1-1-2015

Efficient Synergistic De Novo Co-Assembly Of Bacterial Genomes From Single Cells Using Colored De Bruijn Graph

Narjes Sadat Movahedi Tabrizi
Wayne State University,

EFFICIENT SYNERGISTIC DE NOVO CO-ASSEMBLY OF BACTERIAL GENOMES FROM SINGLE CELLS USING COLORED DE BRUIJN GRAPH

by

NARJES SADAT MOVAHEDI TABRIZI

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2015

MAJOR: COMPUTER SCIENCE

Approved By:

Advisor	Date

DEDICATION

To my family.

ACKNOWLEDGEMENTS

I am heartily thankful to my advisor, Prof. Dongxiao Zhu, for his extensive support and the opportunity he provided for me to conclude my research. His encouragements and patronage inspired me. His dedication to conduct excellent and exceptionally work has always motivated me. I would like to thank my former advisor, Prof. Hamidreza Chitsaz, for his for guidance and encouragement. This research was supported in part by NSF grant DBI-1262565 under his name. I am also grateful to the members of my dissertation committee Dr. Xue-wen Chen, Dr. Farshad Fotouhi, and Dr. Chuanzhu Fan for accepting to be on my dissertation committee. I want to thank Prof. Fotouhi unconditional help, advice, and support, which make me conduct my dissertation work. I want to acknowledge Dr. Zeinab Taghavi, who helped me with this project. I like to acknowledge students and research fellow in Chitsaz lab for being such great colleagues.

I would also like to thank all my friends in Ann Arbor who made my life far away from home enjoyable; especially the ones who commute with me between Ann Arbor and Detroit.

I have been blessed with unconditional and endless love of my family. They have always been supportive of me. Finally, my love, Mahdi, I owe him a great debt of gratitude for his wonderful companionship. I am fortunate to have him in my life. Without him understanding, help and support I would not be able to complete this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Enormous progress towards ubiquitous DNA sequencing has brought a realm of exciting applications within reach, including genomic analysis at single cell resolution. Single cell genome sequencing holds great promise for various areas of biology including environmental biology [12]. In particular, myriad unculturable environmental microorganisms have been studied using single cell genome sequencing powered by high throughput DNA amplification methods [13, 14, 15, 16, 17]. Since the majority of microbes to date are unculturable, single cell sequencing has enabled significant progress in elucidating the genome sequences and metabolic capabilities of these previously inaccessible microorganisms.

Multiple Displacement Amplification (MDA) is the preferred amplification method for single cell sequencing, since it is an isothermal (without thermo cycling) process as opposed to PCR [18, 19]. Compared to PCR-based amplification methods, it produces less amplification coverage bias and error [20, 21, 22].

Although single cell sequencing methods have passed important milestones, such as capturing $\geq 90\%$ of genes in a prokaryotic cell [1] or finishing and closing the genome of a prokaryote using MDA [23], the quality and reliability of genome assemblies from single cells lag behind those of sequencing methods from multi cells due to a bias arising from MDA. The main factors that affect quality are uneven coverage depth and the absence of scattered chunks of the genome in the final collection of reads. There is no known deterministic pattern for the preferred amplified regions, and they are currently treated as the result of a random process. Also, the outcome of MDA is widely variable ranging from total loss of the sample and any information therein to nearly complete reconstruction of the genome. In this sense,

an MDA-based single cell sequencing experiment is currently a gamble that can potentially lead to the loss of the sample and sequencing expenses.

Recently, a new whole genome amplification method was demonstrated on individual human cells, which is called Multiple-Annealing and Looping-Based Amplification Cycles (MALBAC) [11, 24]. The MALBAC result with human genome demonstrates more uniform than MDA result. Despite the improvements achieved by MALBAC, the bias of whole genome amplification methods is far from the desirable [25].

We demonstrate in this work how to hedge against this risk through sequencing and co-assembly of few single cells. Our method replaces a single cell deep sequencing experiment with multiple single cell shallow sequencing experiments, allowing for the acquisition of information about multiple single cells simultaneously.

1.1 Background

In this section, we briefly introduce the background about the genome assembly, especially the assembly from a single cell dataset.

Genome or 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. The first bacterial genome was sequenced using conventional Sanger sequencing data in 1995 [26]. 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 [27]. 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.

1.1.1 *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 $\{0, 1\}$ [28].

In the field of genomic:

- 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

The first two algorithms are not suitable for the data generated by Next Generation Sequencing platforms, which are short reads with high coverage. String graph was proposed as an alternative method for de Bruijn (which is currently the most applied method). These algorithms are described briefly.

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 Next Generation Sequencing 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 [29].

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 [30].

String Graph

String graph, another algorithm for genomic assembly was introduced by Myers [31] 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.

De Bruijn Graph

The most applied method for *de novo* assembly using short reads generated by High Throughput Sequencing (Next Generation Sequencing) platforms is done by constructing a de Bruijn graph [32, 33]. Each vertex of the graph represents a length-k subsequence, called k-mer. A set of k-mers is generated for each sequencing read. Two neighbour k-mers (nodes) are connected by an edge. After graph construction, each non-branching path is condensed into one supernode whose sequence is the merged sequence of all the nodes in the path. Each supernode is called a *contig*. The output of the assembly process using de Bruijn graph is the set of generated contigs.

1.1.2 Single Cell Sequencing

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) [34] 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% [35].

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 at the end of 2013 as its "`Method of the Year`" [36].

Amplification Methods

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) [15, 13, 37]. MDA works using random primers and a special DNA polymerase called Φ29 with interesting characteristics. The enzyme Φ29 is able to open up double stranded DNA and continue its way without external thermal help if it encounters a double stranded region

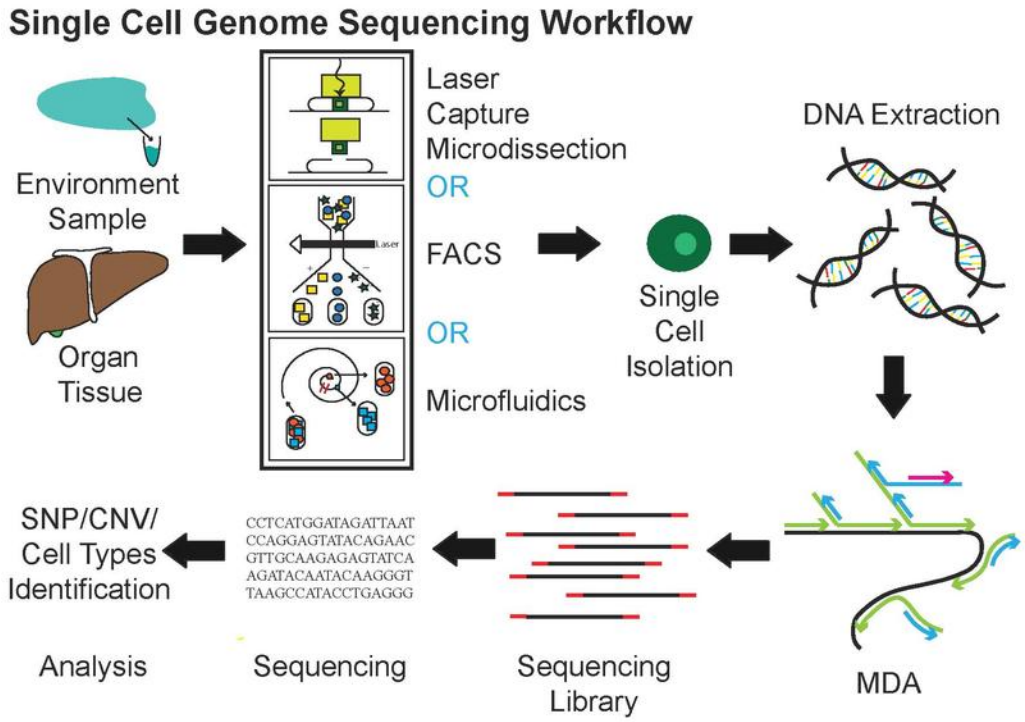


Figure 1.1: The workflow of single cell sequencing using Multiple Displacement Amplification (MDA) technique is illustrated [10]

while it is synthesizing its complementary strand [35, 37]. This unique property of Φ29 makes MDA an isothermal reaction that does not suffer from the side effects of thermocycling such as GC-bias as opposed to PCR. The workflow of single cell sequencing using MDA technique is displayed in Fig 1.1. 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 [38, 11], 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.

Recently, a new amplification method has been proposed on individual human genome, which is called multiple annealing and looping-based amplification cycles (MALBAC) [11], which reports more uniform coverage on human genome (See Figure 1.2). MALBAC coverage of the human genome has less bias than that of MDA. Nevertheless, amplification bias is still a challenge despite the improvements achieved by MALBAC [14].

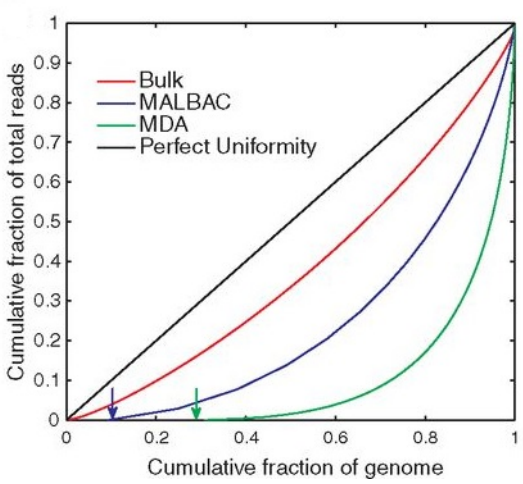


Figure 1.2: Lorenz curves of reads over the entirety of chromosome 1 (chr1) of a single cell from the SW480 cancer cell generated by MALBAC, MDA, and the bulk sample. A diagonal line indicates the perfectly uniform coverage. The deviation from the diagonal indicates the coverage bias. All samples are sequenced at 25x depth. [11]

Furthermore, sensitivity of MALBAC to background noise makes it not suitable for many applications, such as de novo assembly [8]. More detailed study is required to address the amplification bias reduction by this method [36]. 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 currently used and thoroughly evaluated methods [35].

Computational Solution

Fortunately, a computational solution to overcome the MDA bias was proposed by Chitsaz *et al.* [1] in 2011, which handles the un-even depth of coverage challenge. The method is applied by all single cell sequencing tools such as SPAdes [5] and IDBA-UD [9].

Today, the possibility of sequencing a species using one isolated cell is relying on the experimental achievements [39, 40, 41] and the computational solution proposed by Chitsaz *et al.* [1]. 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 [42, 43]. 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 [44].

None of those solutions worked properly for unevenly covered genomes sequenced by single cell sequencing methods. Velvet-SC [1] 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 cut-off value. The cut-off 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. Velvet-SC is followed by other single cell sequencing tools such as SPAdes [5] and IDBA-UD [9]. The main achievement of the method is related to removing errors while saving low-covered bases in the single cell datasets.

1.2 Related Works

To date, colored graph and co-assembly is employed in calling genomic structural variations. Cortex [45] 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 multi-cell datasets and use the information of known coverage distribution, the challenges related to un-even depth of coverage and blackout regions, which are introduced by single cell sequencing technology, are not noticed. Although, Cortex is the only assembly tool that does co-assembly process using colored de Bruijn graph, there is another tool, Magnolya

[46], 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 un-even depth of coverage characteristic. The first work, Velvet-SC and E+V-SC, was proposed by Chitsaz *et al.* [1] , which has been followed by other single cell assembly tools (all employ de Bruijn graph) such as, SPAdes [5] and IDBA-UD [9].

The assembler we employ in this work is HyDA [47], which is a modified version of simple Velvet-SC, the first tool for single cell sequencing. Other single cell assembly tools follow Velvet-SC and apply some further strategies to improve assembly their results. Each new method employed by these state of art assemblers is a module that can be implemented in other assembly tools (including HyDA). Using read error correction, iterative assembly with variable k, 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. We expect the incorporation of our colored assembly method into these state of art assembly tools to improve their assembly result. Also, implementation of the method applied by them is listed as future work to improve our algorithm.

1.3 Our Research Goals

In this section, we present the goals of this dissertation research. We summarize our progress toward these goals.

The single cell sequencing field suffers from the bias generated in amplification stage, which is an essential part of single cell sequencing process. The bias on the coverage of single cell read dataset, is an important challenge, which can be mitigated by computational

methods. One bias that has not been noticed by current computational solution is the existence of blackout regions (which do not get coverage during the amplification process). Even omitting one base of a gene breaks the sequence into two parts and the information gets lost. To assemble a genome whose reference is unknown, we need to cover these blackout regions as much as possible.

We propose an elegant solution to this problem that is the co-assembly a number of single cell datasets and fill the gaps due to amplification bias for each dataset using the information exist in other co-assembled datasets. We use the idea of colored de Bruijn graph [45], which was proposed in a different context (for structural variation detection and genotyping using multi-cell datasets), and designed our graph (which notices the single cell dataset characteristic) and implemented it in a tool, named HyDA: HyBrid *De Novo* Assembler.

To have a high quality assembly, which covers most of the genome, using a number of read dataset of identical cells is suggested. However, if only one of the dataset is of another genome, the result is a chimeric assembly containing the information of another genome. Obviously, identifying the outlier datasets is not possible when the reference genome is unknown. Our algorithm 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.

We modified our algorithm and extended HyDA to make it suitable for assembling various genomes of phylogenetically close species simultaneously. Our algorithm can fill the blackout regions of each genome using the information of other co-assembled genomes. Since phylogenetically close species share many coding sequences, there is a high chance that a blackout region in one genome is existent and covered in other co-assembled datasets. Our algorithm also can compare each two co-assembled datasets based on their genomic similarity and decide whether they are of the same reference genome, in which their assembly result can be combined and counted as one genome assembly. In other words, besides detecting an

outlier among some identical datasets, our algorithm can cluster the various co-assembled datasets based on their genome.

Although colored de Bruijn graph can retrieve the blackout regions in single cell datasets, it looses some contiguity information. Using the idea of iterative assembly, which is employed by two state of art single cell assembly tools, SPAdes [5] and IDBA-UD [9], to increase the contigs length, we designed and implemented the iterative co-assembly algorithm and proposed an advance version of HyDA. Iterative co-assembly can generate a high quality genome assembly with minimum missing bases and contiguity loss.

1.4 Organization

The rest of the document is organized as follows. In Chapter 2, we describe the basic of genome co-assembly using colored de Bruijn graph and the implemented algorithm in our assembly tool, HyDA. In chapter 3, we describe one application of co-assembly, which is the *de novo* assembly of a genome from multiple single cells datasets. We also, provide the evaluation result of the experiments that have been done. In Chapter 4, we describe the ability of HyDA to do synergistic assembly of various genomes of phylogenetically close species and determine the relationship between any two co-assembled input datasets. In Chapter 5, we present the advance version of HyDA, which works based on iterative assembly using various k. Finally, in chapter 6, we conclude the report and propose the future works.

2 COLORED DE BRUIJN GRAPH

2.1 Introduction

To address the problem of blackout regions in the single cell datasets, we employ the idea of colored de Bruijn graph, an augmented and progressed version of de Bruijn graph [32, 33]. Colored de Bruijn graph was proposed for genotyping and structural variation detection for mammalian genomes by Cortex [45] in which the coverage is modelled as over-dispersed Poisson distribution. Hence, the datasets of single cells, which have non-uniform coverage with unknown distribution due to amplification bias, are not suitable candidates for Cortex and its colored de Bruijn graph.

Colored de Bruijn graph is a modified version of normal de Bruijn graph [32, 33], which is the most applied method for *de novo* assembly using short reads generated by High Throughput Sequencing platforms. Some well-known assembly tools using de Bruijn graph are AbySS [48], Velvet [44], Euler-SR [42, 49], and AllPaths [50]. The assembly using de Bruijn graph is done by employing a structure called k-mer. A k-mer is a k length subsequence of a sequencing read whose sequence has $k-1$ bases overlap to the previous and next k-mer's sequence, in which only the last base of a k-mer and last character of its successor's sequence are different. For each sequencing read of size l bases, $l-k+1$ k-mers are generated. Figure 2.1 displays the 3-mers of a sequencing read.

The graph is constructed using the k-mers, in which the graph vertices are the k-mers and the edges between two nodes represent the adjacency of the corresponding k-mers. Besides the sequence information, each node of the de Bruijn graph consists the coverage

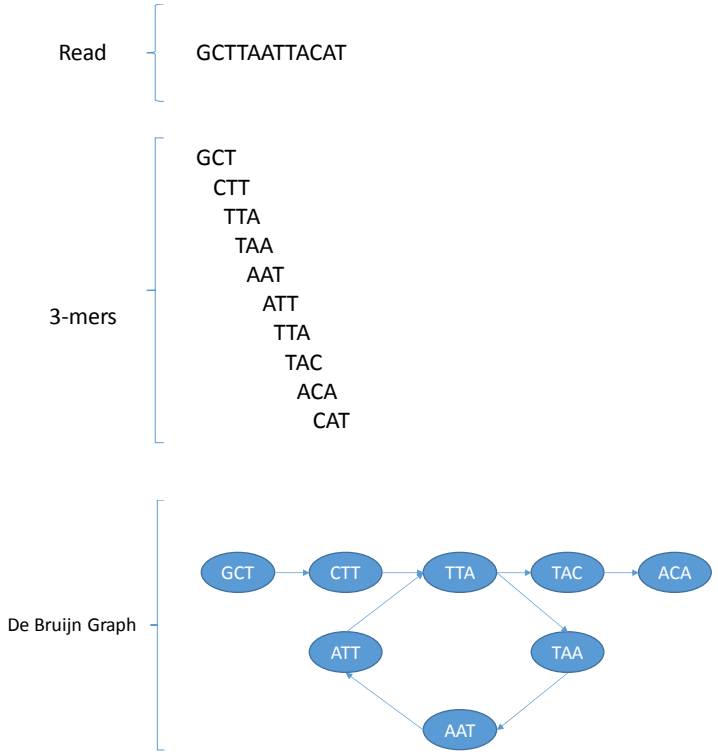


Figure 2.1: The construction of de Bruijn graph for a given read with $k = 3$ is illustrated. Ten 3-mers are generated for the 12 character long sequencing read. Each 3-mer is represented by a node in the graph. Two consecutive 3-mers are connected by a directed edge. A node can represent more than one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

information of the sequence, which is the number that the sequence is appeared in input sequencing reads (input dataset). Figure 2.1 illustrates the de Bruijn graph of a 12 bases length sequencing read. In the first step, 10 3-mers are generated and then these 3-mers are represented by the graph vertices. An edge is placed between any two consecutive 3-mers. Note that the coverage value of node AAT is 2 since the subsequence AAT is appeared in the read twice. Also, the in-degree and out-degree of this node are 2 since each of two 3-mers AAT has its own neighbours and the adjutancy information of each should be noticed in the de Bruijn graph by proper edges.

After graph construction, all the one-in-one-out nodes in each path are condensed together and generate a supernode (condensed node). These condensed nodes are the assembly result, which are called *contigs*. The Sub-sequence repeats in the input dataset increase the coverage of their corresponding k-mers and increment their incoming/outgoing degree. The generated branches are the sides of one-in-one-out paths. Hence, having more repeats causes more branches and consequently smaller contigs.

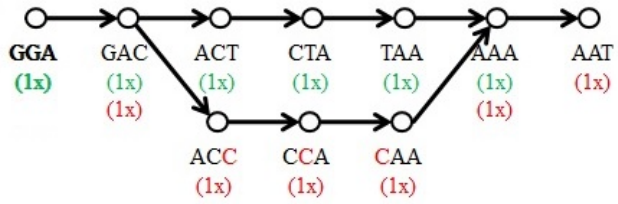
We propose colored de Bruijn graph for assembly of genome using single cell read dataset with non-uniform coverage. The strength of the proposed method is related to the fact that the assembly of each dataset is improved by using the information of the other co-assembled datasets. Especially, the blackout regions dues to the amplification bias, as an essential step in single cell sequencing, are recovered by the information exist in the other co-assembled datasets. We modified the normal de Bruijn graph and implemented the colored de Bruijn graph. In this Chapter, we present the colored de Bruijn graph and the stages of the co-assembly process. Furthermore, we present the proposed the assembly tool, HyDA, and its strategy for co-assembly process.

2.2 Coloring Reads and k-mers

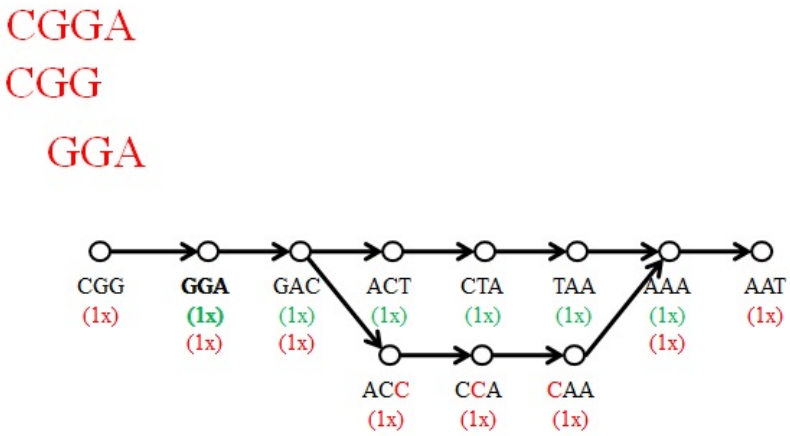
The key concept of co-assembly method is to retain the information of each input dataset in all stages of the process. The first step of the co-assembly is to assign a unique

Figure 2.2: The key value in co-assembly process is the unique colors that are assigned to each read datasets. All the objects originating from a colored dataset, such as reads and k-mers, are painted with the same color as their source's.

color to each dataset. This unique value is used to preserve the source of each information, i.e. all the objects (such as reads and k-mers) originating from a colored dataset are colored by its unique color value. Figure 2.2 illustrates two colored datasets whose reads and generated k-mers from those reads are painted with the color of their origin dataset.



(a) Colored de Bruijn graph



(b) After inserting k-mers

Figure 2.3: 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. Since no node in the graph has the sequence of CGG, a new node is added to the graph with the sequence CGG and the red 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 the same 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).

2.3 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 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 2.3.

2.4 Colored De Bruijn Graph Condensation

After importing all the colored k-mers, the graph is constructed. The next step is to condense the graph as much as possible. Any two following nodes can be merge together if the edge between them is the only outgoing edge of the predecessor node and the only incoming edge to the successor node. Hence, all the one-in one-out connected nodes in a chain (path) are condensed into a supernode, called a *contig* [44], whose sequence is the merged sequence of the path and its coverage is the average coverage of the condensed nodes. Figure 2.4, is a toy example of the condensation result of a colored de Bruijn graph constructed from a red

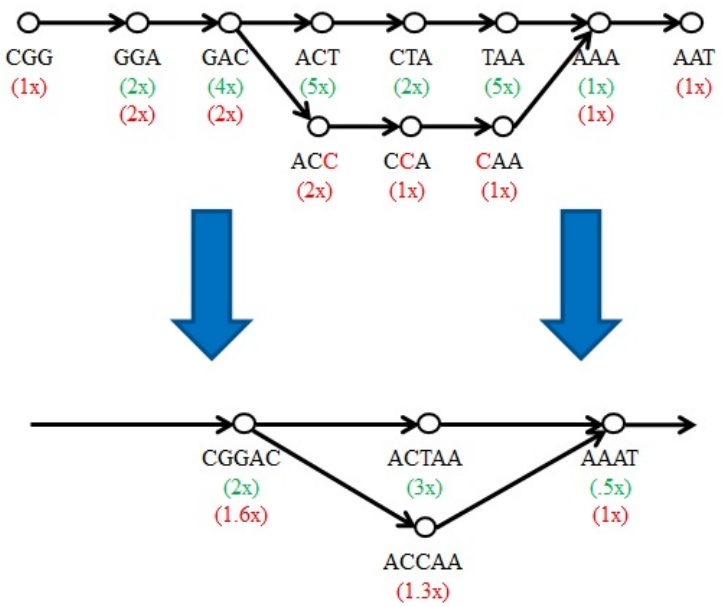
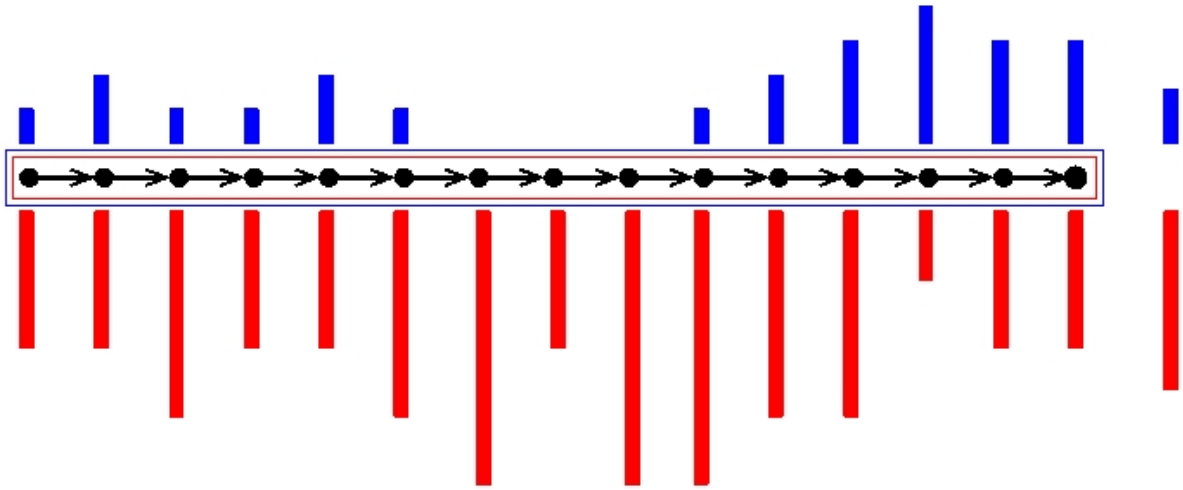


Figure 2.4: Colored de Bruijn graph condensation stage is presented. The colored coverage of the condensed node is the average of the colored coverage of all the nodes in the path.

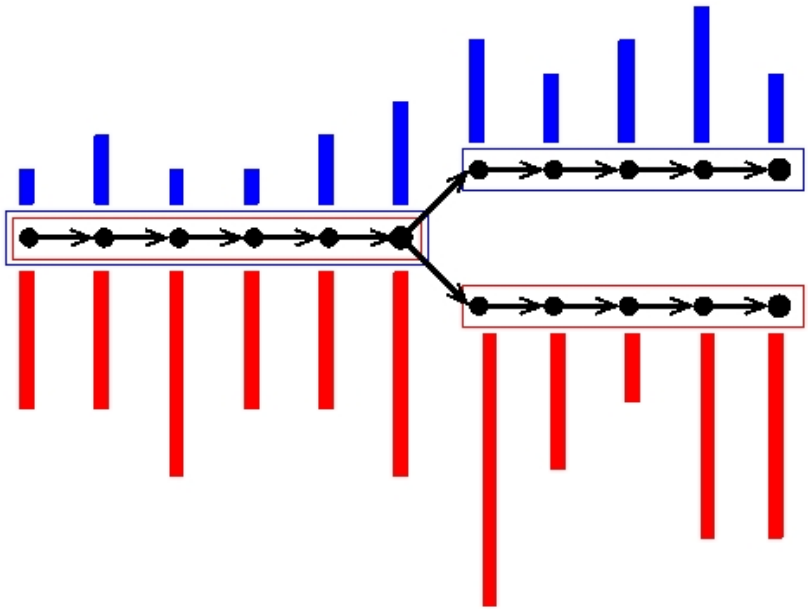
and a green dataset. The key factor in the condensation process is to ignore the coloring information. The only noticed coloring information is for the average coverage calculation. When a part of the region in a colored dataset is blackout due to the amplification bias, the region is broken down and its k-mer path is disconnected (generates 2 sub-graph). However, if the region is fully covered in the another co-assembled dataset the blackout k-mers are imported to the graph, which affix the broken region (See Figure 2.5.a). This shows how a low coverage regions are rescued in co-assembly process. In the case of real variation (with sufficient coverage), two uni-colored path is generated (See Figure 2.5.b).

2.5 Iterative Error Removal

The contigs obtained by the primary condensation should be purified in error correction stage. That is, the contigs whose average coverage is less than a threshold is removed from the final result. When the coverage of a dataset is pretty uniform for all bases, one threshold and one error correction round is sufficient. However, a single threshold does not work for a dataset with un-even depth of coverage and it is highly possible that a valuable

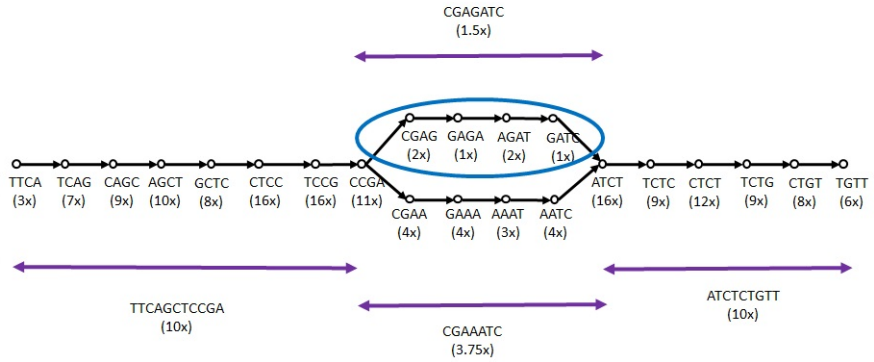


(a)

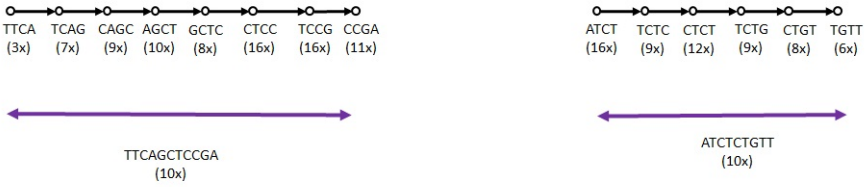


(b)

Figure 2.5: Two sample colored de Bruijn graphs with colors red and blue. Nodes are k-mers and edges represent $k+1$-mers. A colored bar shows multiplicity of the k-mer in the corresponding colored input dataset. Each box is an output contig, and the color of a box shows non-zero colored average coverage, which is shown on the right hand side of the contig in (a). our co-assembly algorithm (a) rescues a poorly covered region of the genome in one color when it is well covered in the other, and (b) allows pairwise comparison of colored assemblies through revealing all of their shared and exclusive pieces of sequence.



(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 2.6: The difference between normal error removal approach and the method used in single cell assembly is illustrated. Consider that cut-off equals to 5. Two supernodes in the graph (a) have a coverage less than the cut-off. 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 cut-off 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 cut-off, all the nodes are saved. That is, the sequence CGAAATC is rescued.

information with low coverage is ignored (see figure 2.6). Therefore, multiple thresholds and a number of error correction processes are needed for single cell datasets since the amplification bias makes the coverage of each base a random value from zero to a very high value. The iterative error removal method with variable threshold was introduced by Velvet-SC [1], a modified version of Velvet [44] for single cell datasets. Like other single cell assembly tools, our error correction process for single cell co-assembly follows Velvet-SC [1] and it is done in an iterative procedure. In each iteration, the contigs with the coverage lower than the cut-off threshold are removed; the graph is recondensed; and the cut-off is increased once. It should be noticed that in co-assembly process a contig should be trimmed based on an array of coverage. One simple solution is to peak the maximum value of the array and remove the contigs whose maximum colored coverage is less than the cut-off threshold. However, the bias of this strategy is high in the case that each input read dataset has different depth of coverage. A normalization cut-off strategy can overcome the bias due to various depth of coverage of each co-assembled dataset. That is, instead of a single cut-off value in each round, we calculate an array of cut-off in which each component of the cut-off array is assigned to one dataset (i.e. we generate colored cut-off). A contig whose each colored coverage is less than its corresponding colored cut-off is removed from the graph.

2.6 Implementation

We implemented our proposed colored de Bruijn graph and co-assembly method in a co-assembly tool, called HyDA: Hybrid *De novo* Assembly tool. HyDA can be categorized as a single cell assembly tool, which follows Velvet-SC [1] to address the uneven depth of coverage as the major problem in single cell sequencing technology.

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.

2.6.1 Import

HyDA gets all the reads from various files at once and annotates them based on their source library (file). Both FASTQ and FASTA format are accepted by HyDA. All the inputs are introduced in a single command. The annotated reads from various FASTA or/and FASTQ files are pre-processed and and stored in a single FASTA file, which is used for the other stages of the co-assembly process by HyDA. The reads without enough information (such as small reads and reads with many unknown bases) are not imported to the final FASTA file. The coloring process is not done in this phase and the reads are annotated only by their origin file, not the colored dataset. Coloring the reads are done in the next step.

2.6.2 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.

Indexing Colored k-mers

To keep the information of co-assembled input datasets, a unique color number up to a compile time constant $MAXCOLORS$ is assigned to each input dataset. In the first step of the algorithm, the reads are imported to one file and the datasets and their color numbers are defined by the user in a config file.

The algorithm uses a hashed collection of splay trees to store de Bruijn graph. The k-mers (for k up to compile time $MAXK$) and their associated information, which include colored multiplicity, internal flags, and in/out edges, are represented in the vertices of the tree. A space in the 2 bit compressed form and 8 bits (one byte) are needed for a k-mer and its in and out edges. First 4 bits are dedicated to incoming edges and the rest are for outgoing edges and each of them represents the extremal left/right nucleotide of the connected k-mer. Since space efficiency is a bottleneck for de Bruijn graph algorithms, we use an array with relative pointers for storing splay trees. A k-mer and its reverse complement (in the case of double stranded data) are considered as a single entry, which saves space.

Condensation of the De Bruijn Graph

A chain of one-in-one-out nodes in the de Bruijn graph can be condensed and replaced with a new structure called *unitig*, which is an augmented vertex with the equivalent sequence. A unitig can be extended until a branch prevents further augmentation. The maximal length unitigs are called *contigs*. The condensation process in HyDA is done in two steps. In the first step, which is done by assemble-unitig, random one-in-one-out k-mers are selected and extended as long as possible into unitigs. The unitigs and the remaining k-mers, which are considered as single k-mer unitigs are stored in one file. In the next step, which is done in assemble-finish, the unitigs are condensed into contigs.

The condensation process is done only based on the graph structure; the coloring information does not influence that process. That is, a chain of one-in-one-out k-mers are

condensed even when the k-mers are not connected in one colored dataset. (see Figure 2.5). This strategy is useful when single cell MDA datasets are co-assembled since a blackout region due to amplification bias in one dataset that is covered in another is rescued.

Memory Footprint Reduction

By implementing a novel technique for partitioning de Bruijn graph into multiple slides, we reduced the peak memory requirement. The key idea is consideration of adjacency information and trying to store k-mers of a contig in one slice. We employ the idea of the minimizer that was introduced for space efficiency in seed and extend method [51].

Let $s_{\max}$ be the number of slices that the graph is partitioned into. Let ω be the minimizer m-mer of k-mer κ in which $m < k$. The k-mer κ is stored in the slice number

$$s(\kappa) = h(\omega) \bmod s_{\max}, \tag{2.1}$$

in which h is an arbitrary uniform hash function. Obviously, the minimizer of adjacent k-mers are mostly identical, which causes minimum slice change along a contig. The slicing approach in HyDA, is implemented in `assemble-unitig`. The m should be large enough to prevent large number of slice changing. However, even in the case of choosing suitable m it is possible that adjacent k-mers are stored in different slices. HyDA is implemented with $m = 8$ based on the observation of having a tradeoff between slice-contiguity and partition balance for both bacterial and mammalian genome assembly.

2.6.3 Assemble-finish

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

Iterative Error Removal

HyDA uses the iterative low coverage contig removal strategy that was introduced by Velvet-SC [1]. To address the un-even coverage generated by MDA bias, a variable coverage cut-off which starts from 1 and gradually increases in each iteration, is used to eliminate low coverage contigs. Before next round, the graph is re-condensed which, generates merged contigs with new average coverage. The maximum cut-off, which is used in the last round is chosen based on the read dataset average coverage. Obviously, when various read datasets with different average coverage are co-assembled, each dataset should be trimmed by its own cut-off value. Hence, HyDA uses a normalization strategy in its iterative error removal stage. The maximum cut-off for each co-assembled dataset is determined based on its average coverage. In each iteration, a series of colored cut-off is calculated and these contigs whose colored coverage is less than the colored cut-off are eliminated. More precisely, a contig that is normally covered in one co-assembled dataset and not covered well in other datasets is rescued. This is how HyDA can retrieve the blackout regions in one cell by using the information of that region in other datasets.

Contig Sets

The output of the iterative error removal stage is a mixture of contigs of all co-assembled datasets.

The contigs are presented 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 2.7. The three contig types, which are illustrated in Figure 2.7 are:

- Full-covered Contig: a contig that is covered in all co-assembled datasets (Figure 2.7-a).
- semi-colored Contig: a contigs that is covered in all co-assembled datasets except one (Figure 2.7-b).
- Uni-color Contig: a contig that existent in only one dataset (Figure 2.7-c).

>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 uni-colored contig

Figure 2.7: An example of a contig of the co-assembly of 5 colored input dataset. (a) a full-colored contig, which is covered in all co-assembled datasets. (b) a contig that is covered in all all input datasets except dataset 4. (c) a uni-colored contig, which is existent only in dataset 4.

The number and length of contigs of each type are important for analysis of the co-assembly result and detecting the outlier datasets, which will be explained more in Chapter 3. Also, to cluster the contigs based on their reference genome, the information of all colored coverage for each contig is an essential value. It is explained more in Chapter 4.

2.7 Summary

In this section we present our co-assembly algorithm based on the colored de Bruijn graph, which was previously proposed for genotyping and structural variation detection by Cortex [45]. In contrary to Coretex, our proposed colored de Bruijn graph considers the uneven coverage characteristic of single cell datasets by pruning the graph using an iterative error removal method, which was introduced by Velvet-SC [1]. Our algorithm also can retrieve the blackout regions of one datasets with the information of other co-assembled dataset. We present the proposed tool, HyDA (Hybrid *De novo* Assembler), and explain its various parts and stages related to co-assembly process.

3 ASSEMBLY OF A GENOME FROM MULTIPLE SINGLE CELLS

3.1 Introduction

In contrary of read dataset of cultivated cells, which have uniform coverage, single cell read data sets have highly variable coverage [52, 53] an even some regions do not get coverage at all (see Figure 3.1), which poses serious challenges for downstream applications such as *de novo* assembly. A number of single cell assemblers including EULER+Velvet-SC [1], SPAdes [5], and IDBA-UD [9] have been developed to mitigate the adverse effects of non-uniform coverage and maximize the transfer of sequencing information into the final assembly. These efforts have been successful, and the existing single cell assemblers are able to extract nearly all of the information contained in the input data set. However, the vast majority of single cell data sets do not encompass the entire genome. We report that combining multiple data sets from the same species significantly improves the final assembly by filling genome gaps (Table 5.1). The challenge presented by this method is to avoid chimeric assemblies due to the outlier dataset in the combined sample since the final assembly includes the information of all datasets (both identical datasets and outliers).

The ideal solution consists in co-assembly of multiple data sets without mixing explicitly the reads so that individual assemblies can benefit from the synergy without suffering from chimerism. We propose and implement this solution using the colored de Bruijn graph[45]. After co-assembly process we analyse the result and verify the similarity of the co-assembled datasets. In the case of outlier existent, the final result is purified to generate

a high quality non-chimeric genome assembly. The algorithm is implemented in a tool called HyDA (Hybrid *De novo* Assembler).

3.2 Similarity Check and Pruning the Assembly Result

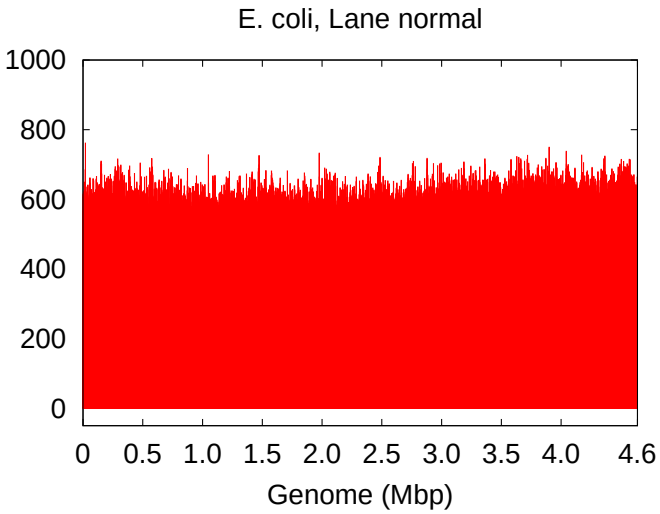
After the co-assembly process on multiple single cell datasets, which are assumed to be identical, the result need to be analysed and purified. To avoid chimeric result, it is necessary to use only the information of identical cells. Hence, after the co-assembly done, the next step is to verify that all the co-assembled cells are identical.

The format of the contig representation in the output is illustrated in Figure 2.7. There are three contig types that are important for clustering process and analysis of the co-assembly result:

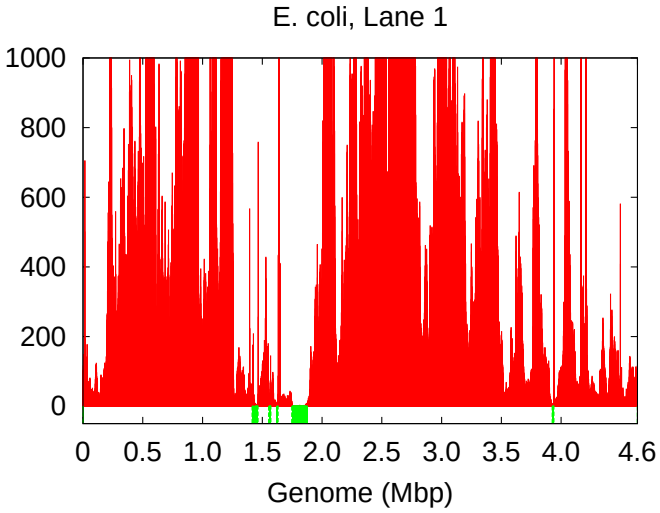
- Uni-color Contig, which is existent only in dataset
- Full-covered Contig, which is covered in all co-assembled datasets.
- semi-colored Contig, which covered in all all input datasets except one dataset

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

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 coverage of contig j. Contig g_j is called full-colored if:



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



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

Figure 3.1: 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 multi-cell 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 [1].

$$\forall a_{ji} \in A_j : a_{ji} > 0$$

$$1 \leq i \leq n \text{ and } 1 \leq j \leq m$$

Contig g_j is called uni-colored 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 uni-colored 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 uni-colored to total contigs are calculated. The high ratio indicates the fact that the dataset is not identical to other datasets. Then all uni-colored 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.

3.3 Materials and Experiments

Genomes amplified from single cells exhibit highly non-uniform genome coverage and multiple gaps, which are called blackout regions [1]. For the evaluation of such coverage characteristics in this study, we used amplified DNA originating from two single *Escherichia coli* cells as well as from one single *Staphylococcus aureus* cell [1]. Although these amplified DNAs were quality checked for preselected genomic loci using quantitative PCR [52], they still did not cover the entire genome (Table 5.1, Figure 3.1). One single *E. coli* cell was sequenced in four technical replicate lanes (1-4) and the other was sequenced in three technical

Data set	No. of blackouts	Mean length	N50	Total (%)
lane 1	94	1220	5558	84K (1.8%)
lane 6	50	193	518	5K (0.1%)
lanes 1 and 6	0	0	0	0 (0.0%)
S. aureus	2	95	83	143 (0.0%)
Replicates of *E. coli* lane 1:				
lane 2	91	1183	4700	77K (1.7%)
lane 3	92	1159	5842	77K (1.7%)
lane 4	88	1225	6156	76K (1.7%)
Replicates of *E. coli* lane 6:				
lane 7	63	153	456	5K (0.1%)
lane 8	61	185	573	6K (0.1%)

Table 3.1: The number, mean length, N50, and total size (percentage) of blackout regions in the *E. coli* and *S. aureus* data sets [1] as explained. All lanes are *E. coli* except the one marked *S. aureus*.

replicate lanes (6-8) each with a sequencing depth of 600 per lane. The single *S. aureus* cell was sequenced in two technical replicate lanes each with a sequencing depth of 1,800. All nine lanes were sequenced on Illumina GAIIx platform in paired 2100 bps read mode.

3.3.1 Coverage Characteristics of Single Cell Read Datasets.

The coverage bias in technical replicates is almost identical, which suggests that the vast majority of bias is caused by MDA. The coverage bias, particularly of the blackout regions, do not always occur at the same genomic loci for different cells of the same genome [1]. Blackout regions in *E. coli* lanes 1 and 6 sequenced from two independently amplified single cells make up 1.8% and 0.1% of the genome respectively, but there are no common blackout regions between these two data sets (Table 5.1). This means that combining the two data sets could fill all gaps and yield a complete genome, which is the property that HyDA exploits with colored co-assembly.

3.3.2 Various Scenarios

To display the strength of co-assembly method, we ran 5 different experiments using above data (see Figure 3.2). All the following datasets are assembled with $k = 55$:

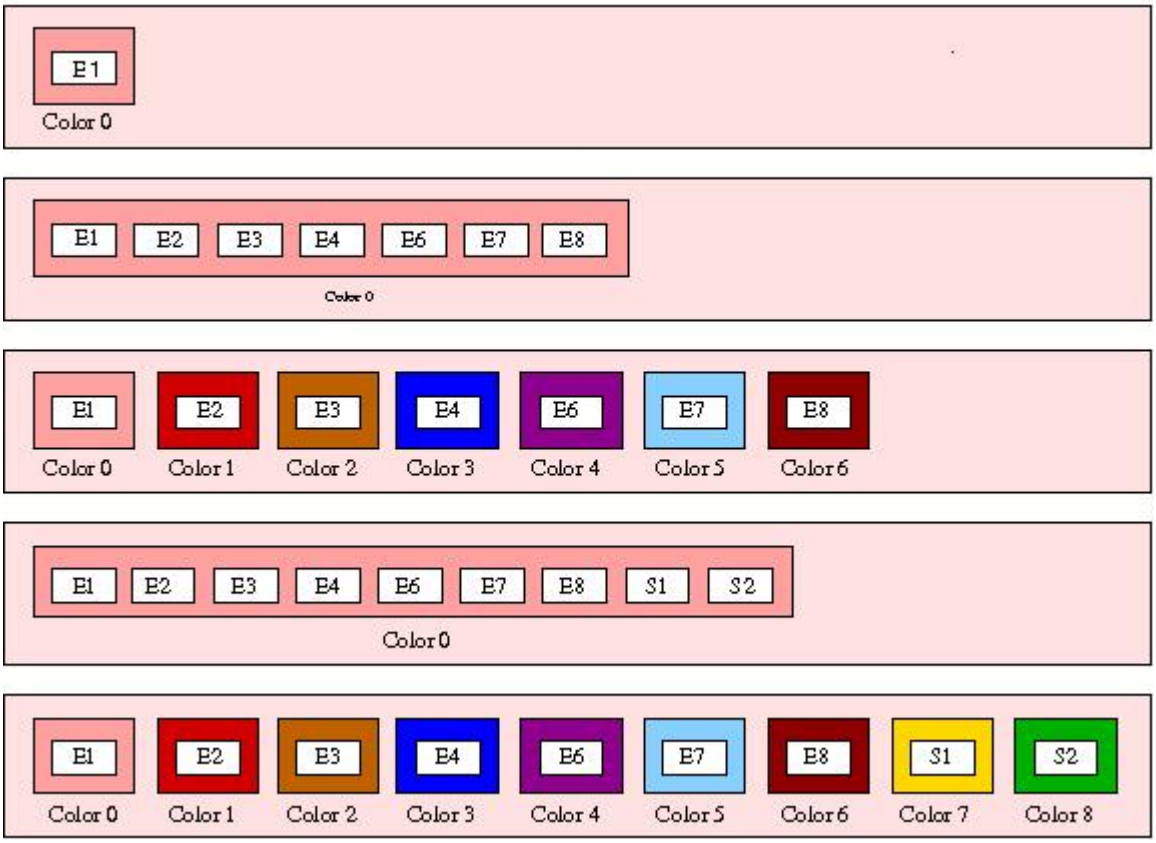


Figure 3.2: The five assembly scenarios: (i) single cell assembly of *E. coli* lane 1; (ii) mixed monochromatic assembly of *E. coli* lanes 1-4 and 6-8, technical replicates of two biologically replicate single cells; (iii) multichromatic co-assembly of *E. coli* lanes 1-4 and 6-8; (iv) mixed monochromatic assembly of non-identical cells: *E. coli* lanes 1-4 and 6-8 and *S. aureus* lanes 7 and 8; (v) multichromatic co-assembly of non-identical cells: *E. coli* lanes 1-4 and 6-8 and *S. aureus* lanes 7,8, each assigned a unique color

- Single cell assembly of Lane 1: we used the E+V-SC single cell assembly of *E. coli* Lane 1 of [1].
- 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.
- Colored assembly of 7 *E. coli* lanes: we assigned a unique color to each of the 7 datasets and assembled them with HyDA using cut-off 100.
- Mixed (uni-colored) assembly of 7 *E. coli* lanes and 2 *S. aureus* lanes: we concatenated the reads in all those lanes and assembled the resulting aggregated dataset with HyDA. We used a coverage cut-off 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 cut-off of 100. We then extracted *E. coli* contigs from the final assembly based on their coverage in the 7 *E. coli* colors.

3.4 Co-assembly of *E. coli* And *S. aureus* Mitigates the Effect Of Dropout Regions

As mentioned above, the ideal solution to assemble a genome from single cells consists in co-assembly of multiple data sets without mixing explicitly the reads so that individual assemblies can benefit from the the information retained in all inputs while avoid chimeric result due to the information of outlier datasets. The 5 Scenarios explained above were experienced and compared to display the advantages of using co-assembly method.

3.4.1 Detecting Outlier

The proposed method of assembling a genome from multiple single cells has two stages: (1) co-assembling the datasets (which is presented in previous Chapter), 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 3.2 94% of contigs in the assembly of 7 *E. coli* lanes (Figure 3.2-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 3.2-v) are fully-colored. In this case, all the lanes are examined and the outlier dataset is detected by calculating the number of their uni-colored

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

Table 3.2: 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 (non-identical 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 non-identical cells assembly are not fully-colored.

contigs (a dataset 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.

3.4.2 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 [2] 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.
- 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.

3.4.3 Evaluating the Quality of Colored Co-assembly of *E. coli*

We used GAGE [2] which is a standard assembly evaluation tool to compare the 4 assemblies described above and the `E+V-SC` single cell assembly of *E. coli* Lane 1 in [1]. The results are presented in Table 3.3. It is clear from Table 3.3 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 3.3 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

	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 3.3: Comparison of GAGE [2] evaluation results for *E. coli* assemblies obtained from a single cell with E+V-SC (Lane 1 of [1]) and multiple identical (Lanes 1-4 and 6-8 of [1]) and non-identical cells (Lanes 1-4 and 6-8 plus two lanes of *S. aureus* in [1]) 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 [2]. *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 multi-cell assemblies is higher than that in single cell one [1]. 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 [3]. All contigs are considered in all assemblies, particularly ultrashort single k-mer ones.

results show that the single cell assembly has 100 SNPs and 34 indels < 5 bps, whereas multi-cell assembly has only 5 SNPs and 6 indels < 5 bps (see Table 3.4).

Mixed and colored assemblies have roughly similar characteristics. The second column of Table 3.3 represents the characteristics of mixed assembly when HyDA assembled a pool of 7 *E. coli* lanes coming from 2 biological replicates with cut-off 400. The third column is the colored assembly of the same dataset with cut-off 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 [1]. The only noticeable difference between the two assemblies is their N50, NG50, and corrected N50 [2, 3]. 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 such as SNP and Indel are the same as the assembly of identical

	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 3.4: 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.

datasets. Also, the mixed and colored assembly do not show any difference in term of the number of errors. Table 3.4 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 coverage. That is why the cut-offs 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 3.3. Provided that the input species are sufficiently different and MDA reactions cover a high portion of the genomes, we

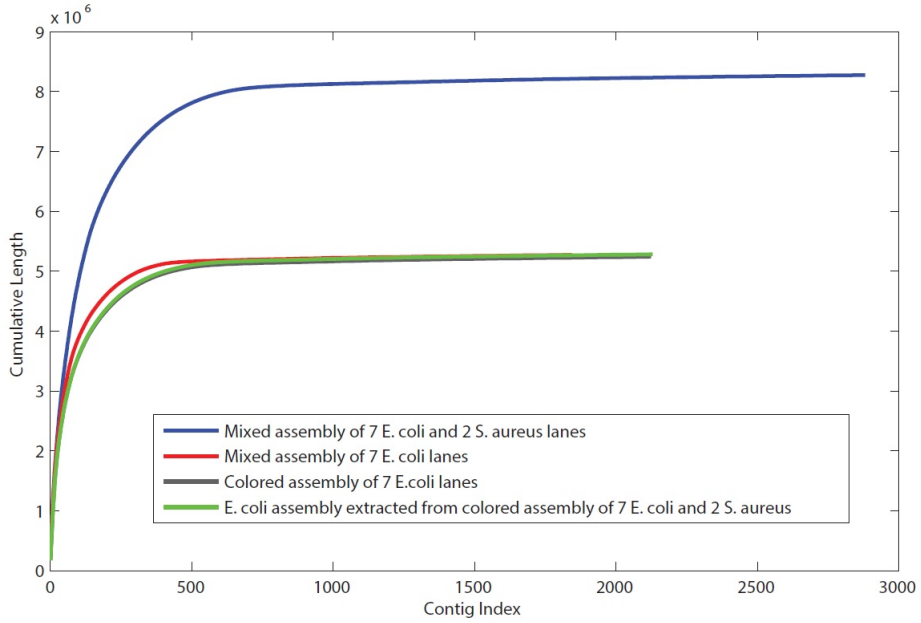
can easily find identical cells by clustering contigs. Performing the same experiment on more than 2 species is left as future work as it requires more data.

As mentioned above, the advantage of colored de Bruijn graph becomes clear when a combination of non-identical cells is assembled. The last two columns of Table 3.3 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 [1].

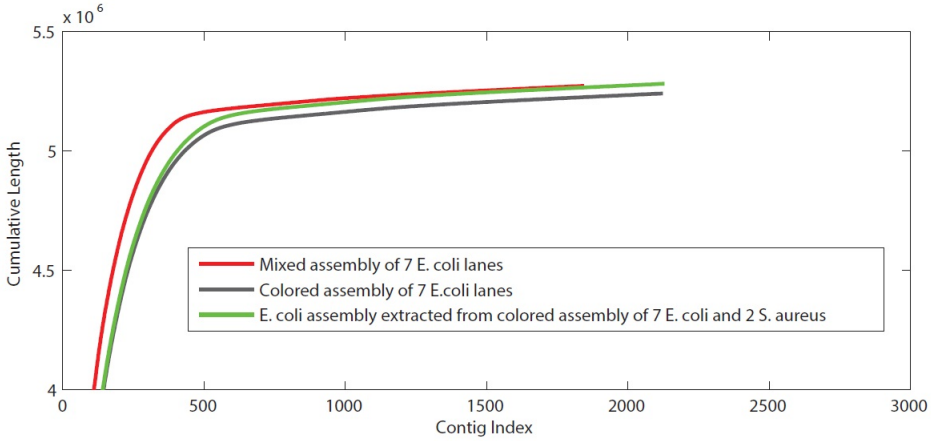
The only weakness of the co-assembly method is related to the problem of contiguity loss. True genomic variation in the identical cells generates branches in the colored de Bruijn graph, which makes contigs shorter. The comparison between contigs generated by co-assembly and normal method is illustrated in Figure 3.3.

3.5 Summary

In this chapter, we present the challenge of assembling a genome using multiple single cell datasets and our proposed solution to this problem. Due to the blackout regions, we need a number of identical cells of a genome to have a high quality assembly. Unfortunately, the assembly result will be chimeric if even one of the inputs is not identical. We proposed our elegant solution based on co-assembly using colored de Bruijn graph. Since the co-assembly result retains the information of each input dataset individually, we can verify whether the input datasets are identical. In the case of outlier existent, our algorithm can detect the non-identical datasets and eliminate their associated data. Hence, our algorithm can recover the genome with small number of missing bases while avoids chimeric result. We provide 5 experimented scenarios' result to display the strength of co-assembly method.



(a)



(b)

Figure 3.3: Comparison of contigs generated by mixed versus colored assembly for single cell *E. coli*. (a) contigs are those presented in Table 3.3 and are sorted from largest to smallest. The y axis shows the cumulative length. (b) a magnified portion of the plot in (a).

4 INEFFICIENT SYNERGISTIC SINGLE CELL GENOME ASSEMBLY

4.1 Introduction

Co-assembly of bacterial genome using multiple single cell generates a high quality result since it combines the information of all identical datasets and trims the errors information that are due to the outlier information. The errors are not detectable by normal assembly since the normal assembly result is based on the information of a mixture of all single cell datasets, including identical and outlier ones.

The advantages of using colored de Bruijn graph and co-assembly is related in: 1- determining the similarity between the co-assembled datasets and 2- retrieving the blackout regions of the genome due to the ampliation bias. In the case of having multiple assumed to be identical single cell datasets, co-assembly detects the outliers and removes their information from final result. Also, it can retrieve the blackout regions in one dataset when it is covered in another dataset.

The application of the co-assembly is not restricted to the situation that we have a number of identical datasets. Both advantages of co-assembly are also exhibited when we have a number of single cells from various phylogenetically close species. By using co-assembly, we can find the similarity between each two co-assembled datasets. Also, a blackout region in each input dataset can be retrieved in the case that the region belongs to a coding sequence (or gene) that is existent and covered in one of the other co-assembled datasets. Since the input datasets are from phylogenetically close species, many genes are common

between their genome and the blackout regions of one dataset are covered in at least one of the other inputs with a high probability.

In this chapter we explain the modification has been done on the colored de Bruijn graph and co-assembly process to make it more suitable for co-assembling some phylogenetically close species. Also, we present our proposed method to check the similarity between them using a variable, *ExclusivityRatio*. Finally, to show the advantages of co-assembly of such species, we run the modified method (which is implemented in HyDA v.1.2) on the MDA datasets of 10 single cells from three dominant but uncultured bacterial members of a methanogenic consortium [54, 55], belonging to the families *Syntrophacea* and *Anaerolineaceae*.

4.2 Sensitivity of Co-assembly Result

The result of the genome assembly process is the collection of maximal sequences (contigs) obtained by the final de Bruijn graph condensation after error removal process. There is a contig set for each colored input dataset, which contains all the contigs with positive coverage for corresponding color. Categorizing the contigs based on positive coverage value works fine when some identical cells are co-assembled. However, this naive method dose not work properly when a number of single cell of different species (especially, when they are of the reference genomes from phylogenetically close families) are co-assembled. We modified our algorithm by using a new parameter ϵ.

Let C be the set of all the 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.

Color A	Color B	Decision	Figure
High	High	Different	4.1-(a)
High	Low	subset	4.1-(c)
Low	High	subset	4.1-(c)
Low	Low	Similar	4.1-(b)

Table 4.1: Making decision about the relationship between two co-assembled color is based on *exclusivity ratio*. Exclusivity ratio of color A with respect to color B is exclusive contigs of A divided to its common contigs with B

We use $\epsilon = 0$ in previous chapter (HyDA v.1.1), but a non-zero ϵ is needed for co-assembling dataset of various species. Even when identical datasets are co-assembled, a positive ϵ value increases the quality of result if there are erroneous or contaminant k-mers in one color which also occur in the true genomic sequence of another color.

4.3 Quantification of Similarities and Differences Between Colored Datasets.

One advantages of co-assembly method is that it can determine the relationship between the genome of the co-assembled datasets. Input datasets can be clustered based on the similarity between their assemblies. For a pair of color A and B, we call the contigs that belong to both the colored datasets *shared* and those contigs that belong to color A but not to color B *exclusive* of color A with respect to color B. Let $E_A(B)$ denotes the *exclusivity ratio* of color A with respect to color B, as the ratio of the size of exclusive color A contigs to the total assembly size of color A. This factor helps us make decision about the relationship between two colors. Having a very high/or low number of common contigs makes the similarity decision easy (see Table 4.1 and Figure 4.1). On the other hand, there is grey area in which we cannot make sure that whether two colored datasets are of the same reference genome.

The exclusivity ratio for the co-assembly experiment explained in previous chapter is calculated in Table 4.3. The exclusivity ratio for both *E. coli* lanes in the pair *E. coli* lane 1-lane 6 (Pair 1 in Table 4.3) is not more than 0.5%, while that ratio for both *E. coli*

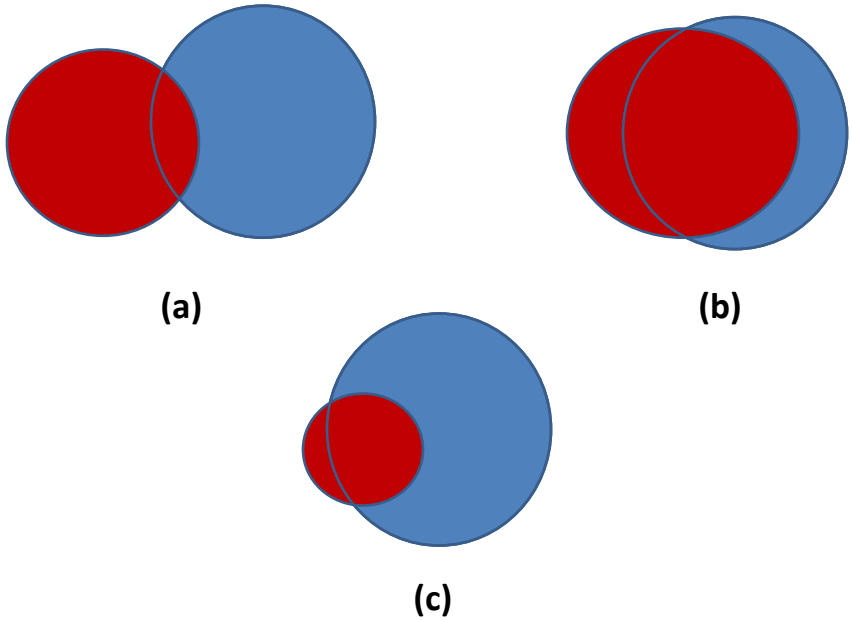


Figure 4.1: The relationship between 2 co-assembled genome can be determined by calculating the common and exclusive contigs of the co-assembled results. (a) displays the assembly of various genomes. (b) displays the assembly of similar genome. (c) displays the assembly of two genomes in which one is the subset of the other.

Pair of Data Sets	Pair 1		Pair 2		Pair 3	
	E. coli lane 1	*E. coli* lane 6	*S. aureus*		*E. coli* lane 1	
Total (bps)	5,228,480	5,240,302	3,366,622		5,228,480	
Shared (bps)	5,210,548		335,648		336,184	
Exclusive (bps)	179,32	29,754	4,904,654	3,030,974	3,030,438	4,892,296
Exclusivity Ratio*	0.003	0.005	0.9359	0.9003	0.9001	0.9357

Table 4.2: * Exclusivity Ratio = Exclusive / Total.

Table 4.3: Pairwise relationships between three co-assembled data sets, *E. coli* lanes 1 and 6 and *S. aureus* lane 7, in a co-assembly of *E. coli* lanes 1-4, 6-8 and *S. aureus* lanes 7, 8. Total is the total size of those contigs that have non-zero coverage in the corresponding color. Shared is the size of those contigs that have non-zero coverage in both colors. Exclusive is the size of those contigs that have non-zero coverage in the corresponding color and zero coverage in the other color in the pair.

and *S. aureus* in the two other pairs is more than 90%. This large margin between the exclusivity ratio for identical cells and that for non-identical cells is expected in this case, as *E. coli* and *S. aureus* are sufficiently divergent species and the genome coverage was enough for both.

4.4 Materials

To display the strength of co-assembling of various genomes from close families, we used the MDA lane of single cells from three dominant but uncultured bacterial members of a methanogenic consortium [54, 55], belonging to the families *Syntrophacea* and *Anaerolineaceae*. Here we explain how the MDA datasets are prepared. The assembly result of each dataset by HyDA is compared with its assembly by two other state of art single cell assembly tools.

4.4.1 Media and Cultivation of the Methanogenic Alkane-Degrading Community.

The microbial community was enriched from sediment from a hydrocarbon-contaminated ditch in Bremen, Germany [55]. The consortium was propagated in the laboratory in anoxic medium containing 0.3 g NH_4Cl, 0.5 g $MgSO_4 \bullet 7H_2O$, 2.5 g $NaHCO_3$, 0.5 g K_2HPO_4, 0.05 g KBr, 0.02 g H_3BO_3, 0.02 g KI, 0.003 g $Na_2WO_2 \bullet 2H_2O$, 0.002 g $NiCl_2 \bullet 6H_2O$, trace elements and trace minerals as previously described [55]. The medium was sparged with a mixture of N_2/CO_2 (80:20 v/v) and the pH was adjusted to 7.0. After autoclaving, anoxic $CaCl_2$ (final concentration 0.25 g/L) and filter-sterilized vitamin solution [55] were added. Cells were supplemented with anoxic hexadecane as previously described [54]. Bottles were degassed as necessary to relieve over-pressurization.

4.4.2 Single cell Sorting, MDA, and Genomes Sequencing.

Individual cells from the alkane-degrading consortium were obtained by staining (SYTO-9 DNA stain) and sorting of single cells by FACS [54]. Single cells were lysed as previously described and the genomic DNA of individual cells was amplified using whole-genome multiple displacement amplification (MDA) [56]. Amplified genomic DNA was screened for *Smithella*-specific 16S rDNA gene sequences. Six amplified *Smithella* genomes were selected for Next Generation Sequencing. The MDA amplified genomes were prepared for Illumina sequencing using the Nextera kit, version 1 (Illumina) using the Nextera protocol (ver. June 2010) and high molecular weight buffer. Libraries with an average insert size of 400 bp were created for these samples and sequenced using an Illumina Genome Analyzer IIx. 34 bp paired-end reads were generated for K05 (20.9 million reads), C04 (23.3 million reads), F02 (26.9 million reads), and A17 (22.2 million reads). 58 bp single-end reads were generated for MEB10 (41.3 million reads), MEK03 (54.1 million reads), and MEL13 (18.0 million reads). 36bp paired-end reads were generated for F16 (11.0 million reads), K04 (27.2 million reads), and K19 (22.9 million reads).

4.5 *De novo* Single Cell Co-assembly of Members of an Alkane-degrading Methanogenic Consortium.

The genomes of 10 cells from three dominant but uncultured bacterial members of a methanogenic consortium [54, 55], belonging to the families *Syntrophacea* and *Anaerolineaceae* were sequenced from their amplified single cell whole DNAs: six cells belonging to *Smithella*, two cells belonging to *Anaerolinea*, and two cells belonging to *Syntrophus*. Single cells were isolated from the consortium by fluorescence-activated cell sorting, and the genomes of individual cells were amplified using MDA. MDA products were sequenced using an Illumina GAIIx with 34, 36, or 58 base pair reads. In total, 10 data sets, one per cell, were obtained. The 10 data sets were co-assembled with HyDA in a *ten-color* setup, and

		Anaerolinea	
		A17	F02
HyDA	Total	54,237	1,278,742
	N50	**2,935**	8,461
HyDA-Colored	Total	**260,386**	1,352,341
	N50	850	8,201
SPAdes	Total	169,413	**1,698,195**
	N50	1,187	5,944
IDBA-UD	Total	144,512	1,441,353
	N50	2,894	**8,756**

		Smithella					
		F16	K04	K19	MEB10	MEK03	MEL13
HyDA	Total	604,769	449,148	371,311	1,182,622	1,666,233	1,150681
	N50	**8,303**	**9,959**	5,416	5,718	6,167	7,315
HyDA-Colored	Total	**1,323,536**	**720,188**	**840,236**	**1,569,709**	1,945,701	**1,590,259**
	N50	6,088	5,239	**7,295**	5,887	5,952	6,977
SPAdes	Total	982,263	618,500	653,866	1,514,813	**1,960,722**	1,415,399
	N50	5,366	9,332	3,834	**8,861**	**11,372**	**10,475**
IDBA-UD	Total	927,009	56,6327	613,399	1,327,742	1,746,656	1,351,465
	N50	3,163	**3,178**	5,751	6,851	8,209	1,0253

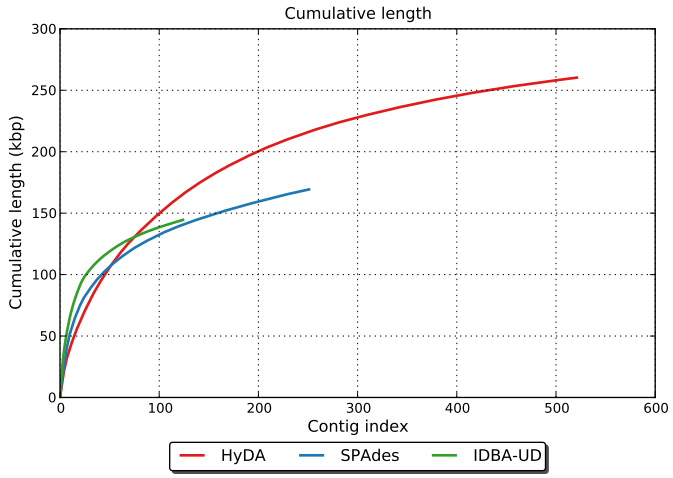
		Syntrophus	
		C04	K05
HyDA	Total	252,402	502,469
	N50	**5,578**	4,963
HyDA-Colored	Total	**465,091**	**1,265,548**
	N50	1,928	3,782
SPAdes	Total	390,923	869,586
	N50	4,234	3,128
IDBA-UD	Total	318,914	804,313
	N50	4,706	**5,618**

Table 4.4: Quast [4] analysis of 10 cells from *Anaerolinea*, *Smithella*, and *Syntrophus* single cell data sets assembled with HyDA (individual assembly), HyDA (10-color co-assembly), SPAdes [5], and IDBA-UD. All statistics are based on contigs of size $\geq$ 100 bp. Only those HyDA contigs that have a coverage of at least 1 in the corresponding color are considered. Coverage cut-off was chosen to be 24 for all HyDA assemblies (-c=24). Total is the total assembly size and N50 is the assembly N50 (the size of the contig, the contigs larger than which cover half of the assembly size).

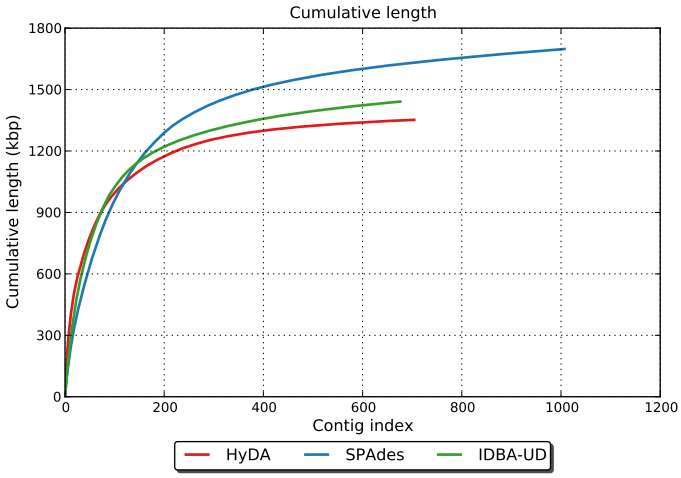
		Colored HyDA improvement versus	
		Individual HyDA %	Best Result 2 %
Anaerolinea	A17	380	**53**
	F02	5	-19
Smithella	F16	119	**35**
	K04	60	**17**
	K19	126	29
	MEB10	32	4
	MEK03	16	-0.7
	MEL13	38	12
Syntrophus	C04	85	**19**
	K05	153	46

Table 4.5: Improvement of colored HyDA in comparison with individual HyDA and the best single cell assembler

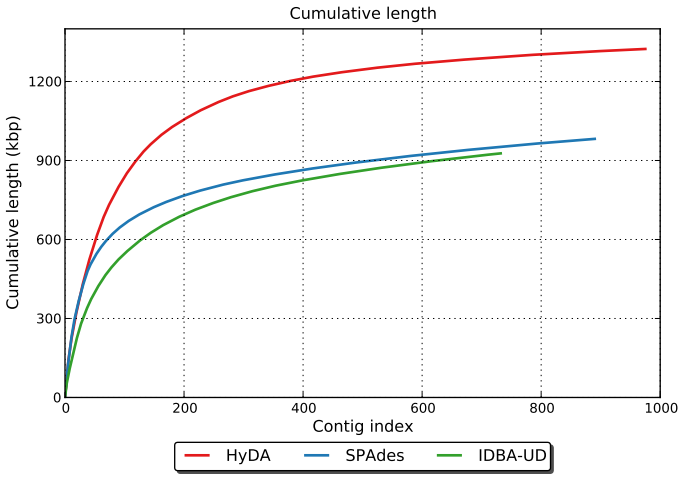
to exhibit the advantage of the co-assembly method, each data set was assembled individually by HyDA. Individual assemblies created by SPAdes [5] and IDBA-UD were used as comparison. The QUAST [4] length statistics of the resulting assemblies ($\geq$= 100 bp contigs) are compared in Table 4.4 and Figure . The comparison between individual-assembly and co-assembly by HyDA demonstrates that co-assembly rescues on average 101.4% more total base pairs for all 10 cells (Table 4.5). Although HyDA does not use advanced assembly features such as variable k-mer sizes and paired read information, it can assemble 3.6% to 54% more total base pairs than both SPAdes and IDBA-UD do in all cells except two cases: *Anaerolinea* F02 and *Smithella* MEK03 (Tables 4.4, 4.5). When all contigs are considered, HyDA co-assemblies of *Anaerolinea* F02 and *Smithella* MEK03 are 11% smaller and 41% larger than their SPAdes counterparts, respectively. *Smithella* MEK03 input reads are longer (58 bp) than the reads in some of the other data sets; therefore, the *Smithella* MEK03 assembly contains many short contigs and suffers because of the small k-mer size (k=25) dictated by the shorter reads.



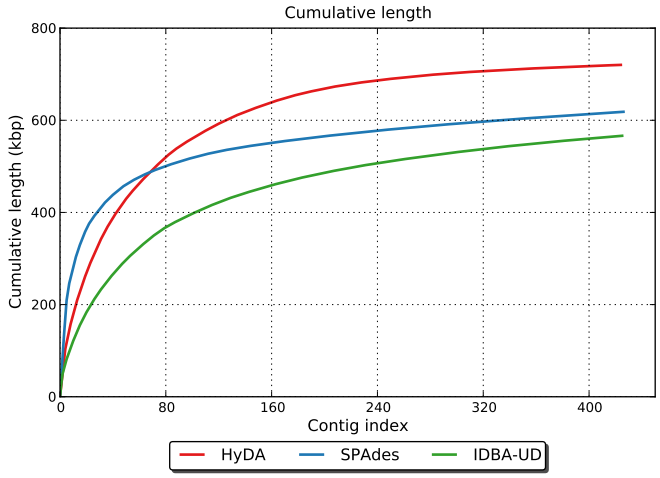
Anaerolinea A17



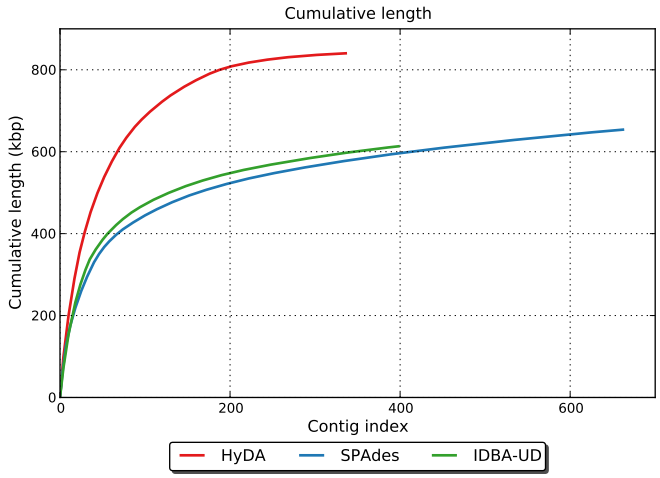
Anaerolinea F02



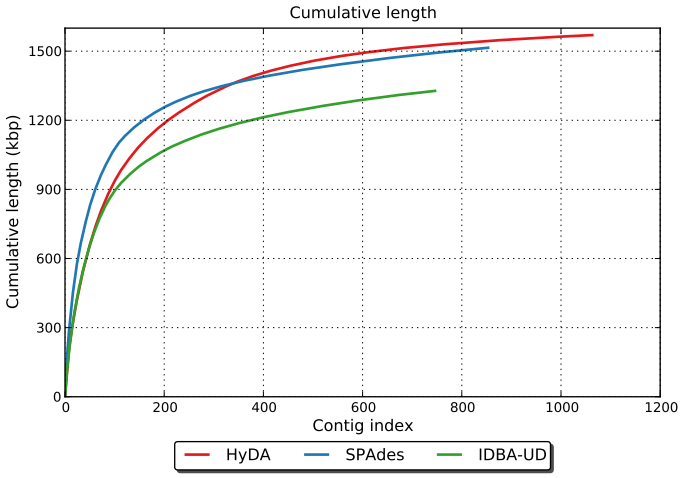
Smithella F16



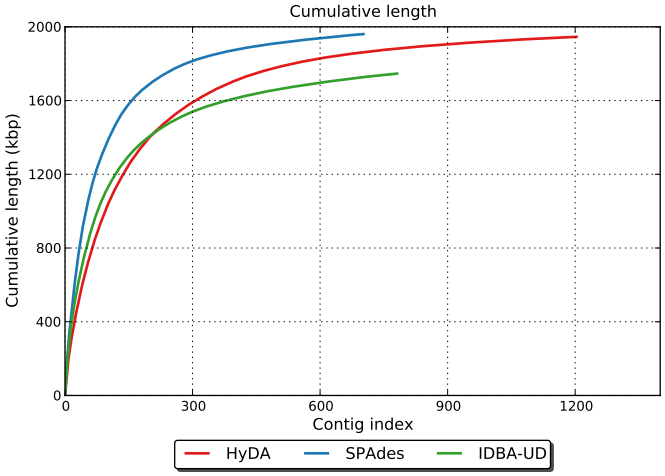
Smithella K04



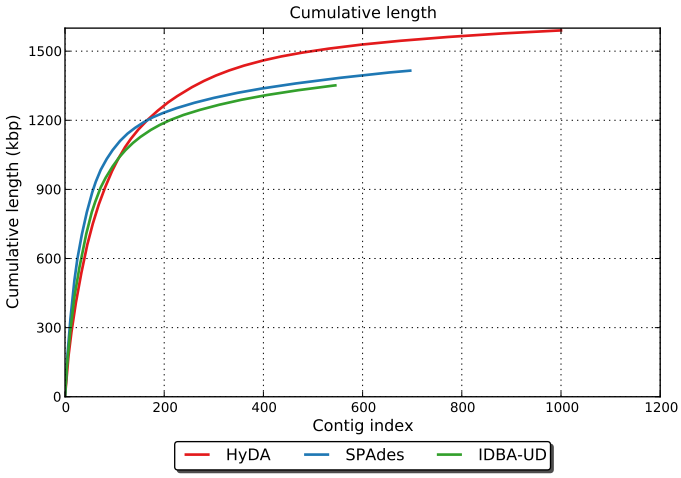
Smithella K19



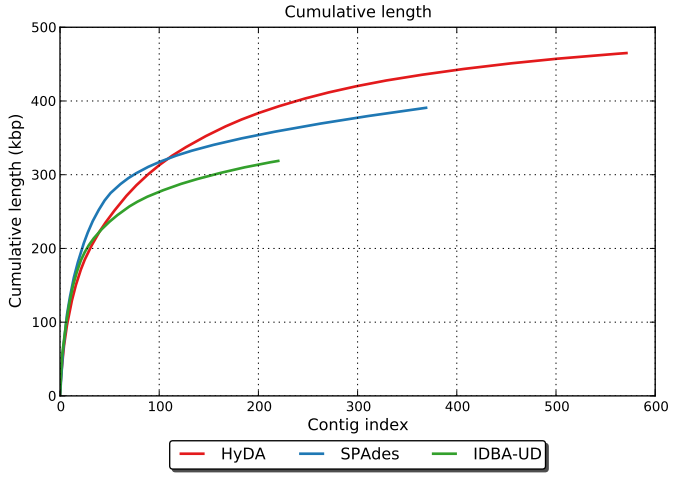
Smithella MEB10



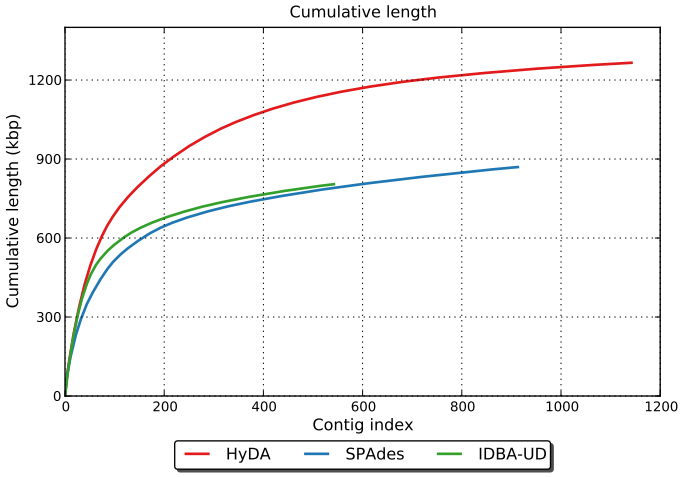
Smithella MEK03



Smithella MEL13



Syntrophus C04



Syntrophus K05

Figure 4.2: QUAST comparison plots for HyDA, SPAdes [5], and IDBA-UD assemblies of Anaerolinea A17, F02, Smithella F16, K04, K19, MEB10, MEK03, MEL13, and Syntrophus C04, K05.

		Anaerolinea		*Smithella*						*Syntrophus*	
		A17	F02	F16	K04	K19	MEB10	MEK03	MEL13	C04	K05
Anaerolinea	A17	0	24	87	95	96	80	82	86	22	19
	F02	77	0	96	98	99	71	68	72	**12**	**5**
Smithella	F16	96	96	0	73	73	37	22	38	96	55
	K04	97	97	49	0	67	42	25	45	97	73
	K19	98	98	54	68	0	35	32	32	98	55
	MEB10	96	96	74	48	69	0	24	39	95	57
	MEK03	97	97	73	54	74	38	0	37	96	58
	MEL13	97	97	76	51	68	39	22	0	97	59
Syntrophus	C04	44	39	89	96	97	85	86	90	0	64
	K05	77	75	54	76	75	45	41	49	73	0

Table 4.6: The exclusivity ratio (%) of row with respect to column for the 10 cells from *Anaerolinea*, *Smithella*, and *Syntrophus* single cell data sets co-assembled using 10 colors with Squeezambler [6], a tool in the HyDA package. Only the contigs of coverage at least 1 in the corresponding color are considered. Coverage cut-off was chosen to be 24 for all HyDA assemblies (-c=24).

4.6 Extensive Analysis of 10 Assemblies from Single Uncultured Bacterial Cells

Analysis of exclusivity of the dataset information revealed that the six Smithella cells clustered into a consistent group as their exclusivity ratios with respect to the two *Anaerolinea* and two *Syntrophus* cells are almost identical (Table 4.6). It is important to note that *Anaerolinea* A17 and *Syntrophus* C04 assemblies are relatively short, meaning the exclusivity ratios must be interpreted with caution. Although *Syntrophus* K05s exclusivity signature with respect to the six *Smithella* cells is indistinguishable from the six *Smithella* signatures with respect to themselves, the exclusivity ratios of *Syntrophus* K05 with respect to the two *Anaerolinea* cells and *Syntrophus* C04 differentiates *Syntrophus* K05 from the six *Smithella* cells. Slight differences between the *Syntrophus* C04 and K05 exclusivity signatures are not surprising because of the existence of potential intraspecies variations.

4.7 Annotation of the Anaerolinea, Smithella, and Syntrophus Assemblies

To assess the quality of co-assemblies with HyDA, IDBA-UD, and SPAdes, we used the RAST [7] server to predict the coding sequences and subsystems present in each assembly. The HyDA assemblies are superior to those of SPAdes and IDBA-UD in terms of the number of coding sequences and captured subsystems for one *Anaerolinea*, four *Smithella*, and both *Syntrophus* assemblies (Table 4.7). For *Smithella* MEB10 and MEK03, the HyDA assembly closely follows the SPAdes assembly, which provides the largest annotation (Table 4.7). For *Smithella* F16 and *Syntrophus* K05, HyDA assemblies contain significantly more coding sequences (33% and 39% respectively) and cover more subsystems (29% and 57% respectively) in comparison to the best of SPAdes and IDBA-UD assemblies.

To confirm the accuracy of the assemblies, the closest related species to each assembly was computed by the RAST server. For the HyDA, SPAdes, and IDBA-UD *Anaerolinea* F02 assemblies, the closest species was Anaerolinea thermophila UNI-1 (GenomeID 926569.3) (no closest genomes data found for *Anaerolinea* A17 by the RAST server). For the HyDA, SPAdes, and IDBA-UD *Smithella* and *Syntrophus* assemblies, the closest species is *Syntrophus* aciditrophicus SB (GenomeIDs 56780.10 and 56780.15). Note that *Syntrophus aciditrophicus SB* is the closest finished genome to the *Smithella* family. This verifies that co-assembly does not create chimeric assemblies, otherwise we would see *Syntrophus aciditrophicus SB* among close neighbors of the *Anaerolinea* assemblies and/or *Anaerolinea thermophila UNI-1* among close neighbors of the *Smithella* and *Syntrophus* assemblies by HyDA.

		HyDA-Colored		*Spades*		*IDBA-UD*	
		coding sequence	subsystem	coding sequence	subsystem	coding sequence	subsystem
Anaerolinea	A17	**212**	8	146	**9**	132	7
	F02	1,283	122	**1,653**	**153**	1,375	121
Smithella	F16	**1,197**	**117**	899	91	866	89
	K04	**659**	**89**	559	75	508	66
	K19	**757**	**82**	581	54	572	57
	MEB10	1,491	151	**1,504**	**156**	1,297	138
	MEK03	1,856	180	**1,955**	**200**	1,178	170
	MEL13	**1,535**	**165**	1,435	154	1,384	148
Syntrophus	C04	**416**	48	375	**49**	320	36
	K05	**1,216**	**121**	873	68	854	77

Table 4.7: Summary of coding sequences and subsystems predicted by the RAST server [7] for HyDA, IDBA-UD, and SPAdes [5] assemblies of the three alkane-degrading bacterial genomes.

4.8 Colored co-assembly of some amplified DNA of single E. coli cells by MALBAC purifies chimeric contigs.

For years, MDA has been used as the dominant whole genome amplification method for single cell sequencing. Recently, Multiple-Annealing and Looping-Based Amplification Cycles (MALBAC) was introduced, which entails more uniform coverage [11, 24][12, 13]. However, experiments indicate that MALBAC is more sensitive to background contamination, which leads to generation of large portions of contaminant reads [8]. Our results indicate that assembly of single cell genomes amplified by MALBAC have low quality due to high errors which cause a large number of unaligned contigs and also low assembly contiguity (see Tables 4.9).

We used the genomes of 5 E. coli cells, which were sequenced from their single-cell whole DNAs amplified by MALBAC. The resulting amplicons were sequenced on the MiSeq platform in the 2250 bp read mode [8]. Our experiments on the sequencing read data sets of 5 E. coli cells indicates that our co-assembly method mitigates the effect of bias generated by MALBAC the same way it abates the MDA bias by retrieving the blackout regions in each co-assembled data set using the information existent in other co-assembled data sets(Table

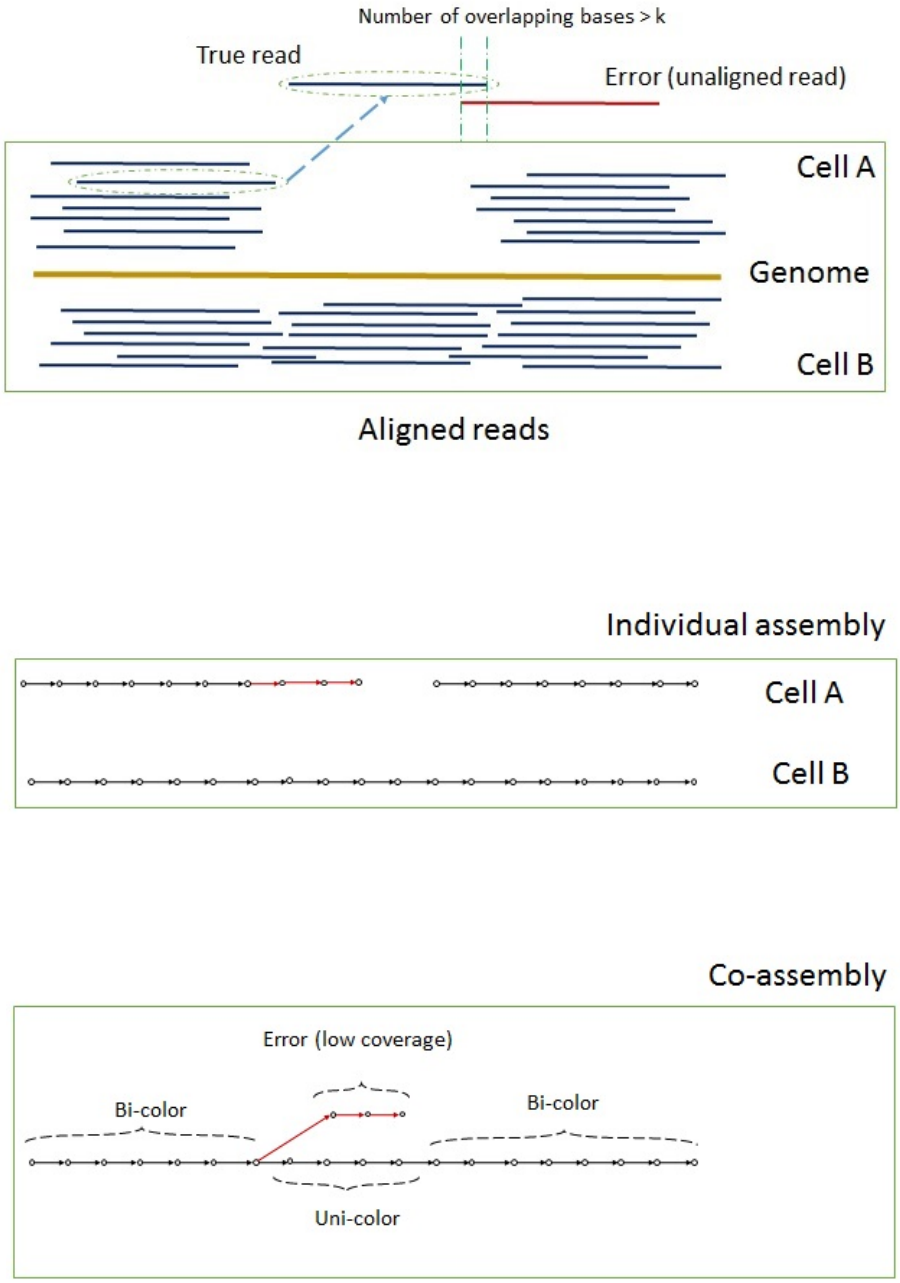


Figure 4.3: co-assembly avoids chimeric contigs generated by noisy data sets. As it is shown for Cell A, the overlap between true reads and error in the vicinity of a blackout region constructs a chain of one-in-one-out nodes in the de Bruijn graph, which is condensed into a chimeric contig. When the blackout region is fully covered in Cell B, the co-assembly process avoids the chimeric contig by separating the erroneous portion at a colored branch.

4.8). Furthermore, using this data, we report the ability of co-assembly method to reduce the number of chimeric contigs in noisy data sets such as MALBAC single cell data sets

	Missing Bases		Improvement %
	Individual	Colored	
Cell 1	2,360,062 (50.87%)	1,528,887 (32.95%)	36 %
Cell 2	3,569,097 (76.93%)	2,747,676 (59.22%)	22 %
Cell 3	702,942 (15.15%)	513,803 (11.07%)	27 %
Cell 4	1,391,317 (29.99%)	815,105 (17.57%)	41 %
Cell 5	964,833 (20.80%)	605,526 (13.05%)	35 %

Table 4.8: Co-assembly can retrieve more bases for each data set than its individual assembly. The evaluation is done by GAGE [2] for assembly of E. coli cells 1-5 [8] using HyDA.

(Table 4.9).

In noisy data sets, many reads have overlap with erroneous reads that causes chimeric contigs. In the vicinity of blackout regions, the overlapping true reads and errors construct a chain of one-in-one-out k-mers, which are condensed into a chimeric contig. Although the coverage of error k-mers is low, the coverage of generated chimeric contig is high enough (average of correct high coverage and low error coverage) to survive in the error removal stage. When a blackout region is fully covered in another data set, co-assembly breaks down chimeric contigs due to a branching in the colored de Bruijn graph caused by correct k-mers of the blackout region. Therefore, the error part is separated and then removed in the error removal stage due to its low coverage. The final result for both data set is a pure contig without a gap. Figure 4.3 displays the advantage of using co-assembly method in term of avoiding chimeric contigs when noisy datasets are used for genome assembly.

4.9 Summary

We demonstrated the power of genome co-assembly of multiple single cell data sets through significant improvement of the assembly quality in terms of predicted functional elements and length statistics. Co-assemblies without any effort to scaffold or close gaps contain significantly more protein coding genes, subsystems, base pairs, and generally longer contigs compared to individual assemblies by the same algorithm as well as the state-of-the-art single cell assemblers (SPAdes [5] and IDBA-UD). The new algorithm is also able to avoid

Cell 1	HyDA: Individual	HyDA: Colored	SPAdes	IDBA-UD
Genome Fraction	47.53	65.175	71.2	60.405
Unaligned bases	44,646,888	55,461,729	155,914,937	72,213,995
Unaligned (%)	94.94	93.7	97.68	97.72
Misassemblies	7	2	50	94
N50	316	246	323	591
NG50	841	703	1,477	1,806
LG50	2,267	2,744	1,247	1,014
LGA50	6,368	7,500	2,390	1,991

Cell 1

Cell 2	HyDA: Individual	HyDA: Colored	SPAdes	IDBA-UD
Genome Fraction	21.476	39.475	48.457	34.263
Unaligned bases	51,370,608	69,705,649	226,346,470	91,371,598
Unaligned (%)	98.01	96.96	98.95	98.28
Misassemblies	17	1	58	83
N50	293	245	302	530
NG50	835	715	1,468	1,811
LG50	2,275	2,719	1,255	893
LGA50	Not provided	Not provided	Not provided	Not provided

Cell 2

Cell 3	HyDA: Individual	HyDA: Colored	SPAdes	IDBA-UD
Genome Fraction	82.41	85.138	90.153	87.424
Unaligned bases	105,001,905	107,105,806	260,637,410	124,303,766
Unaligned (%)	95.42	95.17	97.82	96.72
Misassemblies	12	2	25	133
N50	291	269	287	680
NG50	946	900	1,861	3,478
LG50	2,096	2,196	991	462
LGA50	4,423	5,802	2,010	535

Cell 3

Cell	HyDA: Individual	HyDA: Colored	SPAdes	IDBA-UD
Genome Fraction	68.313	79.72	84.665	78.146
Unaligned bases	52,236,468	63,613,878	168,613,345	79,390,396
Unaligned (%)	93.54	93.03	97.36	95.56
Misassemblies	7	2	30 95	
N50	328	251	361	618
NG50	929	739	1,812	2,330
LG50	2,064	2,633	969	707
LGA50	3,524	6,115	1,549	916

Cell 4

Cell	HyDA: Individual	HyDA: Colored	SPAdes	IDBA-UD
Genome Fraction	77.521	83.974	88.484	84.306
Unaligned bases	48,072,495	58,321,699	154,999,418	74,215,898
Unaligned (%)	92.01	91.96	96.88	94.91
Misassemblies	7	2	25	129
N50	327	247	347	611
NG50	931	724	1,742	2,715
LG50	2,012	2,681	972	573
LGA50	3,096	5,827	1,460	657

Cell 5

Table 4.9: The assembly results of single-cell MALBAC data sets are not reliable due to high number of chimeric contigs. Also, more than 90% of the contigs are error and are not aligned to their target genome. Furthermore, the low N50 and NG50 indicates lack of contiguity information. This evaluation is done by QUAST for assembly of E. coli cells 1-5 datasets as reported previously [8] using HyDA (individual and colored), IDBA-UD, and SPAdes [5]. All statistics are based on contigs of size $\geq$ 100 bp.

chimeric assemblies by detecting and separating shared and exclusive pieces of sequence for input data sets. This suggests that in lieu of single cell assembly, which can lead to failure and loss of the sample or significantly increase sequencing expenses, the co-assembly method can hedge against that risk. Our single cell co-assembler HyDA proved the usefulness of the co-assembly concept and permitted the study of three bacteria. The improved assembly gave insight into the metabolic capability of these microorganisms thereby proving a new tool for the study of uncultured microorganisms. Thus, the co-assembler can readily be applied to study genomic content and the metabolic capability of microorganisms and increase our knowledge of the function of cells related to environmental processes as well as human health and disease. The colored de Bruijn graph uses a single k-mer size for all input data sets,

	HyDA		SPAdes		IDBA-UD
	Individual	Colored	Iterative	Single k	
Cell 1	98.36	8.17	291.59	325.68	236.18
Cell 2	134.58	7.7	417.8	431.96	325.22
Cell 3	40.88	11.09	171.56	154.81	89.12
Cell 4	70.17	9.14	207.70	228.09	159.66
Cell 5	52.85	10.32	154.58	178.95	117.29

Number of Mismatches per 100 kbp

	HyDA		SPAdes		IDBA-UD
	Individual	Colored	Iterative	Single k	
Cell 1	12.02	0.56	33.6	35.85	30.37
Cell 2	15.35	0.49	42.18	45.57	35.92
Cell 3	5.07	1.24	16.69	16.47	12.15
Cell 4	7.98	0.92	22.71	25.67	19.20
Cell 5	6.73	0.95	17.66	21.00	16.0

Number of indels per 100 kbp

Table 4.10: Number of error per 100 kbp: Co-assembly purifies the chimeric contigs of noisy data sets and decreases the error rate. The evaluation is done by QUAST [4] for assembly of E. coli cells 1-5 [8] using HyDA (individual and colored) and IDBA-UD and SPAdes [5]. To indicates that the high error is not due to the iterative algorithm, the result of SPAdes assembly with single k = 55 is included. All statistics are based on contigs of size $\geq$ 100 bp.

which has to be chosen based on the minimum read length across all data sets. For instance, *Smithella MEK03* input reads are longer (58 bp) than the reads in some of the other data sets, while the *Smithella MEK03* assembly contains many short contigs because of the small k-mer size (k=25) dictated by the shorter reads. This minor disadvantage can be remedied by using advanced assembly features such as variable k-mer size, which also can recover more contiguity information even in the case of similar length for all input read sequences.

5 ITERATIVE ASSEMBLY USING VARIABLE K

5.1 Introduction

Today, de Bruijn graph approach is adopted by many *de novo* assembly tools to sequence a genome using short reads generated by next generation sequencing platforms [50, 49, 32, 44]. Each vertex of a de Bruijn graph represents a unique subsequence of a specific length k, which is called k-mer. Existing an edge from vertex a to vertex b indicates that only the first character of a and the last character of b are different and hence the rest of $k-1$ characters of their sequence are identical. That is, connected vertices in the graph are consecutive in a read and also each read is represented by a path of $k-1$ vertices. After the graph construction stage, the errors (nodes with low coverage value) are eliminated and the pruned graph is condensed to conclude the assembly process. The final result of the assembly process is a set of condensed nodes, which are called contigs.

Some important challenges of assembly using de Bruijn graph are [43]:

- errors and contamination, which generate incorrect k-mers.
- low covered or blackout regions, which break the graph into some disconnected sub-graphs; especially when the k is large.
- repeated regions, which introduce branches; especially when the k is small.

The assembly process is extremely sensitive to the value k. To have a high quality assembly result, k should be chosen carefully. When a small k is used, many k-mers are overlapped with large number of regions along genome and this repetitious regions make many branches in the de Bruijn graph, and consequently generate short contigs. Therefore,

the final result is a set of short contigs, which does not contain much contiguity information. The evaluation factors such as N50 and largest contigs of such assembly are poor.

However, increasing k does not necessarily improve the quality of assembly since the number of gaps (especially in the low covered regions) grows when the graph is constructed with a high value of k. In this case, many consecutive regions of the genome are disconnected in the graph due to the lack of overlap between their representing k-mers.

To have a high quality assembly result that preserves the contiguity information, the k value should be chosen intently. An alternative method of assembling a genome using a single efficient k is an iterative assembly using number of different k. This type of assembly can benefit from the advantages of both small and large k. In each round of this method, the assembly is re-done by an increasing k. The short contigs generated in the an iteration with small k, fill the gaps between longer contigs generated by larger k in the next round and concatenate them. Hence, the assembly result contains the contiguity information obtained by both small and large k. The method is used in some single cell assembly tools such as SPAdes [5] and IDBA-UD [9].

5.2 Variable k Graphs

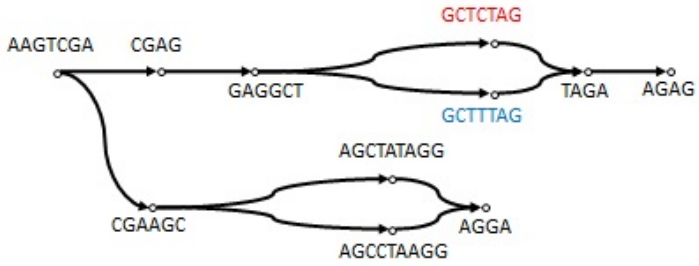
Iterative assembly with variable k outperforms an assembly method that uses a constant k in the term of contiguity information [5, 9]. In each round of a simple iterative assembly process, the contigs generated in previous round are added to the primary sequencing read dataset and the the augmented dataset (consisting of the sequencing read and the contigs of previous round) is assembled using a new k.

Besides adapting the method to the colored de Bruijn graph, we introduce a new structure to use as the information holder to help each iteration to benefit from previous one. In each iteration, HyDA combines the read dataset and the result of previous round, then, assembles the extended dataset with the new parameters assigned to current round

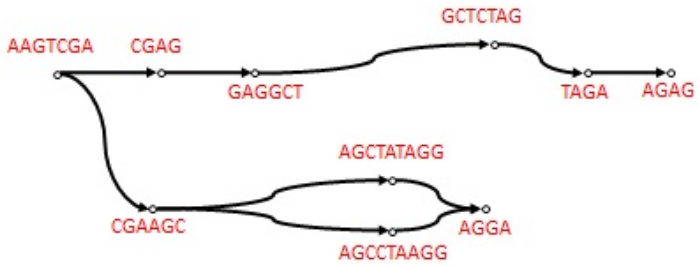
(such as k and cut-off). The assembly information of each round used in the next iteration are the information stored in a novel structure called *extended contig*. A colored extended contig is the longest path of contigs of corresponding color without any ambiguity. Figure 5.1 illustrates the construction of colored extended contigs. The graph for each colored dataset is extracted and then the extended contigs are built for each extracted uni-colored graph. The nodes with one incoming/outgoing edge are covered in extended contig sets once, while nodes with more than one incoming/outgoing edge appear in a number of extended contigs (see Figure 5.1).

Figure 5.1-a represents a colored de Bruijn graph, which contains the nodes of red and blue datasets. Uni-colored nodes are displayed in their own color and bi-colored node are illustrated in black. Figure 5.1-b represents the red de Bruijn graph inside the graph (a), which is a de Bruijn graph considering only the red nodes. Neglecting blue nodes removes a branch and generates a long one-in-one-out red path. Figure 5.1-c displays the extended paths of the graph (b). Extended contigs are the condensed nodes of the extended paths, which are AAGTCGAGGCTCTAGAGAG, AAGTCGA, CGAAGCTATAGGA, and CGAAGCCTAAGGA. The contig CGAAGC appears in 3 extended contigs.

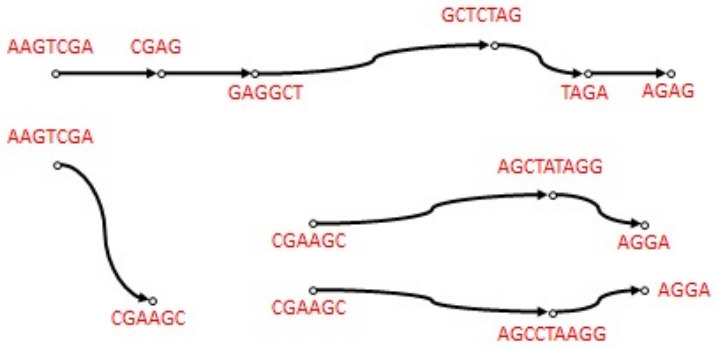
Each colored read dataset is aggregated to its own extended contig set and re-assembled with a new k in the next iteration. To have a more accurate and sensitive assembly, each round has its k also its own ϵ and cut-off. Hence, HyDA accepts different parameters for each round. In fact each assembly round in HyDA is dependent to previous ones only by the number of colors and the generated extended contigs. Algorithm 2 is the pseudo-code of each assembly iteration and Algorithm 1 is the pseudo-code of the whole iterative co-assembly method.



(a) A colred de Bruijn graph of color blue and red. The bi-colored nodes are represented in black



(b) The red de Bruijn graph extracted from graph (a)



(c) Extended paths from graph (b)

Figure 5.1: The procedure of generating extended contigs, from a bi-colored graph is illustrated (a) represents a bi-colored (red and blue) de Bruijn graph,Uni-colored nodes are displayed in their own color and bi-colored node are illustrated in black. (b) represents the extracted red de Bruijn graph. (c) displays the extended paths (extended contigs) of the red graph (b).

Algorithm 1 :Pseudo-code for iterative assembly using variable k by HyDA

function ITERATIVE-CO-ASSEMBLY($kset, cset, R$)
  Initialize $ExtendedContig$ to empty sets of size $MAXCOLORS$.
  for $<i = 1 \ldots NumberofIterations:>$ **do**
    $ExtendedContig \leftarrow$ ITERATION($kset[i], c[i], R, ExtendedContig$)
  end for
  return $ContigSet$
end function

Algorithm 2 Pseudo-code for assembly for each iteration by HyDA

function ITERATION($k, c, R, ExtendedContig$)
  for $<i = 1 \ldots MAXCOLORS:>$ **do**
    $R[i] = R[i] \cup ExtendedContig[i]$
    $Graph \leftarrow$ CONTRUCTGRAPH(k, R)
    $ExtendedContig \leftarrow$ CONDENSEGRAPH($k, Graph$)
  end for
  return $ExtendedContig$
end function

5.3 Results

5.3.1 Datasets

The single cell datasets we used are the ones published by [1]. They are one *S. aureus* and two *E. coli* isolated single cells; their genomes were extracted and amplified independently with MDA. Each amplified DNA was sequenced a number of times (the amplified DNA of *S. aureus* was sequenced in 2 lanes of Illumina GAIIx with an average coverage of $\sim 1800\times$ per lane; the resulting amplified DNA of first and second *E. coli* cells were sequenced in 4 and 3 lanes of Illumina GAIIx in 100 bp long reads with an average coverage of $\sim 600\times$ per lane respectively). Therefore, we have 9 lanes of genomic MDA sequenced from two *E. coli* biological replicates and one *S. aureus* in which the average coverage of each *S. aureus* lane is three times that of each *E. coli* lane.

Data set	No. of blackouts	Mean length	N50	Total (%)
lane 1	**94**	1220	5,558	**84K (1.8%)**
lane 2	91	1183	4,700	77K (1.7%)
lane 3	92	1159	5,842	77K (1.7%)
lane 4	88	**1225**	**6,156**	76K (1.7%)
lane 6	50	193	518	5K (0.1%)
lane 7	63	153	456	5K (0.1%)
lane 8	61	185	573	6K (0.1%)
All lanes	0	0	0	0 (0.0%)

Table 5.1: The number, mean length, N50, and total size (percentage) of blackout regions in the *E. coli* datasets reported previously [1].

5.3.2 Experiments

The strategy used by HyDA is based on using the information of all co-assembled datasets to improve the assembly result of each. Having datasets of various genomic type cannot corrupt the assembly results and generate chimeric assemblies since HyDA distinguishes between co-assembled datasets and keeps the information of each separately. To show this capability, we chose one *E. coli* MDA dataset from the datasets explained above based on maximum N50 of of blackout regions (see Table 5.1) and assembled it with some well-known single cell assembly tools and compared the results with the assembly of this dataset with HyDA when it is co-assembled with other MDA datasets explained above. Velvet-SC [1] are run with $k = 55$ and auto error cut-off. SPAdes [5], IDBA-UD [9], which are iterative assembly tools with variable k are run in default mode (without read error correction). HyDA was run with k= 21,33,55, similar to SPAdes with cutoffs 5,10,20 respectively. Other variables were default.

5.3.3 Evaluation of Assemblies

We used Quast [4] which is a standard assembly evaluation tool. Table 5.2 displays the evaluation results when contigs larger than 100 bases are considered. To compare more evaluation factors, such as the number of missing bases, Quast was run in Gage mode, in which the evaluation results generated by Gage [2] is provided.

Assembly	IDBA-UD	SPAdes	Velvet-SC	HyDA
Total length	4,872,368	4,951,781	4,653,766	**5,266,742**
Unaligned length	408,740	491,827	**325,988**	679,777
Missing ref bases	137,100	125,296	26,1858	**1,576**
%	2.95	2.70	5.64	**0.03**
# misassemblies	179	1	1	**0**
# misassembled contigs	84	1	1	**0**
Misassembled contigs length	60,936	24,274	24188	**0**
# mismatches per 100 kbp	2.01	5.59	2.98	**1.20**
# indels per 100 kbp	0.23	0.56	1.65	**0.13**
N50	**46,468**	29,753	17,039	40,158
NG50	**49,042**	32,596	17,039	44,908
NA50	**46,468**	29,753	16,963	40,158
NGA50	**49,042**	32,596	16,969	44,908
g-Corrected NG50	**48,854**	32,362	16,774	44,741
Largest contig	129,001	121,162	120,295	**221,405**
Largest alignment	129,001	121,109	120,295	**221,352**

Table 5.2: The evaluation results obtained from Quast [4] for assembly of *E. coli* using Velvet-SC [1], SPAdes [5], IDBA-UD [9], and HyDA. HyDA result is the extracted assembly of an iterative k co-assembly of 9 MDA lanes from two *E. coli* and one *S. aureus* cells. All statistics are based on contigs of size $\geq$ 100 bp. Quast was in Gage mode.

The advantages of using co-assembly method is related to retrieving blackout regions due to MDA bias and detecting more bases. HyDA covers almost all the genome and only misses small regions of size 1,576 bases, which is 0.03% of the genome, while other tools loose more than 100 times more, in which IDBA-UD, SPAdes, and Velvet-SC miss 137,100 bp (about 2.95%), 125,296 bp (about 2.70) and 261,858 bp (about 5.64%) respectively. The technique to find blackout regions works for both genome and contamination. Hence, as the number of detected bases increases, more contamination regions (unaligned bases) are assembled too.

The quality of the co-assembly result in other evaluation factors is better or at least similar to other presented tools. One important evaluation factor is related to the misassemblies and number of chimeric contigs. Quast defines misassembled contigs as chimeric contigs in which flanking sequences are located more than 1kbp afar each other in the reference genome. These There is no misassembly in the result generated by HyDA while Spades,

and Velvet-SC generate one misassembly contig respective 24,274 bp, and 24,188 bp long. On the other hand, 84 contigs of IDBA-UD assembly in total of 60,936 bp are misassemblies and Quast reports 179 misassembly errors in the IDBA-UD result. This suggests that IDBA-UD mistakenly concatenates contigs, which increases both the number of misassemblies and contiguity evaluation factors, such as N50 family.

Quast defines N50 as the length in which those contigs longer than it cover half of the assembly; NG50 is similar to N50 except that reference genome; NA50 is similar to N50 and NGA50 is similar to NG50 except they are calculated based on the aligned and corrected contigs, in which misassembled contigs are broken out based on the alignment location. These definitions are similar in other genomic quality assessment tools [**?**], [2], and [3] and display the ability of the assemblers in generation of longer contigs as one important factor of a good assembly. The 40 kbs N50 of the assembly generated by HyDA is longer than the N50 generated by Velvet-SC and SPAdes which are about 17 and 30 kbs respectively. Although the HyDA N50 is shorter than the 46 kbs N50 generated by IDBA-UD, IDBA-UD gains this high value by loosing the accuracy and generating a significant number of misassembly errors. Similar to N50, HyDA generates a high NG50, which is about 45 kbp, while IDBA-UD, SPAdes and Velvet-SC generates N50s, about49, 32, 17 kbp respectively. Since there is no error in HyDA assembly its N50 and NA50 are exactly the same like its NG50 and corrected NG50. However, other tools' assemblies contain misassemblies and their corrected factors, NA50 and NGA50, are slightly different from N50 and NG50.

The strength of co-assembly process (to find more bases) and iterative co-assembly (to keep more contiguity information) is displayed in Table 5.3 when these type of assemblies are compared to the simple assembly of a single MDA with a constant k. All assemblies are done by HyDA in various mode with final k=55 and final coverage cutoff=20.

5.4 Time and Space Complexity

The procedure of generating a genome assembly using de Bruijn graph can be divided into 4 processes, which each has its own computational requirements. The first process is to

	individual assembly	co-assembly	iterative co-assembly
Genome fraction (%)	92.512	97.455	**98.388**
N50	30,285	36,320	**40,993**
NG50	30,285	40,595	**44,741**
Largest contig	132,865	179,118	**221,501**

Table 5.3: Comparison between individual assembly, co-assembly and iterative co-assembly with HyDA.

generate k-mers from the sequencing reads. Having n reads of length d gives $n(d-k+1)$ k-mers. Hence, the time and space complexity of the generating k-mer process is $O(n)$.

The next step is to construct de Bruijn graph, which is the main bottleneck in the term of computational complexities. Since each vertex in the graph represents a unique k-mer sequence, the total number of the nodes (if there is no error) will not exceed the number of k-mers generated from the target genome sequence. That is, the number of vertices is less than $G-k+1$ for a genome of size G. However, there are some errors in the sequencing read dataset, which are represented by some vertices in the de Bruijn graph. On the other hand, repeated subsequences in the genome and also the sequence gaps in the dataset information reduce the number of de Bruijn graph nodes. Generally, the number of nodes are sensitive to the number of repeats in the genome, the sequencing read coverage, and error rate (including contamination and missing bases) [44, 57]. In the case of having resealable depth of coverage, the genome size G is much less than the total length of reads, and the maximum number of vertex is $O(G)$. Obviously, more repentant in the genome means less the number of vertex.

To add each k-mer to the graph, the whole de Bruijn graph should be examined to find a sequence match. If the matched sequence exists, the coverage of the vertex is increased once, unless a new vertex is generated and added to the graph. The search can be done by various methods. For example, a search strategy based on the Dijkstra algorithm can be done using a Fibonacci heap in a time of $O(vlogv)$ in which v is the number of vertex [58].

After graph construction, the process of error removal and graph condensation is done. The required time for this stage is dependent to the cut-off threshold. Since in the

single cell sequencing methods, we use an iterative error removal process with different cut-off threshold in each round, the time complexity is much sensitive.

In co-assembly process, the number of iterations is constant for all colored dataset. The maximum cut-off value and increasing value for each step are calculated for all colored datasets based on the single value *Threshold* and each colored average coverage obtained from the *Graph* (See Algorithm 3).

Let $T = \{t_1, t_2, t_3, \ldots, t_n\}$ be a set of cut-off threshold of n co-assembled dataset and $A = \{a_1, a_2, a_3, \ldots, a_n\}$ be the average coverage of those datasets. If M represents the maximum threshold, which is the cut-off parameter of HyDA, the items in T set are:

$$\forall t_i \in T : \{t_i = M \times a_i\} \tag{5.1}$$

Let the number of iterations is represented by N. The number of iterations is dependent to the maximum cut-off threshold. In co-assembly, the number of iterations is calculated based on the largest dataset, which is the one with the largest average coverage and represented by $MAX(T)$:

$$N = Max(T) - 1 \tag{5.2}$$

The cut-off threshold for the largest dataset is incremented once in each iteration. Other co-assembled datasets have their own incremental value. Let $V = \{v_1, v_2, v_3, \ldots, v_n\}$ is the incremental value set for the n co-assembled dataset. Its items are calculated:

$$\forall v_i \in V : \{v_i = \frac{N}{t_i} - 1\} \tag{5.3}$$

On the other hand, the time required for condensation is dependent to number of vertex. The number of vertex in each iteration is different based on the eliminated vertices as error and the re-condensed nodes. Obviously, the Maximum number, v, is when we want

Algorithm 3 :Pseudo-code for Calculating Normalized cutoff and the increasing value for each round

```
1: function CALCULATE-CUTOFF(Graph, Threshold)
2:     Initialize totalCoverage, AverageCoverage, normalizedCutoff, step to empty sets of size MAXCOLORS.
3:     Initialize AverageCoverage to empty sets of size MAXCOLORS.
4:     Initialize totalSize to Zero.
5:     for all node ∈ Graph do
6:         totalSize ← totalSize + node.size() − k + 1
7:         totalCoverage ← totalCoverage + node.coverage()
8:     end for
9:     AverageCoverage ← totalCoverage/ totalSize
10:    normalizedCutoff ← AverageCoverage ∗ Threshold
11:    step ← normalizedCutoff/ normalizedCutoff.max()
12:    return step
13: end function
```

to condense the graph for the first time (before pruning the graph). After that, we will have less number of nodes and more contiguity information.

Finally, after the last error removal/condensation process, the augmented nodes are reported as the final result and are printed out as the contig set.

In HyDA, we do the assembly process in three steps:

1. `import`
2. `assemble-unitig`
3. `assemble-finish`

5.4.1 Import

In the first step, `import`, we take all the reads and do some preprocess on them, including removing non-informative data and assigning library number, in time and space of $O(N)$, in which N is the total length of the reads. There is no difference between colored and normal assembly in this stage. Figure 5.2 time complexity for `import` process based on 9 datasets of 2 of *E. coli* cells and one *S. aureus* explained before [1]. The number of reads for each dataset them and the exact required time for `import` are indicated in table 5.4.

Cell	Dataset	number of reads	time (s)	time/read
E. coli- Cell 1	Dataset 1	29 M	432	1.50×10^{-2}
	Dataset 2	32 M	486	1.54×10^{-2}
	Dataset 3	32 M	490	1.51×10^{-2}
	Dataset 4	32 M	486	1.51×10^{-2}
E. coli- Cell 2	Dataset 5	27 M	411	1.50×10^{-2}
	Dataset 6	26 M	400	1.51×10^{-2}
	Dataset 7	24 M	371	1.52×10^{-2}
S. aureus	Dataset 8	67 M	1023	1.53×10^{-2}
	Dataset 9	66 M	1013	1.53×10^{-2}

Table 5.4: The number of reads and time required to import them for 9 datasets of 2 of *E. coli* cells and one *S. aureus* cell [1].

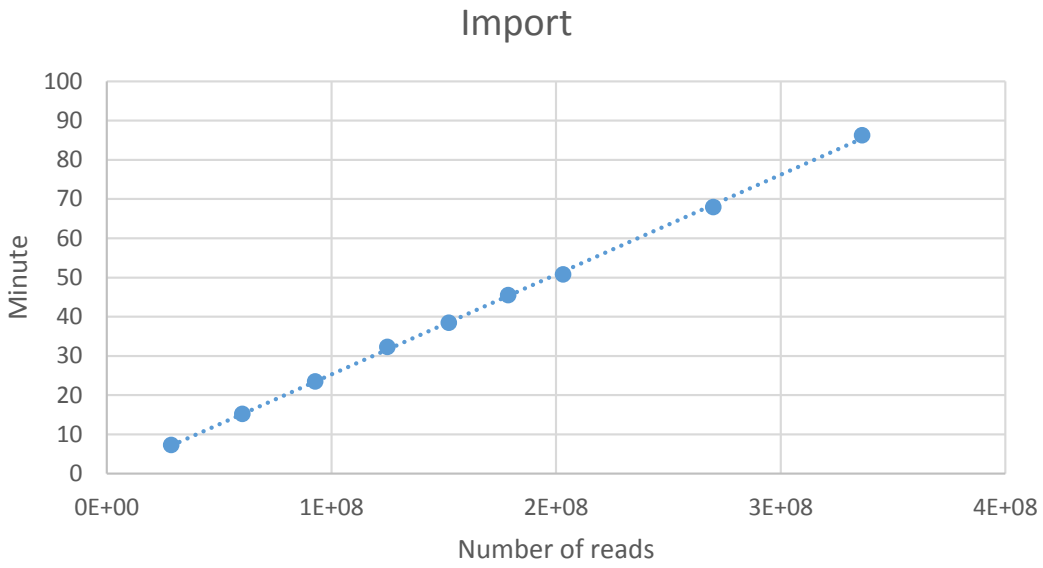


Figure 5.2: The time complexity of `import` stage of HyDA is illustrated based on the real data of 9 lanes of emphE. coli and *S. aureus* cells [1]. The slope is 0.015. The exact number of reads for each dataset is presented in Table 5.4. Each points indicated the time required for importing reads of an accumulated datasets start from dataset 1 to the aggregated of datasets 1-9.

Cell	Dataset	number of reads	time (s)	time/read
E. coli- Cell 1	Dataset 1	29 M	406×10^5	1.41
	Dataset 2	32 M	464×10^5	1.47
	Dataset 3	32 M	450×10^5	1.38
	Dataset 4	32 M	452×10^5	1.41
E. coli- Cell 2	Dataset 5	27 M	385×10^5	1.41
	Dataset 6	26 M	388×10^5	1.47
	Dataset 7	24 M	363×10^5	1.49
S. aureus	Dataset 8	67 M	897×10^5	1.34
	Dataset 9	66 M	930×10^5	1.40

Table 5.5: The number of reads and time required for `assemble-unitig` based on 9 datasets of 2 of *E. coli* cells and one *S. aureus* cell [1]. The de Bruijn graph is constructed by k=21.

5.4.2 Assemble-Unitig

The next stage of HyDA assembly, `assemble-unitig`, is dedicated to the graph construction and also the first round of graph condensation (before error removal). The only difference between normal and colored method in this stage is related to the coverage information for each vertex. The information of the coverage in colored de Bruijn graph is stored in an array of size c, which is the color number up to a compile time constant $MAXCOLORS$, while there is only one coverage value for each vertex of normal de Bruijn graph. Since the number of colors is a constant with a small value (in compare with number of reads), the required time and complexity for both method (with identical input datasets) are not different and both are linear. (See Figure 5.5 and Table 5.5).

5.4.3 Assemble-Finish

The last step, `assemble-finish`, is allocated to iterative error removal and reporting the final output sets. Although the process of colored and normal `assemble-finish` process are pretty similar, the suitable cut-off threshold for each of assembly and co-assembly are different when identical datasets are assembled since the cut-off threshold should be chosen based on average depth of coverage. When the number of colored datasets are co-assembled each of them has its own average coverage. On the other hand, for mixed assembly, the

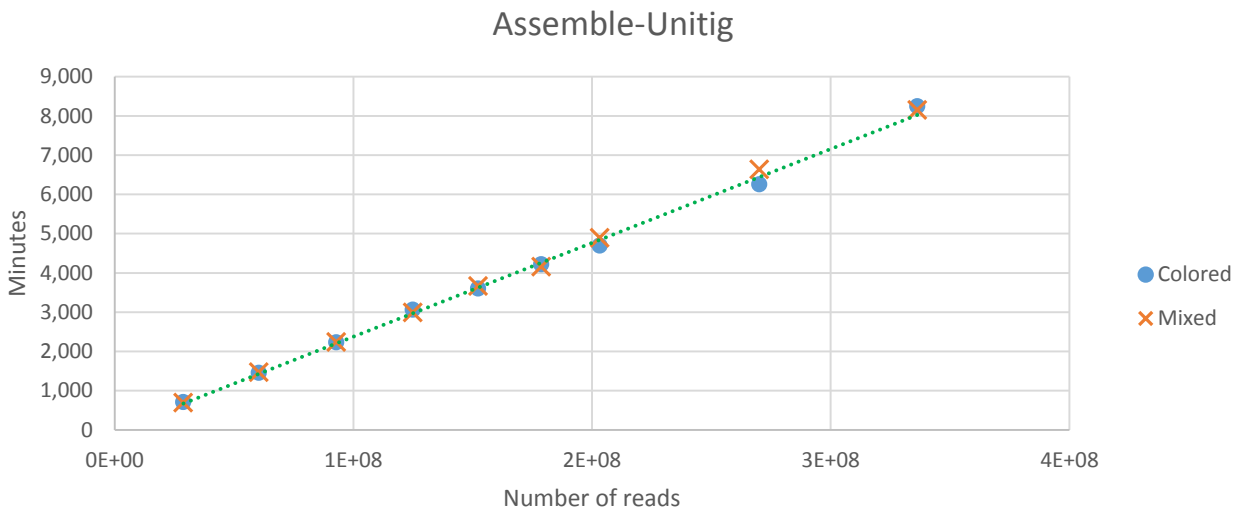


Figure 5.3: The time complexity of `assemble-unitig` stage of HyDA is illustrated based on the real data of 9 MDA lanes of *E. coli* and *S. aureus* cells [1]. In one experiment set, the datasets are assembled using colored de Bruijn graph. In the other experiment set, the datasets are mixed together and assembled using a normal uni-colored de Bruijn graph. The time required for `assemble-unitig` process for both colored and normal de Bruijn graph is similar. The process is done for $k = 21$. The slope is 1.4. The exact number of reads is presented in Table 5.5. Each points indicates the time required for running `assemble-unitig` on reads of an accumulated datasets. The first experiment is based on dataset 1 and the last experiment is based on the aggregation of datasets 1-9.

Cell	Dataset	number of reads	time (ms)	time/read
E. coli- Cell 1	Dataset 1	29 M	5.8×10^5	2.0×10^{-2}
	Dataset 2	32 M	6.4×10^5	2.0×10^{-2}
	Dataset 3	32 M	6.2×10^5	1.9×10^{-2}
	Dataset 4	32 M	6.3×10^5	1.9×10^{-2}
E. coli- Cell 2	Dataset 5	27 M	4.7×10^5	1.7×10^{-2}
	Dataset 6	26 M	4.4×10^5	1.7×10^{-2}
	Dataset 7	24 M	4.6×10^5	1.9×10^{-2}
S. aureus	Dataset 8	67 M	2.6×10^5	0.4×10^{-2}
	Dataset 9	66 M	2.8×10^5	0.4×10^{-2}

Table 5.6: The number of reads and time required for `assemble-finish` based on 9 datasets of 2 of *E. coli* cells and one *S. aureus* cell [1]. The de Bruijn graph is constructed by k=21.

datasets are accumulated and assembled as one dataset with an average coverage. Therefore, in the normal assembly, there is only one maximum cut-off threshold and one fixed increment value. On the other hand, an array of maximum cut-off threshold is generated in colored assembly, in which each item is allocated to a color. The values of this array are based on the depth of coverage of that particular colored dataset, which are calculated using a maximum cut-off constant and simple normalization process.

The process of error removal in colored assembly process is more time consuming since to prune the graph, each node should be examined for each colored valued (colored coverage and colored cut-off threshold). Another part of `assemble-finish` is generating contig/extended-contig set. The process of generating extended contigs and outputting contig/extended-contig set is done in the number of colors time when is is done only once for normal assembly. Therefore, the colored `assemble-finish` process is longer that the normal one (See Figure 5.4 and Table 5.6).

5.4.4 Iterative Assembly

To improve HyDA in the term of contiguity, the assembly process (including `import`, `assemble-unitig`, and `assemble finish`) is done number of time with various k using the information provided by previous rounds. This information is stored in an extra data

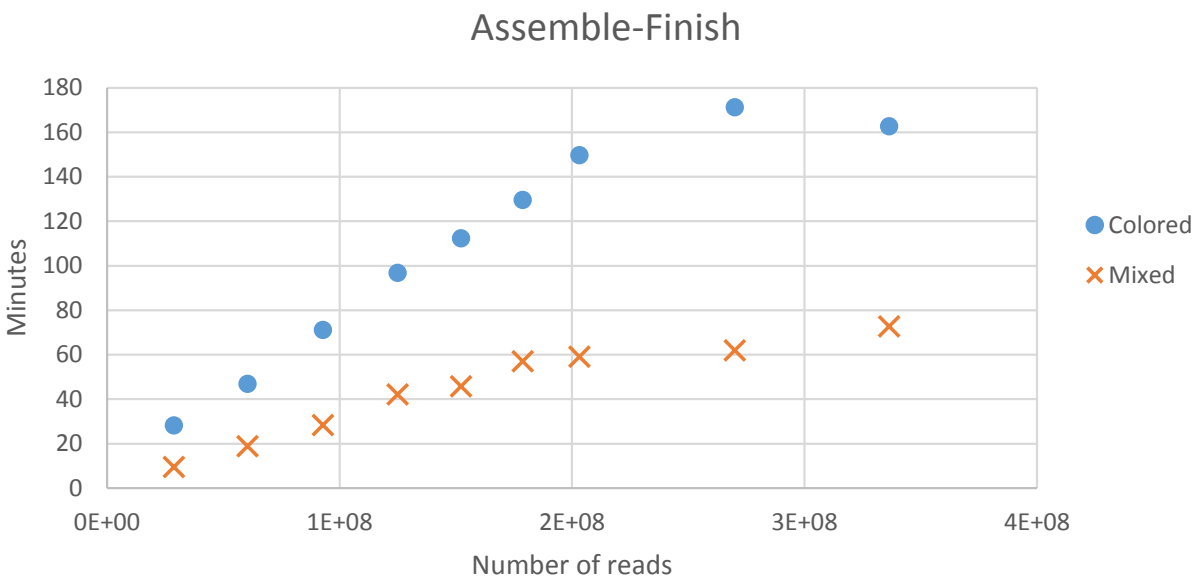


Figure 5.4: The time complexity of `assemble-unitig` stage of HyDA is illustrated based on the real data of 9 MDA lanes of *E. coli* and *S. aureus* cells [1]. In one experiment set, the datasets are assembled using colored de Bruijn graph. In the other experiment set, the datasets are mixed together and assembled using a normal uni-colored de Bruijn graph. Colored assembly requires more time for `assemble-finish` process. The process is done on the unitigs generated by $k = 21$ and specified in Figure 5.3. The exact number of reads is presented in Table 5.5. Each point indicates the time required for running `assemble-finish` on reads of an accumulated datasets. The first experiment is based on dataset 1 and the last experiment is based on the aggregation of datasets 1-9

structure, called extended contig set. It is shown that the required recourses for HyDA to assemble a number of reads as uni-colored or multi-colored data is similar in `import` and `assemble-unitig` process (see Figure 5.2, 5.3). However, `assemble-finish` in co-assembly process is longer than the process of normal uni-color assembly (see Figure 5.4).

In iterative assembly, each round is done with a different k and different cut-off threshold. Although the required resources for `import` and `assemble-unitig` are similar for uni-colored and colored assembly, these requirements are varied for iterative uni-colored assembly and co-assembly since only the input of first iteration is similar for them and next assembly rounds are done using the information generated from previous iteration (which is not similar).

Iterated colored assembly (co-assembly) and uni-colored assembly, in which the colored datasets are mixed together and assembled as one dataset, are compared in term of required resources. Also, we compare HyDA with IDBA-UD [9] and SPAdes [5]. Figure 5.5 and Table 5.7, regarding the required time, indicate that the colored HyDA and mixed HyDA assembled the same datasets at the same duration (colored HyDA is slightly longer). IDBA-UD takes less time to assemble the same datasets and SPAdes is significantly longer in compare with other methods.

Figure 5.6 and Table 5.8 are regarding required memory. Table 5.8 compares the required Virtual Memory Size (VSZ) for each experiment. Figures compares the percentage of maximum Virtual Memory Size (VSZ) divided total memory. They indicate that the colored and mixed HyDA occupy the same memory while IDBA-UD requires slightly less and SPAdes required significantly more memory.

Figure 5.7 compares the maximum required CPU for HyDA, IDBA-UD and SPAdes. It indicates colored and mixed HyDA needs the same CPU, which is more than the one, required by IDBA-UD and SPAdes. It should be noted that although IDBA-UD assembles a genome while it use less resources (time, memory, and CPU), its result is less accurate than the other compared methods (See Table 5.2). The experiments also indicate that the colored

Datasets	total Reads	HyDA-Colored	HyDa-Mixed	IDBA-UD	SPAdes
1	29 M	105	105	65	217
1-2	60 M	225	220	155	455
1-3	93 M	325	300	255	720
1-4	125 M	435	435	330	1080
1-5	152 M	520	510	395	1280
1-6	179 M	610	600	465	1560
1-7	203 M	690	670	545	1760
1-8	270 M	915	850	613	2080
1-9	336 M	1060	960	680	2430

Table 5.7: The comparison between IDBA-UD [9], SPAdes [5], and HyDA (colored and mixed) in the term of required Virtual Memory Size (KByte) is shown. The datasets are the sequencing reads of 9 MDA lane of 2 of *E. coli* cells and one *S. aureus* cell [1], which their exact number of reads are displayed in Table 5.4. For each experiment, the indicated datasets are aggregated and assembled by Spades, IDBA-UD and HyDA-Mixed, while they are co-assembled by HyDA-Colored.

Datasets	total Reads	HyDA-Colored	HyDa-Mixed	IDBA-UD	SPAdes
1	29 M	1.6×10^7	1.6×10^7	1.3×10^7	3.0×10^7
1-2	60 M	2.6×10^7	2.6×10^7	2.0×10^7	5.8×10^7
1-3	93 M	3.2×10^7	3.2×10^7	2.6×10^7	8.4×10^7
1-4	125 M	3.8×10^7	3.8×10^7	3.1×10^7	10.8×10^7
1-5	152 M	4.5×10^7	4.5×10^7	3.7×10^7	12.8×10^7
1-6	179 M	4.8×10^7	4.8×10^7	4.1×10^7	14.6×10^7
1-7	203 M	5.4×10^7	5.4×10^7	4.5×10^7	16.3×10^7
1-8	270 M	6.0×10^7	6.6×10^7	5.1×10^7	17.9×10^7
1-9	336 M	6.3×10^7	6.3×10^7	5.5×10^7	19.4×10^7

Table 5.8: The comparison between IDBA-UD [9], SPAdes [5], and HyDA (colored and mixed) in the term of required time (minutes) is shown. The datasets are the sequencing reads of 9 MDA lane of 2 of *E. coli* cells and one *S. aureus* cell [1], which their exact number of reads are displayed in Table 5.4. For each experiment, the indicated datasets are aggregated and assembled by Spades, IDBA-UD and HyDA-Mixed, while they are co-assembled by HyDA-Colored.

and Mixed HyDA require the same resources when co-assembly generates more accurate results (see Table 3.3 and Table 5.2).

5.5 Summary

To overcome the problem of contiguity loss due to limitation of k, we have improved HyDA by implementing the idea of iterative assembly using variable k , which is the method that employed by SPAdes [5] and IDBA-UD [9]. We modified the idea of iterative assembly

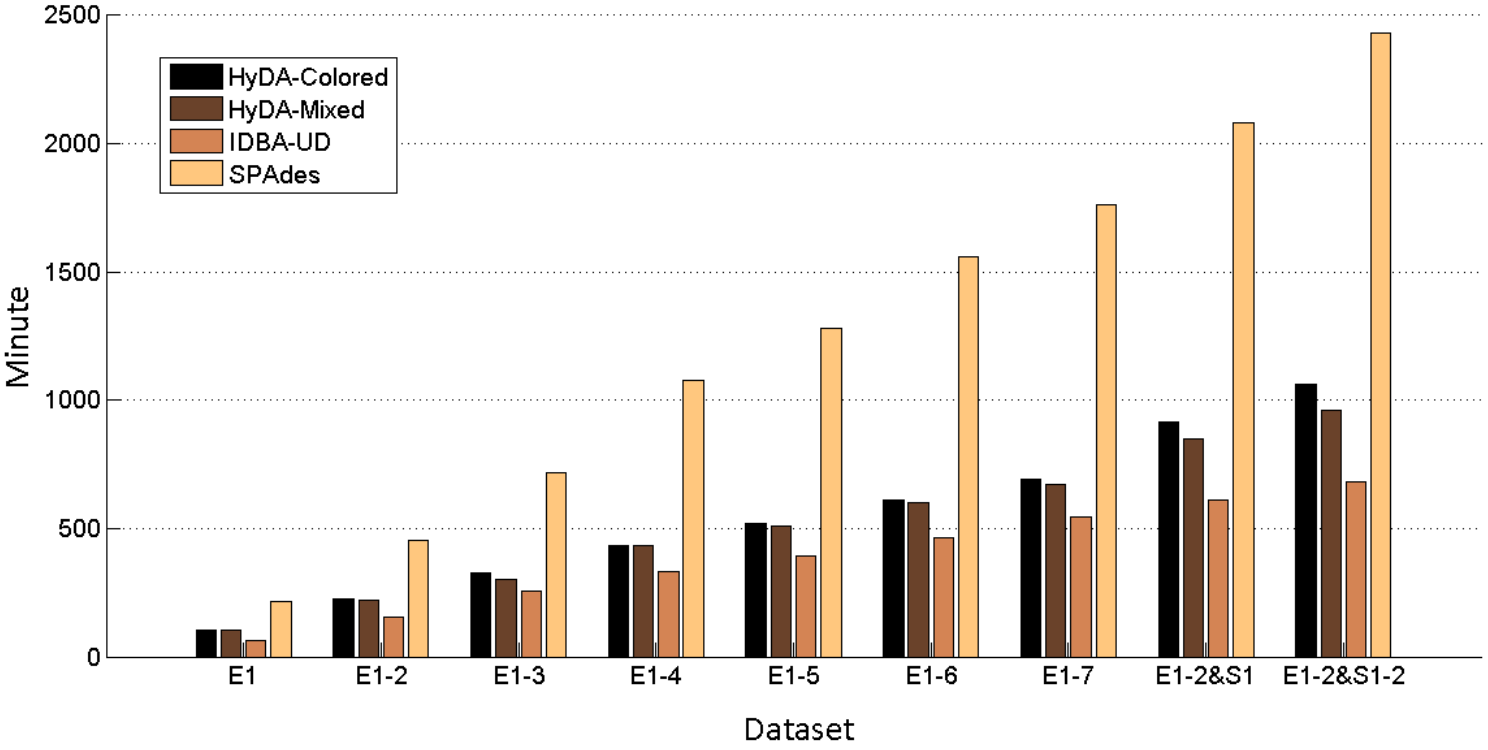


Figure 5.5: The comparison between IDBA-UD [9], SPAdes [5], and HyDA (colored and mixed) in the term of required time is shown. The datasets are the sequencing reads of 9 MDA lane of 2 of *E. coli* cells and one *S. aureus* cell [1], which their exact number of reads are displayed in Table 5.4. For each experiment, the indicated datasets are aggregated and assembled by SPAdes, IDBA-UD and HyDA-Mixed, while they are co-assembled by HyDA-Colored. The total read number and time duration of each experiment are indicated in Table 5.7.

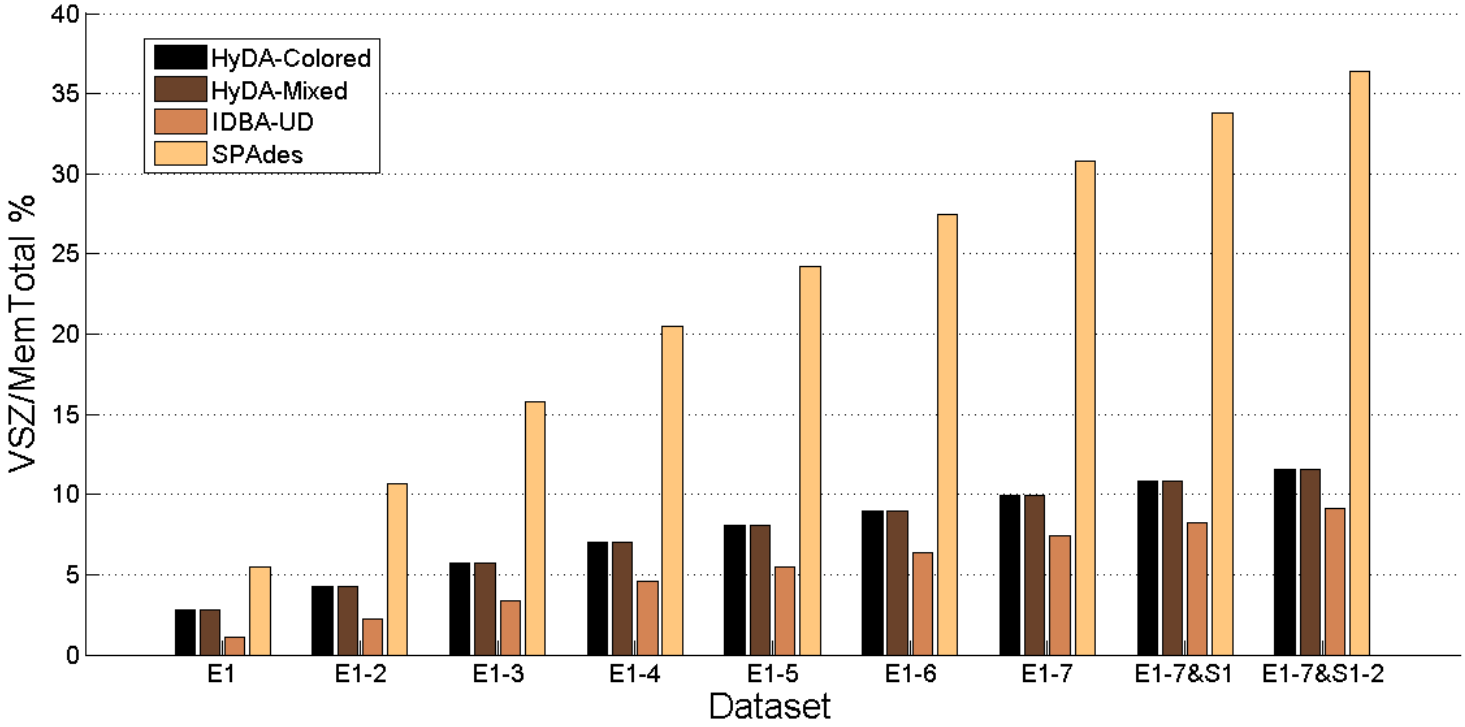


Figure 5.6: The comparison between IDBA-UD [9], SPAdes [5], and HyDA (colored and mixed) in the term of required memory is shown. The datasets are the sequencing reads of 9 MDA lane of 2 of *E. coli* cells and one *S. aureus* cell [1], which their exact number of reads are displayed in Table 5.4. For each experiment, the indicated datasets are aggregated and assembled by SPAdes, IDBA-UD and HyDA-Mixed, while they are co-assembled by HyDA-Colored. The plot is based on the VSZ/MemTotal (Virtual Memory Size / total). The total read number and maximum occupied Virtual Memory Size of each experiment are indicated in Table 5.8.

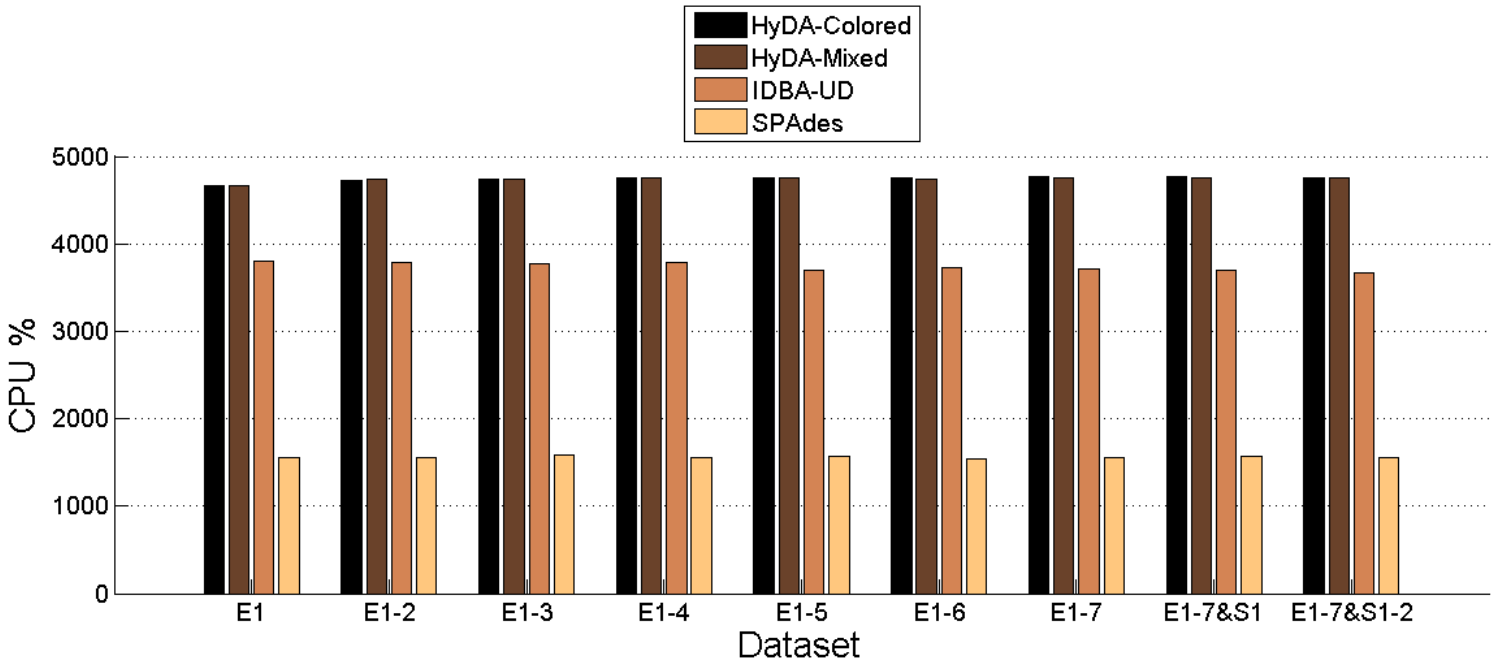


Figure 5.7: The comparison between IDBA-UD [9], SPAdes [5], and HyDA (colored and mixed) in the term of required CPU is shown. The datasets are the sequencing reads of 9 MDA lane of 2 of *E. coli* cells and one *S. aureus* cell [1], which their exact number of reads are displayed in Table 5.4. For each experiment, the indicated datasets are aggregated and assembled by Spades, IDBA-UD and HyDA-Mixed, while they are co-assembled by HyDA-Colored. The experiments are done using 64 cores.

and adapted it with the colored de Bruijn graph. Also, we introduced a new data structure, called extended contig, which is used in iterative assembly to preserve more contiguity information. The result of iterative HyDA is compared with the result generated by SPAdes and IDBA-UD. The comparison shows the strength of HyDA in term of all the evaluation factors except N50, which IDBA-UD works better. Although, IDBA-UD slightly outperforms HyDA in the terms of N50, the high error rates in IDBA-UD result indicates that the good performance on providing high contiguity information is obtained by loosing the accuracy, which make the IDBA-UD result unreliable. While, with the similar error rate, HyDA outperforms SPAdes in the therm of N50. Moreover, HyDA detects more bases of the genome due to the strength of colored assembly method using colored de Bruijn graph. Finally, we compared co-assembly method with others in the terms of the resource requirement. The comparison indicates that when we assign a unique color to each dataset and co-assembled them with HyDA instead of aggregating datasets and assembling them as one mixed dataset, the required resources are pretty similar (co-assembly is slightly more resource consuming than uni-colored assembly). Moreover the comparison results show that SPAdes consume requires more time and memory (and less CPU), while IDBA-UD requires less resources, which is expected since IDBA-UD does not preserve the accuracy and generates results with high error rate.

6 CONCLUSION

6.1 Summary of Original Contribution

Algorithmic paradigms for fragment assembly, such as overlap-layout-consensus and de Bruijn graph, depend on the characteristics of sequencing reads, particularly read length and error profile. Overlap-layout-consensus is a paradigm that is usually applied to assembly projects using long reads, and the de Bruijn graph is another widely adopted paradigm that is used for short read data sets [59]. A number of consecutive k-mers (a sequence of length k nucleotides) replace each read in the de Bruijn graph paradigm. Each k-mer is represented by a unique vertex. An edge is present between two vertices if there is a read in which the two respective k-mers are consecutively overlapping. When there are at least k consecutive common bases, reads share a vertex (respectively $k+1$ common bases for an edge) along which contigs are efficiently constructed.

Colored de Bruijn graph is a method proposed for co-assembly of multiple short read data sets [45]. It is an extension of the classical approach by superimposing different uniquely colored input data sets on a single de Bruijn graph. Each vertex, which is a representation of a k-mer, accompanies an array of colored multiplicities. In this way, input data sets are virtually combined while they are almost fully tracked, enabling separation after assembly. Iqbal *et al.* proposed the colored de Bruijn graph in Cortex [45] for variant calling and genotyping, whereas our tool Hybrid *De novo* Assembler (HyDA) [47] is developed for *de novo* assembly of short read sequences with non-uniform coverage, which is a dominant phenomenon in MDA-based single cell sequencing [1]. To fill the gaps and compare colors,

contigs in HyDA are constructed in a color oblivious manner solely based on the branching structure of the graph. First, this method rescues a poorly covered region of the genome in one data set when it is well covered in at least one of the other input data sets . Second, it allows comparison of colored assemblies by revealing all shared and exclusive pieces of sequence not shorter than k.

Single cell sequencing, which was challenging and limited for years, is now accessible and also attractive [36] for many scientific fields. It helps various type of research and projects such as antibiotics discovery [60], Earth Microbiome Project (EMP) [34], and Human Microbiome Project (HMP) [16]. The importance of the single cell sequencing is partially related to the fact that only 1% of bacteria in the environment have been cultured in the laboratory [35] since they need their natural habitant for cultivation. Also, single cell sequencing can preserve the uniqueness of each cell and its individual mutations and structural variations, which are valuable information, especially in cancer studies. Due to its importance and benefit for other research fields, *Nature* called it `Method of the Year 2013` [36].

Unfortunately, the DNA material of a single cell is not sufficient for sequencing and a whole genome amplification procedure is needed to augment the femtograms material of one cell, into micrograms of DNA material. Today, the dominant amplification method in single cell sequencing technology is the Multiple Displacement Amplification (MDA) [15, 13, 37] method even though it generates a coverage bias, in which the average coverage of the regions are different in orders of magnitude. More preciously, the amplified sequences cover some regions with high average coverage while some other regions are covered poorly or not covered at all.

The uneven depth of coverage of a single cell dataset makes the result of *de novo* assembly tools that consider only uniform sequencing depth [52, 61] inaccurate. This makes the challenges of single cell sequencing more computational rather than experimental [52]. A novel computational solution proposed by [1] overcomes the complication of MDA uneven

depth of coverage. The method is implemented into a tool named Velvet-SC [1] and adapted by other subsequent single cell assembly tools such as SPAdes [5] and IDBA-UD [9], which introduce further algorithmic features and outperform Velvet-SC [1].

However, even state of the art single cell assemblers, applying various computational strategies, cannot assemble the regions that are not covered in the MDA (blackout regions). We propose an elegant solution to retrieve those blackout regions using the information hidden in other MDA datasets. It has been shown that many blackout regions in one MDA dataset are fully covered in other MDA datasets.

Acknowledging this fact, we introduce a co-assembly strategy, which can fill the blackout regions of a MDA dataset by using the information of other co-assembled MDA datasets using the idea of colored de Bruijn graph [45].

Although, colored de Bruijn graph was first introduced for structural variation detection, we modified and implemented the algorithm for single cell sequencing. Furthermore, our algorithm modifies the iterative k assembly algorithm, which is applied by SPAdes [5] and IDBA-UD [9], and adapts it to the colored graph. It has been shown that the weakness of the co-assembly is related to breaking contigs due to various colored branches [47]. Iterative assembly with variable k overcomes the contiguity weakness. Obviously, co-assembling MDA datasets of different species without common sequencing regions does not improve the assembly result of each. Hence, to retrieve the blackout regions the co-assembled MDA datasets should be of phylogenetically close species. However, there is no harm of co-assembling MDA datasets of sufficiently distinct genomes or species.

6.1.1 Original contribution

The original contribution of this work is summarized:

- We present the colored de Bruijn graph and co-assembly process, which is implemented in a single cell assembly tool, named HyDA (Hybrid *De novo* Assembler). We explain the strategy the the proposed co-assembly mitigate the the uneven depth of coverage

of single cell datasets due to amplification bias. Also, we describe how co-assembly can retrieve poorly/not-covered reigns of a single cell dataset when they are fully covered in another co-assembled dataset.

- We present an algorithm to assemble a genome using multiple MDA datasets that are guessed to be identical. HyDA can verify whether the datasets are identical and detect the outliers. By co-assembling the MDA datasets and removing the false contig from the outlier datasets, HyDA assembles the genome with high quality. The most important weakness of this problem is contiguity loss.
- We present how the co-assembly algorithm is extended for assembling various phylogenetically close genomes at a time by clustering the input MDA datasets.
- We present a modified version of HyDA based on iterative assembly using variable k. in each iteration, besides the sequencing reads information, the data generated in last iteration in the *extendedcontig* data format are used.

We outline this PhD work in Figure 6.1.

6.2 Future Research Direction

Many genetic fields today benefit from single cell sequencing technology. Sequencing a genome using one individual cell is a widely employed technique in bacterial genomic sequencing field since only 1% of bacteria in the world have been cultured in laborites. Although single cell sequencing technology is a promising tool to approach many problems in various other genetic areas such as cancer study [36, 62]. In this work, we only target bacterial libraries which are categorized into libraries with known and un-known reference genome. We used sequencing read datasets of *E. coli* and *S. aureus* cells (with known reference genome) and sequencing read datasets obtained of from single cells of *Anaerolinea*, *Smithella*, and *Syntrophus* species.

Bacterial genome are generally smaller than genomes of other species such as animals and single cell eukaryotes and therefore contain less number of genes. They also contain

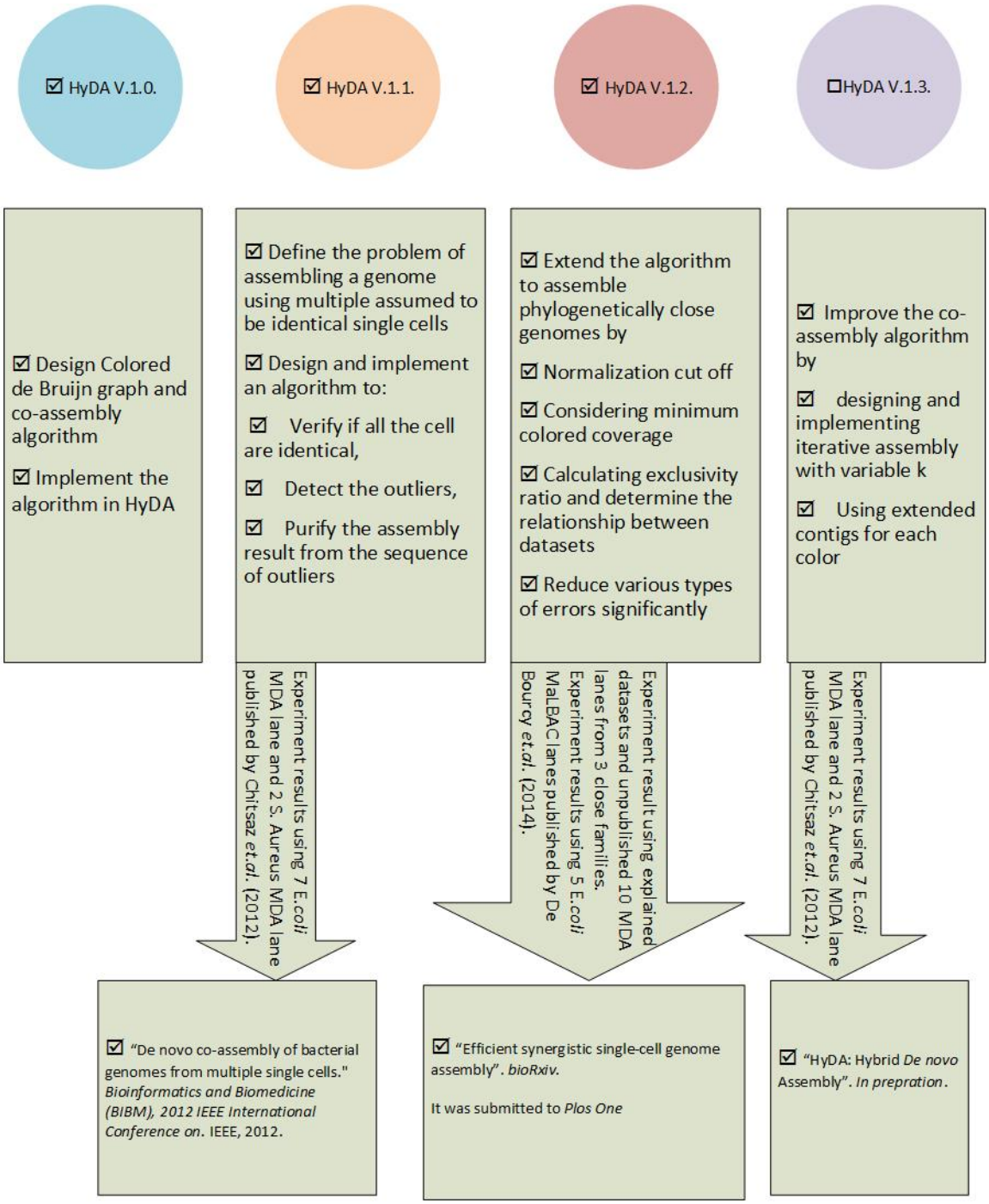


Figure 6.1: The work that has been done toward completion the dissertation is outlined. The work can be divided into 4 part, each is represented in one version of HyDA.

less variant in size in compare with other genomes. Most bacteria have about millions base pairs in their DNA ranging from 139 kbp [63] to 13,000 kbp.[64], containing a few thousand genes. The *E. coli* genome is about size of the Bacterial genomes 4 million base pairs, which is the average genome size for bacteria. On the other hand, the human genome is over 3 billion base pairs. That is, *E. coli* genome is about 0.1 % large as human genome. The computational complexities factor such as time and space are directly related to the size of sequencing read dataset. Obviously, larger genome (and larger depth of coverage) generates larger read dataset. Generally, assembling a human genome and particulary co-assembling a number of human genome is more challenging than a number of small size bacterial genome in term of required cpu, memory and time. In this work, we didn't handle the problems related to shortage of resources due to small input datasets. Since single cell sequencing is beneficial in many genomic studies of human genome, such as cancer research, one future work is to study co-assembling number of human genomes and approach the challenges due to large size of human genome. Moreover, bacterial genomes are simpler with less variant and gene and less non-coding sequence in respect to human genome [65]. Also, human genomes contain many repetitive regions, which is one main difficulty in de Bruijn graph and assembly process. Thus, co-assembling number of human genome will introduce many new challenges, which open new research projects.

Another possible further study in co-assembling genomes is related to the de Bruijn graph variable factors such as cut-off and k. In this work, we use a single cutoff value and calculate the normalized cutoff for each co-assembled dataset. In this work, the main resource to chose the cut-off value is the depth of coverage. One further research is study the impact of the quality of data, error rate, sequencing read length, genome size, and k on error removal process which leads to proposing a new technique to calculate a proper cut-off especially when there are various dataset in co-assembly method. Another important factor that the assembly results are sensitive to is k that de Bruijn graph is construct with. Since each colored dataset has its unique characteristics such as depth of coverage, sequencing read

length, error rate, and probably reference genome, choosing a proper k is a complex task, which needs more study. Especially when we use iterative assembly technique, the task is more complicated. In this work we chose k set for iterative assembly based on the default values proposed by SPAdes [5]. The impact of the combination of k and cut-off series on the quality of co-assembly result is an intrusting topic of research.

One benefit of using colored de Bruijn graph and co-assembly process is to cluster colored datasets based on their genome sequence and decide whether they are of same families or same reference genome. The algorithm we use in this work to determine the relationship between co-assembled dataset is a simple technique. A new study on how to distinguish between input datasets using colored de Bruijn graph is a promising research, which will help to sequence new species with unknown reference genome.

Finally, HyDA can be improved by employing various modules used in state of art assembly tools such as considering the information provided paired-end reads and aligning reads to the graph. Obviously, all the methods should be adapted to colored de Bruijn graph. Also, new studies are needed to explore the influence of these methods when we co-assemble various type of datasets with different characteristics.

REFERENCES

[1] Hamidreza Chitsaz, Joyclyn L. Yee-Greenbaum, Glenn Tesler, Mary-Jane Lombardo, Christopher L. Dupont, Jonathan H. Badger, Mark Novotny, Douglas B. Rusch, Louise J. Fraser, Niall A. Gormley, Ole Schulz-Trieglaff, Geoffrey P. Smith, Dirk J. Evers, Pavel A. Pevzner, and Roger S. Lasken. Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. *Nature Biotech*, 29(10):915–921, Oct 2011.

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

[3] D. Earl, K. Bradnam, J. St John, A. Darling, D. Lin, J. Fass, H. O. Yu, V. Buffalo, D. R. Zerbino, M. Diekhans, N. Nguyen, P. N. Ariyaratne, W. K. Sung, Z. Ning, M. Haimel, J. T. Simpson, N. A. Fonseca, I. Birol, T. R. Docking, I. Y. Ho, D. S. Rokhsar, R. Chikhi, D. Lavenier, G. Chapuis, D. Naquin, N. Maillet, M. C. Schatz, D. R. Kelley, A. M. Phillippy, S. Koren, S. P. Yang, W. Wu, W. C. Chou, A. Srivastava, T. I. Shaw, J. G. Ruby, P. Skewes-Cox, M. Betegon, M. T. Dimon, V. Solovyev, I. Seledtsov, P. Kosarev, D. Vorobyev, R. Ramirez-Gonzalez, R. Leggett, D. Maclean, F. Xia, R. Luo, Z. Li, Y. Xie, B. Liu, S. Gnerre, I. Maccallum, D. Przybylski, F. J. Ribeiro, S. Yin, T. Sharpe, G. Hall, P. J. Kersey, R. Durbin, S. D. Jackman, J. A. Chapman, X. Huang, J. L. Derisi, M. Caccamo, Y. Li, D. B. Jaffe, R. E. Green, D. Haussler, I. Korf, and B. Paten. Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Res.*, 21:2224–2241, Dec 2011.

[4] Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. Quast: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

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

[6] Z. Taghavi, N. S. Movahedi, S. Draghici, and H. Chitsaz. Distilled single-cell genome sequencing and de novo assembly for sparse microbial communities. *Bioinformatics*, 29(19):2395–2401, Oct 2013.

[7] R. K. Aziz, D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko. The RAST Server: rapid annotations using subsystems technology. *BMC Genomics*, 9:75, 2008.

[8] Charles FA de Bourcy, Iwijn De Vlaminck, Jad N Kanbar, Jianbin Wang, Charles Gawad, and Stephen R Quake. A quantitative comparison of single-cell whole genome amplification methods. 2014.

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

[10] Wikipedia. Single cell genome sequencing workflow@ONLINE, 2013.

[11] Chenghang Zong, Sijia Lu, Alec R Chapman, and X Sunney Xie. Genome-wide detection of single-nucleotide and copy-number variations of a single human cell. *Science*, 338(6114):1622–1626, 2012.

[12] J. S. McLean, M. J. Lombardo, J. H. Badger, A. Edlund, M. Novotny, J. Yee-Greenbaum, N. Vyahhi, A. P. Hall, Y. Yang, C. L. Dupont, M. G. Ziegler, H. Chitsaz, A. E. Allen, S. Yooseph, G. Tesler, P. A. Pevzner, R. M. Friedman, K. H. Nealson, J. C. Venter, and R. S. Lasken. Candidate phylum TM6 genome recovered from a hospital sink biofilm provides genomic insights into this uncultivated phylum. *Proc. Natl. Acad. Sci. U.S.A.*, 110(26):E2390–2399, Jun 2013.

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

[14] S. Hosono, A. F. Faruqi, F. B. Dean, Y. Du, Z. Sun, X. Wu, J. Du, S. F. Kingsmore, M. Egholm, and R. S. Lasken. Unbiased whole-genome amplification directly from clinical samples. *Genome Res.*, 13:954–964, May 2003.

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

[16] S. R. Gill, M. Pop, R. T. Deboy, P. B. Eckburg, P. J. Turnbaugh, B. S. Samuel, J. I. Gordon, D. A. Relman, C. M. Fraser-Liggett, and K. E. Nelson. Metagenomic analysis of the human distal gut microbiome. *Science*, 312:1355–1359, Jun 2006.

[17] D. B. Rusch, A. L. Halpern, G. Sutton, K. B. Heidelberg, S. Williamson, S. Yooseph, D. Wu, J. A. Eisen, J. M. Hoffman, K. Remington, K. Beeson, B. Tran, H. Smith, H. Baden-Tillson, C. Stewart, J. Thorpe, J. Freeman, C. Andrews-Pfannkoch, J. E. Venter, K. Li, S. Kravitz, J. F. Heidelberg, T. Utterback, Y. H. Rogers, L. I. Falcon, V. Souza, G. Bonilla-Rosso, L. E. Eguiarte, D. M. Karl, S. Sathyendranath, T. Platt, E. Bermingham, V. Gallardo, G. Tamayo-Castillo, M. R. Ferrari, R. L. Strausberg, K. Nealson, R. Friedman, M. Frazier, and J. C. Venter. The Sorcerer II Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific. *PLoS Biol.*, 5:e77, Mar 2007.

[18] Illumina. Truseq nano dna sample preparation kit@ONLINE, 2013. Pub. No. 770-2013-012.

[19] Illumina. Nextera dna sample preparation kit.@ONLINE, 2014. Pub. No. 770-2011-021.

[20] Robert Pinard, Alex de Winter, Gary J Sarkis, Mark B Gerstein, Karrie R Tartaro, Ramona N Plant, Michael Egholm, Jonathan M Rothberg, and John H Leamon. Assessment of whole genome amplification-induced bias through high-throughput, massively parallel whole genome sequencing. *Bmc Genomics*, 7(1):216, 2006.

[21] JA Esteban, Margarita Salas, and Luis Blanco. Fidelity of phi 29 dna polymerase. comparison between protein-primed initiation and dna polymerization. *Journal of Biological Chemistry*, 268(4):2719–2726, 1993.

[22] Kenneth R Tindall and Thomas A Kunkel. Fidelity of dna synthesis by the thermus aquaticus dna polymerase. *Biochemistry*, 27(16):6008–6013, 1988.

[23] Tanja Woyke, Damon Tighe, Konstantinos Mavromatis, Alicia Clum, Alex Copeland, Wendy Schackwitz, Alla Lapidus, Dongying Wu, John P McCutcheon, Bradon R McDonald, et al. One bacterial cell, one complete genome. *PLoS One*, 5(4):e10314, 2010.

[24] Sijia Lu, Chenghang Zong, Wei Fan, Mingyu Yang, Jinsen Li, Alec R Chapman, Ping Zhu, Xuesong Hu, Liya Xu, Liying Yan, et al. Probing meiotic recombination and aneuploidy of single sperm cells by whole-genome sequencing. *Science*, 338(6114):1627–1630, 2012.

[25] Timothy Daley and Andrew D Smith. Modeling genome coverage in single-cell sequencing. *Bioinformatics*, page btu540, 2014.

[26] Robert D Fleischmann, Mark D Adams, Owen White, Rebecca A Clayton, Ewen F Kirkness, Anthony R Kerlavage, Carol J Bult, Jean-Francois Tomb, Brian A Dougherty, Joseph M Merrick, et al. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, 269(5223):496–512, 1995.

[27] Nicholas J Loman, Chrystala Constantinidou, Jacqueline ZM Chan, Mihail Halachev, Martin Sergeant, Charles W Penn, Esther R Robinson, and Mark J Pallen. High-throughput bacterial genome sequencing: an embarrassment of choice, a world of opportunity. *Nature Reviews Microbiology*, 10(9):599–606, 2012.

[28] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of books in the mathematical sciences. W. H. Freeman, 1979.

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

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

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

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

[33] Paul A Pevzner, Haixu Tang, and Glenn Tesler. De novo repeat classification and fragment assembly. *Genome research*, 14(9):1786–1796, 2004.

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

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

[36] Nature. Method of the year 2013. http://www.nature.com/nmeth/journal/v11/n1/full/nmeth.2801.ht
2001.

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

[38] José M Lage, John H Leamon, Tanja Pejovic, Stefan Hamann, Michelle Lacey, Deborah Dillon, Richard Segraves, Bettina Vossbrinck, Antonio González, Daniel Pinkel, et al. Whole genome analysis of genetic alterations in small dna samples using hyperbranched strand displacement amplification and array–cgh. *Genome Research*, 13(2):294–307, 2003.

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

[40] Saiful Islam, Una Kjällquist, Annalena Moliner, Pawel Zajac, Jian-Bing Fan, Peter Lönnerberg, and Sten Linnarsson. Characterization of the single-cell transcriptional landscape by highly multiplex rna-seq. *Genome research*, 21(7):1160–1167, 2011.

[41] Nicholas Navin, Jude Kendall, Jennifer Troge, Peter Andrews, Linda Rodgers, Jeanne McIndoo, Kerry Cook, Asya Stepansky, Dan Levy, Diane Esposito, et al. Tumour evolution inferred by single-cell sequencing. *Nature*, 472(7341):90–94, 2011.

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

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

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

[45] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de bruijn graphs. *Nat Genetics*, 44:226–232, 2012.

[46] Jurgen F Nijkamp, Marcel A van den Broek, Jan-Maarten A Geertman, Marcel JT Reinders, Jean-Marc G Daran, and Dick de Ridder. De novo detection of copy number variation by co-assembly. *Bioinformatics*, 28(24):3195–3202, 2012.

[47] Narjes S Movahedi, Elmirasadat Forouzmand, and Hamidreza Chitsaz. De novo co-assembly of bacterial genomes from multiple single cells. In *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*, pages 1–5. IEEE, 2012.

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

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

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

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

[52] S. Rodrigue, R. R. Malmstrom, A. M. Berlin, B. W. Birren, M. R. Henn, and S. W. Chisholm. Whole genome amplification and de novo assembly of single bacterial cells. *PLoS ONE*, 4:e6864, 2009.

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

[54] Mallory Embree, Harish Nagarajan, Narjes Movahedi, Hamidreza Chitsaz, and Karsten Zengler. Single-cell genome and metatranscriptome sequencing reveal metabolic interactions of an alkane-degrading methanogenic community. *The ISME journal*, 2013.

[55] Karsten Zengler, Hans H Richnow, Ramon Rosselló-Mora, Walter Michaelis, and Friedrich Widdel. Methane formation from long-chain alkanes by anaerobic microorganisms. *Nature*, 401(6750):266–269, 1999.

[56] Brandon K Swan, Manuel Martinez-Garcia, Christina M Preston, Alexander Sczyrba, Tanja Woyke, Dominique Lamy, Thomas Reinthaler, Nicole J Poulton, E Dashiell P Masland, Monica Lluesma Gomez, et al. Potential for chemolithoautotrophy among ubiquitous bacteria lineages in the dark ocean. *Science*, 333(6047):1296–1300, 2011.

[57] Ramana M Idury and Michael S Waterman. A new algorithm for dna sequence assembly. *Journal of computational biology*, 2(2):291–306, 1995.

[58] Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. CRC press, 2004.

[59] Phillip E.C. Compeau, Pavel A. Pevzner, and Glenn Tesler. How to apply de Bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987–991, 2011.

[60] Jesse W-H Li and John C Vederas. Drug discovery and natural products: end of an era or an endless frontier? *Science*, 325(5937):161–165, 2009.

[61] T. Woyke, G. Xie, A. Copeland, J. M. Gonzalez, C. Han, H. Kiss, J. H. Saw, P. Senin, C. Yang, S. Chatterji, J. F. Cheng, J. A. Eisen, M. E. Sieracki, and R. Stepanauskas. Assembling the marine metagenome, one cell at a time. *PLoS ONE*, 4:e5299, 2009.

[62] Tal Nawy. Single-cell sequencing. *Nature methods*, 11(1):18–18, 2014.

[63] John P McCutcheon and Carol D von Dohlen. An interdependent metabolic patchwork in the nested symbiosis of mealybugs. *Current Biology*, 21(16):1366–1372, 2011.

[64] Yun-Juan Chang, Miriam Land, Loren Hauser, Olga Chertkov, Tijana Glavina Del Rio, Matt Nolan, Alex Copeland, Hope Tice, Jan-Fang Cheng, Susan Lucas, et al. Non-contiguous finished genome sequence and contextual data of the filamentous soil bacterium ktedonobacter racemifer type strain (sosp1-21t). *Standards in genomic sciences*, 5(1):97, 2011.

[65] T Ryan Gregory. Synergy between sequence and size in large-scale genomics. *Nature Reviews Genetics*, 6(9):699–708, 2005.

ABSTRACT

EFFICIENT SYNERGISTIC DE NOVO CO-ASSEMBLY OF BACTERIAL GENOMES FROM SINGLE CELL DATASETS USING COLORED DE BRUIJN GRAPH

by

NARJES SADAT MOVAHEDI TABRIZI

December 2015

Advisor: Dr. Dongxiao Zhu

Major: Computer Science

Degree: Doctor of Philosophy

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 multi-cell genome assembly due to the coverage bias (including uneven depth of coverage and region blackout) and errors caused by MDA. Computational methods try to mitigates the amplification bias. In this document we introduce a *de novo* co-assembly method using colored de Bruijn graph, which can overcome the problem of blackout regions due to amplification bias. The algorithm is implemented in a tool named HyDA (Hybrid *De novo* Assembler). HyDA can assemble various genome of phylogenetically close species simultaneously whit a high quality while it can determine the genomic relationship between each pair of co-assembled dataset based on their common and exclusive contigs. Moreover, Co-assembly can provide a high quality genome assembly from a number of guessed to be identical single cell. Since our algorithm can detect the outlier, it can generate a non-chimeric result with most recovered genome regions. Various techniques, such as iterative assembly with multiple k, are employed in HyDA, which makes it a powerful *de novo* assembly tool.

AUTOBIOGRAPHICAL STATEMENT

Narjes Sadat Movahedi Tabrizi

EDUCATION

- Doctor of Philosophy (Computer Science), 2015

  Wayne State University, Detroit, MI, USA

- 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 *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on. IEEE 2012 (pensilvania)* (2012).
2. Taghavi, Z., Movahedi, N. S., Drghici, S., 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., Zengler, K. Single-cell genome and metatranscriptome sequencing reveal metabolic interactions of an alkane-degrading methanogenic community. In *The ISME journal 2014* (2014).
4. Razavi, S. B. S., Tabrizi, N. S. M., Chitsaz, H., Boucher, C. HyDA-Vista: Towards Optimal Guided Selection of k-mer Size for Sequence Assembly. In *BMC genomics 2014* (2014).
5. Movahedi, NS., Taghavi, Z., Embree, M., Nagarajan, H., Zengler, K., and Chitsaz, H. Efficient synergistic single-cell genome assembly. In *bioRxiv 2014* (2014).
6. Movahedi, NS. and Chitsaz, H. HyDA: Hybrid De Novo assembly. (2015).