Some Relationships Between Sequences and Their Kmer Profiles

By

Charles Liu

Senior Honors Thesis

Department of Computer Science

University of North Carolina at Chapel Hill

April 2020

Approved:

_____

Leonard McMillan, Thesis Advisor

Jan Prins, Reader

Donald Porter, Reader

Acknowledgements

First and foremost, the completion of this thesis would have been impossible without the kind support and nurturing of Dr. Leonard McMillan, my thesis advisor. Perhaps even a satisfactory completion of my degree would have been near impossible without his guidance. It was him who showed me the way to conduct computer science research, and I only regret that till this day I still wish I could have learned more from him.

I would like to extend my sincere thanks to the other members of the committee. Dr. Jan Prins was very kind to join the committee on very short notice and provided invaluable feedback both during the defense and after. Dr. Donald Porter, handling the paperwork of all students' honor defenses, endured my lackluster presentation and procrastination in the paperworks.

Special thanks to Mingming Lang, whose emotional support and company I could never forget, and whose tolerance of my inability in properly finishing this thesis I am still grateful of. Countless thanks to my parents, whom I am indebted to beyond words.

Finally I would like to thank everyone: Ziwei Chen, Anwica Kashfeen, Dr. Jack Snoeyink, Hang Su, etc., whose help during this my undergraduate career so kind that I can only fear that I do not deserve.

Abstract

This thesis explores two problems in bioinformatics via mathematical tools, both involving the choice of an integer parameter k. The first is de Bruijn graph assembly and the choice of the read length k: short read length produces more sequencing errors compared to long read lengths, which leads to the well known insight that longer read sizes are desirable. The second is kmer profiles and the choice of the substring length k: counts of number of occurrences of all size-k substrings of a given DNA string, called kmer profiles, has empirically been used as a computationally cheap way to compare DNA sequences; ideally, distances between kmer profiles should be approximations to edit distances between DNA sequences but this fails in some settings of k. With the goal of minimizing the error of assembly in the first problem and properly approximating the edit distance in the second, these two problems can be both phrased as choosing k s.t. calculating the kmer profile (expressed as the function pf_k) form some abstract isomorphism between fixed-length DNA strings and kmer profiles. This thesis shows that under sufficiently large k, pf_k forms an isomoprhism and admits an inverse function as de Bruijn assembly. Nonetheless even with k small, one direction of the isomorphism still stands in either settings of the problems, which stems from the fact that pf_k is k-Lipchitz, and lends credence to technique of k-mer profiles as a good representation of DNA strings for sequence comparison.

Contents

1 Introduction

DNA sequence data is indispensable for inferring biological relationships. One widely employed way to infer such relationships is by comparing two DNA sequences, which can shed light on evolutionary histories, structural similarities, etc.. It goes without saying that for the purpose of better inferred biologcial relationships, accurate DNA sequence data

and ways to compare DNA sequences are desirable. In recent decades DNA sequence data has proliferated, which has motivated computationally fast methods for DNA comparison.

The question of accuracy of the assembled DNA data arises from the fact that genomes cannot be read in one go. To obtain the original DNA sequence, modern next-generation sequencing methods read small pieces of the genome and then assemble them back together. One popular theoretical framework behind many modern genome assembly algorithms is the de Bruijn graph assembly (DBG). On a high level, DBG generates a graph based on overlapping reads and assembles the sequence from an Eulerian path on the graph. Accuracy comes into question when the graph used in DBG might have multiple Eulerian paths, which would imply that DBG might assemble a different sequence than that it originally read from.

Kmer profiles are reduced representations of DNA sequences that can be cheaply compared against each other. Sequence alignments and their resulting edit distance serve as the gold standard for sequence comparison. However, alignments take quadratic running time and, thus, do not scale well to large sequences. As an alternative method of sequence comparison, kmer profiles offer two benefits: converting a sequence into a kmer profile and then applying vector metrics on the profiles takes linear time; kmer profiles are integer valued vectors, and, thus many more algorithms, for example clustering, can be applied to them. The obvious concern is accuracy: as will be seen in later sections, kmer profiles are quite a simple representation of the original sequence, and it is not obvious if they are good representations for accurate comparisons.

This thesis aims to use mathematical tools to qualify the accuracy of DBG and kmer profiles, as well as identify some minor mathematical properties that kmer profiles offer for some applications. The essential questions this thesis attempts to answer are the following: *how accurate is kmer profiling as a proxy for direct sequence comparison*, *under what conditions are kmer profiles accurate*, and *how and what parameters can be adjusted to make kmer profiles more accurate?*

1.1 Motivation

Bioinformatics frequently deal with DNA strings, that is, strings on the alphabet $\{A, C, T, G\}$. For example AAAA, ACTGACTG, CCCCTTAG are all DNA strings, and there are a countably infinite number of other DNA strings. A **kmer** is a substring of length k of a particular DNA string. A length-n string will have $n - k + 1$ kmers and there exist 4^k distinct kmers.

Edit distance is a frequent metric for comparing sequences used in bioinformatics. Edit distance (as defined by Levenshtein) between string A and B is the smallest number of *edits* one needs to modify A to B (or by symmetry, equivalently number of edits from B to A), where edit can be one of: deleting a character from the string, inserting a character to the string, or substituting one character with another. For example inserting a C to the string AAAA can generate AACAA, and it takes only 1 edit which is minimal in comparison to any other series of modifications. Thus, the edit distance between AAAA and AACAA is 1. Edit distance between two length n strings takes $\Theta(n^2)$ time to compute.[7]

I now present two conceptual "games" that are explored in this thesis. They are not supposed to be actual interesting as games, but they roughly correspond to background material on de Bruijn graph assembly and kmer profiles. I hope the question of *how to play these two games well* motivates interest.

1.1.1 Genome Assembly

An adversary has a long DNA string of length n that you want to identify: and initially you are given only the length of the string, n. You want to determine this string, but you are only allowed to ask questions of the type: "how many times do all possible DNA combinations of length k DNA string occur as a substring of the string?", where $k << n$ is a positive integer that you choose. Clearly, if $k = n$, then the only size n substring of the secret string is the string itself. From the answer to this question, you need to win this game by recovering the original string.

As an example of this game, suppose that your adversary has a length 15 string (say ACTGACTGACTGAAT). Suppose that you choose $k = 4$ (or $k = 3, 5, 6$, etc. could all be

valid choices, but for the sake of example let us choose $k = 4$). You ask the question, and the adversary checks all possible length-4 DNA strings and the number of occurrences of all such strings in the secret string. In this case she will tell you that ACTG occurred 3 times; CTGA occurred 3 times; TGAC occurred 2 times; GACT occurred 2 times; TGAA occurred 1 time; GAAT occurred 1 time, or in more succinct notation that will be used throughout (the multiset notation):

$$\{\text{ACTG}^3, \text{CTGA}^3, \text{TGAC}^2, \text{GACT}^2, \text{TGAA}^1, \text{GAAT}^1\}$$

With this information, is it possible to recover the secret string that your adversary keeps? Can you do it in the general case for all strings? Are their special cases of secret strings that can be solved, and others not? And, what is the strategy for choosing k?

More succinctly stated, this problem asks if you can recover a DNA string by you choosing a k based on the length of the DNA string and all the counts of the possible size-k DNA strings occurring in the original string as a substring, and to make discussion more succinct later, the counts of all possible size-k DNA string occurring as a substring to some fixed DNA string is called the **kmer profile** of that DNA string. When the choice of k is clear or unimportant, it is simply called the **profile**. For example, the following are some DNA strings and their 3mer and 4mer profiles.

Sequence	3mer Profile
AAAA	$\{\text{AAA}^2\}$
CATGATCATGAT	$\{\text{CAT}^2, \text{ATG}^2, \text{TGA}^2, \text{GAT}^2, \text{ATC}^1, \text{TCA}^1\}$
ACTGACTGACTGAAT	$\{\text{ACT}^3, \text{CTG}^3, \text{TGA}^3, \text{GAC}^2, \text{GAA}^1, \text{AAT}^1\}$

Sequence	4mer Profile
AAAA	$\{\text{AAAA}^1\}$
CATGATCATGAT	$\{\text{CATG}^2, \text{ATGA}^2, \text{TGAT}^2, \text{GATC}^1, \text{ATCA}^1, \text{TCAT}^1\}$
ACTGACTGACTGAAT	$\{\text{ACTG}^3, \text{CTGA}^3, \text{TGAC}^2, \text{GACT}^2, \text{TGAA}^1, \text{GAAT}^1\}$

A common problem posed in genome assembly, is based on the counts of size-k substrings. While it is always possible to construct a string that fits these counts, in some cases the constructed string is guaranteed to match with the original string, but in other cases it may well be different.

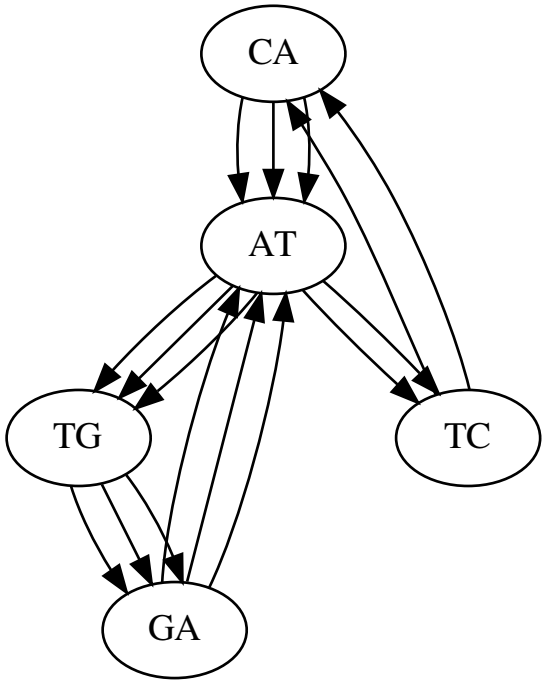
What we arrived at is that one kmer profile might correspond to several sequences, and some sequences in the game presented above admit several possible solutions. Intuitively, information is inevitably lost in the process of only looking at the counts of individual kmers (substrings of DNA strings of size k). Namely, with the help of the to-be-explained graph named **de Bruijn graph**, a multigraph, it is well known that DNA strings are bijective to paths on de Bruijn graphs. Profiles only correspond to the set of nodes visited by the paths, losing the information of how to connect such nodes. In short, if there are multiple ways to connect such paths, we are at a loss as to the original DNA string when we only look at its profile.

Consider the DNA string of CATGATCATGATCATGAT, its 3mer profile is

$$\{\text{CAT}^3, \text{ATG}^3, \text{TGA}^3, \text{GAT}^3, \text{ATC}^2, \text{TCA}^2\}$$

.

The above information, the profile, could be most effectively used in the de Bruijn graph below. The original sequence, without information loss, is mapped to a Eulerian path inside this graph. *But we do not have the path*, and there are multiple possible Eulerian paths, paths which traverse each edge only once, in this graph. There are multiple solutions to this graph. One of them starts from CA to AT, then taking the loop in the direction of TG three times, then do TC, CA, AT for two times, which corresponds to the sequence CATGATGATGATCATCAT, which is different from the original sequence of CATGATCATGATCATGAT.



1.1.2 Edit Distance Approximation via Kmer Profiles

Suppose that now the adversary keeps *two* secret strings of the same length n and she knows the edit distance between the two strings as d. You only know that the two strings are of same length n. Because the genome assembly game is unwinnable in the general case, this game asks you to perform a weaker task. You can still only have a single query with a positive integer $k << n$ that you first choose. You will ask, "how many times do every possible length k DNA string occur as a substring of the first secret string?" and then you ask the same question regarding the second secret string. With these information you are to think of a way to *approximate* the edit distance d within some reasonable error bounds, say that your approximation d' must satisfy $\frac{1}{r}d \leq d \leq rd$ for some $r > 1$.

The intuition is that two DNA strings, with low edit distance between them, should share a large number of length-k substrings. For example if the adversary has two DNA strings that have almost the exact same profile, it is probable, that the original DNA strings are almost the same, implying they have a small edit distance. The key lies in

the word *much probable*. Look at the following two pairs of DNA strings and their 3mer profiles.

Sequence	3mer Profile
AACCACATAAAC	$\{AAA^1, AAC^2, ACA^1, ACC^1, ATA^1, CAC^1, CAT^1, CCA^1, TAA^1\}$
AAACATAACCAC	$\{AAA^1, AAC^2, ACA^1, ACC^1, ATA^1, CAC^1, CAT^1, CCA^1, TAA^1\}$

Table 1: Sequences AACCACATAAAC and AAACATAACCAC (edit distance: 4) and their 3mer profiles

From the above example, it is evident that the DNA strings that look not entirely alike can have the same kmer profiles, yet in reality kmer profiles that look alike can frequently be quite close w.r.t their original sequences especially as k gets larger.

1.2 Kmer Profiles as Representations

The second game presented above was motivated by the general intuition of kmer profiles being an *efficient* representation for DNA strings. The "efficiency" here might warrant some close inspection, as it is in many ways not a space efficient representation, nor really in any way in computer science a canonical one. By the fact that each DNA character can be expressed in 2 bits, a length-n DNA string takes $2n$ bits to express. Kmer profiles can be spatially efficient and can also be spatially wasteful (k-mer counts increase as 4^k), but the more subtle point is that k-mer profiles have nice mathematical properties as representations for DNA strings.

Suppose there is a general efficient ($O(n)$, n length of DNA string) strategy to win the edit distance approximation game; or in more common terms, suppose that there is an $O(n)$ algorithm to approximate the edit distance of two DNA strings given their profiles. This would imply that edit distances can be approximated in linear time, which is not surprising, given that a known linear time $\sqrt{n}$-approximation algorithm for edit distance already exists, but still a nice fact because this might yield a better either in approximation factor or other aspects approximation algorithm. Yet the sad fact is that there is no such strategy. The reality is more subtle. Directly calculating distances between kmer profiles is a good enough strategy for the edit distance approximation game,

or in more common terms, distances between kmer profiles, with k a parameter to adjust, correspond *well enough*, but not in any sense a proper approximation, to the original edit distances between strings. This thesis quantifies this notion of "well enough" later.

The above discussion, about distances between kmer profiles corresponding "well enough" to distances between their original DNA strings might seem a bit appalling to some mathematicians or computer scientists, especially given that we *already have* good approximations ($\sqrt{n}$) for edit distance in linear time, and to bioinformaticians we have heuristics for alignment and edit distance. This qualified nicety regarding distances between kmer profiles is useful in two empirical aspects: the first is that kmer profiles can be treated as vectors; thus the common techniques such as clustering, dimensionality reduction can be directly applied to strings via their kmer profiles, while naively using edit distance might prove too slow for these tasks, but as k gets larger $4^k \ll n$. The second is a general theme of some corners of computational biology, where some methods that work well in real life do not excel in their theoretical groundings. A common theme in computational phylogenetics is to prove some weak theoretical nicety of algorithms and then demonstrate performance on simulated datasets, because weak theoretical nicety is often good enough on real-world datasets [8]. Kmer profiles work well in practice, which will be explained by some theoretical guarantees, some of which will be proven in this thesis, it is well motivated to use kmer profiles in some contexts because in many tasks strong theoretical guarantees are not needed.

Algebraic niceties also pop up in the structure of profiles, which lead to programmatic niceties a la *The Algebra of Programming*[2]. Kmer profiles are in some precise sense a pleasant representation for DNA strings for the purpose of edit distance approximation. This will be elaborated in the immediate next part and later in this thesis.

1.3 Mathematical Ideas

One purpose of this thesis is to use mathematical notation to identify some common and higher structures behind the two problems of *de Bruijn graph assembly* and *kmer profile distance approximation*, which were introduced in the beginning of the thesis in the form

a games. The parameter k plays a key role in the performance of both methods. As such, one of the fundamental problems is how to choose k. This section presents a quick sketch of some of the mathematical ideas.

The de Bruijn graph assembly problem has been introduced in this thesis by its analogous game. By notation that will be followed throughout, $pf_k(s)$ denotes the kmer profile of a DNA string s. The game is succinctly stated to be that we want to determine a k in which an left inverse, $assembly_k$, could be found. To abuse notation:

$$\exists k << n, assembly_k \circ pf_k = id.$$

Or as a general theme we always want to find a good $k << n$ to force nice properties out of kmer profiles, so in general I will drop the $\exists k << n$ qualifier.

$$\text{DNA Sequence} \overset{\sim}{\dashrightarrow} \text{Kmer Profile}$$

The above is the big picture of the main parts this thesis. I am trying to force bijections, or more generally, isomorphisms, between DNA sequences and kmer profiles. With a suitable choice of k, DNA sequences and kmer profiles are bijective, thus they also preserve distance. This picture becomes much more subtle when we consider smaller k. For smaller k, the bijection disappears but *one side* of the metric embedding still lives on: the other direction fails to be a function as it as the inverse ceases to exist. This foreshadows some of the themes of this thesis: large k always provides good theoretical guarantees, but due to either the curse of dimensionality or the nature of real world data, small k might be preferable, and small k still provide some theoretical guarantee in the precise sense that pf_k is k-Lipschitz.

1.4 Prior Works

The accuracy of DBG, and other graph based assembly methods, have been studied to much depth[5]. Results due to de Bruijn are still modified and applied to this problem, and various theoretical investigations have been pointed at the accuracy of DBG. Regrettably this work only adds interpretations to the results that these prior works have presented.

Kmer profiles are widely studied and used [9]. The algorithms for generating kmer profiles, using kmer profiles as a form of alignment free method (as in not requiring the often expensive calculation of alignments), and using kmer profiles as a baseline for other methods have relatively frequently popped up in literature [4]. There is no doubt that other properties and applications of kmer profiles have been investigated.

By extension, kmer based methods have also served as part of the base for ubiquitous tools in bioinformatics such as BLAST[1], which tries to find common words (or equivalently by default 3mers) between sequences before conducting local alignments, or BLAT[3], which relies on fast searching on the database built from non-overlapping kmers in the target from overlapping kmers in the input string.

2 Background

In this section I aim to give mathematical definitions to bioinformatics concepts. Many of these concepts have been introduced in the previous sections, but we can use more precise definition for formality.

2.1 Sequences and Profiles

DNA sequences are simply defined here mathematically. DBG often assumes that the reads generated from the sequence is idealized, which means that the set of reads is precisely the kmer profile. Thus the idealized set of reads used in DBG and kmer profiles are in fact the same construction, called "profiles" here.

For simplicity, we define DNA sequences as strings over a given alphabet.

Definition 1. *The* ***sequences*** *of length n, denoted as L_n, are strings over the alphabet $\{A, C, T, G\}$ of length n. Substrings of sequences of length k are called $k$mers and are the same as L_k.*

The following are examples of sequences:

AACACCACCCAC (1)

AAACCCA (2)

CATGACTAGACTG (3)

Sequences can be concatenated just like strings, the concatenation is simply denoted as $\circ$.

The following definition finally describes what kmer profiles are.

Definition 2. *A **profile** of size $k \leq n$ (called the kmer profile) of a string s of length n is a multiset where elements are L_k and the multiplicity of each element is the number of occurrences of that element as a substring in s. Notice that such multiset can always be represented as a vector of dimension 4^k, and this encoding will be assumed from here on.*

In other words, the profile of a string is simply a tally of all occurrences of its size-k substrings. The above described multiset is also the set of idealized reads (read from the string s) assumed in DBG that will be further assembled via the namesake graph. Now the process of calculating a profile is given a name below:

Definition 3. *The mapping from a string of length n to its profile of size k is denoted as $pf_k : L_n \to pf_k(L_n) \subset \mathbb{N}_0^{4^k}$ ($\mathbb{N}_0$ the set of non-negative integers). Observe that this function can be easily implemented in $\Theta(n)$ time.*

The image of pf is denoted as P.

Some facts about the profile can be immediately inferred. The sum of the components of a profile is simply the number of $k$mers in the original string, i.e. $n - k + 1$. The profile, viewed as a vector, becomes exponentially sparser as k increases; this is undesirable for large k due to the curse of dimensionality.

Recall that multisets (sometimes called "bags") are sets that can contain multiple same elements. For example, $\{A, A\}$ is not a valid set but a valid multiset. For succinctness, $\{A, A\}$ is denoted as $\{A^2\}$: the multiplicities of each element is written as a superscript.

Example 1. *The 3mer profile of the sequence* $s = ACTGACTGACTG$ *is the following multiset:*

$$\{ACT^3, CTG^3, TGA^2, GAC^2\}$$

One canonical way to compare two sequences is through Levenshtein distance, also referred to edit distance from here on. This is the same distance commonly referred to in algorithm textbooks as an example algorithm for dynamic programming. The exact definition is given below:

Definition 4. *The* ***edit distance*** *(denoted* lev*) between two strings* s_1 *and* s_2 *is the minimum number of operations from* s_1 *to* s_2*. The allowed operations, for our purposes, are:*

- *insert a character into a string*
- *delete a character from a string*
- *replace a character of a string by another character*

Calculating the edit distance between two strings takes $\Theta(n^2)$ time. As an example, the edit distance between the act and cat sequence is 3 which is visualized below (mismatches that are to be replaced shown in red, insertions shown in blue):

A C T G A C T A G A C T G

C A T G A C T A G A C T G

Edit distance can become prohibitively expensive in large datasets.

2.2 De Bruijn Assembly

Given some alphabet and a positive integer k, can one construct a string that contains every possible length-k string as a substring? For example, on the binary alphabet $\{0, 1\}$

and $k = 1$, the string 01 contains every possible length-1 binary string as a substring, so does 10, but how about on any general alphabet and $k > 1$?

Motivated by the above problem, de Bruijn devised a special type of graph, the de Bruijn graph, which will be more rigorously defined below.

For simplicity, all *graphs* from here on refer to *connected directed multigraphs*. In a de Bruijn graph, all nodes belong to L_k, and incidence is defined by overlap. With this construction, finding a Eulerian path, which is of polynomial time, can be successfully applied to assemble a sequence. This entire process is made precise in this section.

First recall the notion of Eulerian paths.

Definition 5. *A **Eulerian path** of a graph is a path that traverses each edge of the graph exactly once. A graph is semi-Eulerian if it has an Eulerian path. If the last edge of the path is adjacent to the first edge of this path, this path is also called an Eulerian cycle. A graph is Eulerian if it has an Eulerian cycle.*

For the simplicity of notation below, I define an auxiliary operation of strings denoted as $\diamond$, which would be of help only in this section.

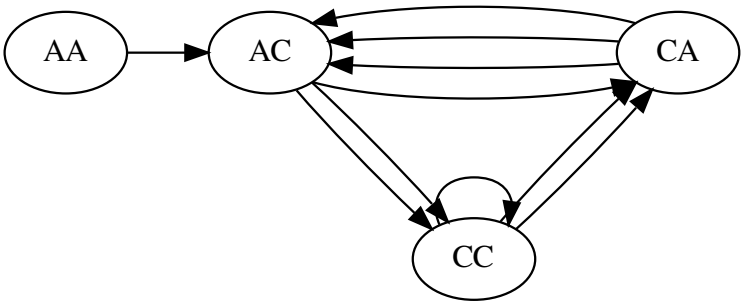
Definition 6. *$a \diamond b$ is defined for strings if the two strings are of same length k with the length $k-1$ suffix of a equal to the $k-1$ prefix of b. Then $a \diamond b$ is equal to the first character of a appended to the start of b. For example "cat" $\diamond$ "ats" = "cats".*

Then the precise construction of the de Bruijn graph is given below. In the definition below k represents what is called the read length.

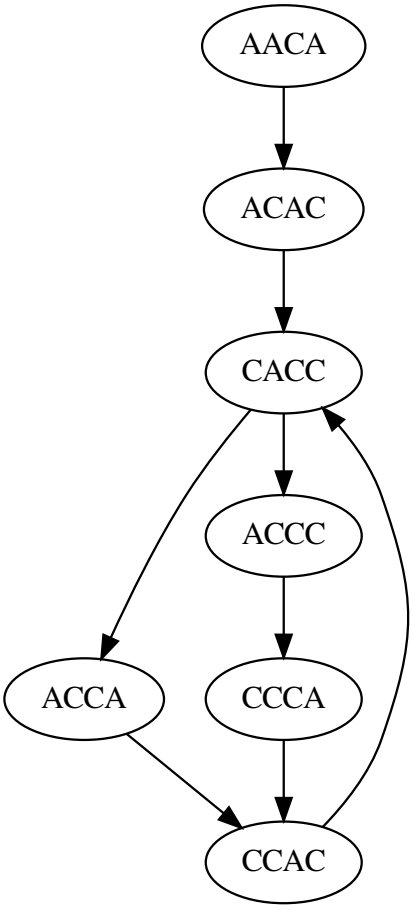
Definition 7. *The **de Bruijn graph** of order k of a sequence of length n is a graph that has all the $(k-1)$mers as the nodes. Node x has an edge (labeled $x \diamond y$) to y if $x \diamond y$ is defined and is a substring of the sequence.*

The above definition might seem not intuitive at first, but by observation, as the edges (labeled as $k$mers) correspond to the original reads, a length-2 path corresponds to a $k+1$mer; length-3 paths correspond to $k+2$mers in the original sequence, etc.. Therefore the original sequence must correspond to some path in the graph, and it can be proven that it must be an Eulerian path.

Example 2. *The de Bruijn graph of the sequence "AACACCACCCAC" of order 3 is as illustrated below (edge labels omitted).*



Example 3. *The de Bruijn graph of the sequence "AACACCACCCAC" (the same sequence as above) of order 5 is as illustrated below.*



Observe that the de Bruijn graphs of the same sequence, as k increases, intuitively becomes more *simple* in the sense that they become more linear.

Now with the graph defined, the final assembly algorithm is simply finding the Eulerian path and then concatenating the edges via the custom operation defined above.

Definition 8. *The assembly function first finds an Eulerian path of the graph (via preexisting algorithms) and then takes all the edges* $(e_1, e_2, e_3, e_4, \ldots, e_i)$ *and "join" the edges via* $\diamond$ *like this:* $e_1 \diamond e_2 \diamond e_3 \diamond \ldots \diamond e_i$.

Then the DBG algorithm simply takes as input the sequence, builds a de Bruijn graph of that sequence, then runs the assembly function defined above on it.

The problem of accuracy can be intuitively described here. The original sequence correspond to an Eulerian path on the de Bruijn graph, but this path is not given to the final assembly step. This information is effectively lost, and if there exist other Eulerian paths in the graph then it is not guaranteed to recover the original path.

De Bruijn sequences provide examples for sequences where every dimension of the profile has order 1 for some k. There also exists closed form formulas for calculating the number of de Bruijn sequences of some order.

Definition 9. *A* ***de Bruijn sequence*** *(on the DNA alphabet) of order* k *contains each kmer exactly once.*

As a side note, by prior work, there exists precisely $\frac{(4!)^{4^{k-1}}}{4^k}$ de Bruijn sequences of order k. By definition of de Bruijn sequences, all these sequences have the same magnitude 1 profile. This immediately shows that different sequences can have the same profile.

Example 4. *"AACATAGCCTCGTTGGA" is a de Bruijn sequence of order 2. All di-mers appear exactly once in the sequence. To list them:*

$$\{AA^1, AC^1, AG^1, AT^1, CA^1, CC^1, CG^1, CT^1, GA^1, GC^1, GG^1, GT^1, TA^1, TC^1, TG^1, TT^1\}$$

2.3 Metric Spaces

DNA sequences in bioinformatics are usually richer in structure than simply considered as strings on the DNA alphabet. As sequences are frequently compared via some "nice" comparison functions, we can say that DNA sequences, paired with some method of comparison that is nice in the sense that is a metric, form a metric space. Kmer profiles are also frequently compared against each other with standard metrics, and thus it also helps to consider kmer profiles as metric spaces.

Definition 10. *A* ***metric space*** *is a set S equipped with a function $d : S \times S \to [0, \infty)$ satisfying the following axioms:*

- *Triangle inequality:* $d(x, y) + d(y, z) \geq d(x, z)$
- *Point inequality:* $0 \geq d(x, x)$(*so* $0 = d(x, x)$)
- *Separation:* $x = y$ *if* $d(x, y) = 0$(*so* $x = y$ *iff* $d(x, y) = 0$)
- *Symmetry:* $d(x, y) = d(y, x)$

A function that maps between metric spaces is called a **metric embedding**. A metric space S equipped with the function d is denoted as (S, d).

Definition 11. *Let f be an embedding from the metric space (X, ρ) into another metric space (Y, μ). We define*

$$\text{expansion}(f) = \max_{x,y \in X} \frac{\mu(f(x), f(y))}{\rho(x, y)}$$
$$\text{contraction}(f) = \max_{x,y \in X} \frac{\rho(x, y)}{\mu(f(x), f(y))}$$

The distortion of an embedding f, distortion(f), is defined as the product of expansion(f) and contraction(f). An embedding f with distortion$(f) = 1$ is called isometric.

The above definitions, like all definitions of this section, is standard and can be found in many mathematics textbooks. Recall the following two metrics that are frequently used.

Example 5. *The L_1 metric (Manhattan distance) together with any subset of $\mathbb{R}$ is a metric space.*

Example 6. *The Levenshtein metric together with L is a metric space.*

For a mathematician, one obvious question to ask regarding this view of sequences and profiles as metric spaces is the property of the profile function pf. The mathematical inquiry simply asks if pf preserves the metric space structure when it takes sequences and converts them to profiles. A standard definition in mathematics is that, intuitively, a function between two metric spaces respect the structures if it does not expand the distances between the objects, in the precise sense below:

The following definition goes rather on a tangent for another niche property of kmer profiles. It captures two essential properties of string: there exists an empty string, and string concatenation is associative.

2.4 Algebraic Structures

A set "merely" contain elements. To say some mathematical object is a set merely says that object can contain elements. To say a set forms a metric space under some function reveals more information. For example, the set of DNA strings contain many strings limited to the alphabet ACTG, but more interestingly we want to calculate Levenstein distance between such strings, hence motivating the language of metric space as defined in the previous subsection.

Sometimes we want to consider different operations in some sets. For example two DNA strings concatenated still produce a DNA string. This motivates the definition of monoids. Other operations on other sets motivate other structures, the structure of metric space has already been defined previously, and only the notiong of semiring will be additional defined.

This subsection will define two notions of homomorphisms between structures. Besides the intuitive explanation as structure preserving maps, homomorphisms can be viewed as not only mapping elements between sets, but also mapping operations among elements from one set to another set.

Definition 12. *A **monoid** is a set M equipped with a binary operation $(\cdot) : M \times M \to M$ and special element $1 \in M$ satisfying the following axioms for all $x, y, z \in M$:*

- $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- $1 \cdot x = x = x \cdot 1$

A monoid $(M, \circ)$ *with* $(\circ)$ *commutative, i.e.* $a \circ b = b \circ a$*, is called an* ***abelian monoid****.*

Note that as an example, any vector space under the operation $(+)$ forms a monoid. The unit element is 0 and addition is associative.

Example 7. *Any vector space under the operation* $(+)$ *forms a monoid with* 0 *as the unit element.*

Example 8. *Lists (for example, linked lists) under the operation list concatentation forms a monoid with the empty list as the unit element.*

The concept described below simply provides machinery for a niche property discussed later.

Definition 13. *A function* $\phi : (M, \cdot) \to (N, \diamond)$ *is a* ***monoid homomorphism*** *if the following property holds for all* $x, y \in M$*:*

$$\phi(x) \diamond \phi(y) = \phi(x \cdot y)$$

In other words, reusing the notation from the definition above, the function ϕ preserves the monoid structure when mapping between M and N. The function ϕ can also be viewed as not only acting as a function mapping elements from M to N but also mapping the operator $\cdot$ to $\diamond$.

Definition 14. *A* ***semiring*** *is a set* R *with binary operations of addition and multiplication, such that* $(R, \cdot)$ *is a monoid under multiplication;* $(R, +)$ *is an abelian monoid under addition; multiplication distributes over addition, i.e. the distributivity laws hold:*

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$
$$(y + z) \cdot x = (y \cdot x) + (z \cdot x)$$

and also the absorption/annihilation laws, which are their nullary version:

$$0 \cdot x = 0 = x \cdot 0$$

Semirings, with the canonical example of the set of non-negative numbers, can be said to be in some sense two monoids in one. Some archetypical abstractions in computer science form semirings.

Example 9. *The set* $\mathbb{R}$ *along with addition as* $\boldsymbol{min}$ *and multiplication as* $(+)$ *forms a semiring denoted as* $(\mathbb{R}, \boldsymbol{min}, +)$*. This is called the tropical semiring and forms the basis for a field of geometry called tropical geometry but also has applications in optimization.*

Definition 15. *If* S *and* T *are semirings, then a semiring homomorphism from* S *to* T *is a map* $f : S \to T$ *which preserves addition and multiplication,* $f(x + y) = f(x) + f(y)$ *and* $f(xy) = f(x)f(y)$*, as well as the neutral elements,* $f(0) = 0$

Sometimes we only desire two things to be approximately equal under some measure of distance. If $x, y \in (S, d)$ where S some metric space, x and y are approximately equal or almost equal if for some "small" ϵ, $d(x, y) < \epsilon$. This generalizes to the notion of "almost a homomorphism" in this thesis, where for function f to be a homomorphism, usually f needs to satisfy conditions in the form $f(x \circ y) = f(x) \diamond f(y)$ for binary operators $(\circ)$ and $(\diamond)$. f in this context is almost a homomorphism between two structures additionally endowed with metrics if for some small ϵ, $d(f(x \circ y), f(x) \diamond f(y)) < \epsilon$.

3 From Profiles to Sequences

Now we begin the exploration of the properties of kmer profiles. We begin with some preliminary deductions.

3.1 First Observations

Some insights can be obtained from first observations of the objects at play.

Fact 1. *The kmer profile metric space* (P, δ_k) *is of topological dimension 0 under any selection of the metric.*

Proof. It suffices to show that the metric space has every subset as an open set. Consider an open ball of size $\frac{1}{k}$, it is obvious that such balls only contain the point itself. By closure of open sets under union, every subset is open. □

The above fact simply shows that the kmer profile space is a discrete space, and no interesting topology or geometry could be easily applied.

Fact 2. *The maximum magnitude of the kmer profile of a length* n *string is monotone non-increasing as* $k \to n$ *and approaches to* 1.

Proof. It cannot ever increase. Suppose it can; it means that some $k+1$mer appears more than a $k$mer, but this would imply that any substring of length k of the $k+1$mer would appear at least as much time as the $k+1$mer. Contradiction.

Consider $k = n$. The maximum magnitude is 1. This would show that at some point the max magnitude is 1. □

The above fact carries more significance, as the maximum magnitude is connected to in-degrees and out-degrees of the generated de Bruijn graph.

3.2 Ambiguity

Perhaps not intuitively, we now begin with the question of the existence of the inverse function of pf. This is made more intuitive when we note that the process of taking profiles and converting them back to sequences is an idealized version of the well studied problem of genome assembly. This connection is elaborated further below.

Given a set of reads, how to assemble the genome back is the central problem underlying contemporary next-generation genome assembly methods. The immediate problem that hinders the mathematical beauty is that theoretically different sequences might generate the same set of reads, in other words, the kmer profile function is not one to one. Then the question becomes why and when does such kmer profile function become not one to one, and how much of a problem is this.

Example 10. *"GGTGCGATTCTACCAA" and "AACATAGCCTCGTTGG" share the same dimer profile; the profile is*

$$\{AA^1, AC^1, AT^1, CA^1, CC^1, CG^1, CT^1, GA^1, GC^1, GG^1, GT^1, TA^1, TC^1, TG^1, TT^1\}$$

In fact, the two sequences contain every possible dimer exactly once.

Sequences that are different but share the same profile, for simplicity, are called "ambiguous", because they cannot be told apart in the profile and by extension in the de Bruijn graph yet their original sequences are different.

We now raise the question with an obvious answer as demonstrated by the previously mentioned example: hypothetically, does there exist a way to convert back from kmer profiles to sequences while still keep close things close? The answer is in general no, because ambiguous sequences have kmer profiles that are as close as possible (distance 0), yet ambiguous sequences can have quite large edit distances.

Proposition 1. *There does not in general exist a metric embedding from* (P, d_P) *to* (L, d_L) *with expansion* $< \infty$ *any selection of the metrics* d_P *and* d_L*.*

Proof. It suffices to show that in general there exists semi-Eulerian de Bruijn graphs that contain two unique Eulerian paths. The most simple case is Eulerian cycle. For n sufficiently large, one can always find a cycle graph that is a valid de Bruijn graph. □

Then when can de Bruijn graphs only have unique sequences behind them, as in existing no ambiguous sequences? The following section explores this topic.

3.3 The Number of Ambiguous Paths

The number of Eulerian paths in a de Bruijn graph has been characterized by a result due to Kingsford et. al, which is written below:

Theorem 1. *Let* $A = (a_{uv})$ *be the adjacency matrix for an n-vertex de Bruijn graph G, with both* $a_{uv} > 1$ *and self-loops allowed. If* $d^+(v) = d^-(v)$ *for all* v*, then choose a vertex* t *arbitrarily, otherwise pick the unique* t *such that* $d^+(t) = d^-(t) - 1$*. Finally, let*

$r_u = d^+(u) + 1$ *if* $u = t$ *or* $r_u = d^+(u)$ *otherwise. Then the number of sequences consistent with* G *that can be spelled ending with node* t *is given by*

$$\mathcal{W}(G, t) = (\det L) \left\{ \prod_{u \in V} (r_u - 1)! \right\} \left\{ \prod_{(u,v) \in E} a_{uu}! \right\}^{-1}$$

where L *is the* $n \times n$ *matrix with* $r_u - a_{uu}$ *along the diagonal and* $-a_{uv}$ *in entry* (u, v)

The above does not immediately lead to many insights, and thus I present a more restrictive form of the above theorem that might lead to more intuition:

Theorem 2. *The number of Eulerian paths in a semi-Eulerian but not Eulerian graph that contains no parallel edges is*

$$t_G \prod_{v \in V} (deg(v) - 1)!$$

where t_G *is the number of unique spanning trees of the graph and* $deg(v)$ *is the maximum of the in-degree and out-degree of* v*.*

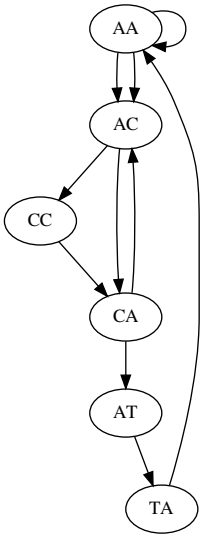
Proof. Any semi-Eulerian graph contains two inbalanced nodes. Modify the graph by connecting the two inbalanced nodes. Now the graph is balanced and Eulerian. In a result due to de Bruijn, we have the BEST theorem, written below:

$$t_G \prod_{v \in V} (deg(v) - 1)!$$

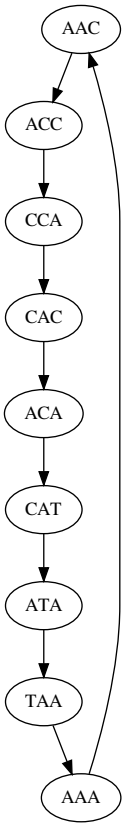
where t_G is the number of unique spanning trees to the graph rooted from given node (it can be shown that this is the same in an Eulerian graph), which by the assumptions is bijective to the number of unique spanning trees in the unmodified version of the graph, and $deg(v)$ is specifically the out-degree, which is equal to the in-degree in a balanced (Eulerian) graph, which implies that even after the deletion of the edge connecting the two inblanaced nodes, the max of the in and out degree will recover the original degree. Observe that the modied graph must have the same number of Eulerian cycles as the original graph have Eulerian paths. □

Recall that Eulerian paths in the de Bruijn graph represents admissible sequences that can be assembled from the reads. Thus if the de Bruijn graph (that fits the assumptions of the above theorem) contains a vertex with a degree of 3, the original sequence must be ambiguous. More deeply, the above theorem suggests the unfortunate consequence of choosing a small read size, or a small k in kmer profiling. For example, when nodes are 2mers, and say we have a node that is "AC". It only takes three of "AAC", "CAC", "GAC', "TAC" to appear in the sequence for this sequence to have a degree 3 vertex. This also leads to the observation that increasing k inevitably decreases degrees of vertices, as it is even hard to ask for a fixed kmer itself to appear twice in the sequence. $k+1$mers having that kmer as a suffix or prefix can only be rarer.

The fact that increasing k decreases the degrees of the nodes in the graph also destroys possible unique spanning trees in the process as the graph becomes increasingly linear. An example of this process can be viewed below, first look at the de Bruijn graph of a sequence of order 3:



and the same de Bruijn graph of this sequence of order 4:



There are multiple Eulerian cycles in the order-3 de Bruijn graph but only one Eulerian path (by coincidence also an Eulerian cycle) in the order-4 graph. This is a general tendency of increasing k: increasing k destroys loops and decreases the degrees of the nodes.

In conclusion of this section, pf is bijective when k increases when the de Bruijn graph of the sequence "unrolls": the loops disappears. Thus the familiar conclusion, of kmer profiles being an accurate representation of DNA sequences is reached.

4 From Sequences to Profiles

Now we look at the opposite direction: the forward direction of pf where we generate profiles from sequences. This has two real world applications: the first is for generating short reads for DBG; the second is for calculating kmer profiles and apply metric on the profiles. With all the machinary defined in previous sections, the expansion and distortion of the kmer profile function can now be introduced.

4.1 The Expansion and Distortion of The Profile Function

Proposition 2. $\text{expansion}(pf_k) \leq 2k$*: the expansion of kmer profile as a metric embedding is no more than* $2k$*.*

Proof. Given two strings $s_1, s_2 \in L$, each edit from s_1 to s_2 only induce an at-most magnitude-$2k$ change to the profile vector.

The window passes through the changed position exactly k times. Each time it passes through, the dimension associated with each original window decrements by one, and the dimension associated with the new window increments by one.

For a given edit distance d, the maximum profile distance is $2kd$. □

To repeat, the above proposition shows that kmer profiles will never expand distance, no matter the choice of k, over the factor of $2k$. In fact, the above proof shows more: each edit will only induce up to $2k$ distance in the original L_1 metric. More informally, edit distance upper bounds kmer profile distance by a factor of $2k$. This would by contrapositive imply that those far in kmer profile distance cannot be close in edit distance, which can help saving running time when filtering for sequences close in edit distance.

Proposition 3. *The contraction of the kmer profile function* pf *can be piecewise expressed below:*

$$\text{contraction}(pf_k) \begin{cases} = \infty & k << n \\ \leq 1 & \textit{otherwise} \end{cases}$$

Proof. It is clear that the first case is true by the pf no longer being one-to-one. The second case is equivalent to saying that edit distance will never contract beyond a certain k. This is true definitely at $k = n$, thus as $k \to n$, at some point it will reach this point where the distance never contracts. As a remark, this is also a fairly common case but regretablly this cannot be made more precise. □

By the result of the previous section, when pf is not one-to-one. The contraction is immediately revealed to be undefined (written as ∞) as two points with non-zero distance becomes zero-distanced.

Theorem 3. *The distortion of the kmer profile function pf can be piecewise expressed below:*

$$\text{distortion}(pf_k) \begin{cases} = \infty & k << n \\ \leq 2k & otherwise \end{cases}$$

where $k << n$ means that k is small enough s.t. pf_k is not a bijection.

Proof. By $\text{distortion}(pf) = \text{expansion}(pf) \cdot \text{distortion}(pf)$ and the two above propositions, this piecewise relationship can be quickly deduced. □

This piecewise function shows that pf, in order to become nice (having low distortion) as a metric embedding, must not have an overly small k to the extent that pf_k is not a bijection. Beyond that, the distortion is capped at $2k$, suggesting that k should be neither too small nor too big.

4.2 The Profile Function as Monoid Homomorphism

This is a niche property that is merely nice to have. As kmer profiles are vectors, one can ask what operations on vectors make sense. The answer is that only addition so far makes sense, because a bit informally, addition on kmer profiles correspond to concatenating sequences, which is made precise below:

Proposition 4. *pf is almost a monoid homomorphism, in the precise sense below:*

$$\delta_k(pf_k(x \circ y), pf_k(x) + pf_k(y)) \leq 2k$$

Proof. Since kmer profile count kmers, observe that only the k kmers in the middle of the concatenated string has not been counted, which will induce at most k magnitude change in the Manhattan distance, thus at most 1 magnitude change in kmer profile distance. □

More intuitively, it does not matter that much whether I first concatenate two sequences, then calculate the kmer profile, or first calculate the kmer profiles, then perform addition on the profiles. This property lends minor confidence to kmer profiles as representations to sequences, as they "concatenate" in a sense.

In fact, as a generalization, it can be posed, that some algebraic completion of DNA strings, if we squint hard enough, form a semiring. That is to say that some modification $(L, \circ, \odot)$, where $(\odot)$ is a modified version of sequence alignment is a semiring. If we consider profiles to be sets and consider the set of profiles of sequences with length *no longer than* some fixed length n that is closed under intersection, we obtain a variation of the semiring of sets. This notion can be made more precise below in the table, but regrettably this will be left not explored further.

Sequence Space	**Profile Space**
DNA Sequence (String)	Profile (Vector)
Concatenation $(\circ)$	$\cup$
Alignment $(\odot)$	$\cap$
Edit Distance (lev)	L_1
Metric Space, "Near Semiring"	Metric Space, Semiring

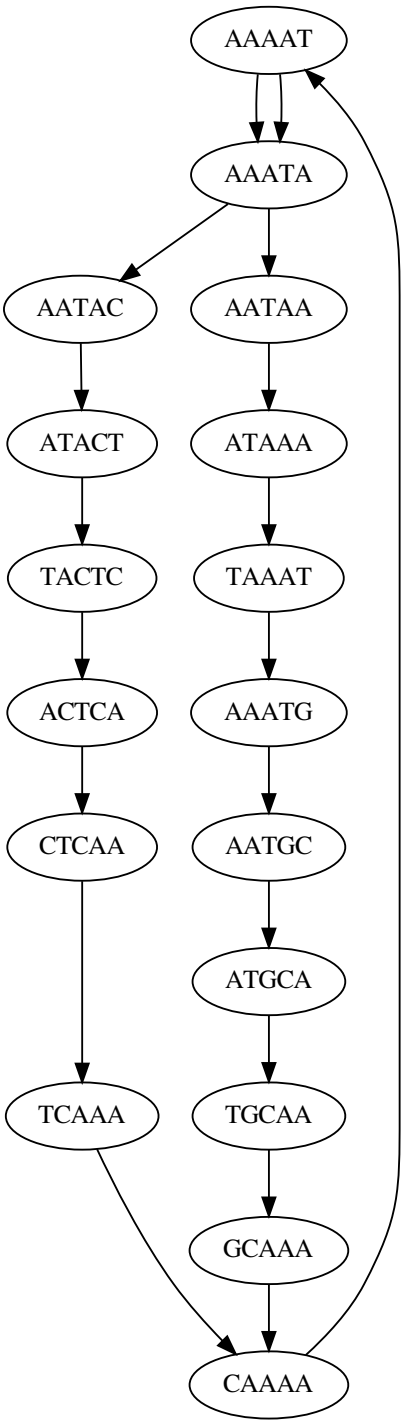
5 Discussion

With the results presented, this section focuses on several questions one might ask after learning about the unfortunate fact of choosing a k, where k too small will lead to ambiguities and the mathematical nice properties break down.

5.1 How Much Error can be Generated

Some sequences map to the same kmer profile. One immediate question is how much can these sequences themselves differ. Pessimistically they can have distances among themselves resembling distances among random sequences, as essentially choosing a different Eulerian path destroys the original order, thus informally forcing the edit distance to find creative ways to connect the two strings.

As an example, consider the following structure of the de Bruijn graph:



The above graph admits two possible ways to reaach the CAAAA node, from the left path or the middle path. As the left path and the middle path can have significant differences, as demonstrated by this example, their edit distances can differ greatly.

5.2 Leaving Kmer Profiles as They are

Since kmer profiles keep close sequences close, it is tempting to simply not recover the original sequence from kmer profiles, but to simply use them as an approximate form of the original seqeunce that can be quickly compared against. Addition of kmer profiles, as shown in this thesis, even approximate a crude form of string concatenation.

When performing kmer profile based sequence comparisons, it makes some vague sense to use the read size for the selection of k, as if the assembly error purely originates from the fact that multiple Eulerian paths exist on the graph, the kmer profiles of the ground truth genome would still agree with the wrongly assembled sequence.

It should also be of note that the ambiguity of kmer profiles, in the asymptotic sense, does not matter. One can use constant time to filter out the ambiguous sequences. Consider the following proposition:

Proposition 5. *For fixed k, the number of ambiguous sequences for a given sequence is expected to a constant.*

Proof. There are 4^k vertices, the number of paths is upper bounded by 4^{4^k}. The number of sequences is 4^k and thus the ratio is bounded by a constant 4^{4^k-k}, still a large number but a constant. □

6 Applications

6.1 The Selection of k

The selection of k can be said to be a problem of tradeoff when actually applying kmer profile based methods to real world data. As first noted, the vector becomes exponentially more sparse as k approaches infinity. Due to the curse of dimensionality, this can lead to problems in further analysis (clustering, visualizing, etc.) of the data.

On the other hand, increasing k will necessarily lead to the decrease of degrees of the vertices of the graph. By the modified BEST theorem, this will necessarily lead to only 1 Eulerian path in the de Bruijn graph. More intuitively, there will be only one way to piece

together these long sequences of length k. In short, or in obvious hindsight, k should be neither too small or too large.

As a folklore result, roughly k should be proportional to log_4 of the length of the string.

6.2 Quickly filter sequences with edit distance within d

Although kmer profiling is much more suited to filtering out sequences that are far away (in edit distance) from some fixed reference sequence, the need to select a subset of sequences that are within d in edit distance to some reference genome might arise occasionally.

Since kmer profiles will never have false negatives, the algorithm can simply proceed by first taking all sequences within d distance in their kmer profile representation, then calculating edit distance based on the original sequences. I hereby give a more formal description of the problem and this method:

Assume that n (for simplicity) length-n strings are given, and a length-n sequence, called the reference sequence is also given. The task is to find all sequences in the list that are of d edit distance within the reference sequence. Assume that $\Theta(\frac{1}{m})$ sequences of the list actually fit this description, which is reasonable for small d. The algorithm proceeds as follows:

- First calculate the n-mer profile for each of the sequence (including the reference). (Linear time)
- Using the modified metric δ_d, take all the sequence (candidates) that has within 1 distance w.r.t the reference sequence's profile. (Linear time)
- As the number of candidates is of the order $\Theta(\frac{1}{n})$, linearly filter all the candidates within edit distance d. This step would take $\Theta(\frac{1}{n})n^2 = \Theta(n)$ (linear) time.

Thus this is a expected linear time algorithm (with the reasonable assumption of the distribution of the sequence), as opposed to a expected quadratic time algorithm.

7 Conclusion

The theme, of the choice of k, appears ubiquitously in the problems of genome assembly and using kmer profiles as reduced representations of DNA sequences for fast comparisons. To recall the discussion on k, a large enough k leads to bijections, and more generally, isomorphisms, between sequences and their profiles. This is due to results from genome assembly: large enough reads, as reminded by in this thesis, lead to unambiguous de Bruijn graphs. With the profiles understood as reads, this is the obvious result of saying that choosing a large enough read size k lead to better assembled sequences. This settles the discussion for the choice of k in genome assembly, but not the entire picture.

The second discussion made is that of kmer profiles being vector representations of DNA sequences that carry significance in the sense that distances between such kmer profiles correspond well enough to distances between their original sequences. This is a much more subtle point than genome assembly, although as an implication of 1-1 correspondences between the profiles and their sequences for large enough k, large k inevitably leads to strictly bounded correspondences between kmer profiles. It is the *curse of dimensionality* that deals the most damage here. It is *not true* for kmer profiles to have an as large a k as possible. Although theoretically it is theoretically nice its properties will get destroyed in any reasonable sense as the dimension increases exponentially with k.

The finer point associated with the above discussion is that kmer profiles *always* correspond greatly with the original edit distances in the precise way that kmer profiles are bounded, by a constant factor from their original edit distances. This is regardless of k, and provides a consistent theoretical guarantee to the usage of kmer profiles as ways to compare sequences without performing alignments, implying that kmer profiles can sometimes be used as representations purely for their contracting property w.r.t DNA strings.

Some finer points are also demonstrated with regards to kmer profiles that are mostly unrelated to the choice of k. Intuitively, the operation of summing up (direct sum in multisets, simply addition on vectors) kmer profiles is equivalent to DNA string concatenation. The operation of intersection on the kmer profiles is roughly equivalent to

aligning sequences in the original DNA strings. This can be made *precise*, in forcing a variant of kmer profiles that have pf_k as a monoid homomorphism.

References

[1] S. F. Altschul et al. "Basic local alignment search tool". eng. In: *Journal of Molecular Biology* 215.3 (Oct. 1990), pp. 403–410. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2.

[2] Richard Bird and Oege De Moor. "The algebra of programming". In: *NATO ASI DPD*. 1996, pp. 167–203.

[3] *BLAT—The BLAST-Like Alignment Tool*. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC187518/ (visited on 03/23/2020).

[4] *fast, lock-free approach for efficient parallel counting of occurrences of k-mers — Bioinformatics — Oxford Academic*. URL: https://academic.oup.com/bioinformatics/article/27/6/764/234905 (visited on 03/23/2020).

[5] Carl Kingsford, Michael C. Schatz, and Mihai Pop. "Assembly complexity of prokaryotic genomes using short reads". In: *BMC Bioinformatics* 11.1 (Jan. 2010), p. 21. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-21. URL: https://doi.org/10.1186/1471-2105-11-21.

[6] Moriarty. *The Dynamics of An Asteroid*. 1881.

[7] Saul B. Needleman and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of Molecular Biology* 48.3 (1970), pp. 443–453. ISSN: 0022-2836. DOI: https://doi.org/10.1016/0022-2836(70)90057-4. URL: http://www.sciencedirect.com/science/article/pii/0022283670900574.

[8] Tandy Warnow. *Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation*. en. ISBN: 9781107184718 9781316882313 Library Catalog: www.cambridge.org Publisher: Cambridge University Press. Oct. 2017. DOI:

10.1017/9781316882313. URL: /core/books/computational-phylogenetics/A3A55F3748D3088FFB6EF25F5C3F15F7 (visited on 04/09/2020).

[9] Andrzej Zielezinski et al. "Alignment-free sequence comparison: benefits, applications, and tools". In: *Genome Biology* 18.1 (Oct. 2017), p. 186. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1319-7. URL: https://doi.org/10.1186/s13059-017-1319-7 (visited on 03/23/2020).