UNIVERSITY OF CALIFORNIA, SAN DIEGO

Use Solid K-mers In MinHash-Based Genome Distance Estimation

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

An Zheng

Committee in charge:

Professor Pavel Pevzner, Chair
Professor Vikas Bansal
Professor Melissa Gymrek

2017

The thesis of An Zheng is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2017

TABLE OF CONTENTS

Signature Page iii

Table of Contents iv

List of Figures v

List of Tables vi

Acknowledgements vii

Abstract of the Thesis viii

Chapter 1 Introduction and background 1
1.1 Genome distance estimation 1
1.2 Current methods 2
1.3 MinHash 3
1.4 Solid k-mer powered MinHash 5

Chapter 2 Method 7
2.1 General scheme 7
2.2 Identification of overlapping read pairs 8
2.2.1 Workflow 8
2.2.2 Data 9
2.2.3 Implementation 9
2.3 Genome identification 10
2.3.1 Workflow 10
2.3.2 Data 10
2.3.3 Implementation 10

Chapter 3 Result 15
3.1 Identification of overlapping read pairs 15
3.1.1 Performance comparison between solid k-mer powered MinHash and regular MinHash 15
3.1.2 Selecting the solid k-mer threshold 17
3.2 Genome identification 19

Chapter 4 Discussion and future work 21

Bibliography 23

LIST OF FIGURES

Figure 1.1: An example of how to use MinHash to compute the resemblance of two genome sequences. 6

Figure 2.1: A simplified example of how the sketch of a genome sequence is generated in the solid k-mer powered MinHash. 12
Figure 2.2: The workflow of the identification of overlapped reads. 13
Figure 2.3: The workflow of genome identification. 14

Figure 3.1: PR curves using the in silico genome data. 16
Figure 3.2: PR curve using the E Coli genome data. 17
Figure 3.3: Comparison of AUC values generated using different solid k-mer thresholds. 19

LIST OF TABLES

Table 3.1: K-mer frequency distribution in the E. coli genome. 18
Table 3.2: K-mer occurrence distributions of the Bacillus anthracis Ames read set and the Bacillus cereus ATCC 10987 read set. 19
Table 3.3: Rankings and p-values of Bacillus anthracis Ames and Bacillus cereus ATCC 10987 generated with different solid k-mer thresholds. 20

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my thesis advisor, Professor Pavel Pevzner, for his continuous support of my research work, for his patience and immense knowledge, and for his being a good mentor and academic role model.

I am also grateful to Professor Melissa Gymrek and Professor Vikas Bansal for their valuable suggestions to my thesis writing.

Last but not least, I would like to give my special thanks to Professor Yu Lin. Without his ideas and guidance, I would have been overwhelmed by all sorts of problems in this research project and could have given up a long time ago.

Right now, I feel any words would be too pale to convey my appreciations to all these people who gave me great help. Thank you!

ABSTRACT OF THE THESIS

Use Solid K-mers In MinHash-Based Genome Distance Estimation

by

An Zheng

Master of Science in Computer Science

University of California, San Diego, 2017

Professor Pavel Pevzner, Chair

MinHash is a popular method for genome distance estimation. However, its requirement for input data quality is relatively strict, and its performance deteriorates if the input sequences are generated from sequencers with high sequencing error rates, especially from long-read sequencers. To solve this problem, in this thesis, we use solid (frequently occurring) k-mers as input to feed MinHash, and prove the effectiveness of this solid k-mer powered MinHash by comparing its performance in genome distance estimation with regular MinHash. In addition, we also discuss how to select the optimal threshold for solid k-mers in order to make the most of our solid k-mer powered MinHash.

Chapter 1

Introduction and background

1.1 Genome distance estimation

Genome distance is a measure of the degree of similarity or overlap between genome sequences sampled from the same reference genome, from different individuals of the same species, or from individuals of different species [Nei87].

A good method to accurately estimate genome distance is required in a wide range of biological scenarios. For example, in de novo genome assembly, in order to find all overlapped sequence segments (namely, reads), we need to estimate the degree of overlap between every read pair and connect them together into a whole genome based on the estimated genome distance [KBP15].

Moreover, through the comparison of genome distances between different populations or species, we can gain insight into how certain populations or species evolved. For example, by comparing a large number of loci from Eurasian and African genome samples and analyzing the genomic distance between these two populations, researchers predicted that Eurasians were derived from African populations, and this divergence happened around 100,000 years ago [NR74]. Similarly, in [WM07], researchers illus-

trated how certain vertebrates and mammals evolved by measuring the genome distance between 28 vertebrate species.

Finally, an accurate and efficient estimator for genome distance is also critical to many metagenomic analyses, such as in the categorization of metagenome sequences with k-means [XY11] or hierarchical clustering [RR13], in which genome distance is used to quantify the degree of relationship between each pair of organisms.

1.2 Current methods

Currently, the most widely used method for genome distance estimation is sequence alignment, which is a way of arranging the sequences of DNA, RNA or protein and identifying the regions of similarity which may be caused by functional, structural, or evolutionary relationships between the sequences [Dav03].

Basic Local Alignment Search Tool (BLAST) [SFAL90] is one of the most popular alignment-based analysis tools. Since 1990 when BLAST was first published, numerous BLAST variations have been proposed and applied for different scenarios; such as PSI-BLAST for protein alignment [SFAL97] and V-BLAST for real-time analysis [PWG98], and BLAT, a BLAST-like alignment tool which is faster yet less sensitive [Ken02].

However, although alignment-based methods are effective in estimating genome distance, they suffer from substantial computational cost [KBP15]. It took more than 10 days to complete the de novo assembly of *Arabidopsis thaliana* (Ler-0) from SMRT reads (approximately 11 Gb data) [Pac16]. Even small bacterial genomes require a day to assemble using the HGAP [CSCK13] or PBcR [SKP13] pipelines.[KBP15]

Several MinHash-based and alignment-free methods were developed to address this problem. MinHash is a hash-based technique that was originally designed to es-

timate the similarity of web pages [Bro97]. It has since been applied to genome distance estimation in large genomes assembly [KBP15][BDOP16], metagenome clustering [RR13][XY11], and human phylogenetic analysis [BDOP16]. MinHash-based methods are not only much more computationally efficient than alignment-based methods, but also can maintain relatively high accuracy comparable to the alignment-based methods. [KBP15] The details of MinHash and its current applications are introduced in the following section.

1.3 MinHash

The basis of MinHash is hash functions. A hash function can be a user-defined or randomly generated function which maps data of arbitrary size to data of fixed size. Each input entry is called a key, and its corresponding result is called the hash value. Equation below is a naive example of a hash function. "%" denotes the remainder calculation. For any integer input, this function calculates the remainder of the input number divided by 10, and projects this number into a single digit (0-9).

$$Hash\ value = key\ \%\ 10\ \ (key \in \mathbb{Z})$$

To solve real world problems, there are different implementations of hash functions for different scenarios, such as dynamic perfect hashing [MDT94], rolling hashing [DL10], and universal hashing [JC79]. MinHash is based on the universal hashing method. This hashing method selects a hash function at random from a family of hash functions as the mapping scheme, in which the probability that any two distinct keys share the same hash value is 1/n, where n is the number of total hash values.

MinHash was first brought forward by Andrei Broder in 1997 in order to quickly compute the resemblance between set pairs. [Bro97] The basic idea is to use a series

of distinct universal hashing functions to randomly sample the given data sets, map the data sets into fixed-length profiles called sketches, and estimate the resemblance of every pair of sets based on their sketches. The resemblance between two sketches equals the fraction of their common sketch elements. For example, if a sketch is [1,2,3,4] and the other one is [1,2,5,6], their resemblance is 2/4, because the first two elements are the same, whereas their last two elements differ.

This technique was later introduced to tackle the computational problem of read overlap estimation in long-read assembly. [KBP15][BDOP16] Figure 1.1 shows a simplified example of the MinHash process. In real-world scenarios, researchers usually first decompose every genome read into 16-25 base pair-long k-mers (oligomers with nucleotide sequence length of k; i.e. 16-mer is a k-mer with 16 base pairs) using a sliding window. Then they utilize MinHash to map every set of k-mers into a sketch. The genome distance between genome sequences is defined as the resemblance of their sketches.

Similarly, in metagenomics analysis, researchers usually build a k-mer set for each genome sequence, and compute the sketches of these sequences. Then, they calculate the genome distance based on these sketches, and construct hierarchical structures of the input genome sequences and their corresponding organisms using clustering algorithms, such as k-means. [XY11]

However, if a genome sequence is generated by a sequencer with a high sequencing error rate, such as PacBio or an Oxford Nanopore MinION [KBP15], the performance of MinHash-based genome distance estimators will deteriorate significantly in both accuracy and efficiency, because these erroneous nucleotides will produce k-mers that do not exist in the original reference genome sequence and thus dilute the k-mer set. This results in two important disadvantages. On one hand, the hashing process has to spend more time to deal with these additional k-mers. On the other hand, the number of common k-mers

between related sequences decreases: if one or more nucleotides in a k-mer mutate, this k-mer will be treated as a completely different k-mer in MinHash. Thus, the estimated genome distance value will be lower than the true value, and this value difference will vary from sequence to sequence due to the randomness of sequencing errors. This will eventually cause some related sequence pairs to fail to be retrieved.

1.4 Solid k-mer powered MinHash

To solve the problem above, we introduce solid k-mer powered MinHash, a modified MinHash procedure, to improve robustness against sequencing errors in the input. Solid k-mers of a genome sequence are the k-mers that occur more than some number of times in the genome sequence of interest. The threshold is variable and usually predefined according to needs. The idea of solid k-mers is widely used in data pre-processing for de novo assembly [PNA11][DL15] in order to filter out regions with erroneous nucleotides derived from sequencing. In our work, we adapt this idea by pre-processing the k-mer sets into solid k-mer sets, and using these solid k-mer sets as input data to MinHash. We call this modified method solid k-mer powered MinHash.

Currently, only a few implementations, such as MASH [BDOP16], mention that these erroneous nucleotides can be a possible source of error in genome distance estimation. However, none have provided experimental proof to demonstrate that solid k-mers improve performance. Nor have they discussed how to select an appropriate threshold for solid k-mers to maximize this benefit. Thus in this thesis, we compare the performance between solid k-mer powered MinHash and regular MinHash in genome distance estimation, and discuss how to set the threshold for solid k-mers. The development and implementation of solid k-mer powered MinHash can enable calculating the genomic distances with higher speed and accuracy.

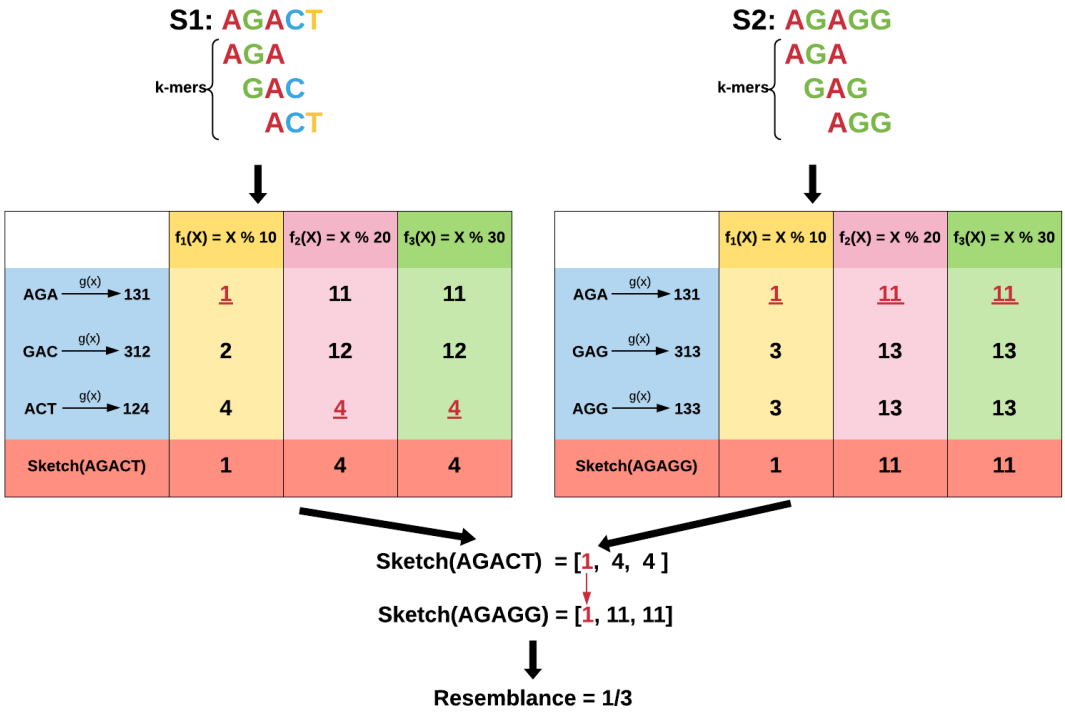

Figure 1.1: An example of how to use MinHash to compute the resemblance of two genome sequences. S1 and S2 are first decomposed into 3-mers. Each hash function $F_i(x)$ contains two steps: $g(x)$ and $f_i(x)$. $g(x)$ transcribes every A into 1, C into 2, G into 3 and T into 4. For example, AGA is transcribed as 131. The second step is defined as $f_i(x) = x\%(10 * i)$. A complete hash function is $F_i(x) = f_i(g(x))$. Each hash function maps the k-mer sets into integer numbers, and the numbers generated by different functions are shown in different colors in the figure. In every column, only the smallest integer is kept in the sketch, and the length of a sketch equals the number of hash functions used. The resemblance of S1 and S2 equals the fraction of the common elements in their sketches.

Chapter 2

Method

2.1 General scheme

The first step of solid k-mer powered MinHash is to build a solid k-mer set for each genome sequence. Here, we utilize DSK [GR13], a k-mer counting software, to find all kmers in each genome sequence and calculate their occurrence frequencies. K in our experiments is set to 16. From all the k-mers in a sequence, a subset of them whose occurrence frequencies are no less than a predefined threshold are selected to compose the solid k-mer set for the corresponding sequence.

We then apply MinHash to the solid k-mer sets to generate the sketches. For a genome sequence S, its solid k-mer set is first mapped into integers by a series of hashing functions, and then its sketch records the minimal mapped integer of every hashing function. To guarantee the sensitivity of MinHash, the number of hashing functions used in MinHash is usually 512 or 1,024 for the analysis of genomes with 1 million to 10 million base pairs. For larger genomes, such as human genomes, 2,048 or more hashing functions are required. Figure 2.1 provides a simplified example to illustrate the whole process.

To evaluate our solid k-mer powered MinHash, we test it in two different scenarios: (1) identifying overlapping reads for long-read assembly (2) identifying the closest organism to a query genome sequence from a genome database containing various genome sequences. Detailed implementations will be included in the next two sections.

2.2 Identification of overlapping read pairs

2.2.1 Workflow

Given a set of genome reads, we apply the workflow used in [KBP15] to search for potentially overlapping read pairs (figure 2.2). The process of read pair identification consists of two stages. In stage 1, the degree of overlap of a read pair is defined as the resemblance of their corresponding sketches. Any read pair with a resemblance score lower than a predefined threshold will be discarded. In stage two, instead of the sketches, the degree of overlap of a read pair is defined as the similarity of their solid k-mer sets. We utilize the Jaccard similarity coefficient (equation below) to quantify the similarity degree. Only read pairs whose Jaccard coefficient are higher than a predefined threshold (different from the one in stage 1) are kept as overlapping reads.

$$Jaccard(A,B) = \frac{\|A \cap B\|}{\|A \cup B\|}$$

This two-stage process enables us to quickly filter out read pairs that are highly unlikely to overlap, and leaves only a small number of them to be examined more carefully. Since the majority of read pairs are highly unlikely to overlap, even if we take random hits into consideration, this two-stage filter can work much more efficiently than a filter with only stage two.

2.2.2 Data

In our experiment, we separately use in silico genome data and *Escherichia coli* (E. coli) genome data as the reference genome to evaluate our solid k-mer powered MinHash. The in silico genome sequence is 1,000,000 base pairs long and was generated with a randomized algorithm. It only contains legitimate nucleotides (namely, A, C, G, and T). The E. coli data ($NC-000913$ [NCBa]), is 4,641,652 base pairs long. It is a complete genome of Escherichia coli $K-12$ MG1655.

To generate genome reads, we utilized kmerStatSimulator, a sequencing simulator used in [KBP15], to simulate the sequencing process of the PacBio sequencer (typically, substitution rate: 0.01, insertion rate: 0.12, deletion rate: 0.02) [BKS16] and the MinION sequencer (typically, substitution rate: 0.049, insertion rate: 0.078 and deletion rate: 0.051)[MJA15]. Each read was 10,000 base pairs long, and we simulated to a target reference genome with coverage of 50x. The truth sets were generated using Blasr [CT12], a read aligner designed by PacBio, which can map reads onto the reference genome and identify all overlapping pairs according to their positions.

2.2.3 Implementation

We implement this process in a python program, including the overlap identification process and evaluation process. The code can be accesses on Github (https://github.com/Pandaman-Ryan/solid_kmer_MinHash).

2.3 Genome identification

2.3.1 Workflow

In the genome identification process, instead of hashing a single read each time, we map a whole assembled genome sequence or a whole set of sequenced reads into one sketch. For every query genome sequence or read set, we compare its sketch with our database, a library of assembled genomes and read sets derived from different organisms, and compute their resemblance scores. Then these organisms are ranked according to their resemblance scores with the query genome sequence or read set. Figure 2.3 illustrates how this process works with a simplified example.

2.3.2 Data

The reference genome database used in this sub-experiment is NCBI RefSeq 70 [KPM12][NCBb] which was released in 2015. It contains genome assemblies and read sets from 54,118 organisms. These sequencing data were generated using various technologies including Illumina HiSeq, Illumina MiSeq, PacBio RSII, and Oxford Nanopore MinION.

We used two reads sets, 1D and 2D Oxford Nanopore MinION read sets of *Bacillus anthracis Ames* and *Bacillus cereus ATCC 10987*, as the query sequences. These datasets were acquired from [BDOP16].

2.3.3 Implementation

To implement this genome identification process, we apply MASH [BDOP16] in our experiment. MASH is a MinHash-based software for genome and metagenome distance estimation. It takes the query sequences and the reference genome database as

the input, and computes a ranked list of the organisms that resemble the query sequences. In addition, it enables us to set the minimum copies of each k-mer required to pass a noise filter for reads, and ignores any k-mers that fail to pass this filter. Thus we can use this parameter to configure the threshold of solid k-mers.

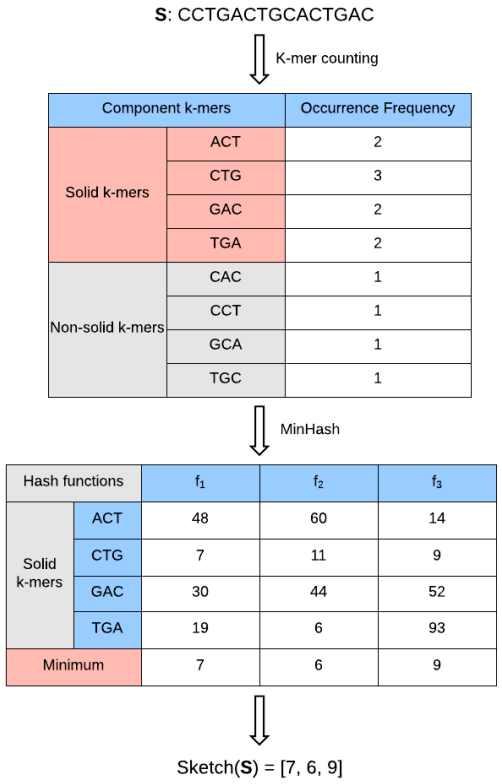

Figure 2.1: A simplified example of how the sketch of a genome sequence is generated in the solid k-mer powered MinHash. In this figure, S is a genome sequence and the occurrence frequencies of its component k-mers are shown in the top table. Here we set the threshold of solid k-mers as 2. From the occurrence frequencies table we can see that only ACT, CTG, GAC and TGA are solid k-mers under this threshold. There are 3 hash functions (f_1, f_2 and f_3) in MinHash. Their mapping schemes are randomly generated, but they satisfy the requirement that the probability that any two distinct keys share the same hash value is 1/n, where n is the number of total hash values. Every hash function maps these four solid k-mers into four integers, and the minimal one of these four integers is recorded into the sketch. The length of the sketch equals the number of hash functions used in MinHash.

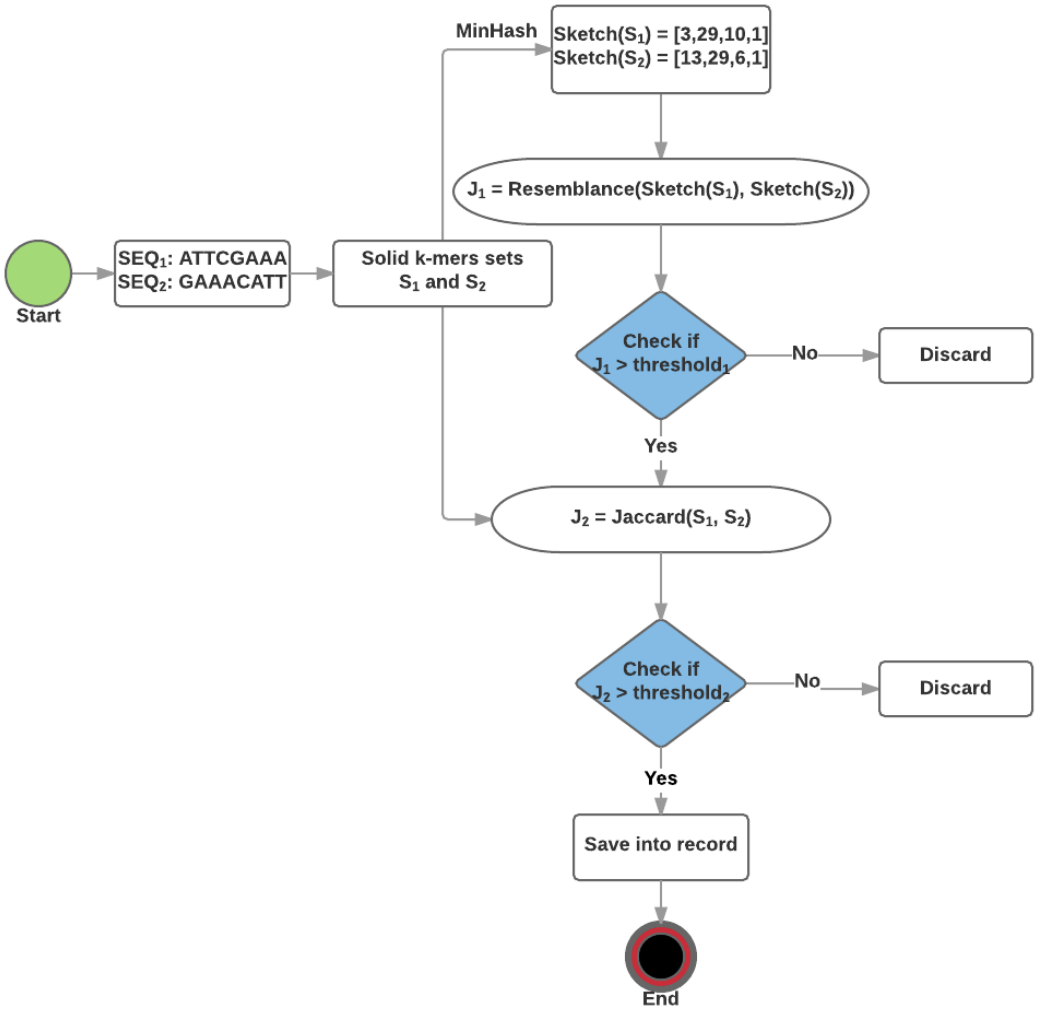

Figure 2.2: The workflow of the identification of overlapped reads. SEQ1 and SEQ2 are the pair of reads we want to check. They are taken as a pair of overlapping reads only if they successfully pass the two-stage filter. The steps for filtering are colored in blue.

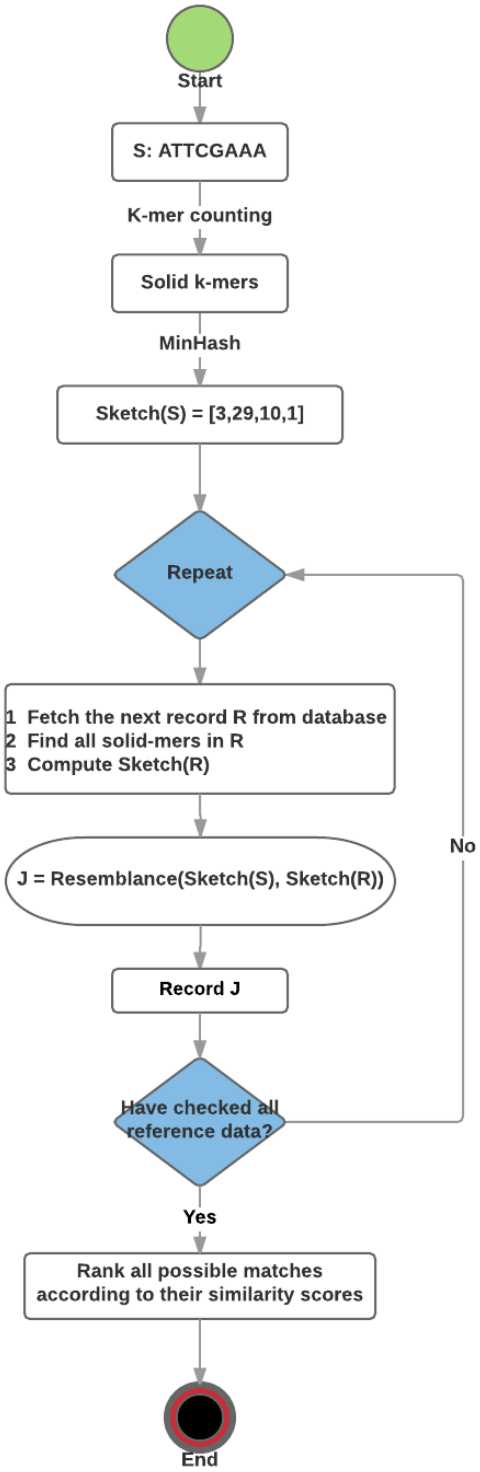

Figure 2.3: The workflow of genome identification. S is a query genome sequence. We compare it with all the genomes in the database, and rank them according to their similarity score with S.

Chapter 3

Result

3.1 Identification of overlapping read pairs

3.1.1 Performance comparison between solid k-mer powered MinHash and regular MinHash

In the identification of overlapping read pairs from read sets, we set the solid k-mer threshold as 2, and then evaluated the performance of solid k-mer powered MinHash and regular MinHash respectively on read sets from the simulated genome and the E. coli genome. For each genome, we simulated two distinct read sets using PacBio sequencer settings and MinION nanopore sequencer settings. Thus, we conducted four groups of experiments.

The threshold of the second stage filter was always set to five times as the threshold of the first stage to prevent false negatives derived from MinHash. The initial threshold thresholds for stage one and stage two were 0.00002 and 0.0001, respectively. This is already a lower bound on filter thresholds, since for 10,000 base pairs reads a single common k-mer between two sequences results in a Jaccard coefficient above

0.0001. We repeated the identification process, incrementing the first stage filter by 0.00002 and the second stage filter by 0.0001 each round, until the threshold of the second stage reached 1.

Predicted read pairs were compared with the truth sets. Precision-recall curves are in figure 4 and figure 5. Each node in these curves corresponds to one round of the experiment. Notably, the precision rates usually cannot start from zero, since even at the minimum threshold (0.0001) most non-overlapping pairs are still filtered. The recall rates also started from a non-zero value (0.1) in our experiment. We define the region inside these thresholds as ”effective area”. In the following sections, the calculation of area under curve (AUC) specifically refers to the AUC within the effective area.

Figure 3.1 shows the precision-recall curves using the in silico data. The left figure was generated using PacBio mutation configurations. The AUC of the solid k-mer powered MinHash method is 0.913692, and the AUC of the regular MinHash method is 0.708507. The right figure was generated using MinION nanopore mutation configurations. The AUC of the solid k-mer powered MinHash method is 0.821010, and the AUC of the regular MinHash method is 0.309791.

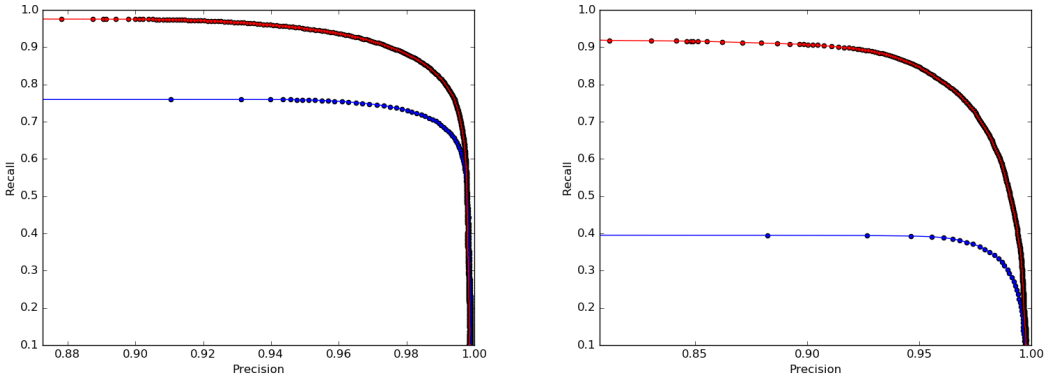

Figure 3.1: PR curves using the in silico genome data. The left figure is generated using PacBio mutation configurations. The right figure is using MinION nanopore mutation configurations. The red lines in both figures denote the solid k-mer powered MinHash method, and the blue lines denote the regular MinHash method.

Figure 3.2 shows the precision-recall curves using the E coli data. The left-side figure was generated using PacBio mutation configurations. The AUC of the solid k-mer powered MinHash method is 0.905992, and the AUC of the regular MinHash method is 0.707547. The right-side figure was generated using MinION nanopore mutation configurations. The AUC of the solid k-mer powered MinHash method is 0.734723, and the AUC of the regular MinHash method is 0.318537.

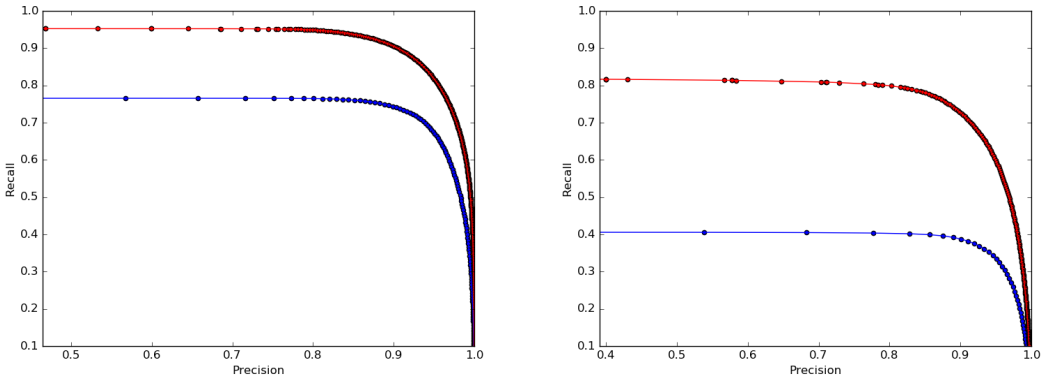

Figure 3.2: PR curve using the E Coli genome data. The left figure is generated using PacBio mutation configurations. The right figure is using MinION nanopore mutation configurations. The red lines in both figures denote the solid k-mer powered MinHash method, and the blue lines denote the regular MinHash method.

From the above results, we can see that the solid k-mer powered MinHash performed better than regular MinHash in identifying overlapping read pairs in all cases. Compared with regular MinHash, the solid k-mer powered approach is more robust to high sequencing error rates. These results were consistent in both in silico genome data and E. coli genome data.

3.1.2 Selecting the solid k-mer threshold

In the experiment above, we set the solid k-mer threshold as 2, and showed that applying solid k-mers as the input was able to increase the performance of the MinHash-based method for overlap detection. However, how to select the solid k-mer threshold

still remains as an important challenges.

Suppose we have a read set R and its reference genome S, and every read in R is sampled from S. If the error rate during sequencing is zero, every k-mer that appears in R should also appear in S. If we set k large enough, for instance k equals 16 as used above, the number of different types of k-mers in R should be around the same size as the length of S, or slightly less than the length of S but with same order of magnitude. However, in the presence of sequencing errors, R will contains many erroneous k-mers. K-mers that infrequently occur in R are very likely to be derived from sequencing errors. To remove these erroneous inputs, we can set the solid k-mer set at a certain threshold so that number of solid k-mers is around the same size as the length of S.

We evaluated our hypothesis above using the E. coli genome data. The length of this E. coli genome was 4,641,652. The k-mer occurrence frequency distribution of its read set is shown in Table 3.1. According to our hypothesis, 3 and 4 should be the best candidates for the solid k-mer threshold.

Table 3.1: k-mer frequency distribution in the E. coli genome.

K-mer occurrence	$>= 1$	$>= 2$	$>= 3$	$>= 4$	$>= 5$
Number of k-mers	225,936,345	23,043,727	4,943,888	2,124,463	1,085,457

We repeated the overlap identification experiment in the previous section eight times, each time with a different solid k-mer threshold, from 1 to 8. The values of the AUC in these eight groups are plotted in figure 3.3. From the result we can see that the AUC value peaks when the threshold is 3 or 4, consistent with our hypothesis.

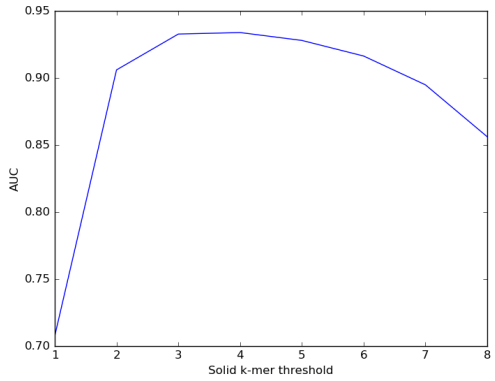

Figure 3.3: Comparison of AUC values generated using different solid k-mer thresholds.

3.2 Genome identification

Solid k-mer powered MinHash can be also applied in genome identification. We tested its ability using MASH, the MinHash-based tool which can rapidly identify isolated genomes from both assemblies and raw sequencing reads. [BDOP16]

In the genome identification experiment, we conducted 6 parallel sub-experiments with different solid k-mer thresholds, from 1 to 6. Table 3.2 shows the k-mer occurrence distributions of the two query read sets, Bacillus anthracis Ames and Bacillus cereus ATCC 10987. Since the length of Bacillus anthracis Ames is 5,227,293 bp [TDR03] and the length of Bacillus cereus ATCC 10987 is 5,432,652 [KEG], according to our theory put forward in the previous section, the optimal solid k-mer threshold for Bacillus anthracis Ames and Bacillus cereus ATCC 10987 should be around 2 to 3.

Table 3.2: K-mer occurrence distributions of the Bacillus anthracis Ames read set and the Bacillus cereus ATCC 10987 read set.

K-mer occurrence	$>= 1$	$>= 2$	$>= 3$	$>= 4$	$>= 5$	$>= 6$
B. anthracis	162,532,281	10,809,747	992,468	176,235	65,733	37,371
B. cereus	236,865,294	22,287,686	3,085,292	815,821	325,198	157,169

Table 3.3 shows the ranking of the two genome sequences we queried among all

54,118 organisms in NCBI RefSeq 70, as well as their corresponding p-values. Compared with regular MinHash (namely, when the solid k-mer threshold equals 1), utilizing solid k-mer powered MinHash was able to increase the genome identification accuracy for both Bacillus anthracis Ames and Bacillus cereus ATCC 10987. The p-value reached to the highest significance when the threshold was set to 3 in both cases, consistent with our prediction.

Table 3.3: Rankings and p-values of Bacillus anthracis Ames and Bacillus cereus ATCC 10987 generated with different solid k-mer thresholds.

Solid k-mer threshold		1	2	3	4	5	6
B. anthracis	rank	#417	#13	#1	#96	#255	#249
	p-value	8.79e-11	1.40e-16	5.07e-17	2.29e-4	6.93e-3	3.76e-3
B. cereus	rank	#464	#3	#1	#1	#1	#3
	p-value	4.01e-08	8.92e-31	7.22e-77	3.05e-72	1.64e-32	2.56e-09

Chapter 4

Discussion and future work

Based on the results in our experiment, we can conclude that compared with regular MinHash-based methods for genome distance estimation, feeding MinHash with solid k-mers as input can increase the accuracy of the distance calculation, enabling a boost of the downstream analysis in both precision and recall rates.

In addition, the solid k-mer powered MinHash method leaves out the less frequent k-mers and thus enables us to save computational time. For example, for the E. coli genome data analyzed here, there are around 200 million different k-mers in its sequenced reads, but only less than 5 million of them appear 3 or more times. Since the time complexity of MinHash is linear to the number of k-mers [Bro97], in theory we only need to spend around 1/40 computational time in the MinHash process using our solid k-mer powered MinHash.

One limitation of our method is that the calculation of the k-mer threshold requires us to know the approximate length of the reference genome beforehand, which is sometimes unknown or difficult to acquire, especially when we deal with read sets from unknown organisms. Fortunately, there are tools, such as preqc [Sim14], that can quickly estimate the length of a reference genome without knowing the actual reference

genome sequence or assembling the reads. Our next step is to incorporate preqc into our method to enable us to estimate genome distance without any extra information apart from the input genome sequences or reads. We will then wrap the whole method into an open source software tool and share it on public platforms such as Github.

Bibliography

[BDOP16] Pll Melsted Adam B. Mallonee Nicholas H. Bergman Sergey Koren Brian D. Ondov, Todd J. Treangen and Adam M. Phillippy. Mash: fast genome and metagenome distance estimation using minhash. *Genome Biology*, 17(1):132, 2016.

[BKS16] Sven Rahmann Bianca K. Stcker, Johannes Kster. Simlord: Simulation of long read data. *Bioinformatics*, 32(17):2704–2706, 2016.

[Bro97] Andrei Z. Broder. On the resemblance and containment of documents. *IEEE*, Compression and Complexity of Sequences 1997(Proceedings), 1997.

[CSCK13] Patrick Marks Aaron A Klammer James Drake Cheryl Heiner Alicia Clum Alex Copeland John Huddleston Evan E Eichler Stephen W Turner Chen-Shan Chin, David H Alexander and Jonas Korlach. Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, 10(6):563–569, 2013.

[CT12] Mark J Chaisson and Glenn Tesler. Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory. *BMC bioinformatics*, 13(1):238, 2012.

[Dav03] W. Mount David. *Bioinformatics: Sequence and Genome Analysis*. Gold Spring Harbor Laboratory press, New York, 2003.

[DL10] Owen Kaserb Daniel Lemirea. Recursive n-gram hashing is pairwise independent, at best. *Computer Speech and Language*, 24(4):698–710, 2010.

[DL15] Ruibang Luo Kunihiko Sadakane Tak-Wah Lam Dinghua Li, Chi-Man Liu. Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10):1674–1676, 2015.

[GR13] Rayan Chikhi Guillaume Rizk, Dominique Lavenier. Dsk: k-mer counting with very low memory usage. *Bioinformatics*, 29(5):652–653, 2013.

[JC79] Mark N. Wegman J.Lawrence Carter. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.

[KBP15] Chen-Shan Chin James P Drake-Jane M Landolin Konstantin Berlin, Sergey Koren and Adam M Phillippy. The primate neocortex in comparative perspective using magnetic resonance imaging. *Nature biotechnology*, 33(6):623–630, 2015.

[KEG] KEGG. Bacillus cereus atcc 10987. *http://www.genome.jp/kegg-bin/show-organism?org=bca*.

[Ken02] W. James Kent. Blatthe blast-like alignment tool. *Genome research*, 12(4):656–664, 2002.

[KPM12] Garth Brown Kim Pruitt, Tatiana Tatusova and Donna Maglott. Ncbi reference sequences (refseq): current status, new features and genome annotation policy. *Bioinformatics*, 40(D1):D130–D135, 2012.

[MDT94] Kurt Mehlhorn Friedhelm Meyer auf der Heide Hans Rohnert Martin Dietzfelbinger, Anna Karlin and Robert E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM Journal on Computing*, 23(4):738–761, 1994.

[MJA15] Karen Miga Hugh Olsen Benedict Paten Miten Jain, Ian Fiddes and Mark Akeson. Improved data analysis for the minion nanopore sequencer. *Nature methods*, 12(4):351–356, 2015.

[NCBa] NCBI. Escherichia coli str. k-12 substr. mg1655, complete genome. *https://www.ncbi.nlm.nih.gov/nuccore/NC-000913*.

[NCBb] NCBI. Refseq release 70 is now available with re-annotated bacterial genomes for uniformity across genomes and species. *https://www.ncbi.nlm.nih.gov/news/05-07-2015-refseq-release-70-reannotation/*.

[Nei87] Masatoshi Nei. *Molecular evolutionary genetics*. Columbia university press, 1987.

[NR74] Masatoshi Nei and Arun K. Roychoudhury. Genic variation within and between the three major races of man, caucasoids, negroids, and mongoloids. *American journal of human genetics*, 26(4):421, 1974.

[Pac16] PacBio. Sequel system data release: Arabidopsis dataset and genome assembly. *http://www.pacb.com/blog/sequel-system-data-release-arabidopsis-dataset-genome-assembly/*, 2016.

[PNA11] Wing-Kin Sung Pramila Nuwantha Ariyaratne. Pe-assembler: de novo assembler using short paired-end reads. *Bioinformatics*, 27(2):167–174, 2011.

[PWG98] G.J. Foschini P.W. Wolniansky and G.D. Golden. V-blast: An architecture for realizing very high data rates over the rich-scattering wireless channel. *Signals, Systems, and Electronics, 1998*, ISSSE 98. 1998 URSI International Symposium on.:295–300, 1998.

[RR13] Zeehasham Rasheed and Huzefa Rangwala. A map-reduce framework for clustering metagenomes. *Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW)*, 2013 IEEE 27th International(IEEE):549–558, 2013.

[SFAL90] Webb Miller Eugene W. Myers Stephen F. Altschul, Warren Gish and David J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[SFAL97] Alejandro A. Schffer Jinghui Zhang Zheng Zhang Webb Miller Stephen F. Altschul, Thomas L. Madden and David J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[Sim14] Jared T. Simpson. Exploring genome characteristics and sequence quality without a reference. *Bioinformatics*, 30(9):12281235, 2014.

[SKP13] Timothy PL Smith James L Bono Dayna M Harhay Scott D Mcvey Diana Radune Nicholas H Bergman Sergey Koren, Gregory P Harhay and Adam M Phillippy. Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome biology*, 14(9):R101, 2013.

[TDR03] Scott N. Peterson Timothy D. Read. The genome sequence of bacillus anthracis ames and comparison to closely related bacteria. *Nature*, 423(6935):81–86, 2003.

[WM07] Ross C. Hardison Minmei Hou James Taylor Brian Raney Richard Burhans Webb Miller, Kate Rosenbloom. 28-way vertebrate alignment and conservation track in the ucsc genome browser. *Genome research*, 17(12):1797–1808, 2007.

[XY11] Srinivas Aluru Xiao Yang, Jaroslaw Zola. Parallel metagenomic sequence clustering via sketching and maximal quasi-clique enumeration on map-reduce clouds. *IEEE*, Parallel and Distributed Processing Symposium (IPDPS)(2011 IEEE International), 2011.