NanZam: A System for Rapid Karyotyping

James J. Bannon

Presented in Partial Fulfillment of
M.S. in Computer Science

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
New York
September 7, 2018

NanZam: A System for Rapid Karyotyping

James J. Bannon

Abstract

The vast majority of organisms store their genetic information in chromosomal DNA. The number, type, and arrangement of genes on a chromosome is called the *karyotype*. Obtaining karyotype information is useful in both research and clinical settings. Accurately obtaining karyotype information, however, can be computationally intractable if one begins with the currently available short-read DNA sequence data.

This thesis is concerned with solving what we call the **binary karyotyping problem** which consists of classifying an organism as having a wild-type or mutant karotypte. The solution takes the form of a new system, called NanZam, built on top of a new DNA mapping technology. We review necessary concepts from biology and statistics and describe and evaluate the system. The thesis contains two novel contributions: a parenthetic structure for describing chromosomal translocations and a novel form of beam search based on empirical Bayes hypothesis testing.

Dedicated to my Grandparents Rose & John Maese.

Acknowledgements

There are many people to whom I want to extend an immense amount of gratitude; without them the completion of this thesis would not have been possible.

First of all, I have to thank my advisor Bud Mishra. Just over a year ago you pointed me towards a book ([22]) and the Stanford Encyclopedia of Philosophy and told me to come back with a report on "how causality and statistics can be related to the natural sciences." While this thesis contains none of that work, it started me in earnest down the path that lead to this thesis. Thank you for your guidance, constantly engaging weekly conversations, and for the patience it must have taken to repeat ideas about NanZam several times before they finally clicked for me. I look forward to collaborating with you as your PhD student.

I also want to thank my second reader Carlos Fernandez-Granda. Your class during my first semester at NYU helped give me the confidence I needed to begin pursuing research. Your course on optimization based data analysis helped me have a more thorough understanding of applied mathematics.

Charles Cantor has been both a friend and mentor since we first met in 2016. Your insights into biology and chemistry are invaluable and I look forward to many more conversations.

Also, Jason Reed & Sean Koebbly at VCU were instrumental in helping me understand the nanomapping technology as well as pointing me towards the Burkitt's Lymphoma chromosomal translocation that shows up in the experimental section of this thesis.

Of course I would be nowhere without the love and support of my parents. Mom and Dad thanks for, quite literally, everything. To be slightly more specific: for being available by phone, text, email, and — presumably, but untested — carrier pigeon through the highs and lows of the last two years, for your unconditional support of my endeavors — academic and otherwise — and for being good friends and great mentors.

Finally, I want to thank my grandparents, Rose and John Maese, to whom this thesis is dedicated. Grandpa: I think of your advice to "master the fundamentals" every day. It serves as a reminder to question what I think I know and to pursue mastery as a goal in and of itself. You are dearly missed. Grandma: your work ethic and generosity are a daily inspiration. You have shown me how little details can make a big difference. While I try to live life inch-by-inch, at portions during the writing of this thesis it was yard-by-yard.

Contents

Chapter 1

Introduction

This thesis introduces NanZam, a hashing scheme for analyzing data produced by a new technology called DNA Nanomapping. By weaving together techniques from image recognition, machine translation, and empirical Bayesian false discovery rate control NanZam solves a difficult and important problem in biology both quickly and accurately.

The particular problem, which we call the *binary karyotyping problem* (BKP), is relevant to both fundamental biological research and clinical oncology. By building on top of nanomapping instead of DNA sequencing technology NanZam can solve the BKP much more quickly than current algorithms and with tuning, we believe, can be nearly perfect in terms of performance.

This thesis contains two research contributions in addition to the system itself. The first is a way of representing chromosomal translocations (defined in Chapter 2) via nested parentheses. To the best of our knowledge this has not been proposed in prior work and has many possible uses. The second contribution is something we have dubbed *Empirical Bayes Beam Search* which uses empirical Bayesian hypothesis testing criteria in the context of heuristic search.

We turn now to a discussion of the binary karyotyping problem.

1.1 The Binary Karyotyping Problem

1.1.1 Mutant and Wild Type Organisms

When characterizing the genetic makeup of a *particular organism* it is common to assign it to one of two categories: **wild type** which means that it is the same as that of the majority of the population or **mutant** meaning it possess a difference. This categorization scheme can be applied to many levels of organismal difference. As a simple example, black panthers are not a truly distinct species of cat, they are in fact members of several different species which have *melanism*. Thus the animals that might be called cougars are *wild type* in terms of fur pigmentation while black panthers are *mutants.*

The state of being a mutant is usually defined with respect to a *specific characteristic*, such as fur pigment or the sequence or expression pattern of a particular gene. Thus an organism may be a mutant with respect to one characteristic and wild type with respect to many others. In this thesis we are concerned with just one: the organism's karyotype.

The **karyotype** refers to the number, type, and and arrangement of genes on a chromosome. If an organism has more than the typical number of chromosomes, has lost or gained genetic material on one or more chromosomes, has genetic material that has *moved* to a chromosome other than where it is typically, or has genetic material moved to a new place on the same chromosome then the organism is a *mutant.*[1] If it is not a mutant then, as mentioned above, it is wild type. Deciding whether or not an organism is wild type is precisely the binary karyotyping problem.

[1] We revisit these concepts in detail in Chapter 2.

1.1.2 Problem Statement

The binary karyotyping problem can be stated as follows:

Binary Karyotyping Problem (BKP): Given a sample of DNA from a single organism, decide whether or not the organism's karyotype is wild type or mutant.

Like almost all scientific problems the computational complexity is dependent on data representation and the binary karyotyping problem provides a stark example. Trying to exhaustively solve the binary karyotyping problem from a collection of short DNA sequences is equivalent to solving the Hamiltonian path problem, which is known to be NP-complete. As short-read DNA sequencing has gained momentums as the subject of a tremendous amount of biochemical engineering efforts as well as biomedical application, fundamental progress has stalled with respect to accurately inferring the long range information to solve the BKP.

DNA nanomapping changes the representation and tames the complexity. Nanomapping is able to look at longer segments of DNA and, by moving to a longer-range resolution, it reduces complexity. By leveraging hashing and beam search NanZam is able to operate with linear expected time complexity in the length of the DNA molecule being analyzed. The development, specification of, and experimentation with NanZam makes up the rest of this thesis.

1.2 Thesis Overview

The rest of thesis can be divided into roughly two parts. Chapters 2 to 4 introduce background material necessary for understanding the motivation for and the design of the NanZam system. In particular Chapter 2 provides a review of prerequisite biology, focusing on the central dogma of genomics as well as transposable elements. We also elucidate the need for NanoMapping technology and the ways it can address challenges introduced by DNA sequencing. Chapter 2 also contains the first contribution mentioned above: the parenthetic representation of chromosomal translocation. It also discusses possible applications of and extensions to this representation scheme.

Chapter 3 introduces the concept of *hashing*. Hashing is a fundamental notion from computer science that involves storing records in an indexed table such that the records can be accessed in constant time. This chapter also discusses a particular application of hashing, called *geometric hashing*, which was originally developed for performing object recognition.

Finally, Chapter 4 introduces false discovery rate (FDR) control and beam search. FDR control is an exciting modern development in the realm of hypothesis testing which has its origins in the 1990s [8] and has been given an *empirical bayesian* justification in the work of Efron [10, 18, 16, 23, 20, 17, 21]. Beam search is a heuristic search technique that employs a hyperparameter called the *beam width* that allows the user to express a tradeoff between *greedy* and *exhaustive* search. This chapter contains the second contribution of this thesis which we call Emprical Bayes Beam Search (EBBS), which is based on choosing the beam members using Bayesian FDR control criteria.

The second part of the thesis consists of Chapters 5 to 7 which describe the system's design, experiments performed with the system on numerical data, and conclude the work. In Chapter 5 we combine the ideas from Chapters 3 and 4 to describe the NanZam system. NanZam is composed two stages — an online stage and an offline stage — which, respectively, build a lookup table and then use it to decide if a new sample is wild type or not.

Chapter 6 applies the NanZam system to a simulated population of 200 patients who possibly have Burkitt's Lymphoma, a blood cancer characterized by chromosomal translocations. In this chapter we demonstrate the system's performance and run time, and experiment with the dependencies of system performance on hyperparameter values.

Finally, Chapter 7 concludes the thesis with a summary as well as a discussion of theoretical and practical directions for future work.

1.2.1 Remarks on Notation

Let us establish some notational conventions. In terms of data structures we treat sets and vectors as being indistinguishable from one another and for indexing we refer to items i through j of an array A with the notation $A[i..j]$ where we assume both endpoints are included. $\mathcal{N}(\mu, \sigma^2)$ corresponds to a normal (Gaussian) distribution with probability density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Finally the symbol $\circ$ denotes concatenation and is applied to strings and paths. If x and y are strings then $x \circ y$ is nothing more than xy, or, to be more concrete

"aa"$\circ$"bb"= "aabb"

If x and y are paths in graphs with vertices v_i, and say say $x = (v_1, \ldots, v_k)$ and $y = (v_{k+1}, \ldots, v_\ell)$ then $x \circ y$ is the path $(v_1, \ldots, v_k, v_{k+1}, \ldots, v_\ell)$ obtained by putting an edge between v_k and v_{k+1}.

Chapter 2

Genomics & Nanomapping

This chapter provides an overview of the biology of genomics as well as some the technological methods used in biological research. While a full description of molecular and cellular biology is well beyond the scope of this thesis, a thorough introduction can be found in [7].

2.1 An Overview of Genomics

Deoxyribonucleic acid - DNA - is the molecule that stores the biological information needed for the vast majority of organisms to exist, procreate, and survive.[1] The double helix structure of DNA is due to the chemical and physical properties of the molecules it comprises, in particular the properties related to hydrogen bonds and purine-pyrimidine stacking. DNA is composed of four *nucleic acids* or *nucleotides*: Adenine, Thymine, Guanine, and Cytosine, often abbreviated A, T, G, and C, respectively. A *strand* of DNA is a series of adjacent nucleotides bound together by a *sugar-phosphate backbone.* The double helix structure of DNA is due to the fact that the four nucleotides have specific bonding affinities: Adenine bonds with Thymine and Guanine bonds with Cytosine. Visually:

$$A \leftrightarrow T \text{ where } \leftrightarrow \text{ indicates a double bond}$$
$$G \Leftrightarrow C \text{ where } \Leftrightarrow \text{ indicates a triple bond}$$

The distinctions in the bonds is due to the fact that A and G are *purines* and C and T are *pyrimidines* which differ in physical sizes and effect the kinds of bonding energies that form. Any one of these complementarily bonded nucleotides is called a *base pair.* The double stranded DNA achieves its double helix structure as a result of the affinities between these molecules and the repelling forces of the same-charged sugar-phosphate backbones.

The physical stacking gives double-stranded DNA it long persistence length and very simple rod like shape. The molecule is chemically inert and physically rigid. This improves accurate information carrying capacity asymmetrically making it hard to read DNA by simple physical and chemical means.

The fact that base pair bindings are essentially deterministic and, as a consequence, it is possible to infer the opposite strand — called the *complementary* strand — from a single DNA sequence means that a common view of DNA is as a one dimensioanl circular or linear sequence of nucleotides. In the language of computation DNA is viewed as a sequence from the alphabet $\Sigma_{\text{DNA}} = \{A, T, C, G\}$. More specifically a DNA sequence σ is a member of ${\Sigma^*}_{\text{DNA}}$, the set of strings created by finite repetition over Σ_{DNA}.

This linear or sequence-based view has origins that are both scientific as well as technical. The scientific reasons find their roots in the **central dogma** of molecular biology which can be stated roughly as follows. Biological information stored in DNA, which is transcribed into Ribonucleic Acid (RNA) which carry the instructions for the creation of proteins. Proteins are large complex molecules that perform key functions in the cell and body as a whole.

[1] This vast majority excludes RNA viruses and prions, self-reproducing molecules that store their information in molecules other than DNA. They are not relevant to the content of the thesis and so we ignore them.

These proteins fulfill many functions. They form structural components of cells, signal nearby cells, form surface receptors as well as ligands to receive and process extracellular signals, and regulate the translation of DNA into (other) proteins, among other functions. The last point shows that the flow of information is not unidirectional, that is, DNA can control proteins which in turn control DNA.

While the full scope of the structure and function of proteins is well beyond the focus here, two points need to be emphasized. First is the idea of a *gene*. Since there is no universally accepted definition of a gene we adopt the following convention: the *gene* for a particular protein will refer to the region in the organisms DNA — the *genome* — that gives rise to that particular protein.

The second important point is that genes and proteins are subject to evolutionary pressure. In single-cell organisms the case is clear: genes encode proteins which help individual cells navigate the environment; they favor cells able to process nutrients others cannot, or that can adapt to environmental changes better. They select the cells by giving them a better chance of survival and thus ability pass on those genes.

In multicellular organisms the *germline* cells encode hereditary information that (nondeterministically) controls the destiny of somatic (body) cells. Thus multicellular organisms have fitnesses decided by how germline cells orchestrate compelx interactions of smomatic cells. Common examples of this process going awry in humans are type I diabetes, lactose intolerance, some autoimmune diseases, among others.

The amount of DNA varies widely from organism to organism, going from several thousand base pairs (several *kilobases* (kbp)), to several million or billion bases pairs (Megabases (Mbp) and Gigabases (Gbp), respectively).

Human beings in particular have approximate 3 billion base pairs in their genome. However since humans are *diploid* organisms — they have 2 full copies of the genome in every cell – the total aggregate number of base pairs is around 6 Gbp.

Since most human cells contain a full copies of the diploid genome, the physical reality of the genome is not a linear array or free floating double helices. From double helix molecules DNA coils tightly around itself — and around both *histone* and *chromatin* molecules — to create dense bundles of genetic information called *chromosomes*. Humans have 23 chromosomes — two copies of each — that look roughly χ shaped. Bacterial chromosomes, on the other hand, are often circular in shape.

As a result of this coiling genes and nucleotides that may be *far away* when the DNA is viewed linearly may be extremely close in three dimensional distance.

2.1.1 LINEs, SINEs, and ALUs

Not all DNA is contained in genes. In fact, not all of the DNA in a region marked out as being part of a gene gets used in its expression. A cell's *transcription program* constantly engages with the nuclear DNA to decide what proteins are present in or excreted by the cell and in what volumes.

It is also true that not all DNA stays put. *Transposable elements* are pieces of DNA that can change their position in the genome [34]. These can induce novel mutations, correct existing ones, and even lengthen the genome in certain instances. There are two main classes of transposons: *retrotransposons* which are transcribed from DNA to RNA and then back, and *DNA transposons*,which use the protein transposase. The mammalian genome has an abundance of transposable elements but it is unclear what fraction are still actively transposing. In humans in particular retrotransposons account for around 42% of the human genome and DNA transposons only 2-3% [32]. We focus on the former in this thesis.

In particular we will discuss three kinds of retrotransposons in the human genome: Long Interspersed Nuclear Elements (LINEs), Short Interspersed Nuclear Elements (SINEs), and Alu elements. These are implicated in multiple biological processes as well as human disease. Alu elements are important in our discussion of nanomapping in Section 2.3.

LINEs

LINEs make up around 21% of the human genome [32] and each is around 7,000 base pairs in length.

LINEs have three subtypes: L1 elements (written LINE1), which are the most common and only active LINE element in human genomes today, as well as L2 (LINE2) and L3 (LINE3) elements [43].

SINEs & Alu Elements

SINEs are, naturally, shorter than LINEs with lengths ranging from 100 to 700 base pairs [47]. They are usually associated with parasitic activity early in the evolution of Eukaryotic genomes.

Alu elements or *alu*s, named for a restriction enzyme from the organism *Arthrobacter luteus* [41], form a subfamily of SINEs that are abundant in the human genome. It is estimated that the 300 base pair *alu* sequence is present at over one million locations in the human genome [45, 13] and is thought to play a role in regulating gene expression [11]. They have also been linked to certain human diseases [14, 6]

The motility of genetic elements leads directly to the notion of genetic mutation.

2.1.2 Genetic Mutation

Above we discussed the role that genes play in organismal fitness. Genes are not perfectly stable; they can vary across generations, across various cells in the same organism, as well as *within in the same cell* over time.

Changes in a genome are called *mutations* and can take many forms. On the nucleotide level there are three typical kinds of *single nucleotide polymorphisms* (SNPs[2]):

- **Insertion:** A single nucleotide is added where it was not before:

$$\sigma = \text{ATCAT} \;\mapsto\; \text{AT}\underline{\text{G}}\text{CAT} = \sigma'$$

- **Deletion:** A single nucleotide is removed from where it was:

$$\sigma = \text{ATCAT} \mapsto \text{ACAT} = \sigma'$$

- **Substitution:** A single nucleotide of one type is replaced by another:

$$\sigma = \text{ATCAT} \;\mapsto\; \text{AT}\underline{\text{G}}\text{AT} = \sigma'$$

These kind of mutations are not limited to just one nucleotide. It's possible to see multiple-nucleotide insertions, substitutions, and deletions. This pattern of generalization can be continued to gene-level events. A gene may be duplicated or deleted and, in extreme cases, the same may happen to entire chromosomes. A change in the number of copies of a gene (and by extension a chromosome) is called a *copy number variation*.[3]

Mutations often have deleterious effects – slight changes in a key protein can render it useless meaning the organism/cell must somehow survive without its function, if it can. The results of a loss of a full gene or a chromosome are not any better. As a consequence many organisms - both single and multi-cellular - have evolved to repair genetic damage, or, in the case of multicellular organisms for cells with mutated DNA to undergo a process called programmed cell death (PCD) via apoptosis.

Key genes tend to be conserved over the history of an organism, but *too* inflexible a genome leads to an organism unable to adapt to exogenous environmental shocks.[4] As a result heterozygosity and gene duplication ([39]) provide organisms with the means to mutate and allow new genes to develop without sacrificing those key for survival.

[2] Pronounced "snips".

[3] In many cancers only one copy of a gene is lost, meaning it is only lost on *one* of the two chromosomes. This is called *loss of heterozygosity*.

[4] This can be viewed as a poor tradeoff between exploration and exploitation.

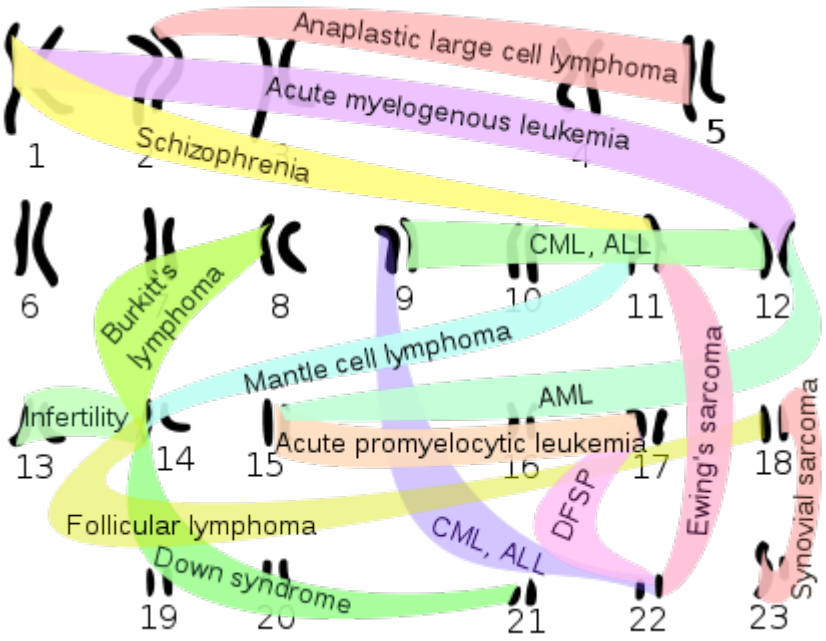

Figure 2.1: A visualization of the role chromosomal rearrangements (i.e: translocations) play in human disease. It is worth noting the preponderance of cancers in the graphic. ([27],Public Domain)

2.1.3 Translocations: old genes in new contexts

One particularly interesting type of mutation is *translocation*. In translocation from some chromosome χ_i to another χ_j — often notated $t(i:j)$ or $t(\chi_i:\chi_j)$ — a portion of the DNA in χ_i gets inserted into χ_j. The point where this happens along χ_j is called a *breakpoint*.

Translocations are interesting because they can result in interactions between genes that previously would not interact. A common phenomenon in cancer – one of particular interest in this thesis — is exhibited by the oncogene myc, a gene that when over-expressed can result in uncontrolled cellular proliferation. The translocation of an upstream promoter gene to the region ahead of myc can lead to an overexpression of myc and a resultant neoplasia. Translocations fig. 2.1 shows how chromosomal translocations are implicated in some human diseases, including many cancers.

2.1.4 Parenthetic Representation of Translocations

The first novel contribution of this thesis is a scheme for representing chromosomal translocations by well-formed parentheses. To begin with as chromosomes can be thought of as linear strands — and since these strands are evidently finite — a wild type or untranslocated chromsome χ_i can be represented by a pair of open and closed parenthesis:

$$(_{\chi_i}\ \ _{\chi_i}) \text{ or } (_i\ \ _i)$$

A simple translocation $t(\chi_j\chi_i)$ can be represented as a a pair of nested parentheses:

$$(_{\chi_i}(_{\chi_j}\ \ _{\chi_j})_{\chi_i}) \text{ or } (_i(_j\ \ _j)_i)$$

More complicated events such as a $t(i:j)$ followed by a $t(k:j)$ where the areas on χ_i of translocation do not intersect can be represented by

$$(_{\chi_i}(_{\chi_j}\ \ _{\chi_j})(_{\chi_k}\ \ _{\chi_k})_{\chi_i}) \text{ or } (_i(_j\ \ _j)(_k\ \ _k)_i)$$

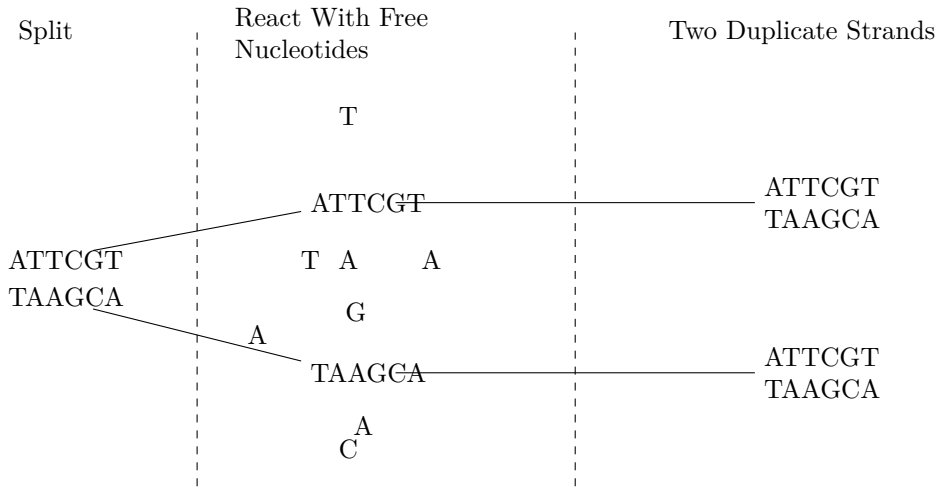

Figure 2.2: Illustration of Action of DNA Polymerase

modulo the order of the translocation. If they intersect the representation will look something like:

$$(_{\chi_i}(_{\chi_j}(_{\chi_k}\ \ _{\chi_k})_{\chi_j})_{\chi_i}) \text{ or } (_i(_j(_k\ \ _k)_j)_i)$$

It's well known that these kinds of parenthetic structures can be represented by (probabilistic) context free grammars [15, 29] which make them both amenable for use as model inputs as well as easily processable by extant algorithms. In particular these parenthetic structures for a Dyck language which makes them easily translatable into the language of automata.

2.2 Technological Considerations

A lingering question remains: how do we obtain the sequence of an organism's genome. Double helix strands are invisible to optical microscopes, and even if they were, manual sequencing would be cumbersome. Thus we can state the **sequencing problem:**

> Given a molecule that is known to be DNA,
> return the $\sigma \in \Sigma^*{}_{\text{DNA}}$ that makes up one strand of the molecule.

Many technologies have been developed to address this problem and they continue to be the subject of biochemical engineering research.

The first marquee achievement of DNA sequencing was the production of the draft genotypic human reference genome by the human genome project (HGP) in the early 2000s. The technology in the HGP was Sanger sequencing and BAC maps. Modern methods, however, use an enzyme called *DNA Polymerase*. DNA polymerase plays a key role in cell replication by cloning DNA. Put roughly, the way DNA polymerase operates is as follows. It "unzips" the double stranded DNA to make two naked single strands. Then, in the presence of freely available A,T, C, and G nucleotides, DNA polymerase proceeds down the single strands and bonds the correct complementary nucleotide to the available pair eventually creating two duplicate strands. The process is shown in fig. 2.2

Most DNA sequencing technologies involves leveraging the action of DNA polymerase, either by making bonding events observable, exerting some control over the rate of bonding, or both. The first widely adopted method of sequencing — Sanger sequencing — used chemically altered nucleotides to terminate the process of DNA polymerase. These are mixed with a template strand, a primer, and DNA polymerase in a small ratio with normal nucleotides and a small amount of a *single* chain terminating nucleotide. Thus we have A, T,C,G mixed in

with a special terminating, T, say. Many fragments are generated and then sequenced by either using gel electrophoresis, dyes, microfluidics, or other techniques.

Modern sequencing is based on similar concepts. These technologies, called next-generation sequencing (NGS) are marked by high throughput (measured in base pairs sequenced per unit time) and greater accuracy. There are several types of NGS that merit discussion.

The first is *pyrosequencing*. In pyrosequencing DNA polymerase is again used but this time the nucleotides are modified to fluoresce upon being incorporated into the new strand. Again a solution with DNA polymerase, a template strand, and a primer is required. This time modified nucleotides are sent into the solution one at a time and a photosensitive camera measures the amount of light emitted. 454 Sequencing is essentially pyrosequencing done in parallel.

By far the most common NGS technique is Illumina/Solexa sequencing. This also relies on DNA polymerase but here the DNA is affixed to a slide and nucleotides are added in rounds which fluoresce with different colors. The nucleotides are chain terminating but can be chemically modified to allow the chains to extend.

2.2.1 Shortcomings of NGS

There are two chief sources of shortcomings of next generation sequencing: its dependence on polymerase chain reaction (PCR) and the short length of the reads that it produces.

Polymerase chain reaction is, as the name suggests, yet another DNA-polymerase-based technique. The purpose of this technique is to increase the amount of DNA available for sequencing from a small sample by cloning it. A special version of DNA polymerase that can operate at high temperature is added to a solution with the sample DNA and free floating nucleotides. The solution is then repeatedly and rapidly heated and cooled to cause the DNA polymerase to act repeatedly, generating multiple clones of the original double-stranded molecule. The issue with using PCR is that errors early on in the PCR process persist throughout, and are undetectable and hence unrecoverable. Multiple such errors may lead to incorrect sequences.

Read length — the length, in base pairs, of a molecule that can be sequenced in one run — for NGS techniques are usually less than 500bp. This allows for *rapid* sequencing but makes for down-stream computational problems. In particular the ordering of the short reads or finding the chromosome of origin is difficult.

2.3 DNA Mapping

Mapping is a technology that takes a higher level view of genomics than sequencing but is no less important. In DNA mapping the goal is to find points on a DNA molecule where a specific enzyme binds,nicks, or cleaves. This can be done on a molecule with lengths on the order of hundreds of Kbp. As a result of its long read length mapping has played a major role in genome assembly and de-novo sequencing (sequencing without a reference).

Another core application of mapping is the detection of structural variations and chromosomal translocations.

Unlike DNA sequencing,mapping — particularly the most well-established form of mapping, optical mapping [3, 2, 1, 5] — has not been as widely adopted or as influential as sequence. The reasons for this have to do with the precision of the tools. The state of the art in optical mapping is such that prohibitively deep coverage is required to make the result minimally workable.

A new technology, on which this thesis is based, called DNA *nanomapping* aims to improve upon these technical issues.

Nanomapping involves targeting certain *alu* repeats with a CRISPR-cas9 molecule as a labeling enzyme. Nanomapping adopts a “fixed molecule, mobile read head approach” in the following sense. The DNA molecule — untouched by PCR — is stretched “flat” on a plate where the CRISPR enzyme binds at specific point. Then a nanocantiliver attached to a DVD optical drive moves along the molecule and maps the points where the CRISPR molecule has

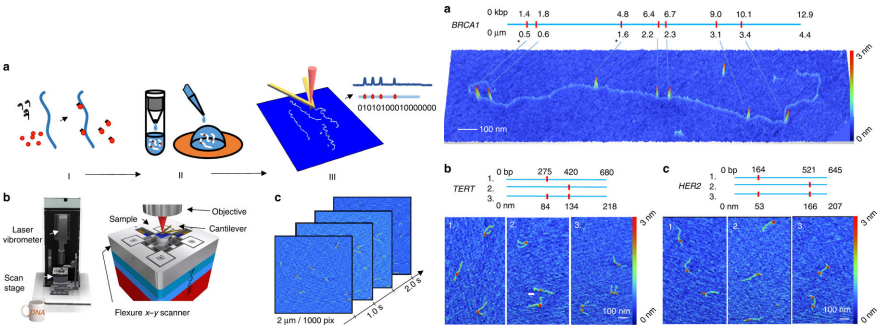

Figure 2.3: A depiction of DNA nanomapping[36]

bonded to DNA. The output of nanomapping can be seen in fig. 2.3. The result of a nanomapping procedure is a set of lengths between enzyme binding points as well as a set of statistics about the probabilistic correctness of the map.

The abundance of *alu*s in the human genome make them good targets for the CRISPR enzyme as they will have reliable bindings at fixed locations.

Modern genomics often proceeds from the point of view of having a *reference genome* that is, a gold standard of what a genome sequence "should" look like. Nanzam will proceeds much the same way, and will presume that a set of reference maps exists. These will be stored and accessed by a hash table, the subject of the next chapter.

Chapter 3

Geometric Hashing

3.1 An Overview of Hashing

3.1.1 Rapid record access

It is a common need in computer science to store and maintain records. In fact the large scale storage combined with rapid access to stored entities forms a large part of modern business applications of database theory and software. A full discussion of hashing can be found in [30].

These large scale implementations trace their roots to *hashing*[26]. Put simply, hashing concerns how to manage a table so that the tasks of inserting, searching, and deleting items from that table can be done rapidly and effectively.

More specifically we have a set $\mathcal{K}$ of *keys*, a *table HT* consisting of n cells and a *hash function* $h : \mathcal{K} \to \{1, \ldots, n\}$. From now on the notation $[n]$ will mean the numbers $\{1, \ldots, n\}$. The goal is to use this machinery to store *records* or *key-value pairs*. A *key-value pair* is just as the name suggests, a tuple of a *key* $k \in \mathcal{K}$ and a value v, which, in principle can be anything. A common example is that k might be a student's name and the v their home address, application status, etc.

The tasks of searching, inserting, and deleting all involve the same basic process: for a record $R = (k_R, v_R)$ compute $h(k_R)$. Go to that cell in the table and either return what's there (search), add v_r to that cell (insert), or remove the object there (deletion). Deletion has some complications that arise from the potential for empty cells and the case when each cell can hold more than one v. In general, delete and search only require a key, not a value.

3.1.2 Collisions and CRPS

A natural question is: what happens if an insertion is attempted on a cell that is *already storing something*? This event is called a *collision* and a fully developed hashing scheme will have a way of handling this, called a *collision resolution procedure* (CRP).

There are two common CRPs: separate chaining and open accessing. In separate chaining you replace single-occupancy cells in the hash table with linked list. Each new insertion is appended to tail (or inserted at the head, or in order) to the linked list. Open hashing works as follows: If the cell is occupied, traverse the hash table until an empty cell is found. If no cell is empty, evict a record.

Collectively, these together form a hashing scheme. More precisely:

Definition 3.1.1 (Hashing Scheme)**.** A hashing scheme is a tuple

$$\mathcal{H} = (\mathcal{K}, \mathcal{V}, h, HT, n, \text{CRP}) \tag{3.1}$$

where $\mathcal{K}$ is the set of all possible keys, $\mathcal{V}$ the set of all possible values (and therefore $R = \mathcal{K} \times \mathcal{V}$ is the set of all possible records), h is a hash function, HT is a hash table of size n, and CRP is a collision resolution procedure.

3.1.3 Optimal Hashing

Given some hashing scheme, how do we evaluate it? Also: what theoretical results are there? For the sake of this discussion we'll confine our attention to properties of h and avoid a discussion of various CRPs, n, and HT choices. A truly comprehensive treatment can be found in [30].

For our purposes v, k will be strings. Usually the behavior of h is modeled probabilistically. The assumpiton is that values are picked uniformly at random, that is :

$$P(h(k) = i) = \frac{1}{n}$$

Yao showed that uniform hashing is optimal [48].

The main concern for the purposes of this thesis is maintaining $\mathcal{O}(1)$ access time. In a linked list scheme we have for inserting M keys into n cells is m/n. Let $1_{j=k}$ be the event that the ith insert is put into cell k. As before this happens with probability $p = \frac{1}{n}$. The number inserted into k is $\sum 1_{j=k}$ which implies we expect m/n to be inserted into k.

3.1.4 Hashing For Retrieval

In typical scenarios the hash function h destroys the structure in the key set $\mathcal{K}$. Objects that may be "close" to each other in key space may be hashed to very different locations. As a trivial example consider $\mathcal{K} = \mathbb{Z}^+$ and $h(k) = k \mod 11$. Then for numbers 10 and 11, which are as close as two distinct elements of $\mathcal{K}$ can be, we have $h(10) = 10, h(11) = 0$ which means their hash values are as far away in the table as possible. Thus, for general similarity search hashing as described here is useless.[1]

However, hashing as described here is deterministic so it's useful for *exact* matching or fingerprinting because *identical keys* map to *identical places.* This has had key applications to string applications to string matching and object recognition, which we now review.

Rabin-Karp String Matching

In a classic paper Rabin & Karp [] provide a set of algorithms for general pattern matching via fingerprinting functions. While a complete review of their paper is beyond the scope of this thesis a simple example is illustrative of their concept of fingerprinting and how we apply it in Nanzam.

Consider two binary strings A and B where A is no longer than B. That is $A \in \{0,1\}^m, B \in \{0,1\}^n, m, n \in \mathbb{Z}^+, m \leq n$. We call A the *pattern* and B the *text.* The challenge is to find the first exact occurrence of the pattern in the text. To do this we first compute

$$\bar{a} = \sum_{i=1}^{n} A_i 2^{n-i} \tag{3.2}$$

Now for some contiguous substring B, $B[j..j+n-1]$ with length n we have that:

$$\bar{b(j)} = \sum_{i=j}^{j+n-1} B_j 2^j$$

is equal to $\bar{a}$ if $B[j...j+n-1] = A$. The typical algorithm runs as follows: First compute $\bar{a}$. Then for $j = 1, \ldots, m-n$ compute $\bar{b(j)}$. If $\bar{b(j)} = \bar{a}$ compare $B[j...j+n-1]$ to A bitwise. Terminate if an exact match occurs. In order to run this algorithm in linear time an update step is used that computes $\bar{b(j)}$ from $\bar{b(j-1)}$ in constant time.

[1] This is not *generally* true of hashing, as a tremendous amount of research [12, 33, 4] has gone into constructing hash functions that map similar keys to similar buckets. These are termed *locality sensitive hash functions.*

Viewing the set of all strings as $\mathcal{K}$ we can see that eq. (3.2) is almost in the form of a hash function. The problem is that the function is not bounded from above and so it does not necessarily map to a finite number of hash table cells. Thus a slight modification is necessary:

$$\bar{a} = (A_1 * 2 \mod q + A_2) * 2 \mod q + \dots$$

where q is a randomly chosen prime in the range $[1, nm^2]$. We can generalize this to a family of hash functions parameterized by two integers p and q, with q a prime greater than p as follows:

$$h_{RK}(A; p, q) = (A_1 * p \mod q + A_2) * p \mod q + \dots \tag{3.3}$$

Put in this light we can see the Rabin-Karp formalism as fingerprint A via the $h_{RK}(A, 2, q)$ into a hash table of size q. Then we scan B left to right and hash substrings of length m (i.e: performing "search"). If a collision occurs then we check bitwise for an exact match.

Rabin and Karp [**RK**] provide many variations on this approach but the most interesting generalization of this technique was to general shape matching. A similar approach applied to object recognition in 2D and 3D images, called *geomtric hashing* is also important to the development of NanZam.

3.2 Geometric Hashing

Suppose your records have labels, that is $V_r = (D_r, L_r)$ where D_r is some data set and L_r is a label such as "spam" or "ham". For *geometric hashing* the data is images and the labels are the content of those images.

More specifically, geometric hashing is a hashing-based technique to recognizing objects in images. We suppose that there are finitely many kinds of objects that the system needs to recognize, i.e: a finite set of *models*. The problem is can be stated as follows: in a given image recognize which models are in the scene and the coordinates that belong to that model. If no model is present in the scene, return the empty set.

Geometric hashing approaches [31] employ a supervised learning point of view and utilize two stages: an *offline* step where a look-up table is built from labeled data and an *online step* where this look-up table is used to perform object recognition.

In the offline step we build a look-up table from labeled training data. We assume that there is a collection $\mathcal{D}$ of images that has each been passed through a *featurizing transform* ϕ to extract the coordinates of the models. Each coordinates are attached to a label ℓ_i of which model the coordinate belongs to. The coordinates are passed through a series of affine transformations that give new coordinates which are then hashed into a table which holds a record containing the transformed coordinates, which model they represent, and some other metadata.

The *online step* uses the same featurizing transform in conjunction with the constructed hash table to detect objects in the model space. Each new image is featurized in the same manner as in the offline step then the coordinates are put through the same affine transformations and hashed. The contents of the cells are tallied and the final result is called via a majority vote. If the vote meets a pre-specified threshold then the object is called. If no object meets the threshold the coordinate is called as belonging in to the null set.

The general schematic of geometric hashing is adopted in Nanzam. However, we change the voting scheme to *beam search* such that false positives rarely happen. This is the subject of the next chapter.

Chapter 4

Beam Search & False Discovery Rates

In this chapter we introduce two key concepts— false discovery rates and beam search — on the way to introducing one of the contributions of this thesis: empirical Bayes beam search (EBBS). EBBS is built on seeing a connection between the optimality as desired in beam search and that of finding significance in a hypothesis testing setting.

4.1 A Brief Overview of Multiple Hypothesis Testing

Hypothesis testing plays a starring role in classical statistics. In the simplest characterization it involves quantifying how much evidence an observation provides for or against a hypothesis.[1] The typical case assumes a *null hypothesis*, often written $\mathcal{H}_0$, and quantifies the evidence given by some observation $\tilde{x}$ by answering the question "under the $\mathcal{H}_0$ what is the probability that I would observe $\tilde{x}$?" As this number gets smaller and smaller $\tilde{x}$ gathers stronger and stronger evidence against the null hypothesis. If the evidence is strong enough — usually written as a p value being below a certain threshold — then we say we *reject* the null hypothesis. Otherwise we say we *fail to reject* the null hypothesis. We will be concerned with two related ways of quantifying evidence: p-values and z-values.

4.1.1 p-values and z-values

We assume we have a *null hypothesis* $\mathcal{H}_0$ which has a specified and known density. For example, one might be concerned

In order to quantify the we can form *rejection regions* R_α for any real $\alpha \in (0, 1)$ as the set of possible observable x (sometimes called the *input space*) satisfying

$$\Pr\{x \in R_\alpha\} = \alpha$$

From this we can define the p-value of an observation $\tilde{x}$ as

$$p(\tilde{x}) = \inf_\alpha \{\tilde{x} \in R_\alpha\}$$

It is possible to show that p values are uniformly distributed over $(0, 1)$ which means that if we write

$$z(x) = \Phi^{-1}(p(x))$$

then

$$z(x) \sim \mathcal{N}(0, 1)$$

[1] There are many forms of hypothesis testing and a full discussion of them is well beyond the scope of this thesis. We confine our attention to one which involves a correspondence between $\mathcal{H}_0$ and a distribution. This is fairly general but we forgo a technicality of where these distributions *come from* such as in permutation testing, bootstrap sampling, etc.

where Φ^{-1} is the inverse standard normal cumulative distribution function. An alternate way to arrive at z-values from a set of data points $x_1, \ldots, x_n$ is to perform what is called *centering and scaling*. For this data set we compute a sample mean $\bar{x}$ and sample standard deviation $\bar{\sigma}$ and transform each x_i to $\tilde{x}_i$ via:

$$\tilde{x}_i = \frac{x_i - \bar{x}}{\bar{\sigma}}$$

By the central limit theorem these will be approximately normal in the limit as $n \to \infty$. Ignoring rates of convergence we gloss over and assume that the $\tilde{x}_i$ are *approximately normal* which we can notate

$$\tilde{x}_i \dot{\sim} \mathcal{N}(0, 1)$$

An advantage of having z values is that it renders accessible a vast body of theoretical work on Gaussian distributions to aid in our calculations. To motivate false discovery rate control we first describe the family wise error rate and the Bonferroni Correction.

4.1.2 Simultaneous Testing and The Bonferroni Correction

In hypothesis testing one key type of error is type I error, which occurs when we reject a genuine null hypothesis. The purpose of p-values is that they essentially quantify the probability of making a type I error.

In the multiple or simultaneous testing scenario things are different. In this case there are a total of n hypothesis $\mathcal{H}_1, \ldots, \mathcal{H}_n$ that are associated with p-values $p_1, \ldots, p_n$ more care is needed. We define the *family-wise error rate* (FWER) as the probability that in this collection of hypotheses we make at least one type I error. If we want this to be less than some predetermined value γ then Bonferroni correction says that we should reject only p-values where

$$p_i \leq \frac{\gamma}{n}$$

This is a stringent control that, for very large n will reject almost everything. False discovery rates provide a more permissive criterion for rejection in the multiple testing scenario.

4.1.3 The Bayesian Two-Groups Model

To formalize this all from a Bayesian point of view we begin with the two groups model:[2]

$$\begin{aligned} \pi_0 = \Pr\{\text{null}\} \quad & f_0(z) = \text{ density of z given case } i \text{ is null} \\ \pi_1 = \Pr\{\text{not null}\} \quad & f_1(z) = \text{ density of z given case } i \text{ is not null} \end{aligned} \tag{4.1}$$

from this we can derive the overall mixture density:

$$f(z) = \pi_0 f_0(z) + \pi_1 f_1(z) \tag{4.2}$$

False discovery rates are ways of assessing the probability of making an error in the above model. There are many ways of estimating false discovery rates but we focus on a particular empirical Bayes quantity called the false discovery rate.

4.1.4 Local False Discovery Rates

The local false discovery rate, written $Fdr(x)$, is the probability that an observation comes from the null distribution given its z value. To be concrete we can image n observations with z values $z_1, \ldots, z_n$. For any given one of them we write the local false discovery rate as:

$$\Pr\{\text{null}|z_i\} = \frac{\Pr\{\text{null}\}\Pr\{z|\text{null}\}}{\Pr\{z\}} = \frac{\pi_0 f_0(z)}{f(z)} \tag{4.3}$$

where π_0, f, f_0 are as in Equation (4.1).

[2] Much of the development in this section follows the development in [19, 22]. The former is much more in-depth than we are here.

4.2 Estimating Local FDR

To use Equation (4.3) in a practical context we need ways to estimate $f(\cdot), f_0(\cdot)$, and π_0. A common method is to assume that $p_0 \approx 1$, say 0.9 or 0.99, and that $f_(x) \sim \mathcal{N}(0,1)$. In this case we say that $f_0(\cdot)$ follows the theoretical null. The only remaining quantity to estimate is the mixture distribution. There are many ways to do this, but we follow one called *Lindsey's method.*

Lindsey's method supposes that $f(z)$ takes the form

$$f(z) = \exp\left\{\sum_{j=0}^{J} \beta_j z^j\right\} \tag{4.4}$$

and then fits the β_j values via a Poisson linear model. As J grows large the estimate in Equation (4.4) becomes increasingly nonparametric. In [19] and in this thesis the value of J is 7.

To fit the β values we proceed as follows. Suppose we have N cases, assume we have z scores $\boldsymbol{z} = (z_1, \ldots, z_N)$. Denote their range in $\mathbb{R}$ by $R_{\boldsymbol{z}}$. We divide $R_{\boldsymbol{z}}$ into v equal-width bins of width ω and keep counts $c_1, \ldots, c_v$ of the number observations in each bin. Letting x_i be the midpoint of the i'th bin we have that the expected number of counts in bin i is $e_i = N\omega f(x_i)$.

On the assumption that the c_i's are independent Poisson random variables each with parameter e_i we construct the Poisson generalized linear model

$$\log(e_i) = \sum_{j=1}^{J} \beta_j x_i^j$$

which can then be fit using standard software packages. Lindsey's method is given in algorithmic notation in Algorithm 1.

There are ways to also fit π_0 and f_0 using the data set. These methods are called fitting an *empirical prior* and *empirical null*. These are described in chapter 7 of [19].

Input: A set of scores $\boldsymbol{s}$ assumed not normalized (if normalized skip lines 1 to 3), a polynomial degree J, bin width ω
Output: A parameter vector β

1 $\mu_{\boldsymbol{s}} \leftarrow \frac{1}{|\boldsymbol{s}|}\sum_{i=1}^{|\boldsymbol{s}|} s_i$;
2 $\sigma_{\boldsymbol{s}} \leftarrow \sqrt{\frac{\sum_{i=1}^{n}(s_i-\mu_{\boldsymbol{s}})^2}{n}}$;
3 $s_i \leftarrow \frac{s_i-\mu_{\boldsymbol{s}}}{\sigma_{\boldsymbol{s}}}, i = 1, \ldots, |\boldsymbol{s}|$;
4 $Bins \leftarrow \text{EqualWidthBinning}(\boldsymbol{s}, \omega)$;
5 $x_i \leftarrow \text{GetMidpoint}(Bin_i), i = 1, \ldots, k$;
6 $c_i \leftarrow \text{CountObservations}(Bin_i)$;
7 $\beta \leftarrow \text{PoissonGLM}(c_i \sim \sum_{j=0}^{J} \beta_j x_i^j)$;
8 **return** β.

Algorithm 1: Lindsey's Method

With a firm handle on the empirical Bayesian view of false discovery rate control we turn now to a description of beam search.

4.3 Beam Search

Beam search is a heuristic search technique which has roots in classical artificial intelligence [42], and has applications to speech recognition, and more recently statistical machine

translation [24, 40] as well as genome assembly and base-calling [38, 35]. The conceptual origins of beam search come from *state-space search* [49]. In state-space search the algorithm is viewed as a search on a directed graph where each node has 0 or more outgoing edges. State space search algorithms start at an origin or root node s_0 and attempt to find a goal node s^* or an optimal path to a node which has no outgoing edges, i.e. a *terminal node*. Greedy search, exhaustive search, and others are part of this paradigm.

Beam search can be seen as a generalization of greedy search that allows for a trade-off between exploration and exploitation to be tuned through a hyper parameter w called the *beam width.* The algorithm proceeds from a start state s_0 which has a number of successor states $S = \left\{ s_1^{(1)}, \ldots, s_\ell^{(1)} \right\}$.

For each of the ℓ paths from s_0 to one of the successor states a **score** is computed and stored — often the score corresponds to a log posterior odds, but we will treat the most general case. At the end of this process the best w scoring sequences $s_0 \mapsto s_i^{(1)}$ are kept and the others permanently discarded.

This process continues, each subsequent round generates candidate sequences/paths for the set of all the successor states, until the end node of the paths/sequences in the beam. The algorithm terminates when there are no successor states left to explore or when a goal node has been reached. Pseudocode for this simplified version of Beam Search is in algorithm 2. With $w = 1$ beam search becomes greedy search and with $w = \infty$ (which can be thought of as storing all candidate sequences) beam search becomes exhaustive search.

Input: A start state s_0, beam width w, a score function $\xi(\cdot)$, data $\mathcal{D}$
Output: A set S of w candidate sequences & their scores

initialize a set S;
$S \leftarrow \emptyset$;
while *There are successor states to at least one* $s \in S$ **do**
  $B \leftarrow \emptyset$;
  for $s \in S$ **do**
    for $s' \in$ *Successors*(s) **do**
      $B.insert((s \oplus s', \xi(s \oplus s')))$;
    end
  end
  Sort B by score.;
  $S \leftarrow B[1..w]$;
end

Algorithm 2: Vanilla Beam Search

4.3.1 Generalizations of Beam Search

There are multiple ways to generalize beam search. Common methods are adding additional structure to the beam, adding backtracking (discussed below), and making the beam width adaptive. Adding additional data structures allows beam search to return more than just a set of scores and sequences. By adding stacks, trees, queues, etc. properties of the sequences can be computed at the same time, enriching the information gleaned by beam search.

Completing Beam Search

Like most heuristic methods beam search is sound but not *complete*, meaning it is possible for it to return suboptimal solutions. The reason that beam search is not complete is that it may abandon paths that are locally non-optimal by removing them from the beam. Beam search will never revisit these decisions and so will return suboptimal results. Two algorithms have been developed to make beam search complete, and these are given in [50, 25].

Post-Processing

Since beam search can return more than one candidate when it terminates (if the beam width $w > 1$) there is a lingering question of how to handle these candidates and how to choose one of them. There are a number of options. If the goal of the algorithm is to find an optimal value then the top scoring sequence can be returned. In the case of classification a majority vote, or a probability distribution can be returned. More complicated schemes can be developed to meet specific problem needs.

All these modifications can be made to empirical Bayes beam search as well.

4.4 Empirical Bayes Beam Search (EBBS)

We now come to one of the key contributions of this thesis: *Empirical Bayesian Beam Search (EBBS)*. The key to creating a successful beam search instance is the decision of good beam width and a good scoring function. We want to find a balance between choosing all positive directions and picking the most promising exploration direction.

In the *state space view* of beam search the future paths can be viewed in the language of the two groups model of Equation (4.1). At each round of beam search the scores can be evaluated as representing one of the two hypotheses:

$$H_0 = \{\text{The path is uninteresting/ likely non-optimal}\}$$
$$H_1 = \{\text{The path is interesting/ likely optimal}\}$$

We want the algorithm to choose to explore as few H_0 paths and as many H_1 paths as possible. If we make the algorithm follow the most interesting paths but they don't ever result in a value that is interesting enough to be considered non-wild-type then we can throw them out. The algorithm for EBBS is given in Algorithm 3.

Input: A set of Root States S_0, beam width function $w(t)$, a score function $s(\cdot)$, data $\mathcal{D}$, FDR Frequency Parameter K,FDR threshold t
Output: A set S of $\hat{w}$ candidate sequences and their cores

```
Initialize S ← S_0;
t ← 0;
while There are elligible successor states for at least one s ∈ S do
    Initialize empty set B;
    for s ∈ S do
        for s' ∈ Successors(s) do
            ŝ ← s ⊕ s';
            Score(ŝ);
            B.insert(ŝ);
        end
    end
    if t mod K = 0 then
        Compute Fdr(b) for all b ∈ B;
        S ← {b ∈ B : Fdr(b) ≤ t};
    end
    else
        Sort B by score;
        S ← B[1..w(t)];
    end
    t ← t + 1;
end
```

Algorithm 3: Empirical Bayes Beam Search

Modifications to EBBS

Algorithm 3 can be modified in many ways to adapt it to specific problems. First, it can be seen as a strict generalization of Beam search: EBBS reduces to beam search if $K = \infty$, line 2 is removed, and we set $w(t)$ to be a constant function $w(t) = w_0$.

This generalization provides some flexibility as well. The score function can be adapted as needed as well as the frequency with which the beam members are chosen using FDR. In particular a value of $K = 1$ means checking FDR on each round and $K = \infty$ corresponds to never checking for false discovery rates.

Line 14 can also be modified as follows to make it closer to true beam search. If we assume B has been sorted in increasing order of $Fdr(b)$ then a very similar formulation is:

$$S \leftarrow \{b \in B : Fdr(b) \leq t\} \cup B[1..w(t)]$$

This modification pads FDR selection with the lowest FDR scoring sequences. In the event that $w(t)$ elements have $Fdr(\cdot) \leq t$ then the two formulations are the same. If there are fewer than $w(t)$ such sequences then the beam will have the prescribed length with members having the lowest $Fdr(\cdot)$ values available. Common forms for $w(t)$ are constant functions and decaying beam widths.

With a development of EBBS we are now able to describe the NanZam system in full.

Chapter 5

NanZam System Description

5.1 Overview

The Nanzam system combines the concepts described in previous sections to solve the *binary karyotyping problem*. In particular it borrows the two-stage model from geometric hashing and uses the techniques of empirical Bayes beam search in the online step to ensure that false discovery rates are minimized.

Nanzam works by constructing a look-up table of hashed strings that can then be rapidly checked in an online setting. The offline step consists of building a hash table.

5.2 Offline Step

The Nanzam system presumes the existence of a collection R of *reference genomes* which are finite sequences $\sigma_1, \ldots \sigma_{|R|}$ over the alphabet Σ_{DNA}. These sequences can be obtained by any of the sequencing methods discussed in Chapter 2. Because we are concerned with karyotyping we also assume that the sequences are *labeled*. Thus a reference *sequence* is such a labeled pair $\langle L_i, \sigma_i \rangle$ where L_i serves as a marker for the source of σ_i. For example L_i could simply be the natural numbers indicating a sample identifier or a more informative string such as "chromosome 8", or "human chromosome 1". For the karyotyping problem them labels serve to inform not simply *that* a translocation has occured but *which*.

From the collection $R = \{\langle L_i, \sigma_i \rangle\}_{i=1}^{|R|}$ we generate *in-silico* a set M_R of *reference nanomaps*.[1] These reference maps represent what we anticipate the nanomapping process would produce if these sequences were mapped *in vitro*.

To simulate the mapping process we first pick a short *alu* sequence to represent the target of the CRISPR-cas9 enzyme. The goal is to find the occurence of this *alu* in each σ_i and compute the distance between these sites. Because the CRISPR enzyme is mostly precise but not perfect we allow for some mismatches. Assume the *alu* has length A (measured in base pairs) then for a fixed parameter $0 \leq p < A$ we generate finite state automata (FSA) to perform regular expression matching of the reference sequence σ_i allowing up to p consecutive mismatches.

Moving left to right along each σ_i we check each subsequence $\sigma_i[\ell \ldots \ell + A]$ against these automata in increasing order of the *number* of mismatches. This procedure treats all FSA with the same number of mismatches — e.g FSA accepting T**CAT and TAGC**T — as equivalent. If any of the automata exits succesfully a match is declared and the procedure resumes at the position $\ell + A + 1$ in σ_i, otherwise the matching starts over at position $\ell + 1$. This is summarized in Algorithm 4.

The collection M_R of reference maps becomes the foundation from which we will construct a hash table that will be used in the online phase. Since each length is unlikely to occur multiple times, the lengths in the reference maps are *quantized* before they are hashed.

[1] In principle we could also simply work from a set of reference maps. However, no such sets exist at present so simulations will suffice. we assume that in time real maps will obviate the need to simulate.

Specifically they are binned and then translated to a *new alphabet* where lengths are mapped to characters their relative size.

To perform this translation we divide the range of lengths into η equal-width bins where η is a positive integer passed as a parameter to the system. Each bin is then assigned a letter from an alphabet $\hat{\Sigma}$. We then translate each

$$a^{(i)} = \left\langle a_1^{(i)}, \ldots, a_{k(i)}^{(i)} \right\rangle$$

by mapping each $a_j^{(i)}$ to the element of $\hat{\Sigma}$ representing the range where it falls. This procedure creates a *translated vector* $\tau^{(i)} = t(a^{(i)})$ where $t(\cdot)$ represents the translation map taking numeric vectors to strings in $\hat{\Sigma}$.

We then construct a hash table HT from $\tilde{M}_R$ using the translated vector $\tau^{(i)} \in \tilde{M}_R$. For a window size — also called a *mer length* — k (i.e: for each sequential subsequence of length k in $\tau^{(i)}$) we compute $m = h_{RK}(\tau^{(i)}[j..j+m], p, q)$ and store $\tau^{(i)}[j..j+m], L_i$, the index j, and other metadata in cell m of HT.

With HT constructed we can use it to rapidly karyotype patients by checking their translated maps against the contents of HT. This task is accomplished in the *online* step.

Input: A collection R of reference genomes
Input: A target *alu* with length A
Input: A tolerance p; $0 \leq p < A$
Output: A set M_R of labeled reference maps.
1 **for** *each* $\langle L_i, \sigma_i \rangle \in R$ **do**
2 CutSites $\leftarrow$ LocationsOfMatch (σ, t, alu).;
3 Insert 0 at head of cutsites and $|\sigma_i|$ at the end.;
4 **for** $j = 1..|CutSites| - 1$ **do**
5 Lengths.push(cutSites[j+1]-cutsites[j];
6 **end**
7 M_R.push$(L_i, lengths, cutsites)$;
8 **end**

Algorithm 4: The *in-silico* Mapping Process

5.3 Online Step

The online step uses the hash table to compare a map of a new DNA sample to the reference maps and starting from a new patient map $\tilde{m}$ we obtain a translated length vector $\tilde{\tau} = t(\tilde{m})$ in the same manner as above. Then we hash each successive k-mer of $\tilde{\tau}$ into the table and the contents of the hashed cell are returned.

The process of constructing the final sequence is performed via the empirical Bayes algorithm described in Chapter 4. When $\tilde{\tau}[0:w]$ is hashed we have an issue of selecting the start state s_0. This task is performed with a heuristic process: we pick the most frequent label $\tilde{\ell}$ in the list and select as s_0 the node with the lowest position number that has label $\tilde{\ell}$.

We then hash s_0 and return the contents of the hashed-to cell. We then evaluate the each of the possible paths and keep the top scoring w of them. This step is repeated until there are no more k-mers to hash.

For the current implementation of NanZam the score function is multiplicative and nonincreasing. All the initial paths are given a score of 1. Successor states that match the label of the current node,are one point ahead, and don't add to the parenthetic structure representing the sequence have their scores left unchanged. Violators of these heuristic rules have their scores multiplied by $0.9, 0.1$, and 0.01, respectively.

This procedure is summarized in Algorithm 5. The call to FDRSort in Line 18 is equivalent to Line 14 in Algorithm 3.

The current implementation of NanZam also includes some additional data structures to track the final map returned as well as the parenthetic structure, introduced in Chapter 2, that represents that map. An example of the metadata returned by NanZam is in Listing 5.1.

Input: Translated patient map $\tilde{\tau} = t(\tilde{m})$.
Input: A beam width w or a function to handle decaying beam width.
Input: A fdr frequence f
Input: A Hash Table HT
Input: A mer length m
Output: w candidate sequences
Output: w parenthetic structures and fdr probabilities

Initialize w empty stacks
Initialize empty beam B
$L_0 \leftarrow HT[h_{RK}(\tilde{\tau}[0...m]]$
$s_0 \leftarrow$ SelectStartState(L_0)
$B.insert(s_0)$
for $i = 1...L - m$ **do**
    $L_i \leftarrow HT[h_{RK}(\tilde{\tau}([i...i + w])]]$
    Candidates $\leftarrow \emptyset$
    for *each* $\sigma \in$ *beam* **do**
        for *each state* $s \in L_i$ **do**
            $\gamma \leftarrow$ Score($sigma \oplus s$)
            $\xi \leftarrow$ StackProcess($\sigma \oplus s$)
            Candidates.$push$($[\sigma \oplus s, \gamma, \xi]$)
        end
    end
    if $i \mod f == 0$ **then**
        $B \leftarrow$ FDRSort($Candidates$);
    end
    else
        $B \leftarrow$ SortByScore($Candidates$);
        $B \leftarrow B[1...w]$;
    end
end
return B;

Algorithm 5: Online Nanzam with EBBS

5.4 The Full Karyotyping System

The above sections characterized NanZam in terms of how it could analyze a single DNA nanomap. The reader may have noticed that we have characterized NanZam as a system for rapid karyotyping which is a broader claim than the system described above can handle. This gap can be remedied easily. We suppose that we are given a full reference genome which has been nanomapped and stored in a hash table. If we are given all the DNA from some new sample we can pass it all through NanZam keeping track of the counts of the labels returned. If these counts match the reference haplotype then we can call the organism wild type. If not NanZam will give the excess or deficient counts — reflecting aneuploidy — or will list the translocations that have occurred. In Chapter 6 we show that the run time for NanZam[2] is such that this step can be performed rapidly, on the order of one second per nanomap.

We now turn to performing experiments on the NanZam system.

`Final Map:`

[2]performed on a 2018 Macbook Pro with processor with a 2.7 GHz Intel Core i7 processor

```
[('chr8', 70, 'SMSS'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS')
    , ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, '
    SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh',
    0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh
    ', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('
    igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'),
     ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL
    '), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, '
    MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh',
    1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh
    ', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('
    igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'),
     ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS
    '), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, '
    SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh',
    0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh
    ', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('
    igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'),
     ('chr8', 71, 'MSSS'), ('igh', 3, 'SLSS'), ('igh', 0, 'SMSS'), ('igh', 1, '
    MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh',
    1, 'MSSL'), ('chr8', 71, 'MSSS'), ('igh', 3, 'SLSS'), ('igh', 0, 'SMSS'), ('
    igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'),
     ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS
    '), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, '
    SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh',
    0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh
    ', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('
    igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'),
     ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL
    '), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, '
    MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh',
    1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh
    ', 1, 'MSSL'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('igh', 10, 'SMSS'),
    ('igh', 11, 'MSSS'), ('chr8', 12, 'SSSS'), ('igh', 0, 'SMSS'), ('chr8', 71, '
    MSSS'), ('igh', 3, 'SLSS'), ('igh', 0, 'SMSS'), ('igh', 1, 'MSSL'), ('chr8',
    70, 'SMSS'), ('chr8', 70, 'SMSS')]


Parenthetic Structure of Final Map:

    (_chr8 (_igh igh_) (_igh igh_) (_igh igh_) (_igh igh_) (_igh igh_) chr8_)
```

Listing 5.1: Sample Output of the NanZam System

Chapter 6

Experiments

To test NanZam's performance and assess its dependency on hyperparameters we simulated a population of 200 patients. In a clinical context these patients can be viewed as possibly having a particular cancer called *Burkitt's Lymphoma* which is characterized by chromosomal translocations. We focused on the most common translocation and evaluated NanZam's ability to detect this translocation.

6.1 Simulating Burkitt's Lymphoma Translocation

Burkitt's Lymphoma is a highly aggressive cancer that most commonly affects children. Though it is the fastest growing human tumor the cure rate is around 90% in more developed countries.[37]. Prognosis in early adult cases and in less developed areas is worse. The disease is characterized by translocations leading to the over-expression of the oncogene *myc*. There is one common type of translocation as well as at least two rare variants:

- A t(8:14) translocation where parts of the IGH heavy locus on chromosome 14 translocates to promote *myc* on chromosome 8. This occurs in 85% of Burkitt's lymphoma cases [28].
- A t(2:8) translocation [44].
- A t(8:22) translocation [44].

To perform a preliminary evaluation of NanZam we focused on the first type of translocation and simulated a population of 200 patients. For this purpose we collected reference DNA sequences corresponding to a 1.8Mbp region containing *myc* from chromosome 8[1] and a sequence for the IGH locus on chromosome 14.[2] Then for each simulated patient we gave them zero, one, or two t(8:14) translocations with probability 90%, 6%, and 4%, respectively. The reason for the probability being weighted towards 0 translocations is that, while the first translocation mentioend above is common among Burkitt's Lymphoma patients, Burkitt's Lymphoma itself is relatively rare, affecting only around 2.3% of patients with lymphoma [46]. Thus in a population of patients with various blood cancers, this cancer and this translocation are likely to be relatively rare.

With the number of translocations selected, we chose breakpoints $j_1, \ldots, j_k$ uniformly at random in the sequence from chromosome 8. For each breakpoint j_r we randomly selected a position $i(j_r)$ and length $\ell(j_r)$ from chromosome 14 and created the sequence

$$chr8[1..j_1] \oplus chr14[i(j_1)..i(j_1)+\ell(j_1)] \cdots \oplus chr14[i(j_k)..i(j_k)+\ell(j_k)] \oplus chr8[j_k+1..|chr8|] \in {\Sigma^*}_{\text{DNA}}$$

[1] NC 000008.11:127700000-129800000 Homo sapiens chromosome 8, GRCh38.p7 Primary Assembly.
[2] NC 000014.9:105586437-106879844 Homo sapiens chromosome 14, GRCh38.p7 Primary Assembly

Total Patients	Wild Type	Mutants
200	181	19

Table 6.1: Breakdown of Simulated Patient Population

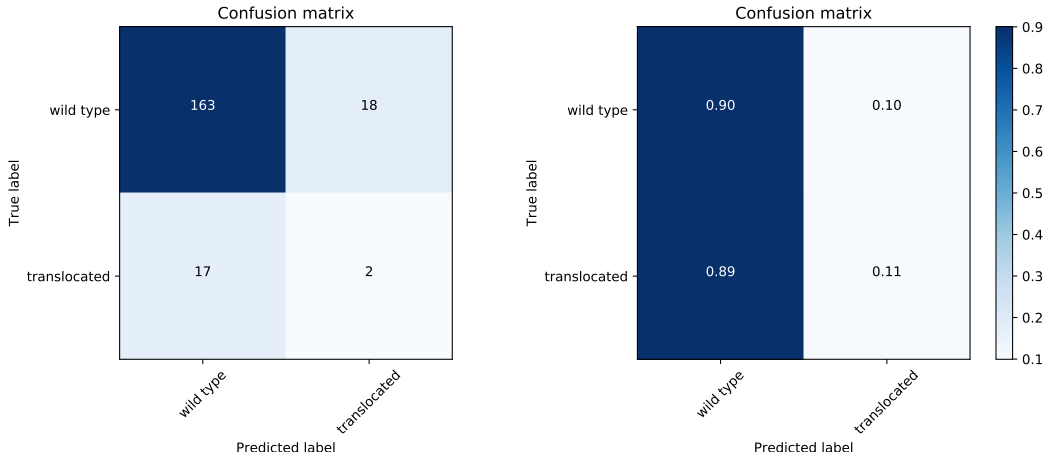

Figure 6.1: Confusion Matrices for NanZam Performance Raw (Left) and Normalized (Right)

With a collection of 200 patient sequences we we then simulated the actual nanomapping process for each of them. We assumed that the targeted *alu* was `TGTAATCCCAGCACTTTGGGAGG`[3] and simulated these patients in a manner very similar to the *in-silco* process described in Chapter 5. As per the specification in Chapter 5 we also created a set of reference maps by doing an *in-silico* simulation of the mapping process on the sequences in chromosome 8 and chromosome 14 that make up the patients.

We then stored the patient maps to use in our evaluation of NanZam. This is realistic as NanZam assumes that there exist reference maps and that the input to the system is a new series of lengths produced by the mapping process. The breakdown of the simulated patients is in Table 6.1.

6.1.1 NanZam Settings

For an initial evaluation of NanZam we ran the offline phase with an alphabet size of 8, a mer length and beam width of 5, and an FDR frequency of 20.

In the last iteration of the algorithm we returned a beam of width 5 and called wild type or mutant using the following procedure. We transformed the scores to z-scores as in Chapter 4 and then computed normal distribution p-values for each sequence. If any of the sequences had a p-value less than 0.05 we called a mutant.

If all the sequences had the same score then we called wild-type vs. mutant by sampling from a strong prior distribution:

$$\Pr\{\text{wild type}\} = 0.98$$

6.2 Preliminary Evaluation

On the simulated cohort of 200 patients we evaluated the system in terms of type I and type II error as well as accuracy. Confusion matrices, both raw and normalized, are shown in Figure 6.1.

[3]This is due to communications with our collaborators at VCU.

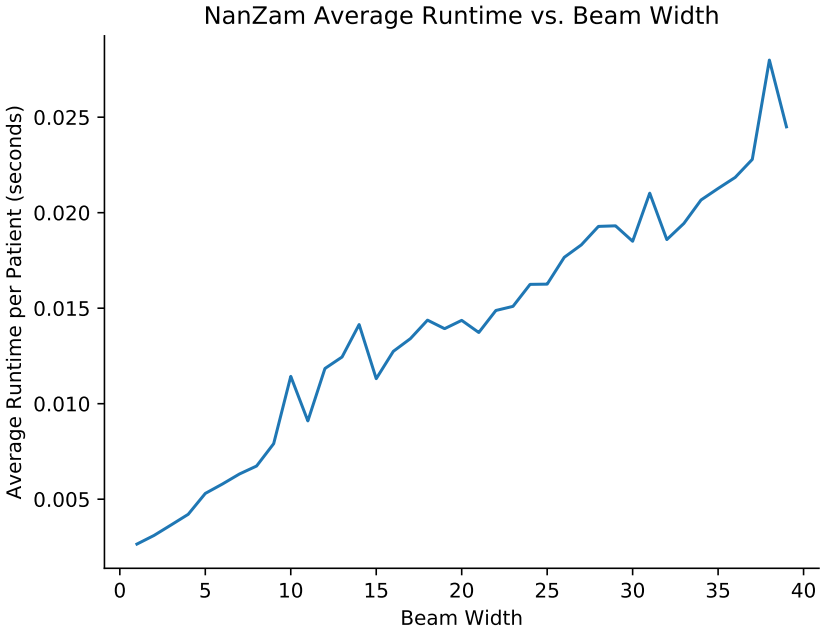

Figure 6.2: Linear Depedenence of NanZam Run Time on Beam Width

These results are fairly promising, though not yet ideal. In the language of statistics the current NanZam system is highly specific but not sensitive. The high specificity is a positive attribute for the system. Since the goal of the NanZam system is to find infrequent translocations the high true negative rate is an advantage. It allows the true wild types to be ignored, freeing research and clinical efforts to be spent verifying positive mutant calls.

Low sensitivity is an issue for the system that needs to be remedied. Missing out on true positives can have negative consequences in both research and clinical practice.

We also remark that the run time is remarkably fast for a biological technology. In [36] the authors state that microscopy system underlying the Nanomapping system can generate two 16-bit, 1000×1000 pixel images per second. Assuming the algorithm converting these images to the lengths required by NanZam runs in a similar time frame then the throughput of the NanZam system could be quite high.

6.3 Hyperparameter Dependencies

Since the results of running NanZam in the previous section were encouraging we investigated ways in which the hyperparameters of the system effect system performance. We focused on two aspects of the NanZam system: run time and accuracy. In these experiments we selected candidates for the beam based on the false discovery rate.

The NanZam system operated quickly, with average run time never breaking more than several tenths of a second. As shown in Figure 6.2 the run time of NanZam scales linearly in the beam width. This statistic suggests that the run time of NanZam is dominated by beam search. However, it is likely that increasing the frequency of the FDR control process would decrease run time. It may still be linear in the beam width but this is an open question at the present time.

Subsequent experiments explored how the system's accuracy and true positive rate depended on the size of the quantization alphabet and the mer length that was used. For alphabet sizes ranging from 2 to 27 characters and mer lengths ranging from 2 to 12 we ran NanZam with a beam width of 5, a mer length of 5, and a The results are shown in Figures 6.3 and 6.4.

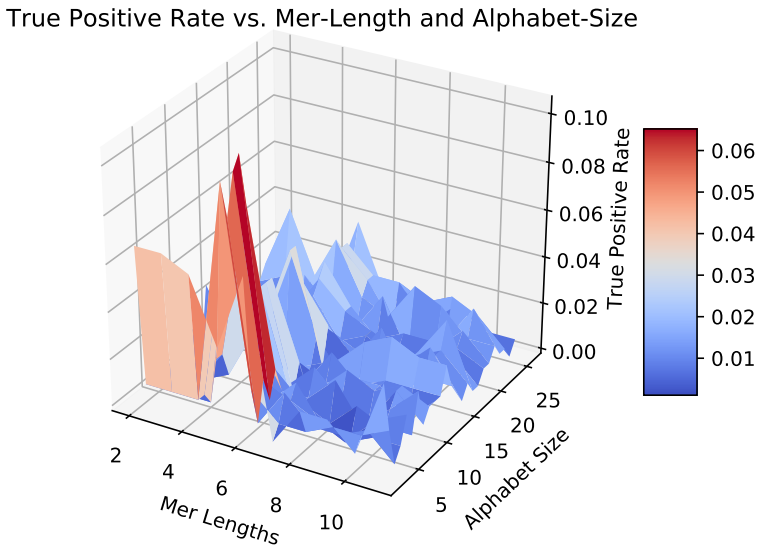

Figure 6.3: Phase Transition of True Positive Rate

These figures are show a possible trade-off in the system design. In the regime where the accuracy decreases the true positive rate increases. This result suggests that there is a tension between biasing the model toward wild type and ignoring true positive cases with potentially weak evidence.

This tension is likely an intrinsic challenge to solving the rapid karyotyping problem as it is reflected in the confusion matrices as well as the surface plots. It potentially can be mitigated through adjustments to the scoring function or making EBBS into a complete algorithm via backtracking.

We can also comment briefly on Listing 5.1. Here the parenthetic structure is correct given the map but the final map is biologically extremely implausible and should never have made it to the final beam. This conflict suggests that the score function needs to be altered or the post-processing step needs to add heuristics to prevent these sorts of obvious errors.

These experiments suggest that the system is viable and with future research could be made more versatile and provably accurate.

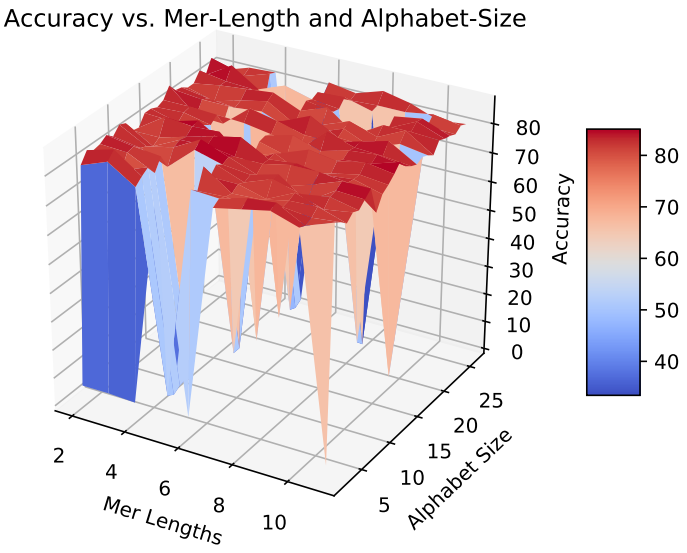

Figure 6.4: Phase Transition of System Accuracy

Chapter 7

Conclusion

7.1 Summary

This thesis presented NanZam, a new hashing scheme that builds on top of a novel technology for solving the *binary karyotyping problem*. We showed how the computational complexity of the problem can be reduced by using DNA nanomaps instead of DNA sequences, demonstrated the efficacy of the hashing scheme in a simulated environment, and explored the dependence of the system's performance on hyper-parameter values . Along the way we provided a new way of representing chromosomal translocations and presented a new variety of beam search based on false discovery rate control. The results of Chapter 6 suggest that with increased tuning the system could be both fast and accurate. We believe that there are several ways to build new research upon NanZam and we review them now.

7.2 Directions for Future Research

There are two umbrella categories for possible future research: *practical experiments* which focus on data analysis and possible ways of modifying the NanZam system, and *theoretical directions* which explore possible ways of establishing performance guarantees for the system.

7.2.1 Practical Experiments

Hyperparameter Tuning

The NanZam system as described has many hyperparameters, many of which were left untouched in the experiments of Chapter 6. A natural first direction would be to expand the ranges and kinds of hyperparameters tuned to address particular system attributes. Some particularly interesting candidates are the J parameter used in Lindsey's method, the method of computing the false discovery rate, as well as the alphabet size.

As mentioned in the discussion of Lindsey's method in Chapter 4, letting the parameter J range over various possible values makes our estimate more and less parametric (J large and small, respectively). In the same chapter we also discussed ways of fitting an *empirical* null and prior from the data. Changing our estimation scheme to being purely empirical and arbitrarily nonparametric would represent an interesting change, though it would run the risk of overfitting with small beam widths.

We experimented with the alphabet size in Chapter 6 but this exploration was cursory. It would be interesting to see if the performance landscape was different with a larger amount of data available, e.g. reference sequences from the whole human genome.

Alternate Scoring Functions

At present the scoring function for NanZam incorporates information about the *size* of the nested parentheses. The scoring function could be made more nuanced such that it favored

sequences that closed open parentheses or continue the previously. We also could alter the scoring function to be a user input or to be problem specific.

Adding User Input

In both clinical and research contexts the experimenters may have more information than the NanZam system presumes. As discussed in Chapter 6, Burkitt's lymphoma patients have a particular set of translocations that are common. NanZam could be altered to allow the user to specify this kind of information. This additional information could determine, in part, the system's behavior, e.g: by seeding certain start states, altering the score function, or informing post-processing decisions. This additional feature would allow researchers to input specific hypotheses into NanZam which could increase the system's versatility and accuracy.

More Nuanced Simulation

The NanZam system as described above is based on an ideal model of data. We do not model the two key kinds of errors that are possible in the nanomapping process: *binding errors* and *sizing errors*. Two types of binding errors are possible: *false positive* errors occur when the Cas9 enzyme binds off target and *false negative* errors occur when the enzyme fails to bind to the correct target. With suprious or missing binding sites the quantization may introduce an error which in turn result in a hashing error. This error can be mitigated by returning more than one cell from the hash table, however this would require additional metadata to be stored in the hash table and could possibly degrade performance.

Sizing errors occur when the nanomapping system incorrectly reports the length between binding sites. This error is usually modeled as standard Gaussian noise added to the lengths reported. Sizing error can also lead to quantization error which can be handled in the same way as hashing error.

Solving Additional Problems

Our implementation of the NanZam system was tuned to specifically solve the binary karyotyping problem. However, NanZam returns the parenthetic structures of the final maps which means that it could be used to solve what might be called the *general karyotyping problem:*

General Karyotyping Problem (GKP): Given a sample of DNA from a person is their karyotype wild type or mutant? If it is mutant, what translocation occured?

This problem could be tested on the same data set here and measured for accuracy in terms of edit distance between the parenthetic structure. Since multiple patients are unlikely to have precisely the same translocation the parenthetic structure provides a lower-resolution compromise that could be used to target more high-resolution investigations, e.g. via sequencing. Moving beyond problems based on the NanZam system itself additional research could explore applications parenthetic structures in other contexts.

7.2.2 Theoretical Directions

Substantial work could be done in terms of justifying the theoretical properties of the system. These efforts could be based on the system as described in Chapters 5 and 6 or one with some of the modifications discussed in Section 7.2.1. There are a few possible directions to take this work.

The first direction involves establishing bounds on error probabilities. For the technological predecessor to nanomapping, the authors in [3, 2, 1, 5] demonstrate that there exists a phase transition in the number of maps after which the probability of exact recovery is nearly 1. In the case of NanZam such an error bound would be useful and could also inform design modifications to the system. If we assume that there exist a correct sequence σ — a ground truth — then the errors in NanZam as specified in this thesis can be attributed to two sources active at any given iteration during the online phase:

1. Hashing Error: The correct element in the optimal sequence is not in the hashed-to cell in the hash table.

2. Selection Error: The correct element in the optimal sequence is not chosen from the available options.

The first error source can be mitigated in two ways: making sure no cell of the hash table is empty, or returning more than one cell per hash. The approach that ensures a full hash table prevents any sort of randomness in the candidate sequences returned from the hashing step of NanZam. A theoretical approach to this problem would be to derive an expression for the probability that the hash table contains an empty cell in terms of the size of the alphabet, the length of the k-mers with which we are hashing, and the amount of data available. This last factor will likely be the most important because with sufficient data we should expect to see a diverse selection from the universe of strings over our quantization alphabet $\hat{\Sigma}$ and so the probability of an empty cell should go to zero.

Selection error is also interesting, and relates to a growing body of literature on limiting false discovery rates in online settings. This literature mostly focuses on false discovery rates in the sense of [9], not that of [22, 19]. It provides a series of techniques for adaptively or reactively controlling FDRs, which could be adapted to refining the EBBS procedure from Chapter 4. While the score function is currently a heuristic score function it might be interesting to see if certain restrictions on this function's behavior — e.g. non-decreasing/non-increasing, differentiable — affect the system's performance as well as the efficacy of the FDR control procedure.

7.3 Final Word

Nanomapping represents an exciting technological development that provides long-range genetic information that is either difficult or impossible to obtain from DNA sequencing methods. NanZam is a first attempt at building an system on top of the nanomapping technology to extract useful information. Continued research effort could make the system provably robust and could enable a widely applicable biotechnology.

Bibliography

[1] T. Anantharaman, B. Mishra, and D. Schwartz. "Genomics via optical mapping. III: Contiging genomic DNA." English (US). In: *Proceedings / . International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology* (1999), pp. 18–27. ISSN: 1553-0833.

[2] THOMAS S. ANANTHARAMAN, BUD MISHRA, and DAVID C. SCHWARTZ. "Genomics via Optical Mapping II: Ordered Restriction Maps". In: *Journal of Computational Biology* 4.2 (1997). PMID: 9228610, pp. 91–118. DOI: 10.1089/cmb.1997.4.91. eprint: https://doi.org/10.1089/cmb.1997.4.91. URL: https://doi.org/10.1089/cmb.1997.4.91.

[3] Thomas Anantharaman and Bud (Bhubaneswar) Mishra. "Genomics via Optical Mapping (I): 0-1 Laws for Mapping with Single Molecules". In: *bioRxiv* (2013). DOI: 10.1101/000844. eprint: https://www.biorxiv.org/content/early/2013/11/22/000844.full.pdf. URL: https://www.biorxiv.org/content/early/2013/11/22/000844.

[4] Alexandr Andoni and Piotr Indyk. "Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions". In: *Commun. ACM* 51.1 (Jan. 2008), pp. 117–122. ISSN: 0001-0782. DOI: 10.1145/1327452.1327494. URL: http://doi.acm.org/10.1145/1327452.1327494.

[5] Marco Antoniotti et al. *Genomics via Optical Mapping IV: Sequence Validation via Optical Map Matching*. Tech. rep. New York, NY, USA, 2001.

[6] Mark A Batzer and Prescott L Deininger. "Alu repeats and human genomic diversity". In: *Nature reviews genetics* 3.5 (2002), p. 370.

[7] Philip N Benfey. *Quickstart Molecular Biology: An Introductory Course for Mathematicians, Physicists, and Computational Scientists*. Cold Spring Harbor Laboratory Press, 2014.

[8] Yoav Benjamini and Yosef Hochberg. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the royal statistical society. Series B (Methodological)* (1995), pp. 289–300.

[9] Yoav Benjamini and Yosef Hochberg. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the royal statistical society. Series B (Methodological)* (1995), pp. 289–300.

[10] Trevor Hastie Bradley Efron. *Computer Age Statistical Inference: Algorithms, Inference, and Data Science*. 2016.

[11] Roy J Britten. "DNA sequence insertion and evolutionary variation in gene regulation". In: *Proceedings of the National Academy of Sciences* 93.18 (1996), pp. 9374–9377.

[12] Moses S. Charikar. "Similarity Estimation Techniques from Rounding Algorithms". In: *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*. STOC '02. Montreal, Quebec, Canada: ACM, 2002, pp. 380–388. ISBN: 1-58113-495-9. DOI: 10.1145/509907.509965. URL: http://doi.acm.org/10.1145/509907.509965.

[13] Richard Cordaux and Mark A Batzer. "The impact of retrotransposons on human genome evolution". In: *Nature Reviews Genetics* 10.10 (2009), p. 691.

[14] Prescott Deininger. "Alu elements: know the SINEs". In: *Genome biology* 12.12 (2011), p. 236.

[15] Richard Durbin et al. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[16] Bradley Efron. *Bayes, Oracle Bayes, and Empirical Bayes*. 2017.

[17] Bradley Efron et al. "Doing thousands of hypothesis tests at the same time". In: *Metron-International Journal of Statistics* 65.1 (2007), pp. 3–21.

[18] Bradley Efron. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. 2010.

[19] Bradley Efron. *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*. Vol. 1. Cambridge University Press, 2012.

[20] Bradley Efron. "Microarrays, empirical Bayes and the two-groups model". In: *Statistical science* (2008), pp. 1–22.

[21] Bradley Efron et al. "Simultaneous inference: When should hypothesis testing problems be combined?" In: *The annals of applied statistics* 2.1 (2008), pp. 197–223.

[22] Bradley Efron and Trevor Hastie. *Computer age statistical inference*. Vol. 5. Cambridge University Press, 2016.

[23] Bradley Efron and Nancy R Zhang. "False discovery rates and copy number variation". In: *Biometrika* 98.2 (2011), pp. 251–271.

[24] Markus Freitag and Yaser Al-Onaizan. "Beam search strategies for neural machine translation". In: *arXiv preprint arXiv:1702.01806* (2017).

[25] David Furcy and Sven Koenig. "Limited discrepancy beam search". In: *IJCAI*. 2005, pp. 125–131.

[26] Ronald L Graham et al. "Concrete mathematics: a foundation for computer science". In: *Computers in Physics* 3.5 (1989), pp. 106–107.

[27] Mikael Häggström. *"Medical gallery of Mikael Häggström 2014". WikiJournal of Medicine 1 (2)*. 2014. DOI: 10.15347/wjm/2014.008.

[28] Ronald Hoffman et al. *Hematology: basic principles and practice*. Elsevier Health Sciences, 2013.

[29] Dan Jurafsky and James H Martin. *Speech and language processing*. Vol. 3. Pearson London, 2014.

[30] Alan G Konheim. *Hashing in computer science: Fifty years of slicing and dicing*. John Wiley & Sons, 2010.

[31] Yehezkel Lamdan and Haim J Wolfson. "Geometric hashing: A general and efficient model-based recognition scheme". In: (1988).

[32] S Lander Eric et al. "Initial sequencing and analysis of the human genome". In: (2001).

[33] Nathan Linial and Ori Sasson. "Non-Expansive Hashing". In: *Combinatorica* 18.1 (Jan. 1998), pp. 121–132. ISSN: 1439-6912. DOI: 10.1007/PL00009805. URL: https://doi.org/10.1007/PL00009805.

[34] Barbara McClintock. "The origin and behavior of mutable loci in maize". In: *Proceedings of the National Academy of Sciences* 36.6 (1950), pp. 344–355.

[35] Fabian Menges, Giuseppe Narzisi, and Bud Mishra. "Totalrecaller: improved accuracy and performance via integrated alignment and base-calling". In: *Bioinformatics* 27.17 (2011), pp. 2330–2337.

[36] Andrey Mikheikin et al. "DNA nanomapping using CRISPR-Cas9 as a programmable nanoparticle". In: *Nature communications* 8.1 (2017), p. 1665.

[37] Elizabeth M Molyneux et al. "Burkitt's lymphoma". In: *The Lancet* 379.9822 (2012), pp. 1234–1244.

[38] Giuseppe Narzisi and Bud Mishra. "Scoring-and-unfolding trimmed tree assembler: concepts, constructs and comparisons". In: *Bioinformatics* 27.2 (2010), pp. 153–160.

[39] Susumu Ohno. *Evolution by Gene Duplication*. 1970.

[40] Alexander Rush, Yin-Wen Chang, and Michael Collins. "Optimal beam search for machine translation". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 210–221.

[41] Carl W Schmid and Prescott L Deininger. "Sequence organization of the human genome". In: *Cell* 6.3 (1975), pp. 345–358.

[42] Stuart C. Shapiro. *Encyclopedia of Artificial Intelligence*. 2nd. New York, NY, USA: John Wiley & Sons, Inc., 1992. ISBN: 0471503053.

[43] Fang-miin Sheen et al. "Reading between the LINEs: human genomic variation induced by LINE-1 retrotransposition". In: *Genome research* 10.10 (2000), pp. 1496–1508.

[44] Jana Smardova et al. "An unusual p53 mutation detected in Burkitt's lymphoma: 30 bp duplication". In: *Oncology reports* 20.4 (2008), pp. 773–778.

[45] Martin N Szmulewicz, Gabriel E Novick, and Rene J Herrera. "Effects of Alu insertions on gene function". In: *Electrophoresis* 19.8-9 (1998), pp. 1260–1264.

[46] Mary Louise Turgeon. *Clinical hematology: theory and procedures*. Lippincott Williams & Wilkins, 2005.

[47] Nikita S Vassetzky and Dmitri A Kramerov. "SINEBase: a database and tool for SINE analysis". In: *Nucleic acids research* 41.D1 (2012), pp. D83–D89.

[48] Andrew C Yao. "Uniform hashing is optimal". In: *Journal of the ACM (JACM)* 32.3 (1985), pp. 687–693.

[49] Weixiong Zhang. *State-space search: Algorithms, complexity, extensions, and applications*. Springer Science & Business Media, 1999.

[50] Rong Zhou and Eric A Hansen. "Beam-Stack Search: Integrating Backtracking with Beam Search." In: *ICAPS*. 2005, pp. 90–98.