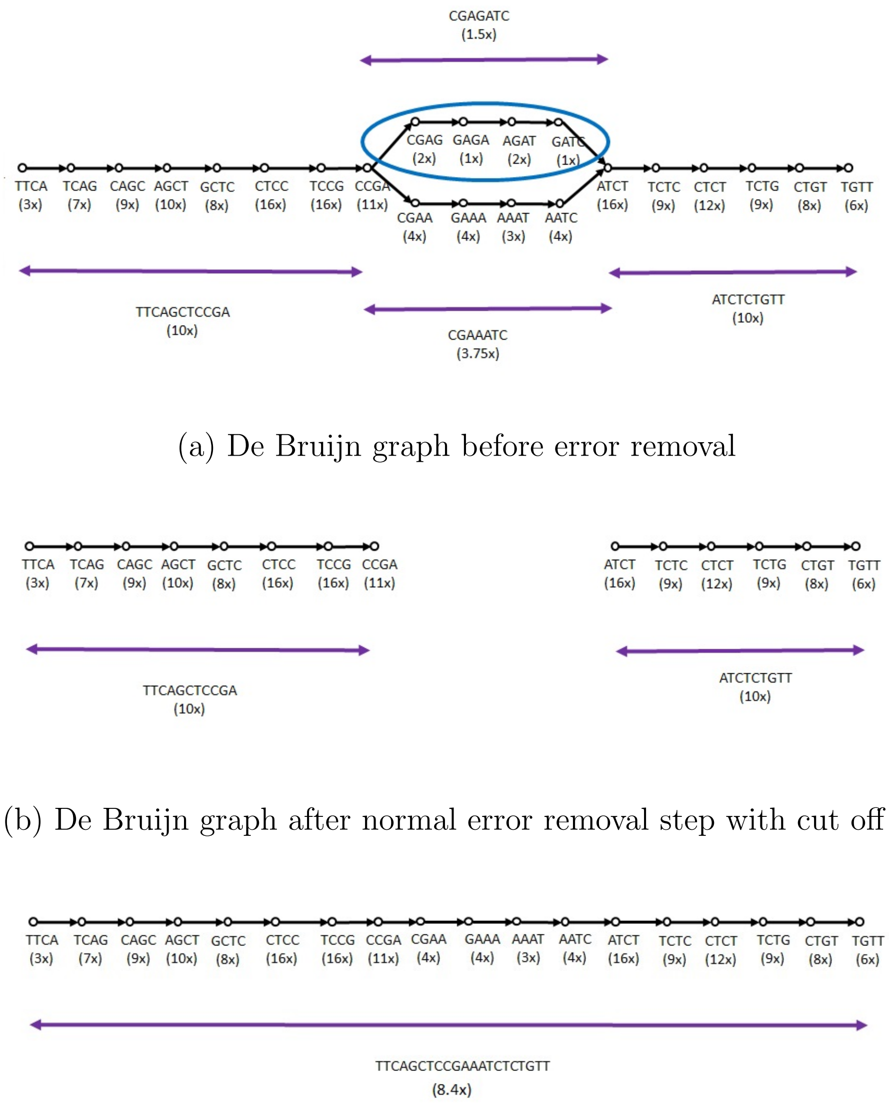
CTT.

Ideally, overlapped k-mers construct one Eulerian path that goes through the DNA sequence and spells out the entire genome (therefore, it is also called Eulerian approach). However, extracting one Eulerian path is not possible due to branches, gaps, and loops. Basically, repeats caused by k restriction, potential gaps, and errors obstruct extraction of one unique traversal of the graph.

Sequencing errors and contamination cause the generation of not only some incorrect k-mers but also some extra edges, which make branchings. For example, vertex CGAGATC in Figure 2.3 (b) is generated due to the sequencing error that mutate A to G (mutate CGAAATC to CGAGATC). Another restriction of this approach is related to the value of k. Repeats make some k-mers represent more than one region in the genome and this introduces many branchings in the graph. To avoid having such repetitive k-mers, a large value of k is required. However, using a large k, introduces some gaps and disconnected subgraphs due to limited coverage and sequencing errors. Two large k-mers are connected when they have a large overlap (in the size of $k-1$), which does not happen often when k is large especially in low covered regions.

Since having one path along the genome is not possible, all the maximal paths (the paths without branching) are exhibited as the assembly result and called contigs. The maximal paths are obtained in the condensation stage (see Figure 2.2). That is, all one-in one-out consecutive nodes are condensed into one supernode (contig).

As mentioned before, the de Bruijn graph approach is the most applied algorithm for genome assembly especially for short reads generated by Next Generation Sequencing platforms. Some well-known assembly tools using de Bruijn graph are Velvet [33], Abyss [32], Euler-SR [6, 5], and AllPaths [3].



(a) De Bruijn graph before error removal

(b) De Bruijn graph after normal error removal step with cut off 5

(c) De Bruijn graph after single cell error removal step with cut off 5

Figure 2.3: The difference between normal error removal approach and the method used in single cell assembly is illustrated. Consider that cutoff equals to 5. Two supernodes in the graph (a) have a coverage less than the cutoff. In normal error removal stage (b), both supernodes are removed and the remained graph is considered as the output. In the error removal method used in single cell assembly, the cutoff gradually increases and after each error removal iteration the graph is recondensed. The graph (c) illustrates the graph after single cell error removal method: after the second iteration, in which the node with the average coverage less than 2 is removed, the graph is condensed. Since the remaining nodes have the average coverage greater than the cutoff, all the nodes are saved. That is the sequence CGAAATC is rescued.

2.2 Single Cell Sequencing

Today, the possibility of sequencing a species using one isolated cell is relying on the experimental achievements [13, 14, 22] and the computational solution proposed by Chitsaz *et al.* [7]. This solution is the main computational method to overcome the uneven coverage generated by MDA as the dominant amplification technique in single cell sequencing technology. As mentioned above, the major problem of sequencing using de Bruijn graph is related to branching and gaps. The problem is aggravated when some regions have low coverage, which normally makes them indistinguishable from errors. There are two major methods for eliminating errors from the graph. In one approach the graph is pruned by elimination of k-mers with low multiplicity and then the maximal paths are extracted as the output [6, 24]. Another method does the error removal process after the condensation step. In this approach, the contigs that are formed by low-coverage k-mers are thrown away. That is, the contigs with the coverage lower than the threshold are not considered in the final result [33].

None of those solutions worked properly for unevenly covered genomes sequenced by single cell sequencing methods. Velvet-SC [7] introduced an iterative approach that can reduce the effect of uneven sequencing depth of coverage. Velvet-SC uses a variable threshold for error removal, which starts from 1 and gradually increases into a specific cutoff value. The cutoff is chosen based on the average depth of coverage. In each iteration, the condensed graph is pruned by elimination of all the supernodes with the coverage less than the current threshold. In each iteration, some branches are removed and the remaining supernodes can be condensed further. The procedure of normal error removal and the method used by single cell sequencing tools are compared in Figure 2.3. Velvet-SC is followed by other single cell sequencing tools such as SPAdes [1] and IDBA-UD [25]. The main achievement of the method is related to removing errors while saving low-covered bases in the single cell datasets.

Chapter 3: Colored De Bruijn Graph

Co-assembly using colored de Bruijn graph was first introduced by Cortex [12] for the purpose of structural variation detection and genotyping. Cortex is not suitable for single cell datasets and its major application is calling structural variations. Our tool, HyDA uses colored de Bruijn graph for co-assembly. However, since it is implemented to work with more general read datasets including single cell read dataset, the implementation of the colored graph and assembly process is more complicated than Cortex. Moreover, HyDA focuses on the assembly improvement and detection of more bases in genome sequencing. In this document, we discuss one application of HyDA, which is the production of a high quality genome assembly using datasets of some single cells with unknown reference genome.

3.1 Coloring reads and k-mers

Each stage of co-assembly process retains most of the information in each dataset, i.e. a unique color is assigned to each dataset and this color value is used to distinguish objects based on their original dataset. Therefore, the reads of a colored dataset and the k-mers generated from those reads are colored too. Figure 3.1 displays two colored datasets. All the reads inside the pink dataset are pink and all the reads inside the purple dataset are purple. Similarly, all the k-mers generated of a colored read are painted with the same color.

3.2 Graph construction

The structure of the colored de Bruijn graph is pretty similar to the normal de Bruijn graph except that its k-mer structure contains an extra object to keep the coloring information. To import a new k-mer to the normal de Bruijn graph, first

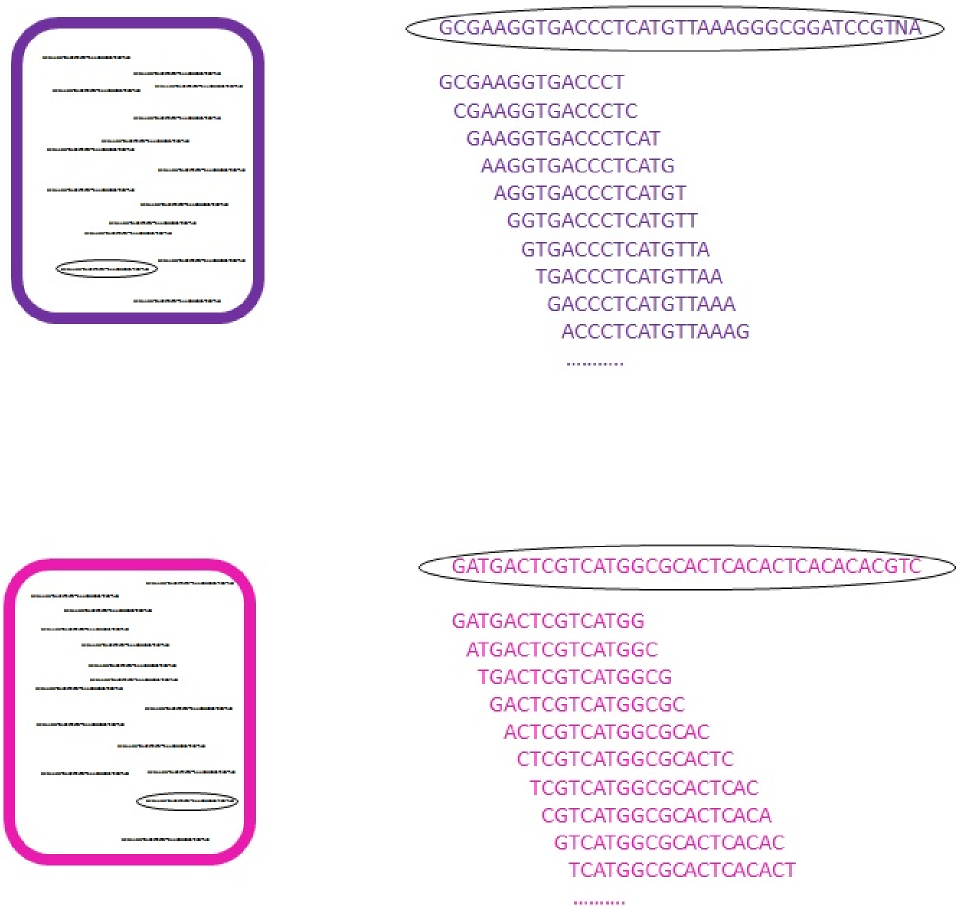
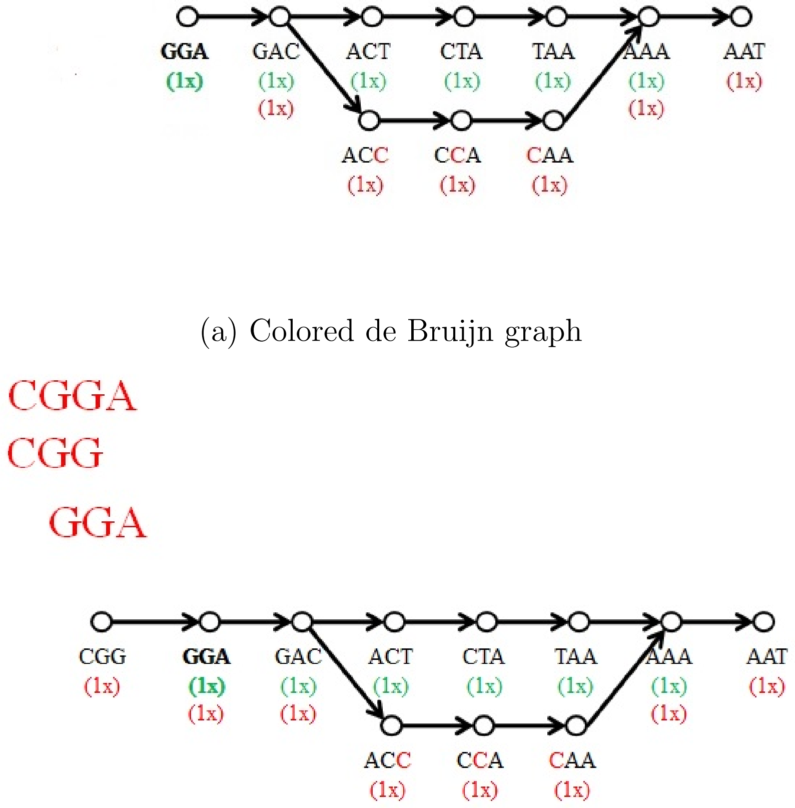


Figure 3.1: In co-assembly process, a unique color is assigned to each read dataset. All the objects, such as reads and k-mers that originated from a dataset are painted with the same color as their source's.

the existence of a similar k-mer (node) is checked. If a k-mer with the same sequence already exists in the graph, no new k-mer (node) is generated and only the multiplicity value of the existing node increases. The multiplicity value of a k-mer (node) indicates the number of sub-regions in the genome with the same sequence as the k-mer. The process of importing a new colored k-mer to the colored de Bruijn graph is similar to the normal one in which the a new node is added to the graph if a node with the same sequence does not exist in the graph. That is, adding a new node in the colored de Bruijn graph is only dependent on the structure of the graph and the coloring information is not considered. To add the information of the new colored k-mer to the existing node, the colored multiplicity increases once. On the other hand, in the absence of the similar k-mer, a new node is added to the graph with information of the imported k-mers (all the colored multiplicity are zero except for the color of imported k-mer). The procedure of adding a new k-mer to the colored de Bruijn graph is illustrated in Figure 3.2.

3.3 Colored de Bruijn graph condensation

After graph construction, all one-in one-out nodes are condensed and supernodes are generated. Each supernode is the longest sequence obtained from a non-branching path and called a contig. The condensation procedure is done only based on the structure of the graph and the coloring information does not affect the sequence of the supernodes. Like colored k-mers, the colored supernodes has an object that keeps the colored coverage information. That is, the coverage of a contig in each dataset is kept separately. The colored coverage is the average of colored multiplicity of k-mers. In Figure 3.3, the red and green coverage for condensed nodes are calculated. After the primary condensation, errors are trimmed. The error removal stage and recondensation are joint and iteratively are run with an increasing cut off value similar to the normal de Bruijn graph. To remove low covered contigs, the maximum colored



(a) Colored de Bruijn graph

(b) After inserting k-mers

Figure 3.2: To insert the information of a new k-mer, the coloring information is not considered. That is, if a node with the same sequence exists in the graph, a new node will not be generated and only the colored coverage is added to the existing node. Otherwise, a new colored node is added to graph.
To add the red 3-mer CGG to the graph (a), the graph structure is examined. No node in the graph has the sequence of CGG. Hence, a new node is added to graph with the sequence CGG and the red color coverage 1. On the other hand, inserting the information of the 3-mer GGA does not change the structure of the graph since a node with this sequence is already present in the graph. Only the colored coverage information is inserted. Adding these new red 3-mers converts graph (a) to graph (b).

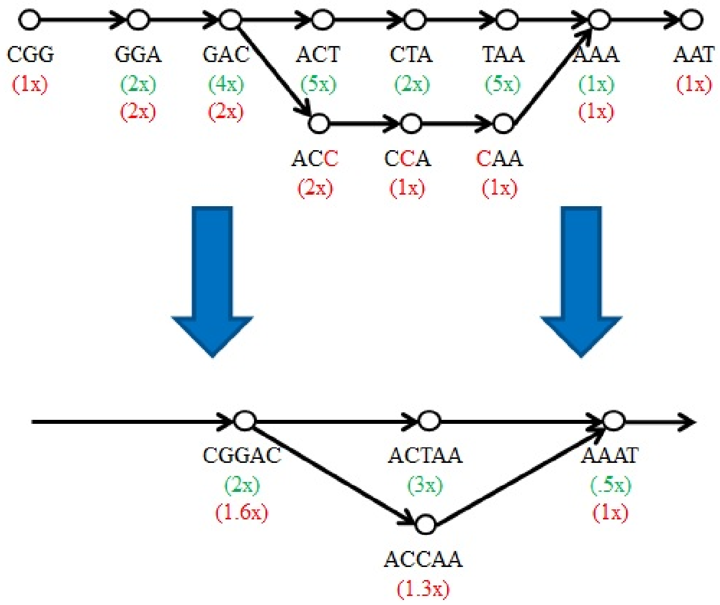


Figure 3.3: Colored de Bruijn graph condensation stage. The colored coverage of the condensed node is the average of the colored coverage of the nodes in the path.

coverage is considered and the condensed node is removed if the maximum value is less than the cutoff threshold. In other words, a supernode is removed from the graph if all of its colored coverage is less than the threshold. The assembly results are the condensed nodes (contigs), which include the depth of coverage for each colored dataset.

Chapter 4: Implementation of the Co-assembly Tool

We implemented our co-assembly tool, HyDA based on the colored de Bruijn graph, which is a modified version of normal de Bruijn graph [27]. HyDA is categorized as a single cell assembly tool, which follows Velvet-SC [7] to address the uneven depth of coverage as the major problem in single cell sequencing technology. The first version of HyDA, which is represented in this document as a simple assembly tool, lacks assembly improvement modules such as variable k graph and read alignment (which are implemented in state of the art single cell assembly tools such as SPAdes and IDBA-UD).

HyDA does the assembly process in three separate stages:

1. `import`: In this stage, the reads coming from various files are labelled and stored in one file. The file employed by the next stage as the input.

2. `assemble-unitig`: The processes of k-mer generation, indexing, and de Bruijn graph construction are done in this stage. A new structure, called `unitig`, is introduced by HyDA. Unitigs are sub-contigs, that are condensed one-in one-out chains of k-mers, used by `assemble-finish` as the input.

3. `assemble-finish`: The graph constructed in `assemble-unitig` stage is imported and after error removal and total graph condensation, the final contigs are exported in one FASTA file as the assembly result.

4.1 Importing sequencing reads

HyDA accepts both FASTQ and FASTA files. It also accepts multiple read files and calls each file, or a pair of paired-read files, a library. The first step of HyDA is called import. In this stage, a number of FASTA or/and FASTQ files are imported to a single FASTA file that keeps all the reads with the information of the original library. Further processes such as eliminating read without enough information (like small reads) are done in this stage. In co-assembly, a unique color (value) is assigned to each dataset. Although all the reads of multiple files are imported to one file, a color can be assigned for a collection of them. That is, a number of libraries can be considered as one colored dataset and therefore, all the reads of those libraries are colored together.

4.2 Assemble-unitig

The second part of HyDA is called assemble-unitig. This stage asks the user to define some variables and provide some information. The user should define the number of colored datasets (which is a compile time constant $MAXCOLORS$) and specify every imported library belongs to which color. A simple config file that assigns a color to each imported library is required. If the config file is not defined, HyDA considers all the libraries as one unicolor dataset and all other colored datasets will be empty. Also, If the user does not specify the number of colors, the assembly will be done based on only one color.

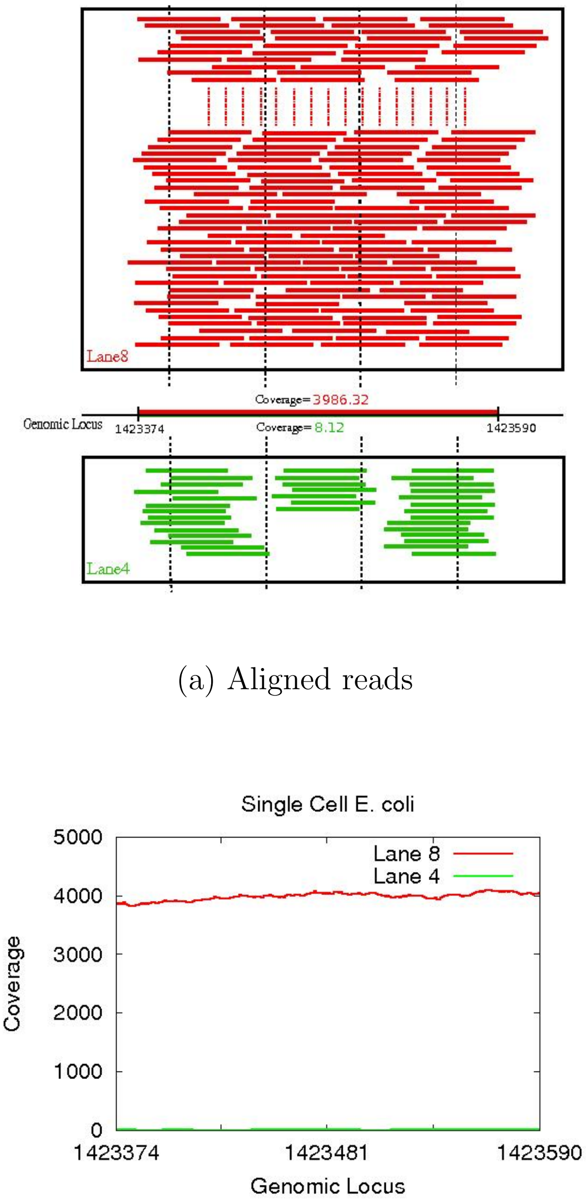
4.2.1 Indexing colored k-mers

A k-mer is stored in a 2-bit compressed form, for k up to a compile time constant $MAXK$. In- and out-edges of a k-mer are $(k+1)$-mers that are represented by their extremal left or right nucleotides. Therefore, in- and out-edges can be stored in a bit array of size 8 (one byte). Our de Bruijn graph of the input reads is stored in a hashed collection of splay trees whose vertices are k-mers with their associated data: colored counts, in- and out-edges, and internal flags. To save space which is the bottleneck for de Bruijn graph algorithms, we implemented splay tree in an array with relative rather than absolute pointers. Each k-mer is stored exactly once; therefore, this data structure defines a map from k-mers to data. In the case of double stranded sequences, which is the default mode, a k-mer and its dual (reverse complement) are treated as a single entry in the map. In that case, the minimum of the k-mer and its dual is stored in the map. The key factor of co-assembly is modifying a normal k-mer to colored one by converting an integer structure for multiplicity to an array of integers. The ith value of the multiplicity array represents the multiplicity of the k-mer in the ith colored dataset.

4.2.2 Condensation of the de Bruijn graph

A one-in one-out chain of k-mers is condensed into an equivalent long vertex which we call a *unitig*. A maximal unitig that cannot be extended further due to a branch in the graph is called a *contig* [33]. In the first step (`assemble-unitig`), our algorithm randomly picks one-in one-out k-mers in the de Bruijn graph, extends them into unitigs, and outputs those unitigs into a file. The remaining k-mers, which are not one-in one-out, are output as single k-mer unitigs. In the second step (`assemble-finish`), one-in one-out unitigs are condensed in the same manner into contigs.

Note that our condensation is solely based on the topology of the graph without any attention to the colorings. Ignoring colorings is particularly helpful for single cell MDA datasets; see Figure 4.1 for a case in which this strategy rescues a contig that is poorly covered in one dataset. Note that condensation of only similarly colored one-in one-out vertices would break some contigs into multiple pieces due to lack of coverage in some regions.



(a) Aligned reads

(b) Coverage per base pair

Figure 4.1: An *E. coli* contig (genomic locus 1,423,374-1,423,590) in Lanes 4 and 8 of single cell datasets [7]. It has low coverage (8.12×) in one dataset and high coverage (3986.32×) in the other. Our co-assembly algorithm rescues this low coverage contig.

4.2.3 Memory footprint reduction

We now describe our novel technique to partition the de Bruijn graph into multiple slices to reduce the peak memory requirement for each slice. The key idea is a fast hash function that assigns a slice to each k-mer. The simplest case is a naïve hash function that does not consider the adjacency of k-mers. On the other hand, we would ideally like a hash function that assigns all k-mers of a contig to the same slice.

Let $w < k$ be an integer; define the *minimizer* of a vertex (k-mer) to be the alphabetically minimum w-mer in the k-mer. The minimizers were previously used to reduce the number of required pairwise alignments in overlap-layout-consensus assembly approaches [30]. We use that idea for a different purpose. Let ω be the minimizer w-mer of k-mer κ. Let $s_{\max}$ be the number of slices into which the graph is partitioned. In that case, κ is assigned to the slice number

$$s(\kappa) = h(\omega) \bmod s_{\max},$$

in which h is an arbitrary uniform hash function. Since adjacent k-mers have identical minimizers with high probability, traversing the graph under this scheme imposes less slice-change overhead in comparison to the naïve approach. In HyDA, this slicing is performed in `assemble-unitig` and the output unitigs of all slices are simply merged to build the input unitigs to `assemble-finish`.

As long as w is large enough, this slicing scheme provides a roughly balanced partition. Note that a contig may still be broken into multiple subsequences in different slices unless w is very small. We observed that $w = 8$ gives a suitable tradeoff between partition balance and slice-contiguity both for bacterial and mammalian genome assembly; HyDA uses $w = 8$ by default.

The process of unitig generation is not influenced by the co-assembly modification, except that the colored coverage information of the unitigs is calculated for each color separately and stored in an array object.

4.3 Assemble-finish

The final stage of the assembly in HyDA is called `assemble-finish`. Error removal and condensation procedure is done in this stage. The maximum cutoff value, which is provided by user in the command line, is used to prune the graph in an iterative process.

4.3.1 Iterative error removal

The variable coverage cutoff, starting from 1 and gradually increasing, is used to remove contigs with low coverage. The low coverage contig removal process is iterated with increasing the cutoff in each round. In each iteration, those contigs whose maximum coverage over all the colors is less than the cutoff are eliminated, and the remaining graph is recondensed. This causes some contigs to merge into larger ones with recomputed average coverages. This process is similar to `Velvet-SC`'s low coverage contig removal, but instead of considering one average coverage, HyDA considers the *maximum* average coverage over all colors [7, 19]. In other words, a path is eliminated when its coverage for all colored dataset is less than the threshold. That is, a contig that is not covered well in one dataset is rescued when it is fully covered in another co-assembled dataset. We chose maximum instead of mean since the information of each color is important. For instance, mean would dilute the effect of one distinct genome among hundreds, and would cause the contigs of the distinct genome to be eliminated, whereas maximum preserves those contigs. Our results in Table 5.2 demonstrate that maximum performs well in eliminating erroneous contigs while it preserves well-covered outlier contigs, which could arise from distinct genomes, for further downstream analysis.

4.3.2 Output contigs

The output of iterative error removal is a collection of contigs (maximal unambiguous subsequences) that are obtained from condensing all chains of one-in one-out nodes.

Let C be the set of remaining contigs after error removal:

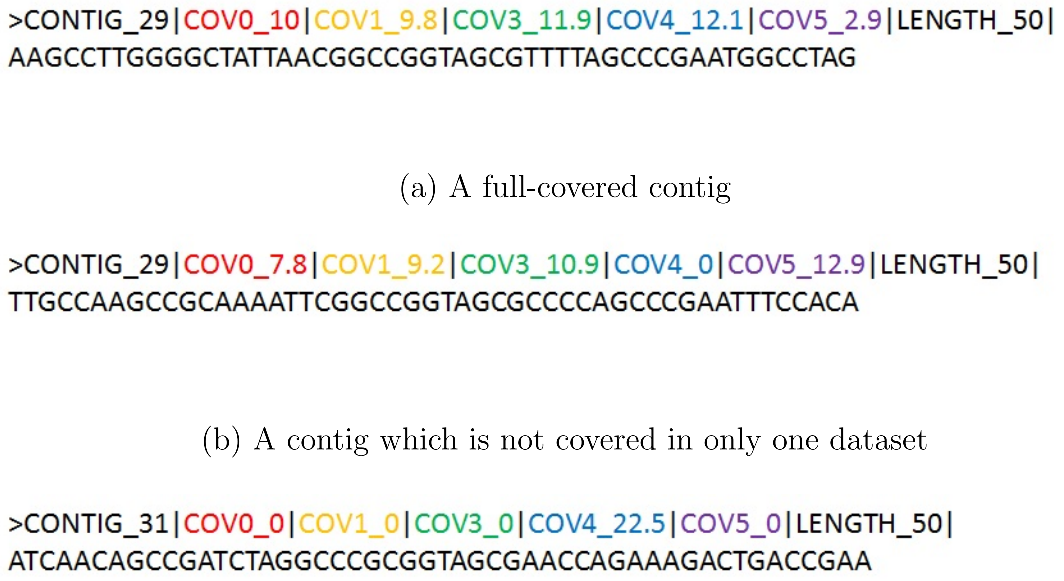
$$C = \{c_1, c_2, \cdots, c_n\}$$

Let $A_j(c_i)$ denote the average coverage of contig c_i in color j, for $1 \leq i \leq n$ and $1 \leq j \leq m$. Pick $\epsilon \geq 0$ and let $C_j = \{c_i \in C \mid A_j(c_i) > \epsilon\} \subset C$ be the set of contigs for each color j that are written in an individual FASTA file. The parameter ϵ determines the tradeoff between specificity and sensitivity. We chose $\epsilon = 0$ in this study, but a non-zero ϵ might be needed if there are erroneous or contaminant k-mers in one color which also occur in the true genomic sequence of another color.

The contigs are represented in a FASTA file, in which the information of each colored dataset is provided. The format of the contig representation in the output is illustrated in Figure 4.2.

4.3.3 Similarity check and pruning the assembly result

The final step is to verify that all the co-assembled cells are identical. As explained, each contig contains the information of its coverage for each colored dataset. If one of the co-assembled datasets belongs to another species, the majority of its contigs will be outlier. Also, the non-identical cell is the source of many contigs, which are not present in the identical cells. That is, if the majority of the contigs are not covered in all co-assembled datasets, the assumption that all the co-assembled cells are identical is wrong (most of the contigs that are aligned to the identical cells are not covered in the outlier datasets and majority of the outlier contigs are not present in the datasets of identical cells). When the cell equality test fails, the outlier datasets should be



```
>CONTIG_29|COV0_10|COV1_9.8|COV3_11.9|COV4_12.1|COV5_2.9|LENGTH_50|
AAGCCTTGGGGCTATTAACGGCCGGTAGCGTTTTAGCCCGAATGGCCTAG
```

(a) A full-covered contig

```
>CONTIG_29|COV0_7.8|COV1_9.2|COV3_10.9|COV4_0|COV5_12.9|LENGTH_50|
TTGCCAAGCCGCAAAATTCGGCCGGTAGCGCCCCAGCCCGAATTTCCACA
```

(b) A contig which is not covered in only one dataset

```
>CONTIG_31|COV0_0|COV1_0|COV3_0|COV4_22.5|COV5_0|LENGTH_50|
ATCAACAGCCGATCTAGGCCCGCGGTAGCGAACCAGAAAGACTGACCGAA
```

(c) A unicolored contig

Figure 4.2: Contigs generated by HyDA when 5 single cells are co-assembled. (a) represents a full-colored contig, which is covered in all co-assembled datasets. (b) represents a contig that is absent in only dataset 4. (c) represents a unicolored contig which is covered only in dataset 4.

identified and their contigs should be eliminated from the assembly result.

Figure 4.2 represents 3 contigs generated by co-assembly of some non-identical cells. Figure 4.2(a) displays a full-colored contig, which is covered in all datasets. Figure 4.2(b) illustrates a true contig, which exists in all identical cells datasets but is absent in the outlier dataset (lane 4). Figure 4.2(c) illustrates a false contig, which has to be eliminated from the output since it is generated by non-identical reads and does not exist in the identical datasets.

Let $S = \{s_1, s_2, s_3, \ldots, s_n\}$ be a set of n single cells that are assumed to be identical. Let $G = \{g_1, g_2, g_3, \ldots, g_m\}$ be the set of contigs generated by HyDA when it co-assembles single cells of set S. Let $A_j = \{a_{j1}, a_{j2}, \cdots, a_{jn}\}$, $1 \leq j \leq m$, be the set of colored average coverages of contig j. Contig g_j is called full-colored if:

$$\forall a_{ji} \in A_j : a_{ji} > 0$$
$$1 \leq i \leq n \text{ and } 1 \leq j \leq m$$

Contig g_j is called unicolored of color m if:

$$\exists a_{jk} \in A_j : \{a_{jk} > 0\}; 0 \leq k \leq n; 1 \leq j \leq m$$
$$\forall a_{ji} \in A_j : \{i \neq k; a_{ji} = 0\}; 0 \leq k \leq n; 0 \leq i \leq n; 1 \leq j \leq m$$

Assume s_i is the single cell that incorrectly assumed to be identical to the other single cells, i.e. it is in fact of another species. In this case, the majority of contigs covered in s_i dataset are unicolored contigs of color i.

After co-assembly, the number of full-colored contigs are calculated, which should be a high percentage of total contig in the case that all the co-assembled cells are identical. If a significant number of contigs are not full-colored, all the co-assembled colored datasets are examined for detecting the outlier cell, which is of another reference genome. For each dataset, the number of contigs covered in the dataset are considered and the ratio of unicolored to total contigs are calculated. The high ratio indicates the fact that the dataset is not identical to other datasets. Then all unicolored contigs of the outlier dataset are removed from the assembly result and the remaining contigs which are purely originated from the interested species are proposed as the assembly result.

Chapter 5: Experiments and Evaluation of Results

We used the datasets published by Chitsaz *et al.* [7] in which two *E. coli* single cells and one *S. aureus* single cell were isolated and their genomes were independently amplified with MDA. The resulting amplified DNA of one *E. coli* cell was sequenced in 4 lanes of Illumina GAIIx (Lanes 1-4), and the resulting amplified DNA of the other *E. coli* cell was sequenced in 3 lanes of Illumina GAIIx (Lanes 6-8). Therefore, we have 7 lanes of *E. coli* genomic MDA sequence from two biological replicates, in 100 bp long reads with an average coverage of $\sim 600\times$ per lane. The resulting amplified DNA of *S. aureus* was sequenced in 2 lanes of Illumina GAIIx with an average coverage of $\sim 1800\times$ per lane. We have two lanes of *S. aureus* each with three times the coverage of one *E. coli* lane.

5.1 Experiments

To display the strength of co-assembly method, we ran 5 different experiments using above data (see Figure 5.1). All the following datasets are assembled with $k = 55$:

- Single cell assembly of Lane 1: we used the `E+V-SC` single cell assembly of *E. coli* Lane 1 of [7].
- Mixed assembly of 7 *E. coli* lanes: we concatenated the reads in the 7 *E. coli* lanes and assembled the resulting aggregated dataset with `HyDA` using cutoff 400.
- Colored assembly of 7 *E. coli* lanes: we assigned a unique color to each of the 7 datasets and assembled them with `HyDA` using cutoff 100.
- Mixed unicolored assembly of 7 *E. coli* lanes and 2 *S. aureus* lanes: we concatenated the reads in all those lanes and assembled the resulting aggregated

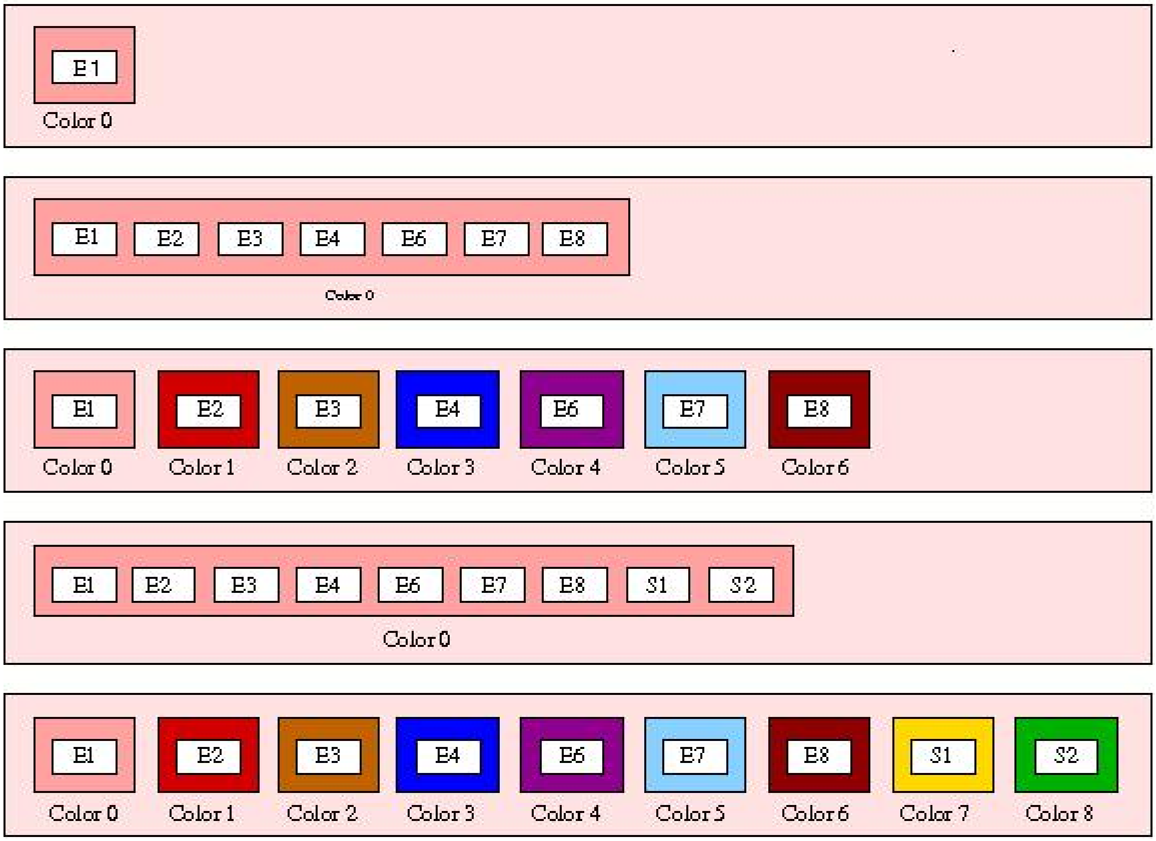


Figure 5.1: The five assembly scenarios: (i) single-cell assembly of *E. coli* lane 1; (ii) mixed monochromatic assembly of *E. coli* lanes 1-4 and 6-8, technical replicates of two biologically replicate single cells; (iii) multichromatic co-assembly of *E. coli* lanes 1-4 and 6-8; (iv) mixed monochromatic assembly of non-identical cells: *E. coli* lanes 1-4 and 6-8 and *S. aureus* lanes 7,8; (v) multichromatic co-assembly of nonidentical cells: *E. coli* lanes 1-4 and 6-8 and *S. aureus* lanes 7,8, each assigned a unique color

dataset with `HyDA`. We used a coverage cutoff of 400.

- Colored assembly of 7 *E. coli* lanes and 2 *S. aureus* lanes: we assigned a unique color to each of the 9 datasets and assembled them with `HyDA`. We used a coverage cutoff of 100. We then extracted *E. coli* contigs from the final assembly based on their coverage in the 7 *E. coli* colors.

	Total	Fully-Colored	Ratio
Identical	2124	1999	94 %
no-identical	3784	201	5 %

Table 5.1: The number of total contigs and fully-colored contigs (the contigs that are present in all co-assembled datasets) are displayed for two co-assembly experiments. (1) the assembly using 7 *E. coli* lanes (identical cells assembly), and (2) the assembly using 7 *E. coli* lanes and 2 *S. aureus* lanes (nonidentical cells assembly).
In the identical cells assembly, most of the contigs are existent in all datasets. On the other hand, The majority of the contigs in nonidentical cells assembly are not fully-colored.

5.2 Detecting outliers

The proposed method of assembling a genome from multiple single cells has two stages: (1) co-assembling the datasets, and (2) detecting the outlier datasets, which are of another species, and cleanse the assembly result from those outlier contigs (if exist). In the second stage (after the assembly), first, the identity of every pair of co-assembled datasets is assessed by calculating the ratio of fully-colored contigs to the total contigs. A high ratio verifies that all the co-assembled datasets are identical. In contrary, low ratio indicates that there is at least one co-assembled dataset that is of another species.

As displayed in Table 5.1 94% of contigs in the assembly of 7 *E. coli* lanes (Figure 5.1-iii) exist in all the datasets; while only 5% of the contigs in the assembly of 7 *E. coli* lanes and 2 *S. aureus* lanes (Figure 5.1-v) are fully-colored. In this case, all the lanes are examined and the outlier dataset is detected by calculating the number of their unicolored contigs (a datset with large number of unicolor contigs is an outlier). In the experiment we have done, the contigs are clustered into two sets: (1) contigs that are present only in *E. coli* lanes, which are 1922 contigs and (2) contigs that are present only in *S. aureus* lanes, which are 1654 contigs. Hence, only the contigs of datasets 1-7 (*E. coli* lanes) are saved and the contigs of datasets 8, 9 (*S. aureus* lanes) are thrown away.

5.3 Evaluation factors

To evaluate the quality of an assembly, many factors can be analyzed. Some of them can be calculated only in presence of the reference genome, while some others are more general and do not need the reference genome information. The most applied factors are:

- Total bases: total size of the generated contigs as the assembly result.
- Missing bases: total size of the reference regions that are not covered in the assembly result.
- Extra bases: total size of the assembly regions that are not aligned to the reference genome.
- N50: the contigs larger than which cover half of the assembly size. Although the N50 is an average type factor similar to a mean or median, longer contigs have a greater weight.
- NG50: the contigs larger than which cover half of the reference genome size. In other words, NG50 is similar to N50, only it considers the genome size instead of the assembly size.
- Corrected N50: N50 of maximal contiguous pieces of corrected contigs that align to the reference genome. The definition of corrected contigs are different for various evaluation tools. For example, GAGE [31] considers those contigs that align to the reference genome without ≥ 5 bp indels.

If the sequence of a reference genome is available, various assembly error factors can be examined:

- SNP: or Single Nucleotide Polymorphism is the mutation of one base.

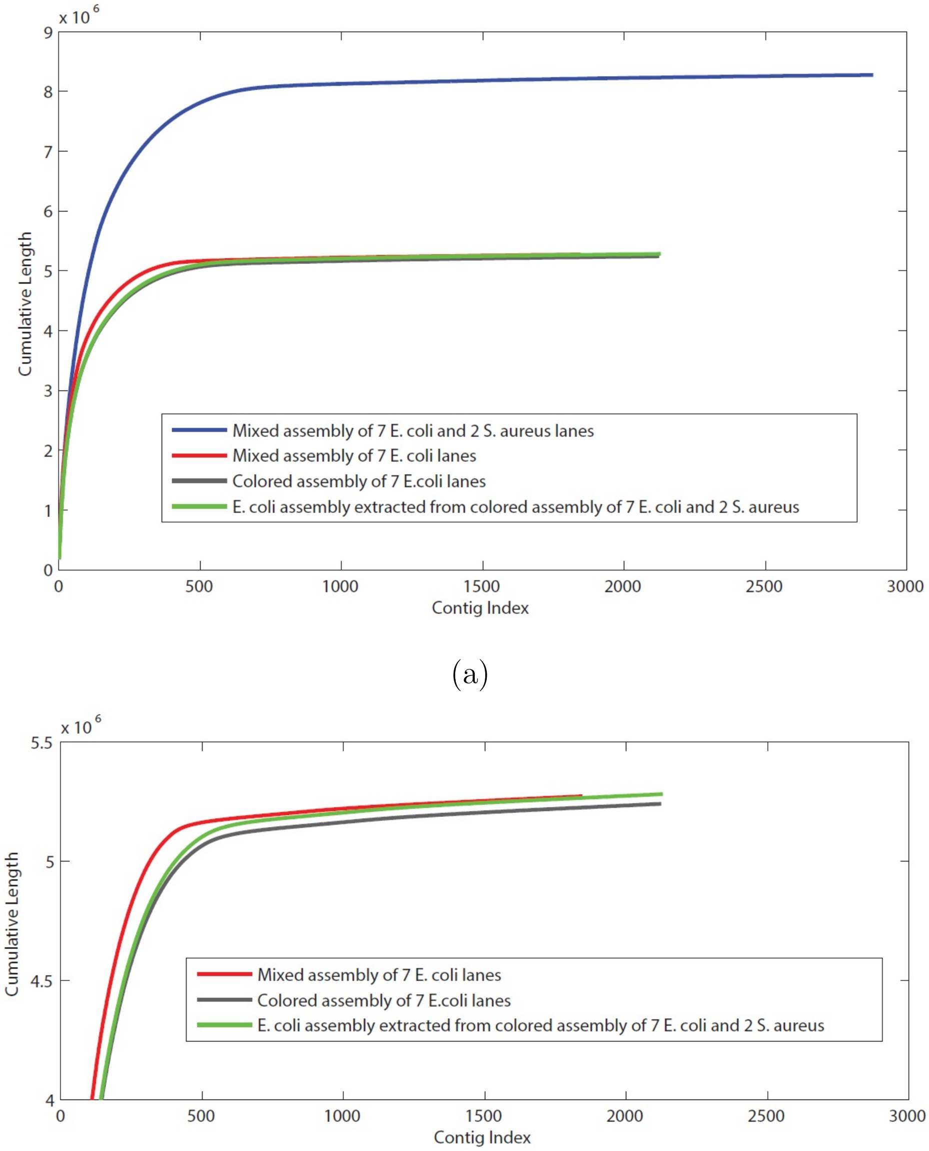
	E+V-SC	HyDA			
	Lane 1	Identical		Non-Identical	
	Single Cell	Mixed	Colored	Mixed	Colored
Assembly Size	4,570,583	5,272,627	5,240,693	8,274,855	5,281,487
Missing Ref Bases	281,060	**1,289**	2,114	**1,289**	2,114
Missing Ref (%)	6.06%	**0.03%**	0.05%	**0.03%**	0.05%
Extra Bases	279,721	659,982	615,808	3,662,149	656,771
Extra (%)	6.12%	12.59%	11.75%	44.26%	12.44%
N50	32,485	34,040	27,562	29,823	26,903
NG50	32,051	39,340	32,051	54,505	32,051
Corrected N50	30,094	**37,682**	31,445	**37,682**	31,445

Table 5.2: Comparison of GAGE [31] evaluation results for *E. coli* assemblies obtained from a single cell with E+V-SC (Lane 1 of [7]) and multiple identical (Lanes 1-4 and 6-8 of [7]) and non-identical cells (Lanes 1-4 and 6-8 plus two lanes of *S. aureus* in [7]) with HyDA in mixed and colored mode. GAGE's corrected N50 is the N50 of maximal contiguous pieces of contigs that align to *E. coli* K-12 reference genome without ≥ 5 bp indels [31]. *E. coli* contigs are extracted from the colored assembly of non-identical cells in a post-processing step based on their coverage in *E. coli* colors. Extra bases are due to contaminants, and that is why the number of extra bases in multicell assemblies is higher than that in single cell one [7]. The best results are shown in bold face. NG50 is the size of the contig the contigs larger than which cover half of the genome size [10]. All contigs are considered in all assemblies, particularly ultrashort single k-mer ones.

- Indel: short insertion or deletion.
- Relocation: the relocation of a contig within a chromosome.
- Translocation: the relocation of a contig between chromosomes.
- Inversion: reversed region of the true genome as part of a contig.

5.4 Evaluation of assemblies

We used GAGE [31] which is a standard assembly evaluation tool to compare the 4 assemblies described above in Section 5.3 and the E+V-SC single cell assembly of *E. coli* Lane 1 in [7]. The results are presented in Table 5.2. It is clear from



(a)

(b)

Figure 5.2: Comparison of contigs generated by mixed versus colored assembly for single cell *E. coli*. (a) contigs are those presented in Table 5.2 and are sorted from largest to smallest. The y axis shows the cumulative length. (b) a magnified portion of the plot in (a).

Table 5.2 that when we assemble a few identical cells the result is more accurate than the assembly of just one cell. The first column of Table 5.2 shows the evaluation results of an *E. coli* assembly using just one uncultivated cell. That assembly has 281,060 missing reference bases (about 6.06%), while another *E. coli* assembly using multiple identical cells has just 1,289 missing reference bases (about 0.03%) shown in the second column. Also, when some identical cells are used in assembly, other important factors such as SNPs, indels, and other variations improve significantly. Our results show that the single cell assembly has 100 SNPs and 34 indels $<$ 5 bps, whereas multicell assembly has only 5 SNPs and 6 indels $<$ 5 bps (see Table 5.3).

Mixed and colored assemblies have roughly similar characteristics. The second column of Table 5.2 represents the characteristics of mixed assembly when HyDA assembled a pool of 7 *E. coli* lanes coming from 2 biological replicates with cutoff 400. The third column is the colored assembly of the same dataset with cutoff 100. Both assemblies are missing less than 0.05% of reference genome bases and containing about 12% extra bases that are shown to be contaminants [7]. The only noticeable difference between the two assemblies is their N50, NG50, and corrected N50 [31, 10]. The N50 of the mixed assembly is 34,040 and its corrected N50 is 37,682, while the N50 and corrected N50 of the colored assembly are about 6 kbps less, at 27,562 and 31,445 respectively.

Although assembly of *E. coli* and *S. aureus* datasets contains many unaligned contigs, the evaluation error factors suach as SNP and Indel are the same as the assembly of identical datasets. Also, the mixed and colored assembly do not show any difference in term of the number of errors. Table 5.3 demonstrates that both methods generate 5 SNPs, 6-8 indels $<$ 5 bps, 4 indels $\geq$ 5 bps, 2-3 relocations, and zero inversions.

The only noticeable weakness of colored assembly is the size of contigs. The reason is that low multiplicity k-mers are iteratively eliminated. The multiplicity of a k-mer in the mixed dataset is the number of its occurrences in all reads from all cells, but the colored multiplicity of a k-mer is an array that keeps the multiplicity of each color separately. The maximum over all colors is computed for determining low coverage contigs in the colored case, whereas the mixed coverage is the sum of the colored coverages. That is why the cutoffs are different for the colored and mixed assemblies. This slight difference causes a mild difference in the N50 and NG50 of mixed and colored assemblies.

The colored assembly using 7 *E. coli* lanes and 2 *S. aureus* lanes coming from 2 uncultivated *E. coli* and one uncultivated *S. aureus* cells contained 3,784 contigs. The coverage in both *S. aureus* colors of 2,130 contigs was zero while most of their *E. coli* colors had positive coverage. On the other hand, the coverage in *E. coli* colors of 1,862 contigs was zero while both *S. aureus* colors had positive coverage. The final assembly of 7 *E. coli* and 2 *S. aureus* lanes with colored de Bruijn graph, when *S. aureus* contigs were eliminated, was similar to the colored assembly of 7 *E. coli* lanes; see Table 5.2. Provided that the input species are sufficiently different and MDA reactions cover a high portion of the genomes, we can easily find identical cells by clustering contigs. Performing the same experiment on more than 2 species is left as future work as it requires more data.

As mentioned above, the advantage of colored de Bruijn graph becomes clear when a combination of nonidentical cells is assembled. The last two columns of Table 5.2 represent the results of the mixed and the *E. coli* portion of the colored assembly of 7 *E. coli* and 2 *S. aureus* lanes. The mixed assembly covers the reference genome very well and just misses 1,289 bps of 4,639,675 bases (about 0.03%) of *E. coli* genome, but it contains 3,662,149 extra bases (about 44%) corresponding to *S. aureus* genome and contaminants. On the other hand, the *E. coli* portion of the colored assembly misses only 2,114 bps (about 0.05%) while it contains nearly one sixth extra bases (526,119 bps) which are only contaminant [7].

	E+V-SC	HyDA	
	Single Cell	Mixed	Colored
SNPs	100	5	5
Indels < 5 bp	34	8	6
Indels ≥ 5 bp	4	4	4
Inversions	0	0	0
Relocations	4	2	3

Table 5.3: Comparison between various types of error generated in an assembly of one single cell by E+V-SC and HyDA in mixed and colored modes. For both mixed and colored assembly, using the data of 7 identical *E. coli* single cells and 9 single cell of *E. coli* and *S. aureus* generates the same errors.

The only weakness of the co-assembly method is related to the problem of contiguity loss. True genomic variation in the identical cells generates branches in the colored de Bruijn graph, which makes contigs shorter. The comparison between contigs generated by co-assembly and normal method is illustrated in Figure 5.2.

Chapter 6: Discussion and Conclusion

We propose an algorithm in the field of single cell sequencing for the improvement of the genome assembly especially for extracting those bases that cannot be rescued by other existing tools. The algorithm is implemented in HyDA, a *de novo* co-assembly tool of bacterial genomes from multiple single cells, which is based on colored de Bruijn graph [12] in which each input dataset is annotated by a unique color. Using HyDA, we showed that co-assembly of bacterial genomes from multiple single cells outperforms single cell assembly of each individual one in all metrics.

Moreover, co-assembly outperforms mixed assembly in all metrics except contiguity. The strength of our method is its sensibility when a genome is assembled using a number of single cell datasets including datasets of another species. In this case, mixed assembly generates a chimeric result while co-assembly produces a high quality assembly by eliminating the false contigs originated from outlier datasets. In the case of read datasets of identical single cells, the mixed and colored assembly generate similar results in all metrics except contiguity (which is related to the nature of the colored de Bruijn graph). To mitigate this mild contiguity loss, we suggest a mixed reassembly of those datasets that fall into the same cluster, i.e. identified as the same species, after clustering the contigs based on their colored coverage.

6.0.1 Future Work

The proposed algorithm is one application of co-assembly method in the single cell sequencing field by using a number of guessed to be identical single cell datasets. The next step of this work is acceptance of single cell datasets from various species. For this purpose, some adaptation strategies and new features are required. For example, a normalization for cut off coverage is essential in the case of datasets with different average coverage. Moreover, a post-correction is needed to refine the assembly of each lane from suspicious contigs. Furthermore, the problem of contiguity loss can be solved by implementation of variable k colored graph.

REFERENCES

[1] Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., Pyshkin, A. V., Sirotkin, A. V., Vyahhi, N., Tesler, G., Alekseyev, M. A., and Pevzner, P. A. SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *J. Comput. Biol. 19*, 5 (May 2012), 455–477.

[2] Batzoglou, S., Jaffe, D. B., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J. P., and Lander, E. S. ARACHNE: a whole-genome shotgun assembler. *Genome Res. 12* (Jan 2002), 177–189.

[3] Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I. A., Belmonte, M. K., Lander, E. S., Nusbaum, C., and Jaffe, D. B. ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res. 18* (May 2008), 810–820.

[4] Caporaso, J. G., Lauber, C. L., Walters, W. A., Berg-Lyons, D., Huntley, J., Fierer, N., Owens, S. M., Betley, J., Fraser, L., Bauer, M., Gormley, N., Gilbert, J. A., Smith, G., and Knight, R. Ultra-high-throughput microbial community analysis on the Illumina HiSeq and MiSeq platforms. *ISME J 6*, 8 (Mar 2012), 1621–1624.

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

[6] Chaisson, M. J., and Pevzner, P. A. Short read fragment assembly of bacterial genomes. *Genome Res. 18* (Feb 2008), 324–330.

[7] Chitsaz, H., Yee-Greenbaum, J. L., Tesler, G., Lombardo, M.-J., Dupont, C. L., Badger, J. H., Novotny, M., Rusch, D. B., Fraser, L. J., Gormley, N. A., Schulz-Trieglaff, O., Smith, G. P., Evers, D. J., Pevzner, P. A., and Lasken, R. S. Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. *Nature Biotech 29*, 10 (Oct 2011), 915–921.

[8] Dean, F. B., Hosono, S., Fang, L., Wu, X., Faruqi, A. F., Bray-Ward, P., Sun, Z., Zong, Q., Du, Y., Du, J., Driscoll, M., Song, W., Kingsmore, S. F., Egholm, M., and Lasken, R. S. Comprehensive human genome amplification using multiple displacement amplification. *Proc. Natl. Acad. Sci. U.S.A. 99* (Apr 2002), 5261–5266.

[9] Dean, F. B., Nelson, J. R., Giesler, T. L., and Lasken, R. S. Rapid amplification of plasmid and phage DNA using Phi 29 DNA polymerase and multiply-primed rolling circle amplification. *Genome Res. 11* (Jun 2001), 1095–1099.

[10] Earl, D., Bradnam, K., St John, J., Darling, A., Lin, D., Fass, J., Yu, H. O., Buffalo, V., Zerbino, D. R., Diekhans, M., Nguyen, N., Ariyaratne, P. N., Sung, W. K., Ning, Z., Haimel, M., Simpson, J. T., Fonseca, N. A., Birol, I., Docking, T. R., Ho, I. Y., Rokhsar, D. S., Chikhi, R., Lavenier, D., Chapuis, G., Naquin, D., Maillet, N., Schatz, M. C., Kelley, D. R., Phillippy, A. M., Koren, S., Yang, S. P., Wu, W., Chou, W. C., Srivastava, A., Shaw, T. I., Ruby, J. G., Skewes-Cox, P., Betegon, M., Dimon, M. T., Solovyev, V., Seledtsov, I., Kosarev, P., Vorobyev, D., Ramirez-Gonzalez, R., Leggett, R., Maclean, D., Xia, F., Luo, R., Li, Z., Xie, Y., Liu, B., Gnerre, S., Maccallum, I., Przybylski, D., Ribeiro, F. J., Yin,

S., Sharpe, T., Hall, G., Kersey, P. J., Durbin, R., Jackman, S. D., Chapman, J. A., Huang, X., Derisi, J. L., Caccamo, M., Li, Y., Jaffe, D. B., Green, R. E., Haussler, D., Korf, I., and Paten, B. Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Res. 21* (Dec 2011), 2224–2241.

[11] Fleischmann, R. D., Adams, M. D., White, O., Clayton, R. A., Kirkness, E. F., Kerlavage, A. R., Bult, C. J., Tomb, J.-F., Dougherty, B. A., Merrick, J. M., et al. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science 269*, 5223 (1995), 496–512.

[12] Iqbal, Z., Caccamo, M., Turner, I., Flicek, P., and McVean, G. De novo assembly and genotyping of variants using colored de bruijn graphs. *Nat Genetics 44* (2012), 226–232.

[13] Ishoey, T., Woyke, T., Stepanauskas, R., Novotny, M., and Lasken, R. S. Genomic sequencing of single microbial cells from environmental samples. *Curr. Opin. Microbiol. 11* (Jun 2008), 198–204.

[14] Islam, S., Kjällquist, U., Moliner, A., Zajac, P., Fan, J.-B., Lönnerberg, P., and Linnarsson, S. Characterization of the single-cell transcriptional landscape by highly multiplex rna-seq. *Genome research 21*, 7 (2011), 1160–1167.

[15] Lage, J. M., Leamon, J. H., Pejovic, T., Hamann, S., Lacey, M., Dillon, D., Segraves, R., Vossbrinck, B., González, A., Pinkel, D., et al. Whole genome analysis of genetic alterations in small dna samples using hyperbranched strand displacement amplification and array–cgh. *Genome Research 13*, 2 (2003), 294–307.

[16] Lasken, R. S. Single-cell genomic sequencing using Multiple Displacement Amplification. *Curr. Opin. Microbiol. 10* (Oct 2007), 510–516.

[17] Lasken, R. S., and Egholm, M. Whole genome amplification: abundant supplies of DNA from precious samples or clinical specimens. *Trends in Biotechnology 21* (Dec 2003), 531–535.

[18] Loman, N. J., Constantinidou, C., Chan, J. Z., Halachev, M., Sergeant, M., Penn, C. W., Robinson, E. R., and Pallen, M. J. High-throughput bacterial genome sequencing: an embarrassment of choice, a world of opportunity. *Nature Reviews Microbiology 10*, 9 (2012), 599–606.

[19] Movahedi, N. S., Forouzmand, E., and Chitsaz, H. De novo co-assembly of bacterial genomes from multiple single cells. In *IEEE Conference on Bioinformatics and Biomedicine* (2012), pp. 561–565.

[20] Myers, E. W. The fragment assembly string graph. *Bioinformatics 21*, suppl 2 (2005), ii79–ii85.

[21] Nature. Method of the year 2013. http://www.nature.com/nmeth/journal/v11/n1/full/nmeth.28 2001.

[22] Navin, N., Kendall, J., Troge, J., Andrews, P., Rodgers, L., McIndoo, J., Cook, K., Stepansky, A., Levy, D., Esposito, D., et al. Tumour evolution inferred by single-cell sequencing. *Nature 472*, 7341 (2011), 90–94.

[23] Nijkamp, J. F., van den Broek, M. A., Geertman, J.-M. A., Reinders, M. J., Daran, J.-M. G., and de Ridder, D. De novo detection of copy number variation by co-assembly. *Bioinformatics 28*, 24 (2012), 3195–3202.

[24] Peng, Y., Leung, H. C., Yiu, S.-M., and Chin, F. Y. Idba–a practical iterative de bruijn graph de novo assembler. In *Research in Computational Molecular Biology* (2010), Springer, pp. 426–440.

[25] Peng, Y., Leung, H. C., Yiu, S. M., and Chin, F. Y. IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics 28*, 11 (Jun 2012), 1420–1428.

[26] Pevzner, P. A., Tang, H., and Tesler, G. De novo repeat classification and fragment assembly. *Genome research 14*, 9 (2004), 1786–1796.

[27] Pevzner, P. A., Tang, H., and Waterman, M. S. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U.S.A. 98* (Aug 2001), 9748–9753.

[28] Pop, M. Genome assembly reborn: recent computational challenges. *Briefings in bioinformatics 10*, 4 (2009), 354–366.

[29] Raghunathan, A., Ferguson, H. R., Bornarth, C. J., Song, W., Driscoll, M., and Lasken, R. S. Genomic DNA amplification from a single bacterium. *Appl. Environ. Microbiol. 71* (Jun 2005), 3342–3347.

[30] Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M., and Yorke, J. A. Reducing storage requirements for biological sequence comparison. *Bioinformatics 20* (Dec 2004), 3363–3369.

[31] Salzberg, S. L., Phillippy, A. M., Zimin, A., Puiu, D., Magoc, T., Koren, S., Treangen, T. J., Schatz, M. C., Delcher, A. L., Roberts, M., Marcais, G., Pop, M., and Yorke, J. A. GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Res. 22*, 3 (Mar 2012), 557–567.

[32] Simpson, J. T., Wong, K., Jackman, S. D., Schein, J. E., Jones, S. J., and Birol, I. ABySS: a parallel assembler for short read sequence data. *Genome Res. 19* (Jun 2009), 1117–1123.

[33] Zerbino, D. R., and Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res. 18* (May 2008), 821–829.

[34] Zong, C., Lu, S., Chapman, A. R., and Xie, X. S. Genome-wide detection of single-nucleotide and copy-number variations of a single human cell. *Science 338*, 6114 (2012), 1622–1626.

ABSTRACT

De Novo Co-assembly of Bacterial Genomes from Multiple Single Cells

by

Narjes Sadat Movahedi Tabrizi

August 2014

Advisor: Dr. Hamidreza Chitsaz

Major: Computer Science

Degree: Master of Science

Recent progress in DNA amplification techniques, particularly multiple displacement amplification (MDA), has made it possible to sequence and assemble bacterial genomes from a single cell. However, the quality of single cell genome assembly has not yet reached the quality of normal multicell genome assembly due to the coverage bias and errors caused by MDA. Using a template of more than one cell for MDA or combining separate MDA products has been shown to improve the result of genome assembly from few single cells, but providing identical single cells, as a necessary step for these approaches, is a challenge. As a solution to this problem, we give an algorithm for *de novo* co-assembly of bacterial genomes from multiple single cells. Our novel method not only detects the outlier cells in a pool, it also identifies and eliminates their genomic sequences from the final assembly. Our proposed co-assembly algorithm is based on colored de Bruijn graph which has been recently proposed for *de novo* structural variation detection. Our results show that *de novo* co-assembly of bacterial genomes from multiple single cells outperforms single cell assembly of each individual one in all standard metrics. Moreover, our *de novo* co-assembly also outperforms the mixed assembly in which the input datasets are simply concatenated. We implemented our algorithm in a software tool called HyDA which is available from `http://chitsazlab.org/software/hyda`.

AUTOBIOGRAPHICAL STATEMENT

Narjes Sadat Movahedi Tabrizi

EDUCATION

- Master of Science (Computer Science), 2014
  Wayne State University, Detroit, MI, USA

- Bachelor of Electrical Engineering (communication Systems Engineering), 2005
  Iran University of Science and Technology, Tehran, Iran

PUBLICATIONS

1. Movahedi, NS. Forouzmand, ES., and Chitsaz, H. *De novo* co-assembly of bacterial genomes from multiple single cells. In *BIBM 2012 (Philadephia, Pennsylvania)* (2012).

2. Taghavi, Z., Movahedi, N. S., Drghici, S., and Chitsaz, H. Distilled single-cell genome sequencing and *De novo* assembly for sparse microbial communities. In *Bioinformatics* , 29(19), 2395-2401. (2013).

3. Embree, M., Nagarajan, H., Movahedi, N., Chitsaz, H., and Zengler, K. Single-cell genome and metatranscriptome sequencing reveal metabolic interactions of an alkane-degrading methanogenic community. In *The ISME journal* (2013).

4. Movahedi, NS., Taghavi, Z., Embree, M., Nagarajan, H., Zengler, K., and Chitsaz, H. Efficient synergistic single-cell genome assembly. In *bioRxiv* and under revision at *Genome Biology* (2014).