Comenius University Bratislave
Faculty of Mathematics, Physics and Informatics

Alignment of Nanopore Sequencing Squiggles

Bachelor thesis

2017
Jakub Havelka

Comenius University Bratislave

Faculty of Mathematics, Physics and Informatics

Alignment of Nanopore Sequencing Squiggles

Bachelor thesis

Study programme:	Computer Science
Study field:	2508 Informatics
Department:	Department of Computer Science
Supervisor:	doc. Mgr. Tomáš Vinař, PhD.

Bratislava, 2017

Jakub Havelka

63565718

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Jakub Havelka
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Alignment of Nanopore Sequencing Squiggles
Zarovnávanie signálov z nanopórového sekvenovania

Cieľ: MinION je platforma sekvenovania DNA, ktorá produkuje dlhé čítania s vysokým stupňom chybovosti. Jednou z príčin vysokej chybovosti je to, že elektrický signál, ktorý prístroj produkuje, je potrebné preložiť do DNA sekvencií a tento proces je zdrojom veľkého množstva chýb. Cieľom práce je preskúmať možnosti hľadania podobností medzi čítaniami priamym porovnávaním elektrických signálov a rýchlo identifikovať podobné sekvencie bez toho, aby elektrické signály bolo potrebné prekladať do DNA sekvencií.

Vedúci:	doc. Mgr. Tomáš Vinař, PhD.
Katedra:	FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry:	prof. Ing. Igor Farkaš, Dr.
Dátum zadania:	27.10.2016

Dátum schválenia: 31.10.2016

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

....
študent

....
vedúci práce

63565718

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Jakub Havelka
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Alignment of Nanopore Sequencing Squiggles

Aim: MinION DNA sequencing platform produces long reads with high error rates. One of the reasons for high error rates is that electrical signals produced by the sequencing machine need to be first translated into DNA sequences by a process called base calling, and this process is error prone. The goal of this thesis is to explore the possibilities of locating similar reads by directly comparing electrical signals (so called squiggles) and quickly identify similar sequences without performing base calling.

Supervisor:	doc. Mgr. Tomáš Vinař, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	27.10.2016
Approved:	31.10.2016

doc. RNDr. Daniel Olejár, PhD.
Guarantor of Study Programme

.... Student

.... Supervisor

Acknowledgement: I would like to thank Tomáš Vinař for his guidance, patience and consultations. I would also like to thank Broňa Brejová for her advice.

I am also thankful to my parents for their support, Ivka for her kind words when I needed them the most and Matúš for letting me know I am not alone.

Abstrakt

Nanopórové sekvenovanie je najnovšia technológia sekvenovania produkujúca dlhé čítania, ktoré však obsahujú veľké množstvá chýb. Jedným zo zdrojov chýb je preklad elektrických signálov produkovaných pri sekvenovaní do DNA sekvencií. Tieto DNA sekvencie sa ďalej zarovnávajú bežnými algoritmami k referenčnej sekvencií. V tejto práci navrhujeme spôsob, akým vieme obísť preklad signálov na DNA sekvencie a ku referenčnej sekvencii zarovnávať priamo elektrické signály.

Kľúčové slová: nanopórové sekvenovanie, dynamic time warping, zarovnávanie

Abstract

Nanopore sequencing is the latest in sequencing technologies, that is producing long reads with hight error rates. One of the sources of these errors are mistakes during translation from electrical signals produced to DNA sequences. These sequences are then usually aligned to a reference sequence using common algorithms for sequence alignment. In this thesis, we propose aligning the electrical signals directly to the reference sequence, bypassing the translation prone to errors.

Keywords: nanopore sequencing, dynamic time warping, alignment

Contents

List of Figures

List of Tables

Introduction

The latest technology in DNA sequencing is Nanopore sequencing. It is based on measuring the current flowing through the nanopore. As a DNA molecule is passing through the nanopore, it alters the current flow. Disruption in the current are then used to determine the structure of the DNA molecule via base calling. However, base calling is a slow process that introduces a large amount of translation errors into the sequence. The data produced from a single molecule measurement, including current and DNA sequences, is called a read.

The most common task to do with sequenced DNA is to align it to a reference genome. In some aplications, we do not need to know the DNA sequence of a sample. For example, we may only need the sequence of one species in a mixed sample. In this case, a traditional aproach would be to base call all of the sampled current sequences and align them with a reference genome to determine the species it belongs to. However, a large amount of the reads may not belong to the studied species, and the resources used to process these reads end up being useless.

The goal of this thesis is to explore different aproaches of aligning the current measurements without base calling and assess their viability. In theory, this should allow us to more quicly isolate selected regions from a sampled sequence, along with introducing less errors into the alignment. Our proposed solution is to use dynamic time warping (DTW). We test several methods to preprocess data for the DTW, postprocess the results, as well as test several scoring functions used in DTW.

In the first chapter, we describe the biological aspect of nanopore sequencing in more detail. The second chapter contains description of DNA sequence alignment, along with traditionaly used algorithms. The third chapter describes squiggles, the sequences of current measurements, in more detail. We introduce the problem of squiggle alignment, as well as Dynamic Time Warping. In the fourth chapter, we describe our testing procedure, proposed variations of squiggle processing and detailed results of our testing. Lastly, in the final chapter we provide an overview of the thesis and possible applications.

Chapter 1

Biological Introduction and Motivation

In this chapter, we introduce concepts in genetics and bioinformatics, along with motivation, problem, and a brief outline of our proposed solution of the fast MinION read alignment.

1.1 Sequencing DNA

Genetic information of all known cellular living organisms is stored in the form of deoxyribonucleic acid (DNA). DNA consists of two biopolymer strands (template and complement) composed of nucleotides, and has a double helical shape. These nucleotides are composed of a phosphate group, deoxyribose and one of four nitrogenous nucleobases (**bases**): adenine, cytosine, thymine, and guanine. We often represent them by single letter abbreviations: A, C, T, G respectively.

Chemical bonds between bases bind the two DNA strands together. The bonds are complementary - adenine in one strand bonds only with thymine in the other and cytosine bonds with guanine. Thus, identifying a nucleotide sequence of the template DNA strand suffices to identify the sequence of the complement strand, and vice versa.

DNA sequencing is the process of determining the sequence of nucleotides in DNA strands. None of the available sequencing technologies is capable of sequencing entire genome at once. Instead, they produce shorter subsequences, referred to as **reads**. First and second generation sequencers produce short reads in range of 50-1000 base pairs (bp) [11]. Third generation sequencers produce much longer reads (up to 200 kbp), but generally have a significantly higher error rate [5].

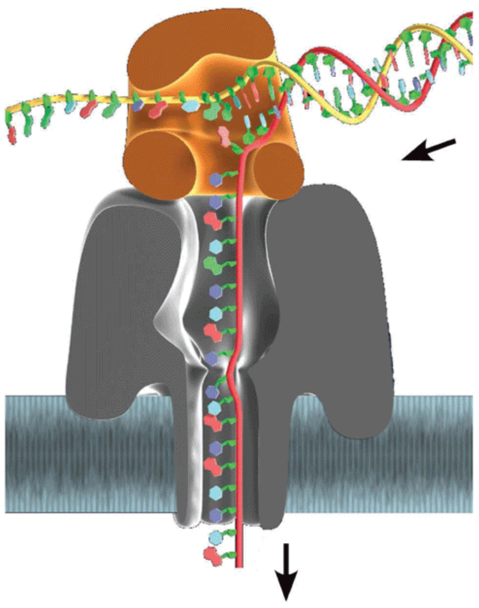

Figure 1.1: Protein nanopore letting one strand of DNA pass through [9]

1.2 MinION

MinION is a third generation, highly portable sequencer developed by Oxford Nanopore Technologies. A nanopore is a miniature opening approximately one nanometer wide. Pores this small are made in membranes with embedded proteins or in sheets of graphene.

MinION uses an array of nanopores large enough to allow passage of one DNA strand, but not both (see Figure 1.1). Hundreds of protein nanopores are embedded into a polymer membrane in a singe use module called flow cell. An ionic solution is placed onto both sides of a membrane. A small voltage is then applied across the membrane, which induces a small ionic current through the nanopores. A sample of DNA is then applied to one side of the membrane.

A strand of DNA passing through the nanopore disrupts the flow of ions, thus changing the measured electric current. The current is affected differently by specific nucleotides, meaning the change in the current is depends directly on the DNA sequence passing through. Ideally, the current change will depend on a single base. In practice, k bases residing in the pore interact. Such a continuous subsequence is called a k-mer. As the DNA strand passes through the pore, the k-mer exposed to the pore changes and so change the current measurements. We call the current measurement sequence a **squiggle**.

This current is measured thousands of times per second. To simplify the processing, the raw sequence of squiggles is split into a sequence of **events** (see example in Figure 1.2). Events condense a series of similar current measurement into a single object, thus greatly reducing data output. Ideally, each event should correspond to a single k-mer.

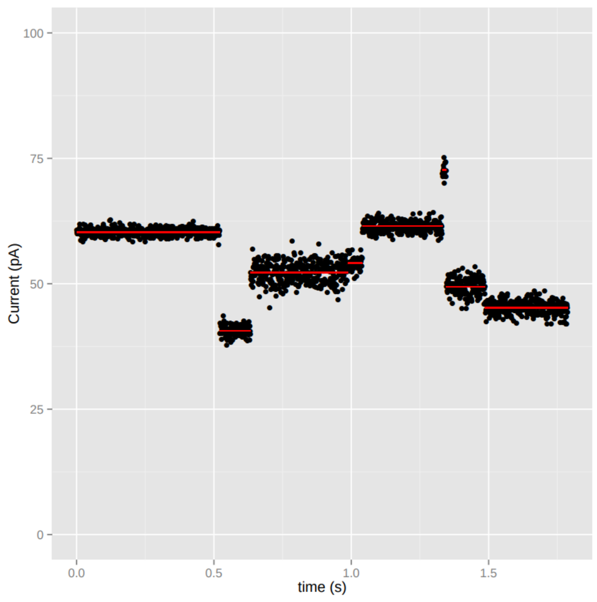

Figure 1.2: Example graph of current measurements, red lines represent events

1.3 Base Calling

Base calling is a process in which events are translated into a sequence of bases (letters A, C, G, T). Raw values, events, results of base calling or a combination of above are stored in **FAST5** file format. Events are described using four parameters - mean value of the current, standard deviation, start of the event and duration of the event.

Base calling is a slow process, which additionally introduces large amount of translational errors. After base calling, we might wish to compare the resulting DNA sequence with a reference database. As an example, we want to determine whether a DNA sample belongs to a known species. In this work, we propose to skip base calling step, and compare raw current measurements. This should allow for faster and more reliable processing.

Chapter 2

Alignment of DNA Sequences

In this chapter, we formalize formalize the problem of aligning two DNA sequences. We also present an overview of standard algorithms solving this problem.

2.1 Sequence Alignment

The sequence alignment is likely the most crucial task in bioinformatics. The purpose of aligning two sequences is to find similarities. These similarities prove useful when analysing organisms. For example, finding similar regions in different organisms likely means these regions have the same biological function, and such alignment also provides an evolutionary connection.

Definition 2.1.1. Let $u = u_1 \dots u_n$, $v = v_1 \dots v_m$ be two sequences and M be a matrix

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & M_{1,3} & \dots & M_{1,k} \\ M_{2,1} & M_{2,2} & M_{2,3} & \dots & M_{2,k} \end{pmatrix}$$

. We call the matrix M an alignment of sequences u and v, if:

1. $\forall M_{i,j} : M_{i,j} \in \{A, C, T, G, -\}$
2. $M_{1,1} \dots M_{1,k}$ is a word created by inserting dashes into u
3. $M_{2,1} \dots M_{2,k}$ is a word created by inserting dashes into v
4. No two dashes occupy the same column

For example, let $u = TGAAGGCCT$ and $v = GAAGGTCTAA$. One of many possible alignments of these sequences is

```
TGAAGGCC-T--
-GAAGGTCCTAA
```

. In this alignment we see four types of columns:

- *Match:* A column containing two identical bases.
- *Mismatch:* A column containing two different bases.
- *Insertion:* The first row contains a dash.
- *Deletion:* The second row contains a dash.

We can easily see that any alignment contains only these four types of columns.

Another example alignment of our sequences u, v may be

```
TGAAG-GCCT------
-----GAAGGTCCTAA
```

While it is still formally an alignment, it is of a poor quality and is of very little practical application.

Typically, we establish a *scoring system* and try to find an "optimal" alignment in this system. The scoring system usually assigns a numerical value for each column of the alignment, which are then summed up to form a *score* of the alignment. A simple example scoring system may assign $+1$ to match columns and -1 to mismatch columns, insertions and deletions.

Definition 2.1.2 (Global alignment)**.** Given two sequences u, v and a scoring system, global alignment of these sequences is an alignment with the highest score.

In some cases we may not wish to align entire sequences. As a real-world example, we may want to find similar areas in genomes of two different species, which will likely contain biologically important information.

Definition 2.1.3 (Local alignment)**.** Given sequences $u = u_1 \dots u_n$, $v = v_1 \dots v_m$, we want to find two subsequences $u_i \dots u_j, v_k \dots v_l$ whose global alignment has the highest score in a given scoring system.

2.2 DNA Sequence Alignment Algorithms

The problem of global alignment is typically solved using *Needleman-Wunsch* algorithm. For the purpose of demostrating this algorithm, we will use the simple scoring system mentioned above: $+1$ for mathes, -1 for mismatches, insertions and deletions.

Let $u = u_1 \dots u_n,\ v = v_1 \dots v_m$ be two input sequences. Needleman-Wunsch algorithm computes a matrix A, where $A[i, j]$ is the largest alignment score of sequences $u_1 \dots u_i$ and $v_1 \dots v_j$.

For cases where $i = 0$ or $j = 0$, there is only one possible alignment. In the case of $i = 0$, we are aligning an empty sequence with $v_1 \dots v_j$, which is only possible using j insertions. The score of such alignment is $-j$. Similarly, the only alignment for a case where $j = 0$ is i deletions, which has a score of $-i$.

Concider the case where $i, j > 0$. We define a function $s(a, b)$, representing the score of match and mismatch columns, as

$$s(a,b) = \begin{cases} 1, & a = b \\ -1, & otherwise \end{cases}$$

There exists an optimal alignment M_{i_j} between $u_1 \dots u_i$ and $v_1 \dots v_j$. Let us analyse the score of this alignment for each case of the last column of this alignment.

Concider the case where the last column contains both u_i and v_j. The score for this column is $s(u_i, v_j)$. By removing this column, we get an alignment between $u_1 \dots u_{i-1}$ and $v_1 \dots v_{j-1}$. This alignment is optimal, as existence of a better alignment would contradict the optimality of $M_{i,j}$. Therefore, the score for alignment $M_{i,j}$ is $A[i-1, j-1] + s(u_i, v_j)$.

If the last column of $M_{i,j}$ contains u_i and a dash, we can make a similar argument, with the score being equal to $A[i-1, j] - 1$, where -1 is the score of the last column. Similarly, if last column contains a dash and v_j, whe score is $A[i, j-1] - 1$.

The final equation for alignment score is:

$$A[i,j] = max \begin{cases} A[i-1, j-1] + s(u_i, v_i) \\ A[i-1, j] - 1 \\ A[i, j-1] - 1 \end{cases}$$

Value of each cell of the matrix thus depends solely on cells to the top and the right of it. This means we can iterate over the matrix using dynamic programming and compute the score for the entire matrix in $O(nm)$. The optimal alignment of sequences $u = u_1 \dots u_n,\ v = v_1 \dots v_m$ is $A[n, m]$.

Local alignment is computed using *Smith-Waterman* algorithm. In principle, it is very similar to Needleman-Wunsch, with two major differences. Firstly, the value of $A[i, j]$ is equal to:

$$A[i,j] = max \begin{cases} A[i-1, j-1] + s(u_i, v_i) \\ A[i-1, j] - 1 \\ A[i, j-1] - 1 \\ 0 \end{cases}$$

This allows the algorithm to choose the starting point of the alignment. Secondly, the optimal score is the maximal value in the matrix rather than $A[n, m]$, allowing the algorithm to choose the end of the alignemnt.

Chapter 3

Squiggles and Alignment

In this chapter we describe squiggles in more detail, along with more technical overview of estabilished workflow in nanopore sequencing. We define the problem of squiggle alignment as a bypass of base calling followed by DNA sequence alignment. Finally, we present our proposed algorithm, DTW, as a solution of squiggle alignment problem.

3.1 Squiggles

At 4 000 measurements per second, a single nanopore produces large amount of data to process. Sampling rate needs to be high to ensure that current is measured for each k-mer passing through the nanopore. Sequence of current measurements are too large to process by non-linear algorithms in a reasonable time. Because of this, the measurements are split into events.

An event is a representations of several contiguous measurements of the current. Events are fully defined by three parameters: mean value, standard deviation and duration.

The sequence of current measurements is split into events in such a way that one event should, in theory, correspond to one k-mer passing through the nanopore. In practice, however, miscellaneous errors may be present in the sequence. These errors then get picked up by the event detector in the form of either skipping or duplicating an event. An event might be *skipped*, if the current measured for previous k-mer is similar to the next one. On the other hand, a single k-mer might be represented by multiple events due to noise in the current measurements, that might be similar to another k-mer.

Two events with the same parameters, on the same setup, should represent the same k-mer. By extension, two similar sequences of events should represent similar sequences, allowing us to align two squiggles and determine their similarity without base calling them first.

3.2 Base calling

As described in chapter 1, base calling is a process of translating raw output from sequencing into DNA sequences.

A pore model is provided by Oxford Nanopore [10]. This model describes expected event parameters for each k-mer. An example model for $k = 5$ might look like this:

5-mer	μ_k	σ_k
`AAAAA`	53.5	1.3
`AAAAC`	54.2	0.9
$\vdots$	$\vdots$	$\vdots$
`TTTTT`	67.1	1.4

This means that measured current while 5-mer `AAAAC` is in the pore is expected to be drawn from Gaussian distribution $\mathcal{N}(54.2, 0.9^2)$.

Reality is, unfortunately, not that simple. Each nanopore produces slightly different readings for the same k-mers. Current may be shifted by absolute value, scaled by some constant or drift during reading. Metrichor computes these constants during base calling, and stores scaled events alongside base called sequence.

3.3 Direct Squiggle Alignment with Dynamic Time Warping

The base calling is a relatively slow process that introduces translation errors into the resulting sequences. However, for some applications, the base calling is not necessary. For example, we may only wish to determine whether a read belongs to a paticular organism, without the need to know the exact DNA sequence.

The simplest, naive approach is to assume that squiggles both have the same linear timescale and try to align them consecutively (and thus assume each event aligns with exactly event from the other squiggle, barring the ends).

3.4 Dynamic time warping

To address the problem of changing time scales in the squiggles, *Mathew Loose, et al.* proposed to use a technique from speech processing called *dynamic time warping* (DTW) [6].

DTW is widely used in music processing, speech recognition, biometric identification and more [1] [4].

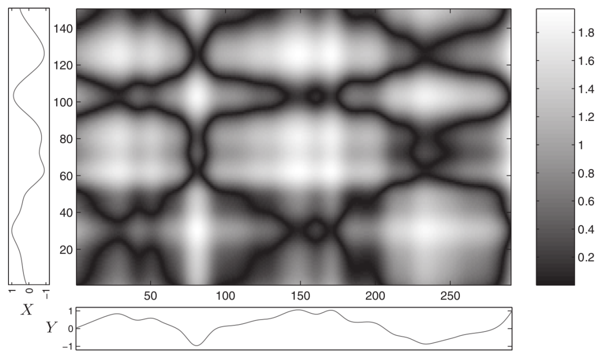

Figure 3.1: Example sequences X, Y and a graph of their cost matrix. [7]

Definition 3.4.1. Let $X = x_1 \dots x_n,\ Y = y_1 \dots y_m$ be two sequences. We define a **local cost** function as a function $c : R \times R \mapsto \mathbb{R}^+$ representing a "distance" of two elements of sequences X, Y. Function $c(a, b)$ shall be small when a is "similar" to b and large when a and b are "dissimilar".

By evaluating this cost for each pair x_i, y_j, we obtain a cost matrix $C(i, j) = c(x_i, y_j)$ (see Figure 3.1 for example).

In DTW, the problem of aligning two sequences then becomes the one of finding a **warping path** of minimal cost in C. Formally, an (N, M)-warping path is a sequence $p_1 \dots p_L$, where p_l is a pair (i, j) of positions in X and Y respectively, such that:

$$p_1 = (1, 1),\ p_i = (n_i, m_i),\ p_L = (N, M), \tag{3.1}$$

$$n_i \leq n_{i+1} \,|\, 1 \leq i < N;\ m_j \leq m_{j+1} \,|\, 1 \leq j < M, \tag{3.2}$$

$$p_{l+1} - p_l \in (0, 1), (1, 0), (1, 1). \tag{3.3}$$

Condition (3.1) is a boundary condition: we want to align entire sequence X to sequence Y.

Condition (3.2) implies that we may not at any point reverse a part of a sequence and, for example, align it multiple times.

Finally, condition (3.3) limits step of the warping path. Without it, we may skip any element of the sequence, even the entire sequence, except for the first and the last element.

Similarly to aligning squiggles, we introduce a scoring function to evaluate a patic-ular alignment. While in DNA sequence alignment, we talked about scoring function, it is more reasonable to call it penalty for DTW, as smaller cost values mean more similar elements.

The penalty of the alignment of X, Y is then a sum of the cost of points in the corresponding warping path: $score = \Sigma_{i=1}^{L} c(p_i)$.

3.4.1 Implementation

Similarly to DNA sequence alignment, a standard implementation of DTW is based on dynamic programming. For input sequences $x_1 \dots x_N, y_1 \dots y_M$ we define a matrix $A_{n,m}$, where $A[i,j]$ contains the smallest penalty for alignment of subsequences $x_1 \dots x_i, y_1 \dots y_j$. Comdition 3.3 directly implies that the path may not skip an element from X and Y, so similarly to cost matrix in DNA sequence alignment, one cell in the matrix depends only on values of three cells directly to the left, above, and diagonal from it. Thus, the cost of $A[i,j]$ is:

$$A[i,j] = c(x_i, y_j) + min \begin{cases} A[i-1, j-1] \\ A[i-1, j] \\ A[i, j-1] \end{cases}$$

The special cases not covered by this definition are cases where $i = 1$ or $j = 1$. If both $i = 1$, and $j = 1$, there is only one possible warping path $(1,1)$. We define $A[1,1]$ as a sum of this path, which is $A[1,1] = c(x_1, y_1)$. For cases where $i = 1$, there still exists only one path, leading directly to $(1,1)$. We define $A[1,j] = A[1,j-1] + c(x_1, y_j)$. Similarly for $j = 1$, $A[i,1] = A[i-1,1] + c(x_i, y_1)$.

DTW defined and implemented in this way ressembles Needleman-Wunsch algorithm for global sequence alignment. Defining local alignments on squiggles is, however, not practical. The scoring function in Smiths-Waterman algorithm contained both positive and negative numbers, thus allowing for "resetting" a cell when its score became negative. The cost function in DTW does not allow this. We can, however, modify DTW to produce *global-local alignments* with relative ease.

Let $X = x_1 \dots x_N$, $Y = y_1 \dots y_M$. If an alignment completely covers sequence X (from x_1 to x_N, meaning it is global from its point of reference), but covers only a portion of sequence Y (y_i to y_j, thus being local), we call such an alignment a global-local alignment. Finding such an alignment means X is a subsequence of Y.

To allow for finding this alignment, we need to modify condition 3.1 to $p_1 = (1,i), p_L = (N,j),\ 1 \leq i \leq j \leq M$.

3.4.2 Warping path problem optimizations

Computing total cost of warping path for all combination of prefixes of X, Y requires lot of computational time still. We can further minimize work required by imposing restrictions on our alignment. If we wish to exclude alignments with low overlap, we may restrict warping path to a narrow Sakoe-Chiba band from $(1,1)$ to (N,M) [7]. See Figure 3.2.

While this reduces the amount of possible warping paths to be examined, it is not compatible with computing global-local alignments.

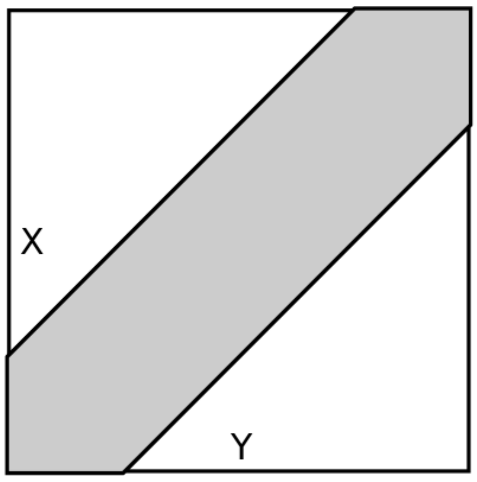

Figure 3.2: Sakoe-Chiba band for limiting possible warping paths

Chapter 4

Empirial Evaluation And Experiments

In this chapter we present our testing methodology, datasets and variations

4.1 Scoring Schemes and Test Variations

We tested several scoring schemes, methods of squiggle reprocessing and various combinations thereof.

4.1.1 Raw Squiggle

The first test will be composed of aligning squiggles without any modifications.

4.1.2 Weighted DTW

In the first iterations of testing, the alignments produced were very "slanted" - a low number of events in one squiggle aligned with the entire second squiggle. To mitigate this, we introduced weighting into our implementation of DTW, allowing us to tag alignment with less "clumping" as more preferable.

4.1.3 Scaling

Even event pre-scaled by Metrichor may not be scaled to the same levels. For this reason, we uniformly scale each squiggle to a common range of values. The uniform scaling itself is done by adding and multiplying each event in the squiggle by a constant.

Min-Max

We uniformly scale the squiggle in such a way that the minimal value is 0 and maximal value is 1.

Mean-Max

Similarly to previous scaling, we uniformly scale the squiggle by a constant. In this case, we shift the mean value to 0 and maximal absolute value to 1. This should mitigate the influence of outlying values that may othervise skew the scaling.

Mean-SD

Yet again, we apply uniform linear transformation to the squiggle. This time, we scale the sequence such that resulting mean value is 0 and standard deviation is 1. We believe this further decreases the influence of outlier values.

4.1.4 Distance function

Distance function is a function that recieves a single value from either squiggle and returns the "distance", or "dissimilarity" of these events.

Euclidean distance

We may define a distance of two events a, b simply as their euclidean distance, $d = abs(a - b)$.

Square of distance

Defining distance as $d = abs(a - b)^2$ will lessent the impact of slight dissimilarities in the squiggles, but it increases the impact of outliers.

4.1.5 Output normalization

An output of DTW in simply the sum of distances along the minimum-cost path. This is fine for comparing alignments within the same pair of squiggles, but when comparing two different pairs, the score is inherently biased towards the alignment of shortest length.

It may be benefitial to divide the final alignment score by the length of the squiggles or alignment path lenght.

4.1.6 Difference Array

It may be beneficial to look only at the differences between events in a squiggle, and align those.

4.2 Design of the Experiment

We used data from *NA12878 Human Reference on Oxford Nanopore MinION [3]*, available for download at `https://github.com/nanopore-wgs-consortium/NA12878`. The dataset contains reads with raw current sequences, events and base called sequences. These are stored as **FAST5** files. FAST5 is a variant of HDF5, a hierarchical binary data format, designed for storing large datasets.

Base called sequences have been pre-aligned against 1000 genomes GRCh38 BWA database using `BWA-MEM`. Output of `BWA-MEM` has been provided as a compressed `BAM` file. Specifications for BAM and SAM formats can be found at `http://samtools.github.io/hts-specs/`.

Four our testing we used a single 10 000 000 base long region of chromosome 14. We extracted base called DNA sequence and events from these reads, along with their pre-aligned position to reference genome.

We split the DNA sequence into 1000 base long "chunks", with splitting positions being multiples of 1000 in reference genome (a sequence aligned with positions $[1047, 4542]$ would produce two chunks: $[2000, 3000]$ and $[3000, 4000]$). For each chunk we extracted the corresponding squiggle.

In the end, we created a dataset composed of squiggles with their known positions when base called and aligned to a reference genome.

Our testing set consists of 500 pairs of squiggles with the same position relative to the reference genome. As a control group, we chose 500 pairs of squiggles of different positions.

Our implementation of DTW is based upon Python module made by Pierre Rouanet [?]. We made modifications to allow for different aproaches for aligning squiggles.

4.3 Evaluation

A test of each proposed combination of preprocessing and scoring function has been executed both testing sets. The output of each test is a list of lowest alignment penalty for each squiggle pair of the testing set.

The only way to identify good squiggle alignments outside of our testing setup is the minimal penalty. With this in mind, we identified a maximal cut-off score for each test, defined as the lowest penalty from the bad alignments set. We then evaluated each test based on the number of good alignments with penalties lower than cut-off threshold.

4.4 Results

The following table contains a statistical overview of alignment penalties for each test. Tests marked `good` were run on matching squiggle pairs, `bad` were run on control group.

Test name	Minimum	Maximum	Mean value
good_euclid	5156.90	89064.40	18505.28
bad_euclid	8326.14	79097.73	18612.28
good_sqrdst	24783.75	4270886.98	349860.90
bad_sqrdst	62595.28	3859561.65	344284.03
good_minmax_euclid	69.19	203.74	150.46
bad_minmax_euclid	128.28	252.65	157.17
good_minmax_sqrdst	4.57	30.60	19.59
bad_minmax_sqrdst	16.33	34.90	20.74
good_meanmax_euclid	131.14	399.71	288.66
bad_meanmax_euclid	220.20	496.01	303.18
good_meanmax_sqrdst	16.68	118.23	72.26
bad_meanmax_sqrdst	46.72	128.44	77.14
bad_meansd_euclid	665.51	1298.50	794.90
good_meansd_sqrdst	114.51	758.62	500.29
good_meansd_euclid	345.73	1047.34	758.46
bad_meansd_sqrdst	448.65	1159.97	532.40
good_meansd_euclid_max	0.15	0.46	0.38
bad_meansd_euclid_max	0.07	0.46	0.40
bad_meansd_euclid_min	0.37	0.72	0.46
good_meansd_euclid_min	0.21	0.60	0.44
good_meansd_euclid_path	0.16	0.50	0.36
bad_meansd_euclid_path	0.27	0.62	0.38
good_meansd_euclid_gloloc	342.14	1985.39	782.27
bad_meansd_euclid_gloloc	675.04	2082.06	820.30
good_meansd_euclid_gloloc_max	0.15	0.48	0.39
bad_meansd_euclid_gloloc_max	0.13	0.48	0.40
good_meansd_euclid_gloloc_min	0.21	1.14	0.45
bad_meansd_euclid_gloloc_min	0.36	1.19	0.47
good_meansd_euclid_gloloc_path	0.16	0.49	0.36
bad_meansd_euclid_gloloc_path	0.14	0.58	0.37

Table 4.1: Experimental results

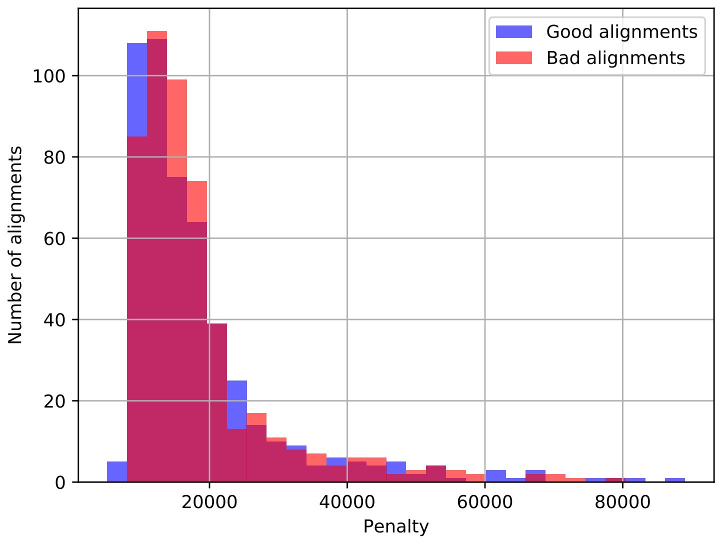

Figure 4.1: Euclidean distance, no preprocessing. The input dataset were two unmodified squiggles directly extracted from the input files. We ran our DTW implementation on both `good` and `bad` data sets, using the euclidean distance function. As we can see, there is no noticable difference in the resulting penalty values.

4.5 Analysis

In this section we analyse the results of our experiment. Firstly, we compare tests focused on squiggle preprocessing.

- *No preprocessing*

  Without any preprocessing, the `good` and `bad` data sets are not recognisable purely by the minimal warping path penalties. We proceeded to test prescaling methods to provide meaningful results.

- *Min-Max prescaling*

  For these two tests we used Min-Max rescaling with both distance functions. This test provided some positive results, see figures 4.5 and 4.5. While the `good` and `bad` sets are not clearly distinguished by their pealty values, up to 10.4% of the `good` alignments had their penalties lower than the `bad` set, making this a viable option for further examination.

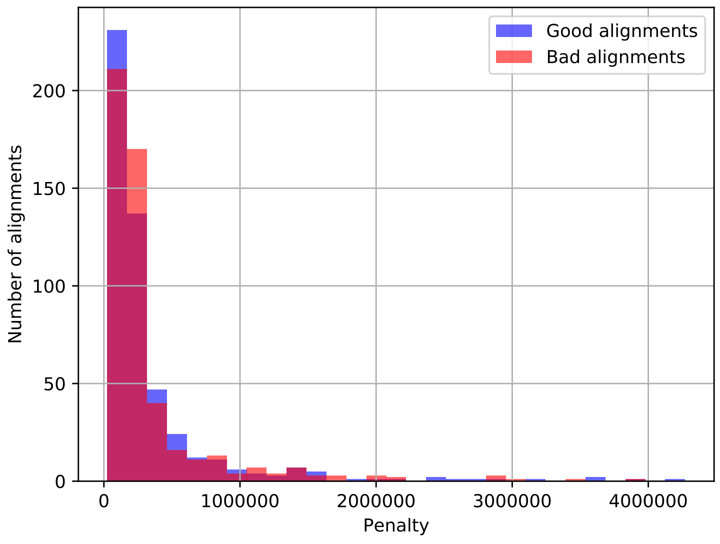

Figure 4.2: Square distance, no preprocessing. On the same setup as the previous test, we saw only marginal difference after switching to square distance function.

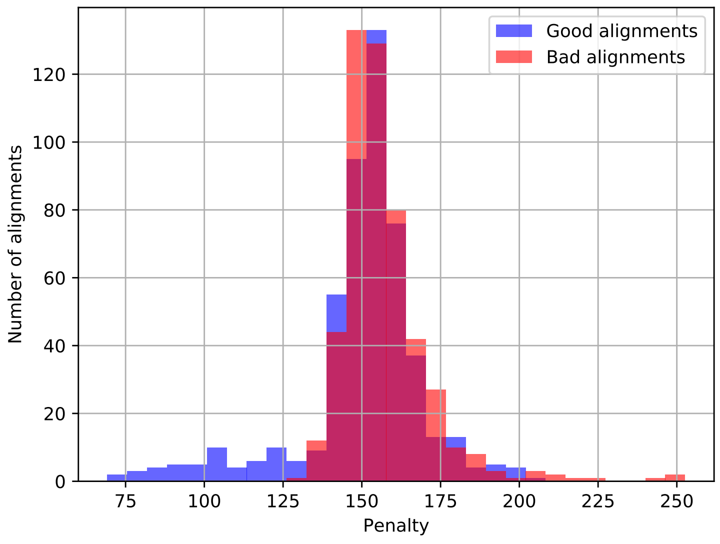

Figure 4.3: Euclidean distance, Min-Max prescaling

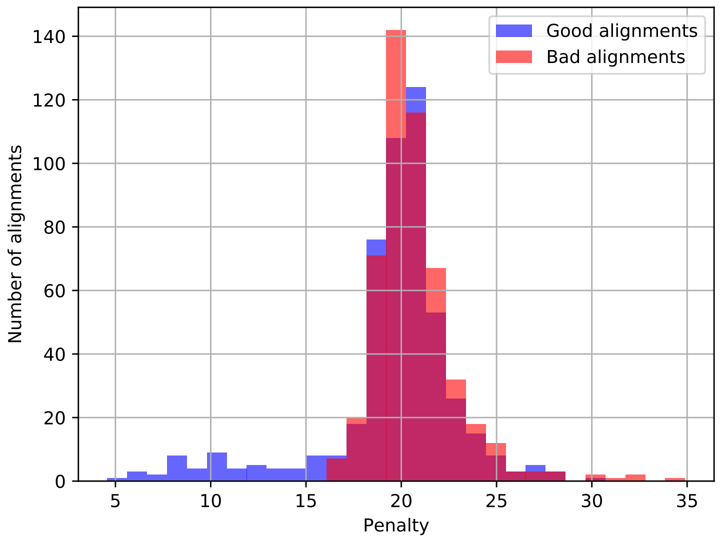

Figure 4.4: Square distance, Min-Max prescaling

- *Mean-Max prescaling* Mean-Max rescaling provided similar results to Min-Max, with the percentage being lower, only 7.2%. This is caused by outlier alignments from the `bad` dataset having low penalties.

- *Mean-SD prescaling*

  Mean-SD rescaling provided the best results yet, with 11.8% of the `good` data set having lower penalties than any alignment of the `bad` data set.

The square distance function assigned higher penalties to `good` alignments when compared to euclidean. We deemed it a unviable strategy and omitted it from future testing.

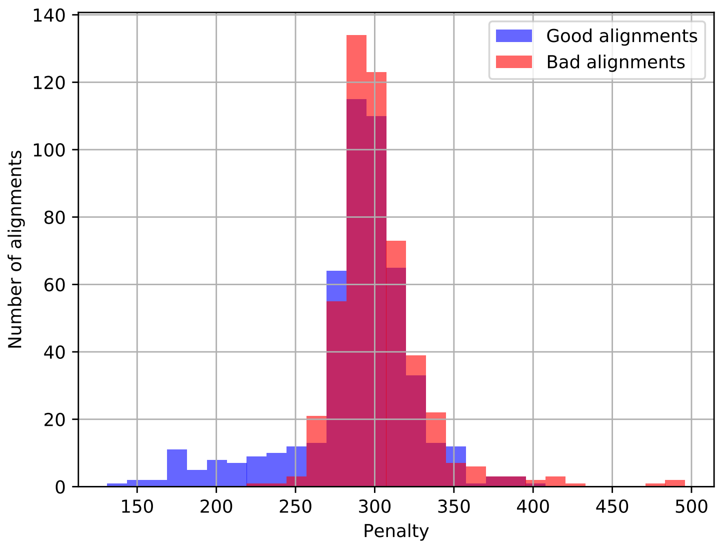

Figure 4.5: Euclidean distance, Mean-Max prescaling

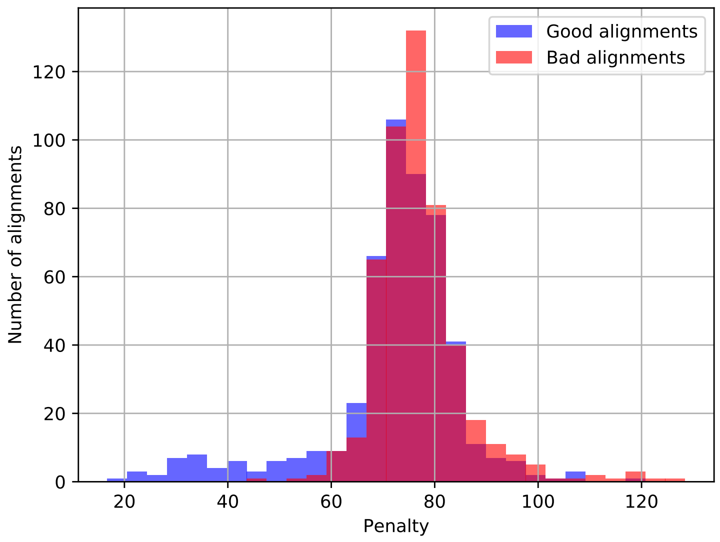

Figure 4.6: Square distance, Mean-Max prescaling

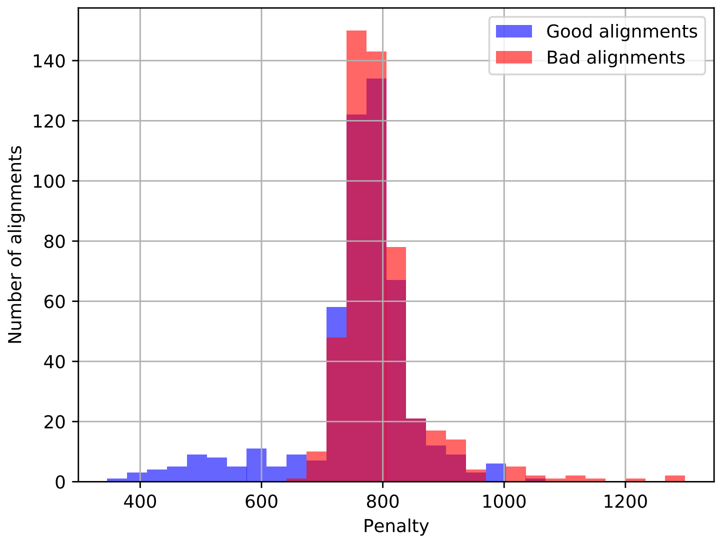

Figure 4.7: Euclidean, Mean-SD rescaling

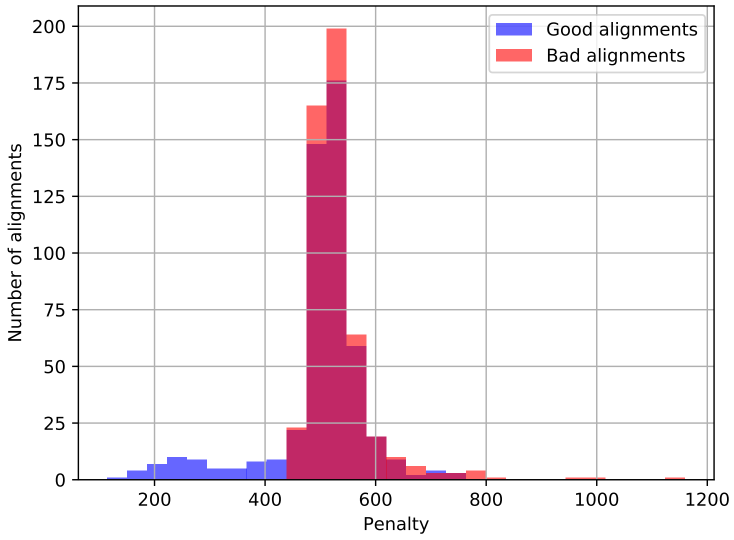

Figure 4.8: Square distance, Mean-SD rescaling

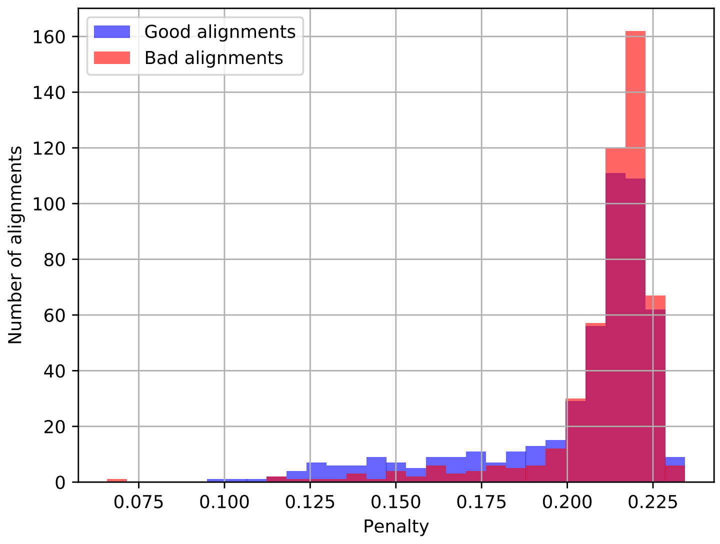

Figure 4.9: Mean-SD scaling, euclidean distance, sum postprocessing

4.5.1 Post-processing

Next, we tested four scaling variables: length of the smaller squiggle, larger squiggle, sum of the squiggle lengths, and the length of the minimal warping path found. For testing the postprocessing options, we chose preprocessing with the best previous results (Mean-SD).

- *Sum of the squiggle lengths* This post-processing method produced unexpected results. While the `good` alignments have slightly lower penalties on average, they are again indistinguishible in the low range of penalties.

- *Maximum of the squiggle length* Scaling by maximum produced similar results to the sum, with even less difference between the `good` and `bad` data sets.

- *Minimum of squiggle length* Minimum squiggle length produced much more reasonable results. 10.2% of the `good` penalties were smaller than those of the `bad` dataset, with the number possibly higher when we allow a small amount of false positives.

- *Warping path length* Similar results to the previous test, with 9% of the penalties distinguishable without false positives.

The *sum* and *max* tests provided surprising results. A relatively large portion of the `bad` data set recieved small penalties, mixing with the lower end of the `good` data set.

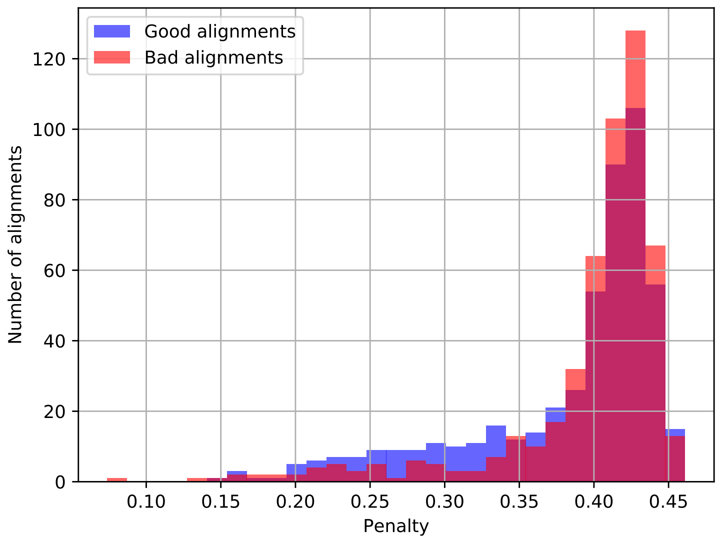

Figure 4.10: Mean-SD scaling, euclidean distance, max postprocessing

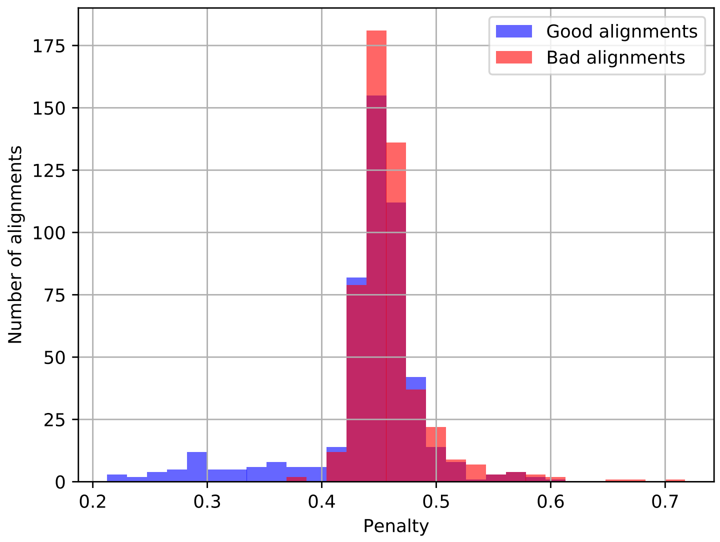

Figure 4.11: Mean-SD preprocessing, Euclidean distance, Min postprocessing.

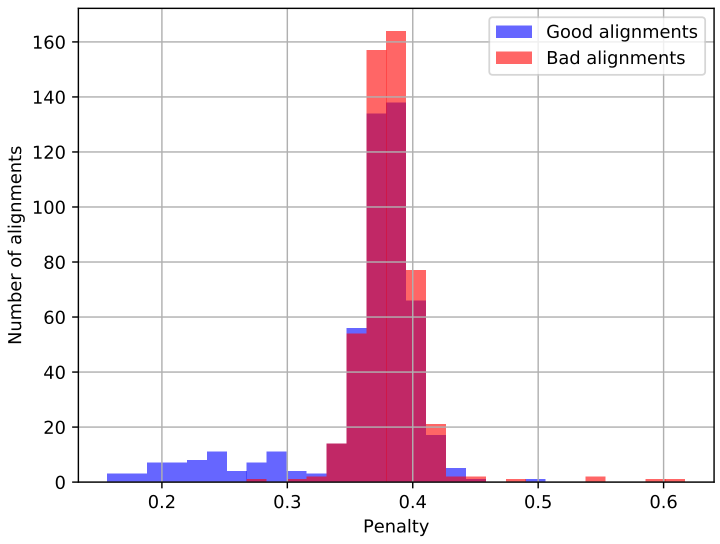

Figure 4.12: Mean-SD scaling, euclidean distance, path postprocessing

During all of the previous tests, a small portion of `good` alignments had a lower penalty than `bad` data set. However, the majority was indistinguishible, as if the `good` dataset had reads that are not aligned. This was likely caused by alignemnt drift in the preparation of the data sets. The data sets were created by fixing the leftmost position of each read and slicing them at each 1000th position (offset such that all reads are split at the same position). However, a read starting at position 0 of reference sequence may not have its 1000th base at position 1000 of reference, as indels shift the sequences relative to each other. The sequences created by nanopore sequencing tend to have a large amount of indels, so by the end of a read a long read the 1000 base long chunk may be completely ofset. This explains the large portion of the `good` data set whose penalty is the same as the `bad` data set.

To mitigate this problem, we opted to use global-local alignment. For each pair of squiggles to be aligned, we let one be a reference, and the other one sample. We trim a third from each end of the sample and align it to the reference squiggle. If the alignments drifted away by a small amount, we should see an improvement in the resulting penalty. Along with solving the drifting, this also simulates a real world application, where we do not know the boundaries of our squiggles.

After running our tests, we saw slight improvement in the penalty of `good` reads, consistent with the drift of the sequences.

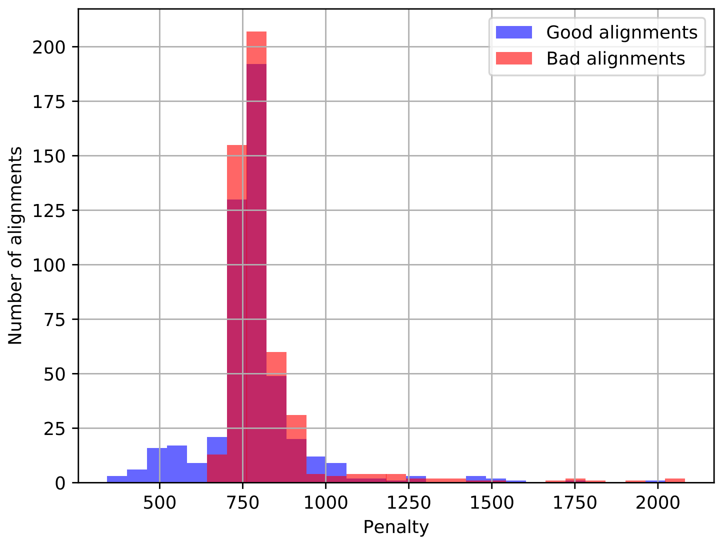

Figure 4.13: Global-Local alignment without postprocessing.

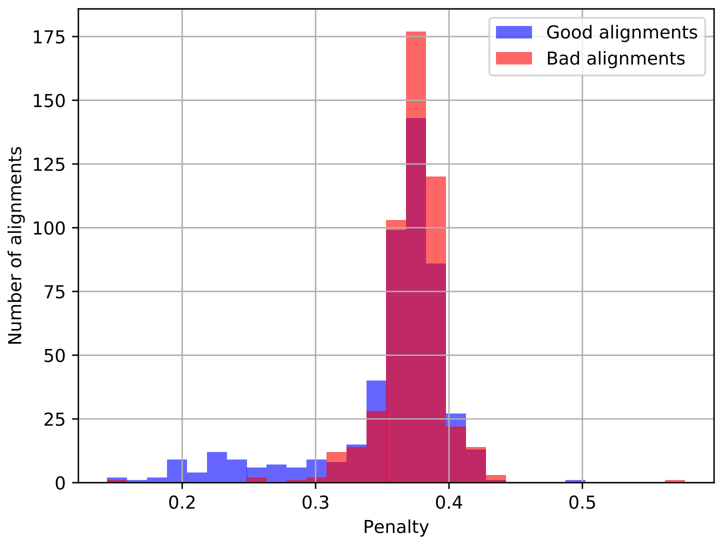

Figure 4.14: Global-Local alignment with warping path lenght post-scaling. When compared with globally aligned positions, there is a small decrease in both of the data sets.

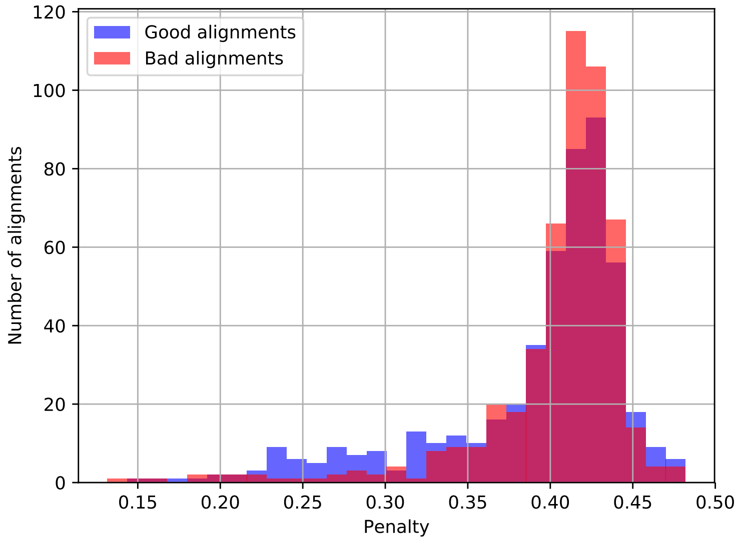

Figure 4.15: Global-Local alignment with Max postprocessing. Global-local alignment had similar effect on this post-processing as on the path post-processing. Scaling output by the size of larger of the two sequences produces worse results than the path post-process.

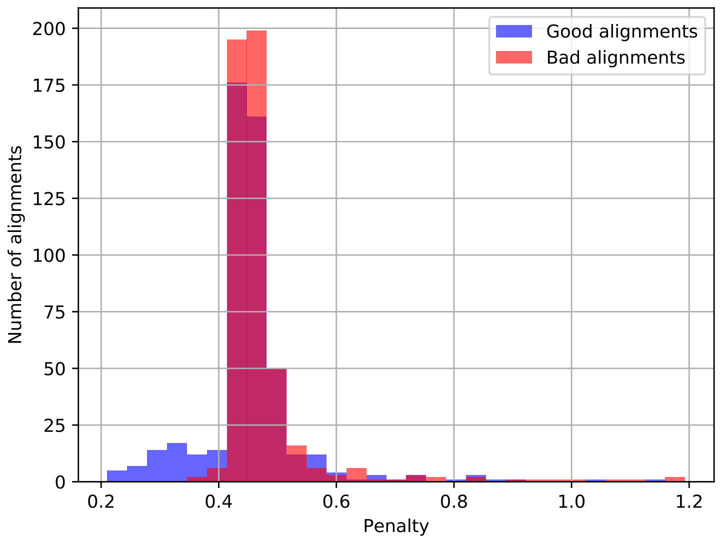

Figure 4.16: Global-Local alignment with Min postprocessing. We can observe similar results to the Path postprocessing.

Conclusion

Our goal in this thesis was to explore different aproaches of aligning the current measurements without base calling and assess their viability. The most common aproach is to base call the current sequence and align the resulting DNA sequence to a reference genome. This method requires significant pocessing power and introduces translational errors into the resulting sequence. Common algorithms used to align DNA sequences are Needleman-Wunsch and Smiths-Waterman. Needleman-Wunsch is used for global alignments, where both sequences are aligned start-to-start, end-to-end. Smiths-Waterman is used for local alignment, where we require two subsequences with the best global alignment, one from each original sequence.

The expected benefit of aligning sequences over the more traditional process of base calling and a subsequent alignment is faster processing and lower error rate. For this purose we used dynamic time warping, a common aproach for comparing sequences used in music industry and speech recognition.

Using DTW to align squiggles as-is does not produce useful alignments. We tested several methods of preprocessing the squiggles, different scoring schemes for the DTW itself, and postprocessing of the results (Chapter 4). The test was composed of running DTW on two datasets. One dataset contained pairs of squiggles, whose sequences were prealigned against each other. The conrol dataset was composed of random pairs of squiggles. Each squiggle represented a DNA sequence of 1000 base pairs. we used DTW to globally align each pair, the output of each test was minimal alignment penalty.

The evaluation of the tested variants was composed on relative differences between the results of the individual tests. While an absolute observation would have been preferable, the fact that the squiggle pairs in the dataset were drifing prevented this. We found that scaling the input squiggle based on their mean value and standard deviation provided the best results, as it managed to distinguish 11.6% of cases without false positives. While both the euclidean and the square distance functions provided similar results, the . Scaling the results from DTW by length of minimal warping path provided slight improvements over no scaling.

In addition to testing DTW on global alignments, we used a modified version to produce global-local alignments, where an entire squiggle was aligned to a subsequence of a reference squiggle. This is a more realistic scenario, where we cannot rely on the ends of squiggles to match.

Aligning squiggles is a useful method of determining the position of the read relative to the reference sequence. In future work, we might want to explore the possibility to use alignments of short squiggles as seeds for DNA sequence alignment algorithms based on the seed-and-expand method.

Bibliography

[1] Waleed H Abdulla, David Chow, and Gary Sin. Cross-words reference template for dtw-based speech recognition systems. In *TENCON 2003. Conference on Convergent Technologies for the Asia-Pacific Region*, volume 4, pages 1576–1579. IEEE, 2003.

[2] Vladimír Boža, Broňa Brejová, and Tomáš Vinař. Deepnano: deep recurrent neural networks for base calling in minion nanopore reads. *arXiv preprint arXiv:1603.09195*, 2016.

[3] Miten Jain, Sergey Koren, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, Sunir Malla, Hannah Marriott, Karen H Miga, Tom Nieto, Justin O'Grady, Hugh E Olsen, Brent S Pedersen, Arang Rhie, Hollian Richardson, Aaron Quinlan, Terrance P Snutch, Louise Tee, Benedict Paten, Adam M. Phillippy, Jared T Simpson, Nicholas James Loman, and Matthew Loose. Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv*, 2017.

[4] Zsolt Miklos Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1266–1276, 2000.

[5] T. Laver, J. Harrison, P.A. O'Neill, K. Moore, A. Farbos, K. Paszkiewicz, and D.J. Studholme. Assessing the performance of the oxford nanopore technologies minion. *Biomolecular Detection and Quantification*, 3:1 – 8, 2015.

[6] Matthew Loose, Sunir Malla, and Michael Stout. Real-time selective sequencing using nanopore technology. *Nature methods*, 13(9):751–754, 2016.

[7] Meinard Müller. *Dynamic time warping.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[8] Pierre Rouanet. Dtw python module, 2014.

[9] Gregory F. Schneider and Cees Dekker. DNA sequencing with nanopores. *Nat Biotech*, 30(4):326–328, Apr 2012. News and Views.

[10] Jared T. Aligning nanopore events to a reference, 2015.

[11] Arnoud H.M. Van Vliet. Next generation sequencing of microbial transcriptomes: challenges and opportunities. *FEMS Microbiology Letters*, 302(1):1, 2010.

Appendix A

This thesis includes an attached CD, containing the source code and testing data mentioned in the text.