Thesis

Enzyme Selection for Optical Mapping is Hard

Submitted by

Laura Adams

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2015

Master's Committee:

Advisor: Christina Boucher

Adele Howe
Patrick Ingram

Abstract

Enzyme Selection for Optical Mapping is Hard

The process of assembling a genome, without access to a reference genome, is prone to a type of error called a misassembly error. These errors are difficult to detect and can mimic true, biological variation. Optical mapping data has been shown to have the potential to reduce misassembly errors in draft genomes. Optical mapping data is generated using digestion enzymes on a genome. In this paper, we formulate the problem of selecting optimal digestion enzymes to create the most informative optical map. We show this process in NP-hard and W[1]-hard. We also propose and evaluate a machine learning method using a support vector machine and feature reduction to estimate the optimal enzymes. Using this method, we were able to predict two optimal enzymes exactly and estimate three more within reasonable similarity.

Table of Contents

CHAPTER 1

Introduction

Genome sequencing using paired-end read data is becoming widely used for a variety of biological applications. This technology enables the large-scale sequencing of genomic data from a variety of organisms, enabling analysis of genetic variation between individuals or species [**?**]. While the first sequencing technology was slow and expensive, modern advances in next-generation sequencing allow for rapid and inexpensive sequencing of whole genomes [**?** **?**]. However, errors and mistakes in sequenced genomes can arise due to a variety of factors—the most prevalent being due to how genomes are constructed by the assembly programs [**?**] and limitations in the size of DNA strands that are capable of being processed. As life science research becomes more dependent upon sequencing and assembly technologies, ensuring their accuracy is increasingly important.

The process of next generation sequencing (NGS) works as follows: genetic material is extracted, fragmented into smaller pieces, loaded onto a sequencing machine (e.g., Illumina sequencing technology), the sequencing machine is run, and sequence *reads* are generated from the machine. Each read is a sequence of nucleotides that corresponds to one small fragment of DNA. Most commonly, the nucleotides in the strands of DNA are read from both directions and *paired-end read* data is created. Paired-end read data means that there are two reads corresponding to each fragment, and these reads are identified to be mate-pairs. Lastly, we mention that each sequence read typically consists of 100 to 150 base pairs (bp) [**?**].

Assembly programs were created in order to take paired-end reads and assemble them into the digitally transcribed genome for the organism of interest. The most common type

of assemblers are Eulerian assemblers [**?** **?**], which build a de Bruijn graph from the read data. This graph is later simplified in order to build longer, contiguous sequences of paired-end reads, called *contigs* [**?**]. Hence, these contigs represent collections of base pairs that were present in the original genome. This process of assembling paired-end reads into a full genome, without access to a reference genome, is called *de novo* assembly. Contigs are frequently joined together using other computational approaches to create longer regions, called *scaffolds*.

The sequencing and assembling of small to moderate size genomes is currently able to be done efficiently. However, one fundamental problem in this process that remains difficult is the introduction of errors in assembled contigs or scaffolds. For our purposes, we will divide these errors into subclasses based on their size. The first type of errors are single nucleotide erroneous changes in the contigs that are referred to as *substitution errors*, and small ($\leq$ 50 bp) insertion and deletions. The second type of error, which we call a *misassembly error* consists of large segments being assembled incorrectly. According to QUAST [**?**], a misassembly error is a region in an assembled contig with a significantly large insertion, deletion, inversion, or rearrangement of base pairs generated by the assembler. This error is caused by the logic of the assembly program rearranging the reads in the incorrect order. Misassembly errors can mimic real, biological variation and thus, can be misconstrued as having some biological relevance [**?**].

Whereas there exist methods to identify and correct small errors (see Ronen et al. [**?**]), misassembly errors are more difficult to identify with short read data alone. Hence, short read data can give a local view of a genome, but data that gives insight to the structure of the genome on a larger scale is needed for misassembly detection. Muggli et al. [**?**]

demonstrated that optical mapping data can provide this independent, large-scale information about the genome. Optical mapping works as follows [**?** **?**]: (1) DNA molecules adhere to a magnetized, glass plate; (2) these molecules are elongated by flowing fluid; (3) next, one or more digestion enzymes are selected to cut the DNA; (4) these cut segments are then dyed and digitally measured under a microscope; (5) the images are analyzed to produce a molecular map, containing the relative order and length of the fragments [**?**]. We note that multiple copies of the genome are processed this way, and a consensus map is formed, showing roughly how many base pairs are between the enzymes' recognition sequences [**?**]. These enzymes cut the DNA strands in certain places; the location of this nucleotide sequence is known in advance.

Despite the vast potential of optical mapping data, very few publicly available tools exist to analyze this type of data, and even fewer tools exist to computationally identify misassembly data. The review article by Menelowitz and Pop [**?**] discussed this former point; they state: "relatively few methods exist for analyzing and using optical mapping data, and even fewer are available in effective publicly-available software packages...There is, thus, a critical need for the continued development and public release of software tools for processing optical mapping data" [**?**]. Muggli et al. [**?**] created misSEQuel in order to address these needs. misSEQuel combines short read data and optical mapping data in order to detect misassembly errors. It works by first taking an assembled contig and simulates digesting it with three digestion enzymes. After this process, called *in silico* digestion, the digested contigs are then aligned to an optical map of the entire genome to determine if the contig was misassembled.

Muggli et al. [**?**] used a combination of real and simulated optical mapping data. They report both the true positive rate (the percentage of misassembled contigs that were deemed

as such) and the false positive rate (the percentage of correctly assembled contains that were deemed to be misassembled). The experiments with the simulated data considered all combinations of three digestion enzymes by constructing a simulated optical map for each combination, completing the *in silico* digestion with those enzymes, and running MISSEQUEL to determine a prediction as to which contigs were misassembled. The best true positive rate (TPR) and false positive rate (FPR) are then reported. However, the work of Muggli et al [**?**] required prior knowledge of a reference genome and hence, demonstrated how good the sensitivity and specificity of their method can be when the enzymes are chosen optimally. Unfortunately, it is not feasible in a laboratory environment to try all combinations of the over 500 digestion enzymes in the REBASE enzyme database [**?**].

In this paper, we focus on the problem of selecting the set of restriction enzymes for misassembly error detection in a *de novo* manner. While Muggli et al [**?**] detected misassembly errors using optical mapping data, they did so by using prior knowledge about the problem. Specifically, they were able to try all possible digestion enzymes because they had the reference genome and were able to simulate the optical map. However, in a *de novo* manner, we only select the restriction enzymes using the REBASE database, and the assembled contigs; a reference genome is not used. What is returned is a prediction as to which enzymes should be used for building the optical map.

As this problem has never been studied before, our formalization and approach is the first in the field. Thus, we begin by formulating the digestion enzyme selection problem and show that it is both NP-complete and W[1]-hard with the respect to the number of enzymes to be chosen. Since the number of enzymes to be chosen is likely the only parameter to remain small in practice, our W[1]-hardness result demonstrates that parameterized complexity will not be fruitful in giving a practical algorithm. Hence, in order to solve the problem from a practical

perspective we develop and implement a machine learning-based method to estimate the optimal enzyme for optical mapping. Our machine learning approach consists of transforming the problem into a binary, or two class, classification problem and training a support vector machine (SVM) from *Francisella tularensis* assembly data. In this transformation, we assign misassembly data from QUAST as the labels and set the digestion enzymes as the features. We then use feature reduction to find the enzymes that add the most information to the classifier.

Our results demonstrate that it is possible to estimate reasonably close digestion enzymes for optical mapping. Using support vector machines, we were able to formulate misassembly error detection as a binary classification problem and use feature reduction to estimate a solution to digestion enzyme selection. Our results were able to predict two of the top eleven digestion enzymes exactly, as well as find an approximate enzyme for three of the top eleven enzymes.

1.1. Related Work

There are three other tools capable of finding and correcting misassembly errors: amosvalidate [**?**], REAPR [**?**] and Pilon [**?**]. REAPR uses both short insert and long insert paired-end sequencing library data. However, it can only use one of these types of sequencing data at a given time. Amosvalidate is included in the AMOS assembly package [**?**]. It was specifically developed for first generation sequencing libraries [**?**]. iMetAMOS [**?**] is an automatic assembly tool that provides both validation of the assembly and error correction. It brings several open-source tools together and creates annotated assemblies from an ensemble of assemblers and tools. Presently, it uses Pilon [**?**] to detect a variety of errors, including misassembly errors in draft genomes; it also uses REAPR for misassembly error

correction. REAPR and Pilon are specifically designed to use short insert and long insert library data. Unlike REAPR and amosvalidate, Pilon is designed specifically for microbial genomes.

As stated by Menelowitz and Pop [**?**], few methods exist to analyze optical mapping data; however, previous work has still been done. SOMA [**?**] uses a dynamic programming algorithm to align *in silico* digested contigs to optical mapping data. AGORA [**?**] uses the optical map information to construct a de Bruijn graph; this graph is used to improve the resulting assembly. TWIN [**?**] is an index-based method for aligning contigs to an optical map; it is capable of aligning *in silico* digested contigs orders of magnitude faster than competing methods. Xavier et al. [**?**] have evaluated proprietary software to detect misassembly errors in bacterial genomes.

CHAPTER 2

The Enzyme Selection Problem is Hard

2.1. Background

Strings. Throughout we consider a string $\mathsf{X} = \mathsf{X}[1..n] = \mathsf{X}[1]\mathsf{X}[2]\ldots\mathsf{X}[n]$ of $|\mathsf{X}| = n$ symbols drawn from the alphabet $[0..\sigma-1]$. For $i = 1,\ldots,n$ we write $\mathsf{X}[i..n]$ to denote the *suffix* of X of length $n-i+1$, that is $\mathsf{X}[i..n] = \mathsf{X}[i]\mathsf{X}[i+1]\ldots\mathsf{X}[n]$. Similarly, we write $\mathsf{X}[1..i]$ to denote the *prefix* of X of length i. $\mathsf{X}[i..j]$ is the *substring* $\mathsf{X}[i]\mathsf{X}[i+1]\ldots\mathsf{X}[j]$ of X that starts at position i and ends at j.

Optical Mapping. From a computational point of view, optical mapping is a process that takes two strings: a genome $\mathsf{A}[1,n]$ and a restriction sequence $\mathsf{B}[1,b]$, and produces an array (string) of integers $\mathsf{M}[1,m]$, such that $\mathsf{M}[i] = j$ if and only if $\mathsf{A}[j..j+b] = \mathsf{B}$ is the ith occurrence of B in A.

For example, if we let $\mathsf{B} = act$ and

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A	*a*	*t*	*a*	*c*	*t*	*t*	*a*	*c*	*t*	*g*	*g*	*a*	*c*	*t*	*a*	*c*	*t*	*a*	*a*	*a*	*c*	*t*

then we would have $\mathsf{M} = 3, 7, 12, 15, 20$. Furthermore, it will also be convenient to view M slightly differently, as an array of fragment sizes, or distances between occurrences of B in A (equivalently differences between adjacent values in M). We denote this *fragment size domain* of M, as the array $\mathsf{F}[1,m]$, defined such that $\mathsf{F}[i] = (\mathsf{M}[i] - \mathsf{M}[i-1])$, with $\mathsf{F}[1] = \mathsf{M}[1] - 1$. Continuing with the example above, we have $\mathsf{F} = 2, 4, 5, 3, 5$.

Parameterized Complexity and Approximation. Algorithms whose complexity can be expressed as a function of a parameter k of the input are called *parameterized algorithms*. The complexity class FPT (fixed-paremeter tractable) contains all problems for which there is an algorithm running in time $f(k) \cdot |x|^{O(1)}$, where $|x|$ is the length of the input, k is a

parameter of the input, and $f(k)$ is a computable function which depends only on k and not on $|x|$. For example, one would like to replace an algorithm which is exponential in the entire input size $|x|$, by one which is exponential only in some (small) parameter k of the input, and otherwise polynomial in $|x|$. If the value of k remains relatively small in practice, and if the function $f(k)$ is not growing too fast, then such an algorithm may be efficient for large datasets. For some problems, it can be shown that there is no FPT algorithm, by showing that there is a so-called *parameterized reduction* to a complexity class called W[1].

Another algorithmic technique to deal with NP-completeness is approximation. A problem admits a *polynomial time approximation scheme* (PTAS) if it finds a solution that is at most a factor $(1+\epsilon)$ worse than the optimum [**?**] in $n^{O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})}$-time. If the exponent of the polynomial in the running time of a PTAS is independent of ϵ then the PTAS is called an *efficient PTAS* (EPTAS). The difference in run time for a PTAS and an EPTAS can be quite dramatic. For instance, running a $O(2^{1/\epsilon}n)$-time algorithm is reasonable for $\epsilon = \frac{1}{10}$ and $n = 1000$, whereas running a $O(n^{1/\epsilon})$-time algorithm is infeasible on this same input. Hence, considerable effort has been devoted to improving PTASs to EPTASs, and showing that such an improvement is unlikely for some problems. The relationship between problems that admit an EPTAS and those that are W[1]-hard is well-defined; Boucher *et al.* [**?**] showed that problems that are W[1]-hard with respect to a parameter k cannot admit a EPTAS with respect to that parameter.

2.2. Problem Definition

In order to show that the problem of selecting an enzyme for optical mapping is both NP-hard and W[1]-hard, we need to formalize the problem mathematically. We begin by giving the following formal definition of the Enzyme Selection problem:

Definition 1 (**Barcode**). *Given a set of t strings $A = a_1, a_2, \ldots, a_t$, and another string $s[1 \ldots n]$, the* barcode *of s with respect to A is the set of positions $p_1, p_2, \ldots p_\ell$ such that for every p_i there exists some a_j that is a prefix of $s[p_i \ldots n]$.*

Enzyme Selection	
Input:	A set of m strings (enzymes) $E = (e_1, e_2, \ldots e_m)$, another set of n strings (contigs) $C = (c_1, c_2, \ldots, c_n)$, and integer parameters k and t.
Parameter:	Subsets E' of E and C' of C, where $\|E\| = k$ and $\|C\| = t$ or E and C are empty.
Question:	Does there exist subsets E' of E and C' of C, where $\|E'\| = k$ and $\|C'\| = t$, such that every $c_i \in C'$ has a different non-empty barcode with respect to the strings in E', and every $c_j \notin C'$ is the empty set with respect to the barcodes in E'? If no such C' and E' exist then the empty set is returned for both.

2.3. Enzyme Selection Problem is NP-Complete

Let G be a bipartite graph with partite sets A and B. We denote by $N(v)$ the set of neighbors of a vertex v. A set $D \subseteq B$ is called a *discriminating code* of G if the following conditions hold:

- for all $v \in A$, $N(v) \cap D \neq \emptyset$, and

- for all $u, v \in A$, $N(u) \cap D \neq N(v) \cap D$.

The following problem was proven NP-complete in [**?**]. Given a bipartite graph G with partite sets A and B, and an integer k, decide whether there exists a discriminating code $D \subseteq B$ such that $|D| \leq k$.

THEOREM 1. *The Enzyme Selection problem is NP-complete even when restricted to ternary alphabets.*

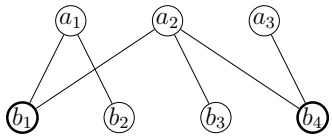
PROOF. Let $G = (A \cup B, F)$ be a bipartite graph with $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$. From G we construct an instance (E, C) to the Enzyme Selection problem, with $E = \{e_1, \ldots, e_m\}$ and $C = \{c_1, \ldots, c_n\}$. The vertices in B will be modeled by enzymes and the vertices in A will be modeled by contigs. We will construct each contig such that it contains exactly one occurrence for each of its neighboring enzymes in G.

For every $b_i \in B$, we construct an enzyme e_i as a binary string of length $\ell := \lceil \log_2 m \rceil$ consisting of the binary encoding of $i - 1$ (possibly padded with '0's). For every $a_j \in A$, contig c_j is obtained as the concatenation of m strings of length ℓ, separated by the character '#': for each $i \in [1..m]$, if $b_i \in N(a_j)$, we set the ith string of c_j to e_i, otherwise we set it to a string made up of ℓ occurrences of '#'. See also Figure 2.1.

We claim that G admits a discriminating code of size x if and only if there is a solution to the Enzyme Selection problem on the instance (E, C) constructed above, with parameters $k = x$ and $t = n$. For the forward implication, let $D = \{b_{i_1}, \ldots, b_{i_k}\}$ be a discriminating code of G, and consider the corresponding set of enzymes $E' = \{e_{i_1}, \ldots, e_{i_k}\}$. By construction, each contig c_i contains (single) occurrences of a different set of enzymes. Since each enzyme occurs at the same position in each contig, we have that each contig in C has a unique barcode with respect to E'. The reverse implication follows analogously, with the additional

observation that the barcode of any contig in C cannot be empty by definition, thus each vertex in A has at least one neighbor in the corresponding discriminating set.

This proves the NP-hardness of the problem. The NP-completeness follows from the fact that given a subset E' of enzymes and a subset C' of the contigs, one can compute in polynomial time the barcodes of the strings in C', and also check in polynomial time that the resulting barcodes satisfy the requirements of of the Enzyme Selection time $\square$



	positions: 1 2 3 4 5 6 7 8 9 10 11	barcodes:
$e_1 = 00$		
$e_2 = 01$	$c_1 = 0\ 0\ \#\ 0\ 1\ \#\#\#\#\#\#$	$1, 4$
$e_3 = 10$	$c_2 = 0\ 0\ \#\#\#\#\ 1\ 0\ \#\ 1\ 1$	$1, 7, 10$
$e_4 = 11$	$c_3 = \#\#\#\#\#\#\#\#\#\ 1\ 1$	10

FIGURE 2.1. On the left, a bipartite graph $G = (A \cup B, F)$; a discriminating code for G is highlighted. On the right, the instance (E, C) to the Enzyme Selection problem, with $E = \{e_1, e_2, e_3, e_4\}$ and $C = \{c_1, c_2, c_3\}$, and the barcodes of the contigs in C. The discriminating code $\{b_1, b_4\}$ of G corresponds to the enzymes $e_1 = 00$ and $e_4 = 11$.

2.4. Enzyme Selection is W[1]-Hard

The most natural question to ask is whether the *Enzyme Selection* problem admits FPT algorithm with respect to k; since k would remain relatively small in practice (aka, less than five), then the algorithm would be expected to be fairly efficient. Unfortunately, we show that this is unlikely to be the case, unless W[1] = FPT. We reduce the problem from the *k-Independent Set* problem on 2-interval graphs to prove that *Enzyme Selection* is W[1]-hard.

In 2008, Fellows et al. [**?**] proved that *k-Independent Set* is W[1]-hard even when the problem is restricted to 2-interval graphs. A multiple-interval graph G can be thought of as a mapping from a graph $G = (V, E)$ to a set of intervals $\mathcal{F}$. The mapping assigns an interval f_i for each vertex v_i in V and a non-empty collection of intervals where the following property holds: two distinct vertices v_i and v_j are adjacent if and only if intervals $f(v_i)$ and $f(v_j)$ are

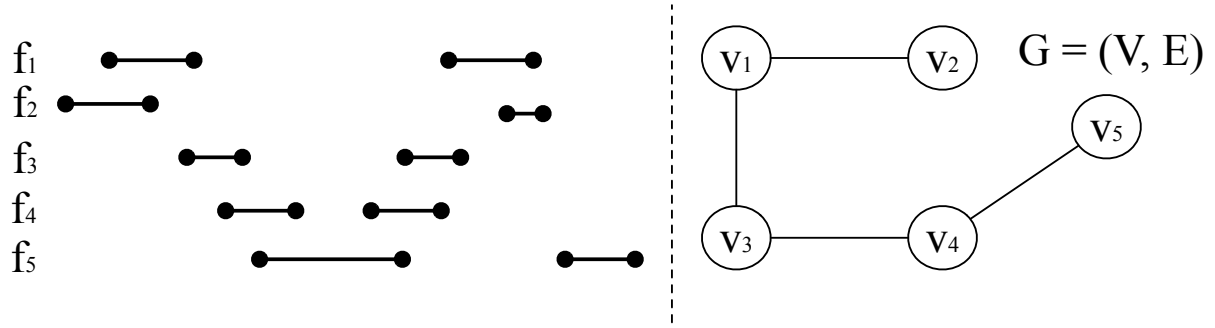


FIGURE 2.2. We illustrate a 2-interval graph of five 2-intervals. Hence, in this example $\mathcal{F} = f_1, f_2, f_3, f_4, f_5$ corresponds to an interval graph $G = (V, E)$ that contains five vertices and four edges (disjoint pair of intervals).

disjoint. We let $|f(v)|$ denote the number of intervals in f_v. The class of 2-interval graphs are those where $|f(v)| = 2$ for each $v \in V$. See Figure 2.2 for an example of a 2-interval graph. The input to the *2-Interval Graph k-Independent Set* problem is a 2-interval graph G, a family of 2-intervals $\mathcal{F}$ and a parameter k and the aim is to determine whether there exists a set of k intervals in $\mathcal{F}$ that are pairwise disjoint, i.e, does there exist a subset $\mathcal{F}' \subseteq \mathcal{F}$, such that $|\mathcal{F}'| = k$ and any pair of interval $(f_i, f_j) \in \mathcal{F}'$ is disjoint?

THEOREM 2. Limited Enzyme Selection *is W[1]-hard with respect to k.*

PROOF. Given an instance $(G, \mathcal{F}, k)$ of the *2-Interval k-Independent Set* problem we produce in $f(k)|\mathcal{F}|^{O(1)}$-time an instance (E, C, t, k^*) of *Enzyme Selection* with the following property. Let m be the number of disjoint pairs in $\mathcal{F}$. Our construction has the following property: $\mathcal{F}$ has a subset of k intervals that are disjoint if and only if there exists subsets E' of enzymes E of size $k^* = k + k^2$ and C' of enzymes C of size $t = 2k^2 + 4(m+1)k^2$ such that every $c_i \in C'$ has a different barcode with respect to the strings in E' and every $c_j \notin C'$ is the empty set, whereas if no independent set of size k exists in $\mathcal{F}$ then no such subsets E' and C' exist. Thus, the reduction is a parameterized, gap-creating reduction where the size of gap decreases as k increases but the decrease is a function of k only.

We describe how the instance (E, C, k^*, t) is constructed from $(G, \mathcal{F}, k)$. In our construction we will create contigs strings in C that encode for edge selection and vertex selection. The edge selection contig strings will correspond to the selection of a vertex in G (i.e. interval in $\mathcal{F}$) and the vertex selection contain strings will ensure that the vertices of the edges selected are chosen, meaning all the k^2 edges must be between the same k vertices. The alphabet for which our enzyme strings and contig strings will be defined is $\{0, 1, \#\}$. To begin the construction, we give the description of E: for each pair of intervals in $\mathcal{F}$, say f_i and f_j (corresponding to vertices v_i and v_j in the interval graph G), we create an enzyme e_{ij} in E, and for each interval in $\mathcal{F}$ we create an enzyme e_i in E. Hence, we have separate enzymes for the vertices and edges in the 2-interval graph. Each enzyme string in E is assigned an unique code using Huffman encoding and therefore, all enzymes in E are unique, binary strings and have the property that no enzyme string is a prefix of another enzyme string. We assume that $k \geq 2$ since $k = 1$ produces trivial cases.

Edge Selection. For each pair of intervals in $\mathcal{F}$, say f_i and f_j, we create two contig strings in C: c_{ij}^1 and c_{ij}^2. The string c_{ij}^1 will be the concatenation of a length-ℓ_{ij1} string of #'s, and e_{ij}, and similarly, c_{ij}^2 will be the concatenation of a length-ℓ_{ij2} string of #'s, and e_{ij}. The following property holds for ℓ_{ij1} and ℓ_{ij2}: if the intervals f_i and f_j are disjoint then $\ell_{ij1} \neq \ell_{ij2}$, otherwise $\ell_{ij1} = \ell_{ij2}$. Clearly, ℓ_{ij1} and ℓ_{ij2} can be assigned uniquely for each disjoint pair f_i and f_j by keeping a counter of the number of disjoint pairs seen so far. Therefore, the intervals f_i and f_j are only selected if and only they are disjoint; since selecting e_{ij} can only make the barcodes unique.

Vertex Selection. We create $4(m + 1)m$ contig strings C, where m is the number of disjoint pairs in $\mathcal{F}$. We split the discussion into two subsets: (1) validation of the selection of the endpoints of the edges and (2) validation of the selection of the edges of the endpoints.

We will begin describing (1). We create $m+1$ contig strings for each disjoint pair in $\mathcal{F}$: ${c'}^1_{i,j}$, ${c'}^2_{i,j}$, ..., ${c'}^{m+1}_{i,j}$. The string ${c'}^k_{ij}$ will be the concatenation of a x_{ij}-length string of #'s, the string e_{ij}, a ${\ell'}_{ijk}$-length string of #'s, and the string e_i, where $k = 1, \ldots, m+1$. Similarly, we create $m+1$ contig strings for each disjoint pair in $\mathcal{F}$: ${c''}^1_{i,j}$, ${c''}^2_{i,j}$, ..., ${c''}^{m+1}_{i,j}$ each of which is the concatenation of the x_{ij} string of #'s, the string e_{ij}, x_{ij} string of #'s, the string e_{ij}, a ${\ell''}_{ijk}$-length string of #'s, the string e_j, where $k = 1, \ldots, m+1$. The selection of these vertices ensures that if you select enzyme e_{ij} then e_i and e_j must also be selected (otherwise, you would have contig strings in C' that do not have an unique barcode).

Next, we construct $2(m+1)m$ contig strings that ensure the opposite is true: if e_i or e_j is selected then e_{ij} must be selected. These are similar to those just defined; the only difference being that e_{ij} is replaced with e_i or e_j. Hence, we create $m+1$ contig strings for each disjoint pair in $\mathcal{F}$: ${c'''}^1_{i,j}$, ${c'''}^2_{i,j}$, ..., ${c'''}^{m+1}_{i,j}$. The string ${c'''}^k_{ij}$ will be the concatenation of a x_{ij}-length string of #'s, the string e_i, a ${\ell'''}_{ijk}$-length string of #'s, and the string e_{ij}, where $k = 1, \ldots, m+1$. Similarly, ${c''''}^1_{i,j}$, ${c''''}^2_{i,j}$, ..., ${c''''}^{m+1}_{i,j}$ are defined in an identical manner as ${c'''}^1_{i,j}$, ${c'''}^2_{i,j}$, ..., ${c'''}^{m+1}_{i,j}$, with the exception that e_i replace with e_j.

One important point to note is that ${\ell'}_{ijk}$, ${\ell''}_{ijk}$, ${\ell''''}_{ijk}$, and ${\ell''''}_{ijk}$, and x_{ij} are unique for all $k = 1, \ldots m+1$ and all pairs of i and j. And these values are unique from all the ℓ_{ij1} and ℓ_{ij2} values assigned for the contig strings used for edge selection. All these values can be trivially assigned uniquely by using using a counter and incrementing it each time one is assigned. Hence, the vertex selection implies that if e_{ij} is selected then e_i and e_j must also be selected from the set E.

Analysis. Our construction has $|\mathcal{F}|+|\mathcal{F}|^2$ total enzymes (size of E), and $|\mathcal{F}|^2+4(m+1)m$ contain strings (size of C) and hence, our construction can be completed in $O(|\mathcal{F}|^2 + 4(m+$

$1)m + |\mathcal{F}| \log |\mathcal{F}|)$-time. The $|\mathcal{F}| \log |\mathcal{F}|$ in this running time is from the time required for enzyme construction using Huffman coding.

Next, we need to show that the construction also ensures that there is a solution to an instance of the *2-Interval k-Independent Set* problem if and only if there is a solution to the constructed instance of *Enzyme Selection*. Clearly, if there is a k-independent set in the 2-interval graph then there exists a subset E' of E of size $k+k^2$ and C' of size $2k^2+4(m+1)k^2$ that satisfies the conditions of the *Enzyme Selection* problem. Therefore, we only need to show that the reverse is true.

Suppose there exists subset E' of E of size $k + k^2$ and C' of size $2k^2 + 4(m + 1)k^2$ such that each $c_i \in C'$ has an unique barcode and every $c_j \notin C'$ is the empty set. We first argue that (1) if e_{ij} is selected then the intervals f_i and f_j in $\mathcal{F}$ are disjoint; (2) if e_{ij} is selected the e_i and e_j must be selected; and (3) if e_i is selected then e_{ij} is selected. The edge selection contig strings ensure that (1) is true; if e_{ij} is selected then $\ell_{ij1} \neq \ell_{ij2}$ since otherwise it would contradict that c_{ij1} and c_{ij2} have unique barcodes. The condition $\ell_{ij1} \neq \ell_{ij2}$ implies f_i and f_j in $\mathcal{F}$ are disjoint. Next, we show (2) is true. Suppose otherwise that e_{ij} is selected but e_i or e_j is not selected then it follows that there exists $2(m+1)$ strings in C' (namely ${c'}^1_{i,j}$, ${c'}^2_{i,j}$, ..., ${c'}^{m+1}_{i,j}$ and ${c''}^1_{i,j}$, ${c''}^2_{i,j}$, ..., ${c''}^{m+1}_{i,j}$) that do not have an unique barcode, which contradicts the definition of C'. Lastly, we show (3) is true. Similarly, suppose otherwise that e_i is selected but e_{ij} is not selected. Then it follows that there exists $m+1$ strings in C' that do not have an unique barcode (namely, ${c'''}^1_{i,j}$, ${c'''}^2_{i,j}$, ..., ${c'''}^{m+1}_{i,j}$) which contradicts the definition of C'.

It follows from (1), (2) and (3) every triple of enzymes $e_i, e_j, e_{ij} \in E'$ corresponds to a pair of disjoint intervals f_i and f_j in $\mathcal{F}$, $4(m + 1)$ contig strings constructed for vertex selection having an unique barcode, and two contig strings for vertex selection having an unique barcode. Hence, this shows that if there exists a subset of E' of E of size $k + k^2$ and

a subset C' of C of size $t = 2k^2 + 4(m+1)k^2$ then there exists a subset of intervals in $\mathcal{F}$ of size k such that each pair of intervals is disjoint. □

We have shown that *Enzyme Selection* is W[1]-hard and the following result follows directly from this and the result of Boucher et al. [**?**].

COROLLARY 1. *There exists no EPTAS for* Enzyme Selection*, unless FPT = W[1].*

CHAPTER 3

A Practical Approach

Our practical approach to this problem involved formulating the problem as a binary classification problem for machine learning approximation. This approach to digestion enzyme selection has three main components: feature acquisition, model training, and feature reduction. The model was trained using the genome of *Francisella tularensis*; this genome has a reference genome and various draft genomes (set of contigs).

3.1. Feature Acquisition

For each enzyme, an optical map was simulated using the reference genome. For each assembly of the data, TWIN [**?**] was used to align the sets of assembled contains to the simulated optical map, reads were aligned to the contigs and discordant read alignments were identified using misSEQuel, and lastly, the output of this processes was parsed to create a feature set. We note that a contig may align more than once to an optical map. For each alignment, TWIN uses the following alignment scoring function:

$$|\sum_{i=s}^{t} F_i - \sum_{j=u}^{v} l_j| \leq F_\sigma \sqrt{\sum_{j=u}^{v} \sigma_j^2},$$

where F_i refers to the list of contigs and l_j are the regions of the optical map. F_σ is a parameter and σ is standard deviation. We refer to this as the Fvalue statistic. As described by Nagarajan et al. [**?**], Fvalue is a heuristic that any good alignment will satisfy; a good alignment will have a low Fvalue, but it is also possible for a poor alignment to also have a low Fvalue. However, a good alignment should not have a high Fvalue. We used the following features for training the SVM:

- Least and greatest FVALUE. As previously mentioned, each alignment has the FVALUE statistics. The least (and greatest) FVALUE is the smallest (and largest) FVALUE witnessed over all alignments of a single contain.
- χ^2 sum. When TWIN finds an alignment, it calculates a deviation for each fragment, standardizes it by dividing it by the 150bp standard deviation, squares the result, and accumulates those across all the aligned fragments. The resulting value is the χ^2 sum. The largest χ^2 sum, comparing over all alignments, is used as a feature.
- Deviation sum. Using the process described in χ^2 sum, TWIN also outputs the sum of the absolute value of the deviations. The greatest deviation sum is used as a feature.
- Discordant read alignment. MISSEQUEL outputs whether there exists a region that has size $\geq 200bp$ that contains improper read alignment, meaning that either the depth of aligned reads was significantly larger or smaller than the expected coverage or the mate-pair alignment is discordant.

3.2. Model Training

The data was next formulated as a binary classification problem. Support vector machines (SVMs) have shown significant success in binary classification and were therefore used as our machine learning algorithm. We used the SVM implementation in Weka[**?**] with the LibSVM[**?**] library. Weka is a generalized machine learning tool that supports a large variety of machine learning algorithms and feature reduction techniques.

The SVMs were trained and evaluated with the linear kernel. We used paired-end read data from the *Francisella tularensis* genome that was assembled in three different assemblers:

ABySS [**?**], SOAPdenovo [**?**], and SPAdes [**?**]. For the SOAPdenovo assembly, we used both an original set and a set balanced with an equal number of positive and negative examples. The original data had too many negative examples and caused the SVM to be over-fitted. We report the performance of both sets of data. The balanced SOAPdenovo set was created by randomly removing negative examples until an equal number of positive and negative examples remain.

The data sets were used in seven different combinations to train and evaluate the SVMs: only ABySS, only SOAPdenovo (original), only SOAPdenovo (balanced), only SPAdes, combined ABySS/SOAPdenovo (balanced), combined ABySS/SPAdes, and combined SPAdes/ SOAPdenovo (balanced). We combined the data sets because the *Francisella tularensis* genome is fairly small and some sets did not have enough examples to properly train the classifier.

3.3. Feature Reduction

We used Weka's *SVMAttributeEval* method from its "attribute selection" framework for feature reduction. Each of the seven data sets were reduced to eleven features, which would represent the best eleven digestion enzymes for optical mapping. The rationale behind this method is that enzymes that are most predictive of misassembly errors will build the best optical map.

SVMAttributeEval evaluates the worth of a feature by training an SVM with all features. Attributes are then ranked by the square of their weight in the SVM. The attributes with the greatest weights are considered the best attributes.

We used this method to reduce each of the seven data sets to their best eleven features. As stated earlier, these best features represent the estimate of the optimal enzymes for optical

mapping. These final, reduced features should be similar to the known optimal features. The data sets were then reduced to contain only the features selected from the above technique. These reduced sets were then used to retrain the SVMs.

CHAPTER 4

Results

In order to evaluate the effectiveness of our method, we used several evaluation metrics. The first describes the results of five-fold cross validation (5CV) for the classifiers trained on data from each assembly and grouping of assemblies. Next, we compare the predicted enzymes for the ABySS assembly with a list of known, optimal enzymes. Finally, we evaluate the effectiveness of all classifiers on data from the *Porphyromonas gingivalis* assembly. The *Porphyromonas gingivalis* genome was assembled using SPAdes.

4.1. *Francisella tularensis* Dataset Description

We use data consisting of approximately 6.9 million paired-end 101 bp reads from the prokaryote genome *Francisella tularensis* for training and cross-validating the models. These data were gathered from the Illumina Genome Analayzer (GA) IIx platform. We use the NCBI Short Read Archive (accession number [SRA:SRR063416]) to obtain the pair-end read data themselves. We obtained the reference genome from the NCBI website (Reference genome [RefSeq:NC_006570.2]). The *Francisella tularensis* genome is 1,892,775 bp in length with a GC content of 32%. In order to ensure quality, we aligned the reads to the *Francisella tularensis* genome using BWA (version 0.5.9) [**?**] with default parameters. A read is *mapped* if BWA outputs an alignment for it and *unmapped* otherwise. We found that 97% of the reads mapped to the reference genome. Given there is no public optical map for *Francisella tularensis*, we created a simulated optical map from the reference genome.

We assembled these *Francisella tularensis* reads with a set of assemblers. The versions used were those that were publicly available before or on September 1, 2014: SPAdes (version 3.1, after repeat resolution) [**?**]; SOAPdenovo (version 2.04) [**?**]; and ABySS (version

TABLE 4.1. The performance statistics for the major assembly tools on the *Francisella tularensis* dataset. We note the dataset had a genome length of 1,892,775 bp, and 6,907,220 number of 101 bp reads, while using QUAST in default mode [**?**]. All statistics are based on contigs no shorter than 500 bp. Contigs with fewer than 500 bp were thrown out of the dataset.

Assembler	**# contigs (# unaligned)**	**N50**	**Largest (bp)**	**Total (bp)**	**MA / local MA (MA (bp))**	**GF (%)**
SOAPdenovo	307 (3 + 31 part)	8,767	39,989	2,018,158	10 / 35 (96,258)	92.05
ABySS	96 (1 part)	27,975	88,275	1,875,628	64 / 32 (1,330,684)	95.87
SPAdes	100 (2 + 17 part)	26,876	87,891	1,797,197	23 / 31 (497,356)	93.75

1.5.2) [**?**]. These assemblers generate both contigs and scaffolds. However, we considered only contigs.

We used QUAST [**?**] in default mode to evaluate the assemblies for misassembly errors. QUAST defines misassembly error as being *extensive* or *local*. An extensively misassembled contig is defined as one that satisfies one of the following conditions: (a) the left flanking sequence aligns over 1 kbp away from the right flanking sequence on the reference; (b) flanking sequences overlap on more than 1 kbp; (c) flanking sequences align to different strands or different chromosomes [**?**]. A local misassembled contig is one that satisfies the following conditions: (a) two or more distinct alignments cover the breakpoint; (b) the gap between left and right flanking sequences is less than 1 kbp; and the left and right flanking sequences both are on the same strand of the same chromosome of the reference genome. A correctly assembled contig is one that does not contain either type of error.

Table 4.1 presents the assembly statistics for the initial alignments. Note that a contig can be both extensively and locally misaligned at the same time. Table 4.1 gives the number of contigs having at least one extensive misassembly error and the number of contigs having at least one local misassembly error.

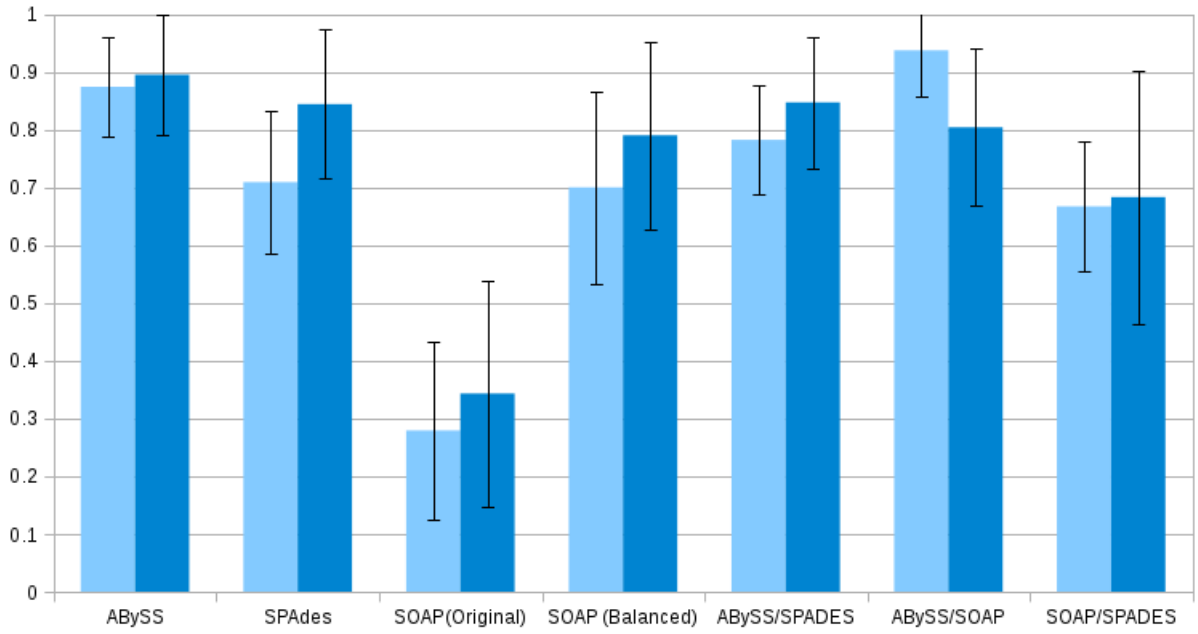


FIGURE 4.1. The true positive rate of the 5CV. The left bars represent the mean TPR before feature reduction and the right bars are after feature reduction. The error bars are the standard deviation over the TPR for the 5CV.

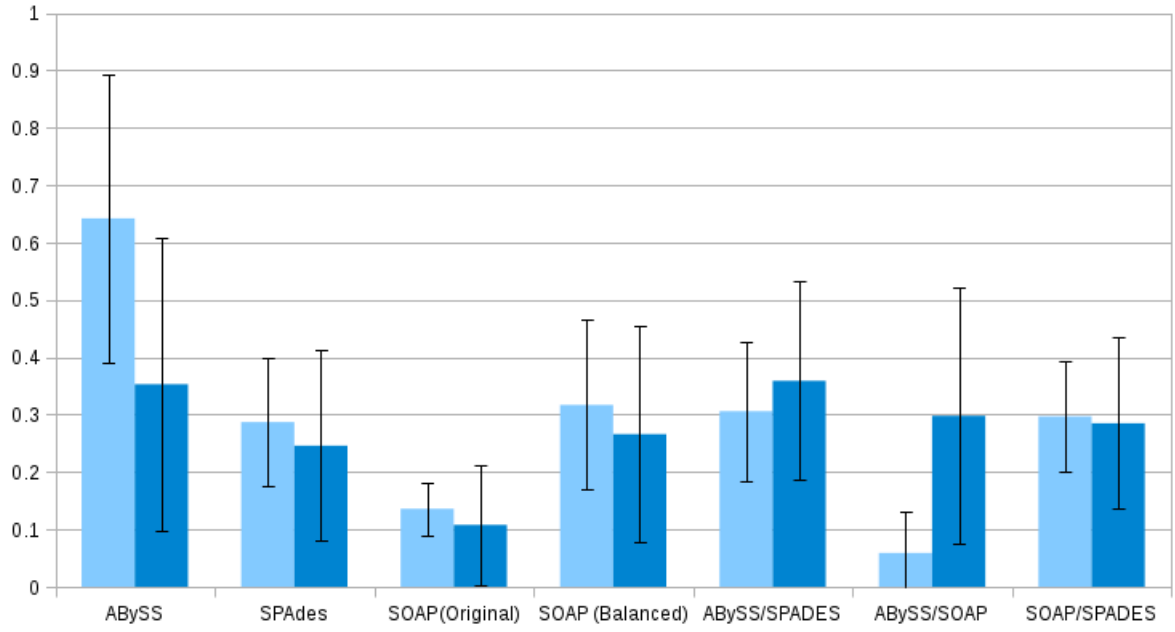


FIGURE 4.2. The false positive result of the 5CV. The left bars represent the mean FPR before feature reduction and the right bars are after feature reduction. The error bars are the standard deviation over the FPR for the 5CV.

TABLE 4.2. The number of positive and negative examples in each machine learning data set. As seen below, SOAPdenovo (original) had far more negative examples than positive examples, which caused the model to be over-fitted. The first seven are from the *Francisella tularensis* data set, while the last example is from *Porphyromonas gingivalis*.

Assembly	**Positive examples (Misassembled contig)**	**Negative examples (Correctly assembled)**	**Total**
ABySS	73	23	96
SOAPdenovo (original)	44	263	307
SOAPdenovo (balanced)	44	44	88
SPAdes	43	57	100
ABySS/SOAP	117	67	184
ABySS/SPAdes	116	80	196
SOAP/SPAdes	87	101	188
P. gingivalis (SPAdes)	14	112	126

4.2. Five-fold Cross Validation

The results of the 5CV for the SVMs trained on each assembly, with all features, are listed in Table 4.3 and Table 4.4. The 5CV results for the SVMs that have been retrained after feature reduction are listed in Table 4.5 and Table 4.6. The TPR and FPR are compared before and after feature reduction in Figure 4.1 and Figure 4.2. The number of positive and negative examples in each of the data sets can be found in Table 4.2.

The assemblies all performed differently, with the combination of ABySS/SOAPdenovo (balanced) having the highest TPR. The original SOAPdenovo showed the poorest TPR, due to over-fitting on the many negative examples in that dataset. The results of the 5CV showed somewhat high variance. This was because the data sets themselves contained few data, which increased the variability between folds.

The combined assemblies had better performance than the individual assemblies by themselves. This is because the combined assemblies had more data and a more balanced set of positive and negative examples. After feature reduction, six of the seven data sets showed

TABLE 4.3. The true positive rate (TPR) results of 5CV on all data sets, with all features.

Assembly	**Mean**	**Std Dev**	**Max**	**Min**
ABySS	0.874	0.085	1.000	0.714
SOAPdenovo (original)	0.279	0.153	0.667	0.000
SOAPdenovo (balanced)	0.700	0.167	1.000	0.250
SPAdes	0.709	0.123	0.889	0.375
ABySS/SOAP	0.617	0.108	0.870	0.375
ABySS/SPAdes	0.782	0.095	0.957	0.565
SOAP/SPAdes	0.548	0.114	0.765	0.333

TABLE 4.4. The false positive rate (FPR) results of 5CV on all data sets, with all features.

Assembly	**Mean**	**Std Dev**	**Max**	**Min**
ABySS	0.642	0.252	1.000	0.000
SOAPdenovo (original)	0.136	0.046	0.308	0.057
SOAPdenovo (balanced)	0.317	0.148	0.667	0.000
SPAdes	0.287	0.111	0.583	0.083
ABySS/SOAP	0.176	0.053	0.368	0.088
ABySS/SPAdes	0.306	0.121	0.563	0.000
SOAP/SPAdes	0.135	0.039	0.250	0.047

TABLE 4.5. The true positive rate (TPR) results of 5CV on all data sets, after feature reduction has completed.

Assembly	**Mean**	**Std Dev**	**Max**	**Min**
ABySS	0.895	0.104	1.000	0.571
SOAPdenovo (original)	0.343	0.196	0.750	0.000
SOAPdenovo (balanced)	0.790	0.162	1.000	0.444
SPAdes	0.844	0.129	1.000	0.625
ABySS/SOAP	0.804	0.136	1.000	0.333
ABySS/SPAdes	0.847	0.114	1.000	0.435
SOAP/SPAdes	0.683	0.219	1.000	0.056

better TPR, while five had better FPR. This is to be expected, as the large number of features caused poorer performance.

4.3. PREDICTED ENZYMES

The goal of this method is to predict the optimal enzymes for creating optical maps. While the TPR/FPR are insightful, they are technically measuring if the classifier can predict

TABLE 4.6. The false positive rate (FPR) results of 5CV on all data sets, after feature reduction has completed.

Assembly	**Mean**	**Std Dev**	**Max**	**Min**
ABySS	0.353	0.255	1.000	0.000
SOAP (original)	0.108	0.104	0.491	0.000
SOAP (balanced)	0.266	0.189	0.778	0.000
SPAdes	0.246	0.166	0.909	0.000
ABySS/SOAP	0.298	0.223	0.833	0.000
ABySS/SPAdes	0.359	0.173	0.688	0.063
SOAP/SPAdes	0.285	0.149	0.650	0.050

misassembly errors and do not directly describe if it selected good enzymes as features. We therefore developed a method of comparison for the predicted enzymes and the optimal enzymes.

In Table 4.8, each predicted enzyme for the ABySS assembly is compared against the optimal enzyme using edit distance between the recognition sequences divided by the sum of the length of the two enzymes. The result will be closer to zero if the enzymes are more similar. The more dissimilar the comparison, the higher this result will be. Table 4.7 shows which of the top eleven optimal enzymes are closest to each predicted enzyme. The enzyme recognition sequences are compared using edit distance because some enzymes have the same or similar recognition sequences. A suboptimal enzyme may still be sufficiently good if its recognition sequence is similar to an optimal enzyme.

Our method was able to predict one of the top eleven enzymes with its first selected feature. Additionally, the seventh selected enzyme is also an exact match with one of the enzymes in the optimal list. Three of the selected enzymes had a normalized edit distance of less than 0.18. The sixth predicted enzyme (AleI) was only different from an optimal enzyme by two nucleotides.

These results suggest that the feature reduction method of enzyme selection is able to predict enzymes that are sufficiently close to optimal.

TABLE 4.7. A comparison of the top 11 best enzymes for ABySS, against which enzymes were selected by the feature reduction. This list comes from the best edit distances from the matrix in Table 4.8. The number is parenthesis is what rank that enzyme is, where 1 is the best enzyme. Note that ∧ is where in the nucleotide sequence the digestion enzyme cleaves the DNA.

Enzyme (Predicted)	**Sequence**	**Enzyme (Optimal)**	**Sequence**	**Edit Distance**
MslI (1)	CAYNN∧_NNRTG	RseI (11)	CAYNN∧_NNRTG	0.000
BetI (2)	W∧CCGG_W	VneI (6)	G∧TGCA_C	0.300
AhlI (3)	A∧CTAG_T	CjeFIII (9)	GCAAGG	0.278
EcoT38I (4)	G_RGCY∧C	HgiAI (7)	GAYNNNNNVTC	0.100
RdeGBI (5)	CCGCAG	CjeFIII (9)	GCAAGG	0.250
AleI (6)	CACNN∧_NNGTG	RseI (11)	CAYNN∧_NNRTG	0.071
CjeFIII (7)	GCAAGG	CjeFIII (9)	GCAAGG	0.000
RdeGBIII (8)	ACCCAG	CjeFIII (9)	GCAAGG	0.250
RpaTI (9)	GRTGGAG	VneI (6)	G∧TGCA_C	0.210
UbaF9I (10)	TACNNNNNRTGT	UbaF13I (1)	GAGNNNNNNCTGG	0.172
Cgl13032II (11)	ACGABGG	CjeFIII (9)	GCAAGG	0.176

TABLE 4.8. A comparison of the top 11 best enzymes for ABySS, against which enzymes were selected by the feature reduction. The first column represents the features selected by the SVM and the first row represents the optimal features. The enzymes were compared using a modified edit distance, where the edit distance between the two recognition sequences was divided by the sum of the length of the two enzymes.

	UbaF13I	**Hpy99XIV**	**Jma19592I**	**Ksp632I**	**Hin4I**	**VneI**	**HgiAI**	**WviI**	**CjeFIII**	**NspI**	**RseI**
MslI	0.207	0.545	0.478	0.310	0.185	0.458	0.500	0.522	0.455	0.458	0.000
BetI	0.480	0.444	0.421	0.400	0.478	0.300	0.400	0.643	0.333	0.300	0.458
AhlI	0.440	0.333	0.368	0.400	0.391	0.300	0.400	0.619	0.278	0.300	0.417
EcoT38I	0.440	0.333	0.316	0.440	0.391	0.200	0.100	0.643	0.389	0.300	0.500
RdeGBI	0.435	0.312	0.353	0.435	0.524	0.278	0.333	0.675	0.250	0.333	0.455
AleI	0.207	0.546	0.478	0.276	0.222	0.458	0.500	0.500	0.409	0.417	0.071
CjeFIII	0.435	0.313	0.294	0.522	0.476	0.333	0.389	0.675	0.000	0.278	0.455
RdeGBIII	0.435	0.313	0.353	0.478	0.476	0.333	0.389	0.675	0.250	0.333	0.455
RpaTI	0.458	0.294	0.278	0.500	0.455	0.211	0.316	0.683	0.235	0.368	0.478
UbaF9I	0.172	0.500	0.435	0.310	0.185	0.500	0.500	0.500	0.455	0.417	0.179
Cgl13032II	0.417	0.353	0.389	0.500	0.455	0.368	0.368	0.659	0.176	0.316	0.435

4.4. CROSS SPECIES EVALUATION

In order to determine the quality and robustness of the classifier against other data, we trained the seven SVMs using all data in the respective sets. These SVMs were then evaluated using training data from the *Porphyromonas gingivalis* genome [**?**]. All assembled contigs over 500 bp were used from this genome. The learned models were trained on

TABLE 4.9. The TPR and FPR for the trained sets, evaluated against the *Porphyromonas gingivalis* data set.

Assembly	TPR	FPR
ABySS	0.643	0.438
SOAP (original)	0.000	0.170
SOAP (balanced)	1.000	0.661
SPAdes	0.071	0.071
ABySS/SOAP	1.000	0.661
ABySS/SPAdes	0.143	0.214
SOAP/SPAdes	0.071	0.188

the *Francisella tularensis* genome, using the same techniques as described in section 3.1. The testing data was created from the *Porphyromonas gingivalis* genome, using the same techniques as described in section 3.1. We used TWIN and misSEQuel to generate the same 2,053 features for the *Porphyromonas gingivalis* data. The SVMs were only evaluated on the data sets with all original 2,053 features and not the data sets after feature reduction. The reason for this is because the digestion enzymes that create a good optical map for one genome may not be the same as the enzymes for another genome.

Table 4.9 shows the result from the cross species evaluation. It is noteworthy that the assemblers had a large range of performance on these data. SOAPdenovo (balanced) and the ABySS/SOAP combination had the highest TPR for the modified data sets, while ABySS had the highest TPR from the original assemblers. These results largely depend on the number of positive examples for the original data. ABySS and SOAPdenovo (balanced) had many positive examples and were able to find more misassembly errors in the *Porphyromonas gingivalis* genome. The possibility of over-fitting is especially apparent in the large difference between the balanced SOAPdenovo data and the original SOAPdenovo data. Note that *Porphyromonas gingivalis* only had 14 misassembly errors in its whole genome, so correctly classifying these examples is difficult.

These results indicate this method is able to make reasonable and informed predictions about the optimal digestion enzymes to use for optical mapping. Over-fitting the model is a concern, especially in the case of small genomes, but training a model with good TPR/FPR is still possible using this technique.

CHAPTER 5

Conclusion

Our contribution is to expand on techniques for using optical mapping data for misassembly error detection. There are few existing methods to detect misassembly errors and few existing methods that make use of the information provided in optical mapping data. We developed a method to predict the optimal digestion enzymes in advance, so that a good optical map can be created for use in *de novo* genome assembly.

This paper formulates the digestion enzyme selection problem for optical mapping and evaluates a practical way to estimate the solution. We show that enzyme selection for optical mapping is both NP-hard and W[1]-hard, so methods to estimate the solution are required. We also propose and evaluate a practical method to solve this problem. Using a support vector machine and feature reduction, we were able to predict two optimal enzymes exactly and predict three enzymes with small edit distance from the list of the top eleven optimal enzymes.

Future work in this area includes evaluating other aspects of the assembly to use as features; there are a wide variety of possible features to use in this machine learning approach. Other machine learning algorithms and feature reduction methods (such as recursive feature selection) should be evaluated as well.

We evaluated each enzyme individually in this paper, but a combinatorial approach has the potential to yield different results. As digestion enzymes are often used in groups of three in the lab, formulating this approach to use groups of enzymes instead of a single enzyme could provide a more accurate estimate.

Finally, while this problem is NP-hard and W[1]-hard, there may be potential approximation algorithms that sufficiently solve the problem. Given the nature of biological data and the fact many enzymes have similar restriction sites, there may exist an approximation algorithm that is able to solve the problem in reasonable time.

CHAPTER 6

Bibliography

[1] T. Anantharaman and B. Mishra. A probabilistic analysis of false positives in optical map alignment and validation. In *Proceedings of WABI*, pages 27–40, 2001.

[2] C. Atson and D.C. Schwartz. *Encyclopedia of Analytic Chemistry*, chapter Optical Mapping in Genomic Analysis. John Wiley and Sons, Ltd, 2006.

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

[4] C. Boucher, C. Lo, and D. Lokshtanov. Outlier detection for dna fragment assembly. *arXiv*, page 1111.0376, 2012.

[5] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[6] P.E.C. Compeau, P.A. Pevzner, and G. Tesler. How to apply de bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11):987–991, 2011.

[7] E.T. Dimalanta et al. Microfluidic system for large DNA molecule arrays. *Analytical Chemistry*, 76(18):5293–5301, 2004.

[8] M.R. Fellows, D. Hermelin, F.A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

[9] A. Gurevich, V. Saveliev, N. Vyahhi, and G. Tesler. QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

[10] David Haussler, Stephen J O'Brien, Oliver A Ryder, F Keith Barker, Michele Clamp, Andrew J Crawford, Robert Hanner, Olivier Hanotte, Warren E Johnson, Jimmy A McGuire, et al. Genome 10k: a proposal to obtain whole-genome sequence for 10 000 vertebrate species. *Journal of Heredity*, 100(6):659–674, 2009.

[11] Geoffrey Holmes, Andrew Donkin, and Ian H Witten. Weka: A machine learning workbench. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 357–361. IEEE, 1994.

[12] M. Hunt, T. Kikuchi, M. Sanders, C. Newbold, M. Berriman, and T. D. Otto. REAPR: A universal tool for genome assembly evaluation. *Genome Biology*, 14:R47, 2013.

[13] R.M Idury and M.S. Waterman. A new algorithm for DNA sequence assembly. *Journal of Computational Biology*, 2(2):291–306, 1995.

[14] Lucian Ilie, Bahlul Haider, Michael Molnar, and Roberto Solis-Oba. Sage: String-overlap assembly of genomes. *BMC bioinformatics*, 15(1):302, 2014.

[15] Sergey Koren, Todd J. Treangen, Christopher M. Hill, Mihai Pop, and Adam M. Phillippy. Automated ensemble assembly and validation of microbial genomes. *BMC Bioinformatics*, 15(126), 2014.

[16] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[17] M. Li, B. Ma, and L. Wang. Finding similar regions in many sequences. *J. Comput. System Sci.*, 65(1):73–96, 2002.

[18] R. Li et al. *De Novo* assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, 2010.

[19] H.C Lin, S. Goldstein, L. Mendelowitz, S. Zhou, J. Wetzel, D.C. Schwartz, and M. Pop. AGORA: Assembly Guided by Optical Restriction Alignment. *BMC Bioinformatics*, 12:189, 2012.

[20] L. Mendelowitz and M. Pop. Computational methods for optical mapping. *GigaScience*, 3, 2014.

[21] Lee Mendelowitz and Mihai Pop. Computational methods for optical mapping. *GigaScience*, 3(1):33, 2014.

[22] M. Muggli, S. Puglisi, and C. Boucher. Efficient Indexed Alignment of Contigs to Optical Maps. In *Proceedings WABI*, pages 68–81, 2014.

[23] Martin D Muggli, Simon J Puglisi, and Christina Boucher. Efficient indexed alignment of contigs to optical maps. In *Algorithms in Bioinformatics*, pages 68–81. Springer, 2014.

[24] M.D. Muggli, S.J. Puglisi, R. Ronen, and C. Boucher. Misassembly detection using paired-end sequence reads and optical mapping data. *To appear in the Journal of Bioinformatics*, 2015.

[25] N. Nagarajan, T.D. Read, and M. Pop. Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics*, 24(10):1229–1235, 2008.

[26] R. K. Neely, J. Deen, and J. Hofkens. Optical mapping of DNA: Single-molecule-based methods for mapping genome. *Biopolymers*, 95(5):298–311, 2011.

[27] P.A. Pevzner, H. Tang, and M.S. Waterman. An eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[28] A. Phillippy, M. Schatz, and M. Pop. Genome assembly forensics: Finding the elusive mis-assembly. *Genome Biology*, 9(3):R55+, 2008.

[29] Richard J Roberts, Tamas Vincze, Janos Posfai, and Dana Macelis. Rebasea database for dna restriction and modification: enzymes, genes and genomes. *Nucleic acids research*, page gkp874, 2009.

[30] R. Ronen, C. Boucher, H. Chitsaz, and P. Pevzner. SEQuel: improving the accuracy of genome assemblies. *Bioinformatics*, 28(12):i188–i196, 2012.

[31] S.L. Salzberg. Beware of mis-assembled genomes. *Bioinformatics*, 21(24):4320–4321, 2005.

[32] Stephan C Schuster. Next-generation sequencing transforms today's biology. *Nature methods*, 5(1):16–18, 2008.

[33] J.T. Simpson, K. Wong, S.D. Jackman, J.E. Schein, S.J. M. Jones, and I Birol. ABySS: A parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009.

[34] Todd J. Treangen, Dan D. Sommer, Florent E. Angly, Sergey Koren, and Mihai Pop. *Next Generation Sequence Assembly with AMOS*, volume 11. John Wiley & Sons, Inc., 2011.

[35] B.J. Walker et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One*, 9(11):e112963, 2014.

[36] B.B. Xavier, J. Sabirova, M. Pieter, J.-P. Hernalsteens, H. de Greve, H. Goossens, and S. Malhotra-Kumar. Employing whole genome mapping for optimal *de novo* assembly of bacterial genomes. *BMC Research Notes*, 7:484, 2014.

[37] XiaoGuang Zhou, LuFeng Ren, YunTao Li, Meng Zhang, YuDe Yu, and Jun Yu. The next-generation sequencing technology: a technology review and future perspective. *Science China Life Sciences*, 53(1):44–57, 2010.