DNA Sequencing by Hybridization

by

© Bradley Sheppard

A thesis submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Science

Department of Mathematics and Statistics

Memorial University of Newfoundland

August 8, 2014

St. John's Newfoundland & Labrador

Abstract

DNA Sequencing by Hybridization (SBH) is a method for reconstructing a DNA sequence based on its k-length subsequences. In this thesis we investigate several issues related to SBH. The set of all k-mers of a sequence is known as the k-spectrum. Using graph theory it is possible to reconstruct the unknown DNA sequence using only the information available in the k-spectrum, but unique reconstruction of the DNA sequence is not always possible. In this thesis we examine probabilistic models which determine the likelihood of a random DNA sequence of length N being uniquely reconstructable based on its k-spectrum. We will also discuss extensions of SBH using both additional information on the k-spectrum and restriction enzymes. The use of restriction enzymes in SBH is a next generation sequencing technique whereby the DNA sequence is split into fragments using restriction enzymes and sequencing is performed on the individual fragments rather than the sequence as a whole. We develop algorithms which use a library of restriction enzymes to cut the sequence

and perform sequencing. The width of DNA graphs is also important in the sense of computational complexity and we investigate the DAG-width of the graphs obtained from SBH. We show that the DAG-width of the these graphs is usually small, enabling polynomial time solvability of the Hamiltonian path problem, which is at the core of sequence reconstruction when the problem is modeled using graphs. In the final section, we discuss a next generation variant on SBH.

Table of Contents

List of Figures

Chapter 1

Introduction

Deoxyribonucleic acid (abbreviated DNA) is a molecule responsible for all of the phenotypes (observable traits) in any organism. Frances Crick and James Watson pioneered the research that led to the preliminary understanding of DNA [12]. DNA sequencing is the process of determining the precise ordering of the nucleotides within a molecule of DNA. Many sequencing methods have been developed, some of which are more biochemistry based than others that are more mathematically based. Some of these sequencing technologies include terminating chains, hybridization to arrays, and parallelized pyrosequencing. [23, 24]. Efficient sequencing techniques are important because they greatly aid in biological and medical research. One such example is the human genome project, which was a major international research project undertaken

to map the entire human genome, in which DNA sequencing techniques played a major role [9].

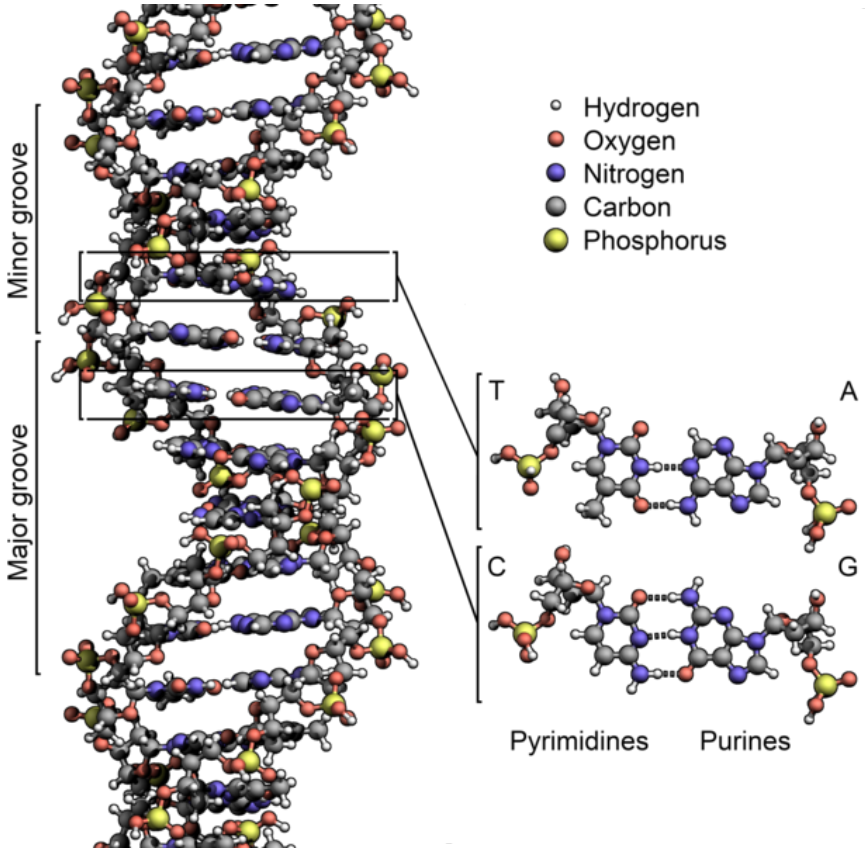


Figure 1.1: The structure of DNA [34].

Bioinformatics is an area of research focusing on understanding, analyzing and manipulating of large amounts of biological information, often including DNA. Bioinformatics lies at the crossroads of biology, computer science, mathematics and statistics. Developments in technology and mathematics has greatly enhanced the ability

for researchers to better understand biological data. Bioinformatics is a vast field of research with many subareas, some of which include sequence alignment, sequence assembly, genome annotation and protein structure prediction [22, 30].

Sequence assembly is the process of merging or putting together many small pieces of DNA together in order to determine the original, longer sequence. Sequence assembly often plays a role in DNA sequencing. Sequencing by hybridization is a DNA sequencing technique which uses sequence assembly. Sequencing by Hybridization (abbreviated SBH) was developed simultaneously by several researchers in the late 1980's [3, 19]. SBH relies on the use of graph theory concepts, such as Hamiltonian paths and Eulerian trails, to reconstruct a DNA sequence using the set of all k-length sub fragments (k-mers) of the DNA sequence in question. This set is known as the k-spectrum. Pevzner demonstrated that unique DNA reconstruction can be found in polynomial time [25]. To construct the k-spectrum, one uses an array of probes that contains all 4^k possible k-mers. A particular k-mer binds to the unknown DNA sequence if its Watson-Crick complementary sequence is contained within the unknown sequence. Despite being relatively simple, SBH has many drawbacks. One such drawback is that unique reconstruction of the DNA sequence is not always possible. Enhancement have been proposed to help overcome this difficulty such as obtaining location information of the subsequences [8] or using interactive sequenc-

ing techniques [27]. In Section 3 we introduce SBH and demonstrate the problem of non-unique reconstruction. We also introduce several variants of SBH that are often studied as well. Finally, we discuss the computational complexity of SBH.

Many researchers have examined the probability that a random DNA sequence of length N will be uniquely reconstructable from its k-spectrum. Asymptotic formulas for this probability were given by Dyer et al. and Arratia et al. [1,14]. In [2] Arratia et al. used combinatorial pairings to determine the probability of exactly K reconstructions of a random DNA sequence of length N. In Section 4 we will examine some of the probabilistic methods developed in [28] and further extend them to include modeling using conditional probability.

SBH is subject to errors when obtaining the DNA sequences k-spectrum. Many studies on SBH assume, for simplicity, that sequencing errors do not occur. Others, however, take into account the presence of both positive and negative sequencing errors. Positive errors refer to an event in the experiment when an element of the k-spectrum does not actually occur as a k-mer of the DNA sequence. Negative errors refer to the event when a particular k-mer of the DNA sequence does not occur in the k-spectrum. Because of these errors, SBH is often formulated in different ways depending on the context of the study. Many of these contexts are outlined in [15] by Formanowicz. In Section 5 we analyze SBH in terms of positive and negative

errors. We also put forward an enhancement involving the use of restriction enzymes to further increase the likelihood of a unique reconstruction. The method is then tested using computer simulations.

In Section 6 we will analyze the graphs obtained using SBH in the context of DAG-width. We will show that the graphs will often have a DAG-width of 1. This implies that we can often achieve polynomial time solvability of the Hamiltonian path problem, which is the problem we need to solve in order to reconstruct an unknown DNA sequence.

In the final section we will discuss a next generation approach to SBH. This approach involves splitting the DNA sequence randomly into fragments and performing sequencing on the individual fragments. The fragments can then be reassembled to obtain the target sequence.

Chapter 2

Basic Concepts in Molecular Biology

In this chapter we introduce some basic concepts in molecular biology that allows us formulation of problems in bioinformatics that will be discussed later. We introduce the concepts of DNA and RNA and their role in gene expression.

2.1 DNA and RNA

Genetics is the branch of molecular biology concerned with heredity and variation in organisms. One of the first studies of genetics was carried out by Gregor Mendel

in the 1800's. During this time, Mendel studied pea plants and noticed that certain traits of the pea plants were passed down to their offspring. He concluded that such traits were controlled by discrete units called *genes*. A *phenotype* of an organism is defined as any observable trait of that organism. For example, if a dog has black fur, then the black fur could be referred to as a phenotype of the dog. Alternative forms of a gene in a particular organism are referred to as *alleles*. For example, the gene which encodes hair color can have different forms. One form could code for blonde hair, while another could code for brown hair. All alleles associated with a particular trait are known as a *genotype*. Alleles are directly responsible for many of the phenotypes in organisms.

A *chromosome* is a structure in cells containing genes. Deoxyribonucleic acid, or DNA, is one of the main components in the chromosomes of organisms. In the mid 1900's it was shown through various experiments that DNA is the carrier of genetic information in organisms. The shape of DNA is often referred to as a double-helix, since it is composed of two strands that are shaped as a curled ladder. Contained in DNA are many small molecules called *nucleotides*. The four nucleotides present in DNA are adenine, cytosine, guanine and thymine, often abbreviated A, C, G, and T respectively. In the double strand, the corresponding nucleotides in each strand are complimentary to each other. The rule for complimentary nucleotides is that A is

complementary to T, and C is complementary to G [12]. Such nucleotides are referred to as Watson-Crick complementary.

Ribonucleic acid, or RNA is a substance which is very similar to DNA. The major differences are that it is single-stranded, and contains the nucleotide uracil, abbreviated U, instead of thymine [18].

2.2 The Central Dogma of Genetics

The Central Dogma of Genetics is the process whereby DNA is copied into RNA, and in turn, is used as a template to produce proteins. Proteins are the building blocks of organisms and are responsible for many of their biological functions.

The first phase of the Central Dogma is known as the transcription phase. During this phase, DNA is transformed into a strand of complementary RNA known as messenger RNA, or mRNA. This is done using an enzyme known as RNA polymerase. Once this has been achieved the mRNA binds to a ribosome. The next phase is the translation phase. In this phase a protein is synthesized based on the mRNA that binded to the ribosome. This is achieved through molecules inside the ribosome called transfer RNA or tRNA. Proteins are made up of long chains of amino acids. Each non-overlapping triple of nucleotides in an mRNA molecule codes for a specific amino

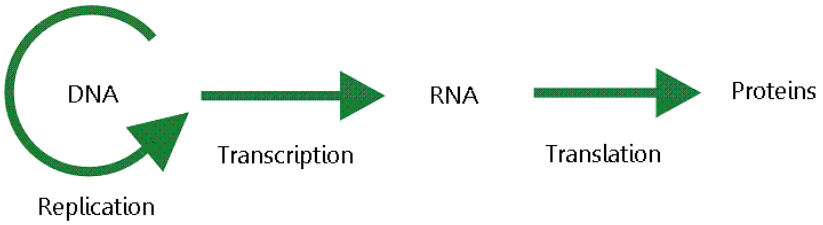


Figure 2.1: The Central Dogma of Genetics.

acid. These triples are known as *codons*. There are a total of twenty amino acids contained in organisms [11, 18].

Replication is another phase of The Central Dogma. This occurs frequently and is essentially to ensure genetic continuity between cells. When replication occurs, the double strand of DNA unzips and each strand acts as a base to form new strands. The new strands are generated using DNA polymerase, which adds base pairs to each strand that are Watson-Crick complementary to the nucleotide bases in each strand.

2.3 Mutations

The *genome* of an organism is all of the genetic material contained in that organism. Mutations are defined as a change in the nucleotide sequence contained within the genome of an organism. If such a change occurs, it could result in a different

amino acid sequence being generated, and possibly a different phenotype present in the organism. Mutations can often yield negative results in organisms. For example, the disease known as Sickle Cell Disease is the direct result of a mutation in the hemoglobin protein in blood. The resulting blood cells become sickle-shaped as oppose to the traditional round shape. This in turn can cause blood flow to be blocked to many of the bodies muscles, which can eventually lead to death [18]. Since mutations can have many negative side effects, many organisms have mechanisms which can repair these mutations [5]. Mutations can happen for a number of reasons. Mutations, in general, are classified into two categories.

The first type of mutations are known as *spontaneous mutations*. They are referred to as spontaneous because it is not a direct result of any outside agent or interference. Spontaneous mutations often happen during the replication of DNA.

The second type of mutations are known as *induced mutations*. These mutations are the direct result of some outside agent. Examples of such outside agents include radiation from the sun, chemicals, as well as X-rays.

Mutations also have classifications based on their effect on nucleotides. For example, a point mutation in a DNA fragment is when one nucleotide is substituted, or replaced, by another. Because a point mutation effects just one nucleotide, it can alter at most one amino acid in the DNA molecule. Insertion and deletion mutations

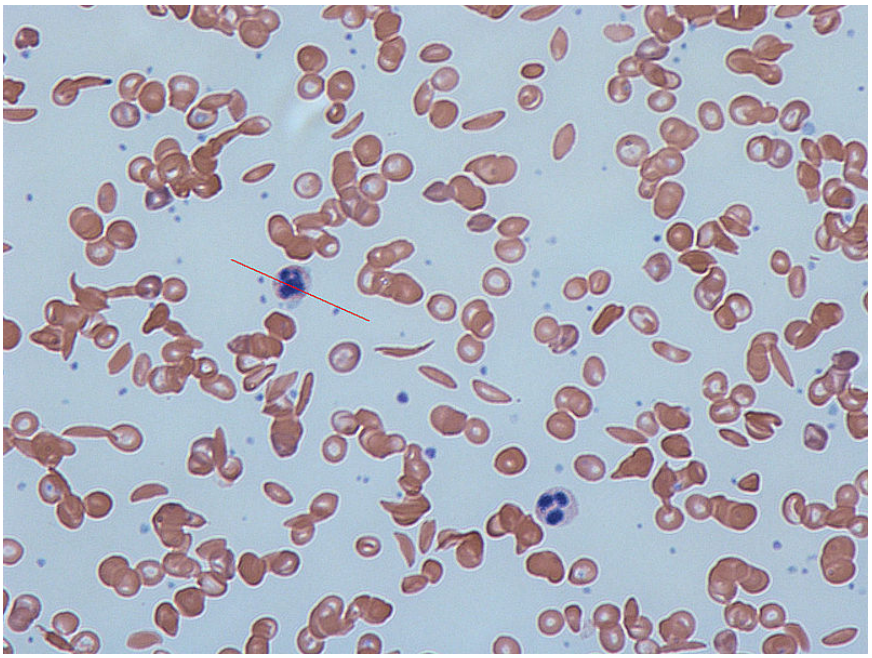


Figure 2.2: Sickle Cell Disease. A collection of both normal blood cells and sickle blood cells. Image taken from [4].

are when one or more nucleotides are inserted into, or deleted from DNA. Such a change results in a shift in the reading frame of the DNA and can have a significant impact on the amino acid sequence generated by the DNA [18].

Chapter 3

Sequencing by Hybridization

In this chapter we will introduce sequencing by hybridization (SBH). We will also introduce several concepts which are important in the study and understanding of SBH. Combinatorics plays an essential role in the reconstruction of the sequences, so we will review basic graph theory concepts which pertain to SBH. We will also introduce some basic notions in computational complexity which will help in the later discussion of the complexity of SBH.

3.1 Preliminaries

3.1.1 Concepts in Graph Theory

Combinatorics, and in particular graph theory plays an essential role in many bioinformatics problems; often seemingly complicated problems can be greatly simplified by using a graph structure to model data. In this section we will discuss many basic concepts in graph theory that will be used in later sections in DNA sequencing. For a more detailed list of graph theory concepts please refer to [33].

A graph is a pair $G = (V_G, E_G)$ where V_G is a set, elements of V_G are called *vertices* or *nodes*, E_G is a collection of unordered pairs of elements in V_G, and each pair in E_G is referred to as an *edge*. The vertices of each edge are referred to as the *endpoints* of the edge. For each edge $e = \{v_i, v_j\} \in E_G$ we say that e is an edge from vertex v_i to vertex v_j (or vice versa). A *walk* in a graph is a sequence $v_0 e_1 v_1 e_2 v_2 \cdots e_k v_k$ such that edge e_i is an edge from v_{i-1} to v_i. A *cycle* is a walk in which the starting and ending vertices are the same. A graph which possesses no cycles is called *acyclic*. A *trail* is a walk in which all edges are distinct and a *path* is a walk in which all vertices are distinct. A *Hamiltonian path* is a path which travels through each vertex exactly once, and an *Eulerian trail* is a trail which travels through each edge exactly once. Graphs which posses a Hamiltonian path are called *Hamiltonian* and graphs which

posses an Eulerian trail are called *Eulerian*.

Graphs are often represented visually to aid in the understanding of their structure. A graph $G = (V_G, E_G)$ can be represented visually by drawing a dot or a square to represent each vertex $v \in V_G$ and a line from v_i to v_j for each $\{v_i, v_j\} \in E_G$. As an example the graph $G = (V_G, E_G)$ where $V_G = \{1, 2, 3, 4, 5\}$ and $E_G = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 5\}, \{4, 5\}\}$ is given below.

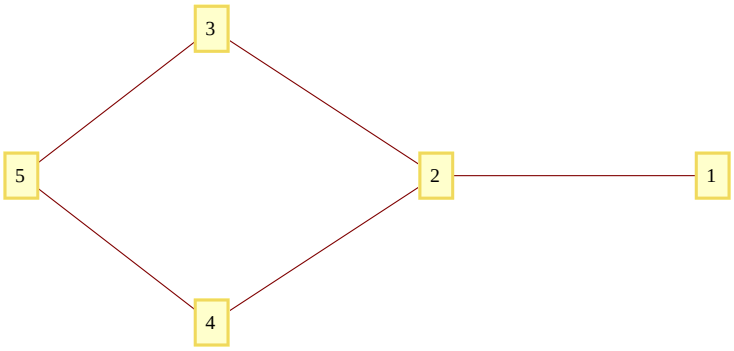


Figure 3.1: The graph G represented pictorially.

Example 1. In the graph G given above the sequence $1 \cdot \{1, 2\} \cdot 2 \cdot \{2, 3\} \cdot 3 \cdot \{3, 5\} \cdot 5$ is an example of a walk. This walk is also a path since each vertex occurs at most once. The path is not Hamiltonian because the vertex 4 is missing. It is also a trail since each edge occurs at most once. The trail is not Eulerian because the edge $\{4, 5\}$ is missing.

Example 2. The sequence $1 \cdot \{1, 2\} \cdot 2 \cdot \{5, 3\} \cdot 3 \cdot \{3, 5\} \cdot 5$ is not a walk in G because

the edge $\{5, 3\}$ occurs after vertex 2.

Example 3. The sequence $1 \cdot \{1, 2\} \cdot 2 \cdot \{2, 3\} \cdot 3 \cdot \{3, 5\} \cdot 5 \cdot \{5, 4\} \cdot 4$ is a Hamiltonian path in G. The sequence $1 \cdot \{1, 2\} \cdot 2 \cdot \{2, 3\} \cdot 3 \cdot \{3, 5\} \cdot 5 \cdot \{5, 4\} \cdot 4 \cdot \{4, 2\} \cdot 2$ is an Eulerian trail in G.

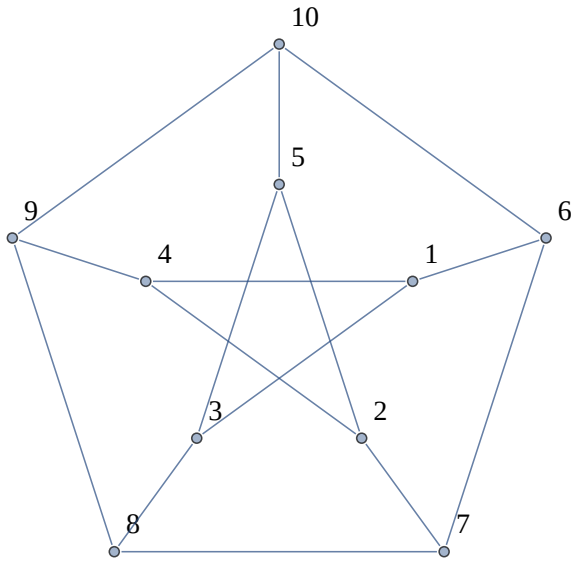


Figure 3.2: The Petersen Graph is a well known graph that has a Hamiltonian path but no Eulerian trail.

In the study of graph theory, researchers are also often interested in a concept known as directed graphs. A directed graph is a pair $D = (V_D, E_D)$ where V_D is a set and E_D is a collection of ordered pairs of elements of V called *arcs* or *directed edges*. For each edge $e = (v_i, v_j) \in E_D$ we say that e is an edge from v_i to v_j. We

also say that v_i is the source node (or vertex) and v_j is the target node. We represent directed graphs visually in the same manner as graphs except that we attach arrows to each edge which point to the target node. The concepts of walk, path, trail, etc. can be defined analogously for directed graphs. As an example the directed graph $D = (V_D, E_D)$ where $V_D = \{1, 2, 3, 4, 5\}$ and $E_D = \{(1, 2), (2, 3), (2, 4), (3, 5), (4, 5)\}$ is given below.

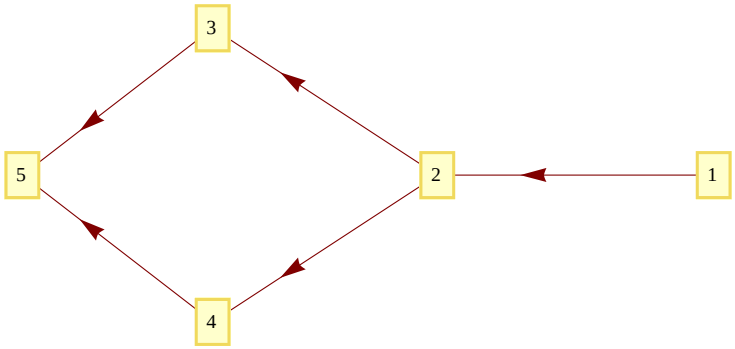


Figure 3.3: The directed graph D represented pictorially.

We will now turn our attention to graph decompositions. Many of the notions we discuss from this topic are based on [6]. Given a directed graph $G = (V, E)$ we define the partial order $\preceq_G$ as the reflexive, transitive closure of the relation given by E. Let $W, V' \subseteq V$, any two subsets W and V' of V. We say that W guards V' if for any $(u, v) \in E$ with $u \in V'$ we have that $v \in V' \cup W$. A *DAG-decomposition* of G is a pair $(D, \mathcal{X})$ where D is a DAG (directed acyclic graph) and $\mathcal{X} = (X_d)_{d \in V_D}$ is a

family of subsets of V_D such that:

1. $\bigcup_{d \in V_D} X_d = V$.

2. For all vertices $d \preceq_D d' \preceq_D d''$, not necessarily different, $X_d \cap X_{d''} \subseteq X_{d'}$.

3. For all edges $(d, d') \in E_D$, $X_d \cap X_{d'}$ guards $X_{\succeq d'} \setminus X_d$, where $X_{\succeq d'}$ stands for $\bigcup_{d' \preceq_D d''} X_{d''}$.

The width of a DAG-decomposition $(D, \mathcal{X})$ is defined as $\max\{|X_d| : d \in V_D\}$. The DAG-width of a directed graph G is the minimum width of any DAG-decomposition of G. It is well known that deciding if the DAG-width of G is at most k is NP-hard [6].

Often with many graph decompositions there are associated games that are directly related to the width of the graphs. One such game is known as the game of *cops and robbers*. Given a directed graph $G = (V_G, E_G)$ the k-cops and robber game on G is played between a cop player and a robber player. Each position in the game is denoted by (X, r) where $X \in [V]^{\leq k}$ are the vertices occupied by the cops and $r \in V$ is the vertex occupied by the robber. The game is carried out as follows:

- At the beginning, the cop player chooses $X_0 \in [V]^{\leq k}$ and the robber player chooses a vertex $r_0 \in V$ giving position (X_0, r_0).

- From position (X_i, r_i) if $r_i \notin X_i$ then the cop player chooses $X_{i+1} \in [V]^{\leq k}$ and

the robber player chooses a vertex r_{i+1} such that there is a directed path from r_i to r_{i+1} in the graph $G \setminus (X_i \cap X_{i+1})$.

- A play in the game is a maximal (finite or infinite) sequence $\pi := (X_0, r_0), (X_1, r_1), \ldots$ of positions given by the above rules.

- A play π is *winning* for the cop player if and only if it is finite. A play is winning for the robber player if and only if it is infinite.

- A (k-cop) strategy for the cop player is a function $f : [V]^{\leq k} \times V \to [V]^{\leq k}$. A play $(X_0, r_0), (X_1, r_1), \ldots$ is *consistent* with a strategy f if $X_{i+1} = f(X_i, r_i)$ for all i. The strategy f is called a *winning strategy* if every play consistent with f is winning for the cop player.

- The *cop number* of a digraph G is the least k such that the cop player has a strategy to win the k-cops and robber game on G.

We now introduce what it means for a particular strategy to be monotonic. Here we assume $G = (V_G, E_G)$ is a digraph.

1. A strategy for the cop player is *cop-monotone* if in playing the strategy, no vertex is visited twice by the cops. Specifically that is, if $(X_0, r_0), (X_1, r_1), \ldots$ is a play consistent with the strategy, then for every $0 \leq i \leq n$ and $v \in X_i \setminus X_{i+1}$, we have that $v \notin X_j$ for all $j > i$.

2. A strategy for the cop player is *robber-monotone* if in playing the strategy, the set of vertices reachable by the robber is non-increasing. That is, if $(X_0, r_0), (X_1, r_1), \ldots$ is a play consistent with the strategy, then $Reach_{G \setminus X_{i+1}}(r_{i+1}) \subseteq Reach_{G \setminus X_i}(r_i)$ for all i.

It turns out that cop-monotone and robber-monotone strategies are closely related to one another as was proven in [6].

Lemma 1 ([6, Lemma 15]). *1. If the cop player has a robber-monotone strategy then he also has a cop-monotone strategy.*

2. Any cop-monotone strategy is also robber-monotone.

The cops and robbers game plays an essential role in the study of graph decompositions. It turns out that monotonic solutions to the cops and robbers game correspond directly to upper bounds on the DAG-width of the associated graph.

Theorem 1 ([6, Theorem 16]). *For any digraph G there is a DAG-decomposition of width at most k if and only if the cop player has a monotone winning strategy in the k-cops and robbers game on G.*

Corollary 1 ([6, Corollary 17]). *Let G be a digraph. Then G has a DAG-width of 1 if and only if G is acyclic.*

3.1.2 Concepts in Computational Complexity

Knowledge in computational complexity is important in the study of Sequencing by Hybridization. This is due to the fact that there are multiple sequencing algorithms and it is important to be able to distinguish which ones are most efficient. In this section we introduce some basic notions in computational complexity that will aid us in our discussions in later sections. Definitions in this section are based on [29, 32].

A *deterministic Turing machine* (abbreviated DTM or simply TM) is defined conceptually as a hypothetical machine that can manipulate symbols on a tape according to a specified function [31]. Formally, a Turing machine is defined as a 7-tuple $(Q, \Gamma, \Sigma, \delta, q_0, q_{accept}, q_{reject})$ whereby:

1. Q is a set of states.
2. Γ is a set of symbols known as the *tape alphabet*. $b \in \Gamma$ is known as the blank symbol.
3. $\Sigma \subseteq \Gamma \setminus \{b\}$ is known as the set of *input symbols*.
4. $q_0 \in Q$ is known as the *initial state*.
5. $q_{accept} \in Q$ is known as the *accepting state*.
6. $q_{reject} \in Q$ is known as the *rejecting state*.

7. $\delta : Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is known as the *transition function.*

For $x = x_1 \cdots x_{|x|} \in \Sigma^*$ (that is, x a string over the alphabet Σ), where $x_i \in \Sigma$ is the i^{th} character in x, we denote $x|_i^k = x_i \cdots x_{i+k-1}$. We define a configuration of a TM as a string of the form aq_ixb, where $a \cdot x \cdot b$ represents the non-blank symbols on the tape, $x \in \Sigma$ is the symbol that the read/write head is pointing to, $a \in \Sigma^*$ are the non-blank symbols to the left of the read/write head, $b \in \Sigma^*$ are the non-blank symbols to the right of the read/write head and q_i is the current state. Given two configurations $C_1 = aq_ixb$ and $C_2 = cq_jyd$ of a TM M we say that C_1 yields C_2 if one of the following holds:

1. $\delta(q_i, x) = (q_j, x', R)$ whereby $ax' = c$ and $yd = b$

2. $\delta(q_i, x) = (q_j, x', L)$ whereby $a_1 \cdots a_{|a|-1} = c$ and $a_{|a|}x'b = d$

We say that a TM M accepts a string $w \in \Sigma^*$ if there is a sequence of configurations $C_1, C_2, \ldots, C_n$ that accept w. That is:

1. $C_1 = q_0w$

2. C_i yields C_{i+1} for $i \in \{1, 2, \ldots, n-1\}$

3. C_n is a configuration with the accepting state.

The above definition also applies to the rejection of a string except that C_n is a configuration with the rejecting state.

We define a *language* as a set of strings over some fixed alphabet Σ. A TM M *recognizes* a language L iff M accepts all the strings in L and no other strings. If some TM recognizes a language L we say that L is recognizable. A TM M *decides* a language L iff M accepts all the strings in L and rejects all other strings. If some TM decides a language L we say that L is decidable.

In computational complexity, researchers are also often interested in *non-deterministic Turing machines* (NTMs). Non-deterministic Turing machines are similar to deterministic Turing machines, except that the machine changes configurations according to a relation Δ. For non-deterministic Turing machines we have that $\Delta \subseteq (Q \setminus \{q_{accept}, q_{reject}\} \times \Sigma) \times (Q \times \Sigma \times \{L, R\})$. Configuration yielding is defined in a similar way for NTMs. We say that configuration $C_1 = aq_ixb$ yields configuration $C_2 = cq_jyd$ if one of the following holds:

1. $(q_i, x, q_j, x', R) \in \Delta$ whereby $ax' = c$ and $yd = b$

2. $(q_i, x, q_j, x', L) \in \Delta$ whereby $a_1 \cdots a_{|a|-1} = c$ and $a_{|a|}x'b = d$

The main difference here is that a given configuration may yield multiple other configurations. For example, if a NTM N, is in configuration C_i, there is a set of possible

next configurations $\{C_{j_1}, \ldots C_{j_k}\}$. A *witness* or *certificate* is a string which defines a sequence of configuration choices for an NTM. When we run an NTM N with an input string w we can also provide it with a certificate c, which designates the choices N will make should it encounter a configuration that yields multiple configurations. For instance, in the previous example we could define certificates $c_{j_1}, \ldots, c_{j_k}$ such that if we run N with input w and witness c_{j_t}, if N encounters configuration C_i then C_i will yield C_{j_t} according to the certificate. We say that a NTM M accepts a string $w \in \Sigma^*$ if there exists at least one certificate c indicating a sequence of configurations ending with an accepting configuration. We say that M rejects a string w if all possible certificates result in configuration sequences that reject w.

The running time of a TM, M that halts on all input, is a function $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of steps that M uses on any input of length n. For a non-deterministic turing machine N that halts on all branches of computation, we define the running time as a function $g : \mathbb{N} \to \mathbb{N}$ where $g(n)$ is the maximum number of steps that N uses for any witness on its computation on any input of length n.

Note that any mathematical object can be encoded as a string over a fixed alphabet Σ. This is actually the case with computers whereby graphs, functions, sets, etc. are all encoded in the computers memory as strings over the alphabet $\{0, 1\}$. We define

the encoding of some mathematical object X over a fixed alphabet Σ as $< X >$. For A and B subsets of Σ^* and T^* respectively and some computable function $f : \Sigma^* \to T^*$ we say that f is a *mapping reduction* from A to B provided $a \in A$ iff $f(a) \in B$. If such a function exists we say that A is *mapping reducible* to B, denoted $A \leq_m B$.

Note that one can simulate an NTM, N, using a TM, M. The basic idea involves having M try all possible configuration sequences for a given input $w \in \Sigma^*$. For a more detailed explanation of this notion refer to [29].

We can draw a relation between TM's and actual computer algorithms. Church and Turing proposed a thesis pertaining to this concept. To this day the thesis has not been proven, however, it is widely accepted as true.

Church-Turing Thesis. *Any real-world computation can be translated into an equivalent computation involving Turing Machines.*

It is also important to note that all reasonable deterministic models of computation are polynomially equivalent, i.e., any one model can simulate the other with only a polynomial change in the time complexity. This allows us to discuss complexity without the concern of the selection of a particular computational model.

In computational complexity a *computational problem* is a question which a computer might be able to solve. These questions often have an input parameter associated with them known as an *instance*. For example, "Given an integer n, what is

the prime factorization of n?" is a computational problem. If we were to ask the question with $n = 123$, for example, then 123 would be the instance of the problem. A *decision problem* is defined as a question with a yes or no answer depending on the value of the instance. For example, the Hamiltonian path problem is the problem of determining whether or not a given graph has a Hamiltonian path. This is clearly a yes or no problem whereby the instance of the problem is a particular graph, hence the Hamiltonian path problem is an example of a decision problem. It is important to note that we can translate this problem into a problem involving TM's. The input to the TM would be the encoding of a graph G, denoted $< G >$ and the TM would accept if $< G >$ is the encoding of a graph containing a Hamiltonian path. If $< G >$ encodes a graph with no Hamiltonian path the TM would terminate in a rejecting state. We can therefore think of a decision problem as a problem of deciding membership in a particular language. In the case of the Hamiltonian path problem, we can define the language $L_H = \{< G >: G \text{ contains a Hamiltonian path}\}$, and the problem becomes that of deciding whether or not a particular graph encoding $< G' >$ belongs to the set L_H. We can therefore talk about languages and decision problems interchangeably.

Let f and g be two functions defined on some subset of the real numbers. We say that $f(n) = O(g(n))$ if and only if there is a positive constant M_0 and n_0 such for all

$n \geq n_0$ we have that

$$|f(n)| \leq M_0 g(n).$$

Similarly we say that $f(n) = \Omega(g(n))$ if and only if there is a positive constants M_1 and n_1 such that for all $n \geq n_1$ we have that

$$M_1 g(n) \leq f(n).$$

Finally, we say that $f(n) = \Theta(g(n))$ if and only if there are positive constants K_1, K_2 and N such that for all $n \geq N$ we have that

$$K_1 g(n) \leq f(n) \leq K_2 g(n).$$

The complexity class TIME($f(n)$) is defined as the set of decision problems that can be solved in $O(f(n))$ time by a deterministic Turing machine. The complexity class P is defined as:

$$\mathrm{P} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}(n^k).$$

Similarly, we define the complexity class NTIME($f(n)$) as the set of decision problems that can be solved in $O(f(n))$ time by a non-deterministic Turing machine. The complexity class NP is defined as:

$$\mathrm{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k).$$

We say that a decision problem Ψ is NP-complete if:

1. $\Psi \in$ NP and

2. $\forall Y \in$ NP, $Y \leq_m \Psi$ in polynomial time.

Given a problem Π, we define D_Π as the set of all instances of Π. We define two functions $L : D_\Pi \to \mathbb{N}$ and $M : D_\Pi \to \mathbb{N}$ which are to be associated with any decision problem. The function L is a length function which is intended to map any instance I of Π to an integer $L(I)$ which corresponds to the number of symbols used to describe I under some encoding scheme for Π. The function M is a max function which is intended to map any instance I to an integer $M(I)$ that corresponds to the magnitude of the largest number in I. We say that two pairs of length and max functions, (L, M) and (L', M') are polynomially related if there exist two-variable polynomials q and q' such that for all $I \in D_\Pi$ we have that

$$M(I) \leq q'(M'(I), L'(I))$$

and

$$M'(I) \leq q(M(I), L(I)).$$

As an example consider the partition problem involving sets.

Example 4. Partition

Instance: A set $A = \{a_1, a_2, \ldots, a_n\}$ and sizes $s(a_1), s(a_2), \ldots, s(a_n) \in \mathbb{Z}^+$.

Question: Is there a set $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

The following are length and max functions such that each pair of length/max function is polynomially related:

$$\begin{aligned} L_1(I) &= |A| + \sum_{a \in A} \lceil \log_2 s(a) \rceil \\ L_2(I) &= |A| + \max\{\lceil \log_2 s(a) \rceil : a \in A\} \\ M_1(I) &= \max\{s(a) : a \in A\} \\ M_2(I) &= \sum_{a \in A} s(a) \end{aligned}$$

All subsequent results in this section will hold for any length and max functions that are polynomially related to the ones we are using.

For any problem Π and any polynomial p, let Π_p denote the subproblem of Π obtained by restricting Π to only instances of I satisfying $M(I) \leq p(L(I))$. We say that a decision problem Ψ is *strongly NP-complete* if Ψ is in NP and there exists a polynomial p over the integers such that Ψ_p is NP-complete.

Given two problems Π_1 and Π_2 we say that Π_1 is polynomial-time *Turing reducible* to Π_2, denoted $\Pi_1 \leq_T \Pi_2$, if Π_1 can be solved using a polynomial number of calls to an algorithm which solves Π_2 and is polynomial outside of the algorithm calls. A problem Π is said to be NP-hard if there is an NP-complete problem L such that

$L \leq_T \Pi$ in polynomial time. Note that this definition of NP-hardness is an extension of the one we described previously, since if $A \leq_m B$ in polynomial time, then $A \leq_T B$ in polynomial time. The converse, however, is not true. The difference with using Turing reductions is that the NP-hard problem need not be a decision problem. We define strong NP-hardness analogously to strong NP-completeness. Specifically, we say that a problem Π is *strongly NP-hard* if there exists a polynomial p over the integers such that Π_p is NP-hard.

3.2 Sequence Reconstruction Using Graph Theory

Sequencing by Hybridization (SBH) is a technique that was developed simultaneously by several researchers as a method for sequencing DNA [3, 19]. The method relies on an array of probes containing all 4^k DNA sequences of length k. The array is used to generate the set of all k-length subsequences (k-mers) of our unknown DNA sequence A with length N. This set is known as the k-spectrum of A, denoted $S_k(A)$. Many papers published on SBH make the assumption that $S_k(A)$ is a multiset [7, 15, 28]. Once we obtain the multiset $S_k(A)$ we must piece the k-mers together and thus, determine our unknown DNA sequence A. It is important to note that there may be more than one way to piece these k-mers together and hence, more than one

candidate for A based on $S_k(A)$. This situation is known as ambiguous, or non-unique reconstruction.

To piece the k-mers together to form a string of length N we generate a graph $G = (V, E)$ whereby each k-mer x in $S_k(A)$ is represented by a vertex v in the graph G, such that the label of v, denoted $l(v)$, is x. Once all of the k-mers in $S_k(A)$ have been added as vertices in G we draw the set of directed edges. A directed edge is drawn from vertex u to vertex v if the last $k-1$ characters of $l(u)$ match the first $k-1$ characters of $l(v)$. Finally, we determine all Hamiltonian paths in G. Each Hamiltonian path in G through vertices $v_1 v_2 \cdots v_{N-k+1}$ corresponds to a DNA sequence with k-mers $l(v_1)l(v_2)\cdots l(v_{N-k+1})$ occurring in that order. This approach to reconstructing A is known as the Hamiltonian approach.

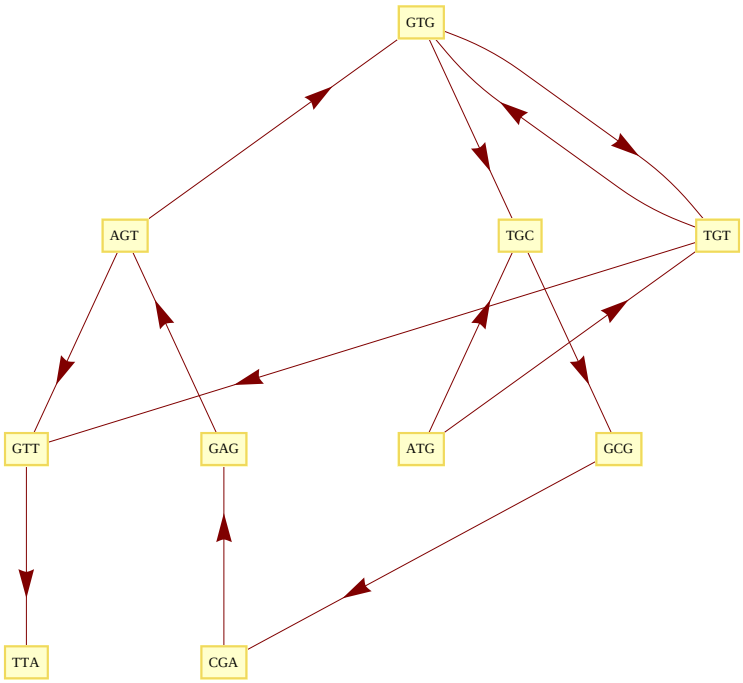


Figure 3.4: An example of the Hamiltonian approach for SBH. The above graph corresponds to the Hamiltonian graph generated from the spectrum {AGG, GGC, GCA, CAT, ATA, TAG, AGG, GGC, GCA, CAT}. There are two Hamiltonian paths corresponding to reconstructions ATGCGAGTGTTA and ATGTGCGAGTTA.

A major problem with the above method is that finding Hamiltonian paths in graphs is NP-hard. Pevzner showed that reconstructing a DNA sequence from its k-spectrum can be done in polynomial time [25]. We achieve this by generating a graph

G where each $k-1$-substring y in $S_k(A)$ is represented by a vertex v in G with label $l(v) = y$, where no two vertices have the same label (no repeated vertices from the same $k-1$ substrings). We then a directed edge $e = (v_1, v_2)$ for each k-mer $x \in S_k(A)$ where $l(v_1)$ is the first $k-1$ characters of x and $l(v_2)$ is the last $k-1$ characters of x. We determine all Eulerian paths in G and each Eulerian path through edges $e_1 e_2 \cdots e_{N-k+1}$ corresponds to a DNA sequence with k-merss $l(e_1)l(e_2)\cdots l(e_{N-k+1})$ occurring in that order. This approach to reconstructing A is known as the Eulerian approach.

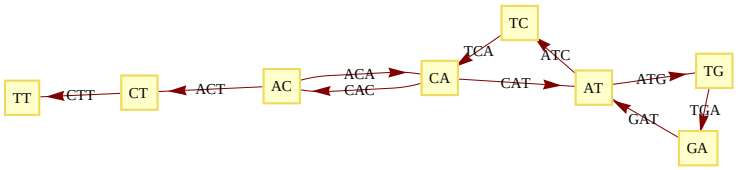


Figure 3.5: An example of the Eulerian graph approach for SBH. The above graph corresponds to the Eulerian graph generated from the spectrum {ACA, CAT, ATG, TGA, GAT, ATC, TCA, CAC, ACT, CTT}. The sole Eulerian trail in the graph corresponds to the reconstruction ACATGATCACTT.

3.3 Variants on the SBH problem

SBH is commonly studied under different contexts with different assumptions made. Sometimes it may be possible to obtain additional information using other lab experiments. This information can sometimes reduce the chance of ambiguous reconstruction. People also sometimes assume that the k-spectrum of an unknown DNA sequence may contain up to a certain number of errors. Such assumptions have been classified as separate problems of their own [15, 26]. In this section we will discuss several of these variations of the SBH problem. As we have seen so far, the classical SBH problem is defined as follows:

Problem 1. Given the multiset $S_k(A)$ of k-mers, determine all sequences A' such that A' contains exactly those k-mers present in $S_k(A)$.

One problem with SBH that is characteristic of the lab experiments required to generate the k-spectrum is the potential presence of errors in the k-spectrum. These errors fall under two distinct categories. This first category of errors is false negative errors. A false negative error occurs when a specific k-mer occurs in the unknown DNA sequence A but does not occur in its k-spectrum obtained from probes. SBH with the potential presence of false negative errors can be formulated as follows:

Problem 2. Given the multiset $S_k^-(A)$ of k-mers such that $S_k^-(A) \subseteq S_k(A)$, deter-

mine all sequences A' such that A' contains all the k-mers present in $S_k^-(A)$ as well as any number of additional k-mers.

The second category of errors is false positive errors. A false positive error occurs when a specific k-mer occurs in A's k-spectrum, but does not occur as a k-mer of A. SBH with the potential presence of false positive errors can be formulated as follows:

Problem 3. Given the multiset $S_k^+(A)$ of k-mers such that $S_k^+(A) \supseteq S_k(A)$, determine all sequences A' such that A' contains all or some of the k-mers present in $S_k^+(A)$.

As previously mentioned it is often possible to obtain additional information about the unknown DNA sequence which helps reduce the chance of ambiguous reconstruction. One such variation of SBH that assumes some additional information is Positional Sequencing by Hybridization (PSBH). PSBH was suggested to improve the resolving power of SBH using additional lab experiments which enables one to find the approximate positions of every k-mer in a DNA sequence [8, 26]. Although this greatly reduces the ambiguity as compared with that of regular SBH, polynomial time algorithms for PSBH are unknown [26]. PSBH can be formulated as follows:

Problem 4. Given the set $S_k(A)$ and interval $I_{k_j} = \{l_{k_j}, h_{k_j}\}$, $l_{k_j} < h_{k_j}$ for each $k_j \in S_k(A)$, find all sequences A' with k-mers $k_1, k_2, \ldots, k_i, \ldots, k_n$ occurring in that order

such that A contains exactly those k-mers present in $S_k(A)$ and for each $1 \leq i \leq n$ we have that $l_{k_i} \leq i \leq h_{k_i}$.

3.4 The Computational Complexity of Sequencing by Hybridization

In this section we will discuss in greater detail the computational complexity of DNA sequencing by hybridization. As mentioned in Section 3.2, sequence reconstruction can be done in polynomial time, provided the reconstruction is unique and no errors are present in the spectrum. If, however, we further generalize the SBH problem to include both positive and negative errors, the problem is no longer polynomial. This was shown to be NP-hard. In this section we will discuss this result as well as several other observations on the complexity of variants of the SBH problem. The definitions, theorems and proofs from this section are taken from [7].

The non-error version of the SBH problem is defined as follows:

Problem 5. Given the set $S_k(A)$ of k-mers and the length N of the original sequence, find a sequence A' of length N such that A' contains exactly those k-mers present in $S_k(A)$.

In Section 3.2 we established that this problem can be solved in polynomial time.

The main reason for this is that the problem can be transformed into an equivalent problem involving finding an Eulerian trail in a directed graph. Since we are only interested in finding one such string A', we only need to find one Eulerian trail in the resulting directed graph, which can be done in polynomial time. Note that the same can be said about the equivalent problem which has multisets for the spectrum, since we again are only interested in one Eulerian trail.

Theorem 2. *Problem 5 is in P*

Proof. We apply the transformation for the Eulerian approach for SBH in Section 3.2. Since we are only required to find one Eulerian trail in the resulting directed graph, and since a single Eulerian trail can be found in polynomial time, the problem is solvable in polynomial time. □

The problem of interest in [7] involved DNA sequencing with the presence of both positive and negative errors. The associated problem is defined as:

Problem 6. Give the set $S_k^{\pm}(A)$ of k-mers and length N of the original sequence, find a sequence of length $\leq N$ containing the maximum number of elements in $S_k^{\pm}(A)$.

In order to determine the complexity of this problem we must first analyze the problem under the restriction of only one type of error. The first type of errors we

consider are false-negative errors. The associated problem with only false negative errors is defined as:

Problem 7 (DNA sequencing with only negative errors - search version Π_{nss})**.** Given the set $S_k^-(A)$ of k-mers and the length N of the original sequence, where $S_k^-(A) \subseteq S_k(A)$, and $S_k(A)$ is the spectrum of the sequence, find a sequence of length $\leq N$ containing all the elements of S.

The corresponding decision problem is defined as follows:

Problem 8 (DNA sequencing with only negative errors - decision version Π_{nsd})**.** Given the set $S_k^-(A)$ of k-mers and length N of the original sequence, where $S_k^-(A) \subseteq S_k(A)$, and $S_k(A)$ is the spectrum of the sequence, determine if there is a sequence of length $\leq N$ containing all the elements of S.

The important thing to note at this point is that we are given in the problem the length of the original DNA sequence as well as the type of error that can occur. However, knowing that only false negative errors can occur and knowing the length of the original sequence implies that we can always reconstruct the DNA sequence in question. This, in turn, implies that the associated decision problem will always have the answer "yes". This leads us to define an alternate version of the decision problem:

Problem 9 (Positive quasi-sequencing - decision version Π_{nqd})**.** Given the set $S_k^*(A)$ of k-mers and length N of the original sequence, determine if there is a sequence of length $\leq N$ containing all the elements of $S_k^*(A)$.

The main difference between this decision problem and the previous one is that not all instances of this problem have the answer "yes". The reason for this is that the spectrum in the instances of this problem do not necessarily have only false negative errors. It is the case, however that any instance of Π_{nsd} which answers yes will also result in an answer of yes for the problem Π_{nqd}, and vice versa. Note that this problem is closely related to a variant of the longest common superstring problem. A string u is a superstring of a string s if it contains s as a substring. The variant of the longest common superstring problem is defined as:

Problem 10. Given a set S of words of equal length k over a finite alphabet, the length N of a superstring to be found, find a superstring of length N containing all elements of S.

Lemma 2 ([7, Lemma 1])**.** *The negative quasi-sequencing problem is strongly NP-complete.*

Proof. In [16] the above variant of the shortest common superstring problem was proven to be strongly NP-complete. Moreover, this is true even if the size of the

alphabet is bounded by a number not smaller than 3. It can also easily be shown that searching for a superstring of length not greater than n does not change the complexity of the problem. Note that this is actually equivalent to the negative quasi-sequencing problem. □

Theorem 3 ([7, Theorem 1]). *DNA sequencing with only negative errors* Π_{nss} *(search version) is strongly NP-hard*

Proof. Proving strong NP-completeness of negative quasi-sequencing Π_{nqd} directly implies the strong NP-hardness of sequencing with only negative errors Π_{nss}. This is true because if we had an algorithm solving Π_{nss} in polynomial time, we could use it to solve problem Π_{nqd} in polynomial time. This could be achieved as follows. Suppose A is the algorithm mentioned and let $P(x)$ be the bound on A's running time on graphs with Hamiltonian paths. We apply A to a graph G obtained from the Hamiltonian approach to SBH. If G has a Hamiltonian path, A will find one and terminate in time $P(|G|)$. If G does not have a Hamiltonian path, then after $P(|G|)$ steps A would not find one and we terminate the algorithm, □

This concludes the discussion of DNA sequencing with only negative errors. We now turn our attention to DNA sequencing with only positive errors. Recall that a false positive error occurs when a particular element of our k-spectrum does not

actually occur as a k-mer of our unknown DNA sequence. The problem of DNA sequencing with only false positive errors is defined as follows:

Problem 11 (DNA sequencing with only positive errors - search version Π_{pss})**.** Given the set $S_k^+(A)$ of k-mers and length N of the original sequence, where $S_k^+(A) \supseteq S_k(A)$, $S_k(A)$ being the actual spectrum of the sequence, find a sequence of length N containing $N - k + 1$ of the elements of $S_k^+(A)$.

Similar to sequencing with negative errors, we must now define the associated decision problem.

Problem 12 (DNA sequencing with only positive errors - decision version Π_{psd})**.** Given the set $S_k^+(A)$ of k-mers and length N of the original sequence, where $S_k^+(A) \supseteq S_k(A)$, $S_k(A)$ being the actual spectrum of the sequence, determine if there exist a sequence of length N containing $N - k + 1$ of the elements of $S_k^+(A)$.

Again we have that all instances of the problem result in the answer "yes". We now define the corresponding quasi-sequencing problem

Problem 13 (Positive quasi-sequencing - decision version Π_{pqd})**.** Given the set $S_k^*(A)$ of k-mers and length N of the original sequence, determine if there is a sequence of length N containing $N - k + 1$ of the elements of $S_k^*(A)$.

As with negative quasi-sequencing, we have that any instance of Π_{psd} which answers yes will also result in an answer of yes for the problem Π_{pqd}, and vice versa. We now proceed by proving the NP-completeness of positive quasi-sequencing. To do this we apply a polynomial time mapping reduction from the NP-complete problem *directed Hamiltonian path between two vertices*. This problem is defined as follows:

Problem 14 (Directed Hamiltonian path between two vertices (DHPBTV))**.** Given a 1-directed graph $D = (V_D, E_D)$ with two specified vertices s and t, determine if there is a Hamiltonian path from s to t in D.

Lemma 3 ([7, Lemma 2])**.** *The positive quasi-sequencing problem Π_{pqd} is strongly NP-complete.*

Proof. Given an instance of DHPBTV, the instance of Π_{pqd} is constructed using the following steps:

- to each vertex $v \in V_D$, assign a unique label $e(v)$ of length $\lceil \log_2 |V_D| \rceil$ over the alphabet $\{A, C\}$.
- let $k = 2\lceil \log_2 |V_D| \rceil + 2$, where k is the length of all constructed elements of $S_k^*(A)$.
- build $S_k^*(A)$ such that for every $v \in V_D$, one oligonucleotide is constructed of the form $e(v) \cdot G \cdot e(v) \cdot T$.

- add to $S_k^*(A)$ $k-1$ oligonucleotides for every arc $(u,v) \in E_D$, where the oligonucleotides are of the form $u_2 \cdot u_3 \cdots u_k \cdot v_1, u_3 \cdot u_4 \cdots v_1 \cdot v_2, \ldots, u_k \cdot v_1 \cdots v_{k-2} \cdot v_{k-1}$ whereby $u_1 \cdot u_2 \cdots u_k$ is an oligonucleotide corresponding to vertex u.

- add to $S_k^*(A)$ k starting oligonucleotides of the form $g_1 \cdot g_2 \cdots g_{k-1} \cdot g_k, g_2 \cdot g_3 \cdots g_k \cdot s_1, \ldots, g_k \cdot s_1 \cdots s_{k-2} \cdot s_{k-1}$ where $g_i = G$ and $s_1 \cdot s_2 \cdots s_k$ is an oligonucleotide corresponding to starting vertex s.

- add to $S_k^*(A)$ k ending oligonucleotides of the form $t_2 \cdot t_3 \cdots t_k \cdot w_1, t_3 \cdot t_4 \cdots w_1 \cdot w_2, \ldots, t_k \cdot w_1 \cdots w_{k-2} \cdot w_{k-1}, w_1 \cdot w_2 \cdots w_{k-1} \cdot G$ where $w_i = T$ and $t_1 \cdot t_2 \cdots t_k$ is an oligonucleotide corresponding to vertex t.

The words generated from this method can be duplicated only if they correspond to different arcs leaving the same vertex, or if they correspond to different arcs entering the same vertex. In the spectrum the word only appears once, but it does not affect the construction of a solution. We now show that a Hamiltonian path from s to t in D exists if and only if such a sequence of length $n = k(|V_D|+2)$ exists which includes the number of different elements of the spectrum $S_k^*(A)$ equal to $k(|V_D|+1)+1$.

Let us assume that D possesses a Hamiltonian path from s to t. One element in $S_k^*(A)$ corresponds to each vertex from the path and $k-1$ elements correspond to each arc in the path. A construction of the elements makes it possible to construct a

string of length $k|V_D|$ letters if all of the $k(|V_D|-1)+1$ elements, in a proper order, are maximally overlapped. If one then adds all starting elements and ending elements (with maximal overlap) we obtain a string of $k(|V_D|+1)+1$ different elements of the spectrum of length $k(|V_D|+2)$.

Now assume that a sequence of letters of length $k(|V_D|+2)$ exists and a number of included elements of $S_k^*(A)$ is equal to $k(|V_D|+1)+1$. This implies neighboring elements in the sequence are maximally overlapped. This can only happen if between any two consecutive elements corresponding to vertices, there are $k-1$ elements corresponding to an arc joining the vertices. If one attempted to construct a sequence using only elements which corresponded to vertices and arcs we would obtain a sequence of at most $|V_D|+(|V_D|+1)(k-1)$ elements. This implies that the sequence would have two elements less than required. Therefore the sequence must consist of starting and ending elements. This forces the first vertex element of the sequence to correspond to s and the last to correspond to t. All other vertex elements must appear between the first and last vertex elements. To connect them by arc elements, arcs joining vertices following each other in the sequence should exist in D. This implies the sequence must contain spectrum elements in the following order:

- k starting elements

- an element corresponding to s
- $k-1$ elements corresponding to an arc leaving s
- other elements from vertices and arcs connecting them
- $k-1$ elements corresponding to an arc entering t
- an element corresponding to t
- k ending elements

This ordering directly corresponds to a Hamiltonian path in D from s to t. Since DHPBTV is strongly NP-complete and the above reduction is polynomial, we have that Π_{pqd} is strongly NP-complete. □

We provide two examples of this transformation, one of which is illustrated in [7] as an example when D contains a Hamiltonian path, and a second of our own which illustrates the transformation when D does not contain a Hamiltonian path.

Example 5. A 1-digraph $D_1 = (V_{D_1}, E_{D_1})$ is given below, which contains a Hamiltonian path from s to t.

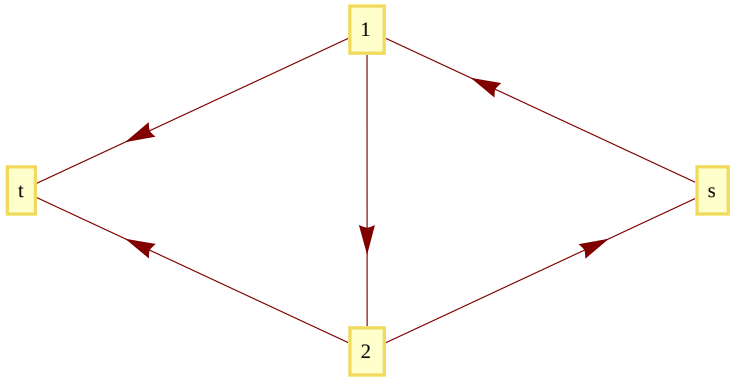


Figure 3.6: Directed graph D_1 corresponding to an instance of directed Hamiltonian path between two vertices

We consider labels for each vertex of length $\lceil \log_2 |V_{D_1}| \rceil = 2$ over $\{A, C\}$. We obtain $s - AA$, $1 - AC$, $2 - CA$, $t - CC$. Next we construct the elements of the spectrum of length $k = 2\lceil \log_2 |V_{D_1}| \rceil + 2 = 6$.

- $s - AAGAAT$
- $1 - ACGACT$
- $2 - CAGCAT$
- $t - CCGCCT$
- $(s, 1) - AGAATA, GAATAC, AATACG, ATACGA, TACGAC$
- $(1, 2) - CGACTC, GACTCA, ACTCAG, CTCAGC, TCAGCA$

- $(1,t) - CGACTC, GACTCC, ACTCCG, CTCCGC, TCCGCC$
- $(2,s) - AGCATA, GCATAA, CATAAG, ATAAGA, TAAGAA$
- $(2,t) - AGCATC, GCATCC, CATCCG, ATCCGC, TCCGCC$

We now add the starting and ending elements: $GGGGGG$, $GGGGGA$, $GGGGAA$, $GGGAAG$, $GGAAGA$, $GAAGAA$, $CGCCTT$, $GCCTTT$, $CCTTTT$, $CTTTTT$, $TTTTTT$, $TTTTTG$. Overall we have the spectrum:

$$S_k^*(A) = \{$$

$$AAGAAT, ACGACT, CAGCAT, CCGCCT, AGAATA, GAATAC,$$

$$AATACG, ATACGA, TACGAC, CGACTC, GACTCA, ACTCAG,$$

$$CTCAGC, TCAGCA, GACTCC, ACTCCG, CTCCGC, TCCGCC,$$

$$AGCATA, GCATAA, CATAAG, ATAAGA, TAAGAA, AGCATC,$$

$$GCATCC, CATCCG, ATCCGC, GGGGGG, GGGGGA, GGGGAA,$$

$$GGGAAG, GGAAGA, GAAGAA, CGCCTT, GCCTTT, CCTTTT,$$

$$CTTTTT, TTTTTT, TTTTTG$$

$$\}$$

The corresponding graph using the Hamiltonian approach is given in the following figure.

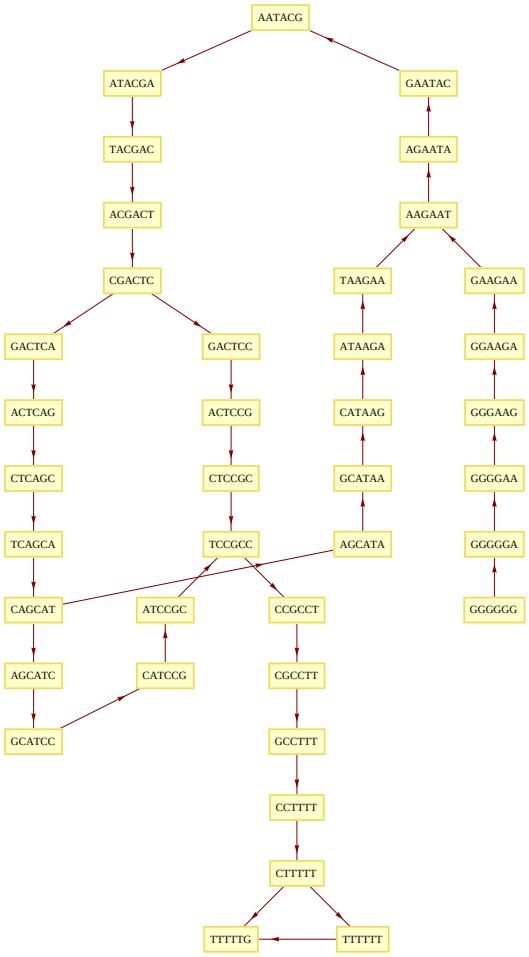


Figure 3.7: Graph generated from $S_k^*(A)$ using the Hamiltonian approach with a reconstruction possible

We are now required to look for a sequence of length $k(|V_{D_1}| + 2) = 36$ using

$k(|V_{D_1}| + 1) + 1 = 31$ elements of the spectrum. The solution here is

$$GGGGGGAAGAATACGACTCAGCATCCGCCTTTTTTG.$$

Example 6. Let us now consider a different example in which the 1-digraph $D_2 = (V_{D_2}, E_{D_2})$ does not contain a Hamiltonian path from s to t.

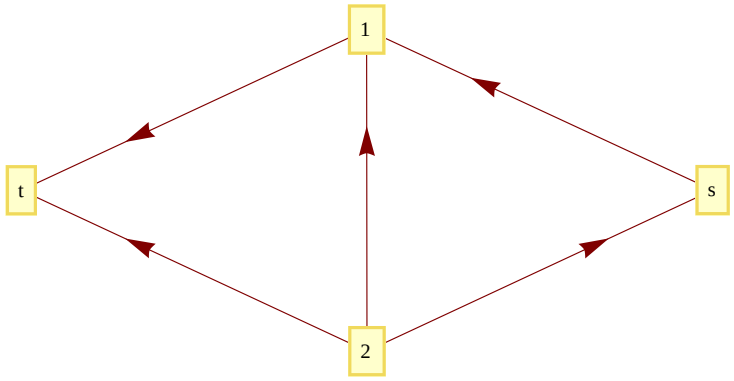


Figure 3.8: Directed graph D_2 corresponding to an instance of directed Hamiltonian path between two vertices

We label the vertices of the graph the same as the graph in the previous example. Since the only difference between this graph and the previous one is that the arc $(1, 2)$ is replaced by the arc $(2, 1)$, we have that the majority of the spectrum elements are the same. The only notable difference is that we replace the elements associated with $(1, 2)$ with the elements:

- $(2, 1) - AGCATA, GCATAC, CATACG, ATACGA, TACGAC$.

The resulting spectrum in this case is given by:

$$S_k^*(A) = \{$$

$$AAGAAT, ACGACT, CAGCAT, CCGCCT, AGAATA, GAATAC,$$

$$AATACG, ATACGA, TACGAC, GCATAC, CATACG, ATACGA,$$

$$TACGAC, CGACTC, GACTCC, ACTCCG, CTCCGC, TCCGCC,$$

$$AGCATA, GCATAA, CATAAG, ATAAGA, TAAGAA, AGCATC,$$

$$GCATCC, CATCCG, ATCCGC, GGGGGG, GGGGGA, GGGGAA,$$

$$GGGAAG, GGAAGA, GAAGAA, CGCCTT, GCCTTT, CCTTTT,$$

$$CTTTTT, TTTTTT, TTTTTG$$

$$\}$$

The corresponding graph using the Hamiltonian approach is given in the following figure.

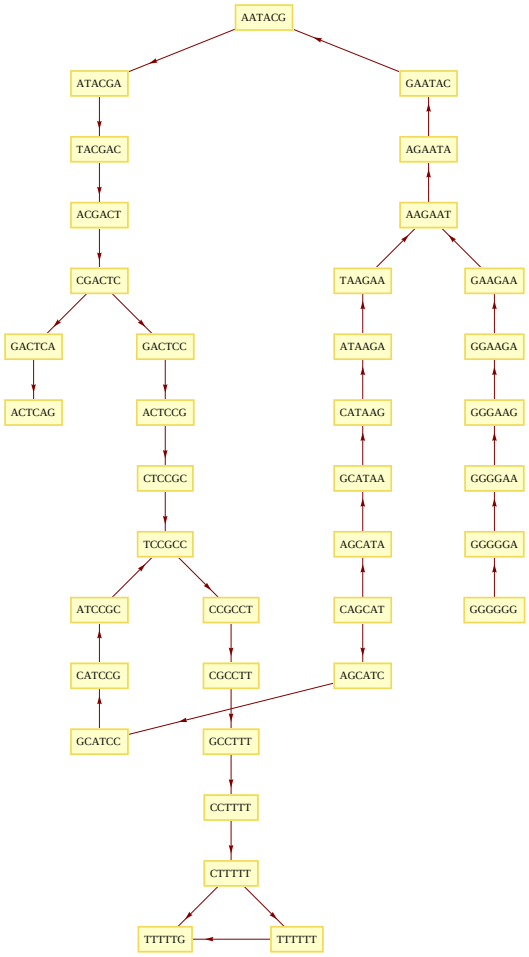


Figure 3.9: Graph generated from $S_k^*(A)$ using the Hamiltonian approach with no reconstructions possible

It is easily seen that there is no path which travels through 31 of the vertices in

the graph, hence there is no reconstruction possible using the spectrum $S_k^*(A)$.

Theorem 4 ([7, Theorem 2])**.** *DNA sequencing with only positive errors* Π_{pss} *(search version) is strongly NP-hard.*

Proof. The proof can be deduced in the same manner as the proof for DNA sequencing with only negative errors Π_{nss} (search version). □

Corollary 2. *DNA sequencing with negative and positive errors (search version) is strongly NP-hard*

Chapter 4

Probability Models for Non-Unique Reconstruction

One aspect of study of SBH for many researchers is the probability that a random DNA sequence of length $N = n+k-1$ has an ambiguous reconstruction using probes of size k [1, 2, 14, 28]. As a general rule of thumb, the larger the value of k the lower the probability of an ambiguous reconstruction. Although this is helpful, having a more accurate idea of the probability is very important before attempting sequencing in order to avoid wasting lab time on sequencing which will likely be unsuccessful. In this section we will explore this notion.

For a given a DNA sequence $A = a_1 a_2 \cdots a_{n+k-1}$, where n and k are positive

integers, we define $A|_i^k = a_i a_{i+1} \cdots a_{i+k-1}$ for $1 \leq i \leq n$. We say that the pair (i, j), with $i < j$ and $j \leq n$, is a k-repeat in A if $a_i \cdots a_{i+k-1} = a_j \cdots a_{j+k-1}$. A k-repeat (i, j) in A is called *rightmost* if $j + k - 1 \neq n + k - 1$ and $(i + 1, j + 1)$ is not a k-repeat. A k-repeat (i, j) is called *weakly rightmost* if (i, j) is rightmost or $j = n$. We say that a pair of k-repeats $((i, j), (i', j'))$ is a *k-R-pair* if (i, j) is rightmost and it is a *k-Rr-pair* if (i, j) is rightmost and (i', j') is weakly rightmost. The pair is called *interleaved* if $i \leq i' < j < j'$.

In [28] a theta expression was given for the probability that a random DNA sequence of length N has an ambiguous reconstruction using size k probes. The expression was developed using several observations regarding ambiguous reconstructions, combinatorial enumeration, and statistics. The observations giving necessary and sufficient conditions for unique reconstructability is as follows, which is based on the following results in [28]

Theorem 5 ([28, Theorem 3.1]). *A sequence A of length $n + k - 1$ is not uniquely recoverable with respect to probes of size k iff either*

1. *A contains an interleaved $(k - 1)$-R-pair*

2. *$A|_1^{k-1} = A|_{n+1}^{k-1}$ and there is an $i \in \{1, \ldots, n + 1\}$ such that $A|_1^{k-1} \neq A|_i^{k-1}$.*

Theorem 6 ([28, Theorem 3.2]). *A sequence A of length $n+k-1$ is not uniquely recoverable with respect to probes of size k iff either*

1. *A contains an interleaved $(k-1)$-Rr-pair*

2. *$A|_1^{k-1} = A|_{n+1}^{k-1}$ and there is an $i \in \{1, \ldots, n+1\}$ such that $A|_1^{k-1} \neq A|_i^{k-1}$.*

Lemma 4 ([28, Lemma 3.3]). *Let $P^{k-1}_{i,i',j,j'}$ denote the probability that $((i,j),(i',j'))$ is an interleaved $(k-1)$-Rr-pair. Then we have that $P^{k-1}_{i,i',j,j'} \in \{0, 9/4^{2k}\}$ for $j' < n+1$ and $P^{k-1}_{i,i',j,j'} \in \{0, 12/4^{2k}\}$ for $j' = n+1$*

Proof. Consider the case where $j' < n+1$. Suppose $A = a_1 a_2 \cdots a_N$. Then we have that

$$P_{i,j,i',j'} = P\left[\begin{array}{c} a_i = a_j, a_{i+1} = a_{j+1}, \ldots, a_{i+k-2} = a_{j+k-2}, a_{i+k-1} \neq a_{j+k-1} \\ \\ a_{i'} = a_{j'}, a_{i'+1} = a_{j'+1}, \ldots, a_{i'+k-2} = a_{j'+k-2}, a_{i'+k-1} \neq a_{j'+k-1} \end{array}\right]$$

We now build a graph $G_{i,i',j,j'}$ whereby the vertices are the indices of the letters that appear in the equalities and inequalities above and the edges correspond to the equalities only. We have that the vertices are given by $\{i, \ldots, i+k-1\} \cup \{j, \ldots, j+k-1\} \cup \{i', \ldots, i'+k-1\} \cup \{j', \ldots, j'+k-1\}$ and the edges are $\{(i+r, j+r) | r = 0, \ldots, k-2\} \cup \{(i'+r, j'+r) | r = 0, \ldots, k-2\}$. Let $V_1, \ldots, V_b$ be the connected components of $G_{i,i',j,j'}$ and let n_l and m_l denote the number of vertices and edges in V_l respectively. The pairs (i,j) and (i',j') are repeats iff for each connected component

V_l, all of the corresponding letters in A are equal. The probability of this occurring is $\prod_{l=1}^{b}(1/4)^{n_l-1}$.

We now consider separate cases. In the first case let us assume $G_{i,i',j,j'}$ contains parallel edges. This implies that $(i+r_1, j+r_1) = (i'+r_2, j'+r_2)$ for $r_1, r_2 \in \{0, \ldots, k-2\}$ with $r_1 > r_2$. Therefore we have $i' - i = j' - j = r$ for some $r \in \{1, \ldots, k-2\}$. A repeat at $(i', j') = (i + r, j + r)$ implies that $a_{i+r+l} = a_{j+r+l}$ for all $l < k - 1$, and in particular, for $l = k - 1 - r$ we get $a_{i+k-1} = a_{j+k-1}$. But if (i, j) is a rightmost repeat we have that $a_{i+k-1} \neq a_{j+k-1}$. So, $((i, j), (i', j'))$ can not be an interleaved $(k-1)$-Rr-pair, so $P^{k-1}_{i,j,i',j'} = 0$.

Let $G'_{i,i',j,j'}$ be the graph obtained from $G_{i,i',j,j'}$ by adding the edges $e_1 = (i + k - 1, j + k - 1)$ and $e_2 = (i' + k - 1, j' + k - 1)$ which correspond to the two inequalities. For this case we assume that $G_{i,i',j,j'}$ has no cycles, and hence $G_{i,i',j,j'}$ has no cycles. This implies that $m_l = n_l - 1$ for all l. Therefore we have that the probability that (i, j) and (i', j') are repeats is $\prod_{l=1}^{b} 1/4^{m_l} = 1/4^{\sum_{l=1}^{b} m_l} = 1/4^{2(k-1)}$ which follows from the fact that $G_{i,i',j,j'}$ has $2(k-1)$ edges. Furthermore, as the edges e_1 and e_2 do not form a cycle we have that $P^{k-1}_{i,j,i',j'} = (1/4)^{2(k-1)}(3/4)^2 = 9/4^{2k}$.

Now, for the final case let us assume that $G'_{i,i'j,j'}$ contains a cycle. Note that the vertex $j' + k - 1$ only has $i' + k - 1$ as its neighbor so any cycle in $G'_{i,i'j,j'}$ cannot pass through e_2. We now claim that $G'_{i,i'j,j'}$ contains a cycle that passes through e_1.

Let $C = [v_1, v_2, \ldots, v_{r-1}, v_r = v_1]$ be some cycle in $G'_{i,i',j,j'}$. If C passes through e_1 then we are done. Otherwise, for any edge $e = (v_l, v_{l+1})$ in C, as $e \neq e_1, e_2$, then $(v_l + 1, v_{l+1} + 1)$ is also an edge in $G'_{i,i',j,j'}$. We repeat this process until we obtain a cycle that passes through e_1 and does not pass through e_2. Therefore the vertices $i+k-1$ and $j+k-1$ are in the same connected component of $G'_{i,i',j,j'}$ which implies $a_{i+k-1} = a_{j+k-1}$ and hence $((i,j),(i',j'))$ is not an interleaved $(k-1)$-Rr-pair, so we have that $P^{k-1}_{i,j,i',j'} = 0$. □

In [28] they define a function $P(n,k)$ as the probability that a random DNA sequence of length $n+k-1$ is not uniquely reconstructable using probes of size k. They further work to determine the asymptotics of $P(n,k)$ by developing upper and lower bounds on the function. The following theorems taken from [28] establish upper and lower bounds on $P(n,k)$.

Theorem 7 ([28, Corollary 3.4]). $P(n,k) \leq (\frac{3}{8}n^3 + \frac{5}{4}n^2) \cdot n/4^{2k} + 1/4^{(k-1)}$.

Proof. First suppose $j' < n+1$. If $i < i'$ there are $\binom{n}{4}$ ways of choosing i, i', j, j' such that $((i,j),(i',j'))$ is an interleaved $(k-1)$-Rr-pair. If $i = i'$ then there are $\binom{n}{3}$ ways. By Pascal's identity we have a total of $\binom{n}{4} + \binom{n}{3} = \binom{n+1}{4}$ possibilities.

If $j' = n+1$ then there are $\binom{n}{3}$ possibilities with $i < i'$ and $\binom{n}{2}$ possibilities with $i = i'$. So by Pascal's identity we have $\binom{n}{3} + \binom{n}{2} = \binom{n+1}{3}$.

We must also address case 2 of Theorem 5. The probability of this occurring is $1/4^{k-1}$. By Lemma 4 we have that

$$\binom{n+1}{4}\frac{9}{4^{2k}} + \binom{n+1}{3}\frac{12}{4^{2k}} + 1/4^{k-1} \leq (\frac{3}{8}n^3 + \frac{5}{4}n^2)\frac{n}{4^{2k}} + 1/4^{k-1}.$$

□

Theorem 8 ([28, Lemma 3.5]). *If $n \geq 4k$ then $P(n,k) \geq L(n,k)(1 - L(n,k)/2)$ where $D = \lfloor \frac{n}{4} \rfloor$ and*

$$L(n,k) = (D-k+1)^4 \frac{9}{4^{2k}} \left(1 - (D-k+1)^2 \frac{3}{4^k}\right)^2.$$

Corollary 3. $P(n,k) = \Theta(n^4/4^{2k})$.

Further studies were later done on other probability models for SBH. One such model developed by the author is based on conditional probability, and has been submitted to a journal [21]. Here we denote $P(n,k,t)$ as the probability that a random DNA sequence of length $N = n+k-1$ is not uniquely reconstructable using probes of size k, given that we know it is not uniquely reconstructable using probes of size $t < k$.

Theorem 9. *Given a random DNA sequence A of length $n+k-1$ we have that $P(n,k,t) = O\left(\frac{n^4 4^{2t}}{(n+k-t)^4 4^{2k}} + \frac{1}{4^{k-t}}\right)$ for $5(t-1) \leq n+k-t \leq 2^{t+1}+4(t-2)$ and $t < k$.*

Proof. Let E_z be the event that A is not uniquely reconstructible with respect to probes of size z, let R_z be the event that A contains an interleaved z-Rr pair and let X_z be the event that $A|_1^z = A|_{n+k-z}^z$. Given our previous definition of $P(n,k,t)$ we have that

$$P(n,k,t) = \Pr(E_k|E_t) = \frac{\Pr(E_k \cap E_t)}{\Pr(E_t)}.$$

Note that if A contains an interleaved $(k-1)$-Rr pair then A contains an interleaved $(t-1)$-Rr pair. This, together with Theorem 6 implies that

$$E_k \cap E_t = R_{k-1} \cup (R_{t-1} \cap X_{k-1}) \cup (X_{k-1} \cap X_{t-1}).$$

so we have

$$\Pr(E_k|E_t) \leq \frac{\Pr(R_{k-1}) + \Pr(R_{t-1} \cap X_{k-1}) + \Pr(X_{k-1} \cap X_{t-1})}{\Pr(E_t)}.$$

By Lemma 4 we have that $\Pr(R_{k-1}) \leq \binom{n+1}{4}\frac{9}{4^{2k}} + \binom{n+1}{3}\frac{12}{4^{2k}}$.

Note that $\Pr(R_{t-1} \cap X_{k-1}) = \Pr(R_{t-1})\Pr(X_{k-1}|R_{t-1})$. By lemma 4 we have that $\Pr(R_{t-1}) \leq \binom{n+(k-t)+1}{4}\frac{9}{4^{2t}} + \binom{n+(k-t)+1}{3}\frac{12}{4^{2t}}$. We now wish to find an upper bound on $\Pr(X_{k-1}|R_{t-1})$. Suppose event R_{t-1} occurs, and we have that $((i,j),(i',j'))$ is an interleaved $(t-1)$-Rr pair. The probability of event X_{k-1} then depends on the position of the interleaved $(t-1)$-Rr pair. Note that the probability of X_{k-1} is maximized when one of the repeats in the interleaved $(t-1)$-Rr pair occurs at the beginning and end of the sequence. This means there are less choices required to make the first $k-1$

characters of the sequence equal to the last $k-1$ characters. Note that if either of the repeats are at positions $(1+\alpha, n+1+\alpha)$, for $0 \leq \alpha < k-t$ then the occurrence of event X_{k-1} would be impossible, because if it did occur, then $(1+\alpha, n+1+\alpha)$ would no longer be rightmost, which is a contradiction. Note that if $\alpha = k-t$ then we have $(1+\alpha, n+1+\alpha) = (k-t+1, n+k-t+1)$. Now, note that event X_{k-1} can also occur because $(k-t+1, n+k-t+1)$ will be weakly rightmost, and the probability of X_{k-1} is $1/4^{k-t}$, hence $\Pr(R_{t-1} \cap X_{k-1}) \leq 1/4^{k-t}\left(\binom{n+(k-t)+1}{4}\frac{9}{4^{2t}} + \binom{n+(k-t)+1}{3}\frac{12}{4^{2t}}\right)$

Note that if event $X_{k-1} \cap X_{t-1}$ occurs then we have that $A|_1^{k-1} = A|_{n+1}^{k-1}$ and $A|_1^{t-1} = A|_{n+k-t+1}^{t-1}$. This implies that $A|_1^{t-1} = A|_{k-1-(t-2)}^{t-1}$. The probability of this occurring is $1/4^{t-1}$. Also, the probability that $A|_1^{k-1} = A|_{n+1}^{k-1}$ is $1/4^{k-1}$. These two facts together imply the occurrence of event $X_{k-1} \cap X_{t-1}$, hence, $\Pr(X_{k-1} \cap X_{t-1}) = 1/4^{k+t-2}$.

So overall we have that $\Pr(R_{k-1}) + \Pr(R_{t-1} \cap X_{k-1}) + \Pr(X_{k-1} \cap X_{t-1})$ is less than or equal to

$$\binom{n+1}{4}\frac{9}{4^{2k}} + \binom{n+1}{3}\frac{12}{4^{2k}} + \frac{1}{4^{k-t}}\left(\binom{n+(k-t)+1}{4}\frac{9}{4^{2t}} + \binom{n+(k-t)+1}{3}\frac{12}{4^{2t}}\right) + 1/4^{k+t-2}.$$

Hence,

$$\Pr(R_{k-1}) + \Pr(R_{t-1} \cap X_{k-1}) + \Pr(X_{k-1} \cap X_{t-1}) = O\left(\frac{n^4}{4^{2k}} + \frac{(n+k-t)^4}{4^{k+t}}\right).$$

Note that from [28] we have that $\Pr(E_t) = \Omega((n+k-t)^4/4^{2t})$ which implies

$$\Pr(E_k|E_t) = \frac{O\left(\frac{n^4}{4^{2k}} + \frac{(n+k-t)^4}{4^{k+t}}\right)}{\Omega((n+k-t)^4/4^{2t})} = O\left(\frac{n^4 4^{2t}}{(n+k-t)^4 4^{2k}} + \frac{1}{4^{k-t}}\right).$$

□

Theorem 10. *Given a random DNA sequence A of length $n+k-1$ we have that $P(n,k,t) = \Omega\left(\frac{n^4}{(n+k-t)^4 4^{2(k-t)}} + \frac{1}{4^k} - \frac{n^4}{4^{3k}}\right)$ for $5(t-1) \leq n+k-t \leq 2^{t+1}+4(t-2)$, $5(k-1) \leq n \leq 2^{k+1}+4(k-2)$ and $t < k$.*

Proof. Note that

$$P(n,k,t) = \Pr(E_k|E_t) \geq \frac{\Pr(R_{k-1}) + \Pr(\overline{R}_{k-1} \cap R_{t-1} \cap X_{k-1})}{\Pr(E_t)}$$

Note that $\Pr(R_k) = \Omega(n^4/4^{2k})$ by Lemma 3.5 of [28].

Consider now $\Pr(\overline{R}_{k-1} \cap R_{t-1} \cap X_{k-1})$. This expression is equivalent to $\Pr(\overline{R}_{k-1}) \Pr(X_{k-1} \cap R_{t-1}|\overline{R}_{k-1})$. Note that from the proof of the previous theorem we have that $\Pr(R_{k-1}) \leq \binom{n+1}{4}\frac{9}{4^{2k}} + \binom{n+1}{3}\frac{12}{4^{2k}}$ which implies $\Pr(\overline{R}_{k-1}) = 1 - \Pr(R_{k-1}) \geq 1 - \left(\binom{n+1}{4}\frac{9}{4^{2k}} + \binom{n+1}{3}\frac{12}{4^{2k}}\right)$. Also note that the event $\overline{R}_{k-1}$ has no effect on the events X_{k-1} or R_{t-1} hence, $\Pr(X_{k-1} \cap R_{t-1}|\overline{R}_{k-1}) = \Pr(X_{k-1} \cap R_{t-1}) = \Pr(R_{t-1}) \Pr(X_{k-1}|R_{t-1})$. Note that

$\Pr(R_{t-1}) = \Omega((n+k-t)^4/4^{2t})$ again by lemma 3.5 of [28]. We now wish to find a lower bound for $\Pr(X_{k-1}|R_{t-1})$. Note that $\Pr(X_{k-1}|R_{t-1}) \geq \Pr(X_{k-1}) = 1/4^{k-1}$.

Overall we have

$$\frac{\Pr(R_{k-1}) + \Pr(\overline{R}_{k-1} \cap R_{t-1} \cap X_{k-1})}{\Pr(E_t)}$$

which is greater than or equal to

$$\frac{\Omega(n^4/4^{2k}) + \left(1 - \left(\binom{n+1}{4}\frac{9}{4^{2k}} + \binom{n+1}{3}\frac{12}{4^{2k}}\right)\right)\Omega((n+k-t)^4/4^{2t})(1/4^{k-1})}{O((n+k-t)^4/4^{2t})}.$$

Simplifying the above expression we obtain

$$\frac{\Omega(n^4/4^{2k}) + \Omega(1-n^4/4^{2k})\Omega((n+k-t)^4/4^{2t+k})}{O((n+k-t)^4/4^{2t})}$$

which is

$$\Omega\left(\frac{n^4}{(n+k-t)^4 4^{2(k-t)}} + \frac{1}{4^k} - \frac{n^4}{4^{3k}}\right)$$

□

The actual value of $P(n,k,t)$ was simulated using a Monte Carlo method. In this method, a random set of DNA sequences of length $N = n+k-1$ is generated that are not reconstructable with respect to probes of size t for some $t < k$. Each sequence is then tested to see whether or not it is reconstructable using probes of size k. The ratio of the number reconstructible using probes of size k to the total number tested is then determined. See Algorithm 1 in Appendix A for a pseudocode description of the simulation.

n	k	t	$\frac{n^4 4^{2t}}{(n+k-t)^4 4^{2k}} + \frac{1}{4^{k-t}}$	$\frac{n^4}{(n+k-t)^4 4^{2(k-t)}} + \frac{1}{4^k} - \frac{n^4}{4^{3k}}$	Simulation
100	10	9	0.3100612715	0.0600622251	0.05
200	10	9	0.3112654701	0.0612664223	0.09
300	10	9	0.3116735651	0.0616745117	0.03
400	10	9	0.3118788868	0.0618798182	0.12
500	10	9	0.3120024900	0.0620033895	0.12
500	12	10	0.06634437003	0.0038444296	0.009
600	12	10	0.06635459782	0.0038546573	0.005
800	12	10	0.06636743043	0.0038674899	0.004
3000	15	12	0.01586816650	0.0002431674	0.000

Figure 4.1: Comparison of estimates of $P(n, k, t)$ using simulations and asymptotics from Theorems 9 and 10.

We can easily see from the table in Figure 4 that the asymptotic upper and lower bounds obtained from Theorems 9 and 10 are accurate.

Chapter 5

Extensions of SBH

In this section we will explore extensions of SBH in order to reduce the number of elements in the reconstruction sets. This is an important topic in SBH because having a large number of elements in the reconstruction set makes it difficult for people to have any idea of the nucleotide structure of the DNA sequence in question. Many researchers have studied alterations of sequencing by hybridization that account of errors and increase accuracy [13, 27, 28]. Many of the ideas in this section have been developed by the author and submitted for publication in [20].

The first thing we will explore is how having additional spectrum information can reduce the number of reconstructions of the DNA sequence. Specifically, given the k-spectrum of a DNA sequence, we will explore how the knowledge of the sequences

reconstructability with respect to probes of size $t < k$ can help us more accurately perform sequencing using probes of size k.

Secondly we will explore sequencing by hybridization using restriction enzymes. In 1989 Drmanac et al. proposed a modification to sequencing by hybridization which will motivate some discussions in this section. Instead of sequencing an entire DNA sequence one could determine the spectra of short random overlapping fragments of the DNA sequence in question. Such overlaps are known as *clones*. One can then infer the position of these clones within the actual DNA sequence. The clone endpoints would partition the DNA sequence into short intervals called *information fragments*. We would then use the spectra of the information fragments to reconstruct each fragment and hence, the entire DNA sequence [13]. In this section we propose a similar approach for cutting the DNA sequence except we instead use specific restriction enzymes depending on k-spectrum of the DNA sequence in question. In this way, the cuts are more specific and we can limit the cuts to only ones which will decrease the number of reconstructions of the DNA sequence.

5.1 Using Additional Spectrum Information

Note that it is sometimes possible to reduce the reconstruction set obtained from SBH by using some additional information. For example, suppose we have an unknown DNA sequence A which is not uniquely reconstructable using probes of size t. Suppose we also want to perform sequencing using probes of size k for some $k > t$. We could use the knowledge of A's non-unique reconstruction with respect to $S_t(A)$ in order to reduce the reconstruction set obtained from $S_k(A)$.

For example, suppose we know that our DNA sequence A is not uniquely reconstructable with respect to probes of size 3. Using probes of size 4 we obtain

$$S_4(A) = \{AAAC, AACG, ACGA, CGAA, GAAA\}$$

and hence,

$$R_4(A) = \{AAACGAAA, AACGAAAC, ACGAAACG, CGAAACGA, GAAACGAA\}.$$

Since A is not uniquely reconstructable using probes of size 3, we can eliminate or *prune* elements in the reconstruction set which are also not uniquely reconstructable using probes of size 3. Note that $|R_3(ACGAAACG)| = 1$ and $|R_3(CGAAACGA)| = 1$, hence we can reduce our reconstruction set to

$$R_4(A) = \{AAACGAAA, AACGAAAC, GAAACGAA\}.$$

This example was carried out using knowledge of non-unique reconstruction using one smaller probe size. We can extend this notion to knowledge of multiple non-unique reconstruction using probes of sizes $\{p_1, p_2, \ldots, p_l\}$ with $p_i < p_{i+1}$. The algorithm involves checking reconstructability with respect to all probes $p_i \in \{p_1, p_2, \ldots, p_l\}$. When performing checks, we first check to see if the unknown sequence A contains an interleaved $(p_i - 1)$-R-pair and secondly if case 2 of Theorem 5. This is because if A contains an interleaved $(p_i - 1)$-R-pair then it contains and interleaved $(p_j - 1)$-R-pair for all $j < i$.

An obvious question here is whether or not it is possible to prune all but one element of the reconstruction set and thus obtain unique reconstruction. We have shown that when knowledge of non-unique reconstruction with one smaller probe is available this is not possible. We also conjecture that this remains the same when any number of smaller probes have obtained non-unique reconstruction.

Theorem 11. *Let A be an unknown DNA sequence of length n such that $|R_t(A)| \neq 1$. If $R_k(A) = \{s_1, \ldots, s_l\}$ where $k > t$, then there exists two distinct i, j where $1 \leq i, j \leq l$, such that $|R_t(s_i)| \neq 1$ and $|R_t(s_j)| \neq 1$.*

Proof. Suppose instead that there is exactly one $1 \leq i \leq l$ such that $|R_t(s_i)| \neq 1$. Note that all s_j share the same k spectrum and hence none are uniquely reconstructible using probes of size k. By theorem 5 we have that either s_j has an interleaved $(k-1)$-

R-pair or that $s_j|_1^{k-1} = s_j|_{n-(k-1)+1}^{k-1}$. Note that for s_j with $j \neq i$ if s_j were to contain an interleaved $(k-1)$-R-pair then s_j would contain an interleaved $(t-1)$-R-pair, and thus s_j would not be uniquely reconstructible using probes of size t which contradicts our assumption. It must therefore be that $s_j|_1^{k-1} = s_j|_{n-(k-1)+1}^{k-1}$ if $j \neq i$ and s_j does not contain an interleaved $(k-1)$-R-pair.

Consider now the sequence s_i. We know by our previous argument that $s_i|_1^{k-1} = s_i|_{n-(k-1)+1}^{k-1}$ or s_i contains an interleaved $(k-1)$-R-pair. We treat each of these cases separately.

Suppose s_i contains an interleaved $(k-1)$-R-pair $((u,v),(u',v'))$. Let $\{t_1, t_2, \ldots, t_{n-(k-1)+1}\}$ be the set of $(k-1)$-substrings of s_i, that is $t_j = s_i|_j^{k-1}$. Since $((u,v),(u',v'))$ is a $(k-1)$-R-pair in s_i we have that

$$g = t_1 \diamond t_2 \diamond \cdots \diamond t_{u-1} \diamond t_v \diamond t_{v+1} \diamond \cdots \diamond t_{v'-1} \diamond t_{u'} \diamond t_{u'+1} \diamond \cdots$$
$$\diamond t_{v-1} \diamond t_u \diamond t_{u+1} \diamond \cdots \diamond t_{u'-1} \diamond t_{v'} \diamond t_{v'+1} \diamond \cdots \diamond t_{n-(k-1)+1} \tag{5.1}$$

is in the set $R_k(s_i)$ and hence in $R_k(A)$. Note that $((u,v),(u',v'))$ is also an interleaved $(k-1)$-R-pair in g. Also note that $s_i|_u^k \neq g|_u^k$ since (u,v) is rightmost. However, this is a contradiction since we previously established that no element in $R_k(A)$ aside from s_i has a $(k-1)$-R-pair, hence s_i can not contain an interleaved $(k-1)$-R-pair.

Now suppose that $s_i|_1^{k-1} = s_i|_{n-(k-1)+1}^{k-1}$. Now since s_i is not uniquely recon-

structible using probes of size t we have that s_i contains an interleaved $(t-1)$-R-pair or $s_i|_1^{t-1} = s_i|_{n-(t-1)+1}^{t-1}$.

Now consider the first sub-case where $s_i|_1^{t-1} = s_i|_{n-(t-1)+1}^{t-1}$. Since $s_i|_1^{t-1} = s_i|_{n-(t-1)+1}^{t-1}$ and $s_i|_1^{k-1} = s_i|_{n-(k-1)+1}^{k-1}$ then s_i is of the form

$$x_1x_2\cdots x_{t-1}y_1y_2\cdots y_{k-1-2(t-1)}x_1x_2\cdots x_{t-1}z\cdots$$

$$x_1x_2\cdots x_{t-1}y_1y_2\cdots y_{k-1-2(t-1)}x_1x_2\cdots x_{t-1}$$

Now since $s_i|_1^{k-1} = s_i|_{n-(k-1)+1}^{k-1}$ we have that the graph obtained from s_i's k-spectrum contains a cycle and that

$$P = x_2\cdots x_{t-1}y_1y_2\cdots y_{k-1-2(t-1)}x_1x_2\cdots x_{t-1}z\cdots$$

$$x_1x_2\cdots x_{t-1}y_1y_2\cdots y_{k-1-2(t-1)}x_1x_2\cdots x_{t-1}z$$

is a reconstruction based on s_i's k-spectrum. Note that if $z \neq y_1$ then P contains an interleaved $(t-1)$-R-pair and hence not uniquely reconstructible based on its t-spectrum. If $z = y_1$ then the last $t-1$ characters of P are equal the first $t-1$ and we arrive at the same conclusion. Hence P is in $R_k(A)$ but $R_t(P) \neq 1$, which contradicts our assumption.

We treat the second sub-case where s_i contains an interleaved $(t-1)$-R-pair given by $((u,v),(u',v'))$. Let t_i be the ith k-mer of s_i, or in other words, $t_j = s_i|_j^k$. If $u \neq 1$

then

$$t_2 \diamond \cdots \diamond t_n \diamond t_1$$

also contains an interleaved $(t-1)$-R-pair. Also the above sequence is a reconstruction of s_i based on its k-spectrum. In addition this sequence is not equal to s_i because if it were then it would imply all k-tuples in s_i would be equal. Now if $u = 1$ and $v' \neq n - k + 1$ then

$$t_n \diamond t_1 \diamond \cdots \diamond t_{n-1}$$

also contains an interleaved $(t-1)$-R-pair and is a reconstruction of s_i based on its k-spectrum. Similarly it is not equal to s_i. Finally, lets consider the case where $u = 1$ and $v' = n - k + 1$. We assume first that $v \leq n - k + 1$ with repeat v occurring in tuple $t_{f(v)}$ then we have that

$$t_{f(v)} \diamond t_{f(v)+1} \diamond \cdots \diamond t_{f(v)-1}$$

contains an interleaved $(t-1)$-R-pair. Also the above sequence is not equal to s_i because (u, v) is rightmost, hence $t_1 \neq t_{f(v)}$. So have that there is another sequence in $R_k(s_i)$ that is not uniquely reconstructible using probes of size t. We now assume that $v > n - k + 1$, hence v occurs in the last k-tuple but does not occur at the beginning of the tuple. Note that if no characters of the $(t-1)$-mer at u' occurs in the last k-tuple (in t_{n-k+1}) then the above sequence will also contain an interleaved

$(t-1)$-R-pair. If, however, some of the characters of that substring occurs in the last k-tuple then we simply replace u' with $k-t+1$. Note that $((u,v),(k-t+1,v'))$ will be an interleaved $(t-1)$-R-pair since $s_i|_1^{k-1} = s_i|_{n-(k-1)+1}^{k-1}$. Also note that the $(t-1)$-mer occurring at position $k-t+1$ will have no characters in the last k-tuple, so we can apply the above transformation to obtain another sequence in $R_k(s_i)$ with an interleaved $(t-1)$-R-pair, and hence, not uniquely reconstructible using probes of size t. □

We also show that if a DNA sequence contains an interleaved $(k-1)$-R-pair then it is impossible to prune all reconstructions using any number of smaller probe sizes. This result is presented in the following theorem.

Theorem 12. *Let A be an unknown DNA sequence of length n with $|R_{k_l}| \neq 1$ for $k_l \in I \subset \mathbb{N}$ and let $r = t_0 \diamond t_1 \diamond \cdots \diamond t_{n-k}$ be an element of $R_k(A)$ with k-tuples t_j, where $0 \leq j \leq n-k$ and $k > k_l$ for $k_l \in I$. If there exists an element $r' \in R_k(A)$ such that $r' \neq t_{i \pmod{n-k+1}} \diamond t_{i+1 \pmod{n-k+1}} \diamond \cdots \diamond t_{i+n-k \pmod{n-k+1}}$ for all integers i, then for each $k_l \in I$ there exists at least two distinct elements $y, z \in R_k(A)$ such that $|R_{k_l}(y)|, |R_{k_l}(z)| \neq 1$.*

Proof. The existence of such an r' implies that A contains an interleaved $(k-1)$-R-pair. Using transformation 5.1 in Theorem 11 we obtain a second sequence r''

with the same k spectrum that contains an interleaved $(k-1)$-R-pair and hence not uniquely reconstructible using probes of size $t_i \in I$. □

Based on these theorems we can see that pruning is only really effective if case 2 of Theorem 5 is satisfied, that is, the last $k-1$ characters of the sequence match the first $k-1$ characters. It is important to note that this doesn't happen very often and as the size of the DNA sequence increases the probability of this diminishes. In [20] we implemented these algorithms in Python and tested them on 100 random DNA sequences of different lengths which are not uniquely reconstructable using probes of both sizes k and $t < k$. The results of this program do indeed support our observations.

N	t	k	Average reduction via pruning
15	4	6	2.23
25	4	6	0.49
30	4	6	0.41
40	4	6	0
15	5	7	2.81
25	5	7	1.82
30	5	7	1.5
40	5	7	0

Figure 5.1: The average number of DNA sequences pruned from the k-reconstruction set of a random sample of 100 DNA sequences of length N. The DNA sequences in the sample are not reconstructible using probes of sizes k and some $t < k$.

5.2 Using Restriction Enzymes

In this section we will discuss how we can use restriction enzymes to obtain more accurate results with SBH without having to increase the probe size. In [13, 28] it was shown how obtaining the spectrum of shorter sub fragments of the DNA sequence in question can yield more accurate sequencing. We will use a similar approach in

this section except we will use specific restriction enzymes depending on the spectrum of the DNA sequence in question.

We first assume that we have a library L storing all possible cut configurations of the restriction enzymes we have at our disposal. For each pair of reconstructions r_1 and r_2 we split r_1 and r_2 into pieces using a common cut configuration from L corresponding to a specific restriction enzyme. We then check to see what is the smallest substring that occurs in a piece from one reconstruction but not the other. We continue this process for all $x \in L$ and take the smallest such substring. We then run a probe through the corresponding pieces to invalidate r_1 or r_2 as a candidate for the unknown DNA sequence. The reason for taking the minimal length substring is to reduce the size of the probe array we must use.

As an example of this algorithm, consider the 3-spectrum $S_3(A) = \{AAA, AAA, AAT, ATG, TGA, GAT, ATC, TCA, CAC, ACA, CAT\}$. The reconstructions based on $S_3(A)$ are $r_1 = AAAATCACATGAT$ and $r_2 = AAAATGATCACAT$. If use a restriction enzyme which cuts the sequence after the occurrence of TCA we obtain the $AAAATCA/CATGAT$ from r_1, and $AAAATGATCA/CAT$ from r_2. Note that the suffixes are the smallest pieces so we compare them. We can see that they differ by a G nucleotide, so if we run a probe detecting the G nucleotide we can determine which of r_1 or r_2 is the actual unknown DNA sequence A.

A program was developed which generates random DNA sequences of length N that are not uniquely reconstructable using probes of size k. The algorithm is then carried out on the DNA sequences k-reconstruction set using probes of size less than or equal to k. The only sequences tested by our algorithm are those which contain a recognition sequence corresponding to some restriction enzyme in our library L.

N	k	Average percentage of reconstruction set pruned	Percentage solved
50	5	53.8096320346	67
50	6	40.4777777778	73
100	6	53.8986721612	69
100	7	44.5833333333	83
150	7	51.6166666667	83
150	8	44	81

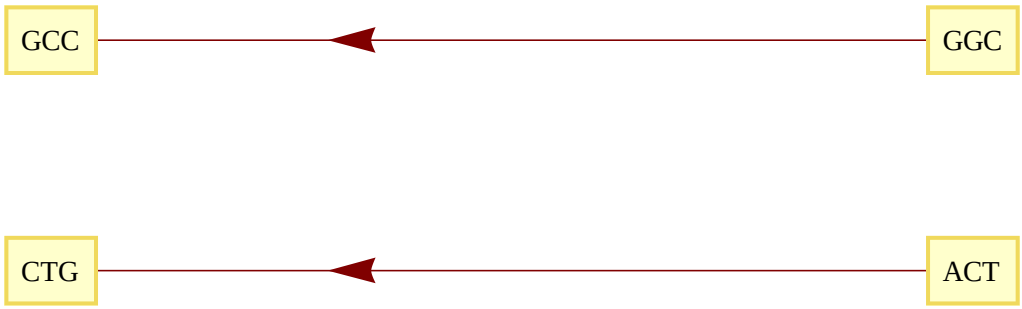
Figure 5.2: The average percentage of the k-reconstruction set pruned and average percentage of instances solved of a random sample of 100 DNA sequences of length N using the second pruning algorithm. Restriction enzymes used: EcoRI, EcoRII, BamHI, HindIII, TaqI, NotI, HinfI, Sau3A, PvuII, SmaI, HaeIII, AluI, EcoRV

The above algorithm can also be extended to account for errors in the k-spectrum. As mentioned in Section 3.3, there are two types of errors which can occur in SBH.

False positive errors refer to the event when a k-mer occurs in the k-spectrum, $S_k(A)$, of the unknown DNA sequence A, but does not actually occur in A. Similarly a false negative error refers to the event when a k-mer does not show up in $S_k(A)$ but does in fact occur in the DNA sequence A. We now define our problem definition for SBH. Our definition is an extension to Problem 2 in Section 3.3. We define the problem of DNA sequencing by hybridization with multiple probe sizes and false negative errors as follows:

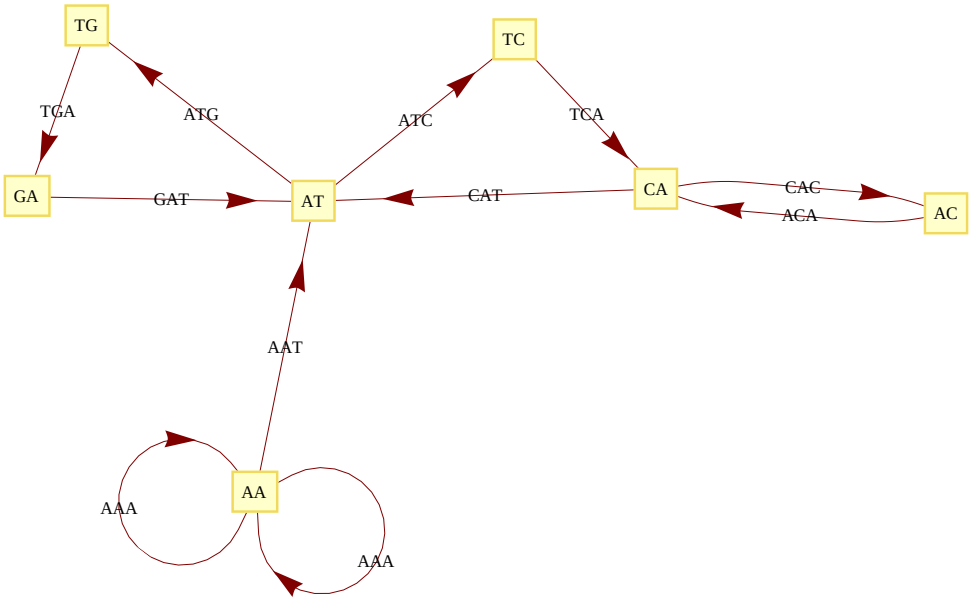
Problem 15. Given the multiset $S_k^-(A) \subseteq S_k(A)$ and $|R_{t_i}(A)| \neq 1$ for all $i \in \{1, \ldots c\}$, determine all sequences A' such that A' contains all the k-mers present in $S_k^-(A)$ as well as up to Δ additional k-mers, all of which are not in $S_k^-(A)$ and A' satisfies Theorem 5 for all probes of size t_i.

We additionally make the assumption that $A|_1^k, A|_n^k$, the first and last k-mers of A, are in $S_k^-(A)$ and that any element $y \in S_k^-(A)$ has the correct multiplicity with respect to A. In other words, y occurs in A as many times as it occurs in $S_k(A)$. It is important to note that a false negative error can impact the graph generated from $S_k(A)$ in a number of different ways. As an example, suppose $S_3(A) = \{ACT, CTG, TGG, GGC, GCC\}$. One can infer from $S_3(A)$ that $A = ACTGGCC$. However, if a false negative error occurred one might have $S_k^-(A) = \{ACT, CTG, GGC, GCC\}$. When building the graph using the Hamiltonian approach we would obtain



which contains two components.

Consider now the spectrum $S_3(A) = \{AAA, AAA, AAT, ATG, TGA, GAT, ATC, TCA, CAC, ACA, CAT\}$. Using the Eulerian approach for SBH we obtain the graph



Note that if instead we had a 3-mer missing, such as ATG, we would obtain a graph with no Eulerian trail, and hence no reconstructions would be possible.

We define k-mers v to be a t-successor of u if the last $(k-t)$ characters of u match the first $(k-t)$ characters of v, where $1 \leq t \leq \Delta + 1$. Now, let w be the length $(k+t)$ string obtained by overlapping the last $(k-t)$ characters of u with the first $(k-t)$ characters of v. Any k-length string in w that is not in $S_k^-(A)$ is referred to as an artificial k-mer. Define $S_k^-(A)_\Delta$ as the set the set of all k-mers from $S_k(A)$ union with the set of all artificial k-mers generated from every pair of k-mers in $S_k^-(A)$.

This is to correct for the possible loss of multiplicity information when false negative errors occur. In the normal circumstance, we would also let each distinct element in $S_k^-(A)_\Delta$ have a Δ increase in multiplicity in comparison to $S_k^-(A)$, however, because of our previous assumption that each element $x \in S_k^-(A)$ has correct multiplicity, this is not necessary.

It is easily seen that if the number of false negative errors Δ is such that $\Delta < k$ then the spectrum $S_k^-(A)_\Delta$ will contain the unknown DNA sequence [35]. Hence, a solution to the above problem in the case where $\Delta < k$ would be to compute $S_k^-(A)_\Delta$, generate the list of reconstructions based on $S_k^-(A)_\Delta$, and prune such reconstructions based on probes of size t_i $(1 \leq i \leq c)$ using the algorithms in the previous section.

Note that pruning with algorithm 2, errors make no difference. So the method is valid. When pruning with the other pruning algorithm we must note that the algorithm does not account for errors, and errors can cause the algorithm to produce incorrect pruning of the reconstruction set. To account for this, we again split each pair of reconstructions into pieces using restriction enzymes but instead we look for the smallest set of $2\Delta + 1$ substrings that occur in one piece but not the other. Since there can be at most Δ errors, when we run probes through each piece using two sequencing projects, the piece which has the $2\Delta + 1$ substrings will have the most matches with our probes.

Let $S_k^+(A)$ be any superset of $S_k(A)$, where $|A| = n + k - 1$ and A is a DNA sequence. We again use Δ as an upper bound on the number of errors, which in this case, is the maximum number of elements in $S_k^+(A)$ that do not occur in the DNA sequence. Here, our definition is an extension of Problem 3 in Section 3.3. The problem of DNA sequencing by hybridization with multiple probes sizes with false positive errors is defined as follows:

Problem 16. Given the multi set $S_k^+(A) \supseteq S_k(A)$ and $|R_{t_i}(A)| \neq 1$ for all $i \in \{1, \ldots c\}$, determine all sequences A' such that A' contains all but at most Δ of the k-mers present in $S_k^+(A)$ and A' satisfies Theorem 5 for all probes of size t_i.

We also make the assumption that if $x \in S_k^+(A)$ occurs in the unknown DNA sequence, then it occurs as many times as it occurs in $S_k^+(A)$. The problem can be solved by generating the Eulerian method graph G based on $S_k^+(A)$ and finding all trails T in the graph G such that the number of edges e in T is such that $|S_k^+(A)| - \Delta \leq t \leq |S_k^+(A)|$, and each edge e belonging to trail T occurs as many times in T as it does in $S_k^+(A)$, or it does not occur at all. If we did not make the above assumption, then it would not be necessary for each edge e to occur in T as many times as it does in $S_k^+(A)$. We then applying our pruning algorithms on each reconstruction generation from each trail.

We can further extend this method to solve the SBH problem with both positive

and negative errors. We define the spectrum with both errors $S_k^{\pm}(A)$ of an unknown DNA sequence A of length $n+k-1$ in such a way that A_1^k, A_n^k, the first and last k-mers of A, are in $S_k^{\pm}(A)$.

Problem 17. Given $S_k^{\pm}(A)$ determine all sequences A' such that the sum of the number of elements in A' but not in $S_k^{\pm}(A)$ and the number of elements in $S_k^{\pm}(A)$ but not in A' is less than or equal to Δ.

We also assume that any element $x \in S_k^{\pm}(A)$ with multiplicity c occurs either c times in the unknown DNA sequence, or not at all. In other words if either a false positive or false negative error occurs it will manifest itself in the spectrum as an additional element (with some multiplicity) that does not occur in $S_k(A)$, or as the removal of an element that occurs in $S_k(A)$ and its multiplicity reduced to zero. The problem can be solved by first computing the set $S_k^{\pm}(A)_{\Delta}$ (which we define in the same manner as we did for $S_k^{-}(A)$). We then generate the Eulerian method graph G based on $S_k^{\pm}(A)_{\Delta}$. We then find all trails T in G such that the sum of the artificial k-mers used in the trail and the k-mers not used in $S_k^{\pm}(A)$ is less than or equal to Δ. Also note that as with the solution to the previous problem, any edge e in trail T must occur in T as many times as it does in $S_k^{\pm}(A)$, or does not occur at all. Finally we run both pruning algorithms on the reconstructions to reduce the size of the reconstruction set.

A program was written which implements this algorithm using 100 random DNA sequences. We keep track of the average percentage of the k-reconstruction set that is pruned, as well as the percentage of the 100 instances that are solved uniquely after pruning.

N	k	Δ	Average percentage of reconstruction set pruned	Percentage solved
50	5	1	58.5563721336	55
50	6	1	50.2801226551	66
100	6	1	60.7826569264	64
100	7	1	51.4694444444	69
150	7	1	55.4095238095	72
50	5	2	59.3653743672	32
50	6	2	52.0017838601	43
100	6	2	64.0176245178	42
100	7	2	53.3569069819	57
150	7	2	61.9327468382	48

Figure 5.3: The average percentage of the k-reconstruction set pruned and average percentage of instances solved of a random sample of 100 DNA sequences of length N using the second pruning algorithm. Restriction enzymes used: EcoRI, EcoRII, BamHI, HindIII, TaqI, NotI, HinfI, Sau3A, PvuII, SmaI, HaeIII, AluI, EcoRV

Chapter 6

The DAG-Width of DNA Graphs

In this chapter we will investigate the width of the graphs obtained by DNA sequencing by hybridization. Graph widths are important because they have many applications in finding efficient graph algorithms. Specifically, if one can find constant upper bounds on certain graph width properties, many NP-complete problems restricted to those graphs can be solved in polynomial time [6, 10].

We will explore the DAG-width of the DNA graphs obtained from sequencing by hybridization of DNA sequences of length $N = n + k - 1$ using probes of size k. Similar to the previous chapter, we assume that the k-spectrum $S_k(A)$ is a multi set with each k-mer of A occurring in $S_k(A)$ as many times as it occurs in A. We also assume that the graphs of interest were obtained using the Hamiltonian approach to

SBH.

An important thing to note is that the DAG-width of the associated DNA graph can vary greatly depending on the structure of the DNA sequence A. This is obvious because some DNA graphs can contain cycles, whereas others may not. Any acyclic graph would have a DAG-width of 1 whereas any graph with cycles would have a DAG-width of at least 2 by Corollary 1. Since the width can vary, even when the parameters N and k are kept constant, we must instead analyze the width properties using probabilistic models.

We first address the issue of cycles in the digraph G corresponding to a particular k-spectrum, $S_k(A)$, of an unknown DNA sequence A. It can easily be shown that there are specific properties of DNA sequences which give rise to cycles in their associated digraphs. Specifically, $k-1$ repeats in the unknown sequence can be directly attributed to such cycles.

Lemma 5. *Let A be an unknown DNA sequence and let G be the associated graph obtained using Hamiltonian SBH with probes of size k. Then G contains a cycle if and only if A contains a $k-1$ repeat.*

Proof. First suppose that the unknown DNA sequence A contains a $k-1$ repeat at positions (i,j) where $i<j$. Note that there are vertices $v_i, v_{i+1}, \ldots, v_{j-1}$ in our digraph with labels $l(v_i) = a_i a_{i+1} \cdots a_{i+k-1}, l(v_{i+1}) = a_{i+1} a_{i+2} \cdots a_{i+k}, \ldots, l(v_{j-1}) =$

$a_{j-1}a_j \cdots a_{j+k-2}$ respectively. Note that for each $t \in \{i, i+1, \ldots, j-2\}$, the last $k-1$ characters of v_t match the first $k-1$ characters of v_{t+1}, hence there is a path through vertices $v_i, v_{i+1}, \ldots, v_{j-1}$. Since (i, j) is a repeat we have that $a_i a_{i+1} \cdots a_{i+k-2} = a_j a_{j+1} \cdots a_{j+k-2}$. This implies that the first $k-1$ characters of v_i match the last $k-1$ characters of v_{j-1}, which in turn implies there is an edge from v_{j-1} to v_i. Adding this edge to our path through $v_i, v_{i+1}, \ldots, v_{j-1}$ we obtain a cycle through the vertices.

Now suppose that G contains a cycle through vertices $v_i, v_{i+1}, \ldots, v_{j-1}$. This implies that the first $k-1$ characters of v_i match the last $k-1$ characters of v_{j-1} which in turn implies a $k-1$ repeat in A. □

We now let $p_d(n, k)$ denote the probability that the DAG-width of the graph obtained using the Hamiltonian approach of SBH with probes of size k from a random DNA sequence of length $N = n + k - 1$ is d.

Theorem 13. *$p_1(n, k) = 1 - \Theta(n^2/4^{k-1})$ for $2(k-1) \leq n \leq 2^{k-1}$.*

Proof. Let $p_{>1}(n, k)$ denote the probability that the DAG-width is greater than 1. By the rules of complementary probability we have that $p_1(n, k) = 1 - p_{>1}(n, k)$. By Corollary 1 we have that the DAG-width is greater than 1 if and only if the graph contains a cycle. By Lemma 5 we have that the associated DNA graph contains a cycle if and only if the unknown DNA sequence has a $k-1$ repeat. Note that the

probability that (i,j) is a $k-1$ repeat is $1/4^{k-1}$. There are $\binom{n}{2}$ ways to select indices i and j such that (i,j) is a $k-1$ repeat. It follows that

$$p_{>1}(n,k) \leq \binom{n}{2}\frac{1}{4^{k-1}} = O(n^2/4^{k-1})$$

For the second part of the proof we consider 2 disjoint sub-intervals of $[1, n+k-1]$ $I_1 = \left[1, \lfloor \frac{N}{2} \rfloor - (k-1)\right]$ and $I_2 = \left[\lfloor \frac{N}{2} \rfloor + 1, n\right]$. We now let the event Z be the event that there is a repeat (i,j) for some $i \in I_1$ and $j \in I_2$. We also let Z_α be the event that $\alpha = (i',j')$ is a repeat for $i' \in I_1$ and $j' \in I_2$. We now let $Y_\alpha = Z_\alpha \wedge \bigwedge_{\beta \in (I_1 \times I_2)\setminus\{\alpha\}} \bar{Z}_\beta$. Note that the events Y_α are disjoint so $\Pr(Y_{\alpha_1} \cap Y_{\alpha_2}) = 0$ for all α_i, hence $\Pr(Z) \geq \Pr\left(\bigvee_{\alpha \in I_1 \times I_2} Y_\alpha\right) = \sum_{\alpha \in I_1 \times I_2} \Pr(Y_\alpha)$. By Lemma 3.5 of [28] we have that

$$\begin{aligned} \Pr(Y_\alpha) &\geq \Pr(Z_\alpha)\left(1 - \sum_{\beta \in (I_1 \times I_2)\setminus\{\alpha\}} \Pr(Z_\beta | Z_\alpha)\right) \\ &= (1/4^{k-1})\left(1 - (\lfloor \frac{N}{2} \rfloor - k + 1)^2 (1/4^{k-1})\right). \end{aligned}$$

This in turn implies that

$$p_{>1}(n,k) \geq \Pr(Z) \geq (\lfloor \frac{N}{2} \rfloor - (k-1))^2 (1/4^{k-1})(1 - (\lfloor \frac{N}{2} \rfloor - (k+1))^2 (1/4^{k-1})) = \Omega(n^2/4^{k-1})$$

□

From this Theorem we can easily see that there is a high probability that the DNA graphs obtained from SBH using the Hamiltonian approach will have a DAG-width

of 1 for many values of n and k. This is advantageous because it allows for efficient algorithms to solve the Hamiltonian path problem.

Another well known decomposition for digraphs is the *arboreal decomposition* introduced in [17]. This decomposition has an associated width known as the *directed tree-width*. In [6] it was shown that if a digraph has DAG-width k, then its directed tree-width is at most $3k + 1$. It was also shown in [17] that the Hamiltonian path problem can be solved in polynomial time on digraphs of directed-tree width bounded by a constant.

Proving that the probability of the DAG-width being 1 is $1 - \Theta(n^2/4^{k-1})$ is therefore equivalent to proving that the probability of the directed tree-width being at most 4 is $1 - \Theta(n^2/4^{k-1})$. Although this is not an upper bound in the strict sense, it implies that the majority of the time we obtain a small directed tree-width and hence, polynomial time solvability of the Hamiltonian path problem.

Chapter 7

Next Generation Sequencing by Hybridization

As discussed in the previous chapters of this thesis, the major problem with SBH is the existence of non-unique reconstruction. Many enhancements have been put forward to help deal with this problem.

One such enhancement was introduced in [13] by Drmanac et al. In this enhancement, the target sequence is fragmented into random, overlapping subsequences known as *clones*. If the clones overlap to a large degree then their spectra would be very similar. This allows us to determine the clone positions in the target sequence. The endpoints of the clones create a partition of the target sequence. The DNA

subsequences between these endpoints are known as *information fragments*. We then obtain the spectra of the information fragments and attempt to reconstruct them using their spectra. In doing this we also reconstruct the target sequence.

Example 7. Suppose we have the unknown DNA sequence A. We perform sequencing using the above enhancement. We obtain clones with spectra $S_3(C_1) = \{ACT, CTA, TAG, AGT, GTT, TTA\}$, $S_3(C_2) = \{TAG, AGT, GTT, TTA, ACT, CTC\}$, $S_3(C_3) = \{TTA, TAC, ACT, CTC, TCT, CTG\}$. We assume that the clone positions on the target sequence are known and that we have

$$A = \underbrace{a_1a_2a_3a_4a_5}_{C_3}\underbrace{a_6a_7a_8}_{C_2}a_9a_{10}a_{11}\underbrace{a_{12}a_{13}}_{C_1}$$

Obviously, the higher the number of clones the greater the intersection between clones and hence the small the length of the information fragments. In this example we obtain the following information fragments.

$$A = \underbrace{a_1a_2}_{I_1}\underbrace{a_3a_4a_5}_{I_2}\underbrace{a_6a_7a_8}_{I_3}\underbrace{a_9a_{10}a_{11}}_{I_4}\underbrace{a_{12}a_{13}}_{I_5}$$

We now reconstruct each of the clones. If this can be done uniquely we need not worry about the spectra of the information fragments. We obtain the following DNA graphs using the Hamiltonian approach.

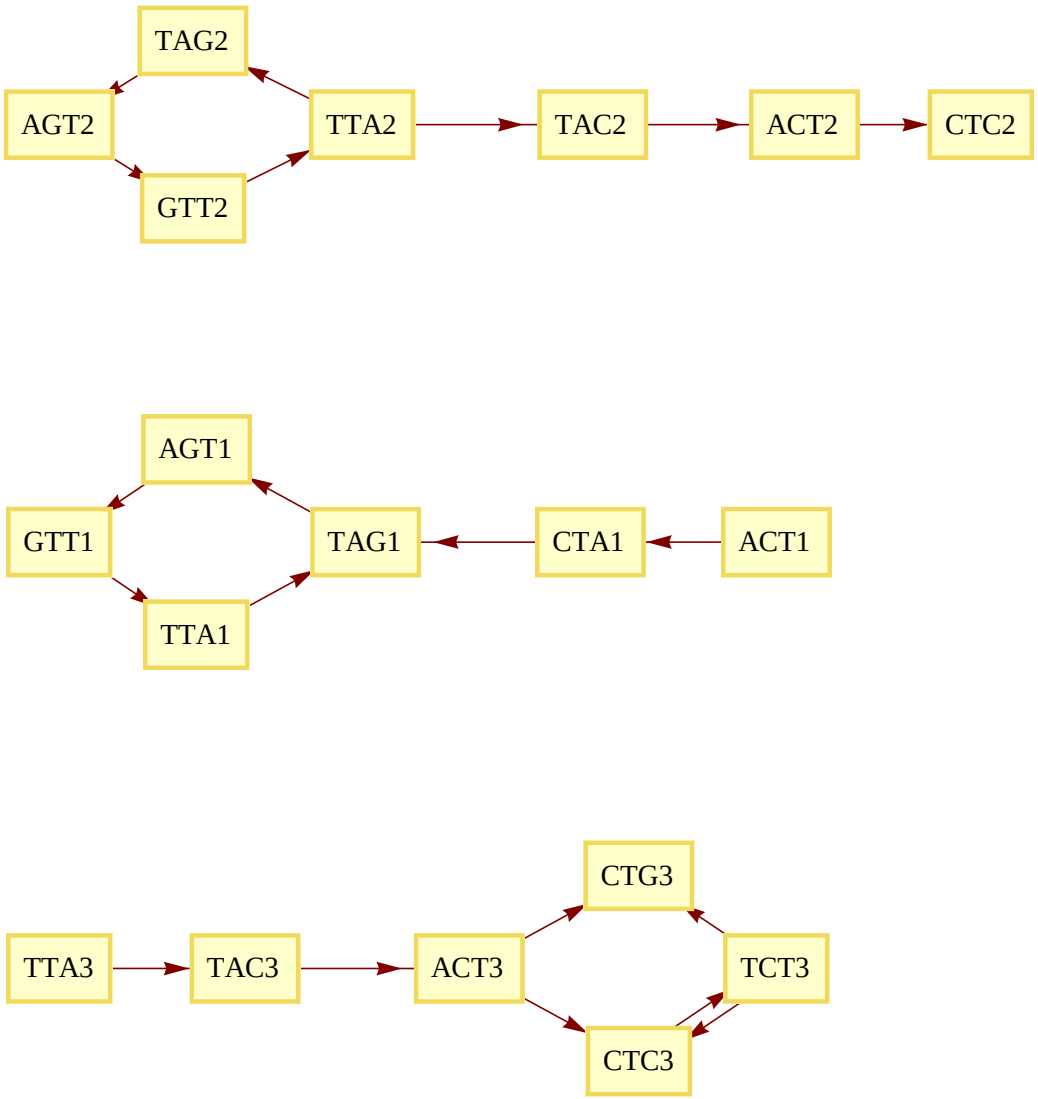


Figure 7.1: Graphs generated using the spectra $S_3(C_1)$, $S_3(C_2)$ and $S_3(C_3)$.

We have that $C_1 = ACTAGTTA$, $C_2 = TAGTTACTC$, and $C_3 = TTACTCTG$ and hence $A = ACTAGTTACTCTG$.

In [13] Drmanac et al. used simulations to test this method of sequence on target sequences with 10^6 nucleotide bases. This was done by obtaining clones which were 500 base pairs long. The probabilistic models discussed in Section 4 can be extended to account for this next generation SBH, as was shown in [28]. The probability for non-unique reconstruction of a DNA sequence of length $N = n + k - 1$ with probes of size k is $\Theta(d^3 n/4^{2k})$, where d is the length of the information fragments obtained by cutting the target sequence. These probability models were also extended to account for the presence of false negative errors [28].

Chapter 8

Conclusions and Open Problems

In this thesis we investigated several aspects of DNA sequencing by hybridization. We highlighted various results by researchers, both classic and contemporary, and expanded on several notions.

In the second section we examined the different variants on the SBH problem and how it is often studied in different contexts [15]. We highlighted how the spectrum can be considered to be either a set, in which the multiplicity of k-mers is unknown, or a multi set, in which it is assumed that knowledge of the multiplicity of k-mers is known. We also examined the role of errors in SBH. We introduced the notions of positive and negative errors.

The third section of the thesis examines the computational complexity of DNA

sequencing by hybridization. The results of [7] were summarized showing how the Hamiltonian approach to SBH with no errors is NP-hard, due to the NP-hardness of the Hamiltonian path problem in directed graphs. It was later shown that the problem of SBH with no errors is solvable in polynomial time when we use the Eulerian approach to SBH and when only a single reconstruction is of interest [25]. When we assume the spectrum associated with a particular sequence can contain either positive or negative errors, the problem then becomes NP-hard, regardless of which method we use [7].

In the fourth section we examined were the probabilistic models involved in SBH. In [1,2,14] probabilistic models were constructed to determine the likelihood of success with SBH on a DNA sequence of length N using probes of size k. This was later expanded to include next generation sequencing whereby the target sequence is cut into fragments [28]. We also introduce conditional probabilistic models that predict the likelihood of sequencing failure for probes of size k, given that a sequencing failure has already occurred with probes of size $t < k$. A further direction of research is whether this can be extended to include more than one previous failure with more than one probe size.

In the fifth section we discussed extensions of SBH. Specifically we looked at SBH with additional spectrum information and the use of restriction enzymes in SBH. Re-

striction enzymes are used in order to cut the DNA sequence at points and determine the spectrum of shorter sub fragments of the DNA sequence in question. We introduce algorithms here that make use of a library of restriction enzymes and known cut configurations to obtain the best cutting strategy in order to reduce ambiguity in sequencing. We also provide computer simulations of the algorithms which demonstrate their performance in reducing ambiguity. A further direction of research here would be to develop probabilistic models to determine the likelihood that the algorithms will produce unique reconstructions of the unknown DNA sequence.

The sixth section focused on examining the DAG-width of the graphs obtained from SBH. The DAG decomposition of a directed graph is a relatively new graph decomposition. The decomposition is analogous to the tree decomposition for undirected graphs [6]. The DAG-width of a DNA graph can vary greatly depending on the DNA sequencing in question, even if the parameters N and k are kept constant. We studied probabilistic models of the DAG-width of DNA graphs obtained from SBH using the Hamiltonian approach. We showed that there is a high probability of the DAG-width being 1 and hence we can usually find Hamiltonian paths in the associated DNA graphs efficiently. A further direction of research here would be to find the probability that the DAG-width is d for $d > 1$.

In the final section we discussed next generation SBH techniques. Specifically, we

discussed a method where the DNA sequence is randomly split into fragments and sequencing is performed on the individual fragments.

Appendix A

Algorithms

Algorithm 1 Monte Carlo simulation of $P(n, k, t)$

Require: Parameters n, k, t and *sample_size*.
Ensure: Simulated value of $P(n, k, t)$ based on a random sample of DNA sequences of length $N = n + k - 1$.

```
num := 0, nonreconstructs := 0
while num < sample_size do
    Randomly generate a DNA sequence A of length N = n + k − 1
    if A is not reconstructible using probes of size t then
        num := num + 1
        if A is not reconstructable using probes of size k then
            nonreconstructs := nonreconstructs + 1
        end if
    end if
end while
return nonreconstructs/num.
```

Algorithm 2 Pruning Algorithm 1: Carries out pruning algorithm 1

Require: Reconstructions R and integers $T = \{t_1, t_2, \dots t_c\}$ with $t_i < t_{i+1}$ such that A of length N is not uniquely reconstructible using probes of size t_i for $i \in \{1, \dots, c\}$.

Ensure: A list of candidates for A

for Each $r \in R$ **do**
  for $j = c$ to 1 **do**
    if r contains an interleaved $(t_j - 1)$-R-pair **then**
      Break
    else if $r_1^{t_j-1} \neq r_{N-t_j+2}^{t_j-1}$ **then**
      Remove r from R
      Break
    end if
  end for
end for
return R

Algorithm 3 Pruning Algorithm 2 w/ errors: Carries out pruning algorithm 2 in the presence of false negative errors.

Require: Reconstructions $R = \{r_1, r_2, \ldots, r_M\}$, cut configurations L and maximum false negative error bound Δ

Ensure: A list of candidates for A

for Each pair $r_i, r_j \in R$ **do**
  $smallest := \infty$
  $\{cut\_num, S\_occurrence, S\_non\_occurrence, c\} := \{NULL, NULL, NULL, NULL\}$
  $P := NULL$
  for Each cut configuration $x \in L$ **do**
    Cut r_i into disjoint substrings $i_1, \ldots, i_u$ and r_j into disjoint substrings $j_1, \ldots, j_v$ based on x
    if $u \neq v$ **then**
      Cut the unknown DNA sequence using restriction enzyme x. If it does not get cut into u sequences remove r_i from R. Also if it does not get cut into v sequences remove r_j from R.
      Return control to outer most loop.
    end if
    for Each pair pair i_k, j_k **do**
      Determine the set S of size $2\Delta + 1$ with the smallest maximum length string such that all the element of S occurs in i_k and not j_k (or j_k and not i_k)
      if $MaxLength(S) < smallest$ **then**
        $smallest := MaxLength(S)$
        if All element of S occur in i_k and none in j_k **then**
          $\{cut\_num, S\_occurrence, S\_non\_occurrence, c\} := \{k, i, j, x\}$
        else
          $\{cut\_num, S\_occurrence, S\_non\_occurrence, c\} := \{k, j, i, x\}$
        end if
        $P := S$
      end if
    end for
  end for
  Obtain the cut_num^{th} substring s' from the unknown sequence upon cutting with the restriction enzymes specified by c
  Run all probes in P through s'
  if More than Δ probes occur in s' **then**
    Remove $r_{S\_non\_occurrence}$ from R
  else
    Remove $r_{S\_occurrence}$ from R
  end if
end for
return R

Algorithm 4 MaxLength(S): Determines the length of the largest string in the set of strings S.

Require: Set S of strings
Ensure: The largest string in S
1: $largest := 0$
2: **for** Each string $x \in S$ **do**
3: **if** $largest < Length(x)$ **then**
4: $largest := Length(x)$
5: **end if**
6: **end for**
7: **return** $largest$

Algorithm 5 SBH with errors: Carries out SBH using pruning algorithms 1 and 2 in the presence of errors.

Require: The error k-spectrum $S_k^{\pm}(A)$ of an unknown DNA sequence A, error bound Δ, and integers $T = \{t_1, t_2, \ldots, t_c\}$ with $t_i < t_{i+1}$ such that A is not uniquely reconstructible using probes of size t_i for t_i for $i \in \{1, \ldots, c\}$.
Ensure: A list of candidates for A
1: Compute $S_k^{\pm}(A)_{\Delta}$
2: Generate the Eulerian method graph G based on $S_k^{\pm}(A)_{\Delta}$
3: Find all reconstructions R of A based on G. Each trail T in the graph corresponding to a reconstruction with m artificial k-substrings and do not contain M elements in $S_k^{\pm}(A)$ must be such that $m+M \leq \Delta$. Each edge e in trail T much occur in T as many times as its corresponding k-substring occurs in $S_k^{\pm}(A)_{\Delta}$.
4: $R :=$ Result of algorithm 2 with input R and T.
5: $R :=$ Result of algorithm 3 with input R, some known cut configuration library L, and Δ
6: **return** R

Bibliography

[1] R. Arratia, B. Bollobás, and D. Coppersmith, *Euler circuits and DNA sequencing by hybridization*, Discrete Applied Mathematics **104** (2000), 63–96.

[2] R. Arratia, D. Martin, G. Reinert, and M. S. Waterman, *Poisson process approximation for sequence repeats, and sequencing by hybridization*, Journal of Computational Biology **3** (1996), no. 3, 425–463.

[3] W. Bains and G.C. Smith, *A novel method for nucleic acid sequence determination*, Journal of Theoretical Biology **135** (1988), 303–307.

[4] G. Beards, `https://en.wikipedia.org/wiki/File:Sickle_cells.jpg`, 2012.

[5] J. Bertram, *The molecular biology of cancer*, Molecular Aspects of Medicine **21** (2000), no. 6, 167–223.

[6] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, and J. Obdržálek, *The DAG-width of directed graphs*, Journal of Combinatorial Theory, Series B **102** (2012), no. 4, 900–923.

[7] J. Blazewicz and M. Kasprzak, *Complexity of DNA sequencing by hybridization*, Theoretical Computer Science **290** (2003), no. 3, 1459–1473.

[8] N. Broude, T. Sano, C. Smith, and C. Cantor, *Enhanced DNA sequencing by hybridization*, Proceedings of the National Academy of Sciences USA, vol. 91, 1994, pp. 3072–3076.

[9] International Human Genome Sequencing Consortium, *Initial sequencing and analysis of the human genome*, Nature **409** (2001), 860–921.

[10] B. Courcelle, *The monadic second-order logic of graphs. I. recognizable sets of finite graphs*, Information and Computation **85** (1990), no. 1, 12–75.

[11] F. Crick, *Central dogma of molecular biology*, Nature **227** (1970), 561–563.

[12] F. Crick and J. Watson, *A structure for deoxyribonucleic acid*, Nature **171** (1953), 737–738.

[13] R. Drmanac, I. Labat, I. Brukner, and R. Crkvenjakov, *Sequencing of megabase plus DNA by hybridization*, Genomics **4** (1989), 114–128.

[14] M. E. Dyer, A. M. Frieze, and S. Suen, *The probability of unique solutions of sequencing by hybridization*, Journal of Computational Biology **1** (1994), 105–110.

[15] P. Formanowicz, *DNA sequencing by hybridization with additional information available*, Computational Methods in Science and Technology **11** (2005), no. 1, 21–29.

[16] J. Gallant, D. Maier, and J. A. Storer, *On finding minimal length superstrings*, Journal of Computer and System Sciences **20** (1980), no. 1, 50–58.

[17] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas, *Directed tree-width*, Journal of Combinatorial Theory, Series B **82** (2001), no. 1, 138–154.

[18] W. Klug, M. Cummings, and C. Spencer, *Concepts of genetics, eighth edition*, Pearson Education, Inc., 2006.

[19] Y. Lysov, V. Floretiev, A. Khorlyn, K. Khrapko, V. Shick, and A. Mirzabekov, *DNA sequencing by hybridization with oligonucleotides*, Dokl. Acad. Sci. **303** (1988), 1508–1511.

[20] M. Mata-Montero, N. Shalaby, and B. Sheppard, *DNA sequencing by hybridization with restriction enzymes*, Submitted to the Journal of Discrete Algorithms, 2013.

[21] ———, *Probabilistic models for sequencing by hybridization*, Submitted to the Journal of Computational Biology, 2013.

[22] DM. Mount, *Bioinformatics: Sequence and genome analysis*, Cold Spring Harbour Labradory Press, 2004.

[23] O. Olsvik, J. Wahlberg, B. Petterson, M. Uhlén, T. Popovic, I. K. Wachsmuth, and P. I. Fields, *Use of automated sequencing of polymerase chain reaction-generated amplicons to identify three types of cholera toxin subunit B in Vibrio cholera O1 strains*, Journal of Clinical Microbiology **31** (1993), no. 1, 22–25.

[24] E. Pettersson, J. Lundeberg, and A. Ahmadian, *Generations of sequencing technologies*, Genomics **93** (2009), no. 2, 105–111.

[25] P. Pevzner, *l-tuple DNA sequencing: Computer analysis*, Journal of Biomolecular Structure and Dynamics **7** (1989), 63–73.

[26] ———, *Computational molecular biology: An algorithmic approach*, The MIT Press, 2000.

[27] V. Phan and S. Skiena, *Dealing with errors in interactive sequencing by hybridization*, Bioinformatics **17** (2001), no. 10, 862–870.

[28] R. Shamir and D. Tsur, *Large scale sequencing by hybridization*, Journal of Computational Biology **9** (2002), no. 2, 413–428.

[29] M. Sipser, *Introduction to the theory of computation*, Thomson Course Technology, 1996.

[30] L. Stein, *Genome annotation: from sequence to biology*, Nature Reviews Genetics **2** (2001), 493–503.

[31] A. Turing, *On computable numbers, with an application to the entscheidungsproblem*, Proceedings of the London Mathematical Society, 2, vol. 43, 1937.

[32] J. van Leeuwen, *Handbook of theoretical computer science volume a: Algorithms and complexity*, The MIT Press, 1990.

[33] D.B. West, *Introduction to graph theory, second edition*, Prentice Hall, 2001.

[34] R. Wheeler, `http://upload.wikimedia.org/wikipedia/commons/4/4c/DNA_Structure%2BKey%2BLabelled.pn_NoBB.png`, 2011.

[35] J. Zhang, L. Wu, and X. Zhang, *Reconstruction of DNA sequencing by hybridization*, Bioinformatics **19** (2003), no. 1, 14–21.